UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

SAMUEL CESCONETTO

UTILIZAÇÃO DE ESPECIFICAÇÕES DO W3C NO DESENVOLVIMENTO DE INTERFACE *WEB* PARA GERENCIAMENTO DAS FERRAMENTAS *IPTABLES* E *SQUID* NO *LINUX*

CRICIÚMA, JULHO DE 2007

UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

SAMUEL CESCONETTO

UTILIZAÇÃO DE ESPECIFICAÇÕES DO W3C NO DESENVOLVIMENTO DE INTERFACE *WEB* PARA GERENCIAMENTO DAS FERRAMENTAS *IPTABLES* E *SQUID* NO *LINUX*

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: Prof. M.Sc. Paulo João Martins

CRICIÚMA, JULHO DE 2007

Aos meus pais, Estevão e Edna, que muito esforço e dedicação empregam para realização de minha educação.

“Se você é capaz de tremer de indignação a cada vez que se comete uma injustiça no mundo, então somos companheiros”

(Ernesto Che Guevara)

RESUMO

Com o crescimento da Internet as organizações começaram a se preocupar e investir, em questões de segurança, para não perder suas informações confidenciais. Isto, devido ao surgimento de novas vulnerabilidades, tornando o perigo de ataques eminente. Desta forma, nas redes de computadores para se estabelecer algumas regras que forneçam certo nível de segurança são utilizados filtros denominados de *Firewalls* que controlam os perímetros destas, bem como as operações realizadas internamente. Além dos *Firewalls*, a utilização de *proxies* favorece a segurança, com o mascaramento das máquinas internas por uma única interface. No ambiente Linux, as ferramentas para gerencia de *Firewalls* e *proxies* de grande utilização, são o *IPtables* e o *Squid* respectivamente. Porém, alguns dos pontos fracos destas ferramentas são as difíceis configurações, muitas vezes realizadas por meio de arquivos textos e a falta de disponibilidade de administração remota, tendo de ser realizada muitas vezes por formas de conexão pouco convenientes. Desta forma, foi desenvolvida uma interface *Web* para gerenciamento destas ferramentas, utilizando especificações do consórcio de padrões W3C. Estas especificações que proporcionam melhorias quanto à organização, estruturação e acessibilidade de aplicações na rede mundial. Assim, o objetivo de tornar as ferramentas disponíveis de forma remota, foi atingido, além de que com a utilização das especificações auxiliou o funcionamento em outros dispositivos e agentes de usuários.

Palavras-chave: Segurança em Redes, Redes de Computadores, *Firewalls*, Especificações W3C, Acessibilidade *Web*.

ABSTRACT

With the growth of Internet, organizations started to care and invest in security, so that they won't lose confidential information. Due to the birth of new exposures, making the danger of atacks higher. Because of this, on the web to establish some rules that provide a certain level of security are used filters, named *Firewalls* that controls these perimeters, as well as on the operations done internally. Besides the *Firewalls*, the use of *proxies* helps security, with the disguise of internal machines by one only interface. In the environment Linux, the management tools for the *Firewalls* and *proxies* widely used are the *IPtables* and the *Squid* respectively. Although, some of the down sides of these tools are the difficult settings, oftenly carried through text files, and the lack of remote administration available, having to be oftenly done by not very convenient connections. This way, a *Web* interface was developed to manage these tools, using specifications from the standard join W3C. These specifications provide improvement to the organization, structuring and accessibility to the applications in the World Wide Web. This way, the goal of turning the available tools more remote, was achieved, besides, the use of specifications helped the working of other devices and user agents.

Key-words: Web Security, Computer Web, *Firewalls*, Specifications W3C, Web Accessibility.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ACL	*Access Control List*
ARPA	*Advanced Research Projects Agency*
CSS	*Cascading Style Sheets*
DHTML	*Dynamic HiperText Markup Language*
DOM	*Document Object Model*
DTD	*Document Type Definition*
ECMA	*European Computer Manufacturers Association*
HTML	*HiperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IE3	*Internet Explorer 3*
ISO	*International Organization for Standardization*
LAN	*Local Area Network*
MAN	*Metropolitan Area Network*
MPs	Módulos Processadores
NSF	*National Science Foundation*
SGML	*Standard Generalized Markup Language*
URIs	*Uniform Resource Identifiers*
W3	*World Wide Web*
W3C	*World Wide Web Consortium*
WAI	*Web Accessibility Initiative*
WAN	*Wide Area Network*
XHTML	*eXtensible HyperText Markup Language*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A rápida expansão da Internet abriu inúmeras possibilidades para disponibilização de serviços computacionais na rede. A *Web* é, atualmente, o principal veículo para a prestação desses serviços, permitindo atingir um número cada vez maior e mais diversificado de usuários.

Porém, a ampla utilização desses meios para fornecimento de tais serviços, passou a demandar maiores preocupações quanto à segurança. As organizações passaram a controlar a utilização destes recursos e serviços de forma a não comprometer seus dados e informações.

Assim, para realizar o controle da utilização da Internet bem como de alguns serviços nela oferecidos, são utilizados alguns *softwares* denominados de *Firewalls* e *proxies*, os quais fornecem recursos para estabelecimento de um conjunto de regras, denominado de políticas de segurança.

Desta forma, para facilitar a utilização e eliminar a possibilidade de erros de sintaxe no desenvolvimento de regras a estes *softwares* de controle, foi desenvolvida uma interface *Web* para gerenciamento das ferramentas *IPtables* e *Squid*, estas tradicionais ao emprego de *Firewalls* e *proxies*, utilizadas para fornecimento de um certo nível de segurança.

Para este desenvolvimento, foram utilizadas corretamente as especificações aplicáveis à interface, do consórcio de padrões *World Wide Web Consortium* (W3C), das linguagens empregadas, bem como de acessibilidade. Tais como a especificação da linguagem estrutural XHTML, da linguagem de apresentação CSS e do documento *Web Content Accessibility Guidelines 1.0*.

Os conceitos destas especificações e as contribuições para a interface, foram fundamentadas no desenvolvimento deste trabalho e apresentadas também na forma de resultados com a implementação.

1.1 OBJETIVO GERAL

Desenvolver uma interface gráfica *Web*, para gerenciamento das ferramentas *IPtables* e *Squid* utilizando especificações do W3C.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa consistem em:

a) identificar e aplicar as especificações *Web* do consórcio de padrões W3C;
b) abordar as vantagens da utilização destas especificações;
c) identificar e aplicar as regras utilizadas nas ferramentas *IPtables* e *Squid* no ambiente *Linux*;
d) possibilitar a administração destas ferramentas remotamente, por meio de computadores *desktop* em rede e outros dispositivos.

1.3 JUSTIFICATIVA

A qualidade no desenvolvimento de códigos para aplicações à Internet, é uma questão atualmente discutida pela comunidade científica, sendo de interesse mútuo entre usuários e desenvolvedores. Teve sua importância ascendida devido ao surgimento

de novos equipamentos que realizam acesso à rede, como os dispositivos móveis, console para jogos e outros.

Desta forma, a determinação de especificações para desenvolvimento foi idealizada de forma a garantir que a execução dos aplicativos não possuam limitações quanto a *hardware* ou *software* utilizado para o acesso.

Segundo Bueno (2000) a acessibilidade entende-se pela facilidade da aproximação, do trato ou obtenção. Sendo assim, podemos afirmar que utilizar especificações do consórcio W3C está intimamente ligado a acessibilidade por proporcionar fácil acesso as informações por meio de diversos dispositivos e sistemas.

De acordo com Macedo (2004) o uso das especificações W3C proporciona diversos benefícios como por exemplo, a economia de recursos com arquivos de código menores e organizados, proporcionado pela divisão do conteúdo da apresentação com o auxílio das folhas de estilos e assim levando a transferência mais rápida das informações.

Aliando estes métodos a soluções implantadas para auxiliar na administração de segurança foram obtidos resultados positivos. Estas soluções, como mecanismos que agem interativamente com as ferramentas de segurança e administradores de redes. Apresentando uma solução prática e viável, por meio do uso de interfaces gráficas, onde o administrador terá uma melhor praticidade e eficácia na gerência do *Firewall.*

Baseado nestes argumentos justifica-se o desenvolvimento de uma interface para o gerenciamento dos *Firewalls IPtables* e *Squid* com interface *Web* padronizada, oferecendo suporte a diversos meios de interconexão com a rede, de forma a facilitar a administração.

1.4 TRABALHOS CORRELACIONADOS

O trabalho de Tatiana B. de Oliveira, Márcia Lilian Sandri e Fabiano Fagundes (2005) cujo título é *Proposta de Adaptação do Webvox a Internet com XML*, apresenta uma proposta de leitura de páginas em XML para deficientes visuais, comparando com a leitura realizada pelo *Webvox* em páginas HTML. Nele são apresentadas as dificuldades encontradas na interpretação das *tags* da linguagem XML, e como a utilização de ontologias poderá auxiliar na problemática apresentada.

Ari Palu Junior, apresentou em 2005 um trabalho intitulado de *Interface administrativa para Firewall de Internet em ambiente Linux*. Este que trouxe questões de segurança, técnicas de *Firewalls*, incluindo a ferramenta utilizada Netfilter/IPtables.

Um outro trabalho correlato, sendo este diretamente relacionado com a finalidade da interface desenvolvida, foi apresentado em 2005 por Régis Maciel Borscheid. Sob o título *Protótipo de aplicação Web para gerenciamento de Firewall Linux*, ele apresenta o desenvolvimento de um protótipo de *software* para gerenciamento de *Firewall Linux* via *Web* detalhando sobre formas de implementação e assuntos relacionados à segurança.

1.5 ESTRUTURA DO TRABALHO

Esta monografia está estruturada de forma a apresentar, no capítulo 1, a introdução e a contextualização do tema bem como a apresentação de trabalhos correlacionados.

O capítulo 2 apresenta um histórico sobre as redes de computadores, surgimento da Internet e descreve sobre as ferramentas nativas do sistema operacional *Linux*, como *IPtables* e *Squid*, utilizadas com propósito de manter uma certo nível de segurança.

No capítulo 3, é contextualizado sobre a *World Wide Web* e seus componentes básicos, descreve sobre o funcionamento de navegadores ou *browsers*, apresenta sobre desenvolvimento *Web*, no qual trás um resumo das especificações do W3C para desenvolvimento e as tecnologias tratadas na pesquisa.

O capítulo 4 apresenta as tecnologias utilizadas no lado do servidor como a linguagem de programação PHP, e a conceituação do servidor *Apache*®.

O capítulo 5 apresenta a interface *Web* desenvolvida para gerenciamento das ferramentas e especificações do W3C tratadas na fundamentação teórica, trazendo também a metodologia utilizada para concepção desta interface.

Finalmente, no capítulo 6, são apresentados os pontos fortes e fracos da interface desenvolvida. Em seguida é realizada a conclusão e a sugestão de trabalhos futuros.

2 REDES DE COMPUTADORES

Uma rede é formada por um conjunto de Módulos Processadores (MPs) para trocar informações e compartilhar recursos interligados por um sistema de comunicação. Estes módulos podem ser qualquer dispositivo capaz de comunicar-se com troca de mensagens por um meio de comunicação, como computadores, copiadoras ou terminais de vídeo texto (SOARES; LEMOS; COLCHER, 1995).

Conforme Hayden (1999) as redes possuem diversos objetivos e destes pode-se abstrair o mais importante, a interligação de itens de mesma natureza, por meio de conjuntos de regras que forneçam um serviço confiável.

As redes de computadores podem ser utilizadas por pessoas ou organizações, com diversas finalidades. Elas podem oferecer compartilhamento de recursos, cujo objetivo é disponibilizar equipamentos e dados para usuários fisicamente distantes, distribuição de processamento entre as máquinas, comunicação entre pessoas, diversão interativa, utilização da *World Wide Web* ou mesmo para aumentar a confiabilidade, distribuindo arquivos duplicados em diversos computadores (*hosts)*, aumentando a disponibilidade (TANEMBAUM, 1997).

Assim, uma *Local Area Network* (LAN), é um grupo de computadores interligados por meio de uma rede em locais não muito extensos. Estas também podem se unir a outras *LANs* para utilização em um espaço distribuído em uma pequena região, como uma cidade, utilizando linhas telefônicas ou *hardwares* especiais, sendo que estas redes locais agrupadas são denominadas de *Metropolitan Area Network* (MAN) (HAYDEN, 1999).

Porém, a distribuição física em grande escala com maiores distâncias das redes locais ou metropolitanas, torna impraticável a utilização. Neste caso precisa-se

montar uma rede remota, *Wide Area Network* (WAN). As redes remotas, são LANs espalhadas geograficamente, ou MANs conectadas por meio de linhas telefônicas, as vezes de alta velocidade (HAYDEN, 1999).

Assim, com a distribuição das redes geograficamente propiciou-se o surgimento da Internet. A próxima seção irá apresentar um pequeno histórico sobre este recurso tecnológico.

2.1 INTERNET

Conforme Isaguirre (2001), no ano de 1969, os americanos temendo um ataque soviético, durante a guerra fria, que destruísse informações e banco de dados fundamentais, necessitaram da transição das informações rapidamente entre computadores geograficamente distribuídos. Em um projeto militar o Departamento de Defesa, junto a *Advanced Research Projects Agency* (ARPA), uniram pequenas LANs posicionadas em locais estratégicos do país por meio das redes de telecomunicação. Ao final da guerra, os militares repassaram a tecnologia para universidades que utilizavam apenas para transição de pesquisas e trabalhos acadêmicos. Posteriormente a *National Science Foundation* (NSF), uniu a rede às agências governamentais e institutos de pesquisa, proporcionando com isto o início da Internet.

Segundo Comer (2001) o intuito da ligação das redes era o aumento do poder computacional para processamento distribuído e compartilhamento dos recursos mais modernos adquiridos pela ARPA. A pesquisa sobre a ligação em rede da organização acabou sendo revolucionária, proporcionando ao fechamento de contratos e focalizando as pesquisas nesta área, levando a um sistema chamado ARPANET. A organização financiou pesquisas sobre tecnologias alternativas, aplicações de rede e

uma tecnologia chamada *ligação inter-redes.* Nos anos 70, este projeto se tornou foco de pesquisa e a Internet acabava de surgir, os estudos continuaram nos anos 80, até a popularização e o sucesso na década de 90.

Assim, a Internet atual é uma grande rede de computadores, composta por redes LANs interconectadas que permitem aos usuários situados em qualquer ponto do mundo, trocar informações de qualquer natureza. Ela pode ser composta por computadores dos bancos, universidades, hospitais, bibliotecas, empresas e residências (DEMETRIO, 2001).

Devido ao agrupamento destas pequenas redes, cada qual com seu propósito, a necessidade do emprego de algumas técnicas de segurança tornou-se mais relevante. Na próxima seção serão apresentados alguns conceitos sobre a segurança nas redes de computadores.

2.2 SEGURANÇA DE REDES DE COMPUTADORES

Com a utilização das redes de computadores em diversas organizações e a extensão continua dessas, o gerenciamento de redes tornou-se indispensável. Esta tarefa envolve supervisão (monitoramento) para controle de segurança e dos recursos nesta distribuídos. Em essência o gerenciamento visa garantir certa qualidade dos serviços e de proteção aos usuários (PEREIRA, 2001).

Os problemas com segurança já existiam antes da popularização da Internet, mas juntamente com o crescimento desta, houve o aumento da disponibilização de serviços computacionais e consequentemente vulnerabilidades. Assim, as organizações para manter seus dados seguros passaram a ter que empregar algumas formas de segurança (HONÓRIO, 2003).

Um dos recursos mais utilizados atualmente por essas corporações para restringir com segurança o acesso de suas redes a Internet, são os *Firewalls*. Estes sistemas de segurança, que traduzido do inglês representam *parede de proteção ao fogo*, localiza-se entre as duas redes como uma suposta parede que filtra todo o fluxo de informações entre estas (ZWICKY, 2000).

Diferentes arquiteturas de *Firewall* de diversos fabricantes para a finalidade específica são disponibilizadas. Estas arquiteturas variam, podendo ser um *hardware* proprietário ou até mesmo um conjunto de aplicações implementadas com *softwares* gratuitos em sistemas operacionais livres, como o popular Linux. Uma das ferramentas nativas deste sistema operacional atualmente distribuída com o sistema padrão, de filtragem de pacotes (tipo de *Firewall*) é o *IPtables* (RUSSELL, 2001).

Porém, para construir um sistema de defesa, é necessário antes determinar o que se precisa proteger, contra quem e quais as formas de proteção disponíveis atualmente. Esta técnica conhecida como determinação de políticas de segurança, possibilita o início do processo de implantação de um sistema de segurança (JUNIOR, 2005).

2.2.1 Políticas de Segurança

Uma política de segurança é um conjunto de leis, regras e práticas que regulamentam como uma organização gerencia, protege e distribui suas informações e recursos. Ela define o que é ou não permitido em termos de segurança, durante a operação de um dado sistema. Portanto um software é considerado seguro em relação a uma política, caso garanta o comprimento das leis, regras e práticas definidas nesta (VILELA, 2001).

A definição de políticas de segurança e sua implantação estão ligadas ao sucesso da organização neste seguimento. Porém esta etapa é um tanto complexa, uma vez que políticas de segurança não tratam apenas da utilização de *Firewalls* nem da configuração de um serviço ou outro, mas de uma série de normas que todas as pessoas da organização devem adotar. Um exemplo dessas normas é o controle do acesso físico aos computadores (MELO; TRIGO, 2004).

Desta forma, uma política de segurança consiste em um conjunto de regras que incluem diversas tarefas, como *backups*, controle de acesso físico, criptografias, controle do uso das redes por meio de *Firewalls* entre outras. Da parte das políticas que condizem a redes de computadores, pode-se observar a existência das de alto e baixo nível. Onde a de alto nível define os serviços que serão capacitados ou explicitamente rejeitados na rede, bem como a forma de estes serem utilizados, enquanto a política de nível mais baixo define como o *Firewall* de fato restringirá o acesso e filtrará os serviços definidos pela política de nível mais alto (GONÇALVEZ, 2000).

Conforme Soares (2003) quando se pensa em segurança, é necessário ter a seguinte premissa em mente: Quanto maior o nível de segurança, menor o nível de privilégios.

Aplicando este conceito para a segurança em redes de computadores, obtém se a conclusão que para uma rede ser o mais segura possível seu usuário irá possuir o menor número de recursos possíveis. Deste modo, ao estruturar políticas de segurança para um determinado ambiente computacional, é de suma importância estabelecer as reais necessidades de recursos que os usuários necessitam, para que conseqüentemente se ofereça a máxima capacidade em segurança (JUNIOR, 2005).

Assim, para se obter alguns pontos estabelecidos nas políticas de segurança é necessário a utilização de *Firewalls*, estes que serão tratados neste trabalho, focando o desenvolvimento de uma interface *Web* para gerenciamento destes.

2.3 *FIREWALLS*

Os países que não protegem suas fronteiras, não podem garantir a segurança de seus cidadãos, nem podem evitar a pirataria e o roubo. Da mesma forma, as redes de computadores sem controle de acesso não podem garantir a segurança e privacidade dos dados armazenados, nem podem garantir que os recursos da rede não sejam explorados por pessoas não autorizadas (STREBE; PERKINS, 2002).

Basicamente um *Firewall* separa uma rede protegida de uma desprotegida, a Internet. Ele filtra todas as conexões que chegam pela rede pública a protegida (corporativa), e vice versa, por meio de um único ponto de verificação de segurança concentrada. Este mecanismo garante o bloqueio do acesso direto a Internet a partir da rede interna bem como a operação reversa. Sendo assim, todo o acesso deve ser intermediado por este ponto de controle (GONÇALVEZ, 2000).

Assim, os *Firewalls* são conhecidos como ferramentas utilizadas para criar pontos de controle nas fronteiras das redes privadas. Eles monitoram todo o tráfego na rede, e com base nesta inspeção eles aprovam ou rejeitam cada tentativa de conexão. Normalmente estes dispositivos, se localizam nas interconexões (*gateways*) que fornecem acesso a outras redes, comumente encontrados entre redes privadas e Internet. Por esta razão os *Firewalls* são considerados proteção de fronteiras (STREBE; PERKINS, 2002).

Conforme Diehl (2001) a Figura 1 apresenta um exemplo típico de uma rede local (LAN) conectada a Internet por meio de um *Firewall*, este realizando a tarefa de filtragem dos dados de entrada e saída, favorecendo a segurança da rede interna.

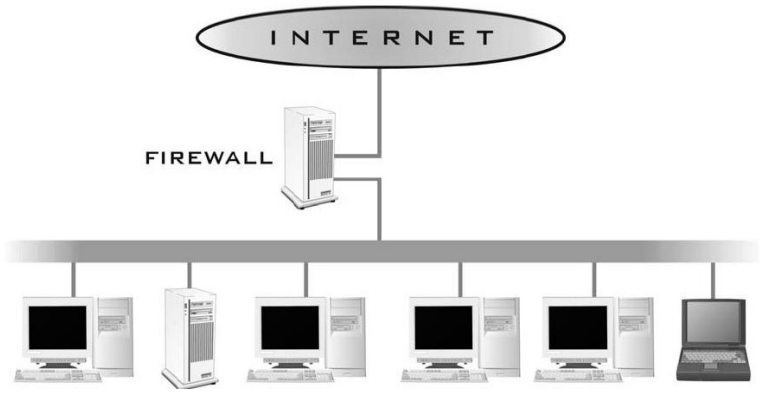

Figura 1. Rede local conectada a Internet por meio de um *Firewall*
Fonte: DIEHL, G. P. (2001)

Desta forma, um *Firewall* atuando como um ponto de indução, ou seja, sendo o único computador diretamente conectado a Internet, poderá de forma segura levar estes serviços a sua rede local, evitando assim que cada *host* seja responsável por sua segurança (NETO, 2004).

Para realizar a filtragem e controle dos dados que entram e saem estes mecanismos possuem um conjunto de regras especificando que tráfego ele permitirá ou negará. Existem vários tipos de *Firewalls* diferentes, incluindo filtros de pacotes e *proxies* (NORTHCUTT et al, 2002).

Os *Firewalls* operam usando três métodos fundamentais:

a) Filtragem de pacotes: rejeita pacotes TCP/IP de *hosts* e tentativas de conexão com serviços, não autorizados;

b) *Network Address Translation* (NAT): converte os endereços IPs dos *hosts* internos para ocultá-los de qualquer monitoração externa. Esta técnica também é conhecida como mascaramento de IP;

c) Serviços *proxy*: faz com que as conexões de aplicativos de alto nível em benefício de computadores internos quebrem completamente a conexão da camada de rede entre hosts internos e externos (STREBE; PERKINS, 2002).

Com base nesta classificação, é importante a contextualização dos métodos focados à interface desenvolvida. Com isto, será apresentado o funcionamento básico dos *Firewalls* fundamentados em filtros de pacotes e *proxy*, bem como das ferramentas utilizadas, *IPtables* e *Squid*.

2.3.1 Filtro de Pacotes

A filtragem dos pacotes é um dos principais mecanismos que, mediante regras definidas pelo administrador em um *Firewall*, permite ou não a passagem de datagramas IP em uma rede (MARCELO 2002).

Estes filtros comparam os pacotes dos protocolos de rede (como o IP) e os de transporte (como o TCP) com um conjunto de regras mantidas em um banco de dados e assim, somente encaminham pacotes que atendam a critérios especificados (STREBE; PERKINS, 2002).

Assim, para que ocorra a comunicação dos sistemas de redes, é necessário que ambas as partes utilizem a mesma linguagem, ou protocolo. Estes protocolos como o TCP/IP, utilizado como a principal linguagem de comunicação na Internet. Para que se torne ainda mais viável, as informações precisam ser divididas em partes

manipuláveis, denominadas *pacotes*. Cada parte possui um respectivo cabeçalho, contendo um segmento de informações fixadas para sua identificação (NORTHCUTT et al, 2002).

Conforme Diehl (2001) as principais informações contidas nestes cabeçalhos são:

a) endereço IP de origem;

b) endereço IP de destino;

c) protocolo (TCP, UDP ou ICMP);

d) porta TCP ou UDP de origem;

e) porta TCP ou UDP de destino;

f) tipo de mensagem ICMP;

g) tamanho do pacote.

Desta forma, a filtragem de pacotes é um dos métodos mais antigos e amplamente disponíveis, para controlar acesso às redes. A determinação se um pacote possui permissão para entrar ou sair da rede, é realizada por meio da comparação de algumas informações de identificação contidas nos cabeçalhos dos pacotes, com regras especificadas em um *Firewall* baseado nesta técnica (NORTHCUTT et al, 2002).

No sistema operacional *Linux* um software denominado *IPtables* é distribuído junto a versão padrão, para o emprego de *Firewall*. Esta ferramenta é baseada em filtros de pacotes. Uma outra ferramenta de segurança, porém baseada em serviço de *proxy*, também é disponibilizada, na interface gráfica, desenvolvida para facilitar a interação do administrador. Porém antes da apresentação destas e uma breve descrição do funcionamento, deve-se fundamentar também os serviços de *proxy*.

2.3.2 Serviços de *Proxy*

Os servidores *proxy*, foram projetados para a funcionalidade de *cache*. Porém eles acabaram revelando algumas possibilidades como ocultar todos os usuários reais de uma rede por trás de uma única máquina, filtrar URLs e impedir a passagem de conteúdo suspeito ou ilegal. Assim, a finalidade principal da maioria dos servidores *proxy* atuais é operar como *Firewall* em vez de *cache* de páginas *Web* (STREBE; PERKINS, 2002).

