FACULDADE DE PINDAMONHANGABA

GLAUBER GONZAGA SILVA

# PROTÓTIPO DE SOFTWARE PARA GERENCIAMENTO DE SERVIDORES LINUX

Pindamonhangaba - SP
2009

FACULDADE DE PINDAMONHANGABA

GLAUBER GONZAGA SILVA

# PROTÓTIPO DE SOFTWARE PARA GERENCIAMENTO DE SERVIDORES LINUX

**Projeto apresentado como parte dos requisitos para obtenção do Diploma de Bacharel em Sistemas de Informação pelo Curso de Bacharelado em Sistemas da Informação da Faculdade de Pindamonhangaba.**

**Orientador: Prof. Fabiano Sabha**

**Pindamonhangaba - SP**
**2009**

FACULDADE DE PINDAMONHANGABA

**GLAUBER GONZAGA SILVA**

**PROTÓTIPO DE SOFTWARE PARA GERENCIAMENTO DE SERVIDORES LINUX**

**Monografia apresentada como parte dos requisitos para obtenção do Diploma de Bacharel em Sistemas de Informação pelo Curso de Bacharelado em Sistemas da Informação da Faculdade de Pindamonhangaba.**

Data: _____

Resultado: _____

BANCA EXAMINADORA

Prof. Vladimir Geraseev Júnior Faculdade de Pindamonhangaba

Assinatura: _____

Prof. Fernando Sales Claro Faculdade de Pindamonhangaba

Assinatura: _____

Prof. Fabiano Sabha Walczak_____ Faculdade de Pindamonhangaba

Assinatura: _____

**AGRADECIMENTOS**

Aos professores que me auxiliaram mesmo que em menor escala na estrutura deste documento.

Ao professor Fabiano Sabha, pela paciência e clareza que me manteve auxiliando em todo o decorrer da criação do meu trabalho de conclusão de curso.

Por fim, a todos da Faculdade de Pindamonhangaba que tiveram alguma participação na elaboração deste trabalho.

## RESUMO

Nos dias de hoje, com a evolução da tecnologia, a quantidade de recursos que os computadores proporcionam é imensa. Essa variedade se intensifica quando vários computadores se comunicam entre si, compartilhando informações. Na mesma proporção, é crescente também a necessidade por profissionais qualificados para a manutenção e suporte desses recursos. Dentro de um ambiente corporativo, com suas estações e servidores interligados de forma organizada, a boa gestão dos recursos disponíveis é tão fundamental quanto o conhecimento técnico para o seu bom uso. Reunir uma equipe técnica que possua individualmente o mesmo nível de conhecimento, no entanto, é um trabalho difícil. Nivelar o conhecimento, atender toda a demanda de serviços e extrair o máximo de funcionalidades do ambiente estão, então, em um mesmo patamar de importância. Como proposta para solução deste problema, e para mostrar como isso é possível, este trabalho de conclusão de curso relata o desenvolvimento de um protótipo que visa apresentar uma alternativa para o gerenciamento de servidores com sistema operacional *Linux*, sem que pra isso seja mandatória a aquisição de novos equipamentos. Para especificar e ilustrar ainda mais o tema, é descrito todo o processo de desenvolvimento baseado nas necessidades reais da Silva Informática, empresa prestadora de serviços em TI, com seus clientes.



Palavras-chave: Gerenciamento, Servidores, Linux, Interface.

## ABSTRACT

Today, with the evolution of technology, the range of resources that computers offer is immense. That variety intensifies when many computers communicate between them, sharing information. In the same proportion, is crescent the need for qualified professionals to maintenance and support of these resources. Within a corporate environment, with their clients and servers interconnected neatly, the good management of the available resources is so fundamental then technical knowledge for good use. Together a technical team that has individually the same knowledge level, however, is a hard work. Leveling the knowledge, meet demand and derive the greatest features of the environment are in the same level of importance. As a proposal for solving this problem, and to show how this is possible, this work reports the development of a prototype that aims to present an alternative to server management with Linux operating system, without requiring purchase of new equipment. To specify and further illustrate the theme, is described the whole process of development based on real needs of Silva Informatica, a provider of Information Technology services with their customers.



Keywords: Management, Servers, Linux, Interface


.

## LISTA DE FIGURAS

**SUMÁRIO**

# 1. INTRODUÇÃO

Atualmente, é comum o uso do *Linux* como sistema operacional em servidores utilizando-se apenas o modo texto, por permitir todo o gerenciamento do sistema sem a necessidade do modo gráfico e, conseqüentemente, de um equipamento com maior poder de processamento. O problema nesses casos é que boa parte do trabalho deve ser feito manualmente pelo administrador do sistema, o que aumenta a chance de erros.

Segundo Pritchard *et al.* (2007), o terminal de comandos do *Linux*, mesmo com a aparência de uma simples interface em modo texto, tem a capacidade de gerenciar praticamente todos os recursos do sistema. Porém, para que o administrador do sistema consiga tal feito, ele precisa ter um conhecimento sólido sobre os aplicativos de linha de comando. É verdade que hoje já existem várias ferramentas com interface gráfica disponíveis para o gerenciamento, mas todas elas demandam um servidor com maior poder de processamento.

O propósito desse Trabalho de Conclusão de Curso é apresentar uma alternativa de gerenciamento sem a necessidade de instalação de uma interface gráfica no servidor, através de um protótipo de *software* com interface *Web* que automatiza e auxilia em parte as funções antes feitas manualmente pelo administrador do sistema, de forma que o administrador não fique totalmente dependente do terminal de comandos do sistema operacional.

Dessa forma, o analista ou administrador que acessar o servidor através da interface *Web* gerada pelo protótipo não precisará ter um nível de conhecimento tão aprofundado de sistemas *Linux*, tendo em vista que boa parte desses profissionais não o tem. Para o bom gerenciamento, ele precisará apenas ter o conhecimento necessário para utilizar a interface *Web*.

## 1.1 Relevância do estudo

Todas as questões levantadas sobre gerenciamento dos servidores *Linux* partiram de um levantamento feito com uma empresa real, aqui descaracterizada com o nome fictício Silva Informática, situada na cidade de Taubaté-SP.

A Silva Informática é uma empresa com mais de dez anos de mercado, atuando como prestadora de soluções na área de TI – Tecnologia da Informação em

todo o Vale do Paraíba. Devido ao rápido crescimento do número de clientes, a empresa contratou novos funcionários para suportar a nova demanda de serviços.

Por questões de segurança e baixo custo, grande parte dos clientes adotou o *Linux* como sistema operacional em seus servidores, tendo apenas o terminal de comandos como interface para gerenciamento.

Dessa forma, com o advento do *Linux*, os analistas de suporte estão tendo que rever alguns conceitos, onde os seus serviços não apenas serão demandados pela sua capacidade, mas sim pelo seu conhecimento. (GORAYEB, JUNIOR, CAPELLI, 2009).

Espera-se, então, que com a implementação do protótipo seja possível padronizar o gerenciamento do servidor, sem que haja a necessidade de um treinamento altamente detalhado sobre o funcionamento do sistema *Linux*, pois boa parte das funções passa a ser feita automaticamente pelo protótipo.

Com o protótipo instalado no servidor, o sistema operacional continuará apenas com o modo texto, mas através dele se torna possível visualizar todas as funções pré-configuradas do sistema e seus principais serviços via rede. Dessa forma, a quantidade de recursos que deverão ser adicionadas no servidor é praticamente insignificante.

## 1.2 Delimitação do Trabalho

Embora a segurança seja um ponto sempre levado em consideração em todas as fases do desenvolvimento do protótipo, este trabalho tem como objetivo principal a melhor administração dos servidores com sistema operacional *Linux* pela equipe de TI e o administrador da rede.

No decorrer deste trabalho, algumas medidas de segurança são sugeridas apenas para poder extrair do protótipo o máximo das suas funcionalidades sem que sejam criadas brechas desnecessárias no sistema operacional.

Em relação ao sistema operacional utilizado, todo o desenvolvimento foi voltado para o perfeito funcionamento no *Debian*, embora seja possível que ele funcione pelo menos parcialmente em outras distribuições *Linux.*

## 1.3 Organização do Trabalho

O primeiro capítulo evidencia o problema ao qual este trabalho propõe uma solução, apresentando o problema de se gerenciar servidores *Linux* sem uma interface gráfica intuitiva e apresentando as vantagens que uma interface *Web* pode trazer para a solução deste problema.

O capítulo seguinte apresenta as ferramentas utilizadas para o desenvolvimento do protótipo e os serviços que serão gerenciados pelo protótipo.

A metodologia que é seguida para o desenvolvimento do protótipo compõe o terceiro capítulo, apresentando as características principais da prototipação evolucionária e os benefícios que a mesma trouxe para o desenvolvimento do protótipo.

O quarto capítulo abrange a fase de análise e projeto, onde são especificados os recursos que o protótipo deverá ter e definidos os requisitos mínimos que o servidor deve ter para suportá-lo.

A fase de implementação e testes é abordada no quinto capítulo, onde o protótipo é instalado em um servidor *Linux* com todos os serviços que serão gerenciados, sendo avaliadas as mudanças necessárias no protótipo e as adaptações que o sistema operacional deve sofrer para o perfeito funcionamento do protótipo.

O capítulo 6 aborda a avaliação do protótipo feita pelos funcionários da empresa Silva Informática na fase de verificação e validação, onde são apontados os pontos positivos e negativos dessa versão do protótipo.

Por fim, o sétimo capítulo apresenta as conclusões após uso da aplicação em um ambiente de produção.

# 2. REVISÃO DA LITERATURA

Para que se possa integrar o protótipo e sua interface ao sistema operacional, antes é fundamental conhecer a maneira que o *Linux* e seus serviços interagem. É feito aqui um breve histórico de cada serviço gerenciado, mostrando sua história e a sua importância.

## 2.1 Sistema Operacional GNU/Linux

O *Linux* é um sistema operacional criado por Linus Torvalds no Departamento de Ciência da Computação da Universidade de Helsinque, Finlândia. Todo o seu desenvolvimento iniciou-se usando como base o *Minix*, um pequeno sistema operacional que por sua vez era uma derivação do sistema *UNIX*, sistema operacional multitarefa e multiusuário criado no final da década de 1960 por Andrew Stuart Tanenbaum. (NEMETH, SNYDER, HEIN, 2007).

A primeira versão oficial do *Linux* foi lançada em outubro de 1991 como um projeto pessoal de Linus. Ele disponibilizou gratuitamente o código-fonte de seu *Linux* para programadores de todo o mundo incrementarem e aperfeiçoarem o sistema. Isso tornou o *Linux*, desde as suas origens, um sistema operacional robusto, estável e funcional. (NEMETH, SNYDER, HEIN, 2007).

O *Linux* ainda é hoje um sistema operacional livre e gratuito. Isso quer dizer que qualquer pessoa tem acesso ao código-fonte, podendo alterá-lo, customizá-lo e até comercializá-lo, desde que mantenha o sistema com todas essas liberdades.

O *Linux* se difere das demais variantes do *UNIX* por conter em seu núcleo apenas o básico do sistema operacional. Nele, podem ser incrementados comandos e serviços diversos, adicionando recursos e funcionalidades, como a interface gráfica. Assim surgiram as distribuições *Linux*, que nada mais são que versões do sistema operacional com serviços e recursos diferenciados. Entre as principais distribuições *Linux* hoje está o *Debian*.

### 2.1.1 DISTRIBUIÇÃO DEBIAN LINUX

Segundo Wiel (2009), o *Debian Linux*, ou simplesmente *Debian*, é um projeto criado por colaboradores do mundo todo. O Projeto *Debian*, como é chamado, foi

criado em 16 de agosto de 1993 por Ian Murdock e até hoje é dotado de um rico conjunto de políticas e procedimentos para manter o alto padrão de qualidade. Atualmente, a versão mais recente do *Debian* é a 5.0 (codinome *Lenny*), lançada em 14 de fevereiro de 2009.

### 2.1.2 O SHELL E O KERNEL

Todo sistema *Linux*, independente da distribuição, possui um núcleo denominado *kernel*. Ele é a parte do sistema mais próxima do *hardware*, sendo o responsável pelas chamadas ao sistema, o acesso a dispositivos e o gerenciamento dos recursos da máquina.

O *shell* é a interface básica entre o *kernel* e o usuário. É o nome genérico de uma classe de programas que funciona como interpretador de comandos e linguagem de programação *script*. Segundo Ferreira (2003), o *shell* padrão do *Linux* é o *bash*.

