UNIVERSIDADE DO SUL DE SANTA CATARINA

ANDRÉ SANTOS DE ALMEIDA

F-WEB MANAGER:
GESTÃO DE NAVEGAÇÃO BASEADO EM CATEGORIAS

Tubarão

2012

ANDRÉ SANTOS DE ALMEIDA

F-WEB MANAGER:
GESTÃO DE NAVEGAÇÃO BASEADO EM CATEGORIAS

Trabalho de conclusão de curso apresentado no curso de Tecnologia em Redes de Computadores, como requisito à obtenção do título de Tecnólogo em Redes de Computadores da Universidade do Sul de Santa Catarina.

Orientador: Prof. Carlos Alberto Luz

Tubarão

2012

ANDRÉ SANTOS DE ALMEIDA

F-WEB MANAGER:
GESTÃO DE NAVEGAÇÃO BASEADO EM CATEGORIAS

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do título de Tecnólogo e aprovado em sua forma final pelo Curso de Graduação em Tecnologia em Redes de Computadores da Universidade do Sul de Santa Catarina.

Tubarão, 07 de Dezembro de 2012.

Prof. e orientador Carlos Alberto Luz, Esp.
Universidade do Sul de Santa Catarina

Prof. Ana Lúcia Rodrigues Wiggers, Esp.
Universidade do Sul de Santa Catarina

Prof. Silvana Madeira Alves Dal-Bó, Msc.
Universidade do Sul de Santa Catarina

André Demetrio da Silva, Esp.
Universidade do Sul de Santa Catarina

Dedico este trabalho aos meus pais, Antônio Geraldo Pereira de Almeida e Maria Madalena Santos de Almeida, que sempre me motivaram e deram o apoio necessário para conclusão deste trabalho.

AGRADECIMENTOS

Agradeço a Deus pelo dom da vida a mim concebido e por ter iluminado o meu caminho durante todos esses anos.

À minha família, que sempre me apoiou: meus pais Geraldo e Madalena, meus irmãos Adriano, Renata e Raquel.

Aos poucos, mas sinceros amigos que me ajudaram a seguir adiante nos momentos de dificuldade.

Ao meu orientador Prof. Carlos Alberto Luz, pelos ensinamentos e palavras de incentivo.

Á todos que, de uma maneira ou de outra passaram pela minha vida, não só durante a execução deste trabalho, mas durante toda a minha jornada acadêmica.

“O sucesso é a soma de pequenos esforços - repetidos dia sim, e no outro dia também.” (Robert Collier)

RESUMO

Devido ao aumento da utilização das redes de computadores e de sistemas ligados a Internet, surgiu uma área muito importante chamada segurança da informação, cujo objetivo é garantir a disponibilidade, confidencialidade e integridade das informações. Os *firewalls* apresentam-se como um conjunto de tecnologias disponíveis para agregar segurança aos ambientes computacionais, principalmente para redes corporativas. No entanto, o modelo de funcionamento e operação deste ativo de segurança da informação geralmente é complexo, exigindo conhecimento técnico para gerenciamento dos recursos disponíveis na ferramenta. Com o intuito de facilitar a administração e agregar valor aos mecanismos de um *firewall* foi desenvolvido o *F-Web Manager*, ferramenta responsável pelo gerenciamento do *Squid*, um firewall de aplicação, gerando uma interface gráfica em PHP para as mais diversas operações. Além disso, o *F-Web Manager* é capaz de gerenciar e atualizar categorias de sites com um aplicativo servidor, além de permitir o gerenciamento de categorias locais, listas de acessos e regras de acesso. O *F-Web Manager* permite através da interface de gerenciamento abstrair a complexidade por trás de um *firewall* de aplicação para gerenciamento das regras de acesso, facilitando o gerenciamento das configurações e a implementação de novas políticas dentro da rede.

Palavras-chave: *Firewall*. *F-Web Manager*. *Squid*.

ABSTRACT

Due to the increase in use of computer networks and systems conected to the Internet, a very important area called Information Security emerged. Its goal is to guarantee information availability, confidentiality and integrity. The firewalls are a set of technologies available to aggregate security to computing environments, especially to corporate networks. However, the operation model and functioning of this information security asset is usually complex, requiring technical knowledge for the management of this tool's available resources. Aiming to facilitate the management and to add value to the firewall mechanisms, the F-Web Manager was developed. This tool is responsible for the management of the Squid, an application firewall that generates a graphical interface in PHP for several operations. Besides, the F-Web Manager is capable of managing and updating site categories with an application server, also allowing the management of local categories, access lists and access rules. The F-Web Manager allows through the management interface to abstract the complexity behind an application firewall for the management of the access rules, facilitating the setting management and the implementation of new policies inside the network.

Keywords: Firewall. F-Web Manager. Squid.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

ACL - Access Control List

ANSI - American National Standards Institute

ARP - Address Resolution Protocol

ARPA - Advanced Research Projects Agency

ARPANET - Advanced Research Projects Agency Network

CERN - European Organization for Nuclear Research

CPD - Centro de Processamento de Dados

CPU - Central Processing Unit

CSS - Cascading Style Sheets

DBMS - Data Base Management System

DML - Data Manipulation Language

DOD - Department of Defense

BSD - Berkeley Software Distribution

CERN - European Organization for Nuclear Research

CETIC.BR - Centro de Estudos sobre as Tecnologias da Informação e da Comunicação

DHCP - Dynamic Host Configuration Protocol

DNS - Domain Name System

DOD - U.S Departament Of Defense

FTP - File Transfer Protocol

GB - Gigabyte

HIDS - Host Intrusion Detection System

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

HTTPS - HyperText Transfer Protocol Secure

ICAP – Internet Content Adaptation Protocol

ICMP - Internet Control Message Protocol

IDS - Intrusion Detection System

IGMP - Internet Group Management Protocol

IP - Internet Protocol

IPv4 - Internet Protocol Version 4

IPv6 - Internet Protocol Version 6

ISO - International Organization for Standardization

ISP - Internet Service Provider

LAN - Local Area Network

LDD - Data Definition Language

NDP - Neighbor Discovery Protocol

NIC.BR - Núcleo de Informação e Coordenação do Ponto BR

NIDS - Network Intrusion Detection System

NLANR - National Laboratory for Applied Network Research

OOP - Object-Oriented Programming

OQL - Object Query Language

OS - Operating System

OSI - Open Systems Interconnection

PHP - Hypertext PreProcessor

RAM - Random Access Memory

RARP - Reverse Address Resolution Protocol

RFC - Request For Comments

SGBD - Sistema de Gerenciamento de Banco de Dados

SMTP - Simple Mail Transfer Protocol

SQL - Structured Query Language

SSL - Secure Sockets Layer

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

VM - Virtual Machine

WAN - Wide Area Network

WWW - World Wide Web

XHTML - eXtended HyperText Markup Language

SUMÁRIO

1 INTRODUÇÃO

Com o aumento cada vez maior do uso da *internet*, as empresas acabam se tornando cada vez mais dependentes de tecnologias. Por causa das facilidades que este meio propicia, as organizações passam a utilizá-la para as mais diversas funções, como por exemplo, os serviços de envio e recebimento de e-mails, busca de informações sobre produtos ou serviços, busca de informações de atividades de pesquisa, serviços bancários e financeiros entre outros.

O uso da *internet* hoje é indispensável para qualquer empresa, pois lhe aproxima dos seus consumidores, parceiros e fornecedores, além de outras facilidades. Além disso, essa ferramenta pode ser utilizada pelos colaboradores para a realização de serviços de pesquisa, vendas, compras entre outros, mas também pode ser utilizada de maneira indevida se não existir um controle efetivo desses acessos.

Por este motivo, faz-se necessário que os administradores realizem ações que delimitem o que pode ou não ser acessado por seus funcionários através da implementação de ações especificas como bloqueios de *sites*, *downloads* e outros acessos que não devem ser permitidos na empresa, visando sua maior segurança.

Essas ações visam monitorar e controlar o acesso de navegação/*Web* feito pelos gerenciados. Para este propósito surgiram ferramentas que realizam diversas funções como: filtro de pacotes (que trabalha na camada de rede) e os servidores *proxy* (que trabalham na camada de aplicação), onde é possível controlar os sites que podem ou não ser acessados, controlando o conteúdo disponibilizado e garantindo maior proteção à organização.

Ante ao exposto, a proposta deste trabalho é desenvolver uma *ferramenta* que possibilite o gerenciamento da navegação/*Web* baseado em categorias pré-definidas, permitindo ao administrador definir a política de acesso à *internet*, definindo quais categorias os gerenciados poderão utilizar e quais deverão ser bloqueados.

A ferramenta permitirá que qualquer administrador sem grande conhecimento técnico possa gerenciar o serviço de navegação/*Web*, pois utilizará uma interface *web* amigável ao invés do tradicional console. Além disso, será desenvolvido uma ferramenta cliente e uma ferramenta servidor, permitindo desta forma que o conteúdo destas categorias pré-definidas no cliente sejam atualizadas com este servidor.

1.1 JUSTIFICATIVA

Devido ao crescimento da *internet*, as empresas acabaram gerando uma dependência muito grande de tecnologias que utilizam este meio e, estão investindo e se preocupando cada vez mais com essas tecnologias.

Para garantir a segurança, integridade e disponibilidade das informações, torna-se cada vez mais necessário o investimento das empresas em sistemas como *Firewall* (filtro de pacotes), *firewall* de aplicação como *proxy web*, entre outros.

A grande maioria dos sistemas de gestão de navegação/*Web* não são interfaceadas e para realizar o gerenciamento o administrador precisa ter um conhecimento apurado para conseguir utilizar todos os recursos disponíveis de uma ferramenta de *proxy web*.

Hoje já existem interfaces para gerenciamento de navegação/*Web*, porém estes sistemas não são amigáveis e nem de fácil entendimento e, desta forma o administrador que não tem grande conhecimento técnico acabam não conseguindo utilizar todos os recursos disponíveis pelo sistema de *proxy web*.

Tendo em vista esta dificuldade de gerenciamento de navegação/*Web*, o *F-Web Manager* permite ao administrador gerenciar a navegação/Web baseado em categorias que são automaticamente atualizadas com um servidor. A ferramenta permite ao administrador definir a política de navegação e quais categorias poderão ser acessadas e quais deverão ser bloqueadas de forma fácil, simples e eficaz.

1.2 OBJETIVOS

Exposto o tema e a proposta, apresenta-se a seguir os objetivos deste trabalho.

1.2.1 Objetivo geral

Desenvolver uma ferramenta que permita ao administrador gerenciar a navegação utilizando categorias pré-definidas através de uma interface *Web* simples e de fácil acesso.

1.2.2 Objetivos específicos

a) Fundamentar, através de referencial teórico acerca de informações consideradas relevantes para o desenvolvimento do tema proposto;
b) Aperfeiçoar os conhecimentos em:
   - Programação;
   - Banco de dados; e
   - Redes de computadores.
c) Desenvolver uma ferramenta que permita:
   - A criação de listas e controle de acesso;
   - Atualização automática do conteúdo das categorias;
   - Controle de acesso baseado nas categorias atualizadas; e
   - A definição de uma política padrão.
d) Permitir que o administrador defina as permissões de cada usuário;
e) Disponibilizar o relatório das alterações na ferramenta, além de permitir a visualização dos *sites* acessados pelos ativos controlados.

1.3 ABRANGÊNCIA

Para garantir a sequência correta da execução do trabalho precisou-se criar delimitações de forma que o mesmo possa ser desenvolvido em tempo hábil e dentro dos objetivos.

A ferramenta desenvolvida rodará no sistema operacional *FreeBSD*;

A ferramenta poderá ser implementada em qualquer ambiente que possua um servidor de *Web Proxy*, desde que a ferramenta utilizada seja o *Squid*;

O computador que terá o *F-Web Manager* instalado deverá conter pelo menos uma interface de rede física, um disco rígido com pelo menos 2 GB disponíveis e no mínimo 512 MB de memória RAM;

O *F-Web Manager* será responsável pela filtragem de conteúdo *Web* apenas para conexões que utilizem o protocolo HTTP e que cheguem até o servidor onde estará instalado a ferramenta;

Todo o *hardware* utilizado no computador no qual o *F-Web Manager* estiver instalado deverá ser compatível com o sistema operacional *FreeBSD* a partir da versão 9.0;

Não serão implementados suporte à autenticação, balanceamento de links, controle de banda e nem suporte ao protocolo.

1.4 METODOLOGIA

Este trabalho terá como base a pesquisa bibliográfica, pois visará, através deste meio, buscar os conhecimentos acerca dos conceitos considerados imprescindíveis para a realização deste mesmo.

Na seqüência haverá o período de maturação do trabalho. Nesse estágio, em conjunto com o orientador, será realizada uma análise de requisitos para a elaboração do trabalho, serão discutidos os detalhes de funcionamento da ferramenta a ser desenvolvida e o cronograma, para que tudo possa ser realizado em tempo hábil.

Depois de definido os objetivos e alinhados os passos e o cronograma, é o momento do desenvolvimento. O desenvolvimento da ferramenta, chamada de *F-Web Manager*, será dividida em dois estágios: *back-end* e *front-end*.

O *Back-end* será toda a estrutura da ferramenta que não é visualizada pelo operador. Esta estrutura interage diretamente com o sistema operacional e sua função é processar as informações coletadas pelo *front-end* e armazenar ou buscar as informações no SGBD, além de executar algumas funções que devem ser feitas automaticamente. O SGBD utilizado será o *SQLite*. O *backend* será desenvolvido com a utilização de *Shell script* e também da linguagem de programação *Python*.

Em contrapartida o *front-end* é a parte visível da ferramenta, onde existe uma interação direta com o operador. Ele será um ambiente gráfico onde o administrador poderá gerenciar toda a ferramenta, visualizando e alterando os dados para o seu funcionamento. O *front-end* será desenvolvido com a utilização de linguagens de programação para web, como HTML, CSS e PHP, onde a última fará a integração com o *back-end*.

Após o desenvolvimento, a ferramenta passará por uma fase de homologação. É neste período que são feitos os testes com o objetivo de encontrar possíveis *bugs*.

É importante informar que os testes e a apresentação do trabalho serão realizados em um ambiente virtualizado, ou seja, com a utilização de máquinas virtuais devido a

dificuldade e custos elevados de implementar um ambiente físico real. A utilização de um ambiente virtualizado nada interfere na utilização da ferramenta e nos resultados alcançados se comparados com um ambiente físico real devido à alta precisão de um ambiente virtual simular um ambiente real.

No final de todas estas etapas serão documentadas as conclusões constatadas com a implantação e utilização do *F-Web Manager*.

2 REDES DE COMPUTADORES

Redes de computadores são dois ou mais computadores que são interconectados por uma única tecnologia e que por estarem interligados permitem a troca de informações entre si. (Tanenbaum, 2003, p. 2).

Para Torres (2009, p. 4), "As redes de computadores surgiram da necessidade da troca de informações, já que é possível ter acesso a um dado que está fisicamente localizado distante de você [...]."

O principal objetivo das redes de computadores é permitir a comunicação de dois ou mais computadores, porém esta interligação provê outros benefícios como o compartilhamento de dados, compartilhamento de recursos e administração centralizada.

2.1 PROTOCOLOS DE COMUNICAÇÃO

Protocolo é um conjunto de regras que especifica os formatos de mensagens, além de descrever o que um computador deve fazer ao receber uma mensagem e como os possíveis erros devem ser tratados. O objetivo do protocolo de comunicação é permitir que todos os computadores que tenham conhecimento desse protocolo possam se comunicar, independente do *hardware* de rede. (Comer, 1998, p. 4; Tanenbaum, 2003, p. 39).

Podemos concluir então que o protocolo é o nome dado a um conjunto de regras que os computadores devem seguir para que a comunicação entre eles sejam estabelecidas.

A seguir falaremos sobre o modelo de referência TCP/IP que foi criada com o objetivo de resolver os problemas de comunicações entre as mais diversas tecnologias.

2.2 MODELO DE REFERÊNCIA TCP/IP

O padrão TCP/IP foi criado pelo Departamento de Defesa dos Estados Unidos para garantir que os dados fossem preservados além de manter a comunicação de dados caso houvesse uma guerra. (Palma e Prates, 2000, p. 5; Tanenbaum, 2003, p. 45).

> Diante da preocupação do Departamento de Defesa dos EUA de que seus preciosos hosts, roteadores e gateways de interconexão de redes fossem destruídos de uma hora para outra, definiu-se também que a rede deveria ser capaz de sobreviver à perda do hardware de sub-redes, com as conversações existentes sendo mantidas em atividade. Em outras palavras, o Departamento de Defesa dos EUA queria que as conexões permanecessem intactas enquanto as máquinas de origem e de destino estivessem funcionando, mesmo que algumas máquinas ou linhas de transmissão intermediárias deixassem de operar repentinamente. (Tanenbaum, 2003, p. 45)

Aos poucos várias universidades e repartições públicas foram conectadas usando linhas telefônicas dedicadas, porém, quando foram criadas as redes de rádio e satélite surgiram vários problemas com os protocolos existentes, forçando a criação de uma nova arquitetura de referência onde seria possível interconectar várias redes, independente da tecnologia utilizada. (Tanenbaum, 2003, p.44; Torres, 2009, p.45).

Essa arquitetura que foi responsável por permitir a comunicação entre as redes ficou conhecida como Modelo de Referência TCP/IP, graças a seus dois principais protocolos que compõem o modelo: os protocolos TCP e o IP. (Comer, 1998, p. 185-186, Tanenbaum, 2003, p. 44).

A pilha TCP/IP possui uma estrutura multi-camadas. A fim de organizar a tarefa de comunicação, foram definidas quatro camadas que estão ilustradas pela Figura 1.

Figura 1 - Camadas do modelo TCP/IP

Fonte: Palma e Prates, 2000, p. 9.

2.2.1 Camada de interface de rede ou acesso a rede

A camada de interface de rede é o nível mais baixo do *software* TCP/IP e é onde

estão implementados os protocolos como ARP e RARP. Esta camada é responsável pela aceitação de datagramas IP e por sua transmissão através de uma rede específica. Esta camada inclui o *driver* de dispositivo do sistema operacional e o respectivo *hardware* de rede, que juntos lidam com todos os detalhes físicos e interação com o cabo, recebendo e transmitindo dados. (Comer, 1998, p. 186; Torres, 2009, p. 51-52).

Alguns dos protocolos que fazem parte da camada de acesso à rede são: as variantes de protocolo Ethernet, Token Ring, FDDI e outros padrões LAN e WAN populares, como o protocolo Ponto-a-Ponto (PPP), X.25 e Frame Relay.

2.2.2 Camada de rede ou internet

Segundo Torres (2009, p. 224), esta camada é responsável por receber os pacotes de dados vindo da camada de transporte e dividi-los em datagramas, adicionando a informação do endereço lógico de origem e destino, e posteriormente é enviado para a camada de interface de rede.

Os protocolos que atuam na camada de rede são o IP, ICMP, IGMP, ARP, RARP e NDP. Conforme Tanenbaum (2003, p. 45), "A camada inter-redes define um formato de pacote oficial e um protocolo chamado IP. A tarefa da camada inter-redes é entregar pacotes IP onde eles são necessários."

2.2.3 Camada de transporte

A primeira função da camada de transporte é prover a comunicação de um programa aplicativo para outro. Esta camada pode regular o fluxo de informações e fornecer um transporte confiável, assegurando que os dados cheguem sem erros e em seqüencia, pois o protocolo faz com que o lado receptor envie confirmações e que o lado transmissor retransmita pacotes perdidos. (Comer, 1998, p.185-186).

Para Torres (2009, p. 202), "Esta camada é responsável por pegar os dados enviados pela camada de aplicação e transformá-los em pacotes, a serem repassados para a camada de rede."

Na camada de transporte são fornecidos dois tipos de serviços, a entrega garantida, pelo protocolo TCP e a entrega do melhor esforço, pelo protocolo UDP.

Tanenbaum (2003, p. 46) afirma que "[...] o TCP [...], é um protocolo orientado a conexões confiável que permite a entrega sem erros [...]" e também afirma que "[...] o UDP [...] é um protocolo sem conexão e não-confiável destinado a aplicações que não querem controle de fluxo nem manutenção da seqüência das mensagens enviadas [...]"

O TCP é um protocolo orientado a conexões e confiável que permite a entrega sem erros de um fluxo de bytes originário de qualquer computador da inter-rede. Já o UDP é utilizado quando o problema de troca de dados confiáveis não existe ou é solucionado pelas ferramentas de uma camada superior ou por aplicações do usuário.

2.2.4 Camada de aplicação

A camada de aplicação contém todos os protocolos de nível mais alto, onde os usuários executam programas aplicativos que acessam serviços disponíveis através de uma interligação em redes TCP/IP. (Comer, 1998, p. 185).

Para Torres (2009, p. 181), a camada de aplicação comunica-se com a camada de transporte utilizando uma porta, para saber qual protocolo será utilizado para transferir os dados, e também saber qual protocolo de aplicação na máquina de destino os dados devem ser entregues.

Dentre os protocolos que atuam na camada de aplicação estão o TELNET, FTP, SMTP, DNS e o HTTP. Cada protocolo da camada de aplicação possui uma porta padrão que são numeradas de zero à 65.535. Por exemplo, em servidores o protocolo SMTP utiliza a porta 25, o protocolo HTTP utiliza a porta 80 e o FTP utiliza a porta 20 (para a transmissão de dados) e 21 (para transmissão de informações de controle). (Torres, 2009, p. 181).

2.3 SERVIÇOS DE REDE

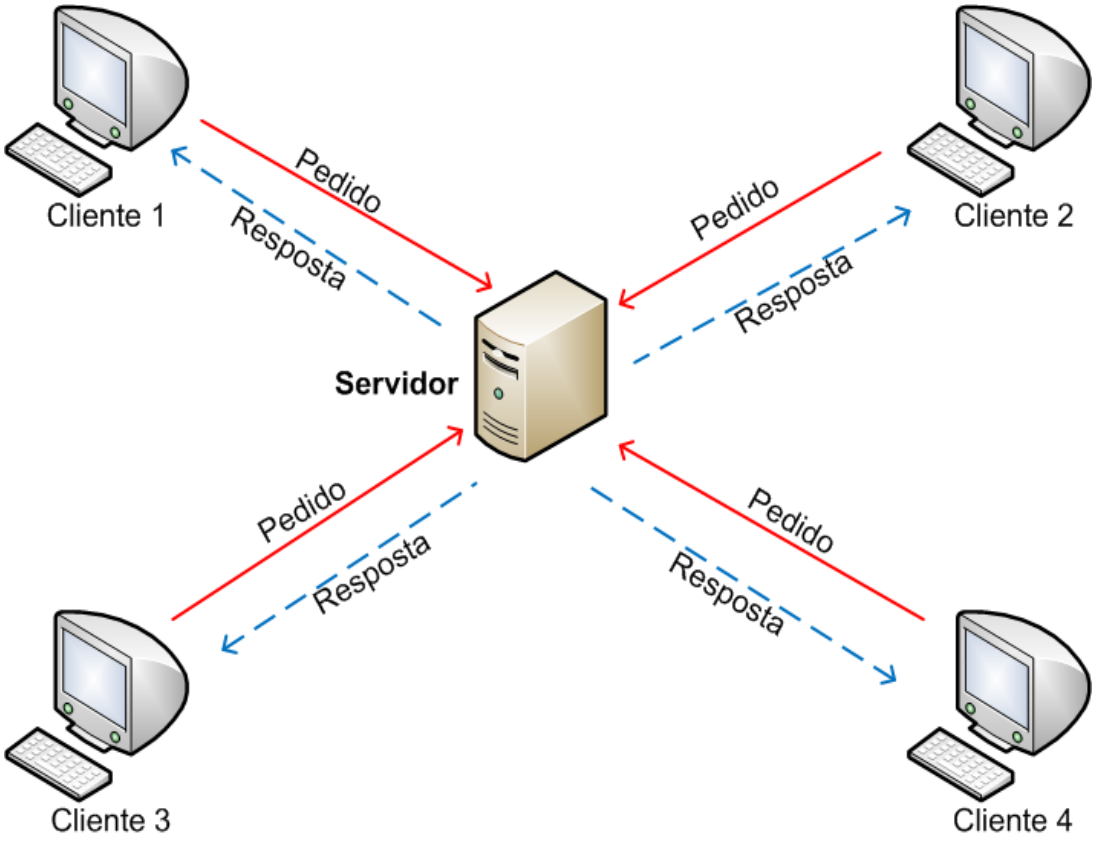
São os serviços que ficam localizados na camada de aplicação da arquitetura Internet, que são visualizados pelo usuário e são eles os responsáveis pela troca de

informações e compartilhamento de recursos. (Costa, 2007, p. 5).

Um serviço de rede pode ser considerado como uma aplicação distribuída, que executa em dois ou mais computadores conectados por uma rede. Geralmente é utilizado um servidor que realiza as funções principais do serviço e o cliente que solicita o serviço ao servidor.

No exemplo ilustrado pela Figura 2, é utilizado um servidor de rede que disponibiliza serviços e envia aos clientes as informações solicitadas.

Figura 2 - Exemplo de utilização de serviços de rede



Fonte: Elaborado pelo Autor, 2012.

2.3.1 Principais serviços

Cada serviço de rede que atua na camada de aplicação utiliza um padrão de porta e protocolo. No Quadro 1 pode ser visto a lista dos principais serviços, juntamente com os protocolos e a porta utilizada por eles.

Quadro 1 - Lista dos principais serviços de rede

Protocolo	Porta	Serviço
FTP	TCP 20/21	Transferência de arquivos.
SSH	TCP 22	Terminal remoto seguro.
TELNET	TCP 23	Terminal remoto não seguro.
SMTP	TCP 25	Transferência de e-mail.
DNS	TCP/UDP 53	Tradução direta e reversa de nomes de domínio em endereços de rede.
DHCP	UDP 68	Configuração automática de interfaces de rede.
TFTP	UDP 69	Transferência rápida de arquivos, sem garantias.
HTTP	TCP 80	Transferência de arquivos hipermídia.
KERBEROS	UDP/TCP 88/464	Gerência de tokens seguros de autenticação.
POP3	TCP 110	Leitura de e-mail.
NTP	UDP 123	Sincronização e ajuste de relógio.
NETBIOS	TCP/UDP 137/138/139	Protocolos de rede proprietários da Microsoft.
IMAP	TCP 143	Leitura de e-mails.
SNMP	TCP 161	Gerência de equipamentos e hosts da rede.
BGP	TCP 179	Controle de rotas entre grandes redes.
LDAP	TCP/UDP 389	Serviços de diretório remoto.
RTSP	TCP/UDP 554	Controle de fluxos multimídia em tempo real.
LDAPS	TCP 636	Serviço de diretório remoto, com acesso criptografado.
HTTPS	TCP 443	Transferência segura de conexão hipermídia.

Fonte: Costa, 2007, p. 6.