Um serviço de *proxy* exige dois componentes: um servidor e um cliente. Os clientes são programas normais, como por exemplo, FTP ou *Telnet*, configurados para comunicarem-se com o servidor *proxy*, em vez do servidor real na Internet. O servidor *proxy* avalia pedidos do cliente e decide o que aprovar e o que negar. Se um pedido é aprovado, este entra em contato com o servidor real em nome do cliente e transmite as respostas do servidor real ao solicitante (DIEHL, 2001).

Um *proxy* configurado como servidor e conhecido como *proxy server*, este que atua como um procurador que aceita as chamadas que chegam e verifica se é uma operação válida. Caso a chamada seja uma operação permitida, o servidor envia adiante a solicitação ao servidor real do recurso desejado (OLIVEIRA, 2001).

As vantagens de utilização de um *proxy*, são muitas, sendo uma delas a própria segurança, mas pode-se classificar assim:

a) controle de conteúdo: o administrador pode realizar o controle de conteúdo, permitindo ou não o acesso a certas informações;
b) controle de acesso: pode-se também controlar quem pode acessar a Internet por meio de autenticação;

c) *cache* de sites: um *proxy* pode armazenar as páginas visitadas para agilizar o tempo de resposta as solicitações de usuários (MELO; TRIGO, 2004).

Como os serviços de *proxy* são procuradores especializados que tomam as solicitações de usuários de serviços da Internet como FTP, *Telnet* e HTTP e os encaminham aos serviços reais, eles fornecem conexões substitutas e atuam como *gateways* para os serviços. Por essa razão, os *proxies* são também reconhecidos como *gateways* do nível de aplicativos (DIEHL, 2001).

Além de apenas encaminhar os pedidos dos usuários para os serviços reais da Internet, os servidores *proxy* podem controlar o que os usuários fazem, de modo a decidir sobre os pedidos que processa, concedendo ou não o acesso a determinados serviços. Por exemplo, o *proxy* HTTP pode impedir o acesso a determinados sites e essas configurações podem ser ou não aplicadas a todos os *hosts* da rede.

Após esta breve discussão sobre serviços de *proxy*, pode-se apresentar as ferramentas nativas do sistema operacional *Linux* empregadas para gerência de *Firewalls* baseados nos métodos apresentados.

2.4 IPTABLES

Um dos softwares utilizados com a função de *Firewall*, no sistema operacional *Linux* é o *IPtables*, substituto do antigo *IPchains*. Este que possibilita especificar portas/endereços de origem/destino bem como protocolos TCP/UDP/ICMP, interfaces de origem/destino e outras opções para criação de regras de filtragem (MELO; TRIGO, 2004).

Segundo Neto (2004), o *IPtables* compõe a quarta geração de sistemas *Firewalls* no *Linux*, que foi incorporada a versão 2.4 do *Kernel*. Ele é uma versão mais completa e tão estável quanto seus antecessores *Ipfwandm* e *Ipchains*, implementados nos núcleos 2.0 e 2.2 respectivamente. O *IPtables* é amplamente utilizado devido às funções de *Firewall* estarem agregadas a própria arquitetura do *Kernel*. O sistema operacional utiliza um recurso independente em termos de núcleo do sistema operacional para controlar e monitorar todo o tipo de fluxo de dados dentro de sua estrutura operacional.

A função do *Kernel* é trabalhar ao lado de processos e tarefas, por esse motivo foi agregado um módulo ao mesmo chamado de *Netfilter* para controlar seu próprio fluxo interno. Desenvolvido por Marc Boucher, James Morris, Harald Welte e Rusty Russel, este é um conjunto de situações unidas inicialmente ao *Kernel* do *Linux* e dividido em tabelas. Sob uma ótica mais prática podemos ver o *Netfilter* como um grande banco de dados que possuem em sua estrutura três tabelas padrões: *Filter, Nat* e *Mangle* (NETO, 2004).

Conforme Borscheid (2005) as funções de *Firewall* são agregadas ao *Kernel* pelo módulo *Netfilter*. As tabelas anteriormente mencionadas possibilitam controlar todas as situações (*chains*). Desta forma, para melhor controle do *Netfilter* conforme as necessidades de cada *host*, rede ou sub-rede foram desenvolvidas ferramentas de *Front-End*[1]. Essas ferramentas permitem o controle das *chains* contidas nas tabelas agregando regras de tráfego. Historicamente o Linux disponibilizou uma nova ferramenta de manipulação nativa (*Front-End*) a cada nova versão oficial de *kernel*:

a) *kernel* 2.0 – *IPfwamd*;

b) *kernel* 2.2 – *IPchains*;

---

[1] Interface disponibilizada aos administradores para gerencia do módulo de *kernel*.

c) *kernel* 2.4/2.6 – *IPtables*.

O *IPtables* é uma ferramenta de *Front-End*, desenvolvida por Rust Russel, membro do projeto de desenvolvimento do *Netfilter*, em colaboração com Michel Neuling e foi incorporada a versão 2.4 do *kernel* em julho de 1999 (BORSCHEID, 2005).

A tabela padrão do *IPtables* chama-se *filtro*, onde cadeias de regras nesta, são utilizadas para filtragem de pacotes do tráfego na rede. A tabela filtro, contém três cadeias-padrão, onde cada pacote manipulado pelo *kernel* passa através de uma destas. Estas cadeias são reconhecidas como *chains*, sendo que as regras contidas na *chain FORWARD* são aplicadas a todos os pacotes que chegam a uma interface de rede e precisam ser encaminhados a outra. Já as regras da *chain INPUT* e *OUTPUT* são aplicadas ao tráfego destinado a, ou originando-se do *host* local respectivamente (NEMETH et al, 2004).

Segundo Marcelo (2002) as regras são comandos passados ao *IPtables*, que permitem que ele realize um determinado evento. Essas são armazenadas nas *chains* e processadas na ordem de sua inserção. Uma *chain* é uma área onde as regras definidas pelo administrador são armazenadas para sua operação. As principais *chains* que fazem parte do *IPtables* são:

a) *INPUT* : define os pacotes de entrada na rede;

b) *OUTPUT*: define os pacotes de saída da rede;

c) *FORWARD*: define os pacotes a serem encaminhados, usados normalmente para mascaramento.

O *IPtables* possui três tipos de tabelas aonde as *chains* são armazenadas, são elas:

a) *FILTER*: filtragem padrão contendo as *chains INPUT*, *OUTPUT* e *FORWARD*;

b) NAT: usada para tradução de endereços, possuindo outras *chains* como, *PREROUTING*, *OUTPUT* e *POSTROUTING*;

c) *MANGLE*: utilizada para alterações especiais, como modificar algum tipo de serviço (JUNIOR, 2005).

Apesar do *IPtables* oferecer suporte ao gerenciamento de *Firewall*s baseados em NAT e outras funcionalidades, na interface desenvolvida, foi tratado somente a tabela padrão (*FILTER*). Desta forma, serão apresentados os principais comandos para manipulação desta ferramenta, utilizados para manipulação desta tabela.

A sintaxe básica do *IPtables* é definida por: *IPtables [-t <tabela>] [comando] [ação] [alvo]* (BORSCHEID, 2005).

Conforme Russell (2001) para que as regras sejam montadas pelo *IPtables e* interpretadas pelo *Netfilter*, existem alguns comandos que são empregados. Além disso, esta ferramenta oferece três políticas:

a) *accept*: aceita o pacote recebido;

b) *drop*: nega pacote e não o retorna de volta;

c) *reject*: nega o pacote e retorna um aviso de erro ao emissor.

Segundo Junior (2005) os principais comandos utilizados nesta ferramenta, em conjunto com as políticas apresentadas são:

a) -A: adiciona ou atualiza uma regra;

b) -I: apenas adiciona uma nova regra;

c) -D: exclui uma regra específica;

d) -P: define a regra padrão;

e) -L: lista todas as regras armazenadas;

f) -F: exclui todas as regras armazenadas;

g) -R: substitui uma regra armazenada;

h) -C: efetua uma checagem das regras básicas;

i) -N: cria uma regra com nome específico;

j) -X: exclui uma regra com nome específico.

Desta forma, para montagem de uma regra, além dos comandos, as ações devem ser especificadas. Como:

a) -p: especifica o protocolo aplicado a regra;

b) -i: especifica a interface de entrada a ser utilizada. Como um *Firewall* possui mais de uma interface, esta regra se torna importante para identificar a qual o filtro deve ser aplicado;

c) -o: especifica a interface de saída, da mesma forma que a ação -i , porém aplicável somente as regras *OUTPUT* e *FORWARD*;

d) -s: especifica a origem do pacote ao qual a regra deve ser aplicada, podendo ser um *host* ou uma rede;

e) -d: especifica o destino do pacote, semelhante à ação -s;

f) !: este sinal significa exclusão, utilizado quando se quer aplicar exceção a uma regra. Utilizado em conjunto com as opções -s, -d, -p, -i e -o;

g) -j: define o alvo (*target*) do pacote caso o mesmo se encaixe a uma regra;

h) --*sport*: utilizado para especificar a porta de origem do pacote, somente aplicados aos protocolos TCP ou UDP;

i) --*dport*: utilizado para especificar a porta de destino, semelhante a ação --*sport* (BORSCHEID, 2005).

Conforme Neto (2004) quando um pacote se enquadra em uma regra especificada ele deve ser direcionado a um alvo, este definido junto a própria regra. Alguns alvos (*targets*) aplicáveis são:

a) *ACCEPT*: corresponde a aceitar, ou seja, permite a entrada/passagem do pacote em questão;
b) *DROP*: corresponde a descartar, com este alvo o emissor não é informado sobre o que houve;
c) *REJECT*: corresponde a rejeitar, a diferença deste para o *DROP* é a questão do retorno de uma mensagem de erro ao host emissor.

Conforme Marcelo (2002) algumas regras aplicáveis ao *Netfilter/IPtables* são apresentadas para efeitos de esclarecimento da sintaxe:

a) negar todos os pacotes vindos de qualquer rede: *iptables -A INPUT -j DROP*;
b) negar todos os pacotes ICMP: *iptables -A INPUT -p icmp -j DROP*;
c) rejeitar todos os pacotes da interface eth0: *iptables -A INPUT -i eth0 -j REJECT*;
d) negar todos os pacotes vindos de qualquer endereço da rede 192.168.15.0: *iptables -A INPUT -s 192.168.15.0/24 -j DROP*.

Baseado nas possíveis dificuldades para montagem das regras, e a necessidade do domínio desses possíveis comandos, ações e alvos para interações com o *IPtables*, foi desenvolvido a interface gráfica para criação de regras e administração da ferramenta. Também foi desenvolvido, funcionalidade para facilitar interação com a ferramenta de *proxy*, o *Squid*, este que será apresentado sobre o funcionamento a seguir.

2.5 *SQUID*

O *proxy* de aplicativo *Squid Cache* é um software livre, que além de servir como *cache*, mantendo cópias dos sites acessados com maior freqüência, serve para permitir ou negar acesso a determinadas *Uniform Resource Locator* (URLs) para certas redes. Esta ferramenta utiliza por padrão a porta TCP 3128 para receber as requisições de acesso dos clientes e repassar os dados por eles solicitados.

Esta é uma ferramenta nativa do sistema operacional *Linux*, à qual é responsável pela ligação da rede externa, com a interna. O *Squid* conversa com as duas redes, como um procurador, impedindo a comunicação direta entre estas. Com a utilização desta ferramenta, se obtém certa segurança, pois é possível controlar alguns dados que transitam por meio deste serviço (MELO; TRIGO, 2004).

O *Squid* trabalha baseado em algumas *Access Control Lists* (ACL), esta que é uma lista de permissões de acesso, configuradas no arquivo de configuração da ferramenta contendo informações para controle na rede (OLIVEIRA, 2001).

As ACLs possibilitam especificar endereços de origem ou destino, domínios, horários, usuários, portas ou métodos de conexão ao *proxy*. Estas informações que servem de base para permitir ou negar acesso, permitindo assim uma grande flexibilidade na configuração dessa ferramenta (MELO; TRIGO, 2004).

O controle de acesso no *Squid* é configurado por meio do arquivo de configuração *squid.conf*, com alguns parâmetros, estes que geralmente estão associadas a uma dada Lista de Controle de Acesso (ACL), que definem o contexto de um determinado controle. Alguns parâmetros são:

a) *proxy_auth_realm*: utilizado para apresentar uma caixa solicitando usuário e senha para autenticação no *Squid*;

b) *http_access*: responsável por liberar ou não acesso HTTP a uma determinada ACL;

c) *acl*: elemento principal das ACLs, sendo responsável pela sua implementação (COSA, 2006).

O parâmetro *acl* é utilizado para definir uma dada lista de controle de acesso, sendo que sua sintaxe possui a forma: *[acl nome] [tipo da acl] [informação]*. Estas são interpretadas pelo *Squid* de cima para baixo, portanto deve-se ter o cuidado no momento de estabelecer a ordem das regras. Isto é importante, pois encontrada uma regra que venha a coincidir com determinada ACL, as demais regras não serão checadas. Existem vários tipos de ACLs, os mais importantes são:

a) *src*: utilizado para indicar o endereço IP de origem, podendo também indicar um endereço de um host;

b) *dst*: indica o endereço IP de destino, semelhante ao *src*;

c) *dstdomain*: indica um domínio de destino;

d) *url_regex*: utilizado para pesquisa na URL pela expressão regular especificada (COSA, 2006).

Conforme Melo, Trigo (2004) utilizando os conceitos apresentados, para realizar o bloqueio de um determinado *site* ou URL por meio do *Squid*, deve ser empregado uma ACL seguida de um parâmetro conforme exemplo:

acl proibir_url dstdomain algo.com.br
http_access deny proibir_url.

Alguns administradores preferem criar uma lista em arquivo a parte, contendo os domínios ou URLs proibidos, possuindo uma ACL neste formato: *acl proibir_url dstdomain ‘‘path/para/arquivo’’*. Nesse caso, *“path/para/arquivo”* é o caminho local do arquivo contendo a relação de sites proibidos. Esse recurso pode ser utilizado em praticamente todos os tipos de ACLs no *Squid* (COSA, 2006).

Esta ferramenta oferece diversas formas de controle, desde autenticação de usuários e autorização de conteúdos com base em determinados horários, porém a interface desenvolvida foi projetada de modo a facilitar a especificação de palavras e URLs proibidas, bem como exceções para possíveis bloqueios indesejados. Desta forma, apenas funcionalidades básicas da ferramenta foram contempladas. Para possíveis configurações avançadas, na própria interface, foi disponibilizado uma área de texto para edição do arquivo *squid.conf*.

Assim, o termo interface é aplicado normalmente ao que interliga dois sistemas. No processo de interação usuário-sistema, a interface é o combinado de *software* e *hardware* necessário para viabilizar e facilitar os processos de comunicação entre o usuário e uma determinada aplicação (CYBYS 1995).

Baseando-se nisso, como mencionado anteriormente foi desenvolvido uma interface para gerenciamento das ferramentas *Squid* e *IPtables*, por meio da *Web*. Esta tem como objetivo fornecer certo nível de segurança às redes de computadores, oferecendo interfaces amigáveis para criação de regras para a tabela *filter* do *IPTables* e especificação de listas de conteúdos ao *Squid*. Opções de gerenciamento das regras por meio de listagens e ações a serem tomadas para cada uma, também foram disponibilizadas.

Assim, como o estudo deste trabalho também está fundamentado nas especificações do consórcio de padrões W3C para desenvolvimento *Web*, quanto aos *Firewalls*, foram oferecidos apenas suporte as funcionalidades básicas de cada ferramenta. No próximo capítulo será apresentado sobre desenvolvimento *Web*, especificações do consórcio de padrões e alguns conceitos de acessibilidade, estes que também foram utilizados na interface.

3 DESENVOLVIMENTO *WEB* E AS ESPECIFICAÇÕES W3C

A história da *Web*, começa com a proposta de Vannevar Bush, um dos principais envolvidos na criação dos laboratórios de pesquisas militares norte-americanos. Em 1945, ele escreveu um ensaio intitulado de *As We May Think*. Neste, ele descreveu o *Memex*, uma máquina para armazenar textos e imagens e criar associações entre elas. Embora o equipamento jamais tenha sido criado, sua fundamentação teórica, lançou as bases para o desenvolvimento do hipertexto (ERCILIA, 2000).

O termo hipertexto foi cunhado por Ted Nelson em 1965, para descrever a escrita não seqüencial, que apresenta informações como um conjunto de nós interligados. O hipertexto foi independentemente proposto por Doug Engelbart o inventor do dispositivo de entrada *mouse* (KRISHNAMURTHY; REXFORD, 2001).

Desta forma, proposta por Tim Berners-Lee em 1989, bolsista do Laboratório de Pesquisa Europeu de Física de Partículas (Cern), a *Web* é o universo de informações acessíveis por meio de computadores conectados a uma rede. As interfaces gráficas permitem que usuários naveguem por páginas da *Web*, sem se preocupar com o local ou formato do conteúdo (KRISHNAMURTHY; REXFORD, 2001).

Conforme Milner (2001) a *Web* é um vasto manancial de informações espalhadas por todo o mundo, baseado em computadores interligados, denominados de servidores da *Web*. Eles contêm os *websites* ou simplesmente *sites*, cuja estrutura pode vincular de uma página simples a milhares eletronicamente.

De acordo com Demétrio (2001) a *Web* é uma grande base de dados baseada em hipertexto, podendo ser acessada de forma gráfica por meio de ligações

reconhecidas por âncoras ou *links*. Esta grande variedade de material é formada por textos, imagens, sons, animações e vídeos.

Assim, o sistema hipertexto forma uma teia de documentos que contém vários vínculos com outros e durante a utilização o usuário é levado a diversos arquivos diferentes (MILNER, 2001).

Esta navegação por escolha entre *links*, pode ser feita de forma aleatória, ou seja, não seqüencial. Segundo Comer (2001) para funcionamento, a *Web* possui três componentes principais:

a) *HyperText Transfer Protocol* (HTTP): padrão de comunicação dos componentes da *Web* onde define o formato e o significado das mensagens trocadas entre os elementos como clientes e servidores;

b) *Uniform Resource Identifiers* (URIs): mecanismos de nomeação universal para identificar recursos na *Web*. Ela é simplesmente uma cadeia de caracteres com a identificação do protocolo para comunicação, do nome do servidor e do recurso nele localizado respectivamente. O formato mais popular de um URI é um *Uniform Resource Locator* (URL) ou simplesmente um endereço como *www.algo.com.br*;

c) *HyperText Markup Language* (HTML): linguagem de desenvolvimento para *Web* conhecida como linguagem de marcação. É uma representação de padrão para documentos de hipertexto.

A próxima seção descreve-se sucintamente sobre navegadores ou *browsers*.

3.1 NAVEGADORES OU *BROWSERS*

Para utilização da *Web*, é necessário um *software* específico para esta função, conhecido como *browser* ou popularmente chamado de navegador. Estes implementam um cliente da *Web*, pois constrói e envia um pedido HTTP, recebe e analisa a resposta para assim apresentar ao usuário. A Figura 2 apresenta as várias etapas no processo envolvido em um pedido do browser, conforme um navegador típico (KRISHNAMURTHY; REXFORD, 2001).

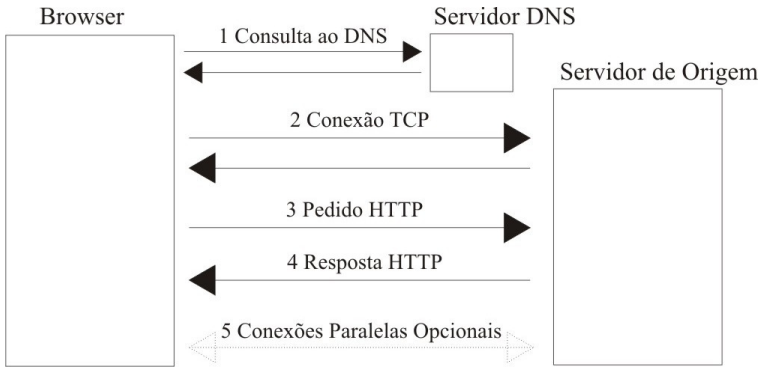

Figura 2. Funcionamento de uma requisição na *Web*
Fonte: KRISHNAMURTHY, B.; REXFORD, J. (2001)

Na primeira etapa apresentada na Figura 2 o *browser* consulta um servidor de nomes (DNS) do provedor de acesso para localizar o servidor referente a URL solicitada. Assim, na segunda etapa é realizada a conexão TCP com o servidor que contém o recurso. Depois de conectado, as etapas 3 e 4 apresentam a transferência dos arquivos com a utilização do protocolo HTTP, sendo ali transmitidos os dados do servidor até o navegador do usuário, que recebe, renderiza e apresenta na tela ou dispositivo de saída padrão os dados solicitados (KRISHNAMURTHY; REXFORD, 2001).

Conforme Demétrio (2001) os navegadores revolucionaram o uso da Internet por tornar fácil e agradável a navegação pelo mundo virtual. Os *browsers* são programas que possibilitam a visualização de páginas com informações que estão disponíveis na rede. Ele é responsável pela leitura e interpretação dos documentos escritos em HTML para apresentar as páginas formatadas para o usuário.

3.2 DESENVOLVIMENTO *WEB*

Como mencionado, os documentos apresentados pelos *browsers*, na maioria das vezes, são escritos, utilizando a linguagem HTML, porém ao tratar de desenvolvimento *Web*, deve-se buscar a origem desta linguagem para o início do encadeamento da abordagem das linguagens utilizadas.

A *Standard Generalized Markup Language* (SGML) é uma linguagem de marcas criada a aproximadamente 30 anos para representação de informações em texto, reconhecida como padrão ISO 8879 em 1986. Ela não é um conjunto predeterminado de marcas e permite a definição de qualquer grupo, ou seja, uma ferramenta auto descritiva, onde cada documento possui consigo sua própria especificação (BAX, 2001).

Assim, a linguagem mais utilizada para o desenvolvimento das páginas *Web* é o *HyperText Markup Language* (HTML) ou linguagem de marcação em hipertexto. Esta é derivada do padrão SGML, uma metalinguagem para descrever outras linguagens (ALVARENGA; SOUZA, 2004).

Inicialmente concebida como uma solução para publicação de documentos científicos em meios eletrônicos, a linguagem de marcação HTML logo se tornou popular no desenvolvimento para a Internet. Esta se expandiu rapidamente, adquirindo recursos para atender expectativas de usuários e sistemas (ALMEIDA, 2002).

Como exemplo, impulsionados pela disputa de mercado, a *Microsoft* e a *Netscape* desenvolveram seus próprios elementos ou *tags* que somente seus navegadores entendiam. Dentre estas estão a <*bgsound*> para agrupar som em um documento suportado apenas pelo *Internet Explorer* e a <*layer*>, para montar conteúdos sobrepostos em uma página, somente para o *Navigator*[2] (VALENTINE; MINNICK, 2001).

Porém, estas funcionalidades (*tags*) que proporcionaram o crescimento da HTML, foram implementadas por empresas diferentes de forma proprietária (desconhecidas por seus concorrentes), tornando inadequado o uso destas na produção de páginas *Web*, pois somente usuários de determinados *browsers* conseguiam visualizar corretamente o conteúdo. Além disso, em geral os navegadores não incluíam suporte as *tags* da linguagem especificada (MARTINEZ, 2000).

Entretanto, a intenção era de que os *sites* ou sítios visualizados em diferentes navegadores possuíssem a mesma aparência. Porém ocorria e atualmente ainda acontece, diversidade em diversos detalhes, isso porque a HTML descreve como as páginas devem ser exibidas e os diferentes programas podem interpretar de maneira diferente estas instruções. Além de que, alguns desenvolvedores de *browser* incluíam suporte a recursos não-oficiais da linguagem (MILNER, 2001).

Assim, com o intuito de resolver estas particularidades e promover a evolução da *Web* garantindo compatibilidade entre as tecnologias, em 1994 foi fundada o *World Wide Web Consortium* (W3C) (ZELDMAN, 2003).

Desta forma, o consórcio escreveu especificações e diretrizes para o desenvolvimento à Internet. Durante anos o W3C referiu-se às especificações como “Recomendações” em uma tentativa de encorajar a *Netscape* e *Microsoft* a

---

[2] Navegador *web*, desenvolvido pela *Netscape Comunications Corporation.*

programarem as especificações. No início de 1998, o *The Web Standards Project* (WASP) rotulou novamente como "Especificações *Web*" persuadindo os fabricantes de *browsers* a oferecer suporte (ZELDMAN, 2003).

Contudo, a HTML é uma linguagem de formatação de texto que possibilita a programação de arquivos para Internet com extensão *htm* ou *html* sendo que esta possui algumas etiquetas (*tags*[3]) que indicam ao *software* de visualização destes documentos, denominados *browser*[4] como apresentar o conteúdo incluído ou referenciado no arquivo, na forma de texto, imagens ou suporte multimídia (NEVES, 2004).

