UNIVERSIDADE SAGRADO CORAÇÃO

BRUNO CASTILHO WOELKE

DESENVOLVIMENTO DE UM SERVIÇO EM LINGUAGEM PHP PARA BACKUP VIA PROTOCOLO FTP RODANDO EM PLATAFORMA LINUX: TESTE DE VIABILIDADE

Bauru

2010

UNIVERSIDADE SAGRADO CORAÇÃO

DESENVOLVIMENTO DE UM SERVIÇO EM LINGUAGEM PHP PARA BACKUP VIA PROTOCOLO FTP RODANDO EM PLATAFORMA LINUX: TESTE DE VIABILIDADE

Trabalho de conclusão de curso apresentado ao Centro de Ciências Exatas e Naturais como parte dos requisitos para obtenção to título de bacharel em Ciências da Computação, sob orientação do Prof. Dr. Kelton A. Pontara da Costa.

Bauru

2010

BRUNO CASTILHO WOELKE

DESENVOLVIMENTO DE UM SERVIÇO EM LINGUAGEM PHP PARA BACKUP VIA PROTOCOLO FTP RODANDO EM PLATAFORMA LINUX: TESTE DE VIABILIDADE

Projeto de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do Título de Bacharel em Ciência da Computação, sob orientação do Prof. Dr. Kelton A. Pontara da Costa.

Banca examinadora:

Prof. Dr. Kelton A. Pontara da Costa
Universidade do Sagrado Coração

Prof. Esp. Andre Luiz Ferraz de Castro
Universidade do Sagrado Coração

Prof. Esp. Henrique Pachioni Martins
Universidade do Sagrado Coração

Bauru, 26 de dezembro de 2010.

Dedico este trabalho aos meus pais, José Luiz e Maria Angela, que me apoiaram nos meus estudos e me deram base desde os primeiros contados com a escola.

Em especial, dedico a minha namorada Mayara, que esteve ao meu lado, tanto em momentos bons quanto ruins, me incentivando a sempre persistir.

AGRADECIMENTOS

Primeiramente a Deus, que mesmo nos momentos mais difíceis da minha vida esteve ao meu lado, me protegendo de qualquer mal e, que com uma força maior, me impulsionou a continuar batalhando pelos meus objetivos.

Agradeço a Universidade do Sagrado Coração por ter oferecido um excelente curso nesses últimos quatro anos de estudos e de grandes esforços.

Ao meu pai, José Luiz, pelo incentivo que sempre me deu com meu curso de Ciências da Computação, me ajudando a chegar até aqui hoje.

Sem esquecer dos meus falecidos avós que foram os responsáveis por me dar um computador novo e com isso proporcionar meu interesse pela computação e por terem feito questão de investir no início dos meus estudo, desde os tempos da escola primária.

LISTA DE ABREVIATURAS E SIGLAS

PHP - Hypertext Preprocessor

PEAR - PHP Extension and Application Repository

FTP - File Transfer Protocol

TCP - Transmission Control Protocol

IP - Internet Protocol

FI - Form Interpreter

HTML - Hypertext Markup Language

DARPA - Defense Advanced Research Projects Agency

ARPANET - Advanced Research Projects Agency Network

OSI - Open Systems Interconnection

NetBIOS - Network Basic Input Output System

MIT - Massachussetts Institute of Techology

ASCII – American Standard Code for Information Interchange

EBCDIC - Extended Binary Coded Decimal Interchange Code

ISO - International Organization for Standardization

GTK – GIMP Toolkit

GIMP - GNU Image Manipulation Program

GNU – General Public License

SQL - Structured Query Language

XHTML - Extensible Hypertext Markup Language

XML - Extensible Markup Language

OS – Operational System

FSF – Free Software Foundation

PDF – Portable Document Format

CLI – Command Line Interface

SOAP - Simple Object Access Protocol

PC – Personal Computer

RFC - Request for Change

CPD – Central de Processamento de Dados

ERP – Enterprise Resource Planning

SIGE – Sistemas Integrados de Gestão Empresarial

LAN – Local Area Network

WAN - Wide Area Network

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Segundo Mendes (2009), diante de um intenso avanço tecnológico e com a expansão global física e mercadológica das empresas houve um grande aumento em seus ambientes computacionais móveis e remotos. Cada dia mais os funcionários armazenam importantes propostas, planejamentos, especificações técnicas e informações confidenciais em seus computadores. Apesar da flexibilidade e melhor produtividade que estes equipamentos portáteis oferecem, há também um risco significativo, visto que na maioria das vezes as informações não ficam protegidas como se estivessem em servidores corporativos.

Mendes (2009) ainda ressalta que em grande parte dos casos o *backup* é negligenciado, pois as pessoas acreditam não estarem sujeitas ao risco de perdas de informações, devido ao fato de que processo de *backup* é demorado ou porque preferem simplesmente manter os dados apenas em discos rígidos locais, sendo que existe a possibilidade de apresentarem erros físicos e lógicos. Além disto, existe a chance de ocorrer a exclusão acidental de arquivos, desastres naturais, ataques de vírus e até mesmo roubo ou perda de computadores. A proteção dos dados em uma estação de trabalho, remota ou móvel, exige cuidados diferentes de um computador que esteja permanentemente conectado à rede e consequentemente a um servidor de arquivos.

Segundo Pinheiro (2006), nenhum sistema de armazenamento está completo sem uma solução adequada de cópias de segurança. Assegurar a integridade dos dados é um dos maiores desafios da área de *Tecnologia da Informação* de uma empresa, principalmente porque soluções como espelhamento remoto e cópia de dados não conseguem garantir essa integridade em situações de erros humanos, sabotagens ou mesmo desastres de proporções não previstas. Em muitos destes casos, somente uma cópia tipo *backup* pode resolver a situação.

Conforme Dextra (2010), a quantidade de aplicações já existentes em *PHP*, a sua robustez, a rapidez na codificação e a facilidade de aprendizagem tornam o PHP uma escolha excelente para o desenvolvimento de aplicações de pequeno e médio porte, sendo cada vez mais um padrão de fato.