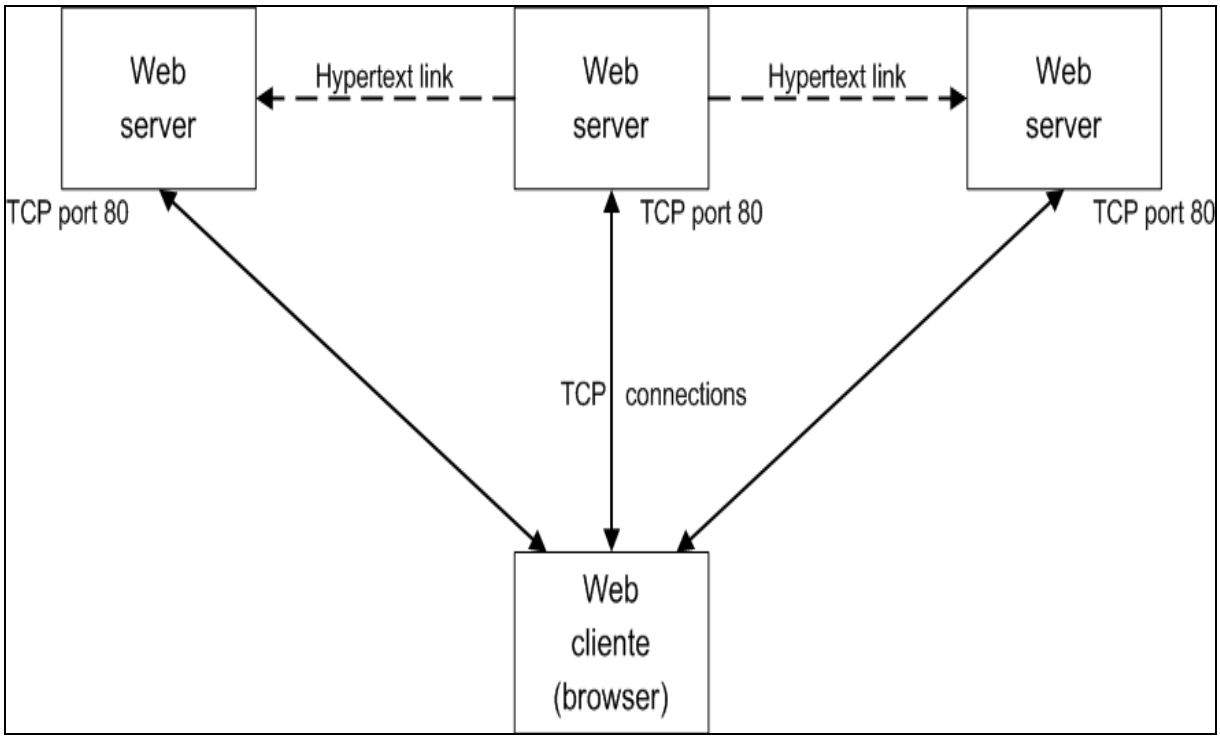
2.3.2 HTTP

De acordo com Tanenbaum (2009, p. 651), a *Web* teve início em 1989 no CERN. O CERN possui vários aceleradores de partículas onde vários cientistas de países diferentes participam e a *Web* nasceu da necessidade de fazer com que esses grupos de cientistas de diferentes nacionalidades pudessem colaborar uns com os outros através da troca de informações.

O HTTP é a base para o WWW que é mais conhecida como simplesmente *Web*. O funcionamento deste protocolo é simples. O cliente estabelece uma conexão TCP com o servidor, emite um pedido e aguarda a resposta. O servidor indica o fim da sua resposta quando fecha a conexão e então, o cliente (*browser*) apresenta para o usuário as informações obtidas do servidor. (Stevens, 2002, p. 161-162).

Na Figura 3 elenca-se o diagrama do funcionamento do cliente e servidor *Web*.

Figura 3 - Diagrama do funcionamento de um cliente e servidor Web



Fonte: Stevens, 2002, p. 162.

Atualmente as páginas são escritas em uma linguagem chamada HTML. Esta linguagem permite que os usuários criem páginas *Web* que incluem texto, gráficos e ponteiros para outras páginas *Web*.

O HTML é uma linguagem de marcação que descreve como os documentos devem ser formatados. As linguagens de marcação contêm comandos explícitos de formatação. Por exemplo, o comando significa o início do modo negrito, e o comando o final. (Tanenbaum, 2009, p. 670).

A primeira versão do HTML é a 1.0, mas com o tempo surgiram novas versões de acordo com as necessidades. As diferenças entre as versões podem ser vista no Quadro 2.

Quadro 2 - Diferenças entre as versões do HTML

Item	**HTML 1.0**	**HTML 2.0**	**HTML 3.0**	**HTML 4.0**
Hiperlinks	X	X	X	X
Imagens	X	X	X	X
Listas	X	X	X	X
Imagens e mapas ativos		X	X	X
Formulários		X	X	X
Equações			X	X
Barras de ferramentas			X	X
Tabelas			X	X
Recursos de acessibilidade				X
Incorporação de objetos				X
Criação de scripts				X

Fonte: Tanenbaum, 2009, p. 676.

De acordo com Tanenbaum (2009, p. 684-685), existe uma versão mais recente do HTML que é denominada XHTML. Existem seis diferenças principais entre o HTML 4 e o XHTML, são elas:

a) As páginas *Web* e os navegadores devem obedecer estritamente ao padrão.
b) Todas as *tags* e atributos devem estar em letras minúsculas.
c) As *tags* de fechamento são obrigatórias.
d) Os atributos devem estar contidos em aspas.
e) As *tags* devem ficar aninhadas de maneira apropriada.
f) Todo documento deve especificar seu tipo de documento.

3 SEGURANÇA DA INFORMAÇÃO

Segurança da informação pode ser definida como uma área de conhecimento que é dedicada à proteção de ativos da informação contra acessos não autorizados, alterações indevidas ou sua indisponibilidade. (Sêmola, 2003, p. 43).

Qualquer informação do meio computacional está sujeita a violação e outras ações criminosas, e o objetivo da segurança da informação é justamente evitar que estas violações ocorram, evitando assim o comprometimento das informações.

A segurança da informação deve proteger tanto o ambiente físico, como também o ambiente lógico e sua política de segurança deve garantir a confidencialidade, integridade e disponibilidade das informações.

3.1 SEGURANÇA FÍSICA

Segundo Furtado (2002, p. 38-39) e Caruso e Steffen (2006, p. 33), a segurança física se refere aos danos que podem ser causados por negligência ou propositalmente, assim como fatores naturais, acidentais ou criminais. O gestor da informática deve proteger os equipamentos de acordo com o nível de importância que o prejuízo pode causar a empresa.

Podem ser colocados equipamentos considerados essenciais em uma sala-cofre para proteger este de incêndios. Outro recurso interessante são os sensores de incêndios, mas este recurso deve ser instalado com cuidado, pois o mesmo pode gerar um grande prejuízo se colocado em uma sala onde tem documentos impressos.

As principais ameaças físicas são incêndios, desabamentos, alagamento, acesso indevido de pessoas e manuseio inadequado do material.

3.2 SEGURANÇA LÓGICA

Conforme Furtado (2002, p. 39-40), a segurança é lógica quando é feita por *software*. Esta segurança é feita em dois níveis: controle e níveis de acesso à informação.

O controle de acesso das informações pode ser elaborado por meio de senhas específicas para cada usuário, as quais devem ser alteradas com certa regularidade. Sua principal função é permitir ou não o acesso a um sistema.

Os níveis de acesso às informações requerem organização de restrições e responsabilidades pelo acesso, que determinam onde e como cada usuário pode acessar informações específicas.

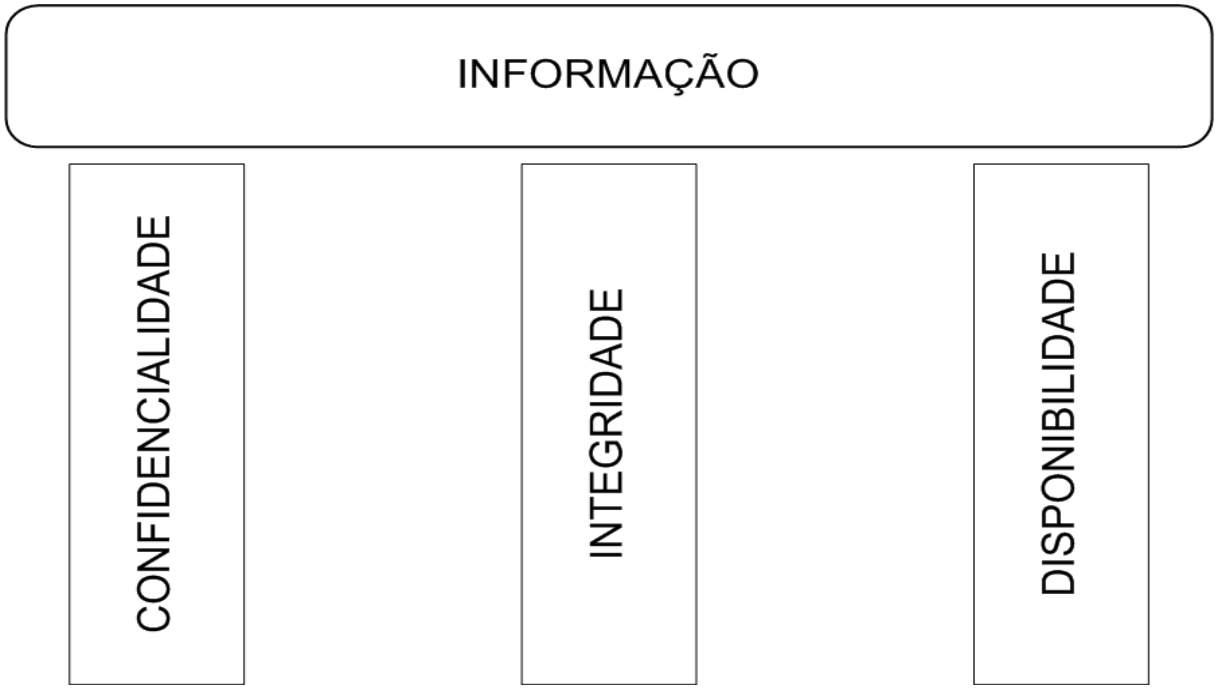
As principais ameaças lógicas são vírus, acessos remotos indevidos, *backup* desatualizados, violação de senhas, violação de ferramentas desatualizadas dentre outros.

3.3 PILARES

Conforme Campos (2006, p5), "Um sistema de segurança da informação baseia-se em três princípios básicos: 1) confidencialidade, 2) integridade e 3) disponibilidade."

Para garantir que a informação não seja acessada ou alterada e esteja disponível sempre que solicitada, precisamos trabalhar nesses pilares para garantir que não haja falhas que podem ser exploradas por possíveis ameaças.

Figura 4 - Pilares da segurança da informação



Fonte: Campos (2006, p.5).

3.3.1 Confidencialidade

Para Sêmola (2003, p. 45), toda informação deve ser protegida com o objetivo de garantir que somente pessoas autorizadas tenham acesso ao conteúdo.

O princípio de confidencialidade deve prevenir o vazamento de informações para indivíduos ou sistemas não autorizados. A partir do momento que pessoas não autorizadas tenham acesso ao conteúdo, configura um incidente de segurança da informação por quebra de confidencialidade.

Podemos citar como exemplo uma transação online utilizando cartão de crédito, onde é necessária que o número do cartão seja transferido entre o computador e o servidor onde está sendo feito a transação. O sistema do site de comércio eletrônico deve garantir a confidencialidade dos dados do cartão, de modo que outras pessoas não tenham acesso a esta informação.

3.3.2 Integridade

De acordo com Campos (2006, p. 6), a informação precisa estar completa, sem alterações e, portanto, confiável.

O princípio da integridade deve evitar que a informação original seja modificada sem a devida autorização. Qualquer alteração da informação original de maneira indevida configura um incidente de segurança da informação por quebra de integridade.

Podemos citar como exemplo um vírus que modifica arquivos de um computador, ou *crackers* que modificam o conteúdo de um site na internet.

3.3.3 Disponibilidade

Conforme Sêmola (2003, p. 45), toda informação gerada ou adquirida por um indivíduo ou instituição deve estar disponível no momento em que os mesmos precisarem para qualquer finalidade.

O princípio da disponibilidade deve garantir que toda informação requisitada por

indivíduos ou equipamentos autorizados deve estar disponível quando for solicitada. Se em algum momento o indivíduo ou equipamento solicitar a informação e a mesma estiver indisponível, configura uma quebra de disponibilidade.

Podemos utilizar como exemplo o site de um banco ou comércio eletrônico que está indisponível no momento em que um usuário precisa acessar para fazer transações.

3.4 RISCOS

De acordo com Lins (2009, p. 145), risco é um possível evento que pode causar perda ou prejuízo ao andamento dos negócios. Este evento é uma combinação de fatores, sendo em parte a identificação de uma ameaça, em outra parte a identificação de uma vulnerabilidade, ou seja, existe o risco da ameaça explorar a vulnerabilidade existente.

Podemos concluir então que quando existe uma vulnerabilidade também existe o risco desta ser explorada por uma ameaça.

3.5 VULNERABILIDADES

Para Campos (2006, p.11), vulnerabilidades são as fraquezas presentes nos ativos de informação, que podem quebrar de um dos três princípios da segurança da informação através das ameaças.

3.5.1 Tipos de vulnerabilidades

Conforme Sêmola (2003, p. 48), as vulnerabilidades por si só não geram incidentes de segurança por serem elementos passivos, necessitando de um agente causador ou condição favorável que são as ameaças.

Alguns tipos de vulnerabilidades podem ser vistos abaixo:

a) Físicas: instalações prediais fora do padrão; salas de CPD mal planejadas; falta

de extintores; detectores de fumaça e de outros recursos para combate a incêndio em sala com armários e fichários estratégicos; risco de explosões, vazamentos ou incêndio.

b) Naturais: computadores são suscetíveis a desastres naturais, como incêndios, enchentes, terremotos, tempestades, e outros, como falta de energia, acúmulo de poeira, aumento de umidade e de temperatura.
c) Hardware: falha nos recursos tecnológicos ou erros durante a instalação.
d) Software: erros na instalação ou na configuração podem acarretar acessos indevidos, vazamento de informações, perda de dados ou indisponibilidade do recurso quando necessário.
e) Mídias: discos, fitas, relatórios e impressos podem ser perdidos ou danificados. A radiação eletromagnética pode afetar diversos tipos de mídias magnéticas.
f) Comunicação: Acessos não autorizados ou perda de comunicação.
g) Humanas: falta de treinamento, compartilhamento de informações confidenciais, não execução de rotinas de segurança, erros ou omissões; ameaça de bomba, sabotagens, distúrbios civis, greves, vandalismo, roubo, destruição da propriedade ou dados, invasões ou guerras.

3.6 AMEAÇAS

De acordo com Campos (2006, p. 13), a ameaça é um agente externo ao ativo da informação que pode quebrar um dos três principais da segurança da informação aproveitando-se das vulnerabilidades desse ativo.

3.6.1 Tipos de ameaças

Para Sêmola (2003, p. 47-48), as ameaças podem ser divididas nos seguintes grupos:

a) Naturais: ameaças decorrentes de fenômenos da natureza, como incêndios naturais, enchentes, terremotos, tempestades eletromagnéticas, maremotos,

aquecimento, poluição, etc.

b) Involuntárias: ameaças inconscientes, quase sempre causadas pelo desconhecimento. Podem ser causados por acidentes, erros, falta de energia, etc.
c) Voluntárias: ameaças propositais causadas por agentes humanos como *hackers*, invasores, espiões, ladrões, criadores e disseminadores de vírus de computador, incendiários.

3.7 MEDIDAS DE SEGURANÇA

Conforme Sêmola (2003, p. 49), "São as práticas, os procedimentos e os mecanismos usados para a proteção da informação e seus ativos, que podem impedir que ameaças explorem vulnerabilidades, [...]".

Sêmola ainda completa que as medidas de segurança podem ter as seguintes características: preventivas, detectáveis e corretivas.

3.7.1 Preventivas

São medidas de segurança utilizadas com o objetivo de evitar que incidentes ocorram. (Campos, 2006, p.91-92; Sêmola, 2003, p. 49). Algumas das ações preventivas podem ser vistas abaixo:

a) Identificar as potenciais não conformidades e suas causas;
b) Determinar e implementar as ações preventivas necessárias;
c) Registrar os resultados das ações tomadas;
d) Revisar as ações tomadas;
e) Identificar mudanças nos riscos de segurança da informação e garantir que o foco das ações esteja sobre as não conformidades que oferecem os maiores riscos.

3.7.2 Detectáveis

De acordo com Sêmola (2003, p. 49), são medidas de segurança que visam identificar causadores de ameaças, evitando assim que estas ameaças explorem vulnerabilidades.

Algumas destas medidas são análises de riscos, sistemas de detecção de intrusão, alertas de segurança, câmeras de vigilância, alarmes, etc.

3.7.3 Corretivas

São medidas utilizadas para correção de estrutura tecnológica e humana para que estas se adaptem as condições de segurança estabelecida pela instituição. Geralmente é utilizada em caso de desastres onde é acionada uma equipe de emergência que entra com um plano de recuperação de desastres. (Campos, 2006, p.90; Sêmola, 2003, p. 49-50).

As ações corretivas não podem ser apenas um acordo informal, mas deve ter um procedimento bem definido e documentado.

De acordo com Campos (2006, p. 90), o procedimento das ações corretivas precisa exigir minimamente o seguinte:

a) A identificação das não conformidades de implementação do sistema;
b) A identificação das causas dessas não conformidades;
c) A avaliação de necessidades de ações para garantir que essas não conformidades não voltem a ocorrer;
d) Estabelecer e implementar as ações corretivas identificadas;
e) Registrar os resultados das ações tomadas;
f) Revisar as ações corretivas tomadas.

3.8 ATIVOS DA INFORMAÇÃO

Para Sêmola (2003, p. 45), um ativo é, “Todo elemento que compõe os processos

que manipulam e processam a informação, a contar a própria informação, o meio em que ela é armazenada, os equipamentos em que ela é manuseada, transportada e descartada."

Segundo Campos (2006, p. 9), "[...] as organizações denominam seus bens patrimoniais como ativos, da mesma forma a informação e os mecanismos de comunicação podem ser chamados de ativos de informação, [...]".

Existem várias formas de se proteger os ativos da informação e os mais comuns são a utilização de *Firewall*, *Firewall* de aplicação e IDS.

3.8.1 Firewall

Conforme (Northcutt, 2002, p. 5), *Firewall* é um dispositivo que determina o tráfego que será permitido ou negado baseado em um conjunto de regras.

O *Firewall* trabalha na camada de rede e de transporte da pilha TCP/IP e, realiza as filtragens com base nas informações do cabeçalho dos pacotes, tais como endereço de origem, endereço de destino, porta de origem, porta de destino, protocolo e a direção das conexões. (Nakamura e Geus, 2007, p. 228).

O *Firewall* que está sendo discutido neste capítulo é um filtro de pacotes, porém existem outros mecanismos para filtragem de pacotes no nível de aplicação. Este tipo de *Firewall* é mais conhecido como *Firewall* de Aplicação ou *Firewall Proxy* e será discutido no próximo item. Existem dois tipos de filtros de pacotes, *firewall* sem estado e *firewall* com estado que são discutidos a seguir.

a) Firewall sem estado: Este tipo de filtro de pacotes não é considerado seguro porque não consegue distinguir entre pacotes verdadeiros e falsificados. Um indivíduo mal intencionado poderia criar um pacote falso para burlar facilmente este tipo de *Firewall*, além disso, é necessário manter um intervalo de portas altas (de 1024 a 65535) abertas para que as conexões sejam estabelecidas. (Nakamura e Geus, 2007, p. 230).
b) Firewall com estado: Este tipo de *Firewall* toma as decisões de filtragem tendo como referência dois elementos: As informações dos cabeçalhos dos pacotes de dados e em uma tabela de estados, que guarda os estados de todas as conexões. A diferença para o *Firewall* sem estado, é que o estado das conexões é monitorado a todo instante, desta forma a ação do *Firewall* pode ser definida

de acordo com o estado das conexões anteriores. (Nakamura e Geus, 2007, p. 232-233).

3.8.2 Firewall de aplicação

São sistemas que atuam como *gateway* entre duas redes, permitindo as requisições dos usuários internos e as respostas dessas requisições, de acordo com a política de segurança definida. O *Firewall* de aplicação é capaz de fazer filtragens de conteúdo, pois trabalha direto na camada de aplicação. (Cheswick, Bellovin e Rubin, 2003, p. 214; Nakamura e Geus, 2007, p. 224).

Podemos citar com exemplo de *Firewall* de aplicação o *Squid* que é um *proxy* capaz de fazer controle de navegação dos equipamentos gerenciados por este. O controle é feito através de um arquivo de configuração que é definido por um administrador.

3.8.3 IDS

O IDS é um sistema utilizado para detectar e gerar alertas em casos de eventos maliciosos. O sistema pode contar com vários sensores IDS que podem ser instalados em lugares estratégicos. (Cheswick, Bellovin e Rubin, 2003, p. 279; Nakamura e Geus, 2007, p. 264; Northcutt, 2002, p.5)

O IDS geralmente trabalha com assinaturas que são informações de ataques conhecidos, isto é bem semelhante ao funcionamento de um antivírus que contém assinatura ou código de vírus. Sendo assim, se o IDS encontrar determinada estrutura de dados no fluxo de rede, será considerado um ataque e poderá ser gerado um alerta, se definido pelo Administrador da rede.

Existem dois tipos de IDS e estes são descritos a seguir.

a) HIDS: O sistema de detecção de intrusão baseado em *host* monitora o sistema, com base das informações de arquivos de *logs* ou de agentes de auditoria. O HIDS é capaz de monitorar acessos e alterações em arquivos do sistema, modificações nos privilégios dos usuários, processos do sistema, programas

que estão sendo executado, uso da CPU, verificação da integridade dos arquivos do sistema, entre outros. (Nakamura e Geus, 2007, p. 270).

b) NIDS: O sistema de detecção de intrusão baseado em rede monitora o tráfego do segmento de rede. A detecção é realizada por meio da captura e análise dos cabeçalhos e conteúdos dos pacotes, que são comparados com padrões ou assinaturas conhecidas. O NIDS é muito eficiente contra ataques como *port scanning*, IP *spoofing*, *SYN flooding*, além de ser capaz de detectar ataques à serviços baseado em assinaturas. (Nakamura e Geus, 2007, p. 272).

4 GERÊNCIA DE REDES DE COMPUTADORES

Com a expansão da utilização da *internet*, as empresas necessitam cada vez mais se proteger, pois se torna demasiado dependente das tecnologias que utilizam este meio. De acordo com Cheswick, Bellovin e Rubin (2003, p. 3) a *Internet* é uma grande cidade e qualquer um pode utilizá-la, e usá-la quase que anonimamente e, por este motivo o autor define a *Internet* como um bairro ruim.

Por causa das inúmeras facilidades, a maioria das empresas hoje tem acesso a este meio. Segundo NIC.BR (2010), do total de empresas pesquisadas de setembro a dezembro de 2010, 95% utilizavam a internet para suas transações, apenas 3% não aproveitavam essa tecnologia e os 2% restantes correspondem a margem de erro utilizada pelo pesquisador. Este recurso é, muitas vezes, empregado tendo como finalidade o trabalho, mas também pode ser utilizado para outros fins não relacionados.

De acordo com Cheswick, Bellovin e Rubin (2003, p. 73) cada vez mais os administradores precisam monitorar e controlar o acesso de navegação/*Web* feito pelos colaboradores, bloqueando os acessos que não tenham relação com a empresa, visando sua proteção.

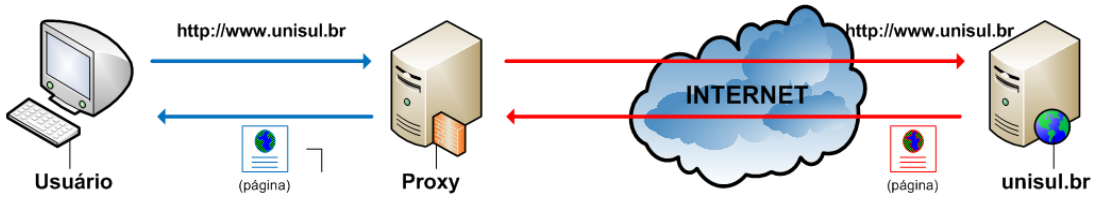
Para este propósito surgiram ferramentas que implementam diversas funções como filtro de pacotes, que trabalha na camada de rede, e os servidores *proxy*, que trabalham na camada de aplicação. Estas camadas baseiam-se no modelo de referência TCP/IP e encontram-se descritas em Comer (1998, p. 185).

4.1 PROXY

Para Saini (2011, p. 7), *proxy* é um sistema que fica entre o cliente que faz uma requisição e o servidor requisitado que fornece as informações solicitadas. Quando o cliente inicia uma conexão com o servidor de destino, o *proxy* seqüestra a conexão e se apresenta ao servidor como cliente, solicitando a requisição em nome do cliente. Assim que o *proxy* recebe as informações solicitadas, estas informações são retornadas para o cliente, dando a sensação que a comunicação tenha sido feita direta com o servidor de destino.

O diagrama de funcionamento do *proxy* pode ser visto na Figura 5.

Figura 5 - Diagrama de funcionamento do proxy



Fonte: Elaborado pelo autor, 2012.

4.1.1 Cache

Conforme Nagaraj (2004, p. 172), a palavra *cache* se refere ao processo de salvar os dados localmente para uso futuro.

De acordo com Watanabe (2000), existem três tipos de *cache*. São eles:

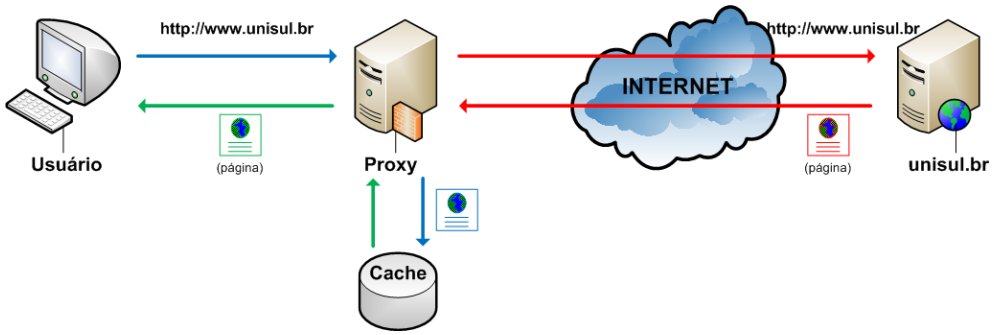
a) **Browser Cache**: a maioria dos navegadores possuem um *cache* próprio, pois é muito comum que os usuários acessem os mesmo objetos frequentemente. Este tipo de *cache* não é compartilhado entre os usuários.

b) **Proxy Cache**: é uma aplicação que pode ser acessada e compartilhada por muitos usuários e esta aplicação fica entre o cliente e o servidor *Web*.

c) **Transparent Proxy Cache**: as conexões são interceptadas pelo *proxy* Web automaticamente, sem necessitar de configuração no *browser* do usuário. Este tipo de *cache* é muito utilizado por provedores de internet.

O diagrama de funcionamento do *proxy* com *cache* pode ser visto na Figura 6.

Figura 6 - Diagrama de funcionamento do proxy com cache



Fonte: Elaborado pelo autor, 2012.

Segundo Wessels (2004, p. 2-3), *cache Web* é o processo de gravação de cópias dos conteúdos obtidos da *Web*, a fim de permitir um acesso futuro mais rápido ao mesmo

conteúdo já gravado, e seu objetivo consiste em aumentar a disponibilidade de largura de banda por reduzir a transmissão de dados; reduzir o congestionamento na rede e melhorar o tempo de resposta das requisições.

4.1.2 Filtros do proxy

Para Saini (2011, p. 8), uma das características de um *proxy Web* é a possibilidade de utilizar filtros, que podem ser utilizados através de regras definidas por um administrador.

Conforme Wessels (2004, p. 54), um *proxy Web* permite ao administrador criar filtros dos mais variados tipos como filtragem de sites, autenticação de usuários entre outros. Os tipos mais comuns são: Endereço IP da estação de trabalho; Endereço do site; Método da requisição; Portas; Autenticação.

4.2 SQUID

Segundo Jeffries (2012) e Wessels (2004, p. 3-4), o Squid é o software baseado no projeto Harvest Cache Daemon da ARPA desenvolvido no início da década de 90, e tem como mentor do seu projeto Duane Wessels, do NLANR que atualmente possui uma grande lista de colaboradores espalhados pelo mundo. Conforme explicam os autores, o termo *Squid* significa Lula, e tem como justificativa distinguir um software do outro.

Para Marcelo (2006, p. 3-6), *Squid* é um dos *proxies* mais utilizados na internet, e suporta todas as características de um *proxy web* como filtragem de sites, autenticação de usuários entre outros.

Atualmente o *Squid* suporta os protocolos de comunicação HTTP, FTP e gopher, e pode ser executadonas plataformas BSDI, DragonfryBSD, FreeBSD, MAC OS X, NetBSD, NeXTStep, OpenBSD, SunOS, GNU/Linux, Digital Unix, IRIX, SCO Unix, AIX, HP-UX, Windows, OS/2 e Solares. (JEFFRIES, 2012; WESSELS, 2004, p. 13).

No arquivo de configuração é necessário definir ao menos três diretivas. São elas o “http_port” onde é definido a porta que o *Squid* aguardará as conexões, a “acl” onde é definida diversos conteúdo como, por exemplo, endereços IP, sites, portas, entre outros e o

“http_access” onde é definido um conjunto de ações utilizando as acls existentes.