Porém, existe a forma correta de realizar esta apresentação do conteúdo, mas conforme a *Web* ganhou popularidade, a HTML começou a ser utilizada com maiores propósitos visuais. Ao invés de utilizar elementos de títulos para manchetes nas páginas, preferia-se pela combinação de fonte e *tags* de negrito para criar o efeito desejado. As tabelas passaram a ser usadas como ferramenta de *layout* em vez do propósito de apresentação de dados e as pessoas utilizavam um <*blockquote*> (comando HTML que define citações) para adicionar espaços em branco. Um exemplo desta má utilização é apresentado na Figura 3, onde o *site* da *abcnews.com* utilizava tabelas para diagramação de *layout*, fonte grande em negrito para os títulos e o código não apresentava estrutura tornando difícil o entendimento (BUDD; MOLL; COLLISON, 2006).

[3] Marcações utilizadas para programação nas linguagens de desenvolvimento *Web*.
[4] *Softwares* utilizados para visualização de documentos da *Web*.

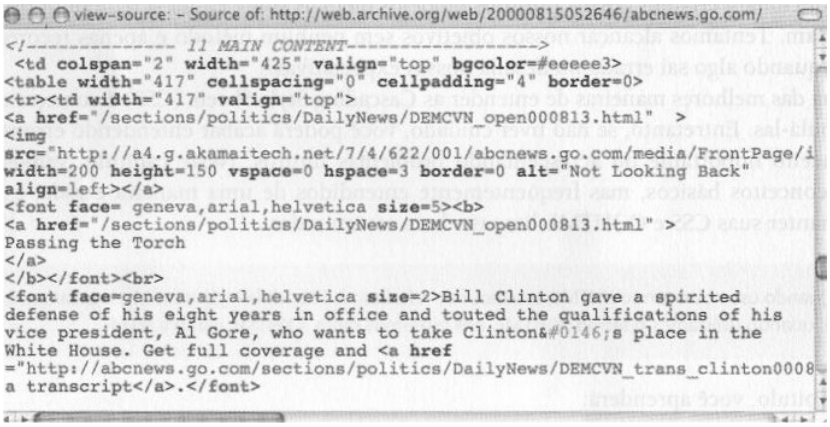

Figura 3. Marcação do material principal da *abcnews.com* em 14 de agosto de 2000
Fonte: BUDDY, A.; MOLL, C; COLLISON, S. (2006)

No ano de 1996, a W3C criou a especificação *Cascading Style Sheets* (CSS) 1.0 ou CSS1, com objetivo de substituir as marcações HTML de formatação, separando a camada de apresentação, das informações. Esta é uma poderosa e precisa linguagem de formatação visual, que reduz drasticamente o consumo de banda na rede (REBITTE; VINICIUS, 2006).

Além da HTML, também existem outras linguagens de marcação abertas e sem patentes, como pode-se citar a *eXtensible Markup Language* (XML) criada com a finalidade de descrever informações. Esta descende da SGML, apresentando uma nova abordagem em relação a HTML, pois oferece uma especificação para descrição, captura, processamento e publicação de informações (MCGRATH, 1999).

A linguagem XML foi idealizada por Jon Bosak, engenheiro da *Sun Microsystems*®, conhecedor e usuário da SGML. Ele apresentou ao W3C suas idéias de explorar ainda mais esta, em aplicações voltadas para Internet. Foi então que surgiu a XML em fevereiro de 1996, a princípio como uma simplificação da SGML, e em fevereiro de 1998, a linguagem tornou-se uma especificação formal, reconhecida pelo consórcio (ALMEIDA, 2002).

Apesar de serem parecidas as duas linguagens quanto ao uso das *tags* e sua sintaxe, a XML surgiu para complementar a HTML e não substituí-la, pois ao invés de

as *tags* representarem como os dados devem ser apresentados, as marcações indicam o que cada informação significa. Esta linguagem surgiu como uma ferramenta para auxiliar a divisão nos documentos do conteúdo, estrutura e apresentação (BRUNETTO, 2002).

Em meio a tudo isso, surgiu a XHTML, uma família de módulos e documentos atuais e futuros que reproduzem, englobam e ampliam o HTML 4. Os documentos desta nova linguagem são baseados em XML e modernamente tem sido projetados para trabalhar em conjunto com aplicações desenvolvidas de usuários. Esta foi concebida para ser uma linguagem de conteúdos em conformidade com o rigor de sua antecessora, seguindo algumas diretrizes simples, devendo ser compatível com aplicações em HTML 4 (W3C, 2000).

Resumindo, a linguagem XHTML é a XML que age como HTML em navegadores *Web* antigos e novos e que também funciona como esperado na maioria dos dispositivos desde telefones com conexão a Internet e suporte a estes protocolos, e leitores de tela, tornando portátil, prática e econômica a sua utilização (ZELDMAN, 2003).

Assim, quando se trata sobre especificações W3C para desenvolvimento *Web*, engloba-se as linguagens estruturais, de apresentação, de *scripting*, e modelos de objetos. Estes que envolvem as especificações HTML, XHTML, CSS, DOM, ECMAScript e outras não mencionadas neste trabalho (REBITTE; VINICIUS, 2006).

Após esta iniciação sobre a situação do desenvolvimento *Web,* na próxima sessão será abordado em maiores detalhes sobre as linguagens de marcas, especificações do W3C para as linguagens e modelos de objeto recomendados pelo consórcio, apresentando exemplos da aplicação de algumas especificações formais da W3C.

3.3 ESPECIFICAÇÕES W3C PARA DESENVOLVIMENTO *WEB*

Como mencionado anteriormente as especificações *Web* são criadas pelo W3C e outros consórcios de padrões para linguagens estruturais como XHTML e XML; de apresentação CSS e XSL; de modelos de objetos W3C DOM e linguagem de *script* como ECMAScript[5] (ZELDMAN, 2003).

O termo *Tableless* também pode ser denotado quando se trata das especificações *Web*, sendo que o WASP é um dos principais fundamentos para os adeptos desta metodologia. O *Tableless* pretende atingir as recomendações W3C. Sua tradução literal equivale a algo como “menos tabelas” sendo que foi denominado assim, devido a principal característica das especificações, que é a extinção do uso de tabelas para diagramação de *layout*. É importante observar que a metodologia não acorda com a extinção da *tag* <*table*> e sim seu uso semântico (REBITTE; VINICIUS, 2006).

O *Web Standards Project*, foi iniciado em 1998 como já mencionado para persuadir a *Netscape*®, *Microsoft*® e outros criadores de navegadores a suportarem as especificações. Um processo que demandou tempo, persistência e estratégia, porém apresentou resultados positivos. Os fornecedores de navegadores compraram a idéia de que a interoperabilidade via especificações comuns era uma necessidade absoluta caso a *Web* fosse à diante. Logo após os navegadores passarem a oferecer suporte significativo as estas recomendações o WASP foi reiniciado em 2002 para estimular os *designers* e desenvolvedores a aprenderem estas tecnologias (ZELDMAN, 2003).

Assim, designadas como recomendações ou especificações, as linguagens estruturais, de apresentação e *scripting* entre outras linguagens e metodologias, estão sendo desenvolvidas pelo W3C com diretivas claras sobre seu uso. Com estas, será

---

[5] Linguagem de script referenciada como “padrão JavaScript” criada pela entidade de padrões *European Computer Manufacturers Association* (ECMA).

possível construir sites estruturados, acessíveis a um número maior de usuários e a diferentes dispositivos de visualização, aliado a redução de custos de produção e maior facilidade de manutenção comparada aos métodos ainda hoje utilizados (MACEDO, 2004).

Tais métodos, como a criação de várias versões de códigos e marcações não padronizadas, voltadas para diversas particularidades de um navegador ou outro, é a origem da perpétua obsolescência. Mesmo com navegadores que suportam estas especificações muitos desenvolvedores o tratam como se não o fosse. Assim eles escrevem *scripts* que procuram pelo *browser, Internet Explorer 6* (IE6) e alimentam com códigos somente para ele, muito embora este navegador possa manipular o *ECMAScript* e o DOM. Estes programadores preferem escrever códigos duplicados e utilizar destes *scripts* de detecção que muitas vezes falham, a um esforço de aprender novas técnicas (ZELDMAN, 2003).

Conforme Nielsen (2000) as vantagens da codificação baseada na apresentação são poucas em relação aos custos e a dificuldade da criação de múltiplas versões para funcionar em diferentes ambientes. Esta técnica atualmente não deve ser empregada, pois sem o uso dos *scripts* de detecção, exige que seja previsível o *software* e o *hardware* do usuário. Isso era possível nos primeiros anos da *Web*, pois de 1991 a 1992, a maioria dos usuários realizavam acesso em modo texto, nos próximos dois anos utilizavam o *Mosaic*[6] e nos dois subseqüentes utilizavam o *Navigator*. Porém, a partir de 1997, a noção de um *browser* utilizado por todos foi abandonada.

Um exemplo da falha dos *scripts* de procura de navegadores é a instalação do *browser* Opera no sistema operacional *Windows*®. No processo de identificação ele

---

[6] *Mosaic* é conhecido como o primeiro navegador web, foi desenvolvido no *National Center for Supercomputing Applications* (NCSA), lançado em 1993.

se passará por *Internet Explorer,* para evitar ser bloqueado por *sites* que procuram pelo IE, como muitos aplicativos de instituições financeiras (ZELDMAN, 2003).

Também na tarefa de prever a arquitetura utilizada, pode-se afirmar que o IE domina o mercado, mas as previsões levam a popularização de aparelhos de informação, cada qual com sua própria característica. É muito pouco provável que um único *browser* ofereça ótima interface com o usuário sob condições variáveis. Os computadores não tradicionais como os assistentes pessoais *Palm*, possuem capacidade de exibição diferente, como a tela menor, além de que estão sendo disponibilizados navegadores utilizáveis por voz, para deficientes, os quais exigem uma marcação semântica para o funcionamento adequado (NIELSEN, 2000).

Além dos *scripts* proprietários de detecção, muitos desenvolvedores escrevem marcações de apresentação que duplicam a largura de banda necessária, tornando menos acessível a motores de busca e navegadores de dispositivos não tradicionais. Tais estratégias frequentemente causam muitos problemas as quais o propósito seria resolver como apresentação inconsistente entre diferentes navegadores. A Figura 4 apresenta um exemplo desta situação, o site *MSN Game Zone* (*zone.msn.com/blog.asp*), utiliza sete folhas de estilos CSS e ainda não consegue apresentação adequada na maioria dos *browsers* modernos, além de utilizar 14 *scripts* pesados de detecção de navegadores em série (ZELDMAN, 2003).

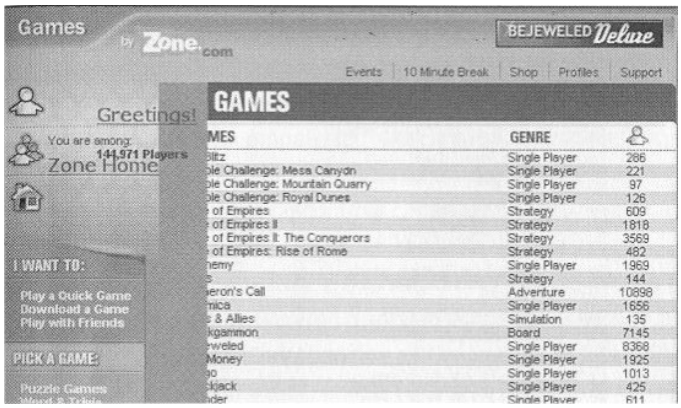

Figura 4. *Site MSN Game Zone*
Fonte: ZELDMAN, J. (2003)

Assim, pode-se observar que a HTML foi concebida como uma linguagem simples e compreensível, mas à medida que o aspecto visual das páginas tornou-se cada vez mais importante o código tornou-se quase ininteligível. Com isto, as páginas ficavam muito complicadas, e a manutenção manual era uma missão difícil e ocasionalmente durante a tarefa outros problemas não previstos apareciam. Na Figura 5, podemos observar a melhoria do site *abcnews.com* em 2006, comparado à estrutura que apresentava no ano 2000 demonstrado na Figura 3, melhorias conseguidas por meio da utilização de XHTML e CSS (BUDD; MOLL; COLLISON, 2006).

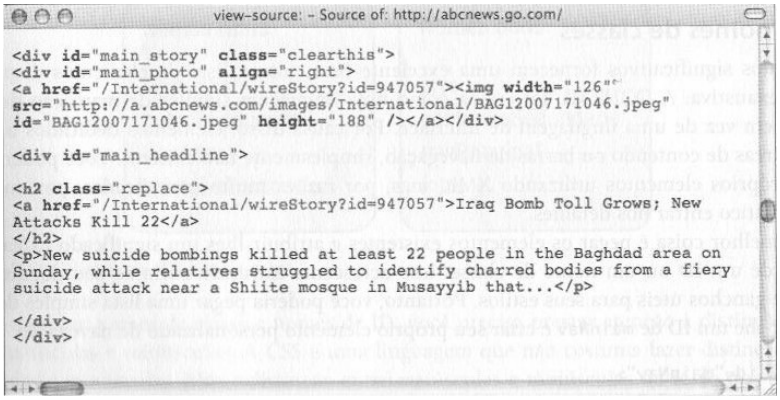

Figura 5. A marcação do material principal da *abcnews.com* em 2006
Fonte: BUDDY, A.; MOLL, C; COLLISON, S. (2006)

Estas especificações adotadas, neste sítio por meio das tecnologias XHTML e CSS, trouxeram a organização aos códigos fonte facilitando a tarefa de manutenção, com economia de recursos e reaproveitamento de código. Com a CSS, tornou-se possível controlar a aparência externa da página e fazer a separação do conteúdo de sua apresentação (BUDD; MOLL; COLLISON, 2006).

Desta forma, substituindo as *tags* <*font*> desatualizadas e outras de apresentação por marcações estruturais e limpas por meio de regras CSS de uma página que não utiliza as especificações, que atualmente consome 60 *kbytes*, pode-se reduzir para 30 *kbytes*. Uma economia de 50% que reflete em velocidade de transferência, satisfação dos usuários, economia de recursos e dinheiro, pois o proprietário reduz significativamente o valor pago pela transferência de dados ao servidor de páginas (ZELDMAN, 2003).

Conforme Bax (2001) a grande vantagem quanto à utilização das especificações de marcações internacionais abertas é a criação de documentos portáveis, isto é, documentos independentes de determinados *softwares*, *hardwares*, ou sistemas operacionais.

Baseado nisso, e não poderia ser diferente, o *site The Web Standards Project* é construído com a XHTML1.0 *Strict* e CSS para o *layout*. A Figura 6 apresenta como o *webstandards.org* é visualizado no *PalmPilot*, e a Figura 7 apresenta o funcionamento no *PocketPC* (ZELDMAN, 2003).

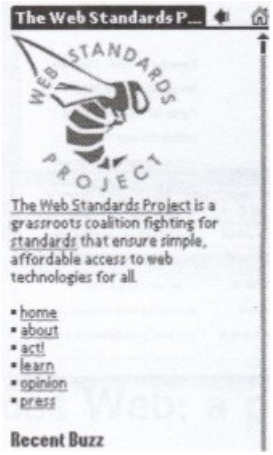

Figura 6. *Site* do WASP apresentado no *PalmPilot*
Fonte: ZELDMAN, J. (2003)

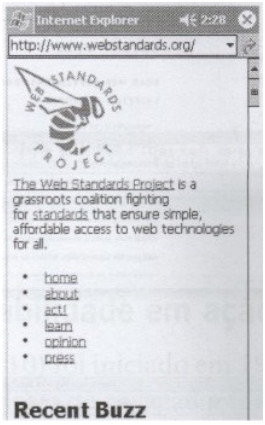

Figura 7. *Site* do WASP apresentado no *PocketPc*
Fonte: ZELDMAN, J. (2003)

Este funcionamento do *site* do WASP nos dispositivos móveis, sem necessitar de múltiplas versões deve ser entendido como suporte básico. Porém a criação de diferentes folhas de estilos com especificações para cada dispositivo pode ser realizada para aperfeiçoamento. O atributo *type* do elemento <*link*> permite estas especificações de mídia, que serão apresentadas em detalhes posteriormente (MACEDO, 2004).

Projetar e construir com as especificações W3C significa não precisar mais criar várias versões de cada *site*. A compatibilidade estrita com estas, também fornece uma enorme vantagem na solução do problema de acessibilidade. Se um site é acessível

por meio de um *PalmPilot*, é muito provável que ele funcione em um leitor de telas como o *Jaws*, embora testar seja sempre coerente (ZELDMAN, 2003).

Conforme Macedo (2004) a adoção destas especificações no desenvolvimento de *sites* proporciona diversos benefícios, sendo:

a) liberdade: por se tratar de domínio público permite diversidade na estruturação e inovação, podendo ser usado por qualquer pessoa em qualquer situação, sem necessidade de autorizações ou qualquer tipo de pagamento;

b) estabilidade: as páginas desenvolvidas com estes métodos permanecerão compatíveis em ambientes com suporte limitados as especificações e pode-se obter acesso completo ao conteúdo.

c) acessibilidade: estas especificações permitem o acesso facilitado (a pessoas com algum tipo de deficiência) ao conteúdo do site e oferecem suporte em outros ambientes, evitando a duplicação de conteúdo;

d) facilidade de criação e manutenção: com a separação do conteúdo da apresentação facilita-se estas tarefas.

Segundo Zeldman (2003) as especificações W3C, resolvem os problemas discutidos separando qualquer página *Web* em três componentes como: estrutura, apresentação e comportamento, conforme apresentado na Figura 8.

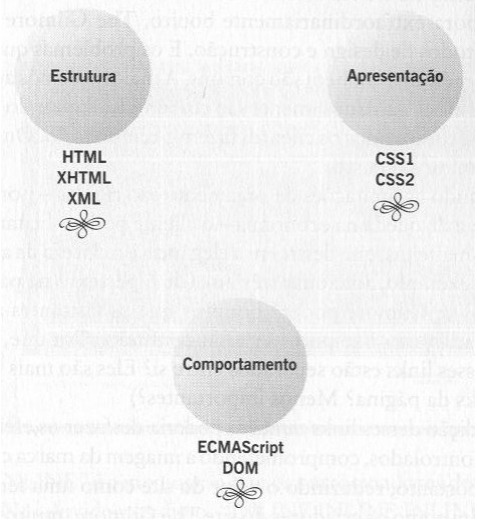

Figura 8. Trio das especificações *Web*
Fonte: ZELDMAN, J. (2003)

Estes componentes e outros criados pelo consórcio, são elaborados por comitês de especialistas, sendo cuidadosamente projetados para proporcionar benefícios ao maior número de usuários. Estas tecnologias formam um mapeamento para um desenvolvimento *Web*, econômico, racional, acessível e sofisticado (ZELDMAN, 2003).

Conforme as vantagens apresentadas por Macedo (2004), pode-se observar que algumas foram abordadas, porém os aspectos vêm à discussão e outros não muito relevantes são pouco mencionados. Baseado neste conceito, as recomendações serão levantadas durante o desenvolvimento cronológico do trabalho, a qual a especificação se encaixa. Assim, os componentes do *Trio das Especificações Web* e seus respectivos itens serão apresentados a seguir, mas antes é necessário entendimento sobre acessibilidade *Web*.

3.3.1 Acessibilidade *Web*

Acessibilidade e especificações W3C possuem muita coisa em comum. Ambas garantem que o produto será útil e disponível ao maior número de leitores,

visitantes e clientes. A acessibilidade está fortemente vinculada às recomendações do consórcio, tanto é que o W3C na década de 90 iniciou a *Web Accessibility Initiative* (WAI) para ajudar os desenvolvedores *Web* em estratégias para consegui-la (ZELDMAN, 2003).

Assim sendo, a acessibilidade de um produto consiste em considerar a diversidade de usuários e peculiaridades de interação destes com o produto, o que pode manifestar-se tanto como preferência do usuário (o que prefere ouvir a ler), quanto nas restrições à qualidade do equipamento utilizado (monitores monocromáticos) ou até mesmo de necessidades especiais que não podem ser ignoradas pelos desenvolvedores (TORRES; MAZZONI, 2004).

Assim, os projetos de acessibilidade a *Web* devem considerar os diferentes contextos, possibilitando atender pessoas que:

a) estiverem incapacitadas de ver, ouvir, deslocar-se, ou interpretar determinados tipos de informações;
b) tenham dificuldades em ler ou compreender textos;
c) não consigam utilizar teclado ou *mouse*;
d) possuam monitor que apresenta apenas texto, ou com dimensões reduzidas, ou ainda uma conexão lenta com a Internet;
e) não falem ou compreendam fluentemente o idioma em que o documento foi escrito;
f) estejam com seus olhos, mãos ou ouvidos ocupados;
g) possuam uma versão ultrapassada de navegador *Web*, diferente dos habituais, um navegador por voz, ou ainda um sistema operacional pouco convencional (FUJISAKI et al, 2004).

Os criadores de conteúdo devem levar em conta as diferentes situações, embora haja uma variedade, ao conceberem páginas para a *Web*. Cada *design* de página, para ser verdadeiramente acessível, deve ser útil a diversos grupos de incapacidades ou deficiências simultaneamente e por extensão, ao universo dos usuários (W3C, 1999).

Uma das questões com a qual a acessibilidade tem de lidar é o preconceito existente em relação à utilização por parte de indivíduos portadores de necessidades especiais de *sites* e ferramentas *Web*. Esse preconceito tem mais a ver com a falta de informação dos desenvolvedores de *sites* e aplicações *Web*, em não perceber a utilidade das suas aplicações para estes indivíduos. Estes criadores de conteúdos parecem desconhecer o verdadeiro objetivo da acessibilidade, que é garantir o pleno aproveitamento por todos os cidadãos dos benefícios que advêm da sociedade da informação (FREITAS; BENJAMIN; PASTOR, 2004).

Assim, portadores de necessidades especiais podem utilizar a *Web* para transações financeiras de compra e venda se os *sites* os permitissem ler as especificações de produtos e a usar seus formulários de negociações e pedidos. Além de que, a maioria dos usuários *Web* com deficiências visuais não são totalmente cegos, ou nem mesmo próximo disso. Grande parte destes que requerem melhoria de acesso varia de pessoas com pouca visão, daltônicas e ligeiramente míopes que também poderiam querer adquirir o produto (ZELDMAN, 2003).

As recomendações da WAI abordam questões de acessibilidade, apresentam soluções de *design* e concentram-se em cenários típicos que podem trazer problemas a usuários com determinadas incapacidades. Por exemplo, a *recomendação 1* explica de que modo os criadores de conteúdo podem tornar as imagens acessíveis. Alguns usuários podem não serem capazes de ver imagens; outros podem utilizar navegadores textuais e que não suportam imagens; e ainda outros podem ter desativado o suporte a

imagens. As recomendações não aconselham que sejam evitadas imagens para que a acessibilidade melhore, apenas explicam que a apresentação de um *equivalente textual* da imagem pode torná-la acessível (W3C, 1999).

Assim a partir da evolução computacional e da preocupação em torná-la mais acessível, surge o termo *acessibilidade digital* que representa a disponibilização de conteúdos da *Web* para o maior número de pessoas possível incluindo os portadores de necessidades especiais (FREITAS; BENJAMIN; PASTOR, 2004).

Para lidar com a questão do preconceito e apresentar a importância das especificações do W3C e suas respectivas vantagens quanto à acessibilidade, existe a teoria do *Bilionário Cego*. Alguns pesquisadores destacam que o motor de busca *Google* é o maior usuário cego da *Web*, e este fornece recomendações na forma de resultados de pesquisa a milhares de clientes constantemente (ZELDMAN, 2003).

O grupo de trabalho da WAI atribuiu a cada ponto de verificação das recomendações um nível de prioridade, com base no impacto em termos de acessibilidade. As prioridades e suas especificações formais são:

a) *Prioridade 1*: é um requisito básico, pois apresenta os pontos que *devem* ser satisfeitos, caso contrário um ou mais grupos ficarão impossibilitados de acessar as informações contidas nos documentos;

b) *Prioridade 2*: especifica os pontos que *deveriam* ser satisfeitos, se não o forem um ou mais grupos de usuários terão dificuldades em acessar as informações contidas nos documentos;

c) *Prioridade 3*: são os pontos em que os desenvolvedores *podem* satisfazer, se não o fizerem, um ou mais grupos poderão se deparar com algumas dificuldades em acessar as informações contidas nos documentos (W3C, 1999).

Assim, muitos países possuem leis contra a negação de acesso aos incapacitados e muitas já aplicaram as leis à nova mídia por meios de decretos de acessibilidade da *Web*, como o *U.S. Section 508*. Algumas destas leis obedecem a prioridade 1 da WAI (ZELDMAN, 2003).

No Brasil, o Decreto-lei 5296 de 2 de dezembro de 2004, regulamenta a Lei n.° 10.098, de 19 de dezembro de 2000, que estabelece normas gerais e critérios básicos para a promoção da acessibilidade (BRASIL, 2004).

Neste decreto, em seu Capítulo VI trata-se sobre o acesso a informação e comunicação, tornando obrigatória a acessibilidade nos portais e sítios eletrônicos da administração pública na rede mundial de computadores (Internet), para o uso das pessoas portadoras de deficiência visual, garantindo-lhes o pleno acesso às informações disponíveis. Também é estabelecido que os sítios eletrônicos acessíveis devam conter símbolo que represente a acessibilidade na rede mundial de computadores, a ser adotado nas respectivas páginas de entrada (BRASIL, 2004).