Conforme Maffeis (2010), uma das principais vantagens é o custo, o *Linux* é um software livre, isto é, sua utilização não tem custos financeiros, você não paga

nada para usá-lo. Em termos de segurança, a compatibilidade com padrões estabelecidos há mais de duas décadas e em constante evolução, faz do *Linux* um sistema reconhecido pela sua estabilidade e robustez, dando uma maior segurança às redes que utilizam este produto.

Devido aos fatos, é visto que há uma grande importância em efetuar *backups* em servidores remotos, os quais podem ter menor confiabilidade que a dos servidores locais ou até mesmo serem afetados por pessoas mau intencionadas, assim tomamos consciência em relação à integridade dos arquivos remotos.

A solução que será apresentada nesse estudo é o desenvolvimento de uma aplicação que funcionará como serviço ou *daemon* como é chamada no *Linux,* que fará todo o processo de *backup* automatizado de modo incremental dos arquivos em servidores remotos das contas de *FTP*, esta aplicação será em ambiente *Linux* por ser seguro, possuir excelente desempenho e praticidade, será desenvolvida na linguagem *PHP* pela facilidade de desenvolvimento, manutenção e, enfim, onde estará o foco da pesquisa, verificar a viabilidade da linguagem *PHP* neste tipo de aplicação.

1.1 OBJETIVO GERAL

O objetivo deste estudo é desenvolver uma aplicação na linguagem *PHP* e que funcionará como serviço em ambiente *Linux* com o propósito de realizar *backups* via protocolo *FTP*, devendo a mesma ser viável para a implantação em escalas até mesmo maiores, como as coorporativas.

1.2 OBJETIVOS ESPECÍFICOS

Desenvolver uma aplicação na linguagem *PHP* que é interpretada, buscando facilidade tanto no desenvolvimento como na manutenção e melhoramentos posteriores, rodar em ambiente *Linux* como serviço ou *daemon* buscando desempenho, estabilidade, gerência padronizada e utilizar o protocolo *FTP* para que a o objetivo principal da pesquisa seja alcançada, no caso, realizar *backup* em

diversos servidores remotos independente da plataforma, pois o *FTP* é um protocolo multi-plataforma.

Enfim, tentar reduzir os custos já que todos os itens acima são *open source* e disponibilizados sem custo de licença além de provar o desempenho a viabilidade da linguagem *PHP* neste tipo de aplicação.

1.3 JUSTIFICATIVA

O trabalho visa sanar as dificuldades de empresas que possuem acesso com restrições ou limitado em relação aos arquivos nos servidores remotos, onde todo o trabalho de enviar ou receber esses arquivos ficam por conta dos serviços de *FTP*. Levando em consideração que podem existir muitas contas de *FTP* e que não há serviço de *backup* local nestes servidores, a solução encontrada foi uma aplicação para efetuar backups em múltiplas contas de *FTP* e que funcione como serviço para facilitar o trabalho de gerenciamento do operador de segurança.

Devido ao fato de que esses arquivos muitas vezes importantes poderem ser acessados facilmente por um usuário e senha apenas e que podem vir a ser extraviados ou roubados dando acesso a pessoas não autorizadas, que por sua vez podem causar algum tipo de estrago como alteração ou exclusão desses arquivos, além de que os próprios usuários podem excluir um arquivo acidentalmente, assim visto, um serviço para *backup* dessas contas de *FTP* traria mais segurança, garantindo a integridade dos arquivos de modo organizado e de fácil gerenciamento.

Por este serviço ser desenvolvido na linguagem *PHP* que é interpretada, de fácil manutenção e de alto nível, seu código pode ser facilmente aperfeiçoado de acordo com as necessidades de cada organização, evitando o processo de compilação e outras dificuldades das linguagens de baixo nível e/ou que precisam ser compiladas.

Em se tratando de aplicações em modo *daemon,* vê se muita a utilização de linguagens de baixo nível e compilada no desenvolvimento das mesmas, é onde entra o principal aspécto desta pesquisa, avaliar a viabilidade de utilizar-se a linguagem *PHP* neste tipo de aplicação, já que a mesma é de alto nível e interpretada e pode apresentar desempenho um pouco ruim.

O sistema operacional *Linux*, que será utilizado no projeto, tem a vantagem de ter de código aberto e livre, constante crescimento, excelente desempenho, principalmente combinado ao *PHP* e disponível sem qualquer custo trará maior segurança, qualidade no resultado da aplicação e menor investimento financeiro.

Observa-se também que a aplicação a ser desenvolvida pode ser tranquilamente aplicada quando há necessidade de se realizar *backups* com muita facilidade em diferentes servidores através da rede ou Internet, independente da plataforma de cada um, bastando apenas possuir um servidor *FTP* corretamente configurado.

A capacidade de ser multi-plataforma vem do *FTP*, um protocolo muito comum e que sua comunicação entre computadores na rede independe do seus respectivos sistemas operacionais.

Devido ao fato do *hardware* hoje estar sendo mais aproveitado pelos sistemas operacionas e baratiado pelo avanço da tecnologia, sendo este mais rápido que o avanço dos *softwares*, então usar a linguagem *PHP* para este tipo de aplicação, não deve apresentar grande problema de performance, que no caso poderá de qualquer maneira ser compensada pela facilidade que se tem no desenvolvimento de manutenção do código da linguagem *PHP.*

Pode-se também supor que pela comunicação via *FTP* ser algo que limita se ao tempo de resposta da conexão devido as diversas requisições que o *FTP* faz para interagir com cada arquivo, pode acarretar lentidão na performance geral da aplicação, fazendo com que a mesma obtenha desempenho de acordo com esta limitação do *FTP,* concluindo assim que a performance do *PHP* neste tipo de aplicação será muito próxima as outras linguagens compiladas e assim, sendo viável sua implementação, compensada pela facilidade que se tem no desenvolvimento de manutenção do código da linguagem *PHP.*

1.4 ESTRUTURA DO TRABALHO

A estrutura deste estudo está dividida em capítulos que serão explicados a seguir.

O primeiro capítulo apresenta a contextualização, problemas de pesquisa, objetivo e justificativa para o desenvolvimento do trabalho.

O segundo capítulo é todo levantamento bibliográfico do estudo onde aborda a importância da integridade dos arquivos e a importância de manter cópias de *backup*, *FTP*, *TCP/IP*, *PHP* e *Linux*, contendo a história e conceitos de cada item.