Quadro 3 - Exemplo de configuração do Squid

```
# cat squid.conf
acl rede_interna src 192.168.5.0/24
acl sites_permitidos url_regex -i .google.com unisul
acl sites_governo dstdomain .gov.br
acl anexos_proibidos urlpath_regex -i \.exe$ \.com$ \.bat$

http_access deny rede_interna anexos_proibidos
http_access allow rede_interna sites_governo
http_access allow rede_interna sites_permitidos
```

Fonte: Elaborado pelo Autor, 2012.

Do exemplo ilustrado pelo Quadro 3, foram adicionados cinco acls e a função de cada uma delas é informada no Quadro 4.

Quadro 4 - Descrição das acls utilizada no Squid

ACL	**Descrição**
rede_interna	Definido o endereço da rede interna. Na ACL do tipo “src” só pode ser definido endereços IP e endereços de rede
sites_permitidos	Foram definidos duas strings. Na ACL do tipo “url_regex” pode ser adicionado qualquer caracter alfanumérico. O Squid pesquisará as palavras cadastradas em todo o conteúdo da URL
sites_governo	Foi definido apenas a string “.gov.br”. Neste tipo de ACL o squid verifica apenas no endereço do domínio acessado
anexos_proibidos	Foi definido o nome de alguns anexos considerados perigosos. Neste tipo de ACL, o Squid não verifica o endereço do domínio e sim apenas a URL informada após o nome do domínio

Fonte: Elaborado pelo Autor, 2012.

Além da criação das listas de acesso, é necessário criar as regras utilizando as listas criadas anteriormente para ter o efeito desejado.

No exemplo do Quadro 3, a primeira regra bloqueia o acesso da rede interna aos

endereços de *sites* que possuem os anexos informados na lista de acesso “anexos_proibidos”. A segunda regra libera o acesso da rede interna para os *sites* do governo que possuem o endereço terminando com “.gov.br”. Já a terceira regra libera o acesso da rede interna para os *sites* cadastrados na lista de acesso “sites_permitidos”.

5 BANCO DE DADOS

É um sistema de armazenamento de dados, cujo objetivo é organizar e guardar as informações.

No banco de dados existem alguns elementos que precisam ser abordados para entender o funcionamento de um banco de dados, são eles: dados, campos, tabelas e informações (Teorey, 2005, p. 1-3):

a) Dado: é o registro propriamente dito que é adicionado na base de dados;
b) Campo: é o destino onde o Dado será gravado;
c) Tabelas: é a representação matricial onde tem as linhas e colunas;
d) Informação: é o conteúdo extraído da base de dados através de uma consulta, onde o requerente cruza as tabelas necessárias, a fim de obter o melhor resultado.

5.1 SGBD

O SGBD é o programa responsável pelo gerenciamento de uma base de dados. Este programa tem como objetivo retirar a responsabilidade do cliente gerenciar as bases de dados, disponibilizando ao cliente uma interface para que este possa incluir, alterar ou consultar dados. (Costa, 2004, p. 2; Souza, 2004, p. 3-5).

Todos os SGBDs precisam ter as seguintes características:

a) Ter uma estrutura de dados otimizada, para que possa manipular uma grande quantidade de informação;
b) Suportar uma linguagem que possibilite a criação, atualização e consulta de dados armazenados;
c) Ter um mecanismo transacional que garanta a consistência entre as operações dos dados armazenados;
d) Suportar as principais linguagens para manipular bancos de dados. As principais linguagens são SQL e suas variações e o OQL.

5.2 LINGUAGEM SQL

Conforme Oppel e Sheldon (2009, p. 4-5), o SQL é uma linguagem de pesquisa declarativa para banco de dados relacionais.

Antes de existir uma a linguagem SQL, cada banco de dados tinha sua própria linguagem de consulta, e com o tempo os fabricantes viram a necessidade de utilizar uma linguagem padrão para todos os bancos de dados.

"Com o desenvolvimento de um número cada vez maior de SGBDs, fez-se necessário especificar um padrão para a linguagem de acesso, e, em 1986, foi lançada, em um trabalho conjunto do ISO e da ANSI, a primeira versão da linguagem SQL, o SQL-86.". (Costa, 2004, p. 14).

Para Costa (2004, p. 14), os comandos da linguagem SQL podem ser divididos em duas sub-linguagens: a DML e a LDD.

A DML trata dos comandos ligados a manipulação de dados, definindo os comandos para a seleção, inclusão, alteração e exclusão de dados das tabelas. Já a LDD, trata dos comandos utilizados para a criação e manutenção de estruturas e objetos do banco de dados como tabelas, visões e índices.

No Quadro 5 pode ser visto alguns comandos SQL. O primeiro comando adiciona um registro na tabela "alunos". Já o comando seguinte mostra todos os registros existentes nesta tabela.

No terceiro comando, é removido o registro que contém o valor "146375" na coluna "codigo_aluno", e o quarto comando mostra novamente o conteúdo da tabela "alunos".

Quadro 5 - Exemplo de utilização de comandos SQL

```
> INSERT INTO alunos (aluno, codigo_aluno) VALUES ('Andre S. Almeida', '146376');
> SELECT * from alunos;
1|Andre S. Almeida|146376
>DELETE FROM alunos WHERE codigo_aluno='146376';
> SELECT * from alunos;
```

Fonte: Elaborado pelo Autor, 2012.

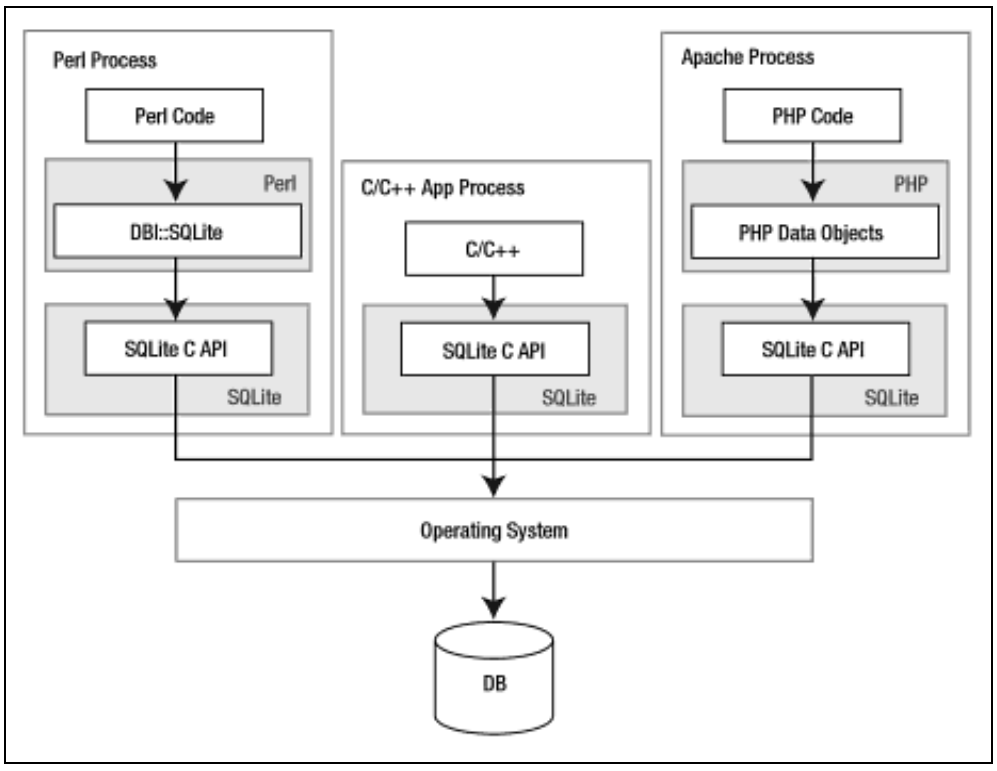
5.3 SQLITE

Segundo SQLite, (2012), o SQLite é uma biblioteca desenvolvida na linguagem de programação C padrão (ANSI), que implementa um banco de dados SQL, que permite o armazenamento de dados em tabelas, além de manipulá-los através de comandos SQL.

O SQLite não necessita de um servidor rodando, pois o mesmo lê e escreve os dados diretamente em arquivos comuns. Este software possui o código fonte aberto e, portanto, é livre ao público para qualquer finalidade, seja ela comercial ou privada.

Segundo Allen e Owens (2010, p. 1-2), a biblioteca do SQLite é suportada por diversas linguagens de programação como por exemplo C/C++, Perl, PHP e Python. Na Figura 7, pode ser visto um exemplo de utilização do SQLite pelas mais diversas linguagens de programação.

Figura 7 - Utilização do SQLite com as mais variadas linguagens de programação



Fonte: Allen e Owens, 2010, p. 2.

No Quadro 6 é ilustrado alguns exemplos de utilização do SQLite. No primeiro comando foi criado o banco que nada mais é do que o arquivo chamado “banco.db”. Dentro do SQLite foi adicionado três tabelas, são elas: “alunos”, “cursos” e “disciplinas”. O quarto comando dentro do SQLite remove a tabela “disciplinas”. O quinto e sexto comando criam índices para as tabelas “alunos” e “cursos”. O sétimo comando mostra a estruturas de todas as tabelas existentes.

Quadro 6 - Utilização do SQLite

```
# sqlite3 banco.db
> CREATE TABLE alunos (id integer, aluno text, codigo_aluno integer, primary key(id));
> CREATE TABLE cursos (id integer, curso text, codigo_curso integer, primary key(id));
> CREATE TABLE disciplinas (id integer, disciplina text, codigo_disciplina integer, primary key(id));
> DROP TABLE disciplinas;
> CREATE INDEX alunos_idx ON alunos(id, aluno, codigo_aluno);
> CREATE INDEX cursos_idx ON cursos(id, curso, codigo_curso);
> .schema
CREATE TABLE alunos (id integer, aluno text, codigo_aluno integer, primary key(id));
CREATE TABLE cursos (id integer, curso text, codigo_curso integer, primary key(id));
CREATE TABLE disciplinas (id integer, disciplina text, codigo_disciplina integer, primary key(id));
CREATE INDEX alunos_idx ON alunos(id, aluno, codigo_aluno);
CREATE INDEX cursos_idx ON cursos(id, curso, codigo_curso);
```

Fonte: Elaborado pelo Autor, 2012.

6 LINGUAGENS DE PROGRAMAÇÃO

Conforme Costa (2007, p. 17), programas são elementos computacionais formados por um conjunto de instruções. Quando solicitado, as instruções que definem um programa de computador são executadas pelo *hardware* dos computadores.

De acordo com Farrer (1989, p. 22), Os computadores só podem executar diretamente os algoritmos expressos em linguagem de máquina, que é um conjunto de instruções capazes de ativar diretamente os dispositivos eletrônicos do computador. Esta linguagem tem vários inconvenientes para os humanos, e é diferente para cada tipo de computador, pois depende de sua arquitetura.

De acordo com Costa (2007, p. 18), foi criada uma linguagem de programação no início da computação moderna chamada linguagem de montagem, mais conhecida como *assembly*. Porém, esta linguagem não era muito simples e com isso era difícil escrever até mesmo pequenos problemas.

Para solucionar este problema foram desenvolvidas linguagens de programação mais fáceis de escrever e gerenciar.

> Para solucionar esses problemas, foram desenvolvidas linguagens especiais, mais fáceis de escrever e gerenciar. Uma operação que demandava dezenas de instruções e que era muito suscetível a erros de escrita pôde ser escrita como uma única instrução. Essas linguagens ficaram conhecidas como linguagens de alto nível. Devido a sua larga utilização, as linguagens de alto nível são chamadas hoje simplesmente de linguagens de programação. (Costa, 2007, p. 18).

6.1 TÉCNICAS DE PROGRAMAÇÃO

De acordo com Lopes (1999, p. 18-23), existem algumas formas de se classificar as linguagens de programação. Uma classificação comumente citada é quanto as técnicas de programação. Pode-se citar como técnicas de programação a programação estruturada, linear, modular e orientada a objetos.

As duas técnicas mais comuns serão abordadas no próximo tópico.

6.1.1 Programação estruturada

Define que todos os programas possíveis podem ser compostos por três estruturas básicas: seleção, repetição e seqüência.

Conforme Costa (2007, p. 23), "O paradigma estruturado propiciou uma grande evolução nas linguagens de programação. Exemplos de linguagens de programação que utilizam esse paradigma são C, *Pascal* e *Basic*."

6.1.2 Programação orientada a objetos

A programação orientada a objetos (OOP – Object-Oriented Programming) se baseia na criação e interação entre diversas unidades de softwares, chamados objetos.

Para Converse (2003, p. 439), "[...] em vez de criar estruturas de dados de um lado e código de outro, suponha que reorganizamos tudo, de modo que as partes associadas do código e dos dados sejam agrupadas."

Segundo Costa (2007, p. 23), a orientação a objetos é uma tendência nas linguagens de programação. As linguagens de programação modernas como, por exemplo, C++, *Java* e *Python* seguem o paradigma orientado a objetos.

6.2 PYTHON

De acordo com Costa (2007, p. 63), o *Python* é uma linguagem de código aberto utilizada para desenvolver aplicações para diversos ambientes, competindo diretamente com outras linguagens de programação como, por exemplo, *Java* e C++.

O Python foi escrito por Guido Van Rossum em 1990, baseando-se na linguagem ABC e foi escrito inicialmente para rodar no sistema operacional Amoeba.

O nome *Python* não tem relação com a família de cobras *python*, ou píton. Na verdade o nome *Python* foi uma homenagem do autor dessa linguagem ao seu programa de TV favorito, "Monty Python's Flying Circus".

Conforme Python (2012), o *Python* está disponível para a maioria dos sistemas operacionais como, por exemplo, *Windows*, *Linux*/*Unix*, OS/2, *Mac*, Amiga e muito outros, inclusive para celulares da série 60 da Nokia.

No Quadro 7 pode ser visto um pequeno programa em Python. No código fonte deste programa foi definido que o arquivo ‘/etc/services’ deve ser lido e no final da leitura deve ser apresentado o número de letras, vogais e consoantes que existem no arquivo.

Quadro 7 - Exemplo de um programa escrito em Python

```
$ cat programa.py
# -*- coding: iso-8859-1 -*-
arquivo = file('/etc/services')
texto = arquivo.read()
vogais = 0
consoantes = 0

print "Contando número de letras, vogais e consoantes\n"

for caracter in texto:
   if caracter in 'aeiou':
      vogais = vogais + 1
   else:
      consoantes = consoantes + 1

print "Vogais: %d\nConsoantes: %d\nTotal de letras: %d" %vogais %consoantes %(len(texto))
```

Fonte: Elaborado pelo autor, 2012.

No Quadro 8 pode ser visto o programa em Python sendo executado e o resultado apresentado pelo programa.

Quadro 8 - Programa em Python sendo executado

```
$ python programa.py
Contando número de letras, vogais e consoantes

Vogais: 104936
Consoantes: 573036
Total de letras: 677972
```

Fonte: Elaborado pelo autor, 2012.

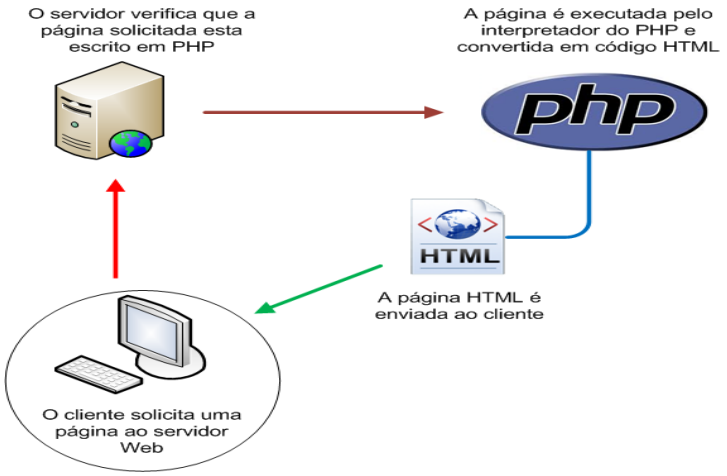
6.3 PHP

De acordo com Converse (2003, p. 3), o PHP é uma linguagem de scripts projetada para *Web*. É comum a utilização de código PHP dentro de uma página HTML,

tornando assim a página dinâmica.

Cada vez que um cliente acessa a página, o código PHP é interpretado pelo servidor *web* e é executado gerando o HTML que será visualizado pelo visitante. Na Figura 8 pode ser visto o diagrama de funcionamento do PHP.

Figura 8 - Diagrama de funcionamento do PHP

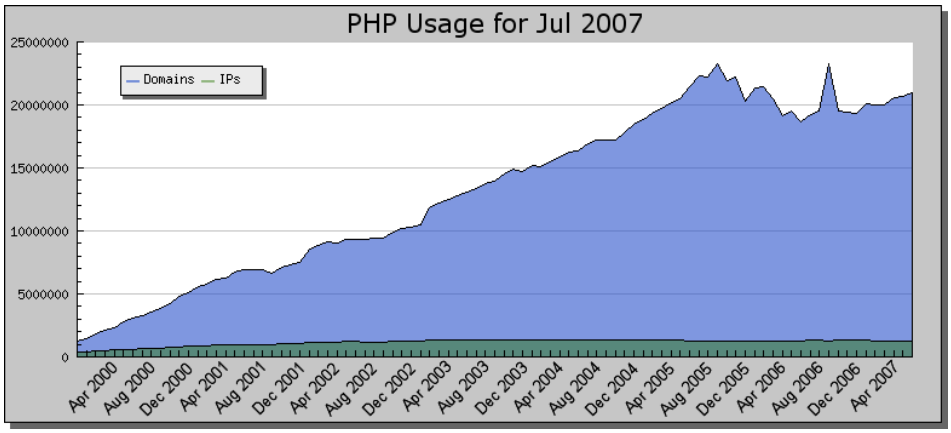


Fonte: Elaborado pelo autor, 2012.

A sigla PHP significava originalmente *Personal Home Page* e atualmente significa *Hypertext PreProcessor*, um acrônimo recursivo que utiliza a própria sigla para se denominar. Em 1994 o trabalho de Rasmus Lerdorf resultou na criação da linguagem PHP. Outras pessoas acabaram adotando e aperfeiçoando o PHP que foi reescrito três vezes para o seu aperfeiçoamento. (Converse, 2003, p. 4).

No final de 2007, o PHP já era utilizado em mais de vinte milhões de domínios do mundo todo como pode ser observado na Figura 9.

Figura 9 - Utilização do PHP na Internet entre abril de 2000 e abril de 2007



Fonte: PHP, 2012.

A sintaxe da linguagem PHP foi baseada em outras linguagens, sendo as principais C, Java e Perl. Esta linguagem de programação pode ser executado na maioria dos sistemas operacionais, como *Windows*, *FreeBSD*, *OpenBSD*, *Linux*, *MAC OS X*, entre outros.

Uma característica importante do PHP é a sua integração com bancos de dados. Ele já possui conexões nativas para vários sistemas de bancos de dados, como por exemplo, o *MySQL*, *PostgreSQL*, *mSQL*, *Oracle*, *Interbase*, *Sybase*, entre outros.

No Quadro 9 pode ser visto um pequeno programa em PHP. Neste programa foi definido que se for informado as variáveis ‘to’, ‘subject’ e ‘content’ via método ‘POST’, o *script* PHP deverá enviar um e-mail para o endereço contido na variável ‘to’.

Caso o método utilizado não seja o ‘POST’, o script será encerrado. Caso as variáveis ‘to’, ‘subject’ e ‘content’ não sejam informadas, o *script* será encerrado informando o motivo antes do final de sua execução.

Quadro 9 - Exemplo de um programa escrito em PHP

```
<?php
if ($_SERVER['REQUEST_METHOD'] != 'POST') die ('Método inválido!');
if (!$_REQUEST['to']) die ('Destinarário não informado!');
if (!$_REQUEST['subject']) die ('Assunto não informado!');
if (!$_REQUEST['content']) die ('Mensagem não informada!');
// Declarando variáveis
$from = 'andre@andre.adm.br';
$to = $_REQUEST['to'];
$subject = $_REQUEST['subject'];
$content = $_REQUEST['content'];
// Preenchendo o cabeçalho do e-mail
$headers = 'MIME-Version: 1.0' . "\r\n";
$headers .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";
$headers .= 'From: André S. Almeida <' . $from . '>' . "\r\n";
$headers .='To: <' . $to . '>' . "\r\n";

// Envia o e-mail
mail($to, $subject, $message, $headers);
?>
```

Fonte: Elaborado pelo autor, 2012.

6.4 SHELL SCRIPT

Segundo Neves (2008, p. 83), *shell* é o módulo que atua como interface entre o usuário e o sistema operacional e possui um conjunto de comandos internos que permite ao usuário solicitar serviços do sistema operacional.

Os sistemas *Unix* possuem uma grande variedade de *shells*, já que os *shells* são programas independentes do *kernel*. O Quadro 10 cita os principais *shells* e suas características.

Quadro 10 - Tipos de shells e suas características

Shell	**Características**
Bourne Shell (sh)	Desenvolvido por Stephen Bourne da Bell Labs. Este Shell também é chamado de Standard Shell e é o mais utilizado por ter sido por vários anos o único Shell existente.
Bourne Again Shell (bash)	Este é o shell padrão do Linux e sua utilização vem crescendo muito. O Bash é quase 100% compatível com o Bourne Shell, além também de trazer consigo implementações do Korn Shell e algumas características do C Shell.
Korn Shell (ksh)	Foi desenvolvido por David Korn da Bell Labs, é uma atualização do Bourne Shell e por esse motivo todos os comandos do Bourne Shell são reconhecidos pelo Korn Shell.
C Shell (csh)	Desenvolvido por Bill Joy da Berkeley University. É o Shell mais utilizado nos ambientes *BSD e XENIX. A estruturação de seus comandos é bem similar à da linguagem C.

Fonte: Neves, 2008, p. 88-89.

Linguagens de script são linguagens imperativas e estruturadas que são destinadas à escrita de *scripts*.

Conforme Jargas (2008, p. 24), *script* é uma lista de comandos para serem executados em seqüência, podendo ser utilizados para automatizar tarefas repetitivas.

Pode-se concluir então que os *shells scripts* são scripts implementados em ambiente *Shell*. São praticamente os mesmos comandos que seriam digitados manualmente em um *Shell* para se obter o mesmo resultado.

6.4.1 Awk

Segundo Robins (2002, p. 18-19), a linguagem *Awk* foi projetada para simplificar muitas tarefas comuns de processamento de textos. O *Awk* é uma linguagem interpretada e poderosa. O Quadro 11 demonstra alguns parâmetros do comando *awk*.

Quadro 11 - Parâmetros do comando awk

Opção	**Descrição**
-f	Define os arquivos que serão analisados.
-F	Esta opção define um delimitador.
-v	Utilizado para declarar variáveis antes de o programa ser executado.

Fonte: Elaborado pelo autor, 2012.

O Quadro 12 exibe alguns exemplos de utilização do comando *Awk*.

Quadro 12 - Utilização do comando awk com alguns parâmetros

```
$ cat /etc/passwd | grep -v \# | awk -F \: {' print "Usuario: " $1 ", UID: " $3 '}
Mostra a primeira e terceira coluna do arquivo /etc/passwd.

$ cat /etc/passwd | grep -v \# | awk -F \: '$3 >=200 { print "Usuario: " $1 ", UID: " $3 }'
Mostra todos os usuários que possuem o UID maior ou igual a 500.


$ ps axww | awk {' print $1 '}
Mostra a identificação de todos os processos existentes.


$ ps axww | awk {' print $1 "|" $5 '}
Mostra a identificação e o nome de todos os processos existentes.
```

Fonte: Elaborado pelo autor, 2012.

6.4.2 Cut

Segundo Jargas (2008, p. 429), o comando *cut* extrai campos ou trechos de uma

linha. Possui um formato bem flexível de especificação de campos e caracteres conforme mostra o Quadro 13.

Quadro 13 - Parâmetros do comando cut

Opção	**Descrição**
-c	Mostra somente os caracteres informados
-d	Define o delimitador. Se não informado, o padrão é o TAB.
-f	Mostra somente os campos informados. Esta opção precisa ser utilizada com a opção '-d'.

Fonte: Elaborado pelo autor, 2012.

No Quadro 14 pode ser visto alguns exemplos de utilização do comando *cut*, além da descrição de cada linha utilizada.

Quadro 14 - Utilização do comando cut com alguns parâmetros

$ echo 'jan/fev/mar/abr/mai/jun/jul/ago/set/out/nov/dez' | cut -d\/ -f5-7 saída: mai/jun/jul Imprime do quinto ao sétimo campo (-f5-7) separado pela barra (-d\/).
$ echo 'jan/fev/mar/abr/mai/jun/jul/ago/set/out/nov/dez' | cut -d\/ -f10- saída: out/nov/dez Imprime do décimo campo em diante (-f10-) tendo como delimitador a barra.
$ echo 'jan/fev/mar/abr/mai/jun/jul/ago/set/out/nov/dez' | cut -c1-3,16-24,41- saída: jan/mai/jun/nov/dez Imprime do primeiro ao terceiro caracter, do décimo sexto ao vigésimo quarto caracter e do quadragésimo primeiro caracter em diante (-c1-3,16-24,41-)

Fonte: Elaborado pelo autor, 2012.

6.4.3 Grep

Conforme Jargas (2008, p. 437), o comando *grep* "procura em arquivo ou textos por linhas que contém determinado padrão de pesquisa."

Quadro 15 - Parâmetros do comando grep

Opção	Descrição
-v	Exibe as linhas que não possuem a expressão especificada
-i	Não leva em consideração a diferença entre letras maiúsculas e minúsculas
-c	Exibe apenas a quantidade de linhas encontradas contendo a expressão especificada
-n	Exibe o número da linha que possui a expressão especificada

Fonte: Elaborado pelo autor, 2012.

O Quadro 16 exibe alguns exemplo de utilização do comando *grep* e seus parâmetros.

Quadro 16 - Utilização do comando grep com alguns parâmetros

```
$ ps w | grep -v PID
  477 s000  S+     0:00.05 -bash
  772 s002  Ss     0:00.24 –bash

$ ps w | grep -i pid
  PID   TT  STAT      TIME COMMAND
 1196 s002  R+     0:00.00 grep -i pid

$ grep -c \/tcp /etc/services
4964

$ grep -n ' 80/tcp' /etc/services
234:http            80/tcp     www www-http # World Wide Web HTTP

$ grep -r ' 80/tcp' /etc/
/etc/services:http            80/tcp     www www-http # World Wide Web HTTP
```

Fonte: Elaborado pelo autor, 2012.

6.4.4 Sed

Segundo Jargas (2008, p. 444), o *sed* é um editor de textos não-interativo e

programável que possibilita a execução de vários comandos para edição de um texto. Geralmente é utilizado para trocar uma string pela outra.

Quadro 17 - Parâmetros do comando sed

Opção	**Descrição**
-E	Interpreta o script utilizando expressão regular moderna

Fonte: Elaborado pelo autor, 2012.

O Quadro 18 exibe alguns exemplos da utilização do comando *sed*.

Quadro 18 - Utilização do comando sed

```
$ sed -n 234p /etc/services
http            80/tcp     www www-http # World Wide Web HTTP

$ echo UNISUL | sed 's%.*%Universidade do Sul de Santa Catarina%'
Universidade do Sul de Santa Catarina

$ sed -n '233,/tcp/p' /etc/services
http            80/udp     www www-http # World Wide Web HTTP
http            80/tcp     www www-http # World Wide Web HTTP
```

Fonte: Elaborado pelo autor, 2012.

7 F-WEB MANAGER

A fim de cumprir com os objetivos traçados para o desenvolvimento do trabalho, este capítulo aborda uma topologia de rede como espelho para grande parte das infra-estruturas de redes, além da arquitetura da ferramenta, bem como todo o desmembramento necessário em termos de recursos para a solução do problema.

Como base na arquitetura proposta foi criado um cenário virtualizado, idêntico, para simular a comunicação entre o equipamento cliente e servidor onde ficará a ferramenta, a fim de facilitar o processo de desenvolvimento. Sobre uma perspectiva de diferenças para este trabalho de um ambiente virtual e real, não há absolutamente nenhum impedimento ou tratamento específico que deva ser conduzido para um ou outro ambiente.