A figura 1 apresenta as camadas do sistema operacional, independente da distribuição:

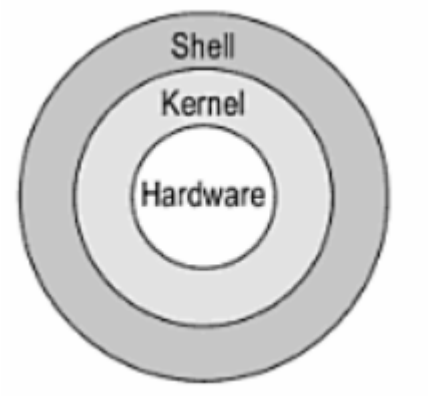


Figura 1 – Representação das camadas do sistema operacional *Linux*

### 2.1.3 O AMBIENTE GRÁFICO

Segundo Ferreira (2003), os sistemas *UNIX* mais antigos são baseados somente em caracteres, porém hoje qualquer distribuição dispõe de *X Windows System* (Sistema de Janelas X), sendo o responsável em definir qual será o ambiente gráfico utilizado pelo sistema sem definir qual será o seu gerenciador de janelas.

Como afirma Pritchard *et al.* (2007), o ambiente gráfico no *Linux* tem o funcionamento semelhante a uma aplicação cliente-servidor, onde os aplicativos funcionam como cliente e o ambiente gráfico como servidor. Os gerenciadores de janelas funcionam, então, como uma aplicação sendo executados sob o *X Windows System*, permitindo a visualização e execução de várias aplicações simultaneamente, além de permitir maior interação do usuário com o sistema. Os gerenciadores de janelas mais utilizados hoje são o *KDE* e o *Gnome*.

## 2.2 Interfaces de Gerenciamento via Web

As duas interfaces de gerenciamento citadas abaixo foram as opções de gerenciamento que foram consideradas pela empresa Silva Informática para utilização em seus clientes. Embora ambas sejam amplamente utilizadas nos mais variados ambientes, os requisitos mínimos de hardware eram altos demais para alguns servidores que a empresa gerencia.

### 2.2.1 KYA PANEL

O *Kya Panel* é um painel para facilitar a administração de servidores *GNU/Linux* melhorando a administração de rede. Atualmente existem dois módulos: Gestão de *Email* e Gestão do Samba.

O primeiro módulo, Gestão de *Email*, gerencia servidores com *Postfix* + *Courier* e duas opções de base de dados: *OpenLDAP* e *SQL*. Caso seja utilizado *SQL*, é possível optar por *PostgreSQL* ou *MySQL*.

Uma de suas principais características é a integração com o *Active Directory* do *Microsoft Windows Server*, que permite a sincronização de usuários e senhas do servidor *Windows* e os usuários de *email* que ficam armazenados no *OpenLDAP*.

O segundo módulo, Gestão do Samba, possui uma interface simples e prática que permite gerenciar compartilhamentos e permissões de acessos dos usuários. Além disso, pode-se ver em tempo real todos os usuários validados e o que cada um deles está acessando no servidor, inclusive com a opção de bloqueá-los se necessário.

O *Kya Panel* foi desenvolvido em três camadas: interface em *PHP*, núcleo em *Shell Script* e base de dados *OpenLDAP* ou *SQL*. Com suas várias opções

administrativas e código simples, ele é uma das ferramentas mais utilizadas na administração de servidores.

### 2.2.2 WEBMIN

O *Webmin* é uma ferramenta de administração gráfica, escrita por *Jamie Cameron* e que utiliza a linguagem *Perl*, o que permite a criação de novos *scripts* adicionando-lhe novas funcionalidades. Ele foi projetado para ser uma ferramenta de administração leve, funcional, e que possa ser facilmente estendida.

A ferramenta está disponível hoje para mais de 20 idiomas, e é considerada a ferramenta oficial de administração em vários sistemas operacionais e distribuições.

O *Webmin* funciona como um centralizador de configurações do sistema, monitoração dos serviços e de servidores, fornecendo uma interface simples, e que quando configurado com um servidor *Web*, pode ser acessado de qualquer local, através de um navegador.

O *Webmin*, assim como o protótipo mostrado neste trabalho, é uma maneira de administrar graficamente o servidor Linux utilizando um *browser*. Ele é compatível com os *browsers* mais conhecidos, como *Firefox, Opera, Konqueror* e *Seamonkey*.

Para ambientes onde a velocidade do *link* e o espaço em disco forem críticos, existe uma versão que disponibiliza apenas a *API - Application Programming Interface* e os principais módulos do *Webmin*. Os demais módulos podem, então, podem ser adicionados gradualmente.

## 2.3 Serviços

Considerando que cada *software* tem seus respectivos serviços e que o sistema operacional interage com eles, como forma de padronização e fácil entendimento, fica definido que neste trabalho todos os recursos gerenciados pelo protótipo serão tratados como serviço, incluindo os recursos gerenciados do próprio sistema operacional.

### 2.3.1 OPENVPN

O *OpenVPN* foi criado no ano de 2002 por James Yonan. Assim como o *Linux*, é livre e gratuito, utilizado para criar redes privadas virtuais, ou *VPNs - Virtual Private Network*. Segundo Montenegro e Ribeiro (2008), as *VPNs* são utilizadas para interligar redes distintas através de um meio de comunicação, como a *Internet*.

Em sua forma mais simples, uma *VPN* é uma conexão que faz com que uma rede remota pareça estar conectada diretamente, mesmo que ela esteja fisicamente a milhares de quilômetros. Para manter a segurança, a conexão deve ser autenticada e o tráfego deve ser criptografado de ponta a ponta. Essa definição é denominada tunelamento seguro (NEMETH, SNYDER, HEIN, 2007).

O *OpenVPN* é capaz de estabelecer conexões diretas entre computadores mesmo que estes estejam atrás de *firewalls* sem a necessidade de reconfiguração da rede. Ele permite criar canais de comunicação através do conceito de tunelamento, onde são criados arquivos de configuração contendo as informações das duas redes onde será estabelecida essa comunicação. Ele cria também uma chave criptográfica que deve ser igual nos dois lados da *VPN*, caso contrário, a comunicação não é estabelecida.

### 2.3.2 SPAMASSASSIN

O *SpamAssassin* é um dos projetos da *ASF - Apache Software Foundation*, criado para filtragem de *spam*. Ele se baseia em uma lista negra de endereços *IP - Internet Protocol* e conteúdos de mensagem catalogados. (MORIMOTO, 2005).

O *SpamAssassin* funciona dando pontuações em cada mensagem que é recebida, analisando o conteúdo da mensagem. Se a mensagem atingir uma pontuação muito alta, ela é marcada como *spam*. Por último, se a mensagem atingir uma pontuação abaixo do limite, ela é deixada na pasta de entrada usual.

Os parâmetros de configuração do *SpamAssassin* são definidos no arquivo local.cf. Nele, além de definir a mensagem de alerta que será exibida nos *emails* marcados como *spam*, é possível configurar as redes confiáveis que não passarão pelo processo de filtragem. Dessa forma, mesmo que a mensagem tenha conteúdo que ultrapasse a pontuação limite, ela não será marcada como *spam*.

O *SpamAssassin* por si só não exclui nenhuma mensagem. Sua função é unicamente de classificar e pontuar cada mensagem através de testes de heurística e filtros *bayesianos*, responsáveis por analisar o conteúdo da mensagem utilizando-se de métodos estatísticos. Dessa forma, o *SpamAssassin* pode ser integrado com os principais servidores de *email*, entre eles, o *Postfix*. Ele fica inativo depois de instalado no *Debian* para evitar que sejam consumidos recursos desnecessários do computador (MORIMOTO, 2005).

### 2.3.3 POSTFIX

O *Postfix* é um agente de transferência de mensagens, ou *MTA - Mail Transfer Agent),* criado por Wietse Venema através de uma licença remunerada do *T. J. Watson Research Center* da *IBM*. Segundo Morimoto (2005), é uma das soluções mais populares entre os aplicativos que gerenciam *emails* em servidores *Linux*, sendo a melhor alternativa ao antigo aplicativo *Sendmail*, muito popular entre os sistemas *UNIX*.

Segundo Trigo (2007), o objetivo do desenvolvimento do *Postfix* foi substituir o *Sendmail* como um servidor de *emails* de código-fonte aberto, rápido, robusto, seguro e de fácil administração. A instalação do *Postfix* pode substituir o *Sendmail* sem provocar grandes alterações no sistema e de forma transparente para o usuário final. As bases de usuário e de correio, inclusive, são as mesmas nos dois aplicativos.

A maior diferença entre o *Postfix* e o *Sendmail* é a segurança. A arquitetura do *Postfix* abrange vários pequenos serviços que atuam de forma cooperativa, descentralizando as funções. O *Sendmail* executa um grande programa que exige privilégios de administrador, abrindo brechas de segurança.

Os arquivos de configuração do *Postfix* são simples, contendo apenas algumas linhas de comandos. Seu principal arquivo de configuração é o main.cf. Embora em muitos casos os seus valores padrão são os mais adequados, é possível especificar mais de trezentos parâmetros diferentes em suas linhas de comandos. (NEMETH, SNYDER, HEIN, 2007).

### 2.3.4 SQUID

O *Squid* é uma ferramenta usada em servidores para *Web*. Conforme Ricci e Mendonça (2006), o conceito de *proxy* pode ser interpretado como um serviço que funciona em conjunto com o servidor e as estações de uma rede, sendo responsável pela filtragem do tráfego de dados da rede local.

Todo o funcionamento do *Squid* é definido no arquivo squid.conf. Nele, além de toda a configuração necessária para o seu funcionamento, são acionadas todas as regras de controle de acesso, em arquivos denominados *ACLs – Access Control List*.

Uma das principais vantagens do *Squid* é a de poder funcionar como um *cache* de páginas e arquivos, armazenando as informações já acessadas. Essa funcionalidade proporciona uma economia de banda na conexão com a *Internet* e torna o acesso mais rápido (MORIMOTO, 2005).

### 2.3.5 APACHE HTTP SERVER

Para que a interface gerada pelo protótipo esteja disponível para ser acessada, é necessário que esteja em um local da rede que permita a interação com os usuários locais. Devido a isso, é fundamental que no servidor onde serão gerenciados os serviços e o sistema operacional haja também uma aplicação que funcione como servidor *Web*, onde todos os arquivos que compõem o protótipo possam ser interpretados, e assim, seja gerada a interface *Web*.

A ferramenta escolhida para ser o servidor *Web* do protótipo é o *Apache*. Como afirma Veiga (2006), o *Apache* é um servidor *Web opensource*, altamente confiável, configurável e extensível, compatível com várias tecnologias de conteúdo dinâmico, como o *PHP*. Ele foi criado em 1995 por Rob McCool e é hoje o servidor *Web* mais utilizado no mundo por ser simples, seguro, robusto e de alto desempenho. Ele é a principal ferramenta da *Apache Software Foundation*, responsável por vários projetos que envolvem tecnologias *Web*.

O *Apache* é compatível com o protocolo *HTTP – HyperText Transfer Protocol*, o que permite através dele gerar aplicações *Web* utilizando-se de linguagens de programação como o *PHP*. Atualmente, existem versões para *Windows, Novell,*

*OS/2* e diversos outros sistemas derivados do *UNIX*, como o *Linux*. Para todas as versões, ele é gratuito mesmo que para uso comercial (VEIGA, 2006).

Este é o único serviço adicional realmente necessário para o funcionamento do protótipo, pois ele é o responsável por gerar a interface *Web* que será acessada de uma estação da rede no servidor *Linux*. Porém, para a sua utilização, é necessário apenas um computador com 48 MB de memória *RAM* e processador *Pentium* 233 ou equivalente (MORAZ, 2005).

## 2.4 Ferramentas utilizadas

Quatro ferramentas foram utilizadas em toda a fase de codificação do protótipo e da interface. Cada uma com a sua função, elas foram responsáveis pela estruturação visual da interface, para a codificação e para o funcionamento propriamente dito do protótipo.

### 2.4.1 DREAMWEAVER MX

O *Macromedia Dreamweaver*, segundo Pinto (2006), é um *software* para a plataforma *Windows* voltado para o desenvolvimento de páginas *Web*. Atualmente está na versão CS4 e pertence a *Adobe*, e suas primeiras versões funcionavam como simples editores *HTML*. Suas versões mais recentes incorporaram o suporte a várias tecnologias *Web*, como *CSS – Cascading Style Sheets*, Javascript, *Ajax* e *ASP – Active Server Pages*. A linguagem *PHP* é suportada pelo Dreamwever desde a versão 4.0. A partir da versão MX, utilizada para o desenvolvimento do protótipo, o *Dreamweaver* incorporou a tecnologia *Ultra-dev*, uma poderosa ferramenta de desenvolvimento para as linguagens *Web*.