No terceiro capítulo serão apresentados os materiais e métodos usados no desenvolvimento do estudo.

O quarto capítulo apresentará os resultados obtidos com o presente estudo.

2 LEVANTAMENTO BIBLIOGRÁFICO

Segundo Morimoto (2004), o *FTP* é ainda o protocolo de transferência de arquivos mais utilizado na Internet e uma opção valiosa também para redes locais.

Conforme Fialho (2007), a segurança dos dados, apesar de ser prioridade em um *CPD*, não deve ser assunto somente em grandes empresas.

Conforme Alvarez (2010), é uma linguagem de programação gratuita e independente de plataforma, rápida, com uma grande livraria de funções e muita documentação.

Alvarez (2010) ainda afirma que *PHP*, no caso de estar montado sobre um servidor *Linux* ou *Unix*, é mais rápido, dado que se executa em um único espaço de memória.

Conforme Guillet (2006), a estabilidade é uma das maiores virtudes do *Linux*. Sua arquitetura de processos é transparente. O *kernel* modularizado permite rodar processos com independência e é raro que o sistema caia.

Assim conclui-se que o *FTP* é uma excelente ferramenta para a manutenção de arquivos remoto, ideal para este tipo de aplicação de *backup* remoto a ser desenvolvido na linguagem *PHP*, a qual possui diversas facilidades tanto para o desenvolvimento quanto para a manutenção, rodando em plataforma *Linux*, que é o ideal para este tipo de aplicação.

2.1 SEGURANÇA DOS ARQUIVOS

Conforme Lyra (2008), dentre os meios mais comuns de proteger dados armazenados está o antivírus. A grande maioria dos problemas relacionados a incidentes de segurança em computadores pessoais é causada por programas maliciosos, dentre os quais estão os vírus, os *worms*, os cavalos de tróia e outros. Um bom sistema antivírus é um elemento essencial para proteção de redes conectadas à internet. O *software* antivírus ajuda a impedir ataques vasculhando arquivos periodicamente em busca de mudanças inesperadas em arquivos, sequência de códigos similares às armazenadas em uma base de dados de vírus conhecidos, anexos de *e-mails* e outros sinais de alerta.

Lyra (2008) ainda afirma que há outros meios importantes que são as criptografias e as esteganografias. A criptografia é definida como arte ou ciência de escrever em cifras ou códigos, com o propósito de restringir ao destinatário a capacidade de decodificá-la e compreendê-la. Mecanismos de criptografia são amplamente adotados em ambientes computacionais para oferecer garantia da autenticação, privacidade e integridade de dados e comunicações, e sem essa tecnologia não seria possível popularizar o comércio eletrônico, por exemplo. As técnicas de esteganografia possibilitam a ocultação de uma formação dentro de outra, usando o princípio da camuflagem. No caso da esteganografia digital, informações podem ser escondidas em arquivos de imagem, som, texto, etc. Normalmente estes arquivos apresentam áreas com dados inúteis ou pouco significativos, que podem ser substituídos pela informação que deseja esconder.

Conforme Mendes (2009), outra fonte de problemas que precisa ser considerada são os defeitos inerentes aos próprios produtos de *hardware*. Medidas devem ser tomadas, de forma preventiva, para garantir o perfeito funcionamento dos equipamentos, com o planejamento de manutenções periódicas para minimizar possíveis cenários de mau funcionamento. Neste caso, *backups* são cruciais em caso de uma perda de dados proveniente de uma falha de *hardware*. A *ISO* 17.799 recomenda que as mídias de computador sejam controladas e fisicamente protegidas. A segurança dos *backups* é uma das grandes preocupações. Armazenamento, controle de acesso às mídias, cópia e descarte devem ser alvo de uma política específica, a fim de proteger este ativo.

2.1.1 Importância do backup dos dados

Conforme Fialho (2007), quanto mais poderoso é um computador (em termos de memória, capacidade de armazenamento e performance), mais os dados que ele armazena devem ser protegidos, pois, geralmente, este computador é um recipiente perfeito para guardar milhares e milhares de megabytes, devido justamente à sua capacidade. A melhor forma de proteger os dados de um sistema é fazendo cópias de segurança ou *backups*, regularmente. As cópias de segurança em computadores são instrumentos importantes para compensar – ou tentar sanar – problemas advindos de *hardware*, como a invasão do sistema por *hackers* ou *crackers*, ataques

de vírus, perda acidental de arquivos e etc. Por isso, a cópia de segurança é a melhor forma de prevenção e recuperação das informações, já que os dados podem voltar fielmente para o disco, quando se for necessário.

Fialho (2007) ainda afirma que em várias empresas que dependem de sistemas e de computadores, a perda de dados representa a perda de capital. Trazendo esta realidade para a sua vida cotidiana, na qual o computador é usado como ferramenta de trabalho, perder dados significa perder tempo, perdendo tempo, perde-se dinheito e, por seguinte, o cliente.

Segundo Pinheiro (2006), atualmente os sistemas corporativos requerem soluções de *backup* cada vez mais velozes, flexíveis e confiáveis, preparadas para atender uma multiplicidade igualmente maior de plataformas. Essa necessidade de garantir a integridade e a segurança da informação é tão grande que os profissionais de redes não podem contar apenas com simples sistemas de armazenamento, necessitando utilizar recursos mais eficientes como os sistemas de *backup* corporativo, por exemplo.

2.1.2 Metodologias dos backups

A frequência dos *backups*, ou seja, o número de vezes que você fará as cópias, depende diretamente da relevância que os dados possuem para a empresa em que trabalha, assim como da quantidade de informações que serão processadas, seja em sistemas de pequeno, médio ou grande porte. Há empresas que fazem suas cópias com frequência diária, devido à movimentação constante de seus dados. Há outras que fazem semanalmente, quinzenalmente; isto dependerá, sobretudo, da importância que a empresa ou usuário dá à segurança de seus dados. Porém, lembre-se de uma dica: não confie sempre na sorte (FIALHO 2007).