Este capítulo aborda também todas as ferramentas externas que foram utilizadas durante o trabalho e que compõe o pacote do *F-Web Manager*. São detalhados todos os processos de instalação bem como a organização do ambiente.

7.1 ARQUITETURA

A Figura 10 ilustra a topologia utilizada para o desenvolvimento do *F-Web Manager*. Os requisitos mínimos para a estrutura devem contemplar:

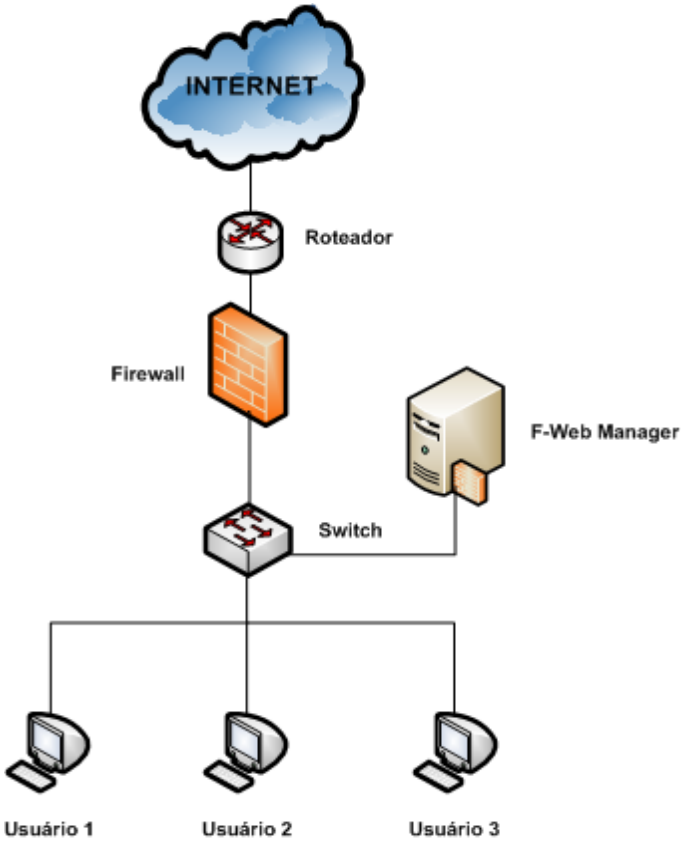
a) Pelo menos uma interface *Ethernet*, do contrário não será possível utilizar a ferramenta;
b) O sistema operacional *FreeBSD*;
c) A ferramenta de *Proxy Web Squid*;
d) Servidor *Web* com PHP;
e) O número máximo de equipamentos gerenciados está limitado aos recursos do equipamento, como por exemplo, memória e CPU.

O *F-Web Manager* também pode ser instalado em um *firewall*, desde que a ferramenta para controle de navegação seja o *Squid*. Caso não houver um *Firewall* na rede, o *F-Web Manager* pode ser instalado em um equipamento separado, desde que este equipamento seja colocado entre a rede interna e a *Internet*.

O Roteador da Figura 10 é o responsável pelo gerenciamento do *link* de *Internet*,

que deve estar conectado diretamente ao *Firewall* que é o ativo responsável pela filtragem e encaminhamento de pacotes. O *Firewall* por sua vez deve estar conectado ao *Switch*, onde estará também o *F-Web Manager* que será o equipamento responsável por gerenciar a navegação das conexões diretas (*proxy* manual). Os equipamentos identificados como Usuário 1, Usuário 2 e Usuário 3 são os ativos que representam os computadores que serão gerenciados pelo *F-Web Manager*.

Figura 10 - Topologia física do ambiente com o F-Web Manager



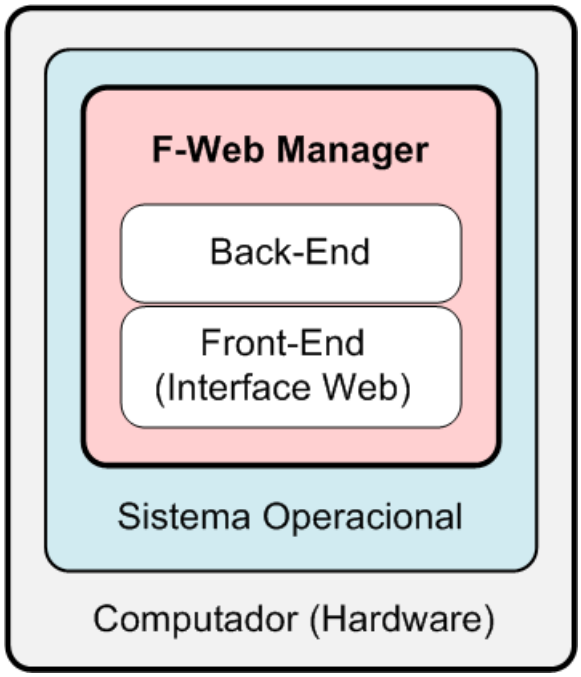
Fonte: Elaborado pelo Autor, 2012.

O papel do *F-Web Manager* é permitir o gerenciamento da navegação *Web* utilizando categorias de *sites* que são atualizados automaticamente com um servidor de atualização de conteúdo.

Cabe ao administrador, diante do *F-Web manager*, criar toda a configuração necessária alinhada a sua infra-estrutura física. Todo o contato do Administrador com o *F-Web Manager* será feito através de uma interface de gerenciamento web, sem necessidade de conhecimentos específicos na tecnologia utilizada para o gerenciamento

Na Figura 11 pode ser visto a estrutura do computador com o *F-Web Manager*.

Figura 11 - Estrutura do Computador com o F-Web Manager



Fonte: Elaborado pelo Autor, 2012.

7.2 AMBIENTE UTILIZADO

A fim de definir o ambiente para o desenvolvimento do *F-Web Manager* este capítulo descreve todo o hardware utilizado, bem como sistemas operacionais, máquinas virtuais e ferramentas envolvidas, além das configurações necessárias para a execução do projeto.

Como citado no capítulo de introdução do trabalho, todo ambiente será criado em máquinas virtuais de maneira a simular um ambiente real, sem que isso afete seu funcionamento. Outros tópicos deste mesmo capítulo detalharão o cenário.

7.2.1 Hardware

Para implementação da ferramenta foi utilizado um *notebook* Apple MacBook MB467LL/A com processador Intel Core 2 Duo de 2,4 Ghz, 4 GB de memória RAM e HD SATA 2 de 500 GB.

Figura 12 - Apple MacBook MB467LL/A

Fonte: NotebookNotes, 2012.

O notebook possui uma interface de rede *Apple Gigabit Ethernet* e uma placa *WiFi AirPort Extreme* 802.11a/b/g/n.

Não são necessárias descrições adicionais do hardware em questão, uma vez que todo o desenvolvimento será feito em um ambiente virtual, sendo essas questões irrelevantes para o trabalho.

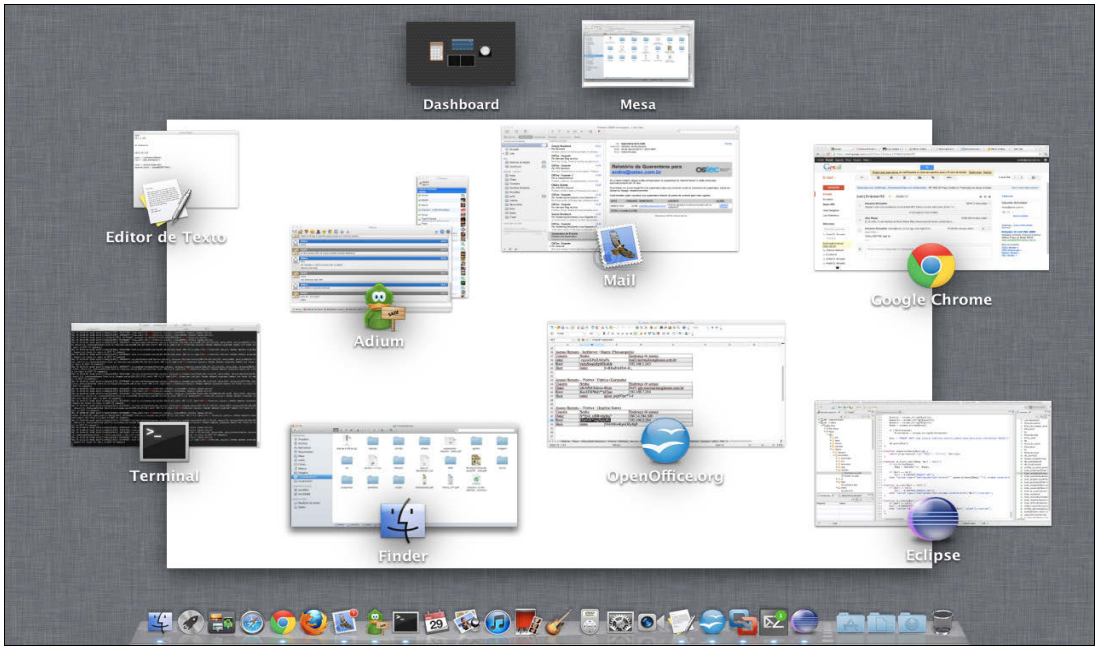
7.2.2 Sistema operacional hospedeiro

O sistema operacional instalado no *notebook* é o *MAC OS X Lion* 10.7.4. Como todo o trabalho foi desenvolvimento sobre máquinas virtuais, nenhuma configuração específica deve ser feita no sistema operacional que irá receber a ferramenta de virtualização.

Neste caso não será mencionado a instalação do *MAC OS X* no notebook que nada interfere no desenvolvimento do trabalho.

Os sistemas operacionais utilizados nas máquinas virtuais serão abordados nos próximos tópicos, onde são definidas as configurações das máquinas.

Figura 13 - Sistema operacional Mac OS X Lion 10.7.4

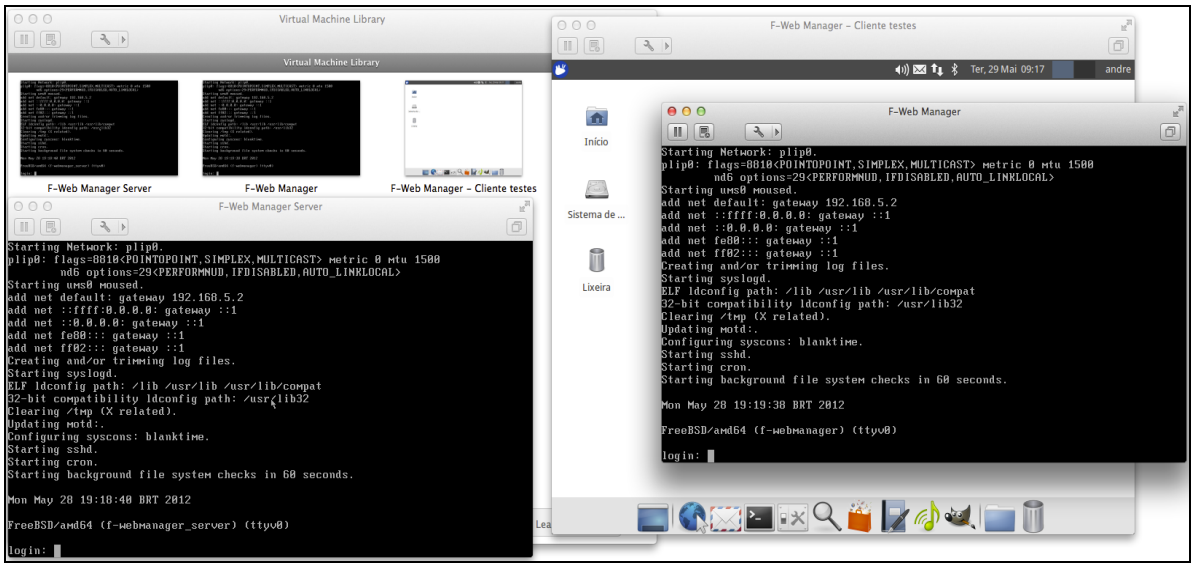


Fonte: Elaborado pelo Autor, 2012.

7.2.3 VMWare Fusion

A ferramenta *VMWare Fusion* foi utilizada para a criação das máquinas virtuais bem como seu gerenciamento e usada a versão 4.1.2, última versão até o início do trabalho.

Figura 14 - VMWare Fusion 4.1.2



Fonte: Elaborado pelo Autor, 2012.

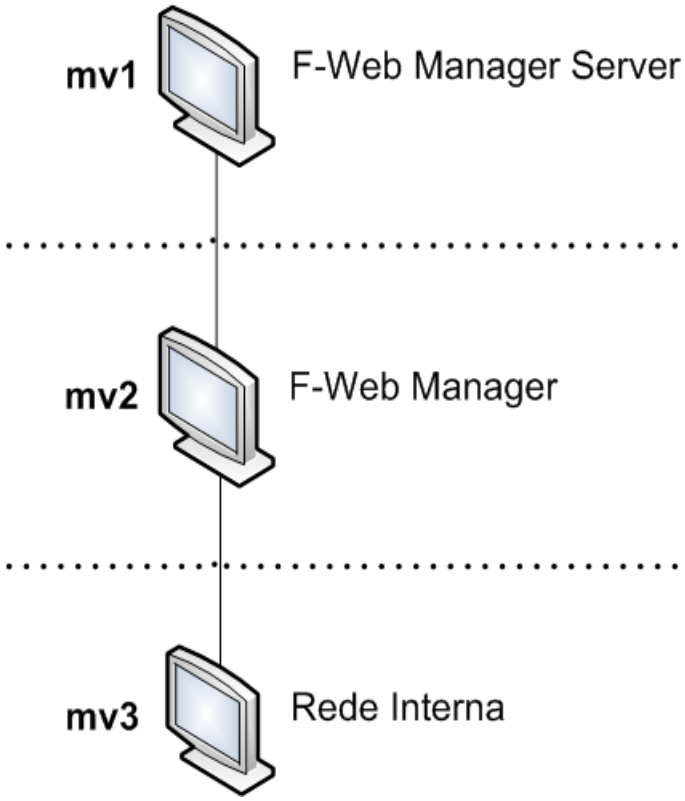
O *VMWare* é uma ferramenta proprietária, portanto a versão utilizada neste trabalho foi adquirida através do site do fabricante e devidamente registrada. Para desenvolvimento do trabalho foi feito o download do arquivo com extensão ".dmg" para o sistema operacional *Mac OS X Lion*.

7.2.3.1 Máquinas virtuais

Após a instalação do *VMWare Fusion*, foram criadas três máquinas virtuais, sendo distribuídas da seguinte maneira:

a) uma onde ficará o servidor do *F-Web Manager* que será responsável por disponibilizar as atualizações das categorias;
b) uma onde será desenvolvida o *F-Web Manager* propriamente dito;
c) uma para simular um computador localizado na rede internet do ambiente, a fim de realizar os testes.

Figura 15 - Estrutura das máquinas virtuais



Fonte: Elaborado pelo Autor, 2012.

7.2.3.2 Máquina do F-Web Manager Server

A máquina que está representada por vm1 da Figura 15, será utilizada pelo aplicativo servidor do *F-Web Manager* que será responsável por disponibilizar as atualizações das categorias, além de gerenciar a conexão do aplicativo cliente.

Foi instalado o sistema operacional *FreeBSD* 9.0 apenas com os pacotes mínimos, além das ferramentas *SQLite* e *Python* que serão utilizados pelo aplicativo servidor.

Para esta máquina foram destinados 128MB de memória RAM e 5Gb de espaço em disco, sendo o suficiente para rodar o aplicativo.

7.2.3.3 Máquina do F-Web Manager

O *F-Web Manager* foi instalado na máquina virtual representada na Figura 15 por mv2. Para esta máquina foram destinados 256MB de memória RAM e 10GB de espaço em disco. O sistema operacional instalado foi também o *FreeBSD* 9.0, versão amd64.

Figura 16 - Menu do Boot Loader do FreeBSD 9.0

Fonte: FreeBSD Handbook, 2012.

A essa máquina foi atribuída apenas uma interface de rede que foi destinada a conexão do *F-Web Manager* com a rede interna e também com a Internet.

7.2.3.4 Máquina para testes

A máquina virtual destinada para testes e representada na Figura 15 como mv3 tem o sistema operacional *GNU/Linux XUbuntu* com a instalação padrão. Esta máquina servirá apenas para testes de acesso a Internet, não sendo relevante um maior detalhamento da mesma para teste trabalho.

7.3 PRÉ-REQUISITOS E EMPACOTAMENTO

Na intenção de no futuro tornar mais fácil a portabilidade da ferramenta para outros sistemas operacionais ou mesmo outras versões do *FreeBSD*, todas as ferramentas necessárias para a instalação e desenvolvimento do *F-Web Manager* foram baixados no formato código fonte.

Isso também facilita a atualização dos programas que são pré-requisitos para o funcionamento do *F-Web Manager*, como *Lighttpd*, PHP, *Squid* e demais bibliotecas relacionadas ao funcionamento destes programas.

7.3.1 Estrutura de diretórios

Todas as ferramentas que são pré-requisitos para a instalação e utilização do *F-Web Manager* foram compilados abaixo do diretório "/fwebmanager" para facilitar o gerenciamento do mesmo, como alterações ou atualizações da ferramenta ou dos programas necessários para o seu funcionamento.

De acordo com o Quadro 19, é possível visualizar todos os subdiretórios abaixo do "/fwebmanager" e suas funções para o ambiente.

Quadro 19 - Subdiretórios abaixo do diretório /fwebmanager

Subdiretório	**Função**
backend	Diretório onde serão armazenados os aplicativos responsáveis pelo gerenciamento da ferramenta e também para atualização das categorias
banco_de_dados	Diretório onde serão armazenados os bancos de dados do F-Web Manager, que serão utilizados pelo backend e frontend para consultas e armazenamentos de informações.
chroot	Contém todas as ferramentas instaladas que são necessárias para o funcionamento do F-Web Mananger.
conf	Contém os arquivos de configuração gerados pelo backend
frontend	Contém a interface gráfica da ferramenta
logs	Diretório utilizado para gravação de logs
pacotes	Diretório onde foram salvos todos os pacotes de instalação das ferramentas necessárias para o funcionamento do F-Web Manager
temp	Diretório utilizado para criação de arquivos e diretórios temporários

Fonte: Elaborado pelo Autor, 2012.

Dentro da pasta "/fwebmanager" foram criadas seis subdiretórios. O subdiretório "backend" contém os scripts escritos em *Python* que são responsáveis por toda a ferramenta, desde a listagem de opções até a alteração de informações no banco de dados

O subdiretório "banco_de_dados" é utilizado único e exclusivamente para armazenar os bancos de dados utilizados. O subdiretório "chroot" é onde foram compilados todos os programas necessários para o funcionamento do *F-Web Manager*.

O subdiretório "conf" é onde ficarão os arquivos de configurações criados pelo *backend*, enquanto o subdiretório "frontend" é onde ficará a parte visual da ferramenta que é a interface na qual o Administrador utilizará para gerenciar a ferramenta.

Por último, o diretório "pacotes" é onde foram armazenados todos os pacotes de instalação, no formato código fonte, necessários para a implementação do *F-Web Manager*.

7.3.2 Instalação de ferramentas

Nos tópicos que seguem serão descritos os passos da instalação dos programas necessários para o desenvolvimento e funcionamento do *F-Web Manager*.

7.3.2.1 Pacotes de desenvolvimento

Para facilitar a instalação das demais ferramentas necessárias ao *F-Web Manager* foram instaladas as seguintes ferramentas através do sistema de *ports* do *FreeBSD*:

a) libtool;
b) gmake;
c) libiconv;
d) gettext;
e) perl;
f) m4;
g) autoconf;
h) pcre.

O processo de instalação pelo *ports* do *FreeBSD* baixa automaticamente os códigos fontes dos programas e em seguinte compila os mesmos. Para isso bastou entrar nas pastas dos pacotes desejados, abaixo do diretório "/usr/ports/" e em seguida digitado o comando "make install clean".

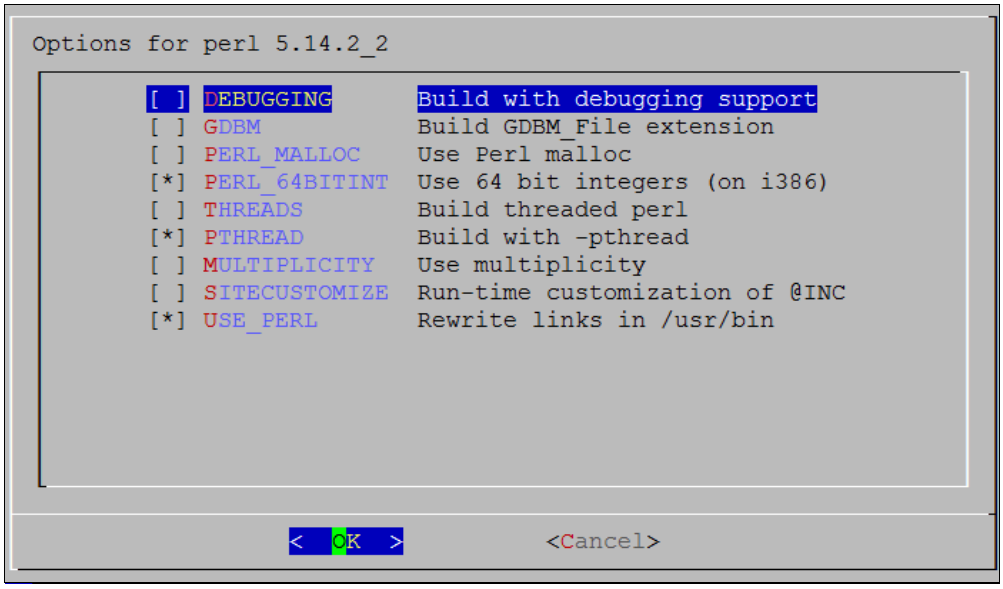
Quadro 20 - Processo de instalação dos pacotes de desenvolvimento

```
f-webmanager# cd /usr/ports/devel/libtool/
f-webmanager# make install clean

f-webmanager# cd /usr/ports/converters/libiconv/
f-webmanager# make install clean

f-webmanager# cd /usr/ports/devel/gettext/
f-webmanager# make install clean

f-webmanager# cd /usr/ports/devel/gmake/
f-webmanager# make install clean

f-webmanager# cd /usr/ports/lang/perl5.14/
f-webmanager# make install clean

f-webmanager# cd /usr/ports/devel/m4/
f-webmanager# make install clean

f-webmanager# cd /usr/ports/devel/autoconf/
f-webmanager# make install clean

f-webmanager# cd /usr/ports/devel/pcre/
f-webmanager# make install clean
```

Fonte: Elaborado pelo Autor, 2012.

Para compilar o aplicativo “perl” é necessário definir os recursos que devem ser ativos na compilação como ilustrado na Figura 17.

Figura 17 - Opções de instalação do Perl



Fonte: Elaborado pelo Autor, 2012.

7.3.2.2 Bibliotecas

As bibliotecas libxml, libjpeg, libpng e freetype foram instaladas a fim de facilitar a implementação da interface gráfica do *F-Web Manager*. O quadro 21 mostra as bibliotecas instaladas e suas versões.

Quadro 21 - Bibliotecas e versões correspondentes

Biblioteca	**Versão**
libxml	2-2.8.0
libjpeg	8d
libpng	1.5.10
freetype	2.4.9

Fonte: Elaborado pelo Autor, 2012.

As instalações das bibliotecas foram realizadas seguindo a seqüência de comandos ilustrada no Quadro 22.

Quadro 22 - Comandos para instalação das bibliotecas necessárias ao F-Web Manager

```
f-webmanager# cd /fwebmanager/pacotes/libxml2-2.8.0/
f-webmanager# ./configure --prefix=/fwebmanager/chroot
f-webmanager# make
f-webmanager# make install

f-webmanager# cd /fwebmanager/pacotes/jpeg-8d/
f-webmanager# ./configure --prefix=/fwebmanager/chroot
f-webmanager# make
f-webmanager# make install

f-webmanager# cd /fwebmanager/pacotes/libpng-1.5.10/
f-webmanager# ./configure --prefix=/fwebmanager/chroot
f-webmanager# make
f-webmanager# make install

f-webmanager# cd /fwebmanager/pacotes/freetype-2.4.9/
f-webmanager# GNUMAKE=/usr/local/bin/gmake ./configure --
prefix=/fwebmanager/chroot
f-webmanager# gmake
f-webmanager# gmake install
```

Fonte: Elaborado pelo Autor, 2012.

O parâmetro “—prefix” utilizado no comando “./configure” informa em qual diretório o programa deve ser instalado. Todas as ferramentas utilizadas para o desenvolvimento do *F-Web Manager* foram compiladas no diretório “/fwebmanager/chroot”.

7.3.2.3 GD

GD é uma biblioteca para criação dinâmica de imagens. É muito utilizado para gerar gráficos e tabelas. Neste trabalho foi utilizado à versão “2.0.35” e o processo de instalação é ilustrado no Quadro 23.

Quadro 23 - Instalação do GD

```
f-webmanager# cd /fwebmanager/pacotes/gd-2.0.35/
f-webmanager# ./configure --prefix=/fwebmanager/chroot --with-jpeg=/fwebmanager/chroot --with-
png=/fwebmanager/chroot –with-freetype=/fwebmanager/chroot
f-webmanager# make
f-webmanager# make install
```

Fonte: Elaborado pelo Autor, 2012.

Na instalação do GD, mais especificadamente no comando "./configure", foram usados os parâmetros "--with-jpeg", "--with-png" e "--with-freetype" para apontar o diretório onde foi instalado as bibliotecas libjpeg, libpng e freetype.

7.3.2.4 SQLite

O *SQLite* é uma biblioteca desenvolvida com a linguagem de programação C que implementa um banco de dados. Isto significa que as ferramentas que utilizam esta biblioteca não precisam executar um processo separado para o banco de dados, pois a própria biblioteca lê e grava os dados diretamente em arquivos.

O *SQLite* é muito conhecido por sua performance superior a outros bancos de dados como por exemplo *MySQL* e *PostgreSQL*, porém se a quantidade de dados ultrapassar os 10GB, este banco de dados perde muita performance se não tiver sendo utilizado chave primária e índices para otimização.

Como neste trabalho o banco de dados vai ser utilizado apenas para armazenar as configurações do *F-Web Manager* e as informações dos sites acessados, que vai ser rotacionado mensalmente, não teremos problemas com o desempenho do banco de dados.

Foi utilizada a versão 3.7.12.1 do *SQLite* para o desenvolvimento deste trabalho e o procedimento de instalação pode ser visto no Quadro 24.

Quadro 24 - Instalação do SQLite

```
f-webmanager# cd /fwebmanager/pacotes/sqlite-autoconf-3071201/
f-webmanager# ./configure --prefix=/fwebmanager/chroot
f-webmanager# make
f-webmanager# make install
```

Fonte: Elaborado pelo Autor, 2012.

O *SQLite* foi instalado com o parâmetro “--prefix” que é definido para indicar o local onde a ferramenta será instalada. Neste caso, o *SQLite* foi instalado no diretório “/fwebmanager/chroot”.

7.3.2.5 Python

O *Python* é uma linguagem de programação interpretada, isto quer dizer que os programas desenvolvidos nesta linguagem não precisam ser compilados para então serem executados como na linguagem C. O interpretador do *Python* lê o código fonte do programa e interpreta-o diretamente durante a sua execução.

Para o desenvolvimento do *backend* da ferramenta foi utilizado a versão 2.7.3 do *Python*.

Quadro 25 - Instalação do Python

f-webmanager# cd /fwebmanager/pacotes/Python-2.7.3/
f-webmanager# vi setup.py
Adicionar no arquivo setup.py o binário '/fwebmanager/chroot/bin/sqlite3' na linha que contém a palavra ‘sqlite_inc_paths’
f-webmanager# ./configure --prefix=/fwebmanager/chroot
f-webmanager# make
f-webmanager# make install

Fonte: Elaborado pelo Autor, 2012.

Na instalação do *Python* foi necessário primeiramente editar o arquivo setup.py e adicionar a linha “/fwebmanager/chroot/bin/sqlite3” para que o *Python* pudesse compilar o módulo para integração com o banco de dados *SQLite*.

Em seguida foi utilizado o comando “./configure” com o parâmetro “--prefix" para definir que o *Python* deveria ser instalado no diretório “/fwebmanager/chroot”. O comando “make” serve para compilar a ferramenta e o comando “make install” para instalar.