Conforme Zeldman (2003) a *Section 508*, requer que todos os sites sob sua jurisdição forneçam acesso igual ou equivalente a todos, incluindo portadores de deficiência visual, física ou epilepsia fotosensitiva com o comprimento de todas as especificações de prioridade 1, especificadas pela WAI. Esta se tornou lei no dia 21 de junho de 2001 e aplica-se aos departamentos e agências federais (incluindo o serviço postal americano) e atividades patrocinadas ou fundadas pelo governo.

Segundo a W3C (1999) além dos níveis de prioridade, foram criados níveis de conformidade, estas que quando especificadas em documentos devem seguir os formatos também indicados pelo consórcio, por meio de imagens fornecidas pela W3C com o *link* para a explicação formal do significado ou escrita, devendo conter o título e URI, da recomendação, os nível de conformidade satisfeito e o âmbito atingido no

documento. Conforme esta descrição: *Esta página está de acordo com o documento do W3C "Web Content Accessibility Guidelines 1.0" disponível em http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505, de nível "Duplo A"*. Os níveis de conformidade definidos são:

a) **A**: utilizado em páginas em que foram satisfeitos todos os pontos de verificação de prioridade 1;

b) **Duplo A**: utilizado em páginas que foram satisfeitos todos os pontos de verificação das prioridades 1 e 2;

c) **Triplo A**: utilizado em páginas onde foram satisfeitos todos os pontos de verificação das prioridades 1, 2 e 3.

Assim, como seu próprio objetivo, a acessibilidade é muito abrangente e envolve diversos aspectos que alimentam muita discussão. Para não escapar do contexto, a seguir serão apresentados alguns pontos de verificação mais relevantes a este trabalho, das principais recomendações *Web Content Accessibility Guidelines 1.0* (WCAG 1.0) do WAI de 5 de maio de 1999.

A primeira recomendação deste documento informa: *Fornecer alternativas ao conteúdo sonoro e visual*. Nesta recomendação dois pontos de verificação podem ser considerados mais relevantes como:

a) *Fornecer um equivalente textual a cada elemento não textual por meio dos atributos alt ou longdesc* (Prioridade 1). Isto abrange imagens, representações gráficas, regiões de mapas de imagem, animações e objetos programados, arte ASCII, *frames*, programas interpretáveis, imagens utilizadas como sinalizadores de pontos de enumeração, espaçadores, botões gráficos, sons (reproduzidos ou não com interação do usuário) e trechos de vídeo;

b) *Fornecer uma descrição sonora das informações importantes veiculadas em trechos visuais das apresentações multimídia, até que os agentes do usuário consigam ler, automaticamente e em voz alta, o equivalente textual dos trechos visuais* (Prioridade 1) (W3C, 1999).

Eliminar o atributo *alt* fará com que usuários do *Lynx*[7], leitores de tela e outros navegadores e dispositivos menos utilizados escutem ou vejam no lugar das imagens ou elementos não textuais algo parecido com *[IMAGE]*. Para imagens sem significado como *GIFs* espaçadores, deve ser usado *alt=""*, também conhecido com atributo *null alt*, planejados para manter efeitos de projetos puramente visuais, algo como sem significado, sem semântica (ZELDMAN, 2003).

Como observado, o atributo *alt* fornece uma descrição alternativa sobre as imagens para *browsers* que não podem apresentar imagens ou estejam com esta funcionalidade desligada. Já o atributo *longdesc*, especifica um caminho para uma descrição mais longa da imagem. Este elaborado para complementar a descrição fornecida pelo *alt* (MACEDO, 2004).

Porém, alguns cuidados quando da utilização do atributo *alt* devem ser tomados. Alguns navegadores líderes exibem erradamente este atributo como dicas de ferramentas quando o cursor do mouse passa sobre a imagem. Assim, este recurso é uma excelente ferramenta de acessibilidade e não um recurso de dica de ferramenta. O W3C diz explicitamente que o texto *alt* deve ser visível somente quando as imagens não podem ser visualizadas (ZELDMAN, 2003).

Um exemplo da utilização do atributo *alt* nas imagens seria: *<img src="logotipo2.gif" width="77" height="31" border="0" alt="Segundo logotipo do Site">*. Neste caso os atributos *src*, *width*, *height* e *border* especificam respectivamente

[7] Navegador modo texto

a localização do arquivo de imagem, largura e altura que a imagem será exibida em *pixels* e a espessura das bordas para a imagem (MACEDO, 2004).

Um outro ponto interessante destacado entre as recomendações de acessibilidade condiz às cores. A segunda recomendação do documento sugere: *Não recorrer apenas à cor*. Alguns cuidados devem ser tomados quanto às cores, pois se ela for o único meio de transmitir informações as pessoas que não forem capazes de diferenciar certas cores ou até mesmo usuários de monitores monocromáticos não receberão estas informações. Outro cuidado é quanto à semelhança dos tons de primeiro plano e fundo, deve-se assegurar de que sejam suficientemente contrastantes e perceptíveis mesmo por telas monocromáticas ou pessoas com diferentes cromodeficiências (Prioridade 2 para imagens e 3 para textos) (W3C, 1999).

Na *Web*, há bons e maus exemplos de combinações de cores desde *sites* em que não se consegue enxergar as letras a outros que utilizam de cores fortes e significantes. A utilização cautelosa assegura o contraste, e desta forma usar fundo claro e para primeiro plano, tons escuros ou inverso é sempre uma boa recomendação (RIOS, 2003).

Deve-se também, evitar fazer referência a cores no texto, algo como: *Veja a caixa amarela para obter ajuda*, isto é uma diretriz inútil para aqueles que não podem ver ou distinguir. Uma escolha cuidadosa deve ser feita para que as diferenças entre os tons de fundo e primeiro plano, *links* e outros pontos de destaque, também sejam aparentes as pessoas com daltonismo (ZELDMAN, 2003).

A terceira recomendação aconselha: *Utilizar corretamente marcações e folhas de estilo*. Isto requer a marcação com elementos estruturais adequados e controlar as aparências por meio de folhas de estilo em cascata, em vez de elementos de apresentação e atributos. Nesta recomendação existem vários pontos de verificação

relevantes à utilização das especificações e ambos possuem nível de prioridade 2, são eles:

a) *Sempre que existir uma linguagem de marcação apropriada, utilizar marcações em vez de imagens para transmitir*. Um exemplo desta utilização seria utilizar *MathML* para marcar equações matemáticas e CSS para paginação. Deve-se também evitar o uso de imagens para representar texto;

b) Criar documentos passíveis de validação por gramáticas formais, publicadas. Por exemplo, incluir uma declaração de tipo de documento no início do arquivo, que se refira a uma *Document Type Declaration* publicada. Estas que serão apresentadas na seção HTML, XML E XHTML;

c) *Utilizar elementos de cabeçalho indicativos de estrutura do documento, de acordo com as especificações*. Isto indica a utilização semântica da *tag* <*h2*> indicando uma subseção de <*h1*>;

d) *Marcar corretamente listas e pontos de enumeração em listas*. Esta indica a utilização correta das listas OL, UL e DL que serão também apresentadas na seção da XHTML;

e) *Marcar as citações. Não utilizar marcações de citação para efeitos de formatação, como, por exemplo, o avanço de texto*. Referindo utilização correta das *tags* <*q*> e <*blockquote*> no seu sentido semântico, marcar citações curtas e mais extensas (W3C, 1999).

As técnicas destes pontos de verificação serão melhores exemplificadas quando se tratar sobre HTML, XML e XHTML e CSS, mas já se pode observar a importância de uma marcação estruturada que transmite significado mesmo quando a

CSS não está presente. Para efeito de testes, pode-se desativar no *browser* a folha de estilos. Como resultado funcionamento não poderá ficar comprometido. (ZELDMAN, 2003).

A quarta recomendação do documento da W3C informa: *Indicar claramente qual o idioma utilizado*. Assim, deve ser especificada a linguagem predominante do documento (Prioridade 3), por meio do atributo *lang* nos cabeçalhos como a *tag* <*meta*>. As mudanças de linguagem no contexto do documento também devem ser informadas (Prioridade 1). Esta recomendação auxilia os sintetizadores de voz e dispositivos braile, a mudarem automaticamente para o novo idioma na leitura dos documentos (W3C, 1999).

Segundo Macedo (2004) o elemento *meta* é utilizado para incorporar informações sobre os dados contidos no documento. Estas informações podem ser utilizadas para identificação, indexação e catalogação do *site*. Alguns dos recursos desta *tag* serão apresentados no contexto da seção XHTML.

O documento também apresenta informações sobre tabelas, indicando alguns pontos de verificação para este recurso. A recomendação referente às tabelas diz: *Criar tabelas passíveis de transformação harmoniosa***.** Assim, deve-se assegurar que estes elementos possuam marcações necessárias para poderem ser transformados de forma correta por navegadores acessíveis e outros agentes de usuário. Os pontos de maior relevância foram destacados:

a) Em tabelas de dados, identificar os cabeçalhos de linha e de coluna (Prioridade 1). Deve-se assegurar a utilização da *tag* <*td*> para identificação dos dados e <*th*> para identificação dos cabeçalhos;

b) *Em tabelas de dados com dois ou mais níveis lógicos de cabeçalhos de linha ou de coluna, utilizar marcações para associar as células de*

dados às células de cabeçalho (Prioridade 1). Para satisfazer este ponto, devem-se utilizar as *tags* <*thead*>, <*tfoot*>, <*tbody*>. Para agrupar linhas e colunas os atributos *col* e *colgroup* e *axis* e *scope* respectivamente. Utilizar também o atributo *headers* para estabelecer relações entre os dados;

c) *Não utilizar tabelas para efeitos de disposição em página, a não ser que a tabela continue a fazer sentido depois de ser linearizada* (Prioridade 2);

d) *Fornecer resumos das tabelas* (Prioridade 3). Por meio do atributo *summary* da *tag* <*table*>;

e) *Fornecer abreviaturas para os rótulos de cabeçalho* (Prioridade 3). Para isto, utilizar o atributo *abbr* na *tag* <*th*> (W3C, 1999).

Segundo Zeldman (2003) uma pessoa com visão normal e que esteja utilizando um navegador gráfico verá facilmente a conexão dos dados nas colunas pelo fato de estarem logo abaixo e assim entenderão a tabela. Mas usuários de leitores de tela precisam de marcações adicionais que conecte o cabeçalho a suas células de dados associadas.

Assim, a sétima recomendação também é muito relevante, sendo: *Assegurar o controle do usuário sobre as alterações temporais do conteúdo*. Deve-se assegurar a possibilidade de interrupção momentânea ou definitiva de movimento, intermitência, transcurso ou atualização automática de objetos na página (W3C, 1999).

Desta forma, seguindo esta recomendação, devem-se evitar páginas *Web* que contenham intermitências, piscam ou cintilam, pois podem iniciar ataques mortais em portadores de epilepsia e textos em movimentos que pode tornar-se de difícil leitura.

Uma solução é oferecer controle para o usuário sobre a velocidade dos textos ou intermitência na tela (ZELDMAN, 2003).

Segundo a W3C (1999) devem-se evitar concepções que possam provocar intermitência da tela (Prioridade 1). Um efeito de pulsar na faixa de 4 a 59 pulsos por segundo (Hertz), sendo o pico de sensibilidade 20 pulsos por segundo, bem como uma rápida passagem de uma quase escuridão para uma iluminação excessiva, pode desencadear ataques ou ausências nas pessoas com epilepsia fotossensível. Deve-se também evitar textos em movimento, pois leitores de tela e agentes de usuários pouco convencionais não conseguem acesso às informações neles contidos. Para este último cabe uma observação, as *tags* <*blink*> e <*marquee*> não são definidas em nenhuma especificação HTML da W3C e desta forma não devem ser utilizadas.

Conforme Marcondes (1998) com a *tag* <*marquee*>, é possível obter efeitos de animação de texto por meio da formatação. Com ela pode-se fazer o texto se mover de um lado para outro das páginas. Da mesma forma a *tag* <*blink*> faz com que o texto fique piscando na tela. Porém, como ele mesmo informa os efeitos somente são apresentados pelo *Internet Explorer*®, abandonando usuários do *Netscape*®.

A nona recomendação diz: *Projetar páginas considerando a independência de dispositivos*. Isto inclui suporte nas aplicações por diversos dispositivos de entrada ou saída de preferência do usuário, como mouse, teclado, voz e outros. Alguns dos pontos de verificação desta recomendação são destacados:

a) deve-se assegurar que qualquer elemento dotado de interface, possa funcionar independente de dispositivos (Prioridade 2);

b) *Criar uma seqüência lógica de tabulação para percorrer links, controles de formulários e objetos* (Prioridade 3). Para isto, deve-se

especificar a ordem de tabulação por meio do atributo *tabindex* ou ter um *design* de página claro e lógico;

c) *Fornecer atalhos por teclado que apontem para links importantes* (Prioridade 3). Requisito possível de ser atendido com a utilização do atributo *accesskey* (W3C, 1999).

Os atributos do elemento *link* como *accesskey* e *tabindex*, oferecem funcionalidades com o mesmo propósito, facilitar a navegação dos usuários. Porém suas funcionalidades possuem diferenças, pois o *accesskey* especifica um atalho de teclado enquanto o *tabindex* fornece a seqüência de tabulação entre os demais *links* (MACEDO, 2004).

Conforme Zeldman (2003), o atributo *tabindex* especifica a ordem de navegação por tabulação entre controles de formulários. Desta forma organizando uma seqüência de tabulação lógica, as pessoas que preferem este recurso ao mouse simplesmente usarão a tecla *tab* para navegar nos *links*.

A décima primeira recomendação do WCAG 1.0, simplesmente informa que *devem ser utilizadas as tecnologias e recomendações do W3C*, pois estas incluem funções de acessibilidade. O principal ponto de verificação informa que devem ser utilizadas as tecnologias adequadas a determinadas tarefas buscando as versões mais recentes, desde que suportadas (Prioridade 2). Esta que também serve como advertência para não utilizar as funcionalidades desatualizadas como a *tag* <*font*> substituindo por folhas de estilo (W3C, 1999).

Assim, as recomendações da W3C proporcionam as páginas que os seguem, suporte em diversos dispositivos não-tradicionais, desde acessórios sem fio e telefones celulares compatíveis com a *Web*, até monitores braile e leitores de tela utilizados por usuários com deficiências físicas (ZELDMAN, 2003).

3.3.2 Linguagens de Marcação HTML, XML e XHTML

As linguagens de marcação referem-se à descrição da forma dos documentos, ou seja, como o conteúdo do documento deve ser interpretado. Uma das linguagens de marcação mais conhecidas é a HTML usada para criar páginas da *Web* (HOLZNER, 2001).

Assim a HTML descreve a estrutura, o conteúdo e a apresentação de um documento e sua relação com outros documentos. Desta forma, os elementos que compõem a estrutura de um documento nesta linguagem são denominados de *tags*. Com isto, um documento é composto de dois tipos de texto, as informações a serem exibidas e as *tags* (elementos de marcação) que definirão como os navegadores irão apresentar estas informações (MACEDO, 2004).

A forma de trabalho desta linguagem é simples e lógica, pois a HTML é escrita com textos e sua leitura é realizada da esquerda para direita e de cima para baixo. Os comandos, denominados de *tags*, são usados para configurar determinadas seções do texto ou conteúdo multimídia, como por exemplo, textos grandes, pequenos, negrito, sublinhados, *links* e outros (RIOS, 2003).

Uma *tag* ou etiqueta, como indica sua tradução, é sempre envolvida por dois caracteres sendo eles: < (menor que) e > (maior que). Muitas vezes na HTML as *tags* possuem uma outra para fechamento, ou seja, uma indica o início de abrangência no documento e outra o final. As *tags* de fechamento são idênticas as de abertura, porém precedidas do caractere / (barra). Nesta linguagem elas podem ser escritas em minúsculo ou maiúsculo e existem algumas *tags* solitárias, sendo que não possuem uma *tag* de fechamento correspondente (MACEDO, 2004).

Conforme Marcondes (1998) um exemplo do formato de uma etiqueta HTML ou *tag* é <*tag*> para abertura e muitas vezes <*/tag*> para fechamento. São raras as exceções que descartam o uso da etiqueta de finalização, alguns exemplos destas são as *tags* para pular de linha <*br*> e de inserir divisórias <*hr*>. Um outro ponto da linguagem, é que ela não sofre influência de caixa, o que significa que pode ser escrito uma marcação como <*html*>, <*HTML*>, <*Html*>, <*hTML*>, ou <*HtMl*> o comando é o mesmo e os navegadores irão interpretar da mesma forma.

Algumas *tags* podem possuir atributos que definem suas características ou propriedades, estes devem ser inseridos na *tag* inicial, utilizando a sintaxe: *nome_do_atributo=”valor”*. Uma etiqueta pode possuir vários atributos separados por espaço em branco e não é obrigatório, mas altamente recomendado que os atributos venham delimitados pelos caracteres de aspas duplas conforme a sintaxe demonstrada (MACEDO, 2004).

O desenvolvimento da linguagem HTML começou no início da década de 90 e desde o surgimento a linguagem possui quatro versões oficiais publicadas pelo consórcio de padrões sendo: 2.0, 3.2, 4.0 e 4.01. Em meio a estas especificações, como mencionado anteriormente, a *Netscape* e a *Microsoft* desenvolveram seus próprios elementos que somente seus navegadores entendiam (VALENTINE; MINNICK, 2001).

Com isto, os programadores de aplicativos para *Web*, possuíam diversos recursos interessantes, porém com funcionalidades independentes, e ao utilizar determinadas opções na construção, o funcionamento correto em diversos navegadores e dispositivos era pouco provável. Desta forma surgiram as especificações da linguagem criadas pelos consórcios de padrões, com o intuito de assegurar o correto funcionamento das tecnologias e acabar com esta disputa dos navegadores conhecida como *Guerra dos Browsers* (ZELDMAN, 2003).

A primeira recomendação da HTML denominada 2.0, foi desenvolvida pela *The Internet Engineering Task Force* (IETF), como *Request for Comment* (RFC) 1866 em novembro de 1995. As especificações contidas neste documento correspondiam às potencialidades do HTML no uso comum antes de junho 1994 (RFC 1866, 1995).

A HTML 3.2 foi publicada no dia 14 de janeiro de 1997, como especificação da W3C, esta foi desenvolvida no final de 1996 junto com empresas como *IBM*®, *Microsoft*®, *Netscape Communications Corporation*®, *Novell*®, *SoftQuad*®, *Spyglass*®, e *Sun Microsystems*®. Esta publicação adicionou funcionalidades em relação à recomendação 2.0 da linguagem, tais como tabelas, *applets* e fluxo de texto em torno das imagens, e fornecendo total compatibilidade com versão até então existente. Esta nova publicação envolveu também outras especificações recomendadas pela IETF como a *RFC1942 – HTML Tables* e *RFC1867 - Form-based File Upload in HTML* (W3C, 1997).

Assim, logo surgiu a especificação da HTML 4.0, publicada pela W3C no dia 18 de dezembro de 1997 e revista a 24 de abril de 1998. Esta ampliava ainda mais os recursos da linguagem, incluindo: mais opções de multimídia, suporte a linguagens de *scripting*, folhas de estilos, facilidades para impressão e documentos mais acessíveis (W3C, 1998).

Em seguida surgiu a especificação HTML 4.01, publicada no dia 24 de dezembro de 1999. Esta é definida como uma sub-versão da HTML 4.0 que inclui as alterações não-editoriais verificadas após a versão de 24 abril de 1998. A título de exemplo, foram efetuadas algumas mudanças nas *Document Type Definitions* (DTDs). Este documento torna as versões HTML 4.0 e anteriores obsoletas e ultrapassadas (W3C, 1999).

Esta especificação, se tornou padrão ISO/IEC 15445, como resultado do trabalho do consórcio W3C junto ao comitê ISO/IEC JTC 1 de junção técnica e tecnologia da informação, e também uma subcomissão denominada SC34 de línguas de descrição. A primeira publicação desta norma foi publicada no dia 15 de maio do ano 2000, sendo que este padrão internacional faz a referência normativa à recomendação de W3C para HTML 4.01 (ISO/IEC 15445, 2000).

A W3C determinou que a especificação HTML 4.01 será a última aprovada e nenhuma funcionalidade será adicionada a linguagem. A intenção é que a XHTML seja a sucessora da linguagem (MACEDO, 2004).

Conforme Valentine, Minnick (2001) a HTML 4.01 representa a iteração final como linguagem independente, pois todos os desenvolvimentos futuros da linguagem serão realizados sobre a bandeira da XHTML. Esta que é apenas uma reformulação da HTML 4.01 sob as regras da XML.

Nesta transição da HTML para XHTML, é apresentada uma grande mudança em relação aos conceitos de conteúdo, estrutura e apresentação. Porém o entendimento só será possível após uma introdução sobre a XML e abordagem das recomendações da XHTML com algumas técnicas exemplificadas. Antes da realização desta conceituação é importante o conhecimento de um termo mencionado anteriormente, que são as *Document Type Definitions* (DTDs).

Uma definição de tipo de documento (DTD) é um conjunto de regras legíveis por máquina que define o que é permitido ou proibido em uma versão particular da HTML, XML ou XHTML. Os agentes de usuários ou navegadores utilizam essas regras ao analisar sintaticamente uma página *Web* para verificar a validade da marcação utilizada e agir correspondentemente (BUDD; MOLL; COLLISON, 2006).

A declaração *doctype*, é de extrema importância na utilização das recomendações, pois é por elas que os navegadores determinam o tipo de visualização mais indicada para o site em questão. Uma DTD é uma especificação das regras que definem um formato. Ela define os elementos, atributos e valores permitidos, além das condições de encadeamento e sua ocorrência (MACEDO, 2004).

Assim, o conjunto de todas as marcações sintáticas que descrevem os dados e comandos para manipulação de documentos (*tags)*, utilizadas pelas derivações da SGML é chamado de *Document Type Definition* (DTD) (ALVARENGA; SOUZA, 2004).

Uma inovação como a possibilidade de ativar ou desativar as recomendações W3C surgiu com a idéia do tecnólogo de interface de usuário *Todd Fahrner*, um colaborador do grupo de trabalho da CSS e da HTML e co-criador do projeto WASP. Ele propôs que os criadores de navegadores incorporassem um mecanismo de troca capaz de alternar entre a apresentação compatível com as especificações e a não compatível. Sua sugestão era de que a presença ou ausência de uma *doctype* fosse usada como uma opção *on/off* (ZELDMAN, 2003).

Esta pequena sugestão também representou significativos passos para assegurar a retrocompatibilidade, pois os navegadores criaram dois modos de exibição, ou renderização: o modo conforme a especificação (*standards mode*) e o modo não-conforme (*quirks mode*), por vezes denominado idiossincrático. No modo conforme as recomendações, o *browser* renderiza as páginas de acordo com as especificações e no modo não - conforme, as páginas são exibidas de maneira mais vaga, emulando os navegadores mais antigos como *Internet Explorer 4* e o *Navigator 4*, afim de evitar que sites mais antigos tenham problemas. Os navegadores *Mozilla*® e *Safari*® possuem um terceiro modo chamado quase conforme, que é igual ao conforme, exceto por algumas

diferenças sutis na maneira em que as tabelas são tratadas (BUDD; MOLL; COLLISON, 2006).

Desta forma, para controlar qual a conduta que o navegador assumirá, deve ser incluído ou omitido um *doctype* completo. Os browsers que utilizam esta opção procuram por DTDs completos, isto quer dizer que eles buscam por estes elementos que incluem um endereço da *Web* com a URL da definição, conforme exemplo: *http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd* (ZELDMAN, 2003).

Tanto a linguagem HTML como a XHTML, especificam três tipos diferentes de DTDs conforme descrição:

a) *Transitional*: utilizada como método de transição que permite o documento utilizar alguns recursos considerados obsoletos, mas ainda úteis por questão de compatibilidade com versões antigas.
b) *Strict*: contém a definição formal da especificação e a maneira como ela deve ser implementada, esta definição exclui todos os elementos considerados obsoletos;
c) *Frameset*: É idêntica a *Transitional*, mas com os elementos necessários para o suporte a *frames* (MACEDO, 2004).

Assim, o navegador escolhe qual método de renderização utilizar com base na existência de uma declaração DTD. Com isto se um documento XHTML possuir uma definição de documento completa de qualquer tipo, *transitional*, *strict* ou *frameset*, os navegadores iniciarão no modo conforme as recomendações (BUDD; MOLL; COLLISON, 2006).

Conforme Zeldman (2003), definições incompletas, as quais não possuem o endereço da especificação na *Web*, fazem com que o *browser* seja iniciado em modo

idiossincrático, compatível com versões anteriores em vez do modo conforme as recomendações, desejado.

O efeito de escolher um modo de renderização com base na existência de um DTD é conhecido como alternância (*switching*) ou (*sniffing*) de *doctype*. A alternância de definição de tipo de documento é um *hack* (truque de programação) utilizado pelos navegadores para distinguir documentos herdados daqueles mais compatíveis com as especificações. Deste modo, mesmo escrevendo a apresentação com folhas de estilo CSS válidas, se a declaração de uma DTD for inadequada ou errada, as páginas serão exibidas no modo não - conforme apresentando comportamento ruim e inadequado. Um exemplo de definição completa é: *<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">* (BUDD; MOLL; COLLISON, 2006).