Segundo LNCC (2010), caso o sistema seja pequeno, pode-se considerar um método pequeno, de *backups* completos, ou seja, que tudo seja copiado todo o dia, e que garante um pequeno número de mídia. O incoveniente é a demora do *backup*. Mas numa grande empresa, precisa-se colocar este sistema de uma maneira funcional, onde não se pode perder grande tempo com o *backup*, e além do mais, muitas empresas não investem em quantidades de mídia. Por isso uma estratégia de

backup é importante. Nesses casos cria-se um sistema de *backup* completo e um incremental, que copiará somente os dados alterados durante o dia, ou semana.

Veja o esquema:

• *Backup* completo no domingo;

• *Backup* incremental na segunda-feira;

• *Backup* incremental na terça-feira;

• *Backup* incremental na quarta-feira;

• *Backup* incremental na quinta-feira;

• *Backup* incremental na sexta-feira;

• *Backup* incremental no sábado;

LNNC (2010) complementa que nesta tabela, utilizam-se sete mídias para efetuar o *backup* do sistema, e no caso de algum problema, restaura-se a última cópia full e os *backups* incrementais que satisfaçam a nossa necessidade. A vantagem deste sistema é a economia de mídias. Pois se faz *backups* completos todos os dias poderiamos utilizar mais de uma mídia por dia. Teste sempre o *backup*, pois precisa-se confiar no sistema, e não deixar para descobrir que ele é ineficiente quando perder todos os dados.

Segundo Pinheiro (2006), é necessário realizar um levantamento relacionando todos os servidores que serão os clientes do *backup/restore* e todos os sistemas operacionais, bancos de dados e aplicativos envolvidos e da rede de interligação dos mesmos, objetivando garantir a confiabilidade da operação, mesmo em situações de falha de alguns dos componentes. O dimensionamento e configuração dos servidores de *backup* e dos dispositivos de *backup* (incluindo número de unidade de fitas que é ilustrada pela figura 1, para cada dispositivo) devem ser indicados, assim como a arquitetura de rede e política de compartilhamento de unidades de fita.

Figura 1: Fita *DAT* de 40GB para *backup*

Fonte: (BARROS, 2009)

2.2 CONCEITOS DO PHP

Segundo Soares (2007), o *PHP* é acrônimo de *Hypertext Preprocessor* (pré-processador de hipertexto), uma poderosa linguagem de programação *open source*, mundialmente conhecida e utilizada, principalmente no ambiente *web*, qual característica pode ser entendida pela figura 2 (apesar de existir a versão *PHP-GTK* para embiente desktop e para console de linha de comandos). Uma das características mais marcantes no *PHP* é sua capacidade de se misturar ao *HTML* com as tags de abertura “<?php” e de fechamento “?>”, tornando mais fácil a geração de páginas *web* dinâmicas.

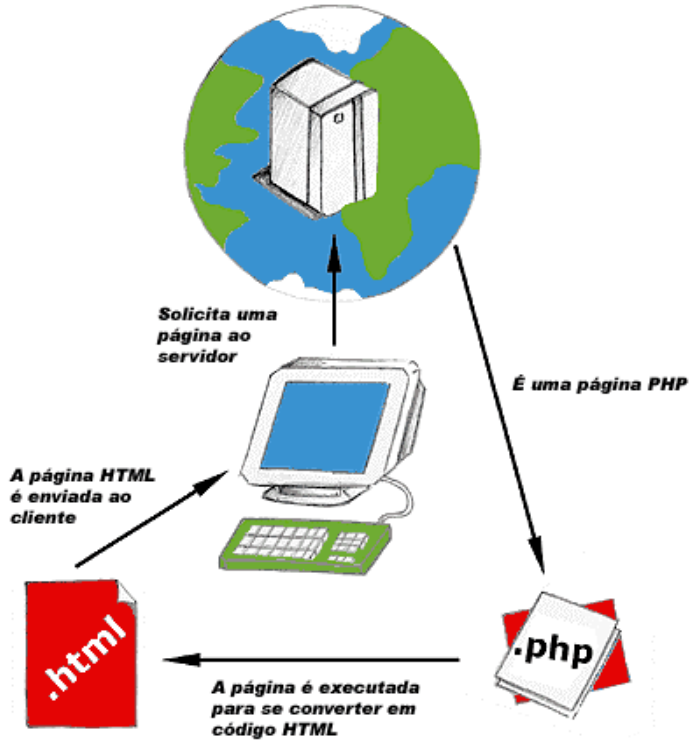

Figura 2: Funcionamento das páginas *PHP*

Fonte: (ALVAREZ, 2004)

Soares (2007) ainda afirma que o *PHP* aceita três bases de tipagem de variáveis, sendo elas a escalar com inteiro, ponto flutuante, *string*, booleano, a composta com vetor e objeto e a especial com recurso ou ponteiro e nulo.

Segundo Buyens (2002), o *PHP* é uma linguagem interpretada, Ela parece muito com a linguagem *C*. Ela também tem alguma coisa da linguagem *Perl* e

encontra-se disponível para todas plataformas, incluindo *Linux*, outras versões da família *Unix* e *Windows*.

Buyens (2002) ainda afirma que as linguagens ou são interpretadas ou compiladas. Um interpretador é um programa que lê um arquivo contendo o código a ser executado e age imediatamente sobre ele. O código no arquivo é chamado de código-fonte. Em geral, esse código pode ser lido e entendido por uma pessoa.

Segundo Dextra (2010), o *PHP* é uma linguagem de programação bastante poderosa e simples. A conseqüência disto é que a curva de aprendizado para os programadores é curta, permitindo uma rápida proficiência dos desenvolvedores nesta linguagem. Caso o programador tenha alguma familiaridade prévia com *C*, a aprendizagem será ainda mais fácil.