7.3.2.6 PHP

Após a instalação do banco de dados *SQLite* e de todas as bibliotecas necessárias para o desenvolvimento do *F-Web Manager*, foi instalado o PHP. O PHP será o responsável por interpretar os códigos dinâmicos da interface gráfica do *F-Web Manager* e transformá-los em código estático para que possa ser interpretado pelos navegadores.

O código em PHP permite que seja criada uma interface entre o *front-end* e o *back-end* do *F-Web Manager*, possibilitando a execução do *back-end* e de comandos do sistema, alteração de arquivos de configuração e dados e o acesso ao banco de dados, tanto na gravação de informações como na consulta das mesmas.

Os comandos listados no Quadro 26 foram utilizados para a instalação do PHP.

Quadro 26 - Instalação do PHP

```
f-webmanager# cd /fwebmanager/pacotes/php-5.4.3/
f-webmanager# ./configure --prefix=/fwebmanager/chroot --with-jpeg-dir=/fwebmanager/chroot --with-png-dir=/fwebmanager/chroot --with-freetype-dir=/fwebmanager/chroot --with-gd --with-libxml-dir=/fwebmanager/chroot --with-sqlite3
f-webmanager# make
f-webmanager# make install
```

Fonte: Elaborado pelo Autor, 2012.

Foram utilizados vários parâmetros no comando "./configure". O Quadro 27 descreve a função de cada um desses parâmetros.

Quadro 27 - Descrição dos parâmetros utilizados para a instalação do PHP

Parâmetros	**Descrição**
--prefix	Local onde o PHP será instalado
--with-jpeg-dir	Diretório onde foi instalada a libjpeg
--with-png-dir	Diretório onde foi instalada a libpng
--with-freetype	Diretório onde foi instalado o freetype
--with-gd	Ativa o suporte ao GD
--with-libxml-dir	Diretório onde foi instada a libxml
--with-sqlite3	Ativa o módulo para integração com o banco de dados SQLite

Fonte: Elaborado pelo Autor, 2012.

Depois de ter compilado o PHP foi necessário copiar o arquivo "php.ini" do

diretório de instalação do PHP para o diretório onde este foi instalado. Este arquivo de exemplo serve para os propósitos deste trabalho e o procedimento da cópia é ilustrado no Quadro 28.

Quadro 28 - Procedimento de cópia do arquivo “php.ini”

```
f-webmanager# cd /fwebmanager/pacotes/php-5.4.3/
f-webmanager# cp php.ini-production /fwebmanager/chroot/lib/php.ini
```

Fonte: Elaborado pelo Autor, 2012.

7.3.2.7 Lighttpd

Lighttpd é um servidor *Web* rápido e seguro que é otimizado para ambientes de alto desempenho. Este servidor *Web* utiliza muito menos recursos de máquina como memória e processamento se comparado a seus concorrentes como o *Apache* e *Nginx*.

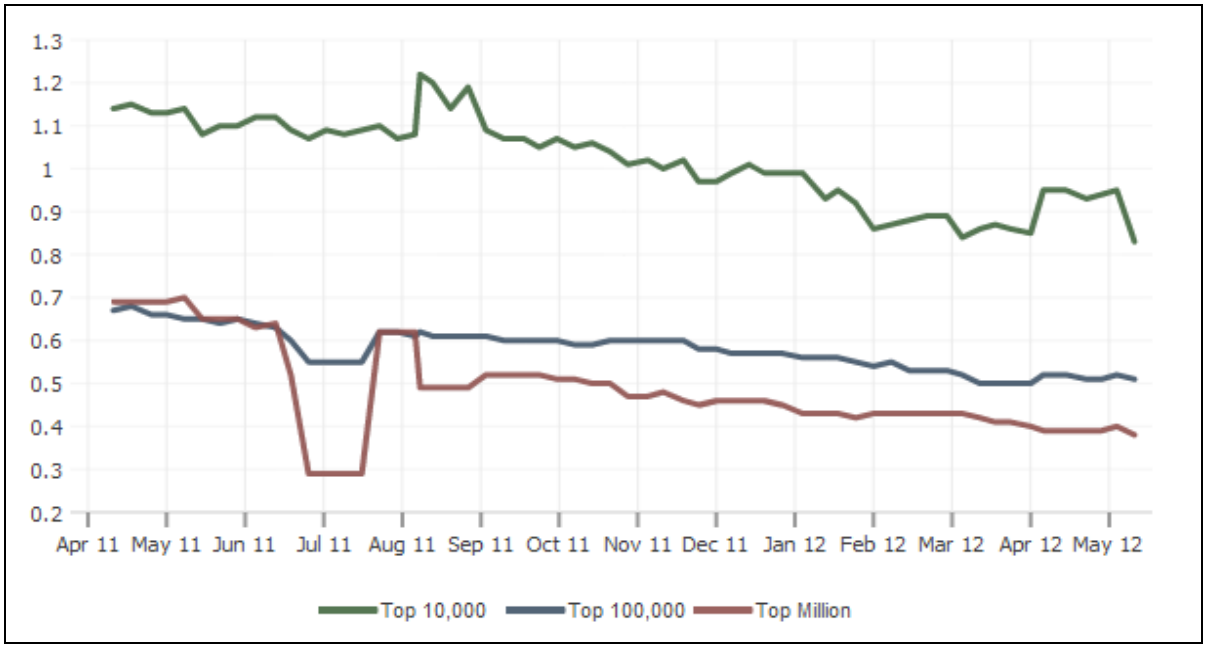
Figura 18 - Logotipo do Lighttpd

Fonte: Lighttpd, 2012.

Este servidor *Web* ainda não é muito popular e por isso não ultrapassa 2% de utilização em toda a Internet, no entanto dos 493.799 sites que utilizam o *Lighttpd* em todo o mundo, 29.325 destes sites estão entre os mais visitados na Internet.

Na Figura 19 pode ser visto a utilização do *Lighttpd* nos 10.000, 100.000 e 1.000.000 de sites mais visitado na Internet.

Figura 19 - Utilização do lighttpd nos sites mais visitados na Internet



Fonte: Builtwith, 2012.

O *Lighttpd* será o servidor Web responsável por permitir o acesso dos clientes à interface gráfica do *F-Web Manager*. Foi utilizada a versão 1.4.31 do *Lighttpd* e o processo de instalação é ilustrado no Quadro 29.

Quadro 29 - Instalação do Lighttpd

```
f-webmanager# cd /fwebmanager/pacotes/lighttpd-1.4.31/
f-webmanager# ./configure --prefix=/fwebmanager/chroot
f-webmanager# make
f-webmanager# make install
```

Fonte: Elaborado pelo Autor, 2012.

O *Lighttpd* foi compilado com o parâmetro “--prefix" onde foi definido o local onde este foi instalado. Em seguida foram utilizados os comandos “make” e “make install” respectivamente para compilar e instalar o aplicativo.

Para correto funcionamento do *Lighttpd* foi criado o arquivo de configuração chamado “lighttpd.conf” no diretório “/fwebmanager/chroot/etc/”.

Neste arquivo foram definidos o diretório onde ficará o *frontend* do *F-Web Manager*, os *logs* do *Lighttpd* e também o caminho do binário do PHP que será o interpretador dos códigos fonte escritos em PHP.

O arquivo de configuração do *Lighttpd* pode ser visto no Quadro 30.

Quadro 30 - Arquivo de configuração do Lighttpd

```
f-webmanager# cat /fwebmanager/chroot/etc/lighttpd.conf
server.modules = (
        "mod_access",
        "mod_accesslog",
        "mod_fastcgi"
)
server.document-root = "/fwebmanager/frontend/"
server.bind = "0.0.0.0"
server.port = 80
server.username = "www"
server.groupname = "www"
server.tag = "Lighttpd"
server.errorlog = "/fwebmanager/logs/lighttpd-erros.log"
accesslog.filename = "/fwebmanager/logs/lighttpd-acessos.log"
server.pid-file = "/fwebmanager/logs/lighttpd.pid"
index-file.names = ( "index.php", "index.html", "index.htm" )
url.access-deny  = ( "~", ".inc" )
dir-listing.activate = "disable"
fastcgi.server = (
        ".php" => ((
                "bin-path" => "/fwebmanager/chroot/bin/php-cgi",
                "socket" => "/fwebmanager/temp/php.socket" + var.PID
))
)
```

Fonte: Elaborado pelo Autor, 2012.

Dentre as principais opções estão a definição do diretório onde ficará o *frontend* que é especificado na opção “server.document-root”. Na opção “index-file.names” é definido

os arquivos que serão utilizados para exibição da página inicial, caso não exista nenhum destes arquivos, não será mostrado nenhum conteúdo, pois a definição “dir-listing-activate” foi desativada.

Na opção “fastcgi.server” foi definido o caminho do php e o diretório onde será criado os arquivos de *socket*. Em seguida foi necessário executar o *Lighttpd*, conforme é ilustrado no Quadro 31.

Quadro 31 - Execução do Lighttpd

```
f-webmanager#/fwebmanager/chroot/sbin/lighttpd -f /fwebmanager/chroot/etc/lighttpd.conf
```

Fonte: Elaborado pelo Autor, 2012.

7.3.2.8 Squid

Squid é um servidor *proxy* utilizado para controle de navegação *Web*, além de prover uma otimização no tempo de resposta das conexões se utilizado em conjunto com cache.

O Objetivo do *F-Web Manager* será disponibilizar uma interface gráfica de fácil acesso para que os administradores possam definir o que os gerenciados poderão utilizar, porém, por baixo a ferramenta responsável por fazer os devidos bloqueios ou liberação será o *Squid* que suporta os protocolos HTTP, HTTPS, além também do protocolo FTP que não será utilizado neste trabalho.

A versão do *Squid* utilizada para o desenvolvimento deste trabalho foi a 3.1.19 e o procedimento de instalação é ilustrado no Quadro 32.

Quadro 32 - Procedimento de instalação do Squid

```
f-webmanager#/fwebmanager/pacotes/squid-3.1.19/
f-webmanager#./configure --prefix=/fwebmanager/chroot --enable-ssl --with-openssl=/usr --with-large-files --enable-large-cache-files --enable-err-languages=Portuguese --enable-default-err-language=Portuguese
f-webmanager#make
f-webmanager# make install
```

Fonte: Elaborado pelo Autor, 2012.

Na instalação do *Squid* foram definidos alguns parâmetros que são detalhados no Quadro 33.

Quadro 33 - Opções de instalação do Squid

Parâmetros	Descrição
--prefix	Local onde será instalado o Squid
--enable-ssl	Habilita o suporte a SSL (Secure Sockets Layer)
--with-openssl	Diretório onde o OpenSSL foi instalado
--with-large-files	Habilita o suporte a arquivos maiores que 2Gb
--with-large-cache-files	Habilita o suporte a cache com arquivos maiores que 2Gb
--enable-err-languages	Define quais idiomas devem ser instalados para as mensagens de bloqueio ou falhas de conexão
--enable-default-err-language	Define qual idioma padrão deverá ser utilizado para apresentação de mensagens de bloqueio ou falhas de conexão

Fonte: Elaborado pelo Autor, 2012.

7.4 TESTES DAS FERRAMENTAS INSTALADAS

Para verificar que o *Lighttpd*, PHP e suas dependências estavam funcionando normalmente foi criado um arquivo no formado “php” com uma função da linguagem PHP que cria um resumo das configurações do PHP e dos recursos habilitados.

Para a criação do arquivo foram seguidos os passos descritos no Quadro 34.

Quadro 34 - Criando arquivo para teste do Lighttpd e PHP

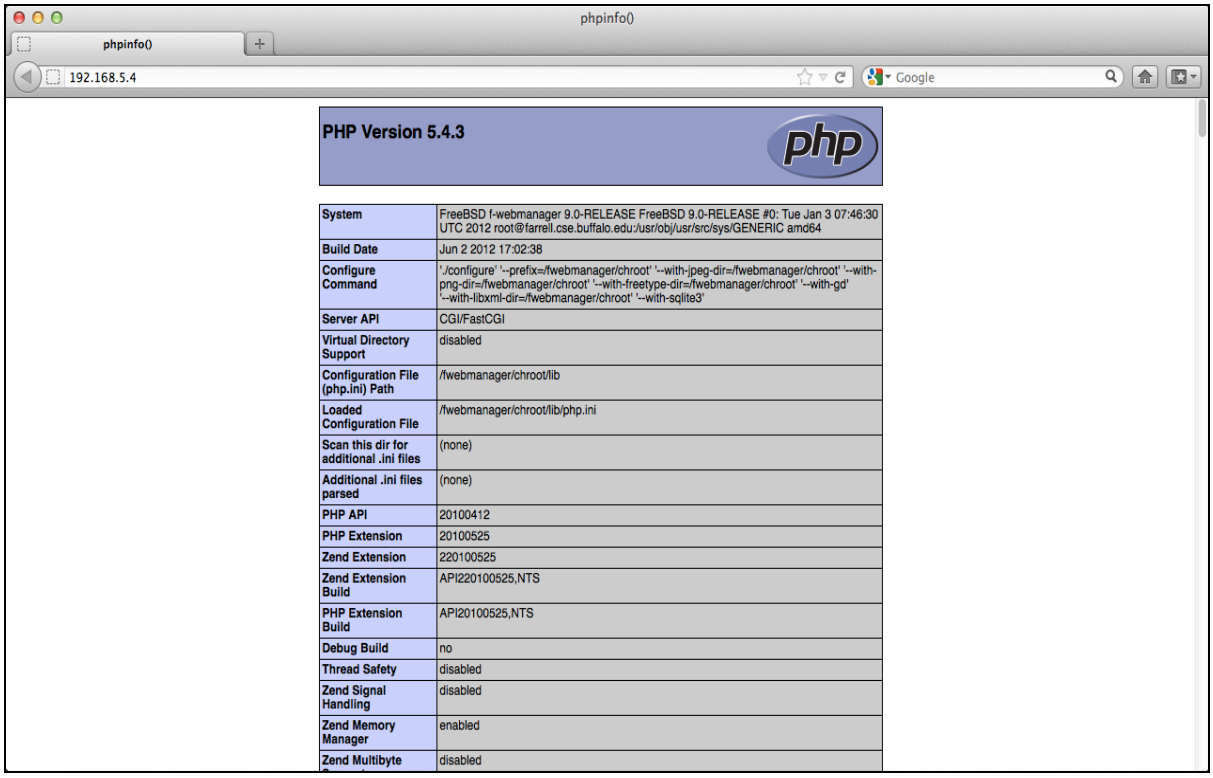
```
f-webmanager# cd /fwebmanager/frontend/
f-webmanager# echo "<? phpinfo(); ?>" >index.php
```

Fonte: Elaborado pelo Autor, 2012.

Após a criação do arquivo o mesmo deve ser acessado através de um navegador *Web*.

A Figura 20 mostra o arquivo aberto no navegador *Firefox*.

Figura 20 - Teste de utilização do Lighttpd com PHP



System	FreeBSD f-webmanager 9.0-RELEASE FreeBSD 9.0-RELEASE #0: Tue Jan 3 07:46:30 UTC 2012 root@farrell.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC amd64
Build Date	Jun 2 2012 17:02:38
Configure Command	'./configure' '--prefix=/fwebmanager/chroot' '--with-jpeg-dir=/fwebmanager/chroot' '--with-png-dir=/fwebmanager/chroot' '--with-freetype-dir=/fwebmanager/chroot' '--with-gd' '--with-libxml-dir=/fwebmanager/chroot' '--with-sqlite3'
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/fwebmanager/chroot/lib
Loaded Configuration File	/fwebmanager/chroot/lib/php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS
PHP Extension Build	API20100525,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte	disabled

Fonte: Elaborado pelo Autor, 2012.

Para verificar que o *Squid* estava funcionando normalmente, foi alterado no arquivo de configuração para bloquear qualquer requisição solicitada.

Quadro 35 - Alteração do arquivo de configuração e execução do Squid

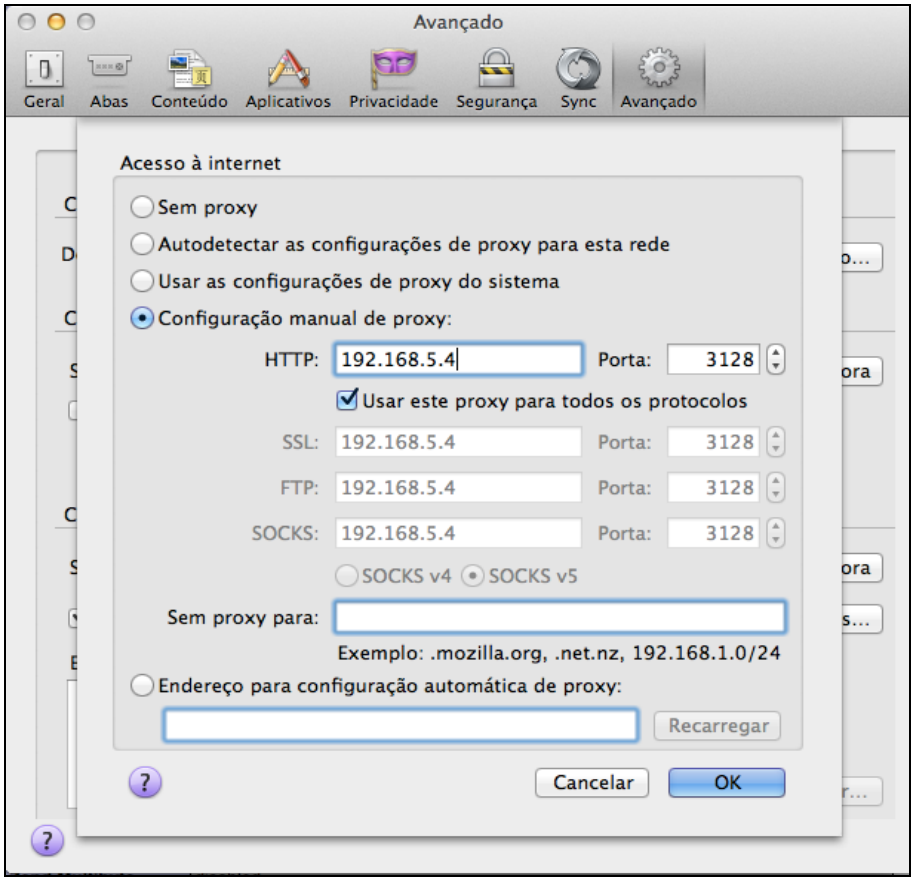
```
f-webmanager# cd /fwebmanager/chroot/etc/
f-webmanager# sed 's%http_access allow localnet%http_access deny localnet%' squid.conf >/fwebmanager/temp/squid.conf
f-webmanager# mv /fwebmanager/temp/squid.conf squid.conf
f-webmanager# chown nobody /fwebmanager/chroot/var/cache/fwebmanager/chroot/var/logs
f-webmanager#/fwebmanager/chroot/sbin/squid
```

Fonte: Elaborado pelo Autor, 2012.

Para poder testar o *Squid* foi necessário definir no navegador, o endereço IP e porta onde está rodando o *Squid*. No ambiente utilizado, o *Squid* ficou sendo executado na

máquina virtual que está com o endereço IP 192.168.5.4 e na porta 3128. O procedimento para esta definição é ilustrado na Figura 21.

Figura 21 - Alteração na definição de proxy no navegador



Fonte: Elaborado pelo Autor, 2012.

Na Figura 22 pode ser visto um teste de utilização do *Squid*. No teste foi acessado o site "www.unisul.br" e o *Squid* bloqueou a requisição conforme configurado.

Figura 22 - Teste de utilização do Squid

Fonte: Elaborado pelo Autor, 2012.

8 MODELAGEM E DESENVOLVIMENTO DA FERRAMENTA

O desenvolvimento do *F-Web Manager* foi dividido em duas partes, o *back-end* e *front-end*. Os próximos tópicos descrevem como a ferramenta foi dividida e desenvolvida, visando facilitar sua implementação.

8.1 ANÁLISE DE REQUISITOS

A análise de requisitos também conhecida como engenharia de requisitos, serve para detalhar as características da ferramenta. Essa análise de requisitos é vital para o desenvolvimento da ferramenta, pois determina o sucesso ou fracasso do projeto.

8.1.1 Análise de requisitos do F-Web Manager

O *F-Web Manager* deve contar com uma interface agradável para o usuário, dispondo de maneira fácil e inteligente os recursos da ferramenta.

As funcionalidades desta interface são:

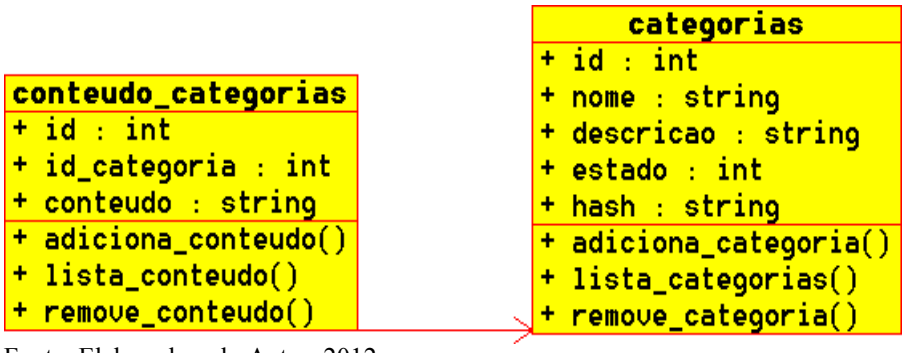
a) Gerenciamento de categorias de sites locais;
b) Gerenciamento de listas de acesso;
c) Gerenciamento de regras e política de acesso;
d) Gerenciamento de atualizações das categorias não locais com o servidor do *F-Web Manager*;
e) Gerenciamento de usuários da ferramenta;
f) Visualização de relatórios das alterações realizadas na ferramenta;
g) Visualização dos sites acessados pelos ativos controlados.

8.2 DIAGRAMA DE CLASSES

Diagrama de classes trata-se de uma estrutura lógica estática em uma superfície de duas dimensões mostrando uma coleção de elementos declarativos de modelo como classes, tipos e seus respectivos conteúdos e relações.

Na Figura 23 pode ser visto o diagrama de classes do *F-Web Manager Server*.

Figura 23 - Diagrama de classes do F-Web Manager Server



Fonte: Elaborado pelo Autor, 2012.

Abaixo pode ser visualizado o diagrama de classes do *F-Web Manager*, onde pode ser visto a lista de todas as tabelas e suas relações, os tipos das colunas e as funções utilizadas em cada tabela.

Figura 24 - Diagrama de classes do F-Web Manager

Fonte: Elaborado pelo Autor, 2012.

No Quadro 36 pode ser visualizada a descrição de cada tabela.

Quadro 36 - Descrição das tabelas do banco de dados

categorias_locais	Tabela que armazena as categorias locais.
conteudo_categorias_locais	Tabela que armazena o conteúdo das categorias locais, além do código da categoria local.
categorias	Tabela que armazena as categorias não locais.
conteudo_categorias	Tabela que armazena o conteúdo das categorias que foram sincronizadas com o servidor, além do código da categoria.
listas_de_acesso	Tabela que armazena as listas de acesso.
conteudo_listas_de_acesso	Tabela que armazena o conteúdo das listas de acesso, assim como o código da lista de acesso relacionada.
regras_de_acesso	Tabela que armazena as regras de acesso, além do código da categoria local, categoria não local, lista de acesso e ação.
politica	Tabela que armazena a informação da política de acesso padrão.
atualizar_categorias	Tabela que armazena a lista das categorias que não estão sincronizadas com o servidor e que precisam ter o seu conteúdo atualizado.
novas_categorias	Tabela que armazena a lista das categorias que existem no servidor, mas não existem no cliente.
usuarios	Tabela que armazena a lista de usuários, assim como as permissões dos usuários.
logs (f-web manager.db)	Tabela que possui os registros das operações realizadas pelo administrador através da interface de gerenciamento.
logs (squid.db)	Tabela que possui os registros dos sites acessados pelo ativos gerenciados.

Fonte: Elaborado pelo Autor, 2012.

Como pôde ser visto no Quadro anterior, o servidor e o cliente possuem duas tabelas com o mesmo nome. O nome das tabelas em questão são “categorias” e “conteudo_categorias”. Estas tabelas foram criadas com o mesmo propósito nas duas ferramentas e por este motivo foi citado apenas uma vez na descrição.

Nos quadros a seguir é mostrada a descrição das colunas de todas as tabelas utilizadas.

Quadro 37 - Dicionário da tabela categorias do F-Web Manager Server

Nome	**Tipo**	**Descriçao**
id	INTEGER	Código da categoria.
nome	TEXT	Nome da categoria.
descricao	TEXT	Descrição da categoria.
estado	INTEGER	Estado da categoria.
hash	TEXT	Hash do conteúdo da categoria.

Fonte: Elaborado pelo Autor, 2012.

Quadro 38 - Dicionário da tabela conteudo_categorias do F-Web Manager Server

Nome	**Tipo**	**Descrição**
id	INTEGER	Código do conteúdo da categoria.
id_categoria	INTEGER	Código da categoria.
conteudo	TEXT	Conteúdo da categoria.

Fonte: Elaborado pelo Autor, 2012.

Quadro 39 - Dicionário da tabela categorias_locais do F-Web Manager

Nome	**Tipo**	**Descrição**
id	INTEGER	Código da categoria local.
nome	TEXT	Nome da categoria local.
descricao	TEXT	Descrição da categoria local.
estado	INTEGER	Estado da categoria local.

Fonte: Elaborado pelo Autor, 2012.

Quadro 40 - Dicionário da tabela conteudo_categorias_locais do F-Web Manager

Nome	**Tipo**	**Descrição**
id	INTEGER	Código do conteúdo da categoria local.
id_categoria	INTEGER	Código da categoria local.
conteudo	TEXT	Conteúdo da categoria local.

Fonte: Elaborado pelo Autor, 2012.

Quadro 41 - Dicionário da tabela conteudo_categorias do F-Web Manager

Nome	**Tipo**	**Descrição**
id	INTEGER	Código do conteúdo.
id_categoria	INTEGER	Código da categoria.
conteudo	TEXT	Conteúdo da categoria.

Fonte: Elaborado pelo Autor, 2012.

Quadro 42 - Dicionário da tabela categorias do F-Web Manager

Nome	**Tipo**	**Descrição**
id	INTEGER	Código da categoria.
nome	TEXT	Nome da categoria.
descricao	TEXT	Descrição da categoria.
estado	INTEGER	Estado da categoria.
atualizar	INTEGER	Campo utilizado para saber se a categoria precisa ser atualizada com o servidor.
hash	TEXT	Hash do conteúdo da categoria, utilizado para saber se o conteúdo é o mesmo que o conteúdo da categoria do servidor.

Fonte: Elaborado pelo Autor, 2012.

Quadro 43 - Dicionário da tabela regras_de_acesso do F-Web Manager

Nome	Tipo	Descrição
id	INTEGER	Código da regra de acesso.
lista	INTEGER	Codigo da lista de acesso.
categoria	INTEGER	Código da categoria.
categoria_local	INTEGER	Código da categoria local.
acao	INTEGER	Ação da regra de acesso. As ações possíveis são: 0 para aceitar e 1 para bloquear.
estado	INTEGER	Estado da regra de acesso.
indice	INTEGER	Índice (ordem) da regra de acesso.

Fonte: Elaborado pelo Autor, 2012.

Quadro 44 - Dicionário da tabela listas_de_acesso do F-Web Manager

Nome	**Tipo**	**Descrição**
id	INTEGER	Código da lista de acesso.
nome	TEXT	Nome da lista de acesso.
tipo	TEXT	Tipo da lista de acesso. O tipo pode ser "ip_origem" ou "porta_destino".
descricao	TEXT	Descrição da lista de acesso.
estado	INTEGER	Estado da categoria.

Fonte: Elaborado pelo Autor, 2012.

Quadro 45 - Dicionário da tabela conteudo_listas_de_acesso do F-Web Manager

Nome	**Tipo**	**Descrição**
id	INTEGER	Código do conteúdo.
id_lista_de_acesso	INTEGER	Código da lista de acesso.
conteudo	TEXT	Conteúdo da lista de acesso.

Fonte: Elaborado pelo Autor, 2012.

Quadro 46 - Dicionário da tabela politica do F-Web Manager

Nome	Tipo	Descrição
politica	INTEGER	Política das regras de acesso. As políticas possíveis são: 0 para aceitar tudo e 1 para bloquear tudo.