O *Dreamweaver* MX possui o modo *Design*, que permite esconder a codificação da aplicação, tornando possível ajustar a parte visual de forma mais rápida.

A figura 2 mostra o *Dreamweaver* MX no modo *Design*:

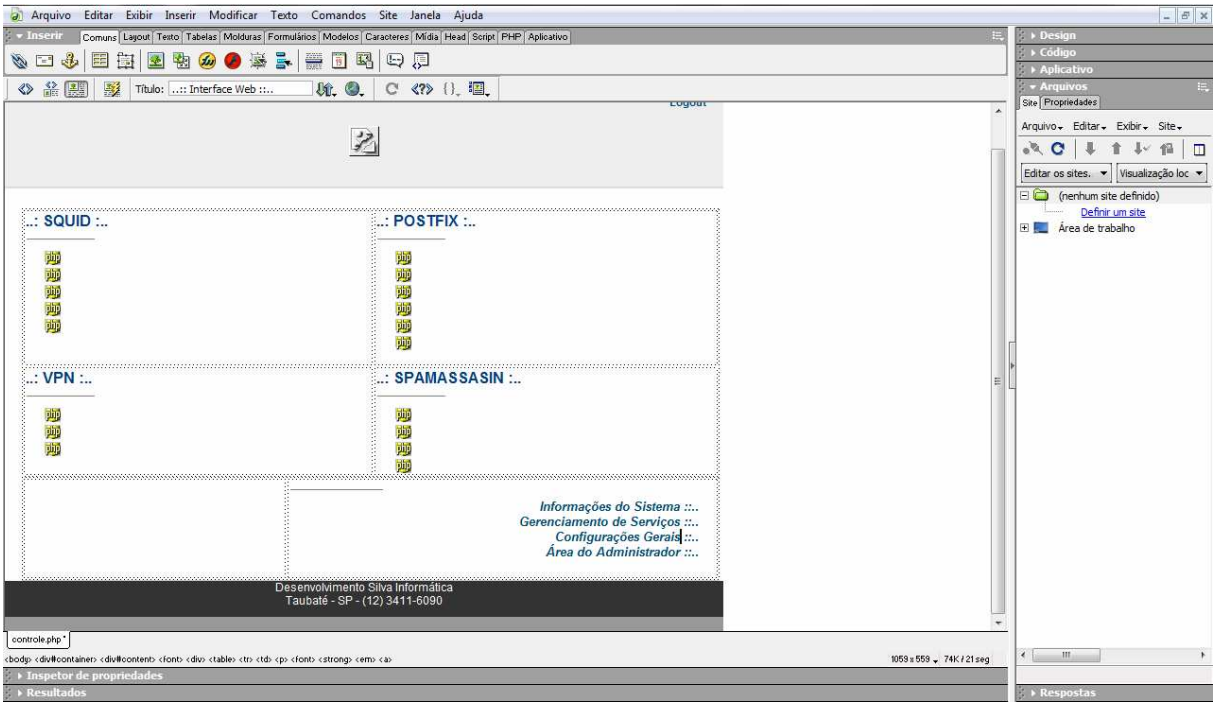


Figura 2 – Modo *Design* do *Macromedia Dreamweaver MX*

Embora facilite a edição de aplicações *Web*, como afirma Pinto (2006), em muitos casos o *Dreamweaver MX* produz códigos desnecessários, o que pode tornar a aplicação mais lenta em computadores mais antigos. Devido a isso, o *Dreamweaver MX* possui também o modo Código, onde o desenvolvedor pode alterar o código gerado e ver o resultado da alteração no modo *Design*.

A figura 3 mostra o *Dreamweaver MX* no modo Código:

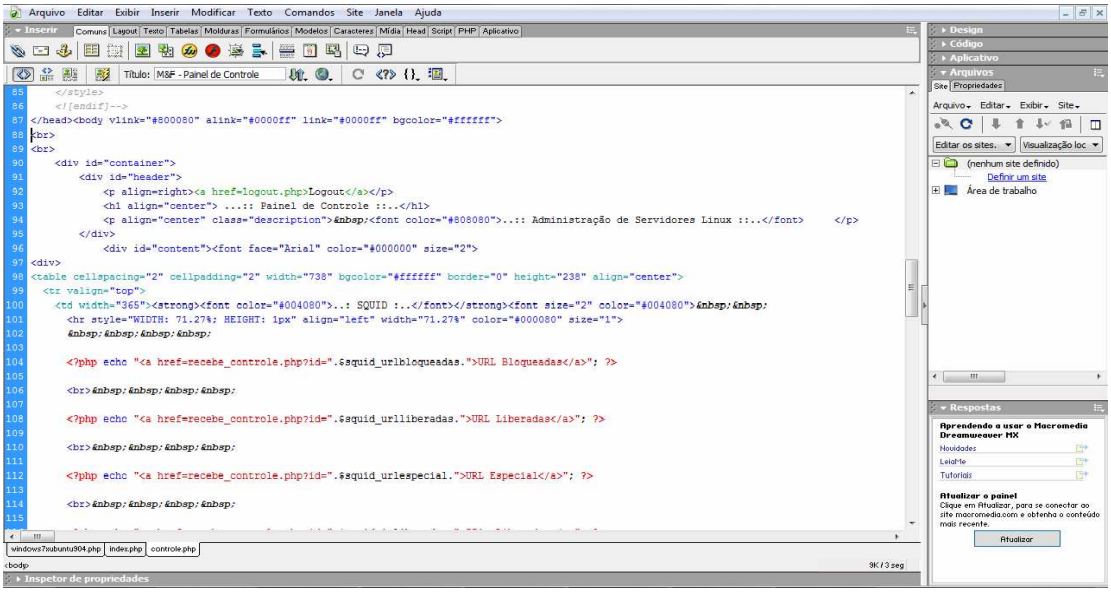


Figura 3 – Modo Código do *Macromedia Dreamweaver MX*

Além dos dois modos de visualização citados, o *Dreamweaver* oferece vários outros recursos que auxiliam no rápido desenvolvimento do protótipo. Dentre esses recursos, pode-se destacar:

- Geração de formulários – permite criar os formulários e os respectivos objetos que vão fazer a interação entre o usuário e o código *PHP* de forma padronizada.
- Geração de tabelas – toda a parte visual da interface, além de alguns componentes funcionais, está organizada utilizando-se de tabelas criadas automaticamente pelo *Dreamweaver*.
- Marcação de *tags* inválidas – permite visualizar todas as *tags HTML* que foram inseridas desnecessariamente na codificação das páginas.

### 2.4.2 EDITPLUS

O *EditPlus* é um editor simples para desenvolvimento de páginas *Web* para a plataforma *Windows* criado pela *ES-Computing*. Além do *HTML* simples, ele suporta o *PHP* e as principais linguagens *Web*.

A figura 4 mostra a interface de desenvolvimento do *EditPlus*:

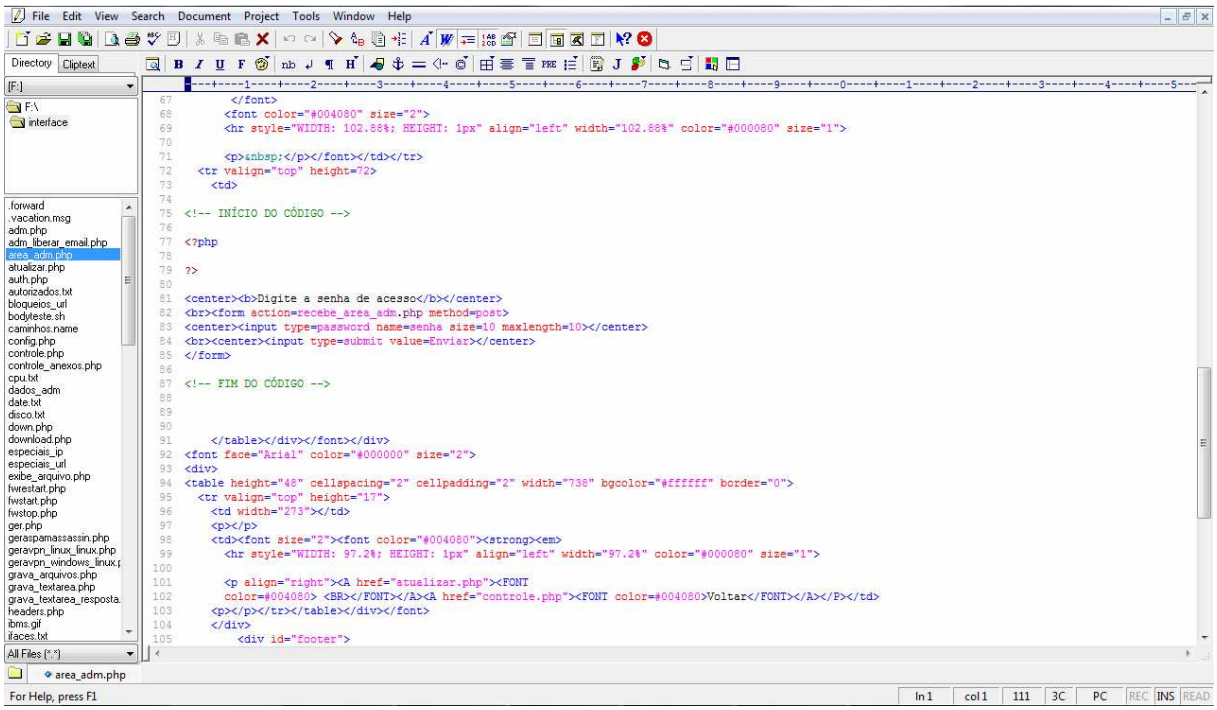


Figura 4 – Interface de desenvolvimento do *EditPlus*

Para o desenvolvimento do protótipo, o *EditPlus* é utilizado para auxiliar na edição das sintaxes *PHP* que compõem a parte funcional da interface. Embora o *Dreamweaver MX* também permita a edição do código *PHP*, o *EditPlus* possui o recurso de sintaxes coloridas, facilitando a edição e organização do código e diminuindo a chance de erros.

A figura 5 mostra parte do código *PHP* do protótipo sendo exibida dentro do *EditPlus*, utilizando o recurso das sintaxes coloridas:

```
1  <?php
2  include ("verifica.php");
3  ?>
4
5  <html><head><title>Informacoes Atualizadas</title></head>
6  <body>
7
8  <?php
9
10
11 echo exec('df -h > disco.txt');
12 echo exec('date > date.txt');
13 echo exec('sudo ifconfig > ifaces.txt');
14 echo exec('lastlog > lastlog.txt');
15 echo exec('sudo mii-tool > mii.txt');
```

Figura 5 – Parte do código *PHP* mostrado no *EditPlus*

### 2.4.3 LINGUAGEM PHP

Atualmente, existem diversas linguagens de programação voltadas para o desenvolvimento *Web*. Dentre elas, foi escolhida a linguagem *PHP* para o desenvolvimento do protótipo por oferecer maior facilidade na sua manutenção e, assim como os serviços gerenciados e o próprio sistema operacional, ser *opensource*.

A linguagem *PHP*, como afirma Moraz (2005), foi desenvolvida em 1994 por Rasmus Lerdorf. Desde a sua criação, é uma linguagem de programação utilizada para criar *sites* dinâmicos, interagindo com formulários, *links* e parâmetros de *URLs*.

O *PHP* é uma linguagem *server-side*, ou seja, ela é executada do lado do servidor. O usuário final não tem acesso ao código *PHP*, somente ao que é gerado por ele. Essa característica torna o *PHP* uma linguagem bastante segura, podendo ser utilizada em aplicações críticas, que manipulam informações confidenciais. (MORAZ, 2005).

A primeira versão do *PHP*, quando foi criada, era conhecida como *Personal Home Page Tools* (Ferramentas para *Homepages* Pessoais). Basicamente, essas ferramentas integravam um interpretador simples, que entendia algumas macros especiais e alguns utilitários de uso comum em *sites* da época, como livros de visitas e contadores de acesso. Ainda em 1994, o interpretador foi reescrito e batizado de *PHP/FI version 2*, onde foi agregada na linguagem o pacote FI ("*Form Interpreter*").

Entre as novas funcionalidades, estava a capacidade de interpretar dados de formulários *HTML*. Lerdorf combinou *scripts* do *PHP* com um interpretador de formulários e adicionou suporte a banco de dados. Depois dessas modificações, o *PHP* cresceu e rapidamente usuários do mundo todo começaram a adicionar-lhe novas funcionalidades.