Um cuidado deve ser tomado quanto às declarações *doctype*, pois muitas ferramentas de criação inserem DTDs as quais tiram do site do W3C, porém de forma errônea estão publicando declarações incompletas e iniciando o processo de renderização de forma indesejada. Um exemplo de definição do tipo de documento incompleta normalmente inserida por estas ferramentas é: *<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "DTD/xhtml1-strict.dtd">* (ZELDMAN, 2003).

Como se pode observar o último exemplo de declaração apresentado por Zeldman, difere do modelo apresentado anteriormente a ele somente na última parte, onde a segunda opção trás consigo apenas um *link* relativo em vez de uma URI completa.

Conforme Zeldman (2003) a definição incompleta apresentada, é válida somente no site do W3C, pois realmente possui um *link* relativo interno do *site* do consórcio. Para utilizar este tipo de declaração, os desenvolvedores deveriam copiar o

DTD do W3C e colocá-lo dentro de um diretório no seu próprio servidor, porém ninguém o faz, e desta maneira o *link* relativo estará apontando para algo inexistente e a exibição ocorre no modo idiossincrático.

Com base no intuito da pesquisa sobre as especificações *Web*, é relevante apresentar a importância da utilização correta de uma DTD, porém existem alguns detalhes não abordados e maiores informações sobre as *doctypes* podem ser adquiridas com pesquisa nas referencias bibliográficas. Neste momento torna-se interessante a apresentação de alguns conceitos sobre as ferramentas de criação.

Atualmente na *Web*, os documentos são estruturados levando em conta principalmente a apresentação, utilizando de marcações muitas vezes geradas por ferramentas automatizadas que na maioria das vezes não levam em conta a sintaxe quanto menos a semântica (TIEGS; MENNA, 2006).

Construir aplicativos por meio destas ferramentas, normalmente conduz a preocupação somente com *layout* da aplicação, esquecendo das recomendações de código fonte limpo e organizado. Estas ferramentas acabam gerando códigos inúteis como, por exemplo, espaços em branco desnecessários para conseguir efeitos desejados. Desta maneira, as especificações são esquecidas tornando os sites poucos funcionais em relação à *browsers* e dispositivos diferenciados.

Porém, provavelmente estas ferramentas em breve estarão oferecendo melhor suporte as recomendações, pois o WASP começou a trabalhar junto aos desenvolvedores destas *cases* de produção, incluindo a *Macromedia*. Logo ao utilizar estas ferramentas, ao preocupar-se com a apresentação o desenvolvedor estará criando códigos válidos involuntariamente (ZELDMAN, 2003).

Contudo, o desenvolvedor deve estar sempre atento aos propósitos das recomendações e utilizar as especificações do consórcio mais atualizadas, a ficar

esperando por ferramentas de criação automatizadas. Tais propósitos como a separação da estrutura, apresentação e comportamento do *Trio das Especificações Web*, o qual nesta seção está sendo abordado às questões das linguagens de marcação estrutural como HTML, XML e XHTML, estas duas últimas que serão contextualizadas a seguir.

O HTML embora projetado para ser uma linguagem de marcação (que descreve seu conteúdo independente da exibição), tornou-se uma linguagem baseada em *tags* que os desenvolvedores utilizam para orientar a aparência e disposição, ou seja, da exibição da informação. Portanto existem muitos aspectos além da aparência quando se trata de disseminação de dados e a linguagem não é a ferramenta apropriada para isso (VALENTINE; MINNICK, 2001).

Assim, a *eXtensible Markup Language* (XML), foi desenvolvida para solucionar as limitações da HTML, sendo uma linguagem de marcação para captura, descrição, processamento e publicação de informações em diferentes tipos de mídias, sendo também um subconjunto da SGML (MACEDO, 2004).

A XML é muito parecida com a HTML, pois ambas são derivadas da SGML. Como na linguagem de hipertexto, nesta os elementos são denominados de *tags* e possuem marcas para início e fim do bloco de abrangência. As informações contidas entre as marcas são denominadas de conteúdo, entretanto, diferente da HTML a linguagem permite o uso de um conjunto indeterminado de marcas que indicam o significado dos dados e não a aparência dos mesmos (PITTS-MOULTIS; KIRK, 2000).

Esta linguagem é uma evolução da HTML, pois ela não tem característica apenas de linguagem de marcação, mas também de conotação semântica a documentos. Isto é, ela proporciona documentos com informação estruturada definindo claramente o conteúdo, significado e apresentação, facilitando recuperação de informações por meio da *Web* (ALMEIDA; VASCONCELOS, 2006).

Assim, a linguagem XML é um conjunto de regras e convenções de sintaxe que pode ser usada para criar conjuntos de elementos de marcação usados para descrever conteúdos. Ela foi desenvolvida porque a HTML não foi projetada para descrever os diferentes tipos de dados que as pessoas desejam enviar pela *Web*, como dados financeiros, normas de instalação de *software*, equações matemáticas entre outros dados (VALENTINE; MINNICK, 2001).

Desta forma, enquanto a HTML especifica o que cada *tag* e atributo significam e por vezes como seu conteúdo aparecerá no navegador, a XML utiliza as *tags* ou elementos, como melhor conceituadas, apenas para delimitar os trechos de dados, deixando a interpretação por parte da aplicação que os lê (MACEDO, 2004).

A título de exemplo, como a XML permite que usuários definam novas marcas para indicar estrutura, um elemento *pessoa* poderia ser criado para conter dados sobre pessoas. Desta forma, a especificação formal (DTD) possuiria as regra sobre os elementos pertencentes às pessoas, como *nome* e *idade*. Assim, seria formalizado o exemplo da seguinte maneira: *<pessoa> <nome> João </nome> <idade> 33 </idade> </pessoa>* (ABITEBOUL; BUNEMAN; SUCIU, 2000).

Esta linguagem, também pode ser definida como uma metalinguagem que integra funcionalidades selecionadas da SGML para que a partir dela possam ser criadas novas linguagens de marcação. Algumas linguagens principais que existem ou estão em desenvolvimento com base na XML podem ser relacionadas:

a) *Chemical Markup Language* (CML): linguagem de marcação para área química, com elementos para descrições moleculares;
b) *Mathematical Markup Language* (MathML): crida para publicar equações matemáticas na *Web*;

c) *Scalable Vector Graphics* (SVG): criada para descrever formas vetoriais, desenhos e gráficos bidimensionais;

d) *Web Services Description Language* (WSDL): é uma linguagem desenvolvida pela *IBM* em conjunto com a *Microsoft* que utiliza o formato XML para *Web services*;

e) *eXtensible HyperText Markup Language* (XHTML): reformulação da HTML 4.01 baseada nas regras da XML (MACEDO, 2004).

Conforme Zeldman (2003) outras linguagens de importância foram desenvolvidas baseadas na XML como a *Extensible Stylesheet Language Transformations* (XSLT), a *Resource Description Framework* (RDF), e ainda um vocabulário superficial para publicação de conteúdos, denominado *Rich Site Summary* (RSS).

Estas linguagens apresentadas podem ser definidas como aplicativos da XML e classificadas em sete categorias, sendo: para *Web* e Internet, metadados e de arquivamento, multimídia e aplicativos de reconhecimento de voz, orientados as áreas de negócios como finanças e comércio, científicos, educacionais, e voltados a linguagens. Desta forma, pode-se definir o RDF como um aplicativo de metadados e arquivamento, MathML e CML como científicos e assim por diante. Existem ainda muitos outros aplicativos não mencionados baseados na XML, como o *Wireless Application Protocol* (WAP), classificado como para *Web* e Internet (PITTS-MOULTIS; KIRK, 2000).

Assim, para não perder o foco da pesquisa, devem-se apresentar claramente onde as recomendações se encaixam em meio a tudo isso. A título de explicação, em 1996, foi criada a linguagem XML, no início como uma versão simplificada de sua

ascendente SGML e em fevereiro de 1998, tornou-se uma especificação formal, reconhecida pelo W3C (ALMEIDA, 2002).

A especificação XML 1.0, foi publicada pelo consórcio W3C no dia 10 de fevereiro de 1998, sendo que ela foi desenvolvida por um *XML Working Group*, formado em 1996 sob os auspícios do consórcio de padrões. Esta recomendação especifica uma sintaxe criada pela sub-definição de um padrão de processamento de textos internacional, o SGML (W3C, 1998).

Esta especificação, juntamente com especificações associadas (*Unicode and ISO/IEC 10646* para caracteres, *Internet RFC1766* para *tags* de identificação da linguagem, *ISO 639* para códigos de nome de linguagem e *ISO 3166* para códigos de nome de país), oferece o necessário para a compreensão da XML 1.0 (HOLZNER, 2001).

O consórcio de padrões W3C, também publica outras recomendações para linguagens baseadas na XML, como a especificação *MathML 2.0* publicada em 21 de outubro de 2003, a *RDF/XML Syntax Specification* e a *RDF Semantics* publicadas em 10 de fevereiro de 2004 e outras. Além de publicar as recomendações para as diversas derivações da XML o W3C as mantém atualizadas, por meio de comitês que pesquisam e constantemente publicam novas especificações (PITTS-MOULTIS; KIRK, 2000).

Segundo Holzner (2001) uma recomendação do consórcio para XML, é que os documentos iniciem com um prólogo, e que neste esteja incluído pelo menos a declaração que indique a versão da linguagem utilizada. Em geral estes prólogos podem conter declaração XML, comentários, instruções de processamento, e DTDs.

Uma declaração XML, simplesmente informa que o documento é XML e identifica sua versão. Um exemplo de declaração é: *<?xml version=”1.0”?>*. Estas declarações também podem ser utilizadas para incluir informações de codificação. Caso

a utilização de codificação seja diferente de ASCII, UTF-8 ou UTF-16, ela deverá ser declarada por meio do prólogo, conforme exemplo: *<?xml version=”1.0” encoding=”ISO-8859-1”?>* (VALENTINE; MINNICK, 2001).

Conforme Holzner (2001) uma das maiores aplicações XML existentes é a XHTML, como anteriormente mencionado, é a tradução da HTML 4.01 para XML desenvolvida pelo W3C.

Assim, este trabalho poderia abranger muito mais dos recursos da XML e de suas derivações, apresentando também as especificações formais do consórcio e exemplificar os recursos de cada linguagem. Porém, optou-se pela apresentação das regras da XML, diretamente sob as mudanças e regras que esta impõe a HTML, na denominada XHTML, pois esta é uma das principais recomendações adotadas nos conceitos de especificações do W3C.

Desta maneira, um dos passos em direção a utilização das especificações é a utilização da linguagem XHTML, que é a grande mudança da HTML desde que a versão 4.01 foi lançada. A recomendação desta linguagem traz o rigor da XML às páginas da *Web* sendo reconhecido como o marco no trabalho da W3C para criar especificações que ofereçam sites mais ricos em uma faixa cada vez maior de plataformas e *browsers* (VALENTINE; MINNICK, 2001).

Conforme a W3C (2000) esta linguagem é o caminho para evolução da Internet. Migrando para XHTML os desenvolvedores de conteúdo estarão entrando no mundo XML com todos seus benefícios assegurando para seus conteúdos a compatibilidade com aplicações passadas e futuras.

Tais benefícios oferecidos pela XML como a portabilidade das aplicações, oferecem a possibilidade de utilização por meio de telefones celulares, televisores,

comunicadores sem fio de pequeno tamanho e *desktops* (VALENTINE; MINNICK, 2001).

Desta forma, a XHTML traz grandes vantagens em termos de acessibilidade já que é uma linguagem independente de dispositivos. Levando em consideração o aumento das plataformas alternativas para acesso a Internet, tais como computadores portáteis, televisores, telefones, agendas eletrônicas, entre outros, é fundamental a existência de uma linguagem com estas características. Em termo de sintaxe, a XHTML não é tolerante como a HTML, pois ela utiliza as regras rígidas da XML na aplicação de *tags* em um documento (MACEDO, 2004).

Conforme Velentine, Minnick (2001) a XHTML é apenas uma reformulação da HTML 4.01 de acordo com as regras da XML. Abaixo uma lista de algumas mudanças desta reformulação:

a) os documentos desta nova linguagem precisam adaptar-se a regras mais estritas como a colocação de aspas ou apóstrofos sempre ao redor dos valores de atributo e o aninhamento correto de elementos;

b) o XML é *case-sensitive*, ou seja, diferencia letras maiúsculas e minúsculas. Portanto, XHTML também o faz, assim os elementos e atributos devem ser escritos em letras minúsculas;

c) como os documentos XML são elaborados separando o conteúdo da marcação, alguns elementos de formatação como a *tag font* é passado para folhas de estilo, que controla a exibição dos seus documentos;

Uma outra mudança quanto a sua antecessora, é quanto as *tags* de fechamento. Na HTML em algumas etiquetas é opcional, na XHTML conforme o rigor da XML é obrigatório em todas as marcas. Também nas tags de quebra de linha como <*br*> e inserção de imagens <*img*>, o fechamento deve ser declarado, ficando da

seguinte forma: <*br* />, <*img* />, sendo necessário a adição de um espaço seguido de uma barra (REBITTE; VINICIUS, 2006).

Uma observação importante é de que os elementos e atributos, como mencionado anteriormente, devem ser escritos todos com letras minúsculas. Porém para os valores e conteúdos dos atributos é indiferente (ZELDMAN, 2003).

Assim, quanto aos atributos, seus valores *id* não devem se repetir em uma mesma página. Deve-se também utilizar este atributo no lugar do atributo *name*, este último que está em desuso e não estará mais disponível nas próximas versões do XHTML (REBITTE; VINICIUS, 2006).

Conforme Valentine, Minnick (2001) os documentos XHTML são arquivos XML totalmente qualificados e desta forma, a inclusão do prólogo XML deve ser realizado.

Muitas páginas XHTML começam com o prólogo opcional, também conhecido como declaração XML. Quando esta declaração aparece, ela precede a declaração *doctype*, e sua missão é informar sua codificação de caracteres e a versão da linguagem. Porém, muitos navegadores atuais não podem lidar com o prólogo XML. Após assimilarem esta declaração os navegadores travam ou exibem a aplicação incorretamente, esta última ocorre quando o IE6/Windows encontra a declaração. Felizmente, há uma solução, no lugar do prólogo, pode ser especificada a codificação por meio da *tag* <*meta*> (ZELDMAN, 2003).

A metodologia utilizada para apresentação um pouco mais avançada da linguagem XHTML será apresentar os principais elementos que compõe a linguagem, mostrando a forma correta de sua utilização de acordo com as especificações do consórcio de padrões.

O elemento *html* é indicado com o uso das *tags* <*html*> e </*html*> e indicam respectivamente o início e o fim do documento. Elas também especificam que o tipo de documento é HTML/XHTML e devem englobar todas as demais *tags* existentes. Nesta nova linguagem, é necessário acrescentar um atributo a *tag* inicial conforme exemplo: <*html xmlns=”http://www.w3.org/1999/xhtml”*>. O atributo *xmlns* (*xml namespace*) indica que o namespace primário utilizado é o DTD XHTML. O *namespace* é uma das partes que compõe uma definição, a qual especifica ao endereço das regras (MACEDO, 2004).

O elemento *head* é um elemento de nível superior presente em todos os documentos XHTML e possui informações como título, informações meta, *links* com outros documentos e informações de índice. Para isto, eles podem conter os elementos *link, meta, script, style, title* (VALENTINE; MINNICK, 2001).

O elemento *meta* é utilizado para incorporar metainformações, ou seja, *informações* sobre as informações do documento. Essas são opcionais e utilizadas para identificação, indexação e catalogação do *site*. Com esta *tag*, é possível fornecer informações sobre quem criou o documento, quando foi modificado, sua descrição e outras. Ela é composta por duas partes, identificador e conteúdo. O identificador de *tag* é referenciado na parte *http-equiv* ou *name* e seu conteúdo, ou valor, pela parte *content* (MACEDO, 2004).

Conforme Rebitte, Vinicius (2006) *meta tags* são etiquetas eletrônicas utilizadas para disponibilizar informações adicionais para facilitar a vida dos indexadores de páginas *engines*. A intenção é fornecer algo como o que é o *site*, quais palavras chaves sobre ele e de sobre o intervalo de tempo desejado para retorno dos *bots* no *site*. Abaixo uma listagem de alguns exemplos da utilização de *meta tags* para:

a) identificar o site por meio de um título: *<meta name="title" content="Título do site podendo conter descrição pequena" />*;

b) fornecer uma descrição do documento: *<meta name="description" content="Descrição maior porém breve sobre o documento" />*;

c) especificar as palavras chaves: *<meta name="keywords" content="Palavras chaves separadas por espaços em branco" />*;

d) identificar autor: *<meta name="author" content="Nome do Autor" />*;

e) especificar identificar o conjunto de caracteres: *<meta http-equiv="content-type" content="text/html" charset="iso-8859-1" />*.

Segundo Macedo (2004) as *meta tags* podem também enviar instruções especiais aos navegadores, como especificar um intervalo em segundos para recarregar a página (*refresh*) ou mesmo carregar outro documento. Um exemplo desta utilização é: *<meta http-equiv="refresh" content="5;url=http://www.site.com.br/index2.html" />*. O uso do atributo *http-equiv*, é usado para chamadas correspondentes ao protocolo HTTP, enquanto o uso do *name* é empregado para outros tipos de controle que não correspondem a estas chamadas.

As máquinas de busca, robôs ou *search engines* são algoritmos (funções) as quais cada uma possui seus critérios. Todos os sites de busca possuem seus respectivos *bots*. O *Google*®, ferramenta mais utilizada atualmente possui o *Googlebot*. Estes robôs variam de dois a três dias para voltar ao site, tempo que pode ser definido por meio de *meta tags* (REBITTE; VINICIUS, 2006).

Existem materiais específicos sobre o funcionamento dos motores de busca, o qual o entendimento de cada robô em específico é uma tarefa complicada. Alguns *sites* de busca fornecem informações de otimização para seus respectivos *bots* aos *webmasters* (criadores de conteúdo *Web*). Porém a maioria dos algoritmos destes *bots*

levam em consideração a semântica (código XHTML) e separação do *design* e conteúdo (CSS) (REBITTE; VINICIUS, 2006).

Voltando aos elementos da XHTML, o *link* é utilizado para definir vínculos com outros documentos ou recursos. Seus principais atributos são:

a) *href*: especifica a localização do recurso por meio de uma URI;
b) *hreflang*: especifica o idioma de base do documento ou recurso designado pelo atributo *href*;
c) *charset*: especifica o conjunto de caracteres do documento ou recurso;
d) *type*: especifica o tipo de conteúdo do documento ou recurso disponível no destino do vínculo. Alguns tipos de conteúdos são: *text/plain*, *text/html*, *text/css*, *text/xml*, *application/pdf*, *text/javascript* e outros;
e) *rel*: descreve a relação do documento com o vínculo;
f) *media*: especifica o meio de destino para o documento ou recurso especificado (MACEDO, 2004).

Os dois últimos atributos merecem maior atenção devido sua relevância quanto as especificações *Web*, pois é por meio deles que é especificado o tipo de mídia destinada e a relação entre os documentos, podendo ser utilizado em conjunto com outros atributos para adquirir a funcionalidade desejada.

Como mencionado anteriormente, o atributo *rel*, descreve a relação do documento com o destino especificado, e alguns valores possíveis muito úteis são:

a) *alternate*: especifica versões alternativas para o documento ou recurso vinculado. Quando utilizado junto ao atributo *lang*, implica em uma versão traduzida do documento e quando junto do atributo *media*, corresponde a uma versão designada para um tipo de mídia diferente;

b) *stylesheet*: refere-se a uma folha de estilos externa e quando utilizada junto ao atributo *alternate*, possibilita folhas de estilos alternativas como opções para o usuário (MACEDO, 2004).

Atualmente a Internet está presente em muitos tipos distintos de dispositivos. Um *site* pode ser apresentado por meio de um monitor de computador, celular, navegador para deficientes visuais ou mesmo impresso. A estrutura fornecida pela XHTML permite isso sem necessidade de criação de várias versões. Porém, para melhoria do suporte básico de acessibilidade a CSS, permite criar códigos pequenos e específicos para cada dispositivo. A inserção adequada na XHTML é realizada com a declaração das folhas de estilos alternativas, juntamente com a declaração do tipo de mídia por meio do atributo *media* (REBITTE; VINICIUS, 2006).

Este atributo possui diversos valores específicos para declaração de cada tipo de saída, como:

a) *screen*: apropriado para monitores de computadores. Este é o valor padrão do atributo;
b) *tty*: apropriado para teletipos, terminais ou dispositivos portáteis com capacidades limitadas;
c) *projection*: apropriada para projetores;
d) *handheld*: Apropriado para dispositivos de mão;
e) *print*: apropriado para materiais destinados a impressão, paginado e opaco;
f) *braille*: apropriado para dispositivos braile;
g) *aural*: apropriado para sintetizadores de voz (MACEDO, 2004).

Estas possibilidades servem para suprirem determinadas ocasiões ou necessidades de determinados usuários. Embora seja um assunto referente à CSS, é

tratado na seção de XHTML, devido os elementos e atributos utilizados para inserção destes estilos nos documentos. Desta forma, também já serve como introdução a próxima seção sobre as folhas de estilos.

Segundo Rebitte, Vinicius (2006) quando um arquivo *.css* é inserido em um documento, ele pode ser relacionado a este, de três formas: persistente, preferido ou alternativo. Uma folha de estilo persistente está sempre ativa no documento e compartilha suas regras com as demais, caso exista. A declaração é realizada por meio do elemento *link*. As folhas alternativas também estão sempre ativas, compartilhando suas regras e sua declaração é realizada da mesma forma que as folhas persistentes, porém deve-se atribuir um título igual a todas as folhas preferidas por meio do atributo *title*. As folhas alternativas estão por padrão desabilitadas, contudo podem ser selecionadas pelos usuários e sua declaração deve seguir o seguinte exemplo: *<link rel=”alternate stylesheet” type=”text/css” title=”Fontes Maiores” href=”fonte_maior.css” />*.

Conforme Zeldman (2003) as folhas de estilos podem ser aplicadas a uma página *Web* HTML/XHTML, de três formas: externas, embutidas e *in-line*.

A folha de estilos externa é apenas um arquivo de texto com extensão *.css*, onde ficam contidas as declarações das regras de estilos. Este arquivo é vinculado a uma ou mais páginas HTML/XHTML, por meio da *tag <link />* que deve ser escrita dentro do cabeçalho da página, ou seja, dentro da marcação *<head></head>*. Pode-se também inserir uma folha do tipo externa por meio da regra *@import*, atributo que funciona na maioria dos navegadores novos (REBITTE; VINICIUS, 2006).

Em vez de utilizar uma folha de estilos separada e inserir por meio de *links* ou importação, o *designer* pode embutir as regras da folha de estilos no elemento *head* de uma página XHTML, usando o elemento *style* como a seguinte sintaxe: *<style*

type=”text/css”><!-- Regras --></style>. Ao contrário dos estilos com *links* ou importados, os estilos embutidos não oferecem nenhum beneficio de largura de banda, pois o usuário deve carregar uma nova folha de estilo embutida sempre que abrir uma nova página (ZELDMAN, 2003).

Conforme Macedo (2004) o elemento *style* é utilizado para inserir formatação de estilos diretamente a página, levando a denominação de estilos embutidos ou incorporados. Este elemento possui dois atributos: *type* e *media*, os quais já foram apresentados.

A terceira maneira de aplicar estilos é a *in-line*, especificando individualmente aos elementos conforme a seguinte sintaxe: *<h1 style=“font-family: verdana, arial, sans-serif;”>Título</h1>*. Como se pode observar que esta forma de aplicar estilos não economiza largura de banda, na verdade aumenta a demanda. Este método é tão perdulário quanto à *tag <font>* (ZELDMAN, 2003).

O elemento *script* é importante e muito utilizado, ele possui as *tags <script>* e *</script>*, que são utilizadas para inserção linguagens de *scripts* no documento. Um *script* é um programa que acompanha ou está vinculado ao documento e permite estender as funcionalidades da linguagem, tornando estes mais interativos. Um *script* pode ser definido dentro de um documento entre as *tags head*, *body* ou em um arquivo externo, especificando por meio dos atributos *type* e *src* (MACEDO, 2004).

Um elemento para definição do título do documento é o *title*, este que precisa ser um filho do elemento *head*. Normalmente esta definição é exibida no topo da janela do *browser* e também utilizada nas listas criadas por usuários, como *bookmarks* ou favoritos (VALENTINE; MINNICK, 2001).

O corpo do documento pode conter parágrafos, listas, tabelas, *links*, imagens, formulários, animações, vídeos, sons e *scripts*. Para especificar o corpo no

documento para inserção destes conteúdos, deve-se utilizar as *tags* <*body*> e <*/body*> (MACEDO, 2004).

Desta forma, serão apresentados os elementos comumente utilizados na produção de conteúdos para as páginas. Como um dos objetivos relevante deste trabalho são as especificações, apenas serão apresentados os principais elementos em conformidade com a recomendação XHTML 1.1 e em seguida apresentado a recomendação do W3C para formatação visual.

Conforme Macedo (2004) a maioria dos elementos utilizados possuem atributos em comum, como:

a) *id*: identifica um elemento, atribuindo a ele um nome, este que deve ser único em todo o documento. Geralmente associado ao uso de folhas de estilos;
b) *class*: identifica que o elemento pertence a uma classe específica. Também associado a utilização de CSS;
c) *title*: define um título para o elemento e sua apresentação pode variar de acordo com navegadores diferentes;
d) *dir*: indica a direção em que o elemento deve ser lido. Possíveis valores são: *ltr* (*left to right* – da esquerda para direita) e *rtl* (*right to left* –da direita para esquerda);
e) *lang*: define o idioma no qual o conteúdo do elemento foi escrito.