Fonte: Elaborado pelo Autor, 2012.

Quadro 47 - Dicionário da tabela atualizar_categorias do F-Web Manager

Nome	Tipo	Descrição
id	INTEGER	Código do item.
categoria	TEXT	Nome da categoria.

Fonte: Elaborado pelo Autor, 2012.

Quadro 48 - Dicionário da tabela novas_categorias do F-Web Manager

Nome	Tipo	Descrição
id	INTEGER	Código do item.
categoria	TEXT	Nome da categoria.

Fonte: Elaborado pelo Autor, 2012.

Quadro 49 - Dicionário da tabela usuarios do F-Web Manager

Nome	Tipo	Descrição
id	INTEGER	Código do usuário.
usuario	TEXT	Nome do usuário utilizado no processo de autenticação.
senha	TEXT	Senha do usuário.
nome	TEXT	Nome completo do usuário.
perml_categorias	INTEGER	Permissão de leitura no módulo categorias.
perml_listas_de_acesso	INTEGER	Permissão de leitura no módulo listas_de_acesso.
perml_regras_de_acesso	INTEGER	Permissão de leitura no módulo regras_de_acesso.
perml_atualizacoes	INTEGER	Permissão de leitura no módulo atualizacoes.
perml_usuarios	INTEGER	Permissão de leitura no módulo usuarios.
perml_logs	INTEGER	Permissão de leitura no módulo logs.
permg_categorias	INTEGER	Permissão de escrita no módulo categorias.
permg_listas_de_acesso	INTEGER	Permissão de escrita no módulo listas_de_acesso.
permg_regras_de_acesso	INTEGER	Permissão de escrita no módulo regras_de_acesso.
permg_atualizacoes	INTEGER	Permissão de escrita no módulo atualizacoes.
permg_usuarios	INTEGER	Permissão de escrita no módulo usuarios.
permg_logs	INTEGER	Permissão de escrita no módulo logs.
estado	INTEGER	Estado do usuário.

Fonte: Elaborado pelo Autor, 2012.

Quadro 50 - Dicionário da tabela logs (f-web manager.db) do F-Web Manager

Nome	Tipo	Descrição
id	INTEGER	Código do registro.
datahora	INTEGER	Data e hora do registro.
usuario	TEXT	Nome do usuário que fez a operação.
modulo	TEXT	Nome do módulo onde foi feita a alteração.
acao	TEXT	Descrição do que foi feito.
informacoes	TEXT	Maiores informações do procedimento realizado.
cod_operacao	INTEGER	Código da operação. O código possível é 0 para “OK” e 1 para “ERRO”.

Fonte: Elaborado pelo Autor, 2012.

Quadro 51 - Dicionário da tabela logs (squid.db) do F-Web Manager

Nome	Tipo	Descrição
id	INTEGER	Código do registro.
datahora	INTEGER	Data e hora do acesso ao site registrado.
endereço_ip	TEXT	Endereço IP do equipamento que fez o acesso.
acao	TEXT	Ação que foi tomada quando o acesso foi feito. As ações possíveis são "ACCESS_DENIED" para acesso bloqueado ou qualquer outro código para acesso permitido.
metodo	TEXT	Método utilizado na conexão. Os tipos que podem ser utilizados na navegação são "GET" ou "POST".
endereco	TEXT	Endereço do site que foi acessado.
codigo_http	INTEGER	Còdigo HTTP retornado pelo servidor de destino.

Fonte: Elaborado pelo Autor, 2012.

8.3 BACK-END

Back-end é a parte da ferramenta que o usuário não visualiza. É o *back-end* quem fez a integração entre a interface gráfica que é visualizada pelo usuário e o banco de dados onde ficam as configurações e conteúdo de categorias e listas de acesso.

Para desenvolvimento do *back-end* foi utilizado a linguagem de programação *Python*.

Esta linguagem de programação já possui módulos para comunicação com o banco de dados sqlite3, manipulação de arquivos e utilização de *socket* (interface que permite conversar com outros programas e computadores utilizando o modelo de comunicação cliente/servidor), além de diversos outros módulos.

Quadro 52 - Lista dos módulos do Python utilizados no back-end

Módulo	Descrição
getopt	Responsável por analisar os argumentos da linha de comando.

hashlib	Implementa uma interface para gerar hash de strings. Os algoritmos suportados são SHA1, SHA224, SHA256, SHA384 e SHA512.
re	Fornece operações de expressões regulares.
socket	Fornece acesso à interface de socket BSD. Utilizado para prover comunicação com outras ferramentas e outros computadores através do modelo cliente/servidor.
subprocess	Permite gerar novos processos, conectar em interfaces de entrada e saída e obter a saída dos processos e o código de retorno.
sys	Permite utilizar algumas variáveis utilizadas e mantidas pelo interpretador e funções que interagem fortemente com o interpretador.

Fonte: Elaborado pelo Autor, 2012.

8.4 FRONT-END

O *front-end* nada mais é do que a interface gráfica que é visualizada pelo usuário. Esta é a parte da ferramenta que interage com o administrador e que envia as solicitações feitas para o *back-end*, que retorna para o *front-end* com o resultado da operação solicitada.

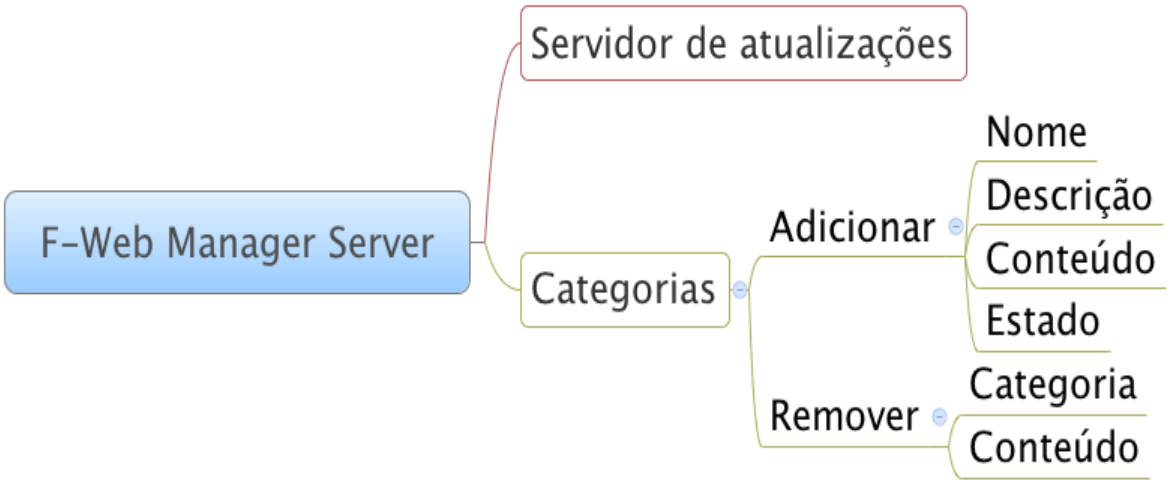
A linguagem de programação utilizada foi o PHP, em conjunto com HTML, CSS e Java Script para desenvolvimento da interface gráfica. Foi utilizada a função exec() do PHP para chamar o *back-end* que é utilizado para qualquer procedimento de adição, edição, listagem e remoção.

8.5 F-WEB MANAGER SERVER

O *F-Web Manager Server* é a ferramenta responsável por disponibilizar para o cliente do *F-Web Manager* as atualizações das categorias.

Nesta ferramenta foi escrito dois *back-ends* que podem ser visualizados na Figura 25 e serão abordados nos próximos itens.

Figura 25 - Back-ends do F-Web Manager Server



Fonte: Elaborado pelo Autor, 2012.

8.5.1 Servidor de atualizações

Este *back-end* é responsável por disponibilizar para o cliente do *F-Web Manager* as atualizações das categorias. Para desenvolvimento desta ferramenta foi utilizada a função 'socket' do *Python*, que permite a comunicação entre equipamentos utilizando uma arquitetura cliente/servidor.

Esta ferramenta quando executada, fica apenas esperando novas conexões na porta 2727/tcp, e a partir do momento que uma conexão é recebida, é esperada uma determinada seqüência de comandos para que o servidor responda as requisições feitas pelo cliente.

No Quadro 53 pode ser visualizado o *F-Web Manager Server* sendo executado, e no comando seguinte pode ser verificado que a porta 2727 está sendo utilizada pelo mesmo.

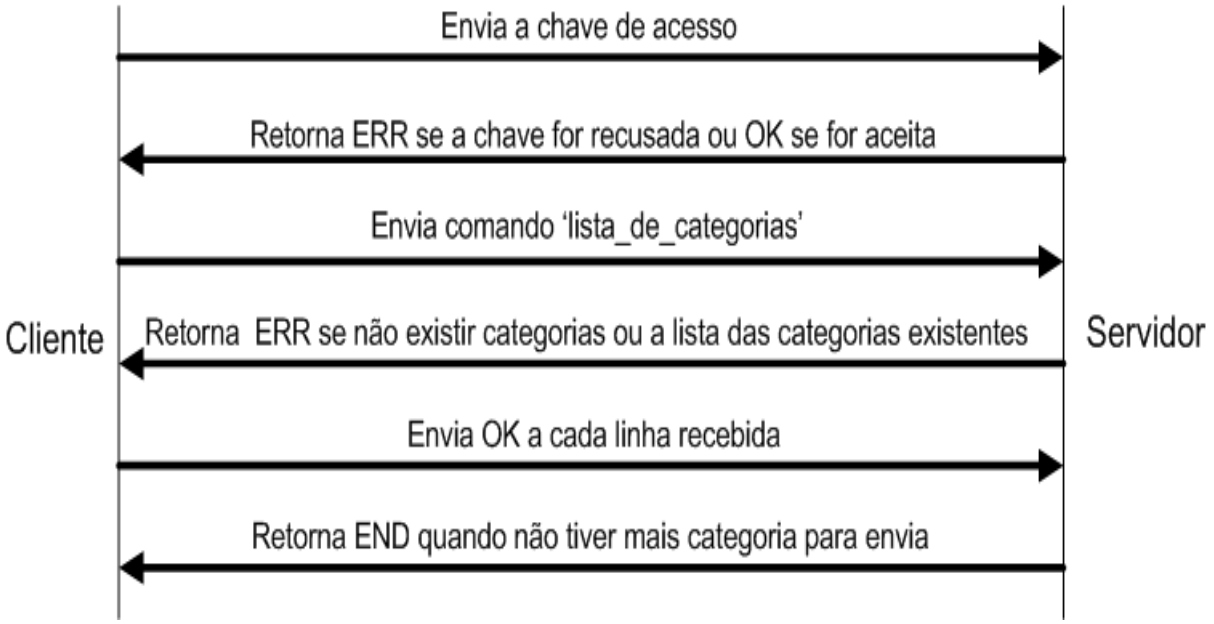
Quadro 53 - Execução do F-Web Manager Server

```
# /fwebmanager/backend/servidor.py &
# sockstat -l | grep python
root     python2.7  1269  3  tcp4   *:2727                *:*
```

Fonte: Elaborado pelo Autor, 2012.

Na Figura 26 pode ser visualizada a estrutura de comunicação entre o *F-Web Manager* e o *F-Web Manager Server* para listagem das categorias existentes no servidor.

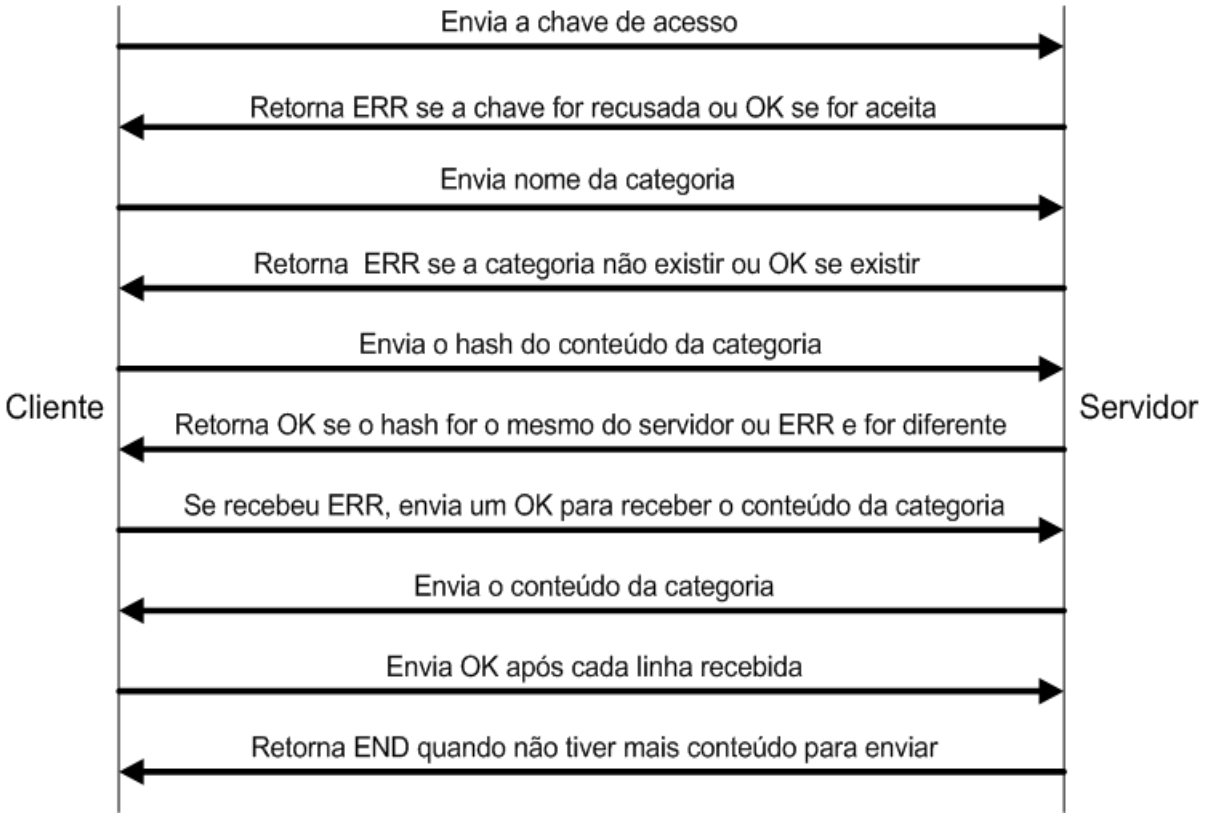
Figura 26 - Estrutura de comunicação para listagem de categorias do servidor



Fonte: Elaborado pelo Autor, 2012.

Na Figura 27 é mostrada a estrutura de comunicação entre o cliente e o servidor do *F-Web Manager* para verificar e atualizar o conteúdo das categorias.

Figura 27 - Estrutura de comunicação para listagem de conteúdo das categorias do servidor



Fonte: Elaborado pelo Autor, 2012.

8.5.2 Categorias

Este *back-end* é responsável por cadastrar novas categorias e conteúdo nas categorias do *F-Web Manager Server*, além de listar e remover as categorias existentes.

No Quadro 54 podem ser visualizados os comandos que foram disponibilizados no *back-end* para gerenciamento das categorias no servidor.

Quadro 54 - Comandos para gerenciamento de categorias no servidor

```
# /fwebmanager/backend/categorias.py --acao=adicionar --opcao=conteudo --categoria=id_categoria --conteudo=http://www.site3.com,http://www.site4.com
# /fwebmanager/backend/categorias.py --acao=adicionar --opcao=arquivo --categoria=id_categoria --arquivo=/tmp/sites.txt

# /fwebmanager/backend/categorias.py --acao=listar --opcao=categorias
# /fwebmanager/backend/categorias.py --acao=listar --opcao=categoria --categoria=id_categoria

# /fwebmanager/backend/categorias.py --acao=remover --opcao=categoria --categoria=id_categoria
# /fwebmanager/backend/categorias.py --acao=remover --opcao=conteudo --categoria=id_categoria --conteudo=id1_conteudo,id2_conteudo
```

Fonte: Elaborado pelo Autor, 2012.

No exemplo descrito no quadro acima, o primeiro comando adiciona uma categoria no servidor especificando dois endereços que devem ser adicionados com conteúdo. O segundo comando também adiciona a categoria, porém utiliza um arquivo com a lista dos sites para gerenciar a categoria. O terceiro comando apenas lista o conteúdo de uma determinada categoria, enquanto os outros comandos são utilizados para remover uma categoria e o conteúdo de uma categoria especifica.

Para facilitar o gerenciamento de categorias no servidor foi desenvolvida uma interface gráfica simples para adicionar, listar e remover as categorias. Na Figura 28 pode ser visualizada a tela de adição de categorias. Na imagem foi definido o nome da categoria, a descrição e o arquivo chamado ‘bancos.txt’ onde contém os sites bancários que devem ser adicionadas à categoria.

Figura 28 - Adição de categoria no F-Web Manager Server

Fonte: Elaborado pelo Autor, 2012.

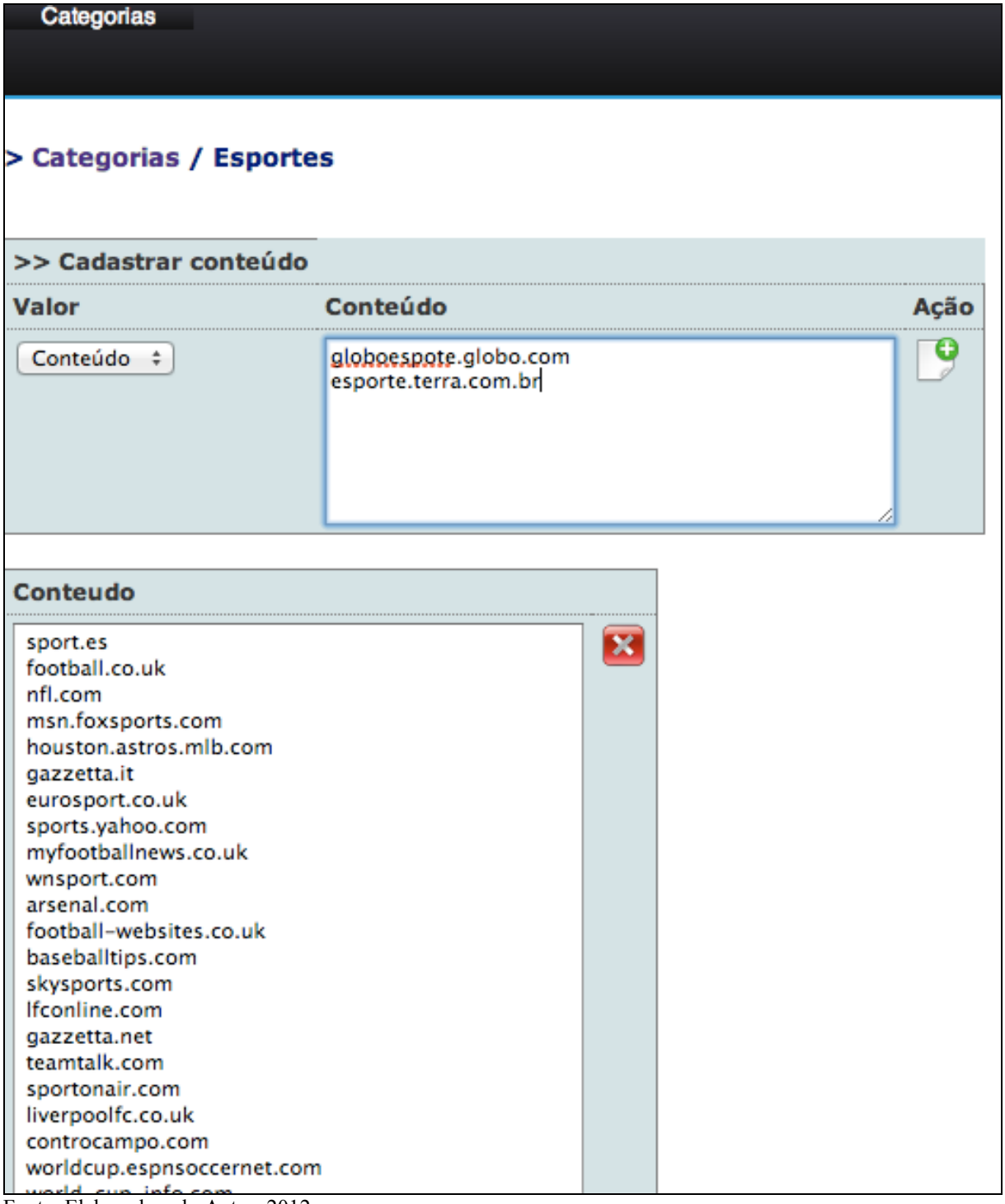
A listagem das categorias existentes pode ser visualizada na Figura 29. Na listagem das categorias existe um ícone que pode ser utilizado para visualizar o conteúdo da categoria e outro que pode ser utilizado para remover a categoria

Figura 29 - Listagem das categorias no F-Web Manager Server

Fonte: Elaborado pelo Autor, 2012.

Na Figura 30 pode ser visualizado o conteúdo de uma categoria no *F-Web Manager Server*. Este item da ferramenta pode ser utilizado também para adicionar conteúdo na categoria selecionada.

Figura 30 - Listagem de conteúdo em uma categoria do F-Web Manager Server



Fonte: Elaborado pelo Autor, 2012.

8.6 MÓDULOS DO F-WEB MANAGER

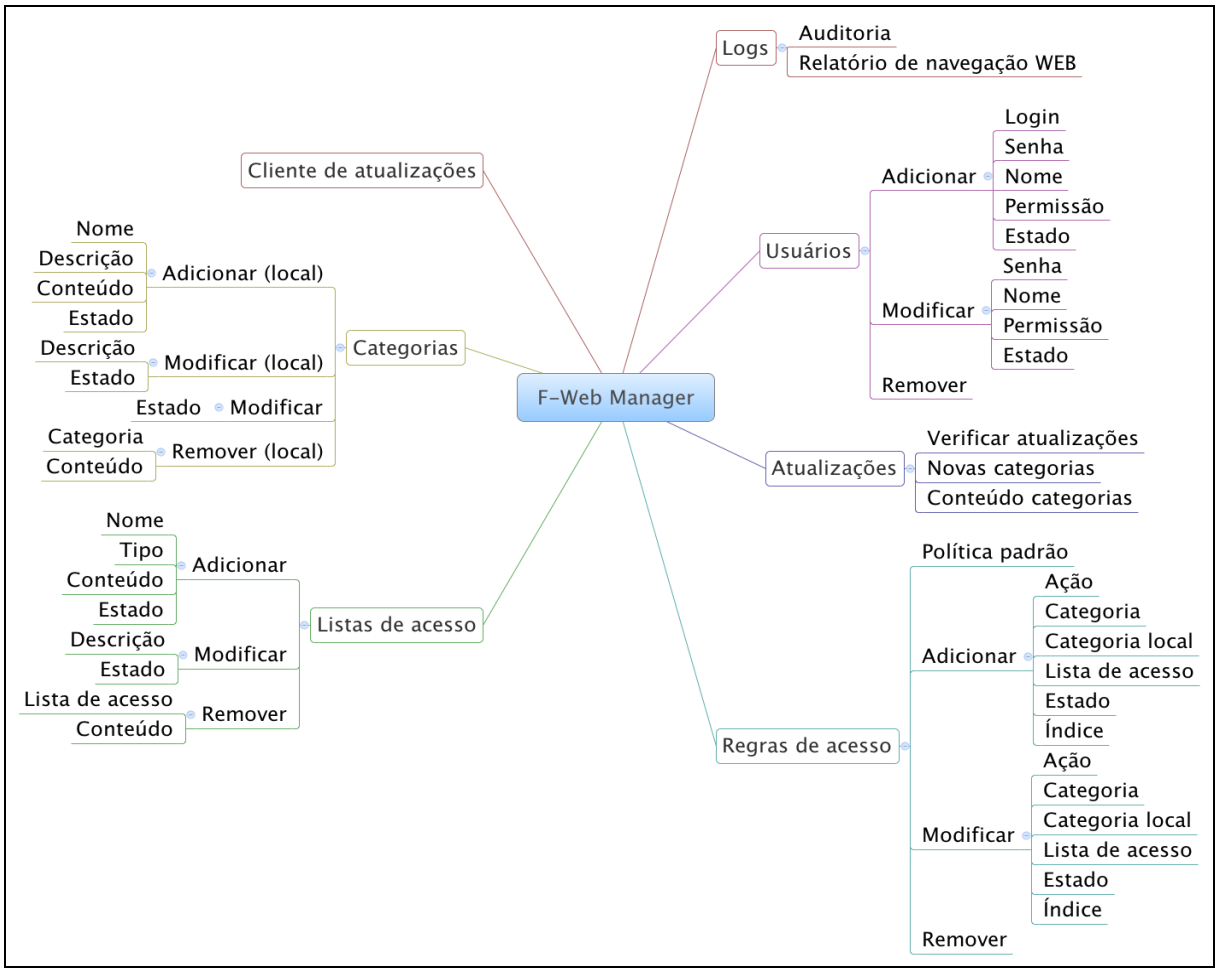
O *F-Web Manager* foi dividido em módulos para facilitar sua implementação e

futuras adições de novos recursos. Cada módulo possui uma função específica dentro da ferramenta, sendo em alguns casos independente dos outros módulos. Cada item do menu principal da interface gráfica do *F-Web Manager* representa um módulo diferente.

A Figura 31 mostra os módulos da ferramenta no formado de mapa conceitual, sendo cada um dos braços do mapa um módulo e também um item do menu da interface web de gerenciamento.

Os tópicos seguintes tratam desses módulos, descrevendo as funções de cada um e suas características.

Figura 31 - Mapa conceitual ilustrando a divisão por módulos do F-Web Manager



Fonte: Elaborado pelo Autor, 2012.

Para que o administrador possa visualizar os módulos e fazer os controles necessários, primeiramente é necessário autenticar na ferramenta com um usuário e senha válido. A ferramenta possui por padrão o usuário “admin” e senha “admin”.

Na figura 32 pode ser visualizada a tela de autenticação do *F-Web Manager*.

Figura 32 – Tela de autenticação do F-Web Manager

Fone: Elaborado pelo Autor, 2012.

8.6.1 Cliente de atualizações

Este *back-end* é responsável por verificar no servidor do *F-Web manager* se as categorias existentes precisam ter o seu conteúdo atualizado ou se novas categorias precisam ser adicionadas. Para o desenvolvimento desta ferramenta, foi utilizado a função 'socket' do *Python*, que permite a comunicação entre equipamentos utilizando uma arquitetura cliente/servidor.

No Quadro 55 podem ser visualizados os comandos disponíveis do cliente de atualizações.

Quadro 55 - Comandos disponíveis no cliente de atualizações

/fwebmanager/backend/cliente.py --acao=listar --opcao=categorias /fwebmanager/backend/cliente.py --acao=listar --opcao=categoria --categoria=Noticias /fwebmanager/backend/cliente.py --acao=atualizar --opcao=categorias /fwebmanager/backend/cliente.py --acao=atualizar --opcao=categoria --categoria=Noticias /fwebmanager/backend/cliente.py --acao=verificar --opcao=categoria --categoria=Noticias

Fonte: Elaborado pelo Autor, 2012.

Conforme ilustrado no Quadro anterior, o primeiro comando lista todas as categorias do servidor, enquanto o segundo comando lista o conteúdo da categoria Noticias do servidor do *F-Web Manager*. O terceiro comando atualiza a lista de categorias do cliente com o servidor, enquanto o quarto comando atualiza o conteúdo da categoria Noticias. O último comando apenas verifica se a categoria informada precisa ou não ser atualizada.

8.6.2 Categorias

Este módulo é responsável pelo gerenciamento das categorias locais e listagem das categorias atualizadas com o servidor. Através deste módulo é possível adicionar categorias locais que não tem nenhuma ligação com as categorias sincronizadas com o *F-Web Manager Server*, além de alterar, listar e remover estas categorias adicionadas previamente.

No Quadro 56 podem ser visualizados os comandos disponíveis para gerenciamento das categorias locais.