2.2.1 História do PHP

Conforme Adonai (2005), a linguagem *PHP* foi concebida durante o outono de 1994 por *Rasmus Lerdorf*. As primeiras versões não foram disponibilizadas, tendo sido utlizadas em sua *homepage* apenas para que ele pudesse ter informações sobre as visitas que estavam sendo feitas. A primeira versão utlilzada por outras pessoas foi disponibilizada em 1995. Era composta por um sistema bastante simples que interpretava alguns macros e alguns utilitários que rodavam por trás das *homepages*: um livro de visitas, um contador e algumas outras coisas. Em meados de 1995 o interpretador foi reescrito e ganhou o nome de *PHP/FI*, o *FI* veio de um outro pacote escrito por *Rasmus* que interpretava dados de formulários *HTML* (*Form Interpreter*). Ele combinou os *scripts* do pacote *Personal Home Page Tools* com o *FI* e adicionou suporte a *MySQL*, nascendo assim o *PHP*/*FI*, que cresceu bastante, e as pessoas passaram a contribuir com o projeto. Estima-se que em 1996 *PHP*/*FI* estava sendo usado por cerca de 15.000 sites pelo mundo, e em meados de 1997 esse número subiu para mais de 50.000, e já em 1999 o crescimento alcançou mais de 100.000 sites por todo o mundo, sendo ela utilizada por diversas organizações como ótima ferramenta para o desenvolvimento *web*. Nessa época houve uma grande mudança no desenvolvlmenho do *PHP*. Após este período de grandes mudanças na linguagem e grande adoção por parte das organizações, colaboradores e desenvolvedores, pode-se ver que ao passar dos anos, o *PHP* tem

levantando grande crescimento no período de 1999 à 2003, coompreende-se melhor atravéz da figura 3.

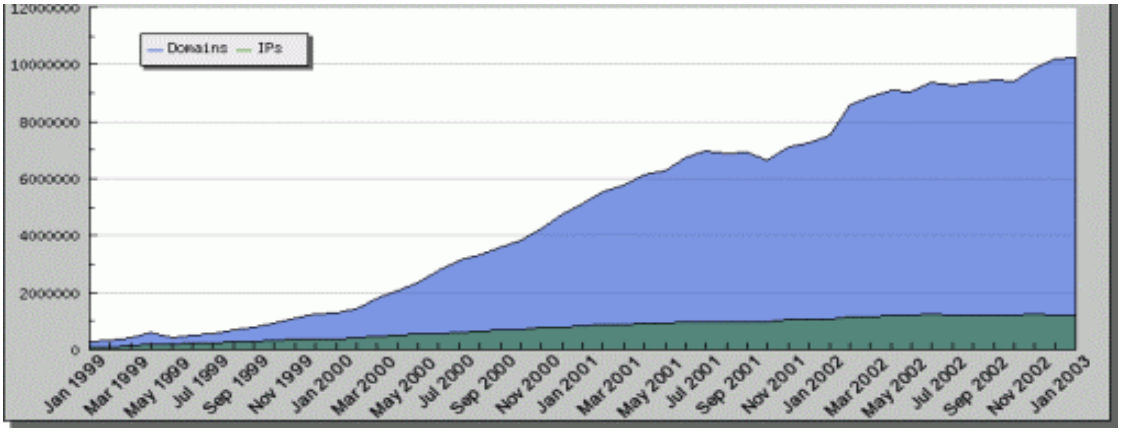

Figura 3: Crescimento do uso do *PHP* de 1999 à 2003

Fonte: (DEXTRA, 2010)

Adonai (2005) ainda afirma, ele deixou de ser um projeto de *Rasmus* com contribuições de outras pessoas para ter uma equipe de desenvolvimento mais organizada. O interpretador foi reescrito por *Zeev Suraslti* e *Andi Gutmans*, e esse novo interpretador foi a base para a versão 3.

Segundo Araújo (2006), depois do lançamento da versão 3, o *PHP* cresceu muito e o *PHP* 3 passou a ser um marco na história da linguagem. E no ano de 1998 (no final), *Zeev* e *Andi* revisaram o *PHP* 3 e resolveram novamente reescrever a linguagem. Em 22 de maio de 2000, o *PHP* 4 foi lançado com um novo paradigma para a execução dos scripts *PHP*. O *PHP* 3 analisava e executava o código ao mesmo tempo enquanto que o *PHP* 4 passou a compilar o código inteiro transformando-o em *byte code* para somente depois ser executado pelo *Zend Engine*. Esse novo procedimento deixou o *PHP* 4 muito mais rápido do que o *PHP* 3.

Araújo (2006) ainda ressalta que em 27 de dezembro de 2002 a versão 4.3.0, última versão significativa da edição 4, foi lançada e trouxe algumas novidades como suporte a *streams*, *CLI* e a bilioteca *GD* inclusa. Depois da versão 4.3.0 a demanda por funcionalidades orientadas a objetos cresceu e por esse motivo *Andi* decidiu reescrever a parte orientada a objetos da linguagem e iniciou o trabalho escrevendo, junto com *Zeev*, o documento "*Zend Engine II*: *Feature Overview and Design*". A partir desse momento, muitas funcionalidades foram adicionadas, retiradas e modificadas, até que em 13 de julho de 2004 o tão esperado *PHP* 5 foi lançado trazendo como principal novidade a orientação a objetos totalmente reescrita. Além

da orientação a objetos, a versão 5 também trouxe funcionalidades muito importantes para a evolução da linguagem, como *SimpleXML*, *SOAP*, *MySQLi* e mais diversas extensões de importância significativa.

Araújo (2006) afirma por último que no dia 11 de novembro de 2005, *Rasmus*, *Zeev* e outros desenvolvedores do núcleo do *PHP*, se reuniram em *Paris* para discutir sobre o futuro da linguagem. Dessa reunião surgiu o documento *Minutes PHP Developers Meeting* escrito por *Derick Rethans* um dos atuais desenvolvedores do núcleo do *PHP*. Esse documento passou a ser o guia para o desenvolvimento do *PHP* 6.

Conforme PHP Group (2004), várias outras pessoas têm contribuído com o *PHP* até hoje, algumas das quais estão listadas na documentação *on-line*.

2.2.2 Funcionalidades do PHP

Segundo Soares (2007), um script *PHP* pode conter código procedural ou orietado a objetos ou ambos mesclados no mesmo código. Assim tornando o *PHP* muito flexível em relação as técnicas de programação implementadas em cada projeto, gerando facilidades desde o desenvolvimento de um projeto até sua fase final de acabamentos, onde está envolvido os testes, depuração e manutenção do código fonte. O código do *PHP* é recompilado automaticamente em cada requisição que se detecta alteração no código, isso faz com que o script seja interpretado apenas uma vez, nas demais vezes um arquivo binário executável gerado é executado, trazendo um grande desempenho em relação as linguagens limitadas apenas a interpretação em tempo real.