Assim, um elemento utilizado para declaração de estrutura é o *div*. Conhecido como este nome, devido à abreviatura de divisão este elemento serve para estruturar o documento. O emprego das especificações impõe a utilização destes elementos para formação de estrutura, ao invés do emprego de tabelas para diagramação de *layout* (ZELDMAN, 2003).

As *tags* <*div*> e </*div*> permitem a divisão em blocos de um documento, criando estrutura lógica. Porém, essas etiquetas não determinam nenhum tipo de formatação ao conteúdo, mas associadas a uma folha de estilos CSS podem aplicar varias regras sobre todos os elementos que fazem parte do bloco. As *tags* <*span*> e </*span*> são similares, porém utilizadas em nível de linha. Estes dois elementos possuem os seguintes atributos: *id*, *class*, *style*, *title*, *dir* e *lang* (MACEDO, 2004).

Para identificação dos níveis das manchetes ou títulos nas páginas, são utilizados na XHTML os elementos *h1*, *h2*, *h3*, *h4*, *h5* e *h6*. Estes devem ser utilizados para identificação de estrutura e não apenas como ferramentas de formatação. O *h1* especifica a maior hierarquia, sendo que os demais são definidos como subtítulos uns dos outros. Assim como outros elementos estes também possuem os atributos *id*, *class* e *style*, que permitem a formatação visual por meio de folhas de estilos (VALENTINE; MINNICK, 2001).

Desta maneira, um exemplo de boas práticas na programação *Web*, é a utilização para um cabeçalho de seção de nível dois a *tag* <*h2*>, ou seja, o nível mais elevado de subcabeçalho abaixo do nível um que normalmente é usado para a página inteira. Esta técnica de codificação é denominada de semântica, esta que foi temporariamente perdida quando alguns fornecedores de *browsers* introduziram *tags* patenteadas para transmissão das informações. Muitos *webdesigners*, por exemplo, utilizam algo como *Garamond negrito caixa alta 18 pixels*, quando poderiam utilizar somente o cabeçalho de nível dois associado a uma folha de estilos (NIELSEN, 2000).

A correta utilização da XHTML garante também ganho em questão de acessibilidade. Um exemplo disto é a aplicação correta das *tags* para definição de textos, que atualmente são muito pouco utilizadas. As *tags* para estruturação dos textos são pouco conhecidas, mas quando utilizadas auxiliam na apresentação, compreensão e

acessibilidade. Abaixo segue algumas *tags* semânticas para definição dos textos em um documento seguindo as especificações W3C:

a) <*p*>...</*p*>: define um bloco de texto como um parágrafo. Uma quebra de linha ocorre antes e depois do elemento;

b) <*abbr*>...</*abbr*>: indica que o texto entre estas etiquetas, é uma abreviação;

c) <*acronym*>...</*acronym*>: indica que o texto incluído é um acrônimo, ou seja, uma abreviatura que utiliza a primeira letra de cada palavra;

d) <*blockquote*>...</*blockquote*>: indica que o texto incluído é uma citação de outro documento. Deve-se utilizar juntamente com o atributo, *cite*, para especificar o endereço do documento original;

e) <*cite*>...</*cite*>: indica uma citação ou referência a uma fonte externa de documentos;

f) <*code*>...</*code*>: indica que o texto incluído é um trecho de código-fonte de alguma linguagem de programação (MACEDO, 2004).

Um outro bom exemplo de utilização correta das *tags* da XHTML é a utilização de listas. Essas podem ser usadas em diferentes situações, como para organização de *menus* de opções com *links* para o usuário. As *tags ul* (lista não ordenada), *ol* (lista ordenada) e *dl* (lista de termos e definições), podem ser utilizadas para agrupar elementos ou itens de listas. Os elementos *ul* e *ol*, possuem como filhos o elemento *li* que especifica um item de lista, enquanto o elemento *dl*, possui dois elementos, *dd* (sub-lista de termos e definições) e *dt* (termo ou definição de uma *dl*). Desta forma, deve-se utilizar elementos de uma lista não ordenada *ul* para concepção de *menus* em sites convencionais (VALENTINE; MINNICK, 2001).

As tabelas também são itens que devem ser corretamente estruturadas, com a utilização dos elementos *caption*, para definir um título para tabela, *th* para definição de células de cabeçalho e *td* para especificação de células de conteúdo. Outros elementos para melhor organização semântica de tabelas podem ser utilizados como *thead*, *tfoot* e *tbody*, que separam os elementos de definição de cabeçalho, rodapé e corpo da tabela. Também é recomendável oferecer resumos das tabelas por meio da *tag summary* (MACEDO, 2004).

Muitos outros elementos existem na linguagem XHTML, porém a grande importância para estes é a correta utilização para realmente cumprir com o propósito desta, estruturar o documento de forma que seja entendível por qualquer pessoa ou dispositivo. Elementos como *fieldset*, *optgroup*, *alt*, *label* os quais ainda não foram apresentados, mas representam contribuição para formatação semântica. Alguns destes elementos serão apresentados, conforme sua utilização na construção da interface *Web* de gerenciamento de *Firewall* (ZELDMAN, 2003).

Assim, pode-se utilizar esta linguagem para o desenvolviemento de aplicações para Internet, em conjunto com outras ferramentas para tornar o objeto acessível e válido. Quanto a parte visual da aplicação utiliza-se as folhas de estilos já mencionadas. Porém estas serão melhor apresentadas a seguir.

3.3.3. *Cascading Style Sheets* (CSS)

Uma linguagem simples e de fácil aprendizagem, para criação de *layouts* para *Web*, é a *Cascading Style Sheets* (CSS), que pode ser traduzida para Folhas de Estilo em Cascata. Seu objetivo, é substituir as marcações HTML de formatação,

separando a camada de apresentação (CSS) e a camada de informação (XHTML ou HTML estrutural) (REBITTE; VINICIUS, 2006).

Estas folhas de estilo estão sendo utilizadas no intuito de recapturar o ideal da *Web*, que é separar apresentação e conteúdo. Como a Internet é acessada por meio de sistemas multiplataformas, seu conteúdo será apresentado em uma variedade de dispositivos. Assim as páginas devem especificar o significado das informações e deixar os detalhes da apresentação a intercalação de folhas de estilos (NIELSEN, 2000).

Conforme Rebitte, Vinicius (2006) separando a formatação do conteúdo, a manutenção torna-se mais fácil, pois alterações no *layout* podem ser feitas sem a necessidade de alterar o documento XHTML. Esta separação torna o site mais acessível, pois caso as formatações estejam atrapalhando a visibilidade ou acessibilidade para algum usuário em especial, este poderá desabilitar os estilos por meio do *browser*.

Desta forma, a especificação CSS 1 foi lançada em 1996, alguns meses mais tarde o IE3 estreou incluindo suporte a especificação, rudimentar entre seus recursos. Este suporte que estava totalmente ausente no Netscape Navigator 3 ofereceu ao navegador da Microsoft maior credibilidade no momento em que o Navigator dominava a *Web*. Porém o IE3, suportava a CSS1 apenas o suficiente para permitir a troca das *tags* <*font*> (ZELDMAN, 2003).

A especificação CSS 2 foi publicada no dia 12 de maio de 1998, apresentando inovações em relação à especificação da CSS 1, possibilitando os desenvolvedores a criar apresentações para *browsers* visuais, dispositivos aurais, impressoras, dispositivos braile e dispositivos de mão. No dia 6 de novembro de 2006 o consórcio W3C, publicou como um esboço de trabalho uma revisão denominada de CSS 2.1 que corrigia alguns erros da especificação anterior, como uma nova definição

de altura/largura de elementos com posicionamento absoluto. A especificação CSS 3 está sob rascunhos do consórcio e ainda não foi publicada (W3C, 2006).

Assim, esta especificação de formatação para documentos HTML/XHTML, permite maior versatilidade no desenvolvimento de *design* para sites sem aumentar a demanda de tamanho. O CSS permite ao *designer* melhor controle sobre os atributos tipográficos de um *site*, como tamanho e cor das fontes, espaçamento entre linhas e caracteres, margem do texto e outros. Permite também a sobreposição de texto sobre texto ou imagens com a utilização de *layers* (camadas) (MACEDO, 2004).

Nesta seção do trabalho, poderia ser aplicado à mesma metodologia utilizada na seção sobre acessibilidade, especificando em detalhes as recomendações do consórcio sobre a tecnologia. Porém, se tornaria extenso, além de que as recomendações da W3C utilizam termos técnicos, afinal é a eles que são destinados estes documentos, assim tornaria difícil o entendimento. Deste modo, a metodologia utilizada, será a contextualização dos segmentos mais relevantes da linguagem de apresentação CSS.

O primeiro segmento, importante para o início do entendimento das folhas de estilo em cascata, é o entendimento sobre seletores, declarações, propriedades e valores. Uma folha de estilo consiste em uma ou mais regras que controlam o modo como elementos selecionados devem ser exibidos. Estas regras consistem de duas partes: um seletor e uma declaração. Conforme exemplo: *p {color: red;}* (ZELDMAN, 2003).

Segundo a regra precedente, *p* é o seletor, enquanto *{color: red;}*, delimitados por chaves, é a declaração. Por sua vez, as declarações também consistem de duas partes: uma propriedade e um valor, no exemplo anterior, *color* é a propriedade e *red* o valor (ZELDMAN, 2003).

Na CSS, existe a possibilidade de agrupamento de seletores que possuam declaração idêntica, evitando muitas linhas e sobrecarga do arquivo de extensão *.css*. Um exemplo é a aplicação da cor vermelha dos títulos *<h1>* até *<h6>*, possuindo a seguinte regra: *h1, h2, h3, h4, h5, h6 {color: #ff0000;}* (REBITTE; VINICIUS, 2006).

De acordo com a CSS, também existe a possibilidade de herança das propriedades do elemento de nível mais alto. Desta maneira, para aplicar um estilo de *font Verdana* em todo o documento, basta somente à declaração: *body {font-family: Verdana, sans-serif;}*. Sem a adição de nenhuma regra, todos os filhos do elemento *body* como *p*, *td*, *ul*, *ol*, *li*, *dl*, *dt* e *dd*, devem ser exibidos em *Verdana* ou em fonte *sans-serif* genérica, assim como os filhos de seus filhos (ZELDMAN, 2003).

Segundo Rebitte, Vinicius (2006) ainda possui subdivisões dos tipos de seletores, como os seletores de identificação. Este tipo de seletor pode ser associado a um único elemento HTML/XHTML, pois sua repetição em mais de um elemento da mesma página, pode causar problemas ao funcionamento das linguagens de *script*, como o W3C DOM que possa ser utilizado. Os seletores de identificação devem ser escritos precedidos do caractere # (cerquilha), seguindo a seguinte sintaxe: *#nome {propriedades;}*. A associação às páginas XHTML deve ser feita por meio do atributo *id* conforme exemplo: * * (REBITTE; VINICIUS, 2006).

Nesta subdivisão, enquadram-se também os seletores de tipos ou elementos, os quais já foram utilizados nos exemplos anteriores. Estes seletores são aqueles utilizados para aplicar estilo a elementos da linguagem como as *tags <p>*, *<a>*, *<h1>* entre outras (BUDD; MOLL; COLLISON, 2006).

Um outro tipo de seletor muito utilizado, é o de classe. Diferente do de identificação, este pode ser aplicado a vários elementos HTML/XHTML, muito úteis para aplicar a mesma formatação a vários elementos diferentes na página. Os seletores

de classe, devem ser escritos precedidos do caractere . (ponto), conforme a sintaxe: *.nome {propriedades;}*. Para aplicar este a uma marcação HTML, utiliza-se o atributo *class*, sendo que seu valor deve ser igual ao nome do seletor, sem o caractere . (ponto) (REBITTE; VINICIUS, 2006).

Conforme Zeldman (2003) os seletores de classe, identificação e de elementos, podem ser combinados para criar efeitos visuais sutis e surpreendentes. Além de que existem também os seletores avançados, como os seletores de atributos, este pouco utilizado e alguns navegadores atuais não oferecem suporte.

Além de todos os seletores mencionados, existem ainda dois seletores que podem ser utilizados com facilidade, como os seletores contextuais, os quais a regra só é aplicável se a sintaxe da página satisfazer a declaração do seletor e os seletores universais, cujas propriedades são aplicadas a todos os elementos da página (REBITTE; VINICIUS, 2006).

Estes seletores que inicialmente podem ocasionar um pouco de confusão, em uma folha de estilos moderadamente complicada, pode proporcionar que duas ou mais regras selecionem o mesmo elemento. Estes conflitos são tratados por meio de um processo conhecido como cascata. Neste processo é atribuído um grau de importância a cada regra. Desta forma, para fornecer maior controle ao usuário, eles podem designar prioridades a qualquer regra, especificando-a como *!important*, mesmo que seja uma já marcada como *!important* pelo autor. Caso duas regras possuam, mesmo grau de prioridade, a última definida terá precedência (BUDD; MOLL; COLLISON, 2006).

Desta maneira, a cascata funciona de acordo com a seguinte ordem: estilos do usuário marcados como *!important*, estilos do autor marcado como *!important*, estilos do autor, estilos do usuário, estilos aplicados pelo agente do usuário (navegador) (BUDD; MOLL; COLLISON, 2006).

Conforme Rebitte, Vinicius (2006), também se pode utilizar, pseudo-classes para aplicar efeitos especiais a um determinado seletor. Estas que não são elementos HTML/XHTML, mas efeitos aplicados a estes. A sintaxe é: *seletor:pseudo-classe {propriedade: valor;}*. Existem pseudo-classes para *links*, estas que foram incluídas na especificação CSS 1, conforme:

a) *link:link*: especifica formatação para links não visitados;
b) *link:visited*: especifica formatação para *links* já visitados;
c) *link:hover*: especifica formatação para *links*, os quais estão com o cursor do mouse sobre eles;
d) *link:active*: especifica formatação a *links* os quais já foram clicados.

Existe ainda outras pseudo-classes, como a *:first-child*, que aplica formatação ao primeiro elemento contido em outro elemento, *:focus*, que aplica formatação ao elemento enquanto ele está em foco e *:lang* que formata elementos de acordo com a linguagem (idioma) contido dentro do elemento. Porém algumas destas últimas técnicas apresentadas não funcionam em navegadores de versões mais antigas e apresentam erros (*bugs*) em alguns utilizados atualmente (REBITTE; VINICIUS, 2006).

Assim, ao encontrar problemas ou soluções para incompatibilidades ou *bugs* ou mesmo simplesmente ao conceber arquivos CSS, o programador deve documentar seus passos por meio de comentários ou notas pessoais, com os dados do autor e formas de contato e/ou possíveis problemas ou soluções encontradas. A CSS permite comentários no código, assim como muitas outras linguagens, porém eles podem aumentar consideravelmente seus arquivos (BUDD; MOLL; COLLISON, 2006).

Como mencionando sobre pseudo-classes, não se pode excluir os pseudo-elementos, que permitem formatar elementos da página que não sejam marcações. Desta forma, o pseudo-elemento *:first-line* permite formatar a primeira linha de um texto

contido em um elemento, bem como o *:first-letter*, permite aplicar formatação ao primeiro caractere. Outros pseudo-elementos, como o *:after* e *:bfore*, permitem gerar conteúdos antes e depois do conteúdo de um elemento HTML/XHTML. Porém estes assim como algumas pseudo-classes, não oferecem suporte a alguns navegadores (REBITTE; VINICIUS, 2006).

Segundo Rebitte, Vinicius (2006) a CSS é muito extensa, possuindo recursos sobre unidades de medida, uso de cores e ainda diversas propriedades para elementos como: *font*, *text*, *border*, *background*, *margin*, *padding*, *list* e muitos outros como as propriedades de posicionamento e apresentação.

Como o objetivo deste trabalho é a apresentação do funcionamento das especificações de forma a utilizar os mesmos em uma interface de sistema e não a explicação detalhada de cada elemento da CSS, alguns conceitos não foram apresentados, tais como os modelos de caixas, margens, flutuação e posicionamento sendo que tais conhecimentos podem ser adquiridos com pesquisa nas referencias bibliográficas apresentadas.

Desta forma, foram apresentados os assuntos mais relevantes como à função das folhas de estilos e suas especificações pelo consórcio W3C, conceito de seletores, pseudo-classes e pseudo-elementos e uma das técnicas avançadas como a ordem de prioridade para regras contraditórias (duas regras selecionarem o mesmo elemento). Também se pode observar que algumas técnicas da CSS, caso mal utilizadas podem apresentar poucas melhorias ou mesmo não apresentar, quanto à adoção das recomendações.

Conforme Rebitte, Vinicius (2006) um dos maiores problemas no desenvolvimento de *Web* sites que seguem as especificações, é a falta de suporte oferecido pelos navegadores. Desta forma o uso de *CSS Hacks* é uma das formas de

contornar este problema. Eles são apenas formas que permitem os desenvolvedores a escreverem uma regra ou propriedade que só será interpretada por determinado(s) *browser*(s).

Em um contexto ideal, CSS adequadamente codificadas funcionariam em todos os navegadores, mas infelizmente não se possui este contexto. A maioria dos navegadores utilizados atualmente, estão cheio de *bugs* e inconsistências. Assim, para realizar a apresentação desejada, em alguns casos pode-se requerer o uso de alguns macetes conhecidos como *hacks* e *filtros* (BUDD; MOLL; COLLISON, 2006).

A próxima seção irá apresentar sucintamente alguns dos *hacks* mais utilizados na concepção de aplicações *Web*.

3.3.4 *CSS Hacks* e Filtros

Um filtro CSS ou *hack* é uma maneira de exibir ou ocultar regras ou declarações, de um navegador em particular ou grupo de navegadores. Os programadores utilizam os filtros baseados nas falhas dos navegadores como *bugs* na análise sintática e código CSS não implementados ou programados incorretamente, para ocultar ou mostrar regras desse navegador (BUDD; MOLL; COLLISON, 2006).

Contudo, os navegadores modernos apresentam um bom suporte as recomendações. O navegador que menos suporta estes é o *Internet Explorer® 6.0* e infelizmente é o mais utilizado, os demais, tanto em sua versão para o sistema operacional Windows como para outros sistemas, possuem suporte próximo da excelência. Pode-se perceber que o quadro geral é propício para desenvolvimento com as especificações, o programador só precisa se preocupar em geral com um ou outro navegador que oferece suporte precário a estas (REBITTE; VINICIUS, 2006).

Um *hack* de CSS é uma maneira deselegante, um truque para fazer com que um navegador se comporte de maneira desejada. Geralmente esta técnica é empregada para evitar erros específicos de um *browser*, como o modelo de caixa do IE. Portanto algumas pessoas preferem o termo *patch,* para indicar que, na verdade, é com o comportamento incorreto dos navegadores que se está lidando. Estes *hacks* podem utilizar filtros para aplicarem diferentes regras a navegadores diversificados. Muitas pessoas confundem o termo *hack* com filtros, sendo que na verdade o filtro é um tipo especifico de *hack* utilizado para filtrar diferentes navegadores (BUDD; MOLL; COLLISON, 2006).

Um exemplo de *hack* é o *underscore*, Neste, toda regra que sua escrita começar com *underscore* ou traço baixo *("_")*, somente será interpretada pelo IE, sendo ignorada pelos outros *browsers* (REBITTE; VINICIUS, 2006).

Outro *hack* muito utilizado, semelhante ao *underscore* é o *!important*, porém este na verdade não é um *hack* e sim um aproveitamento da falha do IE quanto ao suporte de prioridades na cascata apresentada anteriormente. Desta forma o *designer* pode utilizar isso para aplicar uma regra a todos os navegadores com o declaração *!important* e uma outra regra exclusiva logo abaixo para o IE, que irá desconsiderar a anterior, conforme exemplo: *#nav {position: relative !important; position: static;}*. Neste caso a regra *relative* ira funcionar para todos os navegadores, exceto no IE, este que interpretará a regra *static* (BUDD; MOLL; COLLISON, 2006).

De acordo com Rebitte, Vinicius (2006), existem diversos *hacks* ou *filtros* assim como *bugs*, estes que devem ser pesquisados de acordo com os problemas encontrados.

Desta forma, *hacks* ou filtros podem ser uma arma importante no arsenal de CSS do desenvolvedor. Porém os *hacks* devem ser usados criteriosamente e de preferência como um último recurso (BUDD; MOLL; COLLISON, 2006).

3.3.5 DHTML e DOM

A idéia inicial da *Dynamic* HTML ou DHTML era ser um conjunto de inovações do Internet Explorer® 4.0, como um grupo de tecnologias, que disponibilizase para os desenvolvedores de páginas *Web* a habilidade de construir documentos que interagissem com usuários, sem necessidade de processamento por parte do servidor. Assim a DHTML não seria uma linguagem e sim um conjunto de ferramentas, para tornar dinâmica a conhecida HTML (RIOS, 2003).

A DHTML não é uma linguagem de programação, e sim um conjunto de tecnologias que disponibilizam ferramentas para tornar dinâmica a HTML. Tecnologias como:

a) *CSS Positioning*: permite a alteração do posicionamento dos elementos nas páginas como textos e imagens por meio de uma linguagem *client scripting*;
b) *Client Scripting*: pequeno programa que será interpretado pelo *browser* do usuário. Algumas linguagens de *scripting* são *JavaScript*, *JScript*, VBScript e *ECMAScript*;
c) *Document Object Model (DOM)*: modelo de objetos (com suas propriedades e métodos) que são expostas ao programador por meio de um evento ou *event model* (KERLAN, 2000).

A especificação DOM, teve sua primeira manifestação denominada de DOM Level 1. Desde então são atualizadas suas especificações e por insistência da WASP, os navegadores vieram a oferecer suporte a grande parte das especificações Level 1 (ZELDMAN, 2003).

Assim, o DOM é o mecanismo que permite a programação DHTML, com modelos de objetos expostos pelo navegador e acessível a partir de uma linguagem de *client-scripting* (linguagem cujo código é executado no navegador do visitante) como *JavaScript*. Com o simples fato de o *script* ser processado na máquina do usuário, se obtém melhorias de desempenho (RIOS, 2003).

De acordo com Zeldman (2003) o DOM é uma interface independente de navegador e plataforma. É uma linguagem neutra, que permite *scripts* acessarem e atualizarem dinamicamente o conteúdo, estrutura e estilo de documentos. As páginas também podem ser processadas e os resultados reinseridas novamente a elas. Como bonificação, em vez de sobrecarregar o servidor e entupir o processamento com solicitações HTTP, a interatividade orientada ao modelo de objetos acontece no cliente, e a funcionalidade ocorre mesmo se a conexão com a Internet for interrompida.

Na realização desta tarefa, este conjunto de ferramentas, possui um *Event Model* (Modelo de Evento) que permite associar eventos do usuário aos *client-scripts* programados em uma página *Web*. O *event model* permite que o documento reaja a uma ação (evento) do usuário sobre um elemento HTML, com o movimento do *mouse* sobre objetos da página, inserção em caixas de texto, cliques e outros. Cada evento pode disparar um *script* no cliente para alterar as propriedades da página (RIOS, 2003).

Simplificando, a especificação W3C DOM, torna acessível à manipulação dos componentes das páginas *Web*, como as folhas de estilos, elementos de marcação por meio de *scripts* (ZELDMAN, 2003).

Na utilização deste método, cada elemento HTML representa um objeto no *document object model* (DOM), o qual cada um tem seu próprio conjunto de propriedades, métodos e eventos. Normalmente estes são acessados por meio de seu atributo *id* (RIOS, 2003).

Com o uso da tecnologia de modelos de objetos pode-se desenvolver funções de ordenação de listas ou tabelas, controlar posicionamento de elementos ou textos, esconder e reapresentar objetos para o usuário, alternar entre folhas de estilo CSS, ou qualquer processamento da página. Estas tarefas podem ser desencadeadas somente por meio de um clique do usuário sobre um *link* associado a um evento. Desta forma todo o processamento ocorre de modo local, sem necessidade de transferência de dados com o servidor (ZELDMAN, 2003).

A tecnologia DHTML abre caminho para uma nova geração de aplicações *Web*. Por meio deste conceito é possível ter total controle sobre os elementos que compõe uma página HTML/XHTML, permitindo assim grande flexibilidade para interação com o usuário (RIOS, 2003).

Conforme Zeldman (2003) dispositivos de mão e telefones *Web* ainda não oferecem suporte ao DOM e navegadores de texto, como o *Lynx*, nunca oferecerão. Desta forma, deve-se disponibilizar uma versão independente de *scripts*, esta que deve ser especificada com o uso da *tag* <noscript>.

Por esta razão, a tecnologia não foi utilizada na interface. Assim, foram apresentados apenas conceitos das linguagens de *scripting* e do DOM e não seu funcionamento. Estas metodologias não foram empregadas, devido suas funcionalidades não serem essenciais, assim as validações de formulários e *refresh* de páginas, foram realizadas diretamente com a linguagem PHP e XHTML. No capítulo referente a interface, será melhor apresentado a forma de validação utilizada. Por enquanto no

próximo capítulo será apresentado sucintamente as tecnologias utilizadas no lado do servidor.