Quadro 56 - Comandos disponíveis para gerenciamento de categorias locais

```
/fwebmanager/backend/categorias.py --acao=listar --opcao=categorias
/fwebmanager/backend/categorias.py --acao=listar --opcao=categoria --categoria=id_categoria

/fwebmanager/backend/categorias.py --acao=adicionar --opcao=categoria --categoria=Noticias --conteudo=http://www.site1.com,http://www.site2.com --descricao='Categoria de noticias' --estado=1
/fwebmanager/backend/categorias.py --acao=adicionar --opcao=conteudo --categoria=id_categoria --conteudo=http://www.site3.com,http://www.site4.com
/fwebmanager/backend/categorias.py --acao=adicionar --opcao=arquivo --categoria=id_categoria --arquivo=/tmp/sites.txt

/fwebmanager/backend/categorias.py --acao=listar --opcao=categorias_locais
/fwebmanager/backend/categorias.py --acao=listar --opcao=categoria_local --categoria=id_categoria

/fwebmanager/backend/categorias.py --acao=remover --opcao=categoria --categoria=id_categoria
/fwebmanager/backend/categorias.py --acao=remover --opcao=conteudo --categoria=id_categoria --conteudo=id1_conteudo,id2_conteudo

/fwebmanager/backend/categorias.py --acao=editar --opcao=categoria_local --categoria=id_categoria --descricao='Nova descricao' --estado=0
```

Fonte: Elaborado pelo Autor.

No exemplo ilustrado no quadro anterior pode ser visto vários comandos disponíveis para gerenciamento das categorias. Os dois primeiros comandos são utilizados apenas para visualizar as categorias que foram atualizadas com o servidor e o conteúdo das mesmas.

O terceiro, quarto e quinto comando são utilizados para adicionar categorias locais, que não tem nenhuma ligação com as categorias atualizadas com o servidor através do cliente de atualizações. Em seguida é mostrado os comandos utilizados para listar as categorias locais e o conteúdo das mesmas, seguido dos comandos utilizados para remover as categorias locais e o conteúdo das mesmas. O último comando é utilizado apenas para alterar o estado e/ou descrição de uma categoria local.

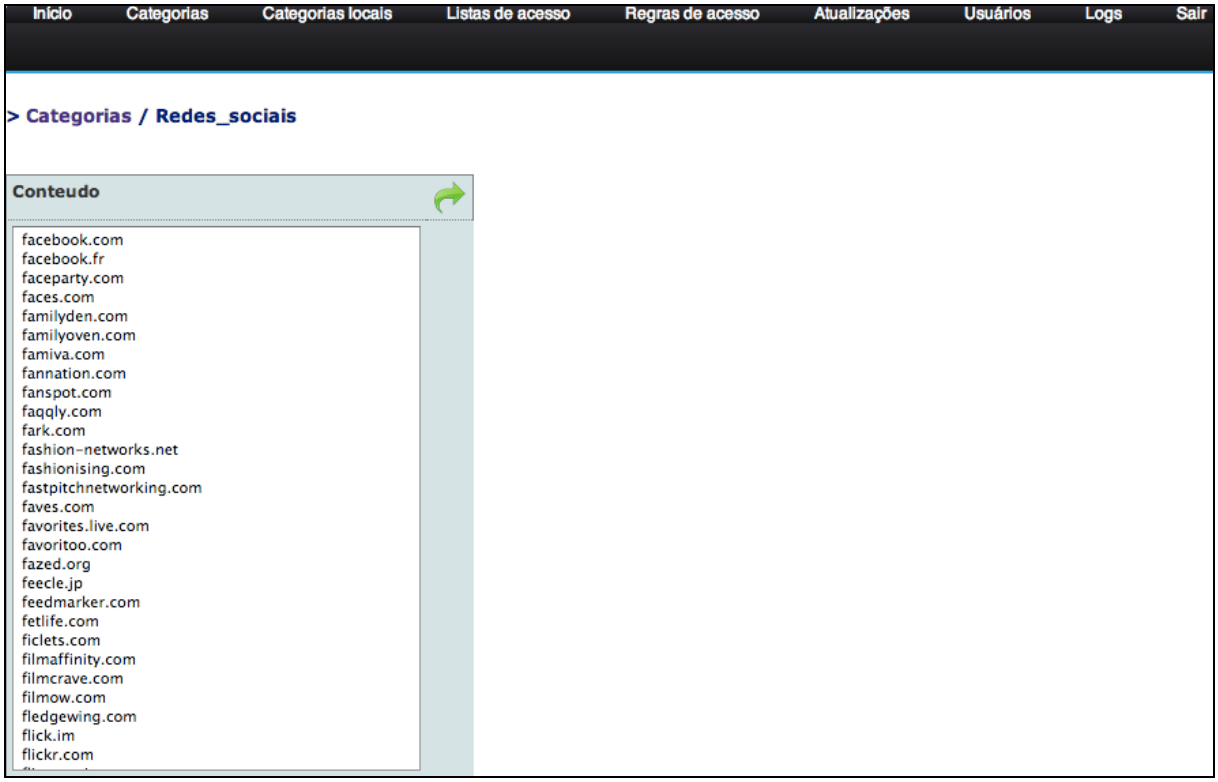
A interface de administração *Web* para gerenciamento deste módulo pode ser visto a seguir. Na Figura 33 podem ser visualizadas as categorias que foram sincronizadas com o servidor do *F-Web Manager*, onde é possível desativar categorias, atualizar as categorias que não estão sincronizadas com o servidor, visualizar o conteúdo das categorias e também remover as categorias que não existem mais no servidor.

Figura 33 - Listagem de categorias no F-Web Manager

Fonte: Elaborado pelo Autor, 2012.

Ao clicar no ícone da lupa ilustrada na Figura anterior, é apresentado o conteúdo da categoria conforme mostra a Figura 34. Nesta tela não é possível remover o conteúdo da categoria, pois a mesma é sincronizada com o servidor de atualizações.

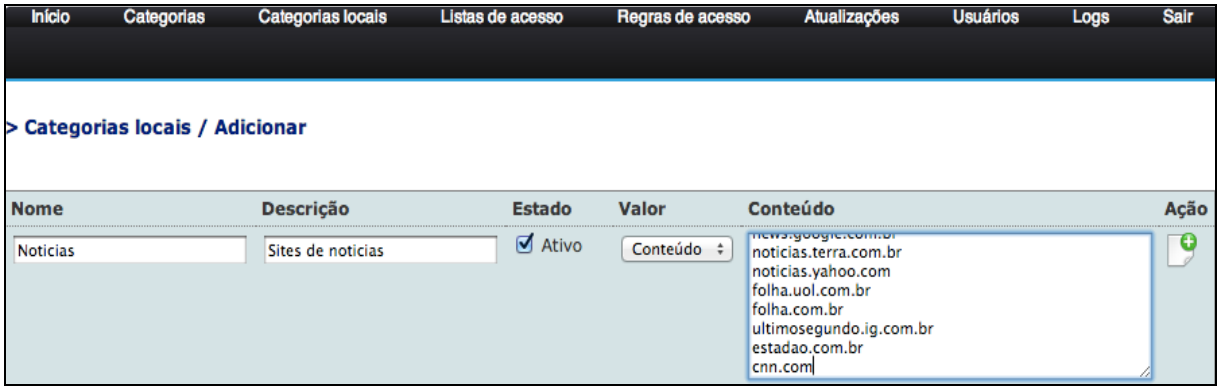
Figura 34 - Conteúdo de uma categoria no F-Web Manager



Fonte: Elaborado pelo Autor, 2012.

Além das categorias que são sincronizadas com o servidor, é possível também cadastrar categorias locais que não tem nenhuma relação com as categorias sincronizadas com o *F-Web Manager Server*. Na Figura 35 pode ser visualizado o cadastro de uma categoria local.

Figura 35 - Cadastro de categoria local no F-Web Manager



Fonte: Elaborado pelo Autor, 2012.

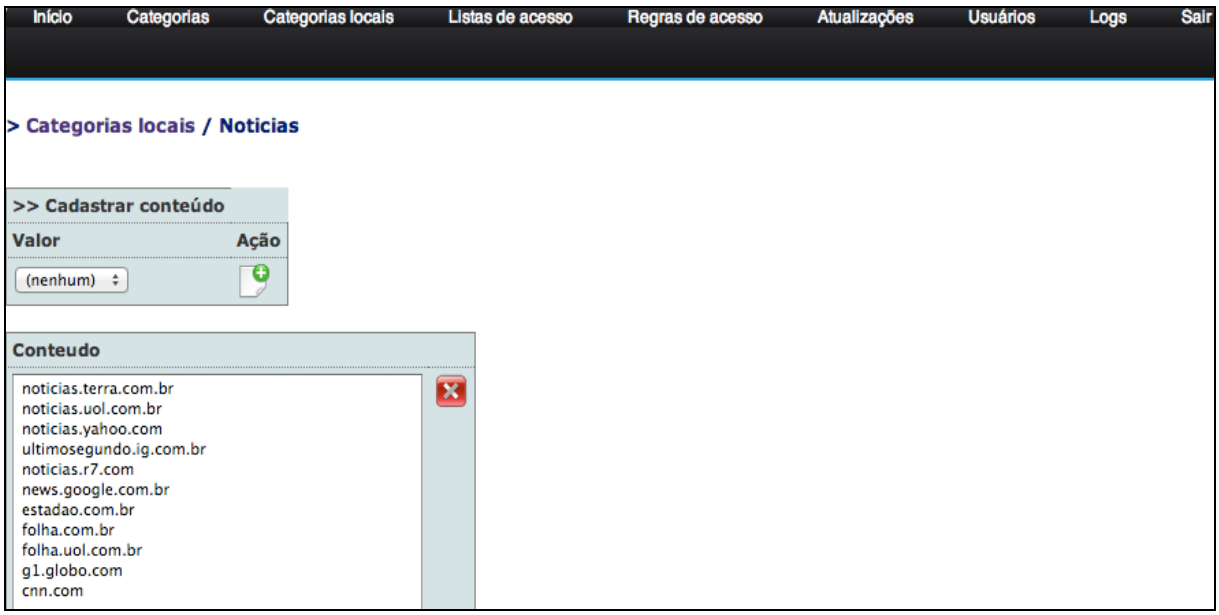
A listagem das categorias locais pode ser visualizada na Figura 36. Nesta tela pode ser alterada a descrição e o estado, visualizado o conteúdo e removido a categoria.

Figura 36 - Listagem de categorias locais no F-Web Manager

Fonte: Elaborado pelo Autor, 2012.

Na figura 37 é mostrado o conteúdo de uma categoria local que pode ser visualizada após ter clicado no ícone da lupa da categoria local 'Noticias'.

Figura 37 - Listagem do conteúdo de uma categoria local no F-Web Manager



Fonte: Elaborado pelo Autor, 2012.

8.6.3 Listas de acesso

Este módulo é utilizado para gerenciar as listas de acesso do *F-Web Manager*. O tipo das listas de acesso suportadas são ip_origem e porta_destino. As listas de acesso são utilizadas para ter um maior controle sobre as regras de acesso criadas, podendo desta forma, liberar determinadas categorias apenas para alguns endereços IPs e/ou portas de destino.

No Quadro 57 encontram-se os comandos que podem ser utilizados para gerenciamento das listas de acesso.

Quadro 57 - Comandos disponíveis para gerenciamento das listas de acesso

```
/fwebmanager/backend/listas_de_acesso.py --acao=adicionar --opcao=lista_de_acesso --lista_de_acesso=Portas --conteudo=80,443 --descricao='Portas liberadas' --estado=0 --tipo=porta_destino
/fwebmanager/backend/listas_de_acesso.py --acao=adicionar --opcao=lista_de_acesso --lista_de_acesso=Equipamentos --arquivo=/tmp/equipamentos.txt --descricao='Lista de equipamentos' --estado=1 --tipo=ip_origem

/fwebmanager/backend/listas_de_acesso.py --acao=listar --opcao=listas_de_acesso
/fwebmanager/backend/listas_de_acesso.py --acao=listar --opcao=lista_de_acesso --lista_de_acesso=id_lista_de_acesso

/fwebmanager/backend/listas_de_acesso.py --acao=remover --opcao=lista_de_acesso --lista_de_acesso=id_lista_de_acesso
/fwebmanager/backend/listas_de_acesso.py --acao=remover --opcao=conteudo --lista_de_acesso=id_lista_de_acesso --conteudo=id1_conteudo,id2_conteudo

/fwebmanager/backend/listas_de_acesso.py --acao=editar --opcao=lista_de_acesso --lista_de_acesso=id_lista_de_acesso --descricao='Nova descricao' --estado=0
```

Fonte: Elaborado pelo Autor, 2012.

Este *back-end* segue a mesma lógica dos outros *back-ends* descritos anteriormente. Os primeiros comandos ilustram a adição de uma lista de acesso, em seguida é mostrada a listagem das listas de acesso e do conteúdo das mesmas. Nos comandos seguintes é mostrado como remover e editar as listas de acesso existentes.

Nas imagens a seguir serão mostradas as telas da interface de gerenciamento Web do módulo ‘Listas de acesso’.

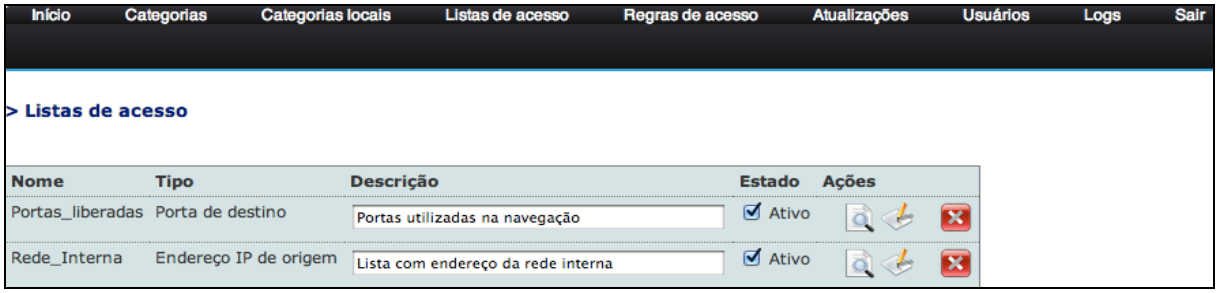
Figura 38 - Cadastro de uma lista de acesso no F-Web Manager

Fonte: Elaborado pelo Autor, 2012.

Na Figura 38 é mostrado o cadastro de uma lista de acesso. No exemplo ilustrado foi utilizado o tipo ‘Endereço IP de origem’, porém pode ser utilizado também o tipo ‘Porta de destino’. Estas listas de acesso poderão ser utilizadas em conjunto com uma categoria ou categoria local para montar uma regra de acesso.

As listas de acesso existentes podem ser visualizadas na Figura 39, onde é possível alterar a descrição e o estado das mesmas, visualizar o conteúdo e remover uma lista.

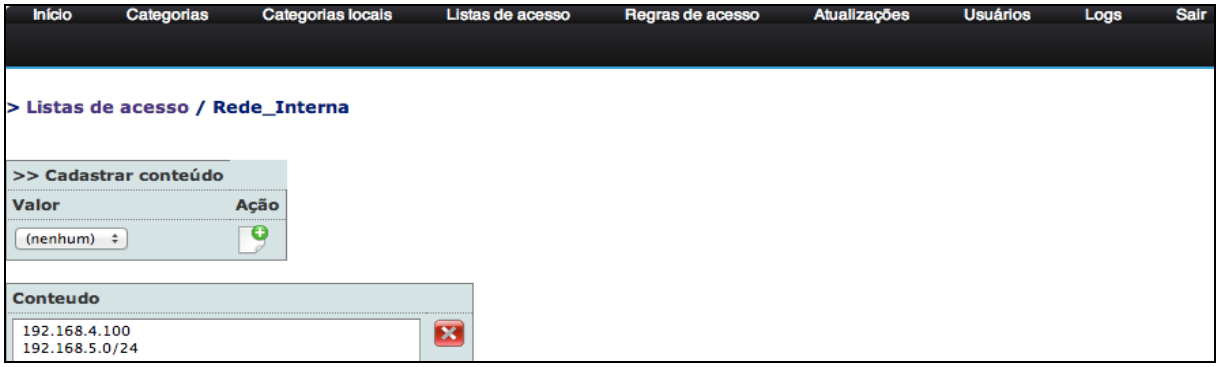
Figura 39 - Listagem das listas de acesso no F-Web Manager



Fonte: Elaborado pelo Autor, 2012.

Na Figura 40 é mostrado o conteúdo da lista de acesso ‘Rede_Interna’ que pode ser visualizada ao clicar na lupa da lista de acesso selecionada. Nesta tela é possível adicionar conteúdo a partir de um arquivo ou especificando o que deve ser adicionado, além de poder remover o conteúdo existente.

Figura 40 - Listagem do conteúdo de uma lista de acesso no F-Web Manager



Fonte: Elaborado pelo Autor, 2012.

8.6.4 Regras de acesso

Este módulo é responsável por gerenciar as regras de acessos do *F-Web Manager*.

As categorias e as listas de acessos por si só não fazem nenhum sentido. Na prática, as categorias e listas de acesso só terão sentido quando estiverem sendo utilizadas nas regras de acesso, onde é feito as definições de quem pode fazer os acessos e para onde estes acessos podem ser feitos.

Os comandos disponíveis para gerenciamento das regras de acesso do *F-Web Manager* são encontrados no Quadro 58.

Quadro 58 - Comandos disponíveis para gerenciamento das regras de acesso

/fwebmanager/backend/regras_de_acesso.py --acao=adicionar --lista_de_acesso=id_lista --categoria=id_categoria --politica=0 --indice=1 --estado=1
/fwebmanager/backend/regras_de_acesso.py --acao=listar
/fwebmanager/backend/regras_de_acesso.py --acao=listar --lista_de_acesso=_fwebmanager_policy
/fwebmanager/backend/regras_de_acesso.py --acao=remover --regra_de_acesso=id_regra_de_acesso
/fwebmanager/backend/regras_de_acesso.py --acao=editar --lista_de_acesso=id_lista --categoria=id_categoria --politica=0 --indice=1 --estado=0 --regra_de_acesso=id_regra_de_acesso
/fwebmanager/backend/regras_de_acesso.py --acao=editar --lista_de_acesso=_fwebmanager_policy --politica=1

Fonte: Elaborado pelo Autor, 2012.

Conforme se percebe no exemplo, o primeiro comando adiciona uma regra de acesso utilizando uma lista de acesso e categoria já existente e definida à política da regra como aceitar. Os próximos comandos são utilizados para listar as regras já existentes e a política padrão atuais.

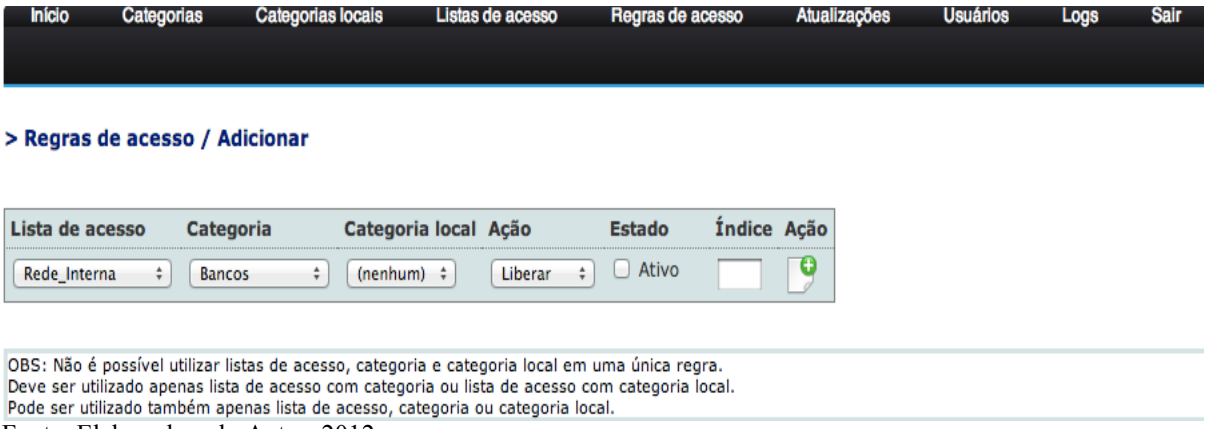
Em seguida é utilizado um comando para editar uma regra de acesso existente, apenas alterando o estado da mesma. O último comando é utilizado para alterar a política padrão para bloquear. Esta política é utilizada para qualquer acesso feito por equipamentos que não batem com nenhuma regra de acesso existente.

Na Figura 41 pode ser visualizada a adição de uma regra de acesso onde pode ser

definida uma lista de acesso e uma categoria ou categoria local; uma ação que pode ser Liberar ou Bloquear; o estado que pode ser Ativo ou não e o índice que é opcional.

Caso o índice seja informado, a regra será colocada na ordem informada e caso já exista uma regra de acesso no índice informado, elas serão automaticamente ajustadas. Se o índice não for especificado, então a regra será colocada no final das regras já existentes.

Figura 41 - Cadastro de uma regra de acesso no F-Web Manager



Fonte: Elaborado pelo Autor, 2012.

Na Figura 42 pode ser visualizada a lista das regras de acesso existentes, onde é possível alterar a lista de acesso e a categoria ou categoria local, a ação, o estado e o índice da regra, além de poder remover a mesma.

Figura 42 - Listagem das regras de acesso no F-Web Manager

Fonte: Elaborado pelo Autor, 2012.

8.6.5 Atualizações

Este módulo é responsável por verificar se todas as categorias estão atualizadas com o servidor e permitir a atualização destas categorias caso seja necessário.

No quadro a seguir, podem ser encontrados os comandos disponíveis para gerenciamento das atualizações.

Quadro 59 - Comandos disponíveis para gerenciamento de atualizações

/fwebmanager/backend/atualizacoes.py --acao=atualizar
/fwebmanager/backend/atualizacoes.py --acao=listar
/fwebmanager/backend/atualizacoes.py --acao=verificar

Fonte: Elaborado pelo Autor, 2012.

O primeiro comando citado no exemplo é utilizado para atualizar todas as categorias existentes e também para adicionar novas categorias que existem no servidor, porém não existem no cliente. O segundo comando apenas mostra o que precisa ser atualizado, enquanto o último adiciona no banco a informação do que precisa ser atualizado.

Na figura a seguir pode ser visualizada a tela de administração do módulo de Atualizações do *F-Web Manager* onde é possível verificar as categorias que existem no servidor, mas que ainda não existem no cliente e atualizar se desejar. É possível também visualizar as categorias que existem no *F-Web Manager* e no servidor de atualizações, porém, com o conteúdo da categoria desatualizada, onde é possível também atualizar estas categorias.

Figura 43 - Listagem de atualizações no F-Web Manager

Fonte: Elaborado pelo Autor, 2012.

8.6.6 Usuários

Este módulo é responsável pelo gerenciamento dos usuários utilizados pelo *front-end* do *F-Web Manager*. Através deste *back-end* é possível adicionar novos usuários, além de editar, listar e remover os usuários já existentes.

No Quadro 60 podem ser visualizados os comandos disponíveis para gerenciamento dos usuários.

Quadro 60 - Comandos disponíveis para gerenciamento de usuários

```
/fwebmanager/backend/usuarios.py --acao=adicionar --usuario=andre --senha=unisul2012 --nome=Andre --perml_categorias=1 --perml_listas_de_acesso=1 --perml_regras_de_acesso=1
--perml_atualizacoes=1 --perml_usuarios=1 --perml_logs=1 --permg_categorias=1 --permg_listas_de_acesso=1 --permg_regras_de_acesso=1 --permg_atualizacoes=1 --permg_usuarios=1 --permg_logs=1 --estado=1

/fwebmanager/backend/usuarios.py --acao=listar

/fwebmanager/backend/usuarios.py --acao=remover --usuario=id_usuario

/fwebmanager/backend/usuarios.py --acao=editar --usuario=id_usuario --senha=unisul2012 --nome='Andre S. Almeida' --perml_categorias=1 --perml_listas_de_acesso=1 --perml_regras_de_acesso=1 --perml_atualizacoes=1 --perml_usuarios=1 --perml_logs=1 --permg_categorias=1 --permg_listas_de_acesso=1 --permg_regras_de_acesso=1 --permg_atualizacoes=1 --permg_usuarios=1 --permg_logs=0 --estado=1

/fwebmanager/backend/usuarios.py --acao=verifica --usuario=andre --senha=unisul2012
```

Fonte: Elaborado pelo Autor, 2012.

Conforme a ilustração, o primeiro comando adiciona o usuário definindo as permissões do mesmo. O segundo apenas lista os usuários existentes, enquanto o terceiro remove e o quarto edita um usuário existente. Por último é utilizado um comando para validar um usuário, neste caso o *backend* retorna ‘1’ se o usuário e senha forem válidos ou ‘0’ se a informação não combinar.

As telas de gerenciamento dos usuários do *F-Web Manager* podem ser visualizadas nas próximas Figuras. Na Figura 44 pode ser visualizada a tela de adição de um usuário. Nesta tela pode ser definido o nome do usuário, a senha, o nome de exibição do usuário, o estado e os módulos de leitura e gravação do usuário em questão.

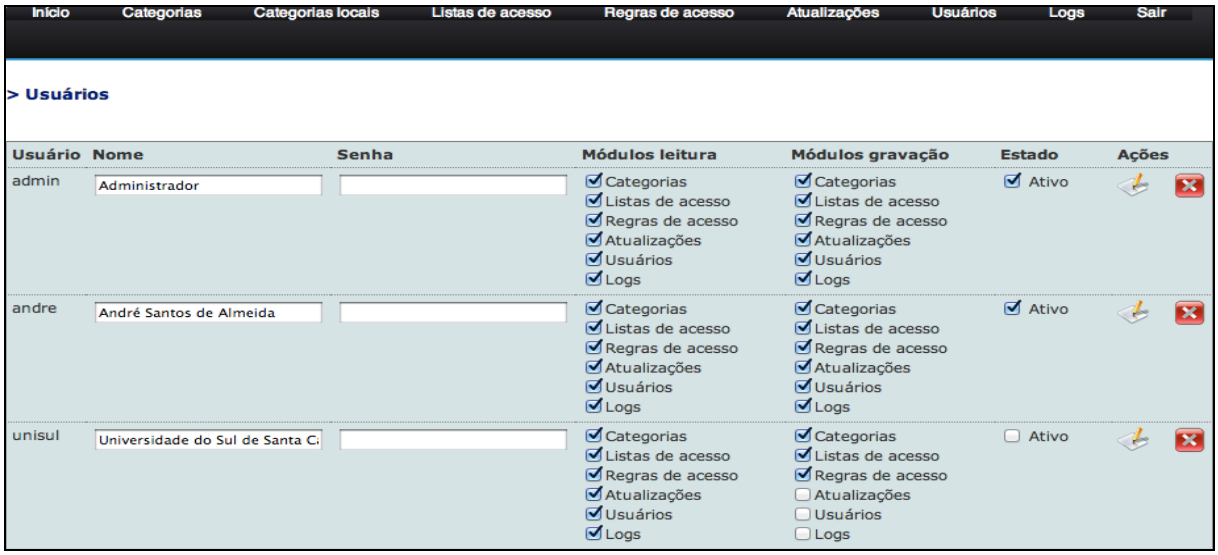
No campo 'módulos leitura' podem ser definidos os módulos que o usuário poderá acessar para visualizar o conteúdo. Caso o usuário que esteja logado no sistema não tiver permissão para acessar um determinado módulo, será exibida uma mensagem de notificação informando que o mesmo não tem acesso ao módulo e o conteúdo do módulo não será exibido. No campo 'módulos gravação' podem ser definidos os módulos onde o usuário poderá fazer cadastros; alterações e remoções dos itens cadastrados, além das atualizações.

Figura 44 - Adição de um usuário no F-Web Manager

Fonte: Elaborado pelo Autor, 2012.

Na Figura 45 pode ser visualizada a lista dos usuários existentes, onde é possível alterar o nome de exibição; a senha; as permissões; o estado do usuário, além de removê-lo.

Figura 45 - Listagem dos usuário do F-Web Manager



Fonte: Elaborado pelo Autor, 2012.

8.6.7 Logs

Este módulo tem como objetivo registrar no banco de dados as operações feitas através da interface de gerenciamento do *F-Web Manager*, e armazenar no banco de dados as informações dos sites acessados.

No quadro a seguir pode ser visualizado o comando que deve ser utilizado pelo *front-end* para logar os registros das operações.