A partir da versão 3 da linguagem, o interpretador do *PHP* foi reescrito por Zeev Suraski e Andi Gutmans. Boa parte dos códigos foi reescrita, tornando a linguagem mais estável e funcional. Atualmente a linguagem está na versão 5.

Como fator determinante na escolha da linguagem, conforme analisa Moraz (2005), o *PHP* é uma linguagem multiplataforma, ou seja, ela pode ser utilizada em sistemas *UNIX*, *Linux* ou *Windows*, permitindo que a interface gerada pelo protótipo seja acessada independente do sistema operacional utilizado na estação.

# 3. METODOLOGIA

Para que se possa cumprir o desenvolvimento do protótipo dentro de prazos e requisitos pré-estabelecidos, este trabalho segue os principais conceitos do modelo de desenvolvimento da prototipação evolucionária, obedecendo a ordem lógica e as especificações que a metodologia impõe.

## 3.1 Prototipação evolucionária

É fundamental que haja em um processo de criação a definição de métodos de desenvolvimento que auxiliem na estruturação da aplicação. Conforme a necessidade de recursos identificadas pela empresa Silva Informática, fica evidente a importância da criação de um protótipo, que deverá cumprir pelo menos parcialmente as necessidades levantadas pelo cliente.

Segundo Maffeo (1992), um protótipo deve incorporar as características do produto real. Deve exibir as interfaces importantes e executar as principais funções, mesmo que ainda não atenda todos os requisitos que o produto final deverá alcançar.

Para o modelador, ou seja, o responsável pela criação do *software*, o protótipo deve ser capaz de concretizar toda a estrutura conceitual, testando toda a sua especificação de requisitos e assim permitir a sua correção, validação e aperfeiçoamento.

Para o usuário final, o protótipo permite que sejam validados todos os aspectos funcionais que o sistema deve possuir para atender todas as necessidades. Ele permite também que o modelador consiga ver o quão intuitiva está a operação do *software* neste estágio de desenvolvimento (MAFFEO, 1992).

Ainda segundo Maffeo (1992), a prototipação evolucionária considera a análise e especificação dos requisitos desejados a parte mais difícil e mais suscetível a erros de todo o processo de desenvolvimento de um *software*. Muitas vezes é praticamente impossível extrair todos os requisitos que o sistema deve ter sem experimentar versões preliminares do produto final.

Para os critérios estabelecidos na fase de análise do protótipo, fica evidente que a prototipação evolucionária é o método mais adequado, pois permite que sejam

incorporadas as funcionalidades do protótipo no servidor gradualmente, conforme são feitas as adaptações necessárias no sistema operacional.

A figura 6 mostra todas as etapas de modelo de desenvolvimento da prototipação evolucionária:

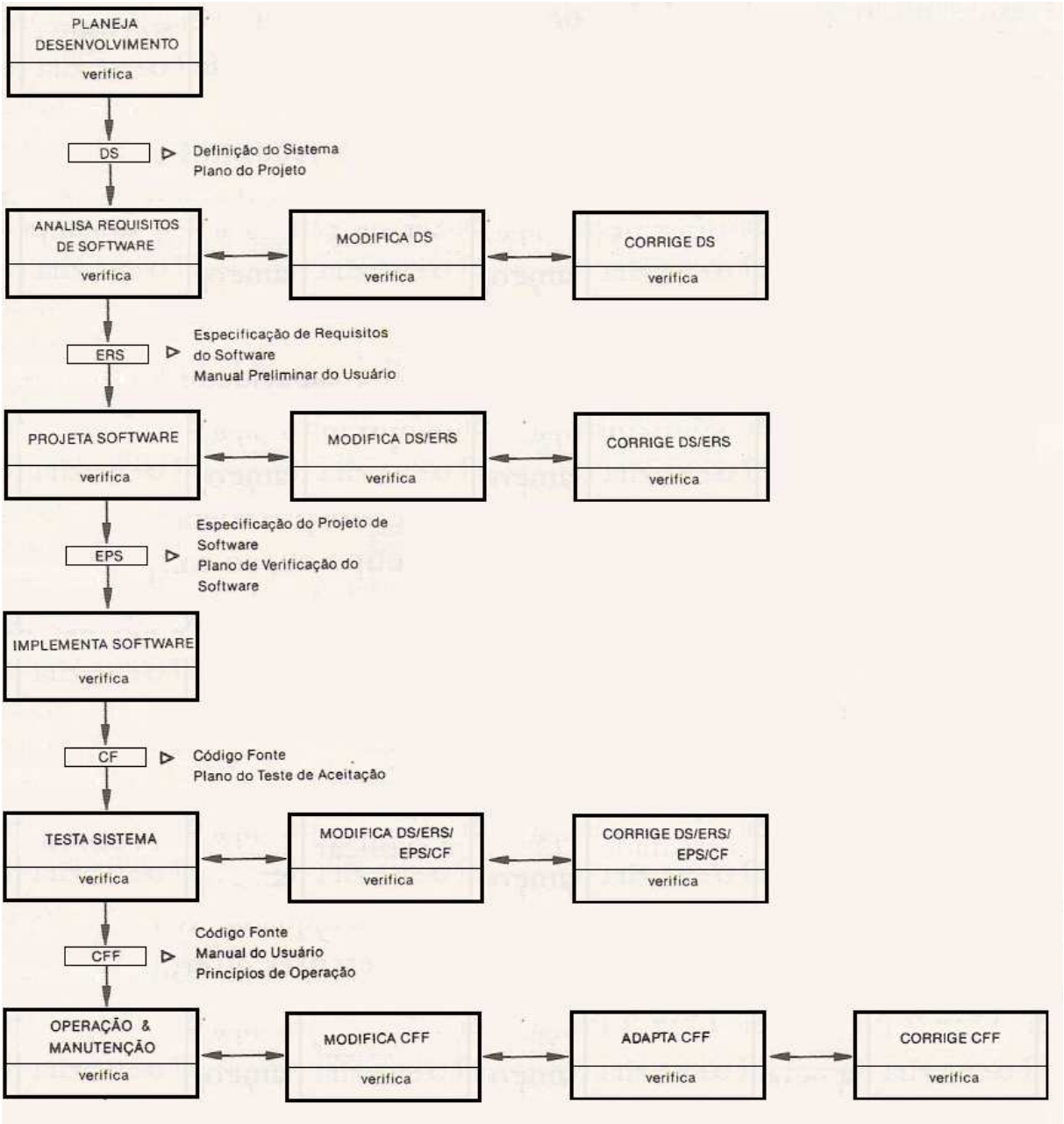


Figura 6 – Seqüência dos eventos na prototipação (MAFFEO, 1992, p. 15)

Nesse modelo de desenvolvimento, conforme analisa Maffeo (1992), os protótipos cobrem cada vez mais requisitos, até que se atinja o produto final desejado. Dessa forma, é possível que os requisitos sejam definidos de forma

progressiva, ou seja, conforme o protótipo evolui, os requisitos podem evoluir e/ou ser redefinidos junto com ele.

Primeiramente, o protótipo terá apenas as funcionalidades básicas requisitadas pelo cliente. Conforme evolui o seu desenvolvimento e ficam mais evidentes as possibilidades de se incorporar novas funcionalidades, o protótipo vai sofrendo pequenas alterações.

Como o protótipo será utilizado por profissionais da área de TI – Tecnologia da Informação, fica mais fácil que o cliente participe ativamente de todos os estágios de desenvolvimento, da fase de análise até a fase de implementação. Isso diminui tempo gasto com verificação e validação ao decorrer do desenvolvimento deste protótipo especificamente, proporcionando uma agilidade ainda maior no processo evolutivo da aplicação.

# 4. ANÁLISE E PROJETO

Para essa fase inicial de análise e projeto, seguindo os conceitos da prototipação evolucionária, especificam-se os requisitos e é criado um esboço do projeto, de acordo com as informações identificadas na empresa Silva Informática.

## 4.1 Especificação dos requisitos

Nessa etapa do processo de desenvolvimento, o objetivo maior é encontrar, mesmo que de forma genérica, todas as funcionalidades que o protótipo deverá possuir. Ainda não há preocupação com a solução técnica que vai ser adotada, porém, será descrita as opções e funcionalidades desejadas pelos usuários.

Como fase inicial do desenvolvimento, algumas informações foram levantadas em relação às funcionalidades que o aplicativo deve conter. Para que isso seja possível, algumas questões tiveram que ser feitas para todos os funcionários da empresa Silva Informática que utilizarão a interface:

- Como é feito hoje o gerenciamento dos servidores *Linux* e seus aplicativos?
- Já é utilizado algum aplicativo com funções similares a do protótipo?
- Que tipo de profissional utilizará a interface gerada pelo protótipo?
- Quantas pessoas têm acesso direto ao servidor e permissão para fazer alterações?
- Quem deverá receber e autorizar ou não as possíveis solicitações feitas pelos usuários comuns que utilizam os recursos da rede?

Baseado nas respostas dessas perguntas é possível ter a noção inicial dos recursos que o protótipo deverá ter. É possível certificar-se também que o objetivo principal do protótipo é ajudar a organizar, agilizar e padronizar o gerenciamento dos servidores *Linux*.

A partir dessa etapa de desenvolvimento, ficam definidos então os principais requisitos que o protótipo deve ter, que são:

- Autenticar os usuários;
- Acessar os arquivos de configuração dos serviços (*Squid, Postfix, SpamAssassin* e *OpenVPN*);

- Criar e manipular arquivos texto dentro do sistema operacional;
- Exibir informações sobre o sistema sem a necessidade de digitação de comandos.

Embora a interface gerada pelo protótipo elimine a necessidade da instalação do modo gráfico no servidor, ainda é necessário que haja uma forma de armazenar dezenas de informações que serão geradas pelo protótipo. A solução trivial para esse problema seria o uso de um banco de dados para trabalhar em conjunto com as páginas *PHP* da interface. Porém, seria necessária a instalação de mais um serviço no servidor somente para que o protótipo funcione, algo que foge do propósito desse protótipo.

Considerando todas as informações até aqui levantadas, ficou definido que o aplicativo, nessa fase inicial do protótipo, deverá ter as seguintes funcionalidades:

- Autenticação dos usuários: para que se tenha acesso a qualquer página da interface, deve-se primeiro autenticar os usuários com *login* e senha. A única exceção é feita para a página onde o usuário comum da rede faz uma solicitação de liberação de *email* ao administrador da rede;
- Gerenciamento dos serviços: após a autenticação, o usuário deve ter acesso a todas as opções de gerenciamento dos serviços de forma simples e em uma única página;
- Área do administrador: o principal responsável pelo gerenciamento dos servidores *Linux* deve ter um local onde estarão disponíveis as funções que somente ele poderá manipular, como a liberação dos *emails* solicitados pelos usuários comuns.
- Informações do sistema: fornecerá as informações básicas do sistema operacional, como uso do disco, as rotas definidas, as interfaces de rede, os usuários existentes, o uso do processador e memória e as portas abertas no servidor;
- Administração dos serviços: iniciar, parar e reiniciar os serviços básicos do servidor, entre eles o *SSH*, o *firewall e* a *VPN*;
- Configurações gerais: onde é definido o caminho que os arquivos de configuração dos serviços estão no sistema;

## 4.2 Desenho do aplicativo

Para a representação gráfica de como deverá funcionar as permissões de acesso do protótipo, será utilizado como referência a figura 7 demonstrada abaixo:

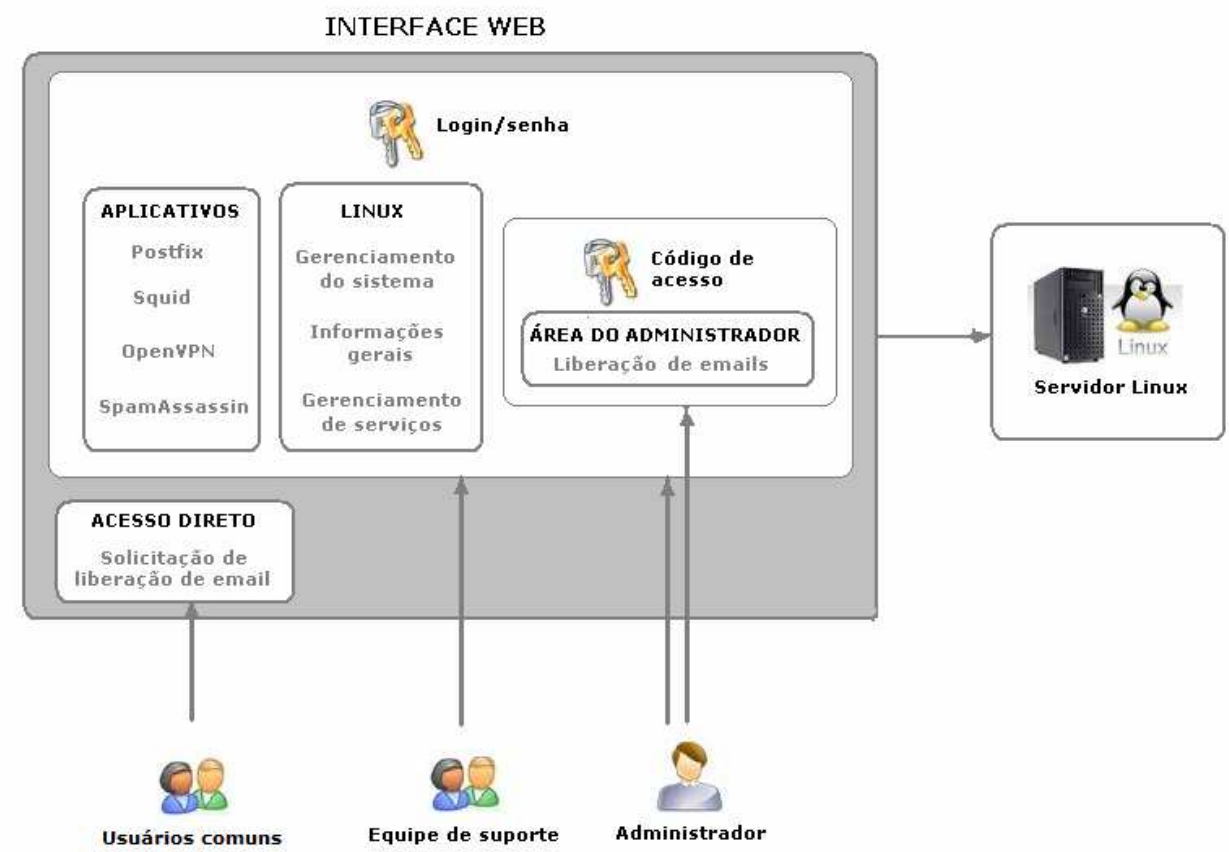


Figura 7 – Restrições de acesso ao servidor *Linux* através da interface

A figura 7 mostra os três tipos de usuário que acessarão o servidor *Linux* através da interface gerada pelo protótipo. O administrador é o principal responsável pelo gerenciamento do servidor, tendo acesso total a todas as funcionalidades do protótipo. A equipe de suporte são os demais funcionários da equipe de TI que terão acesso ao servidor. Os usuários comuns são todos os demais usuários da rede.

## 4.3 Estados do sistema

Mesmo se tratando de uma aplicação que basicamente lê e manipula arquivos texto no servidor *Linux*, fica evidente algumas alterações de estado no

protótipo, bem como a necessidade da transição de estados conforme o usuário acessa a interface.

A figura 8 mostra todas as transições de estado possíveis no protótipo:

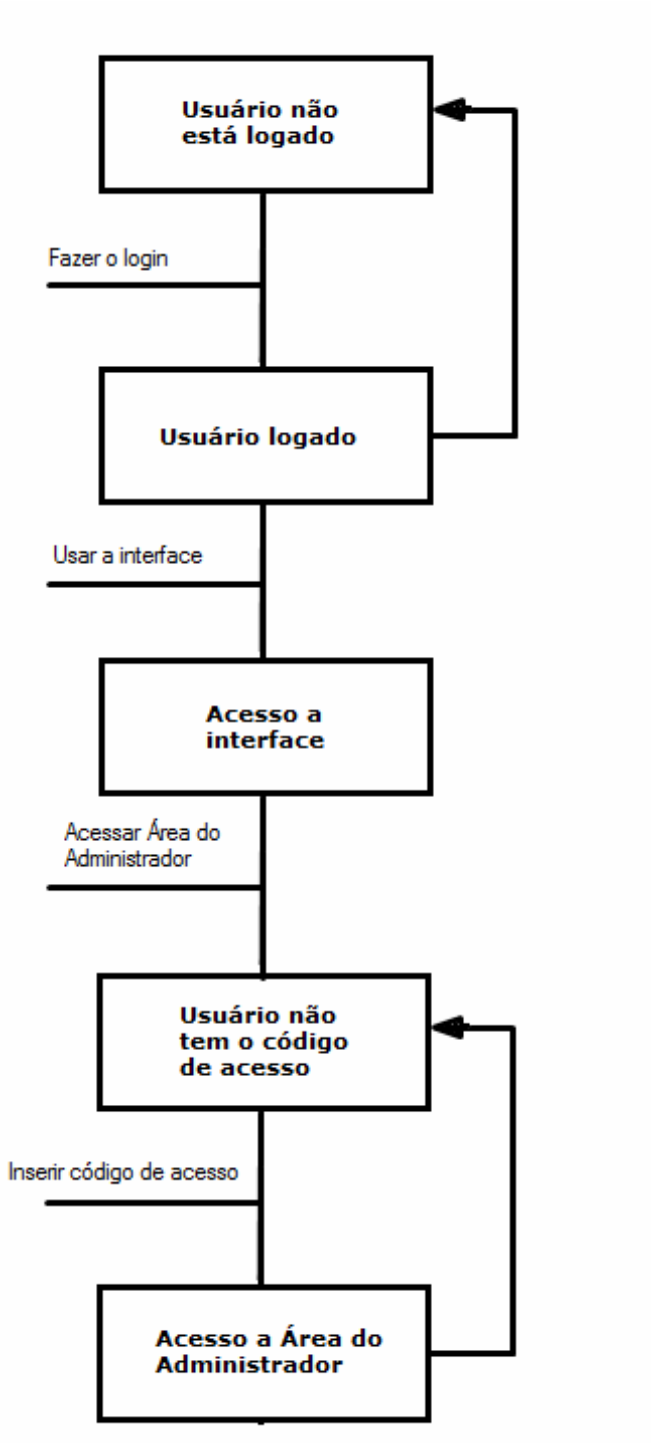


Figura 8 – Diagrama de transição de estados do protótipo

# 5. IMPLEMENTAÇÃO E TESTES

A fase de implementação e testes inclui as etapas de codificação do protótipo, o uso da interface *Web* em uma base de testes simulando o ambiente dos clientes da empresa Silva Informática e o levantamento de todas as alterações que se tornaram necessárias para adaptar o protótipo ao sistema operacional.

## 5.1 Codificação e implementação

Como se trata de um sistema *on-line*, a ser acessado via *browser* em uma rede local, toda a interface é mostrada seguindo o conceito de páginas, como acontece com *sites* da *Internet*.

O objetivo nesta etapa do desenvolvimento do protótipo é, baseado nos dois diagramas mostrados nas figuras 7 e 8 e nos requisitos necessários apontados na fase de análise e projeto, codificar a interface de forma que o servidor seja acessado sem que interfira na segurança do sistema. Para isso, fica definido que:

- A interação entre os formulários *HTML* das páginas, visíveis ao usuário, irão se comunicar com os códigos *PHP* das páginas através do método *POST*. Assim como afirma Niederauer (2004), o método *POST*, ao contrário do método *GET*, não exibe na barra de endereços o conteúdo preenchido pelo usuário nos formulários *HTML*, o que provê mais segurança a aplicação.
- Para cada usuário que se autenticar corretamente para utilizar a interface, será criada uma sessão de 60 minutos no momento em que seu *login* e senha forem aceitos. Isso significa que, mesmo que o usuário tenha permissão para acessar o servidor através da interface, após uma hora será requisitado novamente o seu *login* e senha. Essa é uma medida de prevenção para casos em que o administrador ou alguém da equipe de suporte esqueça de fechar o *browser* depois de utilizar a interface em uma estação da rede. Essa sessão é criada através do código *PHP* contigo na página de validação de *login* e é transparente ao usuário que utiliza a interface.

Neste estágio do protótipo, como a interface por enquanto é utilizada apenas pelos funcionários da empresa Silva Informática, as senhas ainda serão

armazenadas em arquivos não criptografados. Nos próximos ciclos de desenvolvimento da interface, a segurança será o foco e esse assunto será tratado com mais atenção.

### 5.1.1 AUTENTICAÇÃO

Assim que é acessada a interface, é carregada sua página inicial solicitando o *login* e a senha, como demonstra a figura 9:


Figura 9 – Tela de *login* da interface

Após digitar o *login* e a senha, os dados digitados no formulário são confrontados com os dados cadastrados em um arquivo de configuração do protótipo onde ficam armazenados todos os usuários e suas senhas. Na página seguinte, caso o usuário digite um *login* e senha válidos, é criada a sessão conforme demonstra a figura 10:

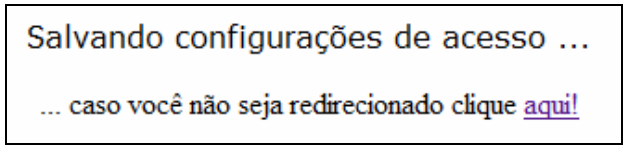


Figura 10 – Tela onde é criada a sessão do usuário

Depois que é criada a sessão, o usuário passa a ter acesso ao menu principal da interface.

A figura 11 demonstra tela inicial da interface após o *login*:

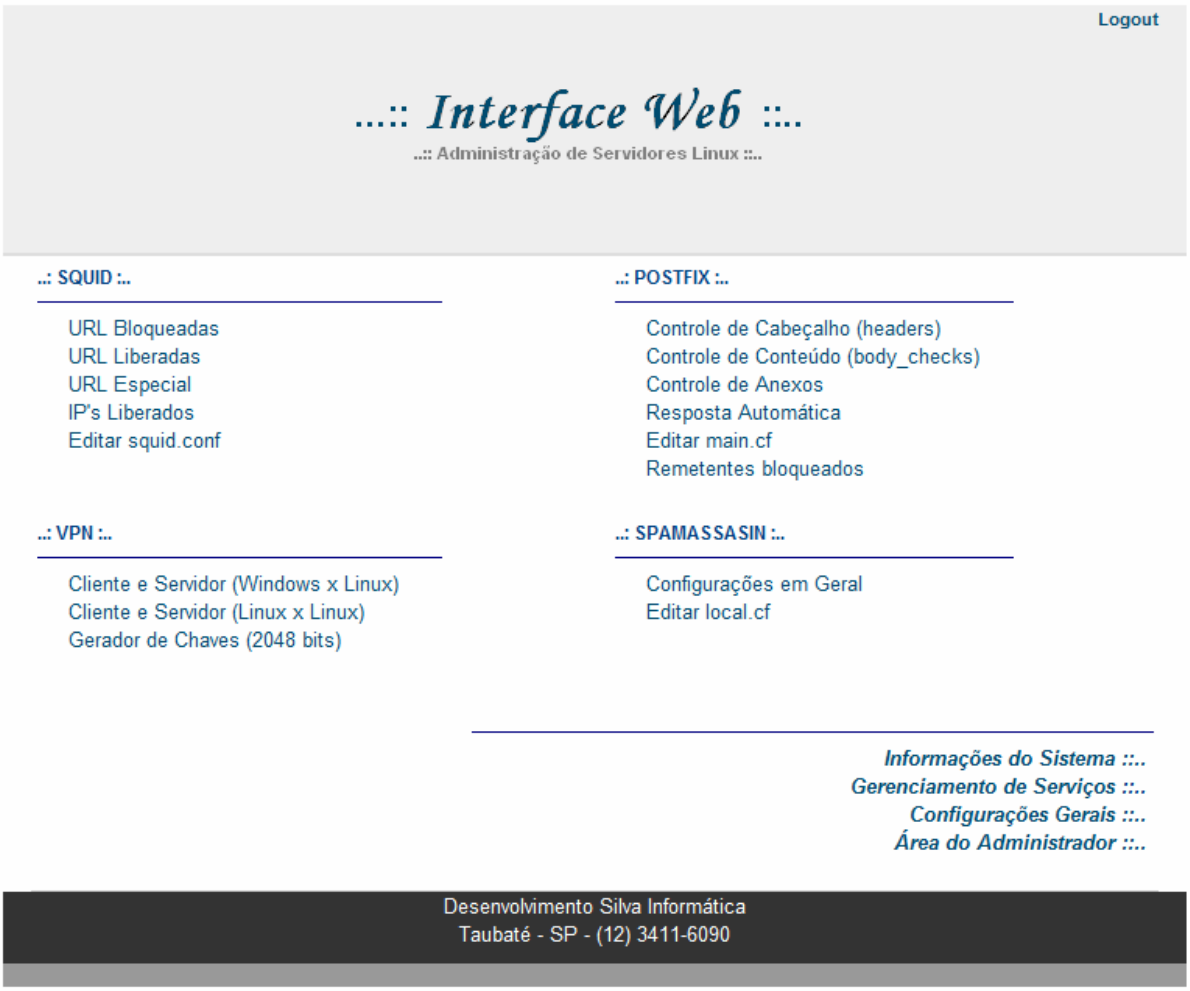


Figura 11 – Menu principal da interface

### 5.1.2 MENU SQUID

Para essa etapa do protótipo, foram criadas cinco funcionalidades no gerenciamento do *Squid*.