Conforme PHP Group (2004), com *PHP* você não está limitado a gerar somente *HTML* dinâmico. As habilidades do *PHP* incluem geração de imagens, arquivos *PDF* e animações *Flash* criados dinamicamente. Você pode facilmente criar qualquer padrão texto, como *XHTML* e outros arquivos *XML*. O *PHP* pode gerar esses padrões e os salvar no sistema de arquivos, em vez de imprimi-los, formando um cache dinâmico de suas informações no lado do servidor. Isso trará ao *script*, independente da aplicação, uma ampla margem de possibilidades de funcionalidades essenciais além do seu foco principal, para que a qualidade do *PHP*

seja reconhecida no mercado mundialmente, entre os desenvolvedores nos tempos modernos da Internet de hoje.

```
class Person {
    var $name;
    function getName() {
        return $this->name;
    }
    function setName($name) {
        $this->name = $name;
    }
    function Person($name) {
        $this->setName($name);
    }
}

function changeName($person, $name) {
    $person->setName($name);
}

$person = new Person("Andi");
changeName($person, "Stig");
print $person->getName();
```

Figura 4: Padrão do código do PHP 5

Fonte: (PHP Group, 2004)

PHP Group (2004) complementa que este pedaço de código, referenciado pela figura 4, quando executado através do *PHP* 4, irá ter como resultado de saída a palavra “Andi”. Isto ocorre porque é passado o objeto “$person” para a função “changeName()” por valor, e assim a função “changeName()” trabalha com um clone de “$person”. No *PHP* 5, a infraestrutura da modelagem do objeto foi reescrita para trabalhar com os ponteiros dos objetos. A menos que um objeto seja explicitamente criado clonando um objeto através de um operador, o gerenciador não irá criar clones de objeto.

Conforme Buyens (2002), é visto que o *PHP* também pode ser montado para ser executar como um programa independente, processando um arquivo contendo o código *PHP*. Isso permite executar programas automaticamente usando o *cron* do *Linux* ou a partir da linha de comando. Toda saída do *PHP* quando este é executado desta forma vai para *stdout (*canal de saída*).* Atravéz deste modo de interpretação do *script*, é possível desenvolver aplicações que incluem serviços, aplicações inicializadas via linha de comando, automatizadas via *cron* e com portabilidade entre sistemas operacionais diferentes. Isso torna o *PHP* uma linguagem que deixa de ser específica para gerar códigos *HTML*.

2.3 CONCEITOS DO SISTEMA OPERACIONAL LINUX

Segundo Danash (2000), o termo Linux é, na verdade, um tanto vago. *Linux* é usado de duas maneiras: especificamente para se referir ao *kernel* em si, o coração de qualquer versão de *Linux*, e geralmente para se referir a qualquer conjunto de aplicativos que sejam executados no *kernel*, normalmente referidos como distribuição. A tarefa do *kernel* é fornecer o ambiente global em que os aplicativos possam ser executados, incluindo as interfaces básicas com o *hardware* e os sistemas de gerenciamento de tarefas e programas que estejam em execução. No sentido específico, existe apenas uma versão atual do *Linux* a qualquer tempo: a revisão atual do *kernel*. *Linus Torvalds* mantém o *kernel* como seu domínio no mundo do desenvolvimento *Linux*, deixando todos os aplicativos e serviços que dependem do *kernel* para milhares de outros projetistas da comunidade *Linux*.

Danash (2000) ainda ressalta que no significado geral do termo, se referindo aos coesivos conjuntos de aplicativos executados no *kernel* do *Linux*, existem numerosas versões de *Linux*. Cada distribuição tem duas características próprias, incluindo diferentes métodos de instalação, diferentes conjuntos de recursos e diferentes caminhos de atualização. Mas como todas as distribuições são fundamentalmente *Linux*, em quase todos os casos um aplicativo que funciona com uma versão atual de uma distribuição também funcionará com a versão atual de outra. O interessante a respeito desse uso dicotômico do termo *Linux* é que ele estabelece o mesmo paralelo com a utilização confusa do termo “sistema operacional”. No sentido comercial, um sistema operacional significa um grande conjunto de aplicativos centralizados em torno de um *kernel*. É isso o que é o *Windows*. É isso que é o *Mac OS*. No sentido técnico purista, um sistema operacional é um *kernel* de núcleo muito menor, que oferece as funções de sistema básicas necessárias para se desenvolver quaisquer aplicativos.

2.3.1 História do Linux

De acordo com Danash (2000), a aparição do *Linux* no cenário da computação teve origem na cultura *Unix*. Como sistema operacional (na realidade, uma variedade de diferentes sistemas operacionais com características

semelhantes), o *Unix* é bem anterior à era dos computadores *desktop*, tendo sido desenvolvido em meados dos anos 70, quando os minicomputadores e os computadores de grande porte eram a norma no mundo coorporativo.

Conforme Silva (2006), em 1989, um estudante finlandês chamado *Linus Torvalds* inicia um processo pessoal de aprimoramento do *kernel* do *Minix* um sistema operacional do tipo *Unix* escrito por *Andrew Tannenbaum*, chamando esta vertente de *Linux* como abreviação de *Linus´s Minix*. Depois de um certo tempo de trabalho, *Linus* envia a seguinte mensagem para o grupo de discussão *comp.os.minix*:

> "Você sente saudade dos bons dias do minix-1.1, quando homens eram homens e escreviam seus próprios device drivers? Você está sem um bom projeto e morrendo de vontade de colocar as mãos em um sistema operacional o qual possa modificar de acordo com suas necessidades? Você acha frustante quando tudo funciona bem no Minix? Sem mais noites em claro para fazer com que um programa funcione? Então esta mensagem pode ser exatamente para você. :-)
> Como eu mencionei há um mês, estou trabalhando em uma versão livre de um sistema operacional similar ao minix para computadores AT-386. Ele finalmente alcançou o estágio onde pode ser utilizado (ou não, dependendo do que você deseja), e eu estou disposto a colocar os fontes disponíveis para ampla distribuição. Ele está apenas na versão 0.02, mas eu tenho executado nele, sem problemas, programas como bash, gcc, gnu-make, gnu-sed, compress, etc." (SILVA, 2006)

Silva (2006) afirma que em 1990, a *FSF* já tinha obtido ou escrito vários componentes importantes do sistema operacional *GNU*, com exceção de um *kernel*. Em 1991, em 5 de outubro deste ano, *Linus Torvalds* anuncia a primeira versão oficial do *Linux*. Em 1992, o *Linux* se integra a *GNU* com o objetivo de produzir um sistema operacional completo. Desde então, muitos programadores espalhados pelo globo terrestre têm seguido os ideais de *Richard Stallman* e *Linus Torvalds*.