4 TECNOLOGIAS DO LADO DO SERVIDOR

Algumas tecnologias foram empregadas, para desenvolvimento da interface de gerenciamento de *Firewall* via *Web*. Com exemplificado no Capítulo 3, Seção 3.1 Figura 2, para funcionamento da *Web* é utilizado um *software* denominado *browser*, que realiza algumas tarefas como a comunicação com o servidor e a interpretação dos arquivos enviados por este para apresentação ao usuário.

As linguagens até agora apresentadas, são tecnologias de marcação de texto, como as linguagens estruturais e de apresentação, estas que depois de transferidas para a máquina do usuário são interpretadas pelo *browser* e apresentadas. Neste meio existem as linguagens de *scripting* que funcionam na máquina do cliente, servindo para dinamização das aplicações. Porém a maioria dos *scripts* utilizados na *Web*, são *scripts* que funcionam ao lado do servidor como, por exemplo, o PHP.

O *Hypertext Preprocessor* (PHP) é uma linguagem de criação de *scripts* embutida em HTML/XHTML no servidor. Este é um módulo oficial do servidor HTTP *Apache*, sendo assim, isto significa que os *scripts* desta linguagem são interpretados no próprio servidor (CONVERSE; PARK, 2001).

O processo de colocar código em uma página para o servidor executar, frequentemente é denominado de *server-side scripting*, ou seja, *script* que roda no servidor. A *Microsoft* denomina sua versão *server-side* de *Active Server Pages* (ASP). O PHP é licenciado conforme a Licença Pública GNU (GPL), isto significa que ele é livre para usar em qualquer máquina com qualquer finalidade (BUYENS, 2002).

Este foi um dos fatos, que pesaram na escolha desta linguagem, para desenvolvimento dos *scripts* de processamento da ferramenta de gerenciamento de *Firewall*, por meio da *Web*.

Para funcionamento, como mencionado esta linguagem necessita de um servidor *Web* configurado. O servidor *Apache*, assim como o PHP, é desenvolvido pela *Apache Software Foundation* e licenciados pela GPL. Este servidor *Web*, é estável e altamente expansível por meio de módulos, recurso que torna possível o suporte a maioria das linguagens e formatos de computador utilizados atualmente (BUYENS, 2002).

Desta forma, foi utilizado a linguagem PHP versão 5 e o servidor *Apache* 2 configurados em um sistema operacional *Linux*, distribuição *Debian* versão 4.0 *Etch*.

5 DESENVOLVIMENTO DA INTERFACE *WEB* PARA GERENCIAMENTO DAS FERRAMENTAS *IPTABLES* E *SQUID* NO *LINUX*

Com base nos estudos realizados, foi desenvolvido uma interface *Web*, para gerenciamento de um *Firewall IPtables* e *Squid* no sistema operacional *Linux*. Esta interface, como apresentado nos objetivos da pesquisa, utilizou as especificações de desenvolvimento *Web*, recomendados pelo consórcio de padrões W3C. Neste capítulo, serão apresentadas as especificações para as linguagens utilizadas na concepção deste *software*, bem como as recomendações de acessibilidade satisfeitas na aplicação.

Antes de exemplificar a utilização das especificações, devem-se apresentar os conceitos do sistema, seu propósito, e funcionalidades.

O objetivo da interface é proporcionar o gerenciamento das ferramentas de segurança mencionadas anteriormente por meio da *Web*, facilitando a gerência de redes por parte de administradores. As especificações apresentadas nesta área aplicadas podem proporcionar benefícios, como a melhoria do acesso a tarefas do controle de redes, com fornecimento de outros meios de utilização.

Neste *software* de gerenciamento de segurança em redes, pode-se configurar facilmente a faixa de IP da rede de utilização, cadastrar, editar e excluir palavras e urls proibidas, bem como exceções de alguns eventuais bloqueios. Todas estas funcionalidades em relação à ferramenta *Squid*.

Quanto ao *IPtables* foram desenvolvidas interfaces que tratam da tabela *filter*, estas que envolvem as *chains INPUT, OUTPUT* e *FORWARD*. Desta forma as regras geradas pela interface do sistema, são totalmente funcionais e válidas, evitando um processo longo de escrita, diminuindo a possibilidade de erros e automatizando o processo de criação das regras de segurança. Para esta ferramenta, também foi

disponibilizado meios para manipulação das regras já inseridas no arquivo, caso administradores necessitem alterar a ordem destas regras ou mesmo das *chains*.

Desta forma, as linguagens especificadas empregadas nesta interface, foram a XHTML 1.1 *Strict*, para organização e montagem da estrutura e CSS level 2 para apresentação visual e posicionamento. Estas linguagens também foram utilizadas para aplicar algumas técnicas de acessibilidade das recomendações WCAG 1.0.

As linguagens de *scripting* apresentadas na fundamentação teórica, como ECMAScript (*JavaScript* padrão) e DOM, não foram empregadas, devido a alguns motivos como:

a) as tecnologias não oferecem suporte aos dispositivos móveis e navegadores texto como o *Lynx*;
b) não existem formulários extensos na interface, desta forma a validação com a linguagem PHP (servidor) foi possível, tornando a aplicação ainda mais independente de dispositivos.

Assim, foi utilizado a linguagem de *script* PHP 5 para processamento de informações na interface. Esta foi empregada com objetivo de manipular arquivos textos, enviar instruções ao sistema operacional, controlar acesso por meio de sessão, gerar regras de *Firewall*, validar formulários e outros processamentos.

Para desenvolvimento deste projeto, foi escolhida como base a distribuição *Linux Debian* versão 4.0 denominada *etch*. Como se trata de uma ferramenta *Web*, utilizando ferramentas livres, esta é totalmente portável, podendo ser configurada em outras distribuições e sistemas operacionais. Porém para isso, é necessário alterações em algumas constantes de caminhos dos diretórios, estas que podem variar de acordo com o sistema.

Para melhor abordagem do *software*, serão apresentadas algumas imagens das telas do sistema, para entendimento das tecnologias empregadas em cada etapa, bem como o funcionamento da administração por meio da interface. Assim a Figura 9 apresenta a tela inicial do sistema, contendo um formulário com os campos de usuário e senha para acesso ao gerenciador de *Firewall*.

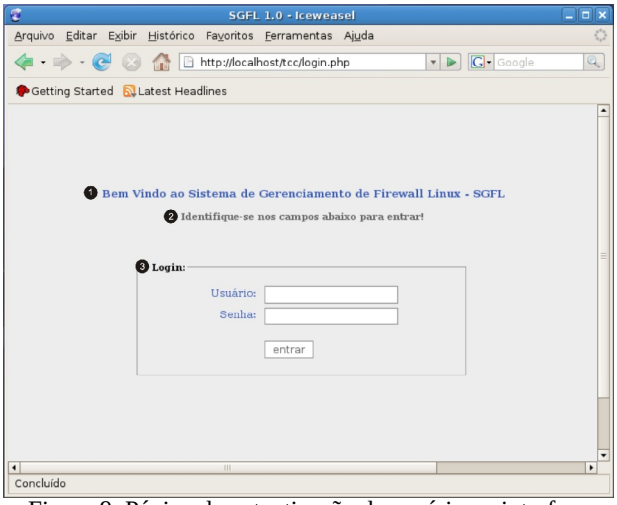

Figura 9. Página de autenticação de usuário na interface

Como se pode observar a Figura 9 apresenta dois títulos, um apresentando a frase de boas vindas (número 1) e outro informando que deve ser realizada a autenticação para obter acesso (número 2). Para diagramação semântica, foram utilizadas as *tags* *<h1>* e *<h2>* as quais identificam como título e subtítulo, desta forma foram utilizados os elementos em seus verdadeiros propósitos.

Na mesma figura, o número 3, indica o quadro que envolve os campos de usuário e senha e ainda o botão de *entrar* do formulário. Este efeito trás organização visual e estrutural a página. Como esta página apresenta baixa quantidade de dados, os únicos benefícios com a utilização deste elemento foram em relação à apresentação visual, porém nos formulários de inserção de regras que será demonstrado, os benefícios foram maiores.

Para apresentação deste quadro, foi utilizado a *tag* <*fieldset*> e sua descendente <*legend*> as quais especificam o grupo de informações envolvidas dentro da caixa e o nome para o grupo de elementos respectivamente. Quanto ao funcionamento, é interessante ressaltar a sessão com usuário e senha, aspectos de segurança onde foi empregado criptografia com o algoritmo *MD5*.

A Figura 10 apresenta um trecho do código fonte desta página, onde se pode observar a estrutura semântica e organizada.

```
<h1 class="bem_vindo">Bem Vindo ao Sistema de Gerenciamento de Firewall Linux - SGFL</h1>
<h2 class="msg_login">Identifique-se nos campos abaixo para entrar!</h2>
<p class="erro_login">
<?php if(isset($_GET[erro])&&($_GET[erro]==1)){
      echo "Erro de Usuário e/ou Senha!";
} ?>
</p>
<div id="cx_login">
   <fieldset>
      <legend>Login: </legend>
         <form action="./login.php" method="post">
            <div>
               <label for="usuario" class="labellogin">Usuário: </label>
                  <input type="text" name="usuario" id="usuario" /><br />
               <label for="senha" class="labellogin">Senha: </label>
                  <input type="password" name="senha" id="senha" /><br /><br />
               <input class="escondido" type="hidden" name="acao" value="1" />
               <input class="centro" type="submit" name="entrar" value="entrar" accesskey="1"/>
            </div>
         </form>
   </fieldset>
</div>
```

Figura 10. Código fonte da página de autenticação de usuário

Alguns outros elementos e atributos utilizados podem ser observados, como a utilização associada da XHTML com a CSS. Para aplicação de estilos visuais aos títulos, foram empregados seletores de classes, como *class=”bem_vindo”* e *class=”msg_login”* estas que referenciam um código CSS da folha de estilos. Assim, foram extintas as *tags font* de formatação visual, separando corretamente a estrutura da apresentação.

Esta mesma página, assim como todo o *software* desenvolvido pode ser apresentada, com uma segunda folha de estilos CSS alternativa. Esta foi desenvolvida para melhorar acessibilidade aos portadores de deficiências visuais, aplicando maior contraste de cores e fontes grandes. Conforme pode ser visto na Figura 11.

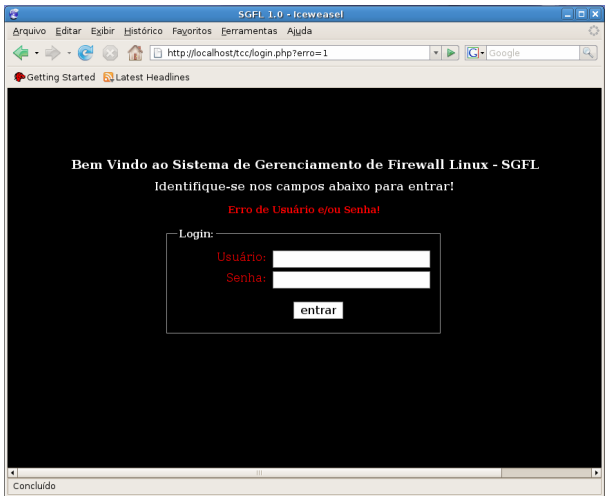

Figura 11. Página de autenticação de usuário, utilizando CSS alternativa acessível

Esta folha de estilo em cascata alternativa pode ser escolhida pelo usuário por meio de opções no navegador. Porém para disponibilizar corretamente, foi necessário aplicação da *tag link* e seus atributos *rel*, *type* e *title*, conforme apresentado na Figura 12.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="pt-br">
<head>
<title><?php include('titulo.php'); ?></title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta http-equiv="Content-Language" content="pt-br">
<link href="estilo.css" rel="stylesheet" type="text/css" />
<link href="acessivel.css" rel="alternate stylesheet" type="text/css" title="Acessivel Grande" />
</head>
```

Figura 12. Código apresentando a inserção de uma CSS alternativa

O atributo *rel*, teve de receber o valor *alternate stylesheet*. Foi necessário também o empregado o atributo *title* para que o *browser* oferece-se suporte a troca das folhas de estilo, sem este atributo a CSS não aparece nas opções para o usuário.

Outro ponto que pode ser observado é a declaração DTD, esta que faz referência ao modo de apresentação que o navegador deve assumir. A declaração assume a especificação do consórcio W3C para XHTML 1.0 *Strict*, referente à última revisão, denominada 1.1. Também pode ser observada a presença de algumas *tags meta*, que especificam o idioma e a codificação do documento.

Conforme a barra de auxílio *Web developers* do *browser Firefox*, pode-se observar o modo de renderização que o navegador assume a diferentes páginas. A Figura 13 apresenta o modo assumido pelo sistema de acordo com a especificação DTD.

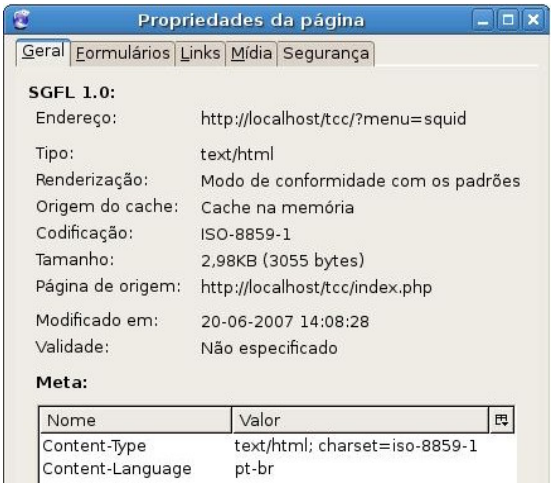

Figura 13. Propriedades da página e modo de renderização

Quanto à estrutura, esta interface foi desenvolvida utilizando divisões para o *layout*, empregando-as corretamente de forma a tornar portável e entendível a diversos dispositivos e plataformas. Com esta metodologia, as tabelas para diagramação foram excluídas do processo, levando em conta o principal fundamento do *Tableless*. Com esta atitude, a estrutura e o código ficam mais organizados, bem como a manutenção facilitada.

As divisões foram utilizadas conforme a necessidade para realizar a organização em blocos de informações. O posicionamento destas *divs* foi realizado por meio das folhas de estilo, por se tratar de aspecto visual. A Figura 14 apresenta um esboço do planejamento visual da interface e as divisões utilizadas.

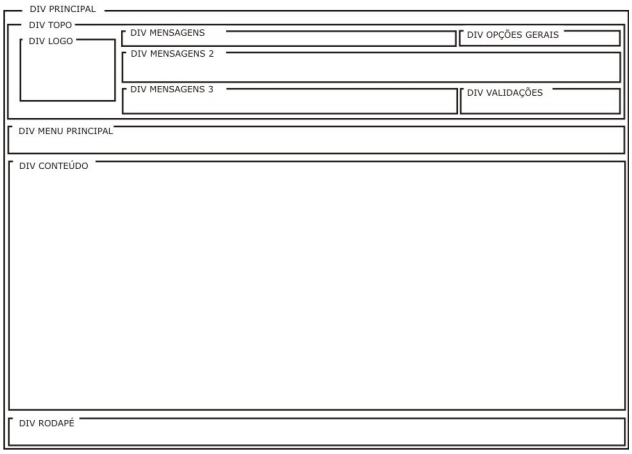

Figura 14. Estrutura da interface, com posicionamento por meio de CSS

Um trecho do código que realiza a estrutura geral da interface pode ser observado na Figura 15. Esta que apresenta a programação de algumas *divs* como, menu principal, conteúdo e rodapé da aplicação. A identificação de cada *div* é realizada por meio do atributo *id*, estes que associados às folhas de estilos, aplicam efeitos desejados em qualquer elemento.

```
<div id="menuprincipal">
   <?php
      switch($_GET[menu]){
         case "squid" : echo "<ul>
                              <li><a href='./?menu=squid&pagina=cadastro' accesskey='a'>Cadastro URL
                              <li><a href='./?menu=squid&pagina=listar_squid' accesskey='b'>Listar/E
                              <li><a href='./?menu=squid&pagina=configurarsquid' accesskey='c'>Reini
                           </ul>";
                        break;
         case "iptables" : echo "<ul>
                              <li><a href='./?menu=iptables&pagina=inserir_regra&ident=i' accesskey=
                              <li><a href='./?menu=iptables&pagina=inserir_regra&ident=o' accesskey=
                              <li><a href='./?menu=iptables&pagina=inserir_regra&ident=f' accesskey=
                              <li><a href='./?menu=iptables&pagina=listar_iptables' accesskey='d'>Li
                              <li><a href='./?menu=iptables&pagina=configurariptables'accesskey='e'>
                           </ul>";
                        break;
         default : echo "<ul></ul>";
                        break;
      }
   ?>
</div>
<div id="conteudo">
   <?php include($pagina).".php"; ?>
</div>
<div id="rodape">
   <p class="font_branca texto_centro">SGFL v1.0 Beta - Sistema de Gerenciamento de Firewall Linux</p>
</div>
```

Figura 15. Código fonte, apresentando parte da estrutura da interface

Desta forma, a Figura 16 apresenta a interface desenvolvida, seguindo o planejamento das divisões. Assim, os menus e opções que possam influenciar o

conteúdo ficaram dispostos no cabeçalho, facilitando o entendimento e acesso as informações.

Figura 16. Interface desenvolvida seguindo a estrutura planejada

Os menus de opções das ferramentas podem ser acessados por meio de um *link* disposto no cabeçalho, onde deve ser feito a escolha da ferramenta a ser manipulada, como *Squid* ou *IPtables*. Após o clique na opção desejada, a lista de *links* referentes a ferramenta é apresentado. Neste caso a escolha pelo *IPtables*, demonstrou o menu de opções, favorecendo o acesso as tarefas de inserção de regras nas *chains* *INPUT*, *OUTPUT* e *FORWARD*, bem como a listagem dos comandos inseridos e algumas ações a serem tomadas com base nesta. Também é oferecida a opção de reiniciar as regras junto ao núcleo do sistema.

Observe o uso do termo *lista*. Novamente, desviando do modo tradicional e adotando as especificações do W3C, os *menus* foram desenvolvidos utilizando listas com o elemento *ul* e *li* e dispostos visualmente com o uso de CSS. Na Figura 15, também se pode observar esta utilização para disposição dos menus, quando no modo não conforme as especificações, seriam utilizados tabelas unidas aos elementos *links*, *font*, *align* e outros, tornando a semântica esquecida e cada vez menos acessível. Na

figura, pode parecer um pouco complicado, isto devido à união da estrutura a um *script php* que realiza a disposição destas listas conforme a escolha do usuário.

Outra especificação utilizada que beneficia a acessibilidade, por meio de dispositivos aurais, e a organização, foi o emprego de tabelas, em seu sentido semântico. As tabelas foram utilizadas para apresentação de dados, como as palavras e *urls* proibidas e exceções cadastradas no sistema. Para formação desta tabela, foram empregadas corretamente as *tags* para agrupamento de informações como grupo de dados de cabeçalho, rodapé e conteúdo, que são elas: <*thead*>, <*tfoot*>, <*tbody*> respectivamente. A Figura 17 apresenta esta utilização.

```
<table cellspacing="0" class="tabela" summary="Tabela contendo palavras proibidas na rede.">
   <caption>Tabela de Palavras Proibidas</caption>
<thead>
<tr>
   <th scope="col">Palavra</th>
   <th scope="col" class="coluna_secundaria">Ações</th>
</tr>
</thead>
<tfoot>
<tr>
   <th scope="col" colspan="2" class="font_branca">Gerado por SGFL</th>
   <th scope="col"></th>
</tr>
</tfoot>
<tbody>
<?php
   for($i = 4; $i < $tam; $i++){?>
   <tr>
      <td class="coluna_principal"><?php echo $leitura[$i]; ?></td>
      <td class="coluna_secundaria"><?php echo "<a href='./?menu=squid&pagina=editar_palavras
      <?php echo "<a href='./?menu=squid&pagina=excluir_palavras&palavra=$leitura[$i]'>"; ?><
   </tr>
   <?php } ?>
</tbody>
```

Figura 17. Código fonte, apresentando estrutura e acessibilidades em uma tabela

Como pode ser observado, também foi utilizado as *tags* <*caption*> para identificação do título da tabela, <*th*> para títulos de colunas e o atributo *summary*, para representar um resumo textual do conteúdo da tabela, poucos detalhes que facilitam os agentes de usuário.

A visualização desta tabela, formatada por um agente de usuário (navegador), cujo código fonte foi exemplificado na Figura 17, pode ser visualizada na

Figura 18. Neste caso o *browser* utilizado foi o *Iceweasel*, uma derivação do conhecido *Firefox*.

Figura 18. Apresentação da tabela de palavras proibidas

Nesta figura, também se pode observar a presença de *links* na tabela para edição ou exclusão das palavras presentes na relação de palavras proibidas. Estes representados pelas imagens. Caso a opção seja exclusão, o sistema deverá efetuar um *refresh* na página carregando o novo estado da tabela, isto devido a não utilização dos *scripts* no lado do usuário. Alguns benefícios são perdidos para aperfeiçoamento e ganho de outros, conforme os motivos apresentados para não utilização destas ferramentas.

Desta forma, foi empregado a *tag* <*meta*> para realizar este tipo de *refresh* e redirecionamento. Esta *tag*, também foi utilizada para apresentar mensagens de erros e retorno aos formulários que não passaram na validação. As Figuras 19 e 20 representam respectivamente os exemplos de mensagens de erro e sucesso apresentadas ao usuário, enquanto a *tag meta* cumpria com os segundos até o redirecionamento.

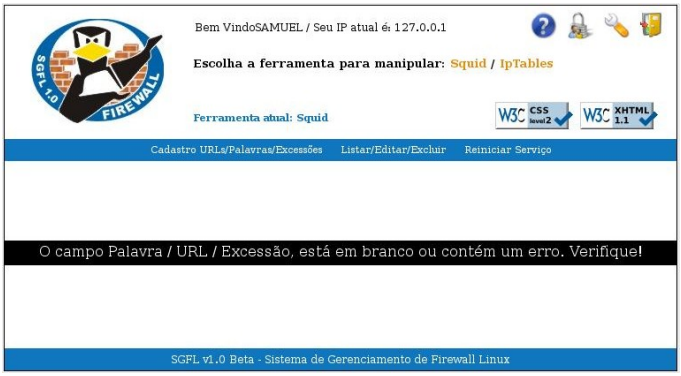

Figura 19. Exemplo de mensagem de erro por meio da *tag meta*

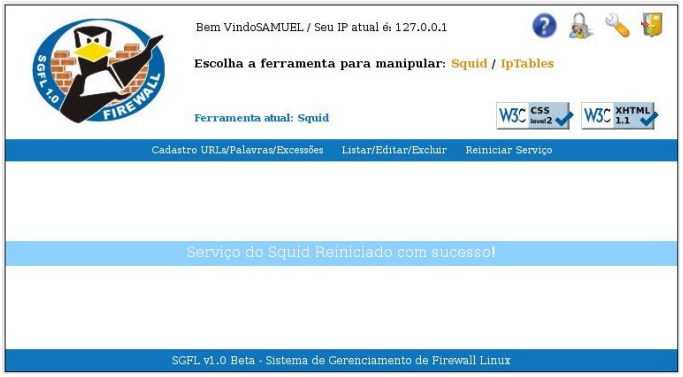

Figura 20. Exemplo de mensagem de sucesso por meio da *tag meta*

O código fonte utilizado na *tag meta* para apresentar estas mensagens, está associado a uma folha de estilos a qual é relacionado a uma *div*, esta que constrói a faixa azul ou preta apresentada nas imagens. A Figura 21 apresenta o código empregado para obtenção deste resultado.

```
<div id="erro">
   <?php echo "O arquivo de regras do IpTables não foi encontrado/criado! Verifique as configurações."?>
   <meta http-equiv="Refresh" content="5; url=./?menu=iptables&pagina=configurariptables" />
</div>
```

Figura 21. Código fonte referente à *tag meta*

Como mencionado anteriormente sobre a organização de estruturas com a *tag <fieldset>* pode-se observar na Figura 22 a melhor identificação dos grupos para preenchimento no formulário, conforme os dados que devem ser inseridos. As páginas

de inserção de regras do *IPtables* necessitaram deste recurso para organização, tornando menos confuso para usuários não tanto familiarizados com o *software*.

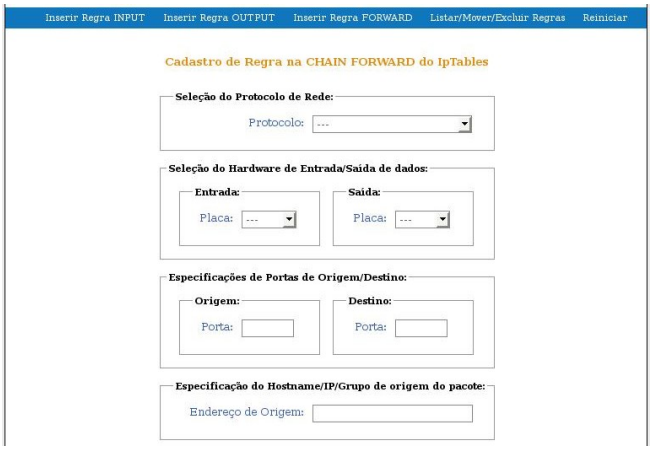

Figura 22. Organização em grupos por meio de *fieldset* nos formulários

Neste formulário é visualizada a possível especificação de protocolo, placa de rede de entrada e saída, especificações de portas de entrada e saída e endereço de origem, outras informações podem ser inseridas neste para construção da regra a *chain FORWARD*, porém devido ao tamanho não foi possível a apresentação total na figura.

Para construção desta interface, foram empregados diversos aspectos que condizem as boas práticas de programação, especificações e recomendações de acessibilidade. Neles foram utilizadas *tags* como: <*label*> para associar rótulos aos seus respectivos *text areas* e <*optgroup*> para classificação em grupos de informações na tag <*select*>. A Figura 23 apresenta esta utilização.