Como já mencionado anteriormente, o servidor *proxy* Squid pode ser todo estruturado utilizando-se de listas de controle de acesso, ou *ACLs*. De acordo com Ricci e Mendonça (2006), o *Squid* se baseia em regras de acesso definidas no arquivo squid.conf para bloquear, retardar, registrar e autorizar acessos de acordo

com diversos fatores, como horário, origem e destino da requisição, endereços *MAC*, disponibilidade de banda, entre outros. As *ACLs* permitem ao administrador dividir os usuários em grupos, de forma que as regras passam a valer para um grupo de usuários e não para cada usuário.

Além de agrupar usuários, as *ACLs* permitem também agrupar os endereços que serão tratados pelo *Squid*, de forma que não seja necessário adicionar uma nova linha de comando para cada endereço bloqueado ou liberado.

O protótipo permite que, através da interface, sejam editadas quatro *ACLs* básicas, que são:

- *URL* Bloqueadas: armazena todos os endereços que serão bloqueados o acesso;
- *URL* Liberadas: armazena todos os endereços que serão liberados o acesso;
- *URL* Especial: armazena os endereços que são exceção à regra, como sites onde se deve liberar o acesso temporariamente;
- *IPs* Liberados: armazena todos os endereços *IP* da rede local onde não será feito o filtro de endereços.

As alterações feitas nas quatro *ACLs* acima serão salvas em um arquivo texto que é salvo no mesmo diretório onde está o arquivo squid.conf.

É importante ressaltar que, conforme afirma Ricci e Mendonça (2006), as diretivas do arquivo squid.conf são interpretadas de cima para baixo, ou seja, na ordem em que elas estão escritas no arquivo de configuração do *Squid*.

Esta definição é muito importante para o bom gerenciamento do *Squid* através da interface, pois mesmo que as *ACLs* estejam editadas corretamente, é muito provável que o administrador precise adicionar regras manualmente no arquivo squid.conf. Para suprir essa necessidade, além das quatro *ACLs* citadas acima, é possível também editar o arquivo squid.conf diretamente da interface, como mostra a figura 12. O *layout* da interface para edição das *ACLs* e do arquivo squid.conf é o mesmo.

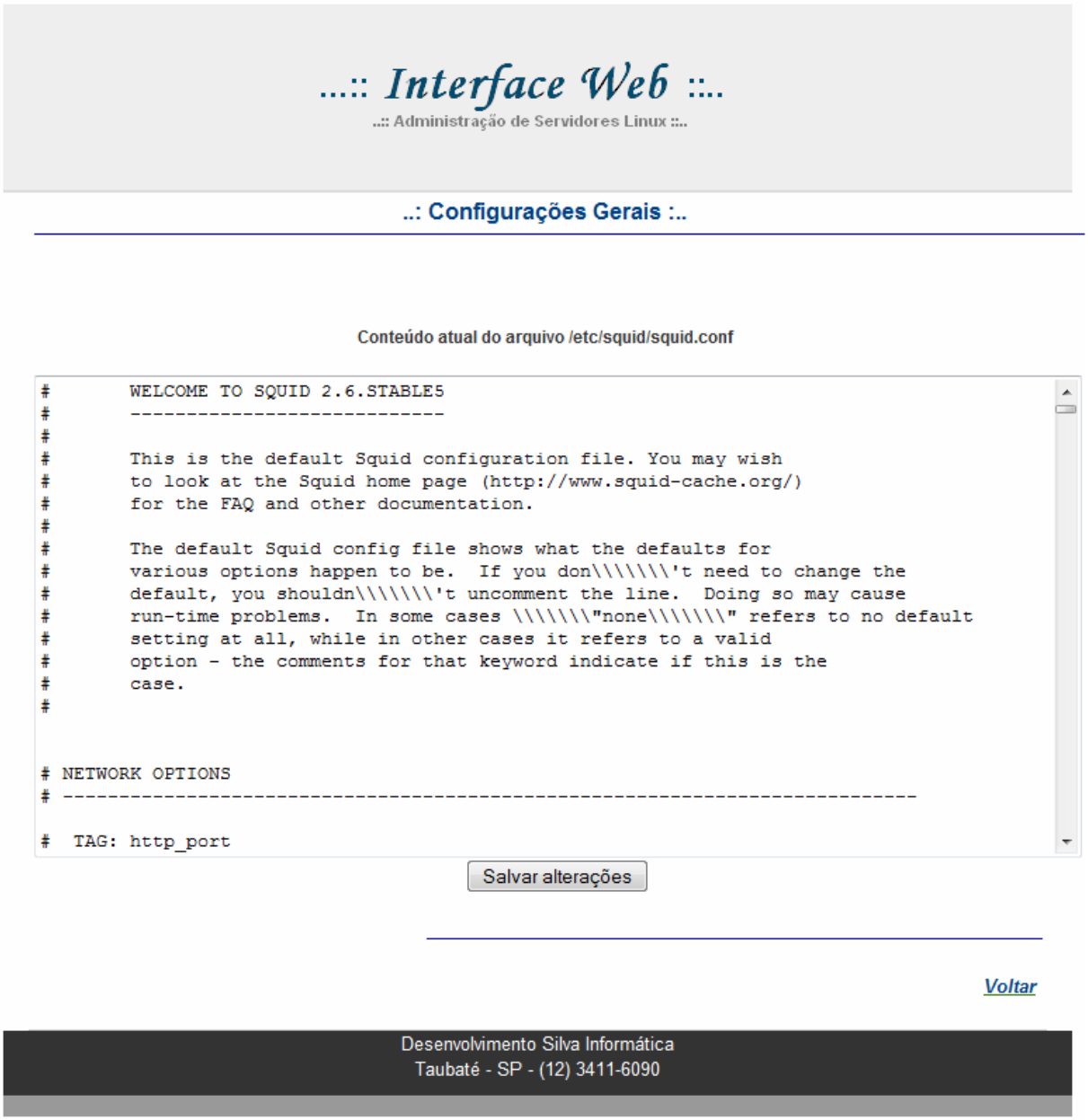


Figura 12 – Layout da tela de edição das *ACLs* e do arquivo squid.conf

### 5.1.3 MENU OPENVPN

Segundo Vasques e Schuber (2008), existem basicamente dois tipos de *VPN*: *client-to-gateway* e *gateway-to-gateway*. No primeiro caso, o tunelamento da *VPN* é estabelecido entre o *gateway* de uma rede e um computador comum. No segundo, nas duas pontas do tunelamento está o *gateway* da rede, interligando assim duas redes locais distintas.

Considerando estas informações, o desenvolvimento do protótipo se baseou em duas situações para o gerenciamento do *OpenVPN* através da interface:

- *VPN Windows – Linux*: para a configuração de uma *VPN client-to-gateway*, considera-se que no *gateway* da rede esteja o *Debian Linux* e do outro lado do tunelamento haja um computador com sistema operacional *Windows*.

A figura 13 demonstra a tela onde é possível gerar os arquivos de configuração para uma *VPN Windows – Linux*:


Figura 13 – Tela de criação dos arquivos para *VPN Windows - Linux*

Nesse caso, o *OpenVPN* gera um arquivo com a extensão .ovpn para uso na estação com o *Windows* e um arquivo com a extensão .conf no *Debian Linux*.

- *VPN Linux – Linux*: para uso em *VPNs gateway-to-gateway*, considera-se que nos dois lados do tunelamento há o sistema operacional *Debian Linux* instalado.

A figura 14 demonstra a tela onde é configurada a *VPN Linux-Linux*:


Figura 14 – Tela de criação dos arquivos para *VPN Linux – Linux*

Para estabelecer *VPNs Linux-Linux*, considerando que em umas das pontas do tunelamento está o *Debian Linux*, serão gerados dois arquivos de extensão .conf.

Os arquivos de configuração em ambos os casos serão criados dentro do diretório */home/dados*. No caso da *VPN Windows – Linux* será criado também um arquivo com extensão .bat para ser executado na estação com *Windows*. Ele deverá ser executado para que seja possível estabelecer o tunelamento.

Para os dois tipos de *VPN*, como é possível verificar nas figuras 13 e 14, cria-se um arquivo para o cliente da *VPN* e outro para o servidor. Isso significa que, o local onde está instalado o *OpenVPN* é o servidor e a outra ponta do tunelamento será o cliente.

Além dos arquivos de configuração, para que o tunelamento seja estabelecido é necessária também uma chave criptográfica, conforme afirma Montenegro e Ribeiro (2008). Através da interface, é possível a geração da chave criptográfica pela opção Gerador de Chaves (2048 *bits*). A chave será criada, armazenada dentro do diretório onde está instalado o *OpenVPN* e uma página disponibilizando o *download* da chave aparecerá na interface, como demonstra a figura 15. Uma página com o mesmo *layout* também é mostrada para o *download* dos arquivos de configuração.


Figura 15 – Tela onde é disponibilizada o *download* da chave criptográfica

É importante citar que quem estabelece a *VPN* entre os dois pontos é o *OpenVPN*, e não o protótipo. O protótipo se encarrega apenas de gerar os arquivos de configuração e a chave criptográfica. Isso significa que através do protótipo é possível gerar todos os arquivos necessários para o estabelecimento de uma *VPN* mesmo que ele não esteja sendo usado em nenhuma das pontas do tunelamento. Basta haver o *OpenVP*N instalado em uma das pontas.

### 5.1.4 MENU POSTFIX

Assim como no *Squid*, o *Postfix* permite criar arquivos de configuração semelhante às *ACLs*, agrupando diretivas para funcionar em conjunto com o seu arquivo de configuração main.cf.

As opções Controle de Cabeçalho, Controle de Conteúdo e Controle de Anexos disponíveis na interface irão gerar os arquivos texto que serão referenciados no arquivo main.cf.

Na primeira e terceira opções, como ilustra a figura 16, o usuário irá digitar o assunto ou o tipo de anexo que serão bloqueados utilizando um desses três parâmetros do *Postfix*. Suas definições, segundo Trigo (2007), são:

- REJECT: rejeita a mensagem inteira
- IGNORE: rejeita apenas o cabeçalho ou o anexo
- DISCARD: rejeita a mensagem de forma silenciosa, ou seja, sem aviso.

A figura 16 mostra a tela de configuração do *Postfix* onde define-se as extensões de arquivo que serão bloqueadas:


Figura 16 – Tela de Controle de Anexos do Postfix

Na segunda opção, Controle de Conteúdo, é possível inserir através da interface todas as regras para o controle do corpo da mensagem. Como se trata de um arquivo extenso e que pode receber vários tipos de modificações, ficou definido que nessa fase do protótipo será disponibilizado apenas a edição manual do arquivo texto, sem a inserção automática de parâmetros pelo protótipo.

A figura 17 mostra o layout da página de edição do arquivo texto, onde ele já é criado dentro do diretório do *Postfix* pelo protótipo. Nessa figura, foi dado ao arquivo o nome *body_checks*, porém esse nome fica a critério de quem gerencia o servidor.


Figura 17 – Tela de Controle de Conteúdo do *Postfix*

Além das três opções de controle do *Postfix* disponíveis no protótipo, através da interface o administrador e a equipe de suporte poderão configurar respostas automáticas e redirecionamento. Essa funcionalidade é normalmente usada em casos em que uma pessoa não poderá responder seus *emails* por um período de tempo e deseja que o remetente receba uma mensagem notificando essa situação. A mensagem enviada pelo remetente também pode ser redirecionada para uma outra conta de *email*, como demonstra a figura 18.


Figura 18 – Tela de configuração da Resposta Automática

Para que essa página consiga interagir com o *Postfix*, o protótipo irá manipular os arquivos .forward e vacation.msg, que segundo Trigo (2007), são os dois responsáveis pela configuração das respostas automáticas e redirecionamentos de todas as contas de *email*.

Por fim, utilizando o mesmo *layout* da página de Controle de Conteúdo, é possível manipular o arquivo *access_sender*, que também é referenciado no arquivo main.cf para indicar os remetentes que serão bloqueados pelo *Postfix*. Assim como foi feito no *Squid*, e tendo em vista a quantidade de parâmetros que podem ser inseridos no arquivo main.cf, o protótipo disponibiliza através da interface a edição manual do arquivo main.cf.