2.3.2 Funcionalidades em geral do Linux

Segundo Taylor (2000), *Unix* é um sistema operacional para computador, um programa de controle que trabalha com usuários visando executar programas, gerenciar recursos e comunicar-se com outros sistemas de computadores. Várias pessoas podem usar um computador *Unix* ao mesmo tempo; por essa razão, o *Unix* é chamado sistema multiusuário. Qualquer um desses usuários pode executar vários programas ao mesmo tempo; por essa razão o *Unix* é chamado multitarefa. Como o *Unix* é um projeto realizado em partes, ele é muito mais do que um simples sistema

operacional. O *Unix* tem mais de 250 comandos individuais, que variam desde comandos simples, para copiar um arquivo, por exemplo, até os bastantes complexos como aqueles usados em redes de alta velocidade, gerenciamento de revisão de arquivo e desenvolvimento de *software*.

Conforme Conectiva (2010), o *Linux* é um sistema operacional derivado do *Unix* feito para rodar em computadores pessoais. O *Linux* faz tudo o que você poderia esperar de um *Unix* moderno e completo. Suporta multitarefa real, memória virtual, bibliotecas dinâmicas, redes *TCP/IP*, nomes de arquivos com até 255 caracteres e proteção entre processos (*crash protection*), além de muitas outras funcionalidades que deixariam esta lista extensa demais. Um grande atrativo que o *Linux* oferece é o fato de poder trabalhar tanto como servidor de aplicações quanto como estação de trabalho, sem que haja necessidade de grandes modificações no seu sistema. Na figura 6 pode-se ver a estrutura modularizada do *Linux,* um dos responsáveis pela facilidade de funcionar como servidor ou estação de trabalho:

Figura 5: Esquema modular do *Linux*/*Unix*

Fonte: (NCE, 2000)

Segundo Oliveira (2010), os ambientes gráficos mais utilizados são o *KDE* e o *Gnome*. Pode-se usar qualquer um deles, conforme dermos mais importância à simplicidade, rapidez, ou ao design, pois estes diferem na forma como estão organizados, e nos programas que já trazem pré-instalados.

2.3.3 Diretórios específicos para o presente estudo

Conforme Morimoto (2004), o diretório “/etc” concentra os arquivos de configuração do sistema, de certa forma substituindo o registro do *Windows*. A

vantagem é que enquanto o registro é uma espécie de caixa preta, os *scripts* do diretório “/etc” são desenvolvidos de modo a facilitar a edição manual.

Conforme Filho (2003), o diretório /var contém arquivos que são modificados com o decorrer do uso do sistema (*e-mail*, temporários, filas de impressão, manuais). O diretório **“**/var/log” contém arquivos de *log* do sistema (erros, *logins*, etc..).

2.3.4 Definição de processos e serviços do Linux

Segundo Taylor (2000), quer você esteja solicitando um manual com *man*, listando arquivos com *ls*, iniciando o *vi* ou executando qualquer comando do *Unix*, você estará iniciando um ou mais processos. No *Unix* qualquer programa que esteja sendo executado é um processo. Você pode ter vários processos sendo executados de uma só vez. O canal “*ls –l* | *sort* | *more*” solicita três processos: *ls*, *sort*, *more*. Os processos, tanto no “*shell C”* quanto no *Korn*, também são conhecidos como serviços e o programa que você está executando é conhecido como serviço atual.

Taylor (2000) ainda afirma que qualquer processo pode ter vários estados, “executando” é o estado típico. Há muitos aspectos relacionados a processos e no *Unix*, principalmente considerando o nível de controle oferecido pelo *shell*.

2.3.5 O Shell do Linux (linha de comando)

Segundo Danash (2000), ao contrário do *Windows*, onde o *prompt* é um ambiente fixo com flexibilidade limitada, os *shells* do *Unix* são pequenos programas aplicativos, executados como processos quando você se conecta, que fornecem uma variedade de características de interface de linhas de comando e recursos de acordo com diferente usuários e aplicativos. Assim, no *Linux* existem numerosos *shell* para se escolher. Cada *shell* oferece um conjunto de características e recursos diferentes, e a maioria oferece sua própria linguagem de *script*.

Na figura 6 pode-se ver onde o *shell* atua em relação ao usuário.

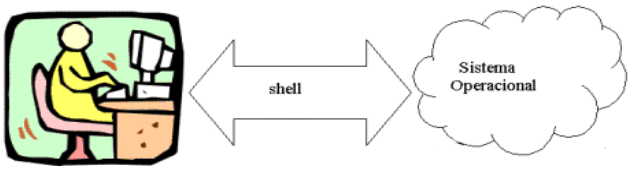

Figura 6: Esquema de interação usuário-*shell*-sistema

Fonte: (FILHO, 2000)

O sistema operacional e na figura 7 demonstra a interação do usuário com o *shell* do *Linux*.

Figura 7: Exemplo de um *shell Linux*

Fonte: (ALIENAÇÃO, 2010)

2.4 CONCEITOS DE REDE POR TCP/IP

Segundo Santos (2004), o *TCP* e o *IP* são protocolos da camada de transporte e rede respectivamente, onde o *TCP* é um protocolo orientado à conexão e com garantia de entrega do pacote. O protocolo *IP* é responsável por traçar um caminho por onde os dados serão enviados. Em uma rede de computadores, além dos protocolos que são responsáveis pela comunicação, existem os dispositivos de rede que são as portas de entrada e de saída dos dados de cada equipamento de rede. A esses dispositivos são atribuídos endereços *IP*,

que são os endereços lógicos de cada dispositivo, responsáveis por identificar cada equipamento na rede.

Santos (2004) ainda afirma que protocolo é a linguagem usada pelos dispositivos de uma rede de modo que eles consigam se entender, isto é, trocar informações entre si. Para que todos os dispositivos de uma rede consigam conversar entre si, todos eles deverão estar usando uma mesma linguagem, isto é, um mesmo protocolo.