Quadro 61 - Comando do módulo logs para registros as operações do F-Web Manager

```
/fwebmanager/backend/logs.py --datahora=1234567890 --usuario=andre --modulo=categoria --acao='Adicionou categoria' --informacoes='Nome da categoria: Noticias' --cod_operacao=0
```

Fonte: Elaborado pelo Autor, 2012.

O comando demonstrado no quadro acima mostra com deve ser registrada uma operação.

No exemplo citado, é informado o horário da operação no formato de *timestamp* que nada mais é do que a data, iniciado em 01 de janeiro de 1970 até o dia e hora atual, informada em segundos.

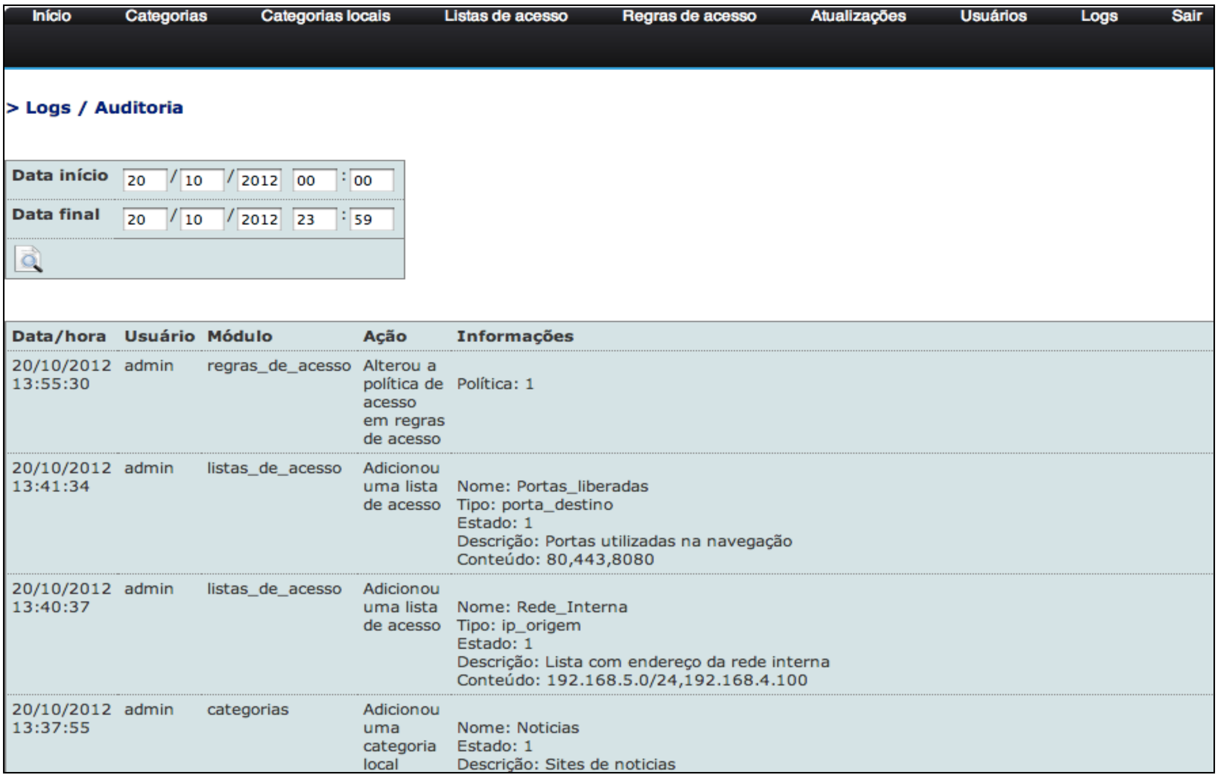
Nos outros parâmetros do comando é informado o usuário, módulo, ação, informações a adicionar e o código da operação. O código sempre será ‘0’ se a operação for executada com sucesso ou ‘1’ se algum erro for encontrado.

Todas as alterações feitas através da interface de gerenciamento *Web* são registradas no banco de dados e podem ser visualizadas pelo Administrador.

Desse modo, na Figura 46 visualiza-se a auditoria da ferramenta, onde é possível verificar a data e hora que foi feita a operação; o usuário que fez a operação; o módulo onde foi operado; a ação que foi tomada e as informações adicionais, onde podem ser visualizados os valores informados pelo usuário.

No módulo chamado ‘Logs’ pode ser visualizado também o registro dos sites acessados pelos equipamentos que estão sendo gerenciados pelo *F-Web Manager*, onde o administrador pode verificar o que os gerenciados estão acessando e se o acesso feito está sendo liberado ou bloqueado.

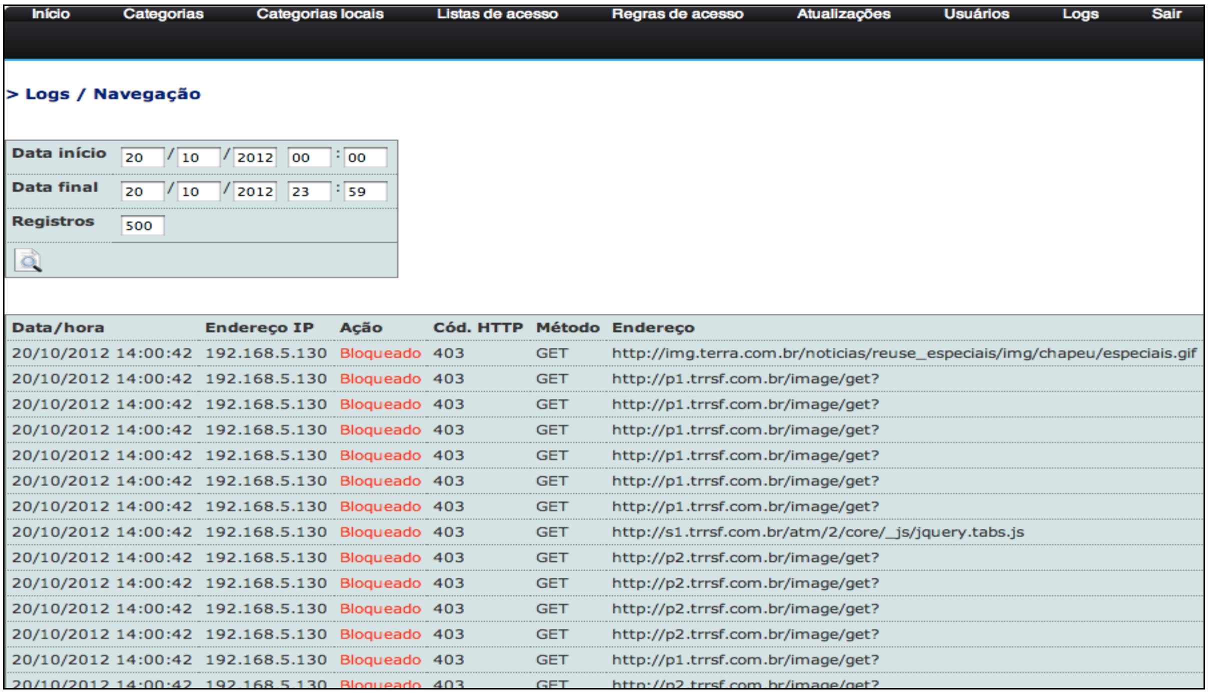
Figura 46 - Auditoria do F-Web Manager



Fonte: Elaborado pelo Autor, 2012.

Na Figura 47 encontra-se o relatório de navegação no *F-Web Manager*, conforme consta:

Figura 47 - Relatório de navegação no F-Web Manager



Fonte: Elaborado pelo Autor, 2012.

De acordo com a ilustração proposta, verifica-se a tela onde é possível visualizar os sites acessados; a data e hora que o acesso foi feito; o endereço IP do equipamento que fez o acesso; a ação que foi tomada onde é informado se o acesso foi aceito ou não; o código HTTP e o método utilizado para a conexão com o site acessado.

Por padrão é apresentado 500 registros e apenas do dia atual. Se o administrador quiser mais informações dos sites acessados, basta especificar a data, hora e o número de registros desejado.

9 ANÁLISE DOS RESULTADOS

Com o intuito de validar o funcionamento e a importância do *F-Web Manager* para as empresas de modo geral, foi criado um ambiente virtual onde foi possível simular uma rede corporativa real.

Para compor o ambiente virtual, foi necessária a criação de três máquinas virtuais: uma onde foi desenvolvida o *F-Web Manager Server*, outra onde foi desenvolvido o *F-Web Manager* e uma terceira máquina virtual que foi utilizada para simular um equipamento que utilizaria o *F-Web Manager* para navegar na Web.

Conforme ilustra todas as imagens do capítulo 8, no *F-Web Manager* foram atualizadas as categorias com o servidor de atualizações e também criadas algumas categorias locais, listas de acesso e regras de acesso.

Para efeito de testes, foram criadas três regras de acesso e a definição de cada regra de acesso pode ser visto abaixo:

a) a primeira regra libera o acesso aos sites que estão cadastrados na categoria local “Notícias”, caso a porta de destino utilizada no acesso também esteja cadastrada na lista de acesso “Portas_Liberadas”;
b) a segunda regra bloqueia o acesso aos sites que estão cadastrados na categoria “Redes_Sociais”, caso o endereço IP de origem esteja cadastrado ou faça parte das redes cadastradas na lista de acesso “Rede_Interna”;
c) A terceira regra libera o acesso aos sites que estão cadastrados na categoria “Bancos”, caso o endereço IP de origem esteja cadastrado ou faça parte das redes cadastradas na lista de acesso “Rede_Interna”, no entanto esta regra está desativada.

Além das regras de acesso, foi definida também a política padrão como ‘Bloquear’, ou seja, se a conexão solicitada não tiver sido explicitamente liberada e/ou bloqueada, então o acesso será bloqueado pela política padrão.

Para homologar[1] o ambiente, foi utilizado uma máquina virtual para fazer os acessos aos sites. No ambiente de testes, o *F-Web Manager* ficou configurado com o endereço IP 192.168.5.4, e a máquina virtual de testes com o endereço IP 192.168.5.130. Foi necessário

[1] O termo homologar se refere ao processo de avaliação da ferramenta, cujo objetivo é verificar se o mesmo está funcionando corretamente.

também definir o endereço IP do *F-Web Manager* na configuração do navegador da máquina virtual de teste. O navegador utilizado para testes foi o Firefox 16.0.1.

Para validar a primeira regra de acesso, foi feito um acesso ao site "g1.globo.com" que está cadastrado na categoria local "Notícias". Como pode ser visualizado na Figura 48, o acesso foi aceito e o site foi visualizado conforme definido na primeira regra.

Figura 48 - Teste de acesso ao site G1

Fonte: Elaborado pelo Autor, 2012.

A confirmação do acesso pode ser visto no módulo de *logs* do F-Web Manager conforme mostra a Figura 49. Nos *logs* foram apresentados alguns endereços que não foram definidos no browser, porém estes endereços são acessados automaticamente pelo browser na busca por imagens e outros objetos conforme definição do desenvolvedor do site em questão.

Figura 49 - Log de acesso ao site G1

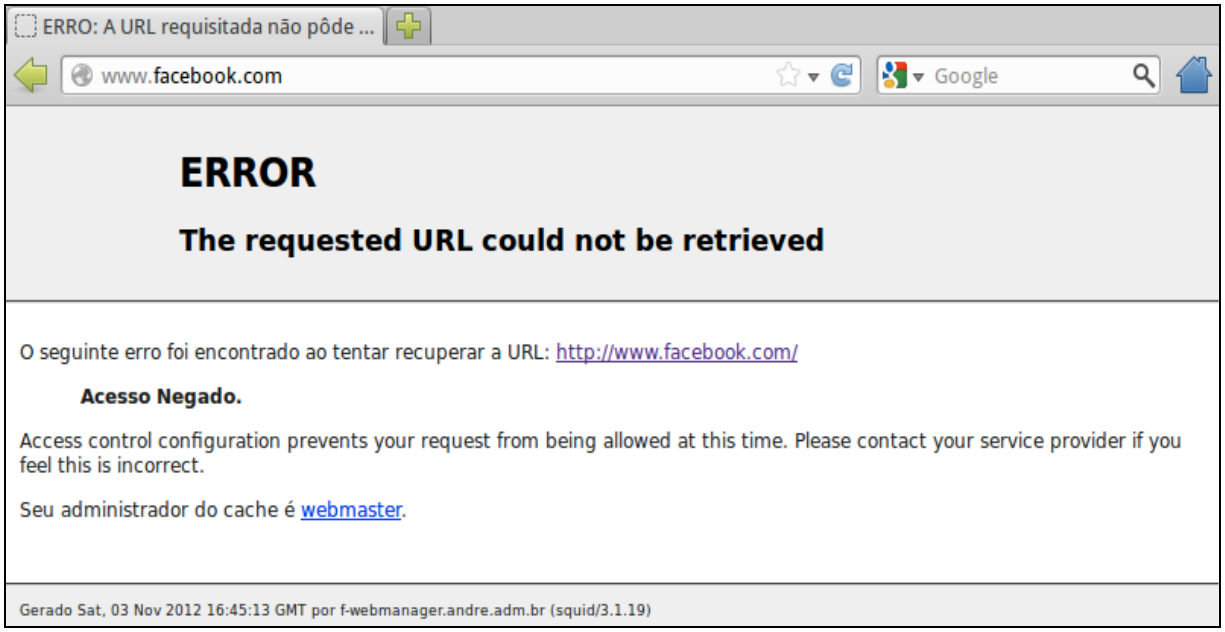
03/11/2012 14:41:59	192.168.5.130	Aceito	200	GET	http://g1.globo.com/
03/11/2012 14:41:20	192.168.5.130	Aceito	200	GET	http://b.scorecardresearch.com/p?
03/11/2012 14:41:19	192.168.5.130	Aceito	304	GET	http://s.glbimg.com/jo/g1/static/portal/img/desktop/componentes/page/portalg1/cabecalho/bg-menu-rodape-g1.gif?
03/11/2012 14:41:19	192.168.5.130	Aceito	304	GET	http://s.glbimg.com/jo/g1/static/header_g1/img/sprite_menu_item.png
03/11/2012 14:41:19	192.168.5.130	Aceito	304	GET	http://s.glbimg.com/jo/g1/static/common/img/menu/bkg_item_menu.png?
03/11/2012 14:41:19	192.168.5.130	Aceito	304	GET	http://s.glbimg.com/jo/g1/static/header_g1/img/bg_header.png

Fonte: Elaborado pelo Autor, 2012.

Para validar a segunda regra, foi realizado o acesso ao site "www.facebook.com"

que está cadastrado na categoria “Redes_Sociais”. Como pode ser visto na Figura 50, o acesso foi bloqueado e o site não pôde ser visualizado conforme definido na regra de acesso.

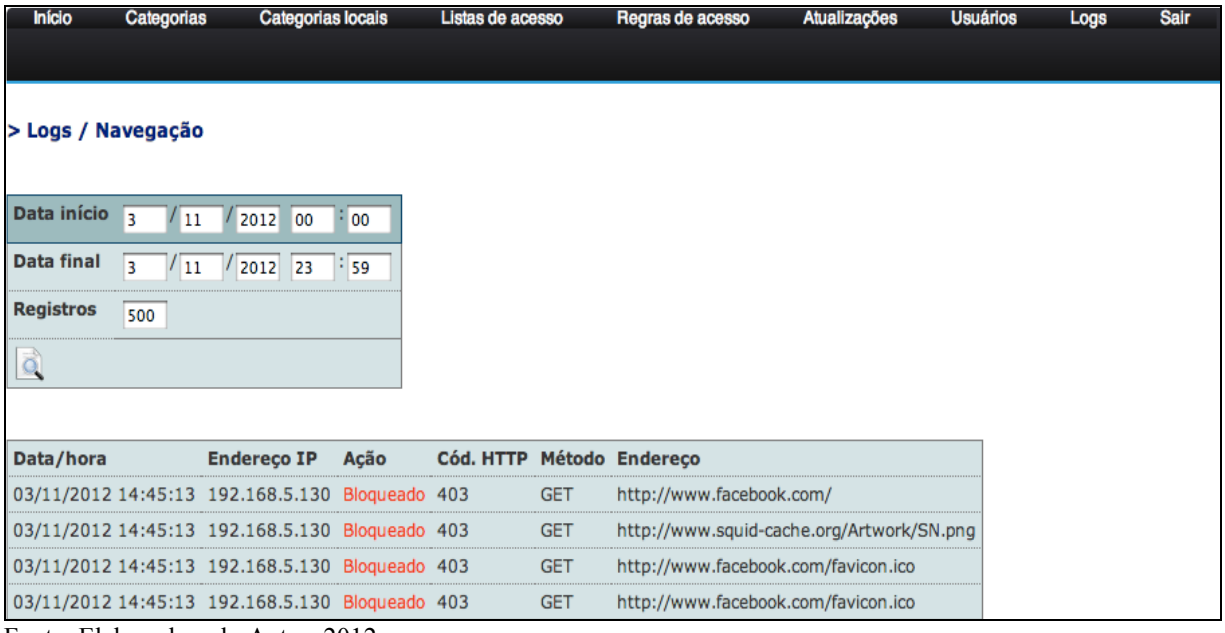
Figura 50 - Teste de acesso ao site Facebook



Fonte: Elaborado pelo Autor, 2012.

A confirmação do acesso bloqueado pode ser visto no módulo de *logs* conforme mostra a Figura 51.

Figura 51 - Log de acesso ao site Facebook



Fonte: Elaborado pelo Autor, 2012.

Conforme pode ser visto, a ferramenta se mostrou eficaz no seu propósito de prover uma interface *Web* em que o administrador possa gerenciar as categorias, listas de acesso e regras de acesso de forma simples e eficiente. Além de facilitar o gerenciamento através da interface *Web*, o *F-Web Manager* se mostrou eficaz no controle dos acessos realizados, visto que o acesso foi bloqueado quando deveria, e liberado quando era permitido.

10 CONCLUSÃO

Devido ao aumento da utilização de computadores, sistemas e principalmente da internet, as empresas estão cada dia mais dependentes destas tecnologias. Por esses motivos está mais comum o investimento em segurança e disponibilidade em redes de computadores.

Através da revisão bibliográfica ficou claro que firewalls de aplicação são ativos freqüentemente utilizados em infra-estruturas de rede, com o objetivo de controlar o tráfego que passa através do ambiente interno e externo, geralmente representado pela Internet. Vimos que um *Proxy Web* é um *Firewall* de aplicação muito importante, pois através dele é possível ter um controle mais detalhado sobre o tráfego gerenciado. No entanto os *proxies Web* existentes não são de fácil entendimento e o Administrador precisa ter certo conhecimento para conseguir gerenciar uma rede corporativa com estas ferramentas.

Pensando nas dificuldades de configuração e de encontrar ferramentas no mercado que gerenciem o tráfego *Web*, foi desenvolvido o *F-Web Manager*, ferramenta que permite ao administrador uma série de controle, tirando o máximo de proveito de um *Proxy Web*.

Desta forma surgiu o *F-Web Manager*, representado neste trabalho através da modelagem de um *Proxy Web*. O objetivo desta ferramenta é demonstrar as informações da forma mais clara possível e permitir ao administrador gerenciar o tráfego *Web* da rede corporativa. Para isto foi criado uma interface em PHP, onde o administrador pode trabalhar com categorias de sites e listas de acesso, além de poder criar regras de acesso baseado nas categorias e/ou listas de acesso que podem ser criados e/ou customizados pelo Administrador.

A ferramenta também permite a atualização de categorias e o conteúdo destas categorias com um servidor de atualizações, onde poderia ser mantida uma base de categorias pré-definidas com os sites relacionados. Além disso, o administrador pode definir os usuários que deverão ter acesso à ferramenta, além de definir os módulos que estes usuários podem utilizar, tendo desta forma maior controle no uso do *F-Web Manager*.

Quando a ferramenta foi colocada em produção para homologação, foi visualizada uma série de sites acessados pelo gerenciado na qual um administrador não teria conhecimento sem um *Proxy Web*, pois os sites estavam sendo acessados através de softwares instalados no equipamento, abusando de certa forma, da utilização da Internet.

O benefício direto na utilização do *F-Web manager* é a facilidade que o Administrador teria para visualizar e gerenciar a rede corporativa, diminuindo assim o tempo de resposta do Administrador para as operações necessárias.

10.1 DIFICULDADES ENCONTRADAS

Entre as dificuldades encontradas durante a realização deste trabalho, talvez a mais significativa tenha sido o fato do autor trabalhar 40 horas por semana, onde em muitas vezes foi necessário trabalhar também aos finais de semana. Desta forma, administrar a vida profissional agitada e conciliar com a vida acadêmica acabam ocupando a maior parte da semana, sobrando muito pouco tempo para as atividades com o trabalho de conclusão.

Quanto à implementação propriamente dita, a maior dificuldade deu-se pela falta de experiência do autor na linguagem de programação *Python* e principalmente em modelagem de software, tomando muito tempo para projeto e em alguns momentos exigindo mudanças em relação ao que tinha sido inicialmente planejado.

10.2 SUGESTÕES PARA TRABALHOS FUTUROS

Em função de o projeto propor o desenvolvimento de uma ferramenta, sempre existirá a possibilidade de realização de melhorias, tanto no aspecto visual quanto na usabilidade, e acréscimo de novos recursos que podem agregar às funções já existentes ou incorporar funções inteiramente novas à ferramenta.

Durante o desenvolvimento da ferramenta, vislumbrando a continuidade dos trabalhos com o *F-Web Manager*, foram registrados um conjunto de melhorias e novas funcionalidades, dentro as quais se podem destacar:

a) Melhorias do visual e da usabilidade da interface gráfica;

b) Suporte a controle por autenticação de usuários;

c) Suporte a controle de banda;

d) Homologar a ferramenta para outros sistemas operacionais como *OpenBSD* e *GNU/Linux*.

REFERÊNCIAS

ALLEN, Grant; OWENS, Mike. **The Definitive Guide to SQLite, Second Edition**. Springer Science: New York, 2010.

BUILTWITH. **BuiltWith Technology Lookup**: Trends, Optimization and Lead Generation. Disponível em: <http://trends.builtwith.com/Web-Server/lighttpd>. Acesso em 02 jun. 2012.

CAMPOS, André L. N. **Sistema de Segurança da Informação**: Controlando os Riscos. Editora Visual Books Ltda: Florianópolis, 2006.

CETIC.BR. **Centro de Estudos sobre as Tecnologias da Informação e da Comunicação**: Disponível em <http://cetic.br/>. Acessoem: 17 mar. 2012.

CHESWICK, William R.; BELLOVIN, Steven M.; RUBIN, Aviel D. **Firewalls and Internet Security:** Repelling the Wily Hacker. Addison-Wesley: Boston, 2003.

CARUSO, Carlos, A. A; STEFFEN, Flávio Deny. **Segurança em informática e de informações, 3ª Edição**. Editora Senac: São Paulo, 2006.

COMER, Douglas E. **Interligação em Rede com TCP/IP, Volume 1, Princípios, Protocolos e Arquitetura**. Campus: Rio de Janeiro, 1998.

CONVERSE, Tim; PARK, Joyce. **PHP**: a Bíblia, 2ª edição. Elsevier Editora Ltda: Rio de Janeiro, 2003.

COSTA, Daniel Gouveia. **Administração de redes com scripts**: bash script, python e VBScript. Brasport Livros e Multimídia Ltda: Rio de Janeiro, 2007.

_____, Rogério Luís de Carvalho. **SQL**: Guia Prático. Brasport Livros e Multimídia Ltda: Rio de Janeiro, 2004.

FARRER, Harry. et al. **Programação Estruturada de Computadores**: Algoritmos Estruturados, Terceira edição. LTC - Livros Técnicos e Científicos Editora S.A: Rio de Janeiro, 1999.

FREEBSD. **THE FREEBSD PROJECT**. Disponível em: <http://www.freebsd.org/>. Acesso em 19 mar. 2012.

FREEBSD HANDBOOK. **Chapter 3 Installing FreeBSD 9.x and Later**. Disponível em: <http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/bsdinstall/bsdinstall-boot-loader-menu.png>. Acesso em 27 mar. 2012.....

FURTADO, Vasco. **Tecnologia e Gestão da Informação na Segurança Pública**. Editora Garamond Ltda: Rio de Janeiro, 2002.

JARGAS, Aurélio Marinho. **Shell Script Profissional**. Novatec Editora Ltda: São Paulo, 2008.

JEFFRIES, Amos. **Squid-cache wiki**. Disponível em: <http://wiki.squid-cache.org/SquidFaq/AboutSquid>. Acesso em: 01abr. 2012.

LINS, Sérgio. **Desafios sistêmicos**: lições aprendidas por consultores e executivos que vivenciaram a implantação de sistemas. E-papers Serviços Editoriais Ltda: Rio de Janeiro, 2009.

LIGHTTPD. **LIGHTTPD Fly light**. Disponível em: <http://www.lighttpd.net/story>. Acesso em: 02 jun. 2012.

LOPES, Arthur Vargas. **Estrutura de dados para a construção de softwares**. Editora da Ulbra: Canoas, 1999.

MARCELO, Antonio. **SQUID: configurando o proxy para Linux**. 5ª Edição. Brasport Livros e Multimídia Ltda: Rio de Janeiro, 2006.

MOTTA, Alexandre de M. **O TCC e o fazer científico: da elaboração a apresentação pública**. Copiart: Tubarão, 2009.

NAGARAJ, S. V. **Web Caching and Its Applications**. Kluwer Academic Publishers: Boston, 2004.

NIC.BR. **Núcleo de Informação e Coordenação do Ponto BR**. Disponível em: <http://www.cetic.br/empresas/2010/c-int-01.htm>. Acesso em: 17 mar. 2012.

NEVES, Julio Cezar. **Programação Shell Linux, 7ª Edição**. Brasport Livros e Multimídia Ltda: Rio de Janeiro, 2008.

NORTHCUTT, Stephen. et al. **Desvendando**: Segurança em Redes. Campus: Rio de Janeiro, 2002.

NOTEBOOKNOTES. **NotebookNotes.com**. Disponível em: <http://www.notebooknotes.com/wp-content/gallery/apple-macbook-mb467lla/apple-macbook-01.jpg>. Acesso em 26 mai. 2012.

OPPEL, Andy; SHELDON, Robert. **SQL**: A Beginner's Guide, Third Edition. McGraw-Hill Osborne Media: United States, 2009.

PALMA, Luciano.; PRATES, Rubens. **TCP/IP**: Guia de Consulta Rápida. Novatec: São Paulo, 2000.

PHP. Disponível em: <http://www.php.net/usage.php>. Acesso em 10 mar. 2012.

PYTHON. **Python ProgrammingLanguage**. Disponível em: <http://www.python.org/about/>. Acesso em 10 mar. 2012.

ROBBINS, Arnold. **Sed & awk Pocket Reference, Second Edition**. O'Reilly Media Inc: Sebastopol, 2002.

SAINI, Kulbir. **Squid Proxy Server 3.1**: Beginner'sGuide. PacktPublishingLtd: Birmingham, 2011.

SÊMOLA, Marcos. **Gestão da Segurança da Informação**: Uma Visão Executiva. Elsevier Editora Ltda: Rio de Janeiro, 2003.

SOUZA, Marco Aurélio de. **SQL, PL/SQL, SQL *Plus**: manual de referência completo e objetivo. Editora Ciência Moderna Ltda: Rio de Janeiro, 2004.

SQLITE. **About SQLite**. Disponível em: <http://www.sqlite.org/about.html>. Acesso em 06 mai. 2012.

STEVENS, W. Richard. **TCP/IP Illustrated, Volume 3**: TCP for Transactions, HTTP, NNTP, and the Unix Domain Protocolos.Pearson Education Corporate Sales Division: Indianapolis, 2002.

TANENBAUM, Andrew S. **Redes de Computadores**. Elsevier Editora Ltda: Rio de Janeiro, 2003.

TEOREY, Toby. et al. **Database Modeling and Design, 5TH EDITION**: Logical Design. Morgan Kaufmann Publishers: Burlington, 2005.

TORRES, Gabriel. **Redes de computadores**. Novaterra Editora e Distribuidora Ltda: Rio de Janeiro, 2009.

WATANABE, Cláudia Shizue. **Introdução ao Cache de Web**. Disponível em: <http:// www.rnp.br/newsgen/0003/cache.html>. Acesso em 29 mar. 2012.

WESSELS, Duane. **Squid: The Definitive Guide**. O'Reilly Media, Inc: Sebastopol, 2004.