A figura 19 demonstra o *layout* da tela de edição do arquivo main.cf:

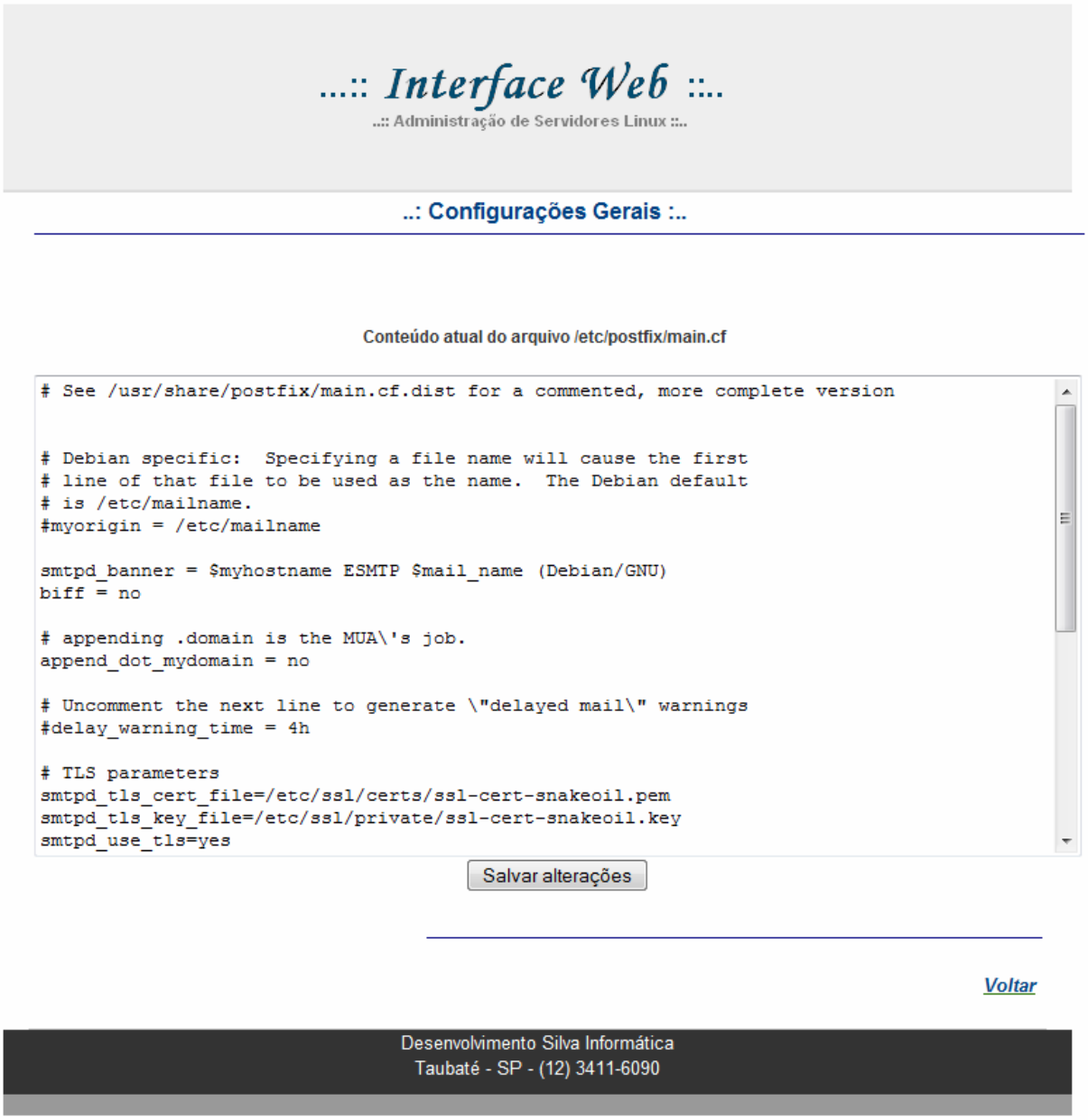


Figura 19 – Tela de edição do arquivo main.cf

### 5.1.5 MENU SPAMASSASSIN

Segundo Ferreira (2003), o *SpamAssassin* é todo codificado na linguagem *Perl* e pode ser totalmente utilizado através do *shell* do *Linux*. Essa característica, aliada a simplicidade do arquivo de configuração do *SpamAssassin*, permite que ele seja totalmente gerenciado através da interface.

Para o gerenciamento do *SpamAssassin*, foram criadas duas opções na interface: Configurações em Geral e Editar local.cf.

Na prática as duas opções vão manipular o arquivo local.cf, porém, na primeira opção o administrador ou a equipe de suporte poderão inserir valores de forma que o protótipo se encarregue de preencher os parâmetros de configuração, deixando o aplicativo pronto para ser usado.

Na segunda opção, como demonstra a figura 20, a interface concede acesso direto ao arquivo local.cf, podendo inserir os parâmetros de configuração manualmente.


Figura 20 – Tela de edição do arquivo local.cf

Os parâmetros que poderão ser inseridos automaticamente por intermédio do protótipo são quatro:

- Assunto: se o *SpamAssassin* identificar uma mensagem como spam, nesse campo o administrador pode definir qual mensagem aparecerá sinalizando ao destinatário que a mensagem é um *spam*. O texto padrão é *****SPAM*****;
- Pontuação mínima: define a pontuação que a mensagem precisa alcançar para ser identificada como *spam*. O valor padrão é 5;
- Relatório: o *SpamAssassin* permite que as mensagens identificadas como *spam* sejam enviadas para o seu destino em um arquivo anexado. O administrador pode, então, desativar essa opção, deixá-la ativada ou fazer com que a mensagem original seja enviada como anexo em um arquivo de texto simples.
- Redes confiáveis: o administrador pode definir, quais redes não precisarão passar pelo filtro do *SpamAssassin*. Assim, se o remetente da mensagem for de uma rede confiável, a mensagem não será marcada como *spam* mesmo que ultrapasse a pontuação mínima.

A figura 21 demonstra a tela onde é possível configurar as opções citadas acima:


Figura 21 – Tela Configurações do *SpamAssassin*

Além dos quatro principais parâmetros disponíveis para configuração do *SpamAssassin* através da interface, é possível adicionar outros parâmetros de configuração do filtro *bayesiano* que o *SpamAssassin* utiliza. Essas configurações, embora importantes, não são fundamentais para o funcionamento do aplicativo, pois como afirma Ferreira (2003), seus valores padrão são normalmente os mais utilizados. No entanto, caso o administrador queira alterar esses valores, ele o poderá fazer editando manualmente o arquivo local.cf.

### 5.1.6 DEBIAN LINUX

A opção de gerenciar serviços básicos do sistema operacional foi desenvolvida voltada principalmente para uso da equipe de suporte. Considerando que o administrador é o profissional com maior conhecimento sobre o funcionamento do *Debian Linux*, é provável que ele continue acessando o servidor da maneira convencional, ou seja, através do *shell*.

Para o gerenciamento das funções básicas do *Debian Linux* através da interface, foram criadas duas opções: Informações do sistema e Gerenciamento e serviços.

A primeira opção disponibiliza informações do estado do sistema operacional naquele exato momento. Essas informações podem ser extraídas através de simples comandos no *shell* do *Debian Linux*, como afirma Pritchard *et al.* (2007). Porém, para que isso seja possível, talvez seja necessário o usuário ter privilégios de administrador, o que pode expor o servidor a problemas mais sérios de segurança.

Para cada informação acessada, o protótipo utiliza o mesmo comando que seria utilizado no *shell* e extrai o resultado para um arquivo texto. Este arquivo está localizado no mesmo diretório onde está o protótipo e, portanto, através da interface é possível visualizá-lo.

A figura 22 demonstra a tela onde é possível acessar todas as informações relacionadas ao sistema operacional disponíveis na interface:


Figura 22 – Tela Informações do Sistema

A segunda opção permite iniciar e parar alguns dos principais serviços do *Debian Linux*. Para essa etapa de desenvolvimento do protótipo, é possível alterar o estado de cinco serviços: *squid*, *postfix*, *spamassassin*, *openvpn* e *ssh*, conforme demonstra a figura 23.

O serviço *ssh*, segundo Ferreira (2003), é o responsável pelo funcionamento do *Secure Shell*, que permite o acesso ao *shell* do *Linux* remotamente.

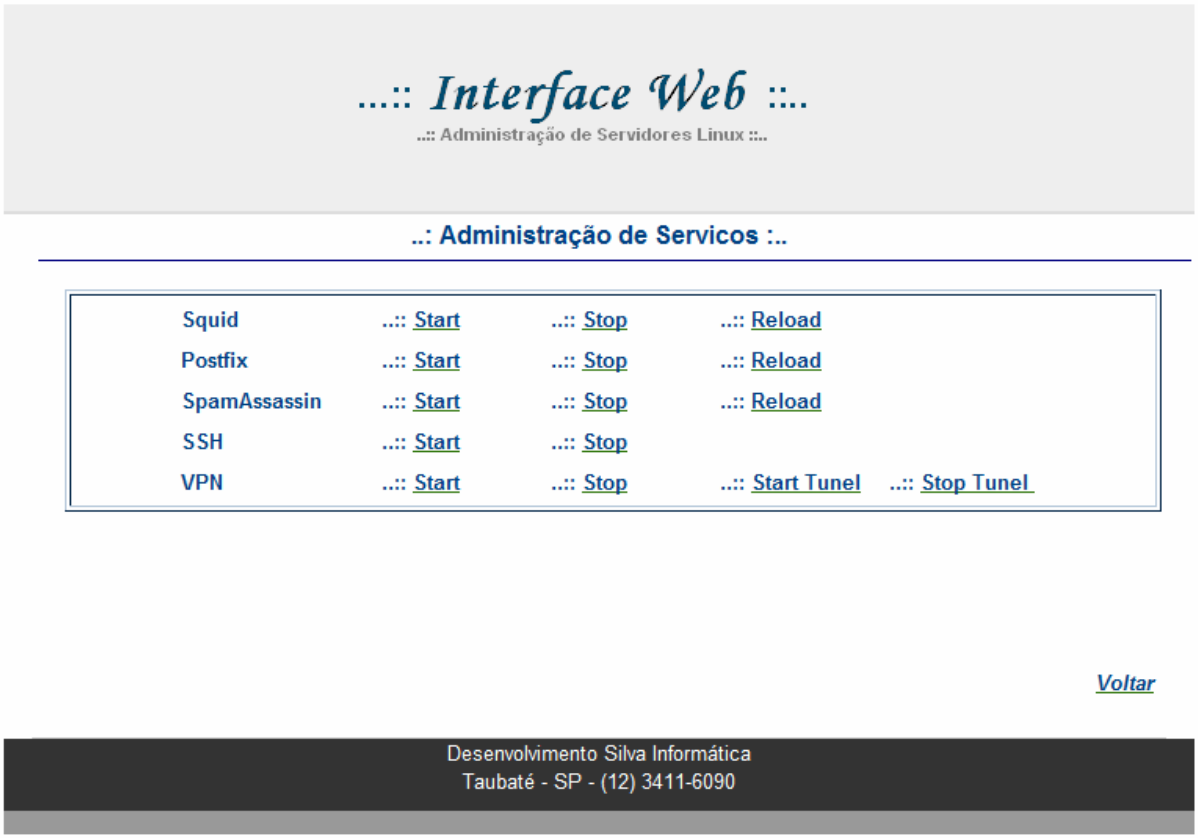


Figura 23 – Tela de Gerenciamento de Serviços

Para que o protótipo possa criar os seus arquivos texto, é necessário que haja permissão de leitura e escrita no diretório onde eles ficam armazenados. Para não comprometer a segurança do sistema operacional, conforme afirma Pritchard *et al.* (2007), os arquivos precisam ser protegidos contra modificações feitas por outros usuários e, em alguns casos, precisam ser mantidos completamente privados.

Baseado nessa afirmação, no início da implementação do protótipo, foi criado um novo usuário no *Debian Linux*, denominado dados. Os arquivos texto que armazenam as informações temporárias do protótipo ficam todos dentro do diretório */home/dados*, que é criado automaticamente pelo sistema operacional no momento da criação do usuário dados.

Após a criação do usuário dados, a segunda etapa da implementação é prover as devidas permissões para o usuário *www-data*. Esse usuário, segundo Veiga (2006), tem permissão restrita ao sistema e é utilizado somente para executar o *Apache*, ou seja, será o usuário *www-data* que irá executar todas as operações do protótipo.