```
<label for="protocolo">Protocolo:</label>
   <select name="protocolo" class="input" id="protocolo">
      <option selected>---</option>
         <optgroup label="Camada de Transporte">
            <option value="TCP">TCP</option>
            <option value="UDP">UDP</option>
         </optgroup>
         <optgroup label="Camada de Rede">
            <option value="ICMP">ICMP</option>
         </optgroup>
   </select>
```

Figura 23. utilização *das tags optgroup*, *label* e *select*

Um atributo também utilizado que acrescenta em relação a acessibilidade foi o *accesskey*, permitindo teclas de atalho para utilização dos *links*, estas que podem ser percebidas nas Figuras: 10 e 15.

Assim, um dos maiores testes realizados com a interface em relação a sua acessibilidade, foi seu funcionamento adequado ao navegador *Lynx*, um *browser* modo texto, que não suporta linguagens de *scripting* e imagens. As Figuras 24, 25 e 26 apresentam a utilização neste agente de usuário.

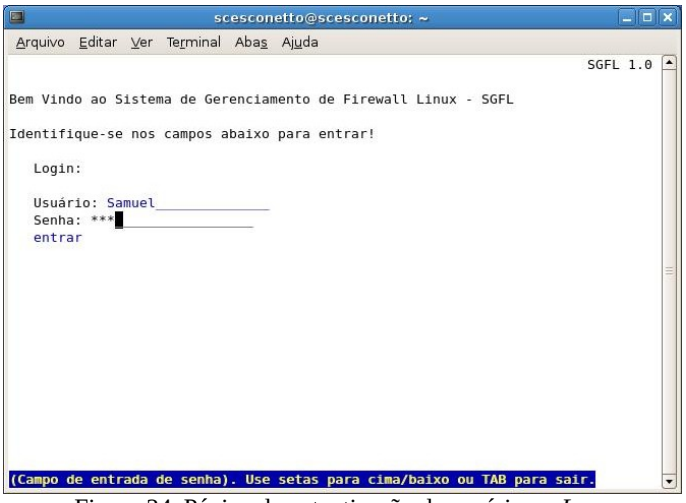

Figura 24. Página de autenticação de usuário no *Lynx*

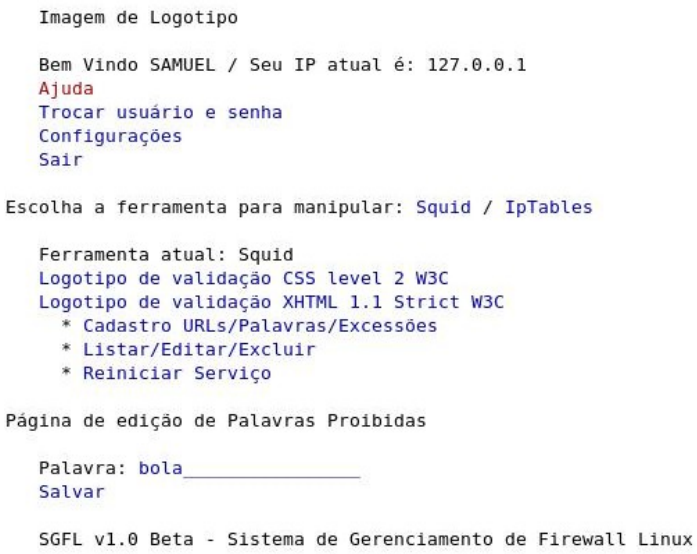

Figura 25. Página de listagem de palavras proibidas no *Lynx*

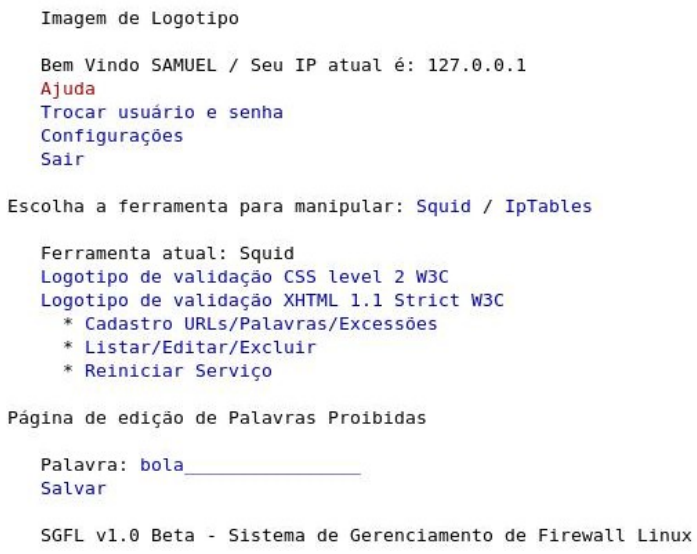

Figura 26. Página de listagem de palavras proibidas no *Lynx*

Esta funcionalidade em modo texto, teve sua apresentação adequada devido à organização estrutural e a separação do conteúdo da apresentação. Esta divisão que se obtém com a utilização das especificações das linguagens XHTML e CSS. Algumas questões de acessibilidade, como a utilização dos atributos *alt* e *longdesc* para

apresentação da descrição em texto de forma alternativa as imagens, também proporcionaram o funcionamento. Como se pode observar na Figura 26 em vez da apresentação do logotipo, o *Lynx* apresentou o conteúdo do atributo *alt*, ou seja, *Imagem de Logotipo*.

5.1 METODOLOGIA

A metodologia adotada para desenvolvimento deste trabalho, consistiu em algumas etapas seqüenciais como: levantamento bibliográfico, estudo do sistema operacional *Linux*, busca de conhecimento das ferramentas de segurança *Squid* e *IPtables*, estudo das especificações de desenvolvimento *Web*, estudo das questões de acessibilidade, busca por trabalhos correlacionados, modelagem e implementação da interface *Web* e finalizando com testes no *software* desenvolvido.

Algumas etapas desta metodologia foram apresentadas anteriormente na fundamentação teórica. Estas como o levantamento bibliográfico, estudos sobre as ferramentas *Squid* e *IPtables*, levantamento de trabalhos correlacionados, estudo das especificações de desenvolvimento *Web* e das questões de acessibilidade.

A etapa de modelagem e implementação do sistema foi a parte mais demorada do projeto, onde se obteve alguns diagramas de caso de uso, para iniciar o processo de abstração do problema. Nestes diagramas mesmo que aparentemente básicos, tornaram possível a observação dos principais objetivos que a interface deveria atender.

As Figuras 27 e 28 apresentam os diagramas UML de caso de uso desenvolvidos, os quais tratam com clareza as tarefas que o administrador de redes pode desenvolver por meio desta interface.

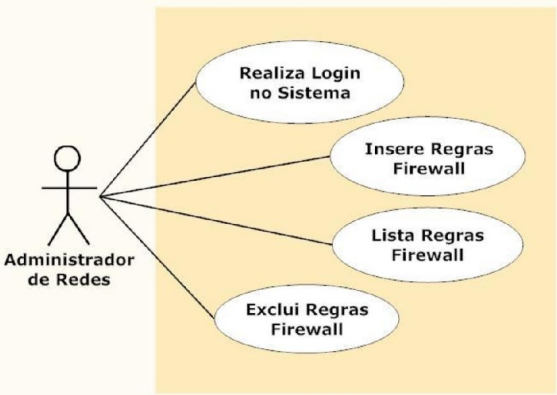

Figura 27. Diagrama de caso de uso da interface

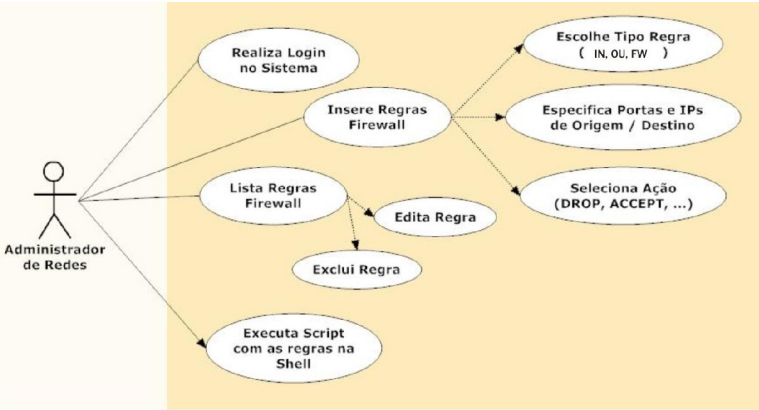

Figura 28. Diagrama de caso de uso, especifico *IPtables*

Após este breve esboço do funcionamento, foi realizada a modelagem da estrutura da interface conforme apresentado na Figura 14. Logo após, foi iniciado o processo de desenvolvimento.

No desenvolvimento, foi necessário o estudo básico sobre *Shell Script* para efetuar o armazenamento das regras do *IPtables* nesta e realizar a execução por meio de uma chamada *system()* (função *php* para enviar instruções a *shell*) que ativa estas regras.

Um outro problema foi encontrado, quanto ao sistema de permissão do *Linux*, pois o usuário *www-data* do *Apache*, não possui permissão de execução de alguns comandos o qual é necessária permissão de *root*[8]. Desta forma, foi preciso a

[8] Administrador em sistemas tipo *Linux*

utilização do comando *sudo*, previamente configurado no arquivo */etc/sudoers*. Este que se tornou um dos pontos fracos e vulneráveis do sistema.

A etapa de testes foi realizada em uma rede local doméstica (LAN), com dois computadores, onde a interface demonstrou ser totalmente funcional quanto a seus propósitos. Foram realizados testes de acessibilidade no navegador texto *Lynx*, testadas as funcionalidades com as folhas de estilos desabilitadas e nos modos de simulação de dispositivos móveis, disponível na barra de ferramenta *webdevelopers* do *browser Mozilla Firefox*.

Desta forma, no próximo capítulo serão apresentados os pontos fortes e fracos encontrados na interface durantes as etapas de desenvolvimento e testes. Sendo que os pontos fracos relacionados neste trabalho poderão ser utilizados como objetivos para trabalhos futuros.

6 CONSIDERAÇÕES DA INTERFACE

Durante o desenvolvimento do trabalho, nas etapas de modelagem, implementação e testes do *software*, foram identificados alguns pontos fortes e fracos. Estes que podem ser apresentados para melhor concepção das conclusões obtidas.

6.1 PONTOS FORTES

Os pontos fortes mais relevantes aos objetivos do trabalho podem ser listados:

a) a interface se tornou acessível devido a utilização das especificações de desenvolvimento como as linguagens XHTML e CSS;
b) foram empregadas corretamente as marcações com seus respectivos objetivos, tornando o conteúdo estruturado e portável;
c) a interface apresentou funcionamento adequado em navegação modo texto, e simulações de outros dispositivos;

Dos pontos de verificação das recomendações de acessibilidade WCAG, foram satisfeitos:

a) *fornecer um equivalente textual a cada elemento não textual por meio dos atributos alt ou longdesc* (Prioridade 1);
b) *não recorrer apenas à cor* (Prioridade 3);
c) *utilizar corretamente marcações e folhas de estilo* (Prioridade 2);
d) *criar documentos passíveis de validação por gramáticas formais, publicadas* (Prioridade 2);

e) *utilizar elementos de cabeçalho indicativos de estrutura do documento, de acordo com as especificações* (Prioridade 2);

f) *marcar corretamente listas e pontos de enumeração* (Prioridade 2);

g) *indicar claramente qual o idioma utilizado* (Prioridade 3);

h) *em tabelas de dados com dois ou mais níveis lógicos de cabeçalhos de linha ou de coluna, utilizar marcações para associar as células de dados às células de cabeçalho* (Prioridade 1);

i) *não utilizar tabelas para efeitos de disposição em página, a não ser que a esta continue a fazer sentido depois de ser linearizada* (Prioridade 2);

j) *fornecer resumos das tabelas* (Prioridade 3);

k) *fornecer atalhos por teclado para links importantes* (Prioridade 3).

l) *fornecer informações de contexto e orientações* (Prioridade 2);

m) *fornecer mecanismos de navegação claros* (Prioridade 2).

Desta forma, estas questões e pontos de verificação levantados podem ser considerados pontos fortes em relação a interface desenvolvida. Porém da mesma maneira, foram identificados alguns pontos fracos, dentre os quais podem ser relacionados a seguir.

6.2 PONTOS FRACOS

Os pontos fracos identificados foram:

a) devido às validações de dados serem realizadas por parte do servidor, quando algum erro é encontrado ele é identificado, o formulário retorna totalmente sem preenchimento e os dados digitados são perdidos. Assim, deve ser novamente preenchido, quando deveria apenas ser corrigido o

que estava errado. Uma possível solução seria a utilização de linguagens de *scripting* ao lado do cliente, porém a acessibilidade em alguns dispositivos pode ficar comprometida;

b) para funcionamento, a interface precisou utilizar o *sudo*, ferramenta *Linux* para dar permissões maiores ao usuário. Desta forma foi concedida permissão total no sistema ao usuário *www-data* quando poderia ser oferecida somente permissão às tarefas que este precisa realizar. Tornando um ponto fraco em relação a segurança, que é um dos objetivos que este pretende oferecer;

c) para configuração da interface em outros sistemas operacionais ou distribuições, pode ser necessário configurar algumas constantes de caminho, às quais não foram oferecidos meios para esta tarefa, devendo ser alterada no código fonte da aplicação.

d) não foi desenvolvido instalador da ferramenta, sendo que para esta tarefa deve ser realizado alguns comandos manuais para movimentação das pastas ao local desejado e concedidas as permissões de acesso a elas;

e) poderia ter sido empregado o uso de conexão segura como HTTPS, mas o sistema apresenta apenas nível de segurança por sessão com os dados criptografados;

f) poderia ser disponibilizado mais opções de gerência de *Firewalls*, com muito mais funcionalidades no *Squid* bem como ser tratado as tabelas *NAT* e *MANGLE* do *IPtables*;

g) Armazenamento das regras mantidas junto ao *Firewall* em arquivos de formato texto, aonde poderia ser empregado bancos de dados oferecendo maior robustez ao trabalho.

CONCLUSÃO

A pesquisa realizada apresentou as especificações de desenvolvimento *Web*, como as linguagens estruturais HTML, XML e XHTML, as quais contribuem para organização semântica dos dados; linguagem de apresentação CSS, que permite a disposição visual dos elementos; conceitos de linguagens de *scripting ECMAScript* e o modelo de objetos DOM. No processo de fundamentação teórica da pesquisa, foram abordadas as vantagens de utilização destas especificações, inclusive demonstrados por meio da interface desenvolvida.

Esta interface, possibilitou a gerencia das ferramentas livres para estabelecimento de certo nível de segurança nas redes de computadores *IPtables* e *Squid*. Foram facilitadas as questões de criação de regras e estabelecimento alguns controles de conteúdo como palavras e URLs proibidas e também exceções de alguns bloqueios indevidos.

A interface foi desenvolvida com as linguagens *Web* especificadas, apresentadas na fundamentação, junto a algumas tecnologias por parte do servidor como PHP 5 e *Apache* 2. Neste *software*, também foram inseridos características de acessibilidade às quais tornou a aplicação portável e acessível a diversos dispositivos.

As recomendações de acessibilidade também fundamentadas apresentaram os principais pontos de verificação do consórcio W3C bem como os conceitos utilizados.

Desta forma, com a utilização correta das recomendações e especificações das linguagens pelo W3C, pode-se observar com base na aplicação desenvolvida a portabilidade e acessibilidade em diversos navegadores e dispositivos sem a necessidade da criação de múltiplas versões com particularidade para cada meio.

Seguindo os métodos desta monografia, e a concepção da interface, analisando seus pontos fortes e fracos, são propostos como trabalhos futuros a realização de:

a) pesquisas em relação as especificações de linguagens de *scripting* como ECMAScript e DOM, para aprimoramento da interface nos navegadores atuais, mantendo compatibilidade com os navegadores texto e dispositivos móveis;
b) ampliação das funcionalidades da interface, acrescentando mais recursos a administração de redes, bem como melhorias nos requisitos de segurança utilizados na própria aplicação, utilizando certificação digital e acesso por meio de conexão HTTP segura;
c) desenvolvimento de um módulo de logs, para registro do tráfego na rede, para aplicação de agentes inteligentes na busca de tentativas de intrusões e acessos não autorizados. Também sendo possível este sugerir novas regras de segurança a serem aplicadas;
d) emprego de bancos de dados relacionais para armazenamento das regras, oferecendo maior capacidade, confiabilidade e robustez a interface.

REFERÊNCIAS

ABITEBOUL, Serge; BUNEMAN, Peter; SUCIU, Dan. **Gerenciando dados na web**. Rio de Janeiro. Campus: 2000.

ALMEIDA, Leonardo J.; VASCONCELOS, Antonio S. **Utilização de XML e RDF na construção da Web Semântica**. Disponível em: http://twiki.im.ufba.br/pub/MAT054/SemestreArtigos20061/XML_RDF_WEB_SEMANTICA.pdf>. Acesso em: 30 novembro 2006.

ALMEIDA, Maurício Barcelos. **Uma introdução ao XML, sua utilização na Internet e alguns conceitos complementares**. Brasília, 2002. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-19652002000200001&lng=es&nrm=iso>. Acesso em: 30 novembro 2006.

BAX, Marcelo Peixoto. **Introdução às Linguagens de Marcas**. Brasília, 2001. Disponível em: <http://cuba.paradigma.com.br/paradigma/artigos/artigos_02.pdf>. Acesso em: 30 novembro 2006.

BUDD, Andy; MOLL, Cameron; COLLISON, Simon. **Criando páginas web com CSS** – Soluções avançadas para padrões. São Paulo: Pearson, 2006.

BUENO, Francisco da Silveira. **Minidicionário da Língua Portuguesa.** São Paulo: FTD, 2000.

BREITMAN, Karin Koogan. **Web Semântica**: a internet do futuro. Rio de Janeiro: LTC, 2005.

BUYENS, Jim; VIEIRA, Marcos. **Truques rápidos para a criação de páginas web**. São Paulo: Berkeley, 1999.

BUYENS, Jim.**Aprendendo MySQL & PHP**. São Paulo: Makron Books, 2002.

COMER, Douglas E. **Redes de computadores e internet**: abrange transmissão de dados, ligação inter-redes e web. 2.ed. Porto Alegre: Bookman, 2001.

DEMÉTRIO, Rinaldo. **Internet**. São Paulo: Érica, 2001.

DZIEKANIAK, Gisele Vasconcelos; KIRINUS, Josiane Boeira. **Web Semântica**. Florianópolis, 2004. Disponível em: <www.encontros-bibli.ufsc.br/Edicao_18/2_Web_Semantica.pdf>. Acesso em: 30 novembro 2006.

ERCILIA, Maria. **A Internet**. São Paulo: PubliFolha, 2000.

FREITAS, João Vitor; BENJAMIN, Marcelo Braga; PASTOR, Saulo Oliveira.**Usabilidade e Acessibilidade para portadores de necessidades especiais na Web**. Disponível em: <http://www.lepinskidesign.com.br/_personal/tcc/Portable%20Documents/usabilidade.pdf>. Acesso em: 30 novembro 2006.

FRIEDLÉIN, Ashley. **Como gerenciar sites Web de sucesso.** Rio de Janeiro: Campus, 2003.

FUJISAKI, Silvio T. et al. **Desenvolvimento de Ferramentas para Páginas web com Recursos de Acessibilidade para Pessoas com Necessidades Especiais**. In: CONGRESSO BRASILEIRO DE INFORMÁTICA EM SAÚDE, 9, 2004, Ribeirão Preto. **Anais...** Ribeirão Preto, 2004.

HAYDEN, Matt. **Aprenda em 24 horas redes**. 2.ed. Rio de Janeiro: Campus, 1999.

HOLZNER, Steven. **Desvendando XML**. Rio de Janeiro: Campus, 2001.

ISAGUIRE, Katya Regina. **Internet**: responsabilidade das empresas que desenvolvem os sites para web-com. Curitiba: Juruá, 2001.

KRISHNAMURTHY, Balachander; REXFORD, Jennifer. **Redes para a web**. Rio de Janeiro: Campus, 2001.

LEVINE, John R. **Segredos da internet para leigos**. São Paulo: Berkeley Brasil, 1995.

MACEDO, Marcelo da Silva. **Construindo Sites Adotando Padrões Web**. Rio de Janeiro: Editora Ciência Moderna, 2004.

MCGRATH, Sean. **XML**: Aplicações práticas. Rio de Janeiro: Campus, 1999.

MARCELO, Antonio. **Firewalls em Linux**. 3 ed. Rio de Janeiro: Brasport, 2002.

MARCONDES, Christian Alfim. **Programando em HTML 4.0**. São Paulo; Érica, 1998.

MARTINEZ, Maria Laura. **Usabilidade no design gráfico de web sites**. In: (Graphica' 2000) III International Conference on Graphics Engineering for Arts and Design, 2000, Ouro Preto. **Anais…** Ouro Preto, 2004.

MELO, Sandro; TRIGO, Clodonil H.. **Projeto de Segurança em Software Livre**. Rio de Janeiro: Alta Books, 2004.

MILNER, Annalisa. **Como navegar na web**. São Paulo: PubliFolha, 2001.

MOULTIS-PITTS, Natanya; KIRK, Cheryl. **XML Black Book**. São Paulo: Makron Books, 2000.

NETO, Urubatan. **Dominado Linux Firewall IPtables**. Rio de Janeiro: Ciência Moderna, 2004.

NEVES, Pedro M. C. **O Guia Prático da HTML**. Portugal: Centro Atlântico, 2004.

NIELSEN, Jakob. **Projetando websites**. Rio de Janeiro: Campus, 2000.

NIELSEN, Jakob; TAHIR, Marie. **Homepage, usabilidade**: 50 websites desconstruídos. Rio de Janeiro: Campus, 2002.

OLIVEIRA, Wilson José de. **Segurança da Informação:** Técnicas e Soluções. Florianópolis: Visual Books, 2001.

REBITTE, Leonardo; VINICIUS, Marcus BP. **Dominando Tableless**. Rio de Janeiro: Alta Books, 2006.

RIOS, Rosângela Silqueira Hickson. **Projeto de sistemas Web orientados a interface.** Rio de Janeiro: Campus, 2003.

RUSSEL, Rusty. **Linux 2.4 Packet Filtering HOWTO . Revision: 1.19. Netfilter, 2001**. Disponível em: http://www.netfilter.org/documentation/HOWTO/pt/packet-filtering-HOWTO.html. [Online 07/2007];

SOARES, Luiz Fernando Gomes; LEMOS, Guido; COLCHER, Sérgio. **Redes de Computadores das LANs**, **MANs e WANs às Redes ATM**. 6. ed. Rio de Janeiro: Campus, 1995.

SOUZA, Renato Rocha; ALVARENGA, Lídia. **A Web Semântica e suas contribuições para a ciência da informação**. Brasília, 2004. Disponível em: <http://www.ibict.br/cionline/viewarticle.php?id=71>. Acesso em: 30 novembro 2006.

STERNE, Jim. **Marketing na web:** integrando a web à sua estratégia de marketing. 2. ed. Rio de Janeiro: Campus, 2000.

STREBE, Matthew; PERKINS, Charles. **Firewalls**. São Paulo: Makron Books, 2002.

TANENBAUM, Andrew S. **Redes de computadores**. 5.ed. Rio de Janeiro: Campus, 1997.

TAROUCO, Liane Margarida Rockenbach. **Redes de computadores locais** e **de longa** distância. São Paulo: Makron Books, 1986.

TIEGS, Dirceu Pereira; MENNA, Eduardo da Silva. **Conceitos de Web Semântica**. In: II Congresso Sul Catarinense de Computação SULCOMP, 2006, Criciúma. **Anais...** Criciúma, 2006.

TORRES, Elisabeth Fátima; MAZZONI, Alberto Angel. **Conteúdos digitais multimídia:** o foco na usabilidade e acessibilidade. Brasília, 2004. Disponível em: <http://www.scielo.br/pdf/ci/v33n2/a16v33n2.pdf>. Acesso em: 30 novembro 2006.

VALENTINE**,** Chelsea; MINNICK**,** Chris. **XHTML**. Rio de Janeiro: Ed. Campus, 2001.

ZELDMAN, Jeffrey. **Projetando web sites compatíveis:** como construir web sites compatíveis com browsers e dispositivos variados. Rio de Janeiro: Elsevier, 2003.

ZWICKY, Elizabeth. **Construindo Firewalls para a Internet**. São Paulo: Campus 2000.

W3C. **X*HTML* 1.0 – The Extensible Hiper Text Markup Language.** W3C Recomendation, 26 de janeiro de 2000. [Online] http://www.w3c.org/TR/2000/REC-x*HTML*-20000126.

_____.**HTML 4.01 Specification.** W3C Recommendation 24 December 1999. [Online] http://www.w3.org/TR/html401/.

_____.**Web Content Accessibility Guidelines 1.0.** W3C Recommendation 5-May-1999. [Online] http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/.