2.4.1 História da rede e dos protocolos

Conforme Comer (2006), parte que torna o protocolo *TCP*/*IP* tão interessante é sua adoção universal e o tamanho e taxa de crescimento da internet global. A *DARPA* começou a trabalhar em direção a uma tecnologia de interconexão de redes em meados dos anos 70, com a arquitetura e os protocolos tomando sua forma atual por volta de 1977-79.

Comer (2006) ainda afirma que a internet global começou por volta de 1980, quando a *DARPA* começou a converter máquinas conectadas às suas redes de pesquisa aos novos protocolos *TCP*/*IP*. Após sete anos de seu início, a internet cresceu para se espalhar por centenas de redes individuais, localizadas por todos os *Estados Unidos* e *Europa*. Conectava quase 20.000 computadores em universidades, governos e laboratórios de pesquisa. Tanto o tamanho quando o uso da Internet continuou a crescer muito mais rapidamente do que foi previsto. Ao final de 1987, estimou-se que o crescimento tinha atingido 15% por mês. Em 2005, a internet global atingiu quase 300 milhões de computadores em 209 países.

2.4.2 Camadas que envolvem sua funcionalidade

Segundo Parihar (2002), o *TCP/IP* não é um único protocolo; é um conjunto de protocolos. Por causa da diversidade do *TCP/IP*, ele não utiliza diretamente o modelo *OSI*. Em vez disso, ele utiliza um modelo de quatro camadas para comunicação. A quarta camada do modelo de referência do *TCP/IP* é a camada de aplicativo. Essa camada é responsável pelos aplicativos do *TCP/IP*. Há dois tipos de aplicativos nessa camada: aplicativos baseados em soquete e aplicativos do sistema

básico de saída e entrada da rede (*NetBIOS*), conforme pode ser melhor coompreendido na figura 8.

Modelo OSI	Conjunto de protocolos TCP/IP
Aplicativo	Aplicativo: Telnet, FTP e outros
Apresentação	
Sessão	
Transporte	Transporte: TCP
Rede	Internet: IP, ARP, ICMP
Enlace	
Física	Física

Figura 8: Comparativo do Modelo *OSI* com o *TCP/IP*

Fonte: (PARIHAR, 2002)

Parihar (2002) ainda afirma que aplicativos baseados em soquete existem em todos os clientes que utilizam o *TCP/IP*. Três elementos são exigidos para os aplicativos baseados em soquete: um endereço de *IP*, uma porta e um tipo de serviço. Como mencionado anteriormente, cada cliente que utiliza o *TCP/IP* terá um endereço único de 32 bits. Cada endereço tem 65.536 pontos de entrada chamados de portas. Os aplicativos de *TCP/IP* operam em portas particulares. (aplicativos comuns de *TCP/IP* serão definidos mais adiante nesta seção, junto com a porta que cada aplicativo utiliza na comunicação.)

Comer (2006) afirma que soquete, no caso de redes de computadores, é virtual, seria algo como um enlace entre a camada do SO responsável por receber e trasmitir os pacotes que vem da camada construtora de pacotes com os dados que anteriormente veio de uma aplicação, para a camada responsável por fazer o transporte dos pacotes na rede.

2.5 CONCEITOS DE FTP

Segundo Vita (2005), o protocolo de transfêrencia de arquivos da internet, *FTP* foi desenvolvido com o objetivo de transferir arquivos de maneira eficiente e confiável entre dois computadores na rede, e através disto incentivar o compartilhamento de arquivos entre diferentes máquinas, escondendo do usuário as diferenças dos sistemas de arquivos entre as máquinas.

Segundo Souza (2006), a transferência de dados em redes de computadores envolve normalmente transferência de dados e acesso a sistemas de dados remotos (com a mesma interface usada nos dados locais). O *FTP* é baseado no *TCP*, mas é anterior à pilha de protocolos *TCP/IP*, sendo posteriormente adaptado para o *TCP/IP*. É o padrão da pilha *TCP/IP* para transferir dados, é um protocolo genérico independente de *hardware* e do sistema operativo que transfere dados por livre arbítrio, tendo em conta restrições de acesso e propriedades dos dados.

2.5.1 História do FTP

Segundo Ancorador (2010), a necessidade de transferirem-se arquivos entre as diversas redes de computadores que deram origem a internet fez com que surgisse o *FTP* no início dos anos de 1970, mais precisamente em 1971 no *MIT*, e foi melhorado e teve inovações importantes acrescidas a ele em julho de 1973 de forma a ser usado com o sistema operacional *Unix*. Sua simplicidade de uso e a facilidade de implementação e de manutenção o fizeram ser utilizado até os dias atuais.

Segundo Tittel (2002), o *FTP* originalmente fazia parte dos protocolos *ARPANET* e era usado antes de o *IP* e o *TCP* serem desenvolvidos. Quando o conjunto de protocolos IP foi lançado, o *FTP* foi adaptado para operar com ela. No início da internet, antes do surgimento dos elementos gráficos, aproximadamente um terço de todo o tráfego da internet baseava-se no *FTP*.

Segundo Comer (2006), a transferência de arquivos está entre as aplicações *TCP/IP* mais utilizadas e ainda considera uma quantidade significativa de tráfego na Internet.

2.5.2 Recursos do FTP em geral

Segundo Tittel (2002), as especificações do protocolo *FTP* do *IP* fomecem a capacidade de transferir arquivos de qualquer tipo entre sistemas operacionais diferentes. Se o usuário que estiver iniciando a transferência não tiver uma conta no

sistema de destino, o *FTP* suporta conexões anônimas e a capacidade de controlar as ações do usuário anônimo.

Segundo Comer (2006), dado um protocolo de transporte ponto a ponto confiável como o *TCP*, a tranferência de arquivos pode parecer simples. Entretanto, como as sessões anteriores enfatizaram, os detalhes de autorização, nomeação e representação entre máquinas heterogêneas tornaram o protocolo complexo. Além disso, o *FTP* oferece muitos recursos além da função de transferência propriamente dita. Embora o *FTP* seja projetado para ser usado por programas, a maioria das importações também fornece uma interface