Para que o protótipo possa manipular os arquivos texto do diretório */home/dados*, a primeira mudança que deve ser feita no *Debian Linux* é conceder ao usuário *www-data* acesso de leitura e escrita a todo o conteúdo desse diretório. Essa permissão só pode ser concedida acessando o servidor pelo *shell*.

O próximo passo da implementação é incluir o usuário *www-data* na lista de usuários que podem usar o *sudo*. Segundo Pritchard *et al.* (2007), o *sudo* permite executar vários comandos com privilégios de administrador do sistema, sem que seja necessário estar logado como administrador.

O comando *sudo* é fundamental para o protótipo, pois é com ele que o protótipo vai manipular os arquivos de configuração dos serviços gerenciados sem que seja necessário atribuir permissões de leitura e escrita para o usuário *www-data* no diretório onde ficam esses arquivos.

Para adicionar o usuário *www-data* na lista de usuários que utilizam o *sudo*, é necessário editar o arquivo */etc/sudoers*, deixando o arquivo da seguinte maneira:

```
Default         env_reset
root            ALL=(ALL) ALL
www-data        ALL=NOPASSWD:/sbin/ifconfig
www-data        ALL=NOPASSWD:/sbin/mii-tool
www-data        ALL=NOPASSWD:/sbin/route -n
www-data        ALL=NOPASSWD:/sbin/postqueuei
www-data        ALL=NOPASSWD:/sbin/chown
```

### 5.1.7 MENU ÁREA DO ADMINISTRADOR

A área do administrador foi projetada para que o principal responsável pela administração da rede tenha acesso exclusivo a operações que só ele tem autorização de executar. Para essa etapa de desenvolvimento do protótipo, o objetivo é criar esse espaço dentro da interface e, conforme solicitação do cliente, adicionar novas funcionalidades dentro dele no futuro.

Para acessar a área do administrador, o usuário que usa a interface deve digitar uma nova senha de acesso. Essa senha fica armazenada em um arquivo de configuração do protótipo junto com as informações do administrador do sistema.

Para o primeiro acesso, a senha de acesso é a palavra senha, sendo que logo após a autenticação ela pode ser alterada, como mostra a figura 24.


Figura 24 – Tela da área do administrador após a autenticação

### 5.1.8 SOLICITAÇÃO DE LIBERAÇÃO DE EMAIL

Para essa etapa de desenvolvimento do protótipo, além da possibilidade do administrador inserir os seus dados pessoais e definir uma nova senha de acesso, foi criada uma funcionalidade dentro da área do administrador: a visualização das solicitações de liberação de *emails*, feita pelos usuários comuns da rede.

Essa funcionalidade foi criada conforme solicitado pela empresa Silva Informática durante o desenvolvimento do protótipo. Muitos clientes da empresa reclamam de problemas com o envio de mensagens para determinados endereços de *email*. Para o desbloqueio desses endereços, o usuário precisa comunicar o

administrador que nem sempre está disponível para fazer o desbloqueio naquele momento.

Para solucionar esse problema, uma página é gerada pelo protótipo sem a necessidade de autenticação pelo usuário, onde o usuário insere o domínio ou um endereço de *email* específico para liberação, como mostra a figura 25.


Figura 25 – Tela de solicitação de liberação de *email*

A solicitação do usuário é gravada em um arquivo texto no mesmo diretório do protótipo. Esse arquivo é lido e interpretado, de forma que todas as solicitações só podem ser acessadas através do *link* dentro da área do administrador, como mostra a figura 24.

As solicitações são listadas para o administrador por ordem de chegada, informando o usuário que fez a solicitação, seguido pelo domínio e/ou endereço de *email.* O administrador deve, então, marcar as solicitações que ele autoriza a liberação e clicar no botão que executa o comando, como mostra a figura 26. Feito isso, a lista de solicitações pendentes fica automaticamente vazia.


Figura 26 – Tela com as solicitações pendentes feitas pelos usuários

## 5.2 Testes

No modelo de desenvolvimento da prototipação evolucionária, segundo Maffeo (1992), o protótipo pode ser usado para treinamento do usuário e teste do sistema. Para a fase de testes, foi utilizado um servidor com as mesmas características dos servidores gerenciados pela empresa Silva Informática. Nele foi instalado o sistema básico do *Debian Linux* 4.0 e os quatro serviços gerenciados pelo protótipo, além do *Apache*.

Antes de implementar o protótipo nos clientes da empresa Silva Informática, a empresa optou por utiliza-lo em seus servidores de email e Internet, de forma que o administrador e a equipe de suporte possam se adaptar a interface, agilizando as verificações e validações.

### 5.2.1 BROWSERS SUPORTADOS

Logo no início dos testes do protótipo foi constatado que alguns *browsers* não carregam o *layout* da interface da maneira que ela foi projetada. A situação mais crítica foi com o *browser Internet Explorer* nas versões 6 e 7, onde mesmo o usuário inserindo seu *login* e senha corretamente, o *browser* não redireciona para a página com o menu principal da interface.

O *browser Mozilla Firefox* 3.5 foi o único que carregou o *layout* corretamente e manteve o protótipo funcional. Os *browsers Opera* 9.6, *Konqueror* 4.0, *Internet Explorer* 8 e *Google Chrome* 1.0 apresentaram mudanças significativas no *layout* da interface, porém, o protótipo continuou funcional.

# 6. AVALIAÇÃO

Seguindo os passos conforme o modelo de desenvolvimento da prototipação evolucionária, o protótipo alcança a etapa de verificação e validação. Assim como afirma Maffeo (1992), a verificação serve para confirmar que o protótipo está sendo desenvolvido da maneira correta, atendendo as especificações. A validação permite que o cliente avalie se o protótipo tem as funcionalidades que se imaginava no início do desenvolvimento.

Esta versão do protótipo foi utilizada pelos funcionários da empresa Silva Informática por aproximadamente dois meses. Durante esse período, foi possível apontar os pontos positivos e negativos do protótipo, que serão mencionados a seguir.

## 6.1 Retornos positivos

Após os dois meses de uso do protótipo pelos funcionários da empresa Silva Informática, ficou constatado que, conforme era esperado:

- As opções disponíveis para gerenciamento do *Squid* funcionaram como esperado. Tanto as *ACLs* quanto o arquivo squid.conf puderam ser editados diretamente da interface sem apresentar erros de permissão leitura e escrita;
- As opções do *OpenVPN* funcionaram corretamente. Tanto a criação quanto o *download* dos arquivos de configuração e da chave criptográfica foram feitos diretamente pela interface;
- As duas opções de edição do arquivo local.cf do *SpamAssassin* disponíveis na interface funcionaram sem problemas;
- As opções disponíveis no item Informações do sistema responderam da forma esperada, agilizando o acesso às informações básicas do sistema operacional assim como o gerenciamento dos serviços do *Debian Linux* e a página de Configurações gerais da interface;
- Área do administrador funcionou corretamente.

## 6.2 Retornos negativos

Em alguns aspectos o protótipo se comportou diferente da maneira esperada. Para esses aspectos, o protótipo retorna para as fases anteriores de desenvolvimento para manutenção desses problemas.

De acordo com os funcionários da empresa Silva Informática, os seguintes itens devem ser melhorados:

- Os serviços do *Postfix* não puderam ser gerenciados totalmente através da interface. Os arquivos gerados pelas opções de Controle de Conteúdo, Controle de Cabeçalho, Controle de Anexos, Remetentes Bloqueados e Resposta Automática não interagiram com o arquivo main.cf, sendo necessário referenciá-los manualmente no arquivo main.cf;
- Embora a Área do administrador tenha funcionado corretamente, as liberações de *emails* ainda têm que ser feitas manualmente pelo administrador da rede, devido às falhas apresentadas no gerenciamento do *Postfix* através da interface;

## 7. CONCLUSÃO

Finalizando este ciclo de desenvolvimento do protótipo, o próximo passo é aprimorar as deficiências apontadas pelos funcionários da empresa Silva Informática durante o período de testes.

Pode-se afirmar que o protótipo obteve êxito em grande parte dos requisitos demandados nesse estágio. O principal responsável por esse êxito foi a constante participação do cliente no desenvolvimento, diminuindo drasticamente o tempo gasto com a verificação e validação do protótipo.

Embora a questão da segurança do sistema sempre fosse levada em consideração no desenvolvimento do protótipo, para as próximas etapas, ela deverá ser o foco, de forma que sua implementação seja menos impactante.

Ficou constatado que este protótipo já se encontra em um estágio avançado de desenvolvimento, estando bem próximo do produto final idealizado. O uso de uma interface *Web* para gerenciamento dos serviços básicos do sistema mostrou-se eficaz e de fácil assimilação pelos funcionários da área de TI. Além disso, não foi necessário, como se esperava, que fosse feita nenhuma aquisição ou alteração de equipamentos para melhorar o gerenciamento do servidor *Linux*

Com base nos pontos positivos e negativos levantados durante o período de testes do protótipo, algumas melhorias e mudanças já puderam ser apontadas para as próximas versões do protótipo. A empresa Silva Informática usa atualmente uma versão adaptada do *Debian Linux*, alterada por eles para uma instalação mais fácil e rápida. Como sugestão, a integração da interface *Web* a essa versão modificada do *Debian Linux* representaria para a empresa maior agilidade na implementação e controle maior dos servidores de seus clientes.

Também como melhoria futura, os serviços gerenciados através da interface poderiam ser divididos em módulos. Assim, seria adicionado ao protótipo apenas os serviços que o sistema operacional possui naquele momento.

Por fim, a ser decidido pela empresa, a interface se mostrou funcional apenas no *browser Mozilla Firefox*. Adaptações ao código poderão ser feitas para que a interface tenha o mesmo comportamento nos demais *browsers*, porém, esse poderia ser também mais um obstáculo para que usuários não autorizados utilizem a interface, já que se imagina que nas estações da rede é utilizado o sistema operacional *Windows* com o *browser Internet Explorer*.

## REFERÊNCIAS BIBLIOGRÁFICAS

GORAYEB, I. L.; JÚNIOR, A. F.; CAPELLI, W. Y. **Usabilidade e acessibilidade no Software Livre baseado em perfis de usuários**, Belém. Disponível em: <http://www.unibratec.com.br/anaisdecongresso/diretorio/UFPA_ILG%20Revisado.doc> Acesso em: 17 mar. 2009.

FERREIRA, R. E. **Linux – Guia do Administrador do Sistema 1ª edição**. Rio de Janeiro: Novatec Editora, 720 p., 2003.

FILHO, W. P. P. **Engenharia de Software. Fundamentos, métodos e padrões 2ª edição**. Rio de Janeiro: Editora LTC, 600p., 2003.

MAFFEO, B. **Engenharia de Software e Especificação de Sistemas**. Rio de Janeiro: Editora Campus, 484p., 1992.

MONTENEGRO, E. ; RIBEIRO, R. **VPN e tunelamento**. Porto Velho. Disponível em: <http://abusar.org/manuais/VPN-alan-rafael.pdf> Acesso em: 03 nov. 2008.

MORAZ, E. **Treinamento prático em PHP. Atualizado até a versão 5.05**. São Paulo: Digerati Books, 191p., 2005.

MORIMOTO, C. E. **Redes e servidores Linux: guia prático.** Porto Alegre: Sul Editores, 302p., 2005.

NEMETH, E.; SNYDER, G.; HEIN, T. R. **Manual Completo do Linux. Guia do Administrador.** São Paulo: Pearson Prentice Hall, 683p., 2007.

NIEDERAUER, J. **Guia de consulta rápida – PHP 5**. São Paulo: Novatec Editora, 143p., 2004.

PINTO, S. R. B. **Treinamento prático em Dreamweaver**. São Paulo: Digerati Books, 128 p., 2006.

PRITCHARD, Steven *et al.* **Certificação Linux LPI: rápido e prático.** Rio de Janeiro: Alta Books, 485 p., 2007.

RICCI, B.; MENDONÇA, N. **Squid: solução definitiva.** Rio de Janeiro: Ciência moderna, 151 p., 2006.

TRIGO, C. H. **OpenLDAP – Uma Abordagem Integrada**. Rio de Janeiro: Novatec, 240 p., 2007.

VASQUES, A. T.; SCHUBER, R. P. **Implementando VPN em Linux com IPSec.** Belém. Disponível em: <http://abusar.org/manuais/VPN-alan-rafael.pdf>. Acesso em: 03 nov. 2008

VEIGA, R. G. A. **Guia de consulta Rápida - Apache.** São Paulo: Novatec, 112 p., 2006.

WIEL, F. A. **Guia de Instalação do Debian GNU/Linux**. Curitiba. Disponível em: <http://www.debian.org/releases/stable/i386/ch01s01.html.pt_BR>. Acesso em: 28 mar. 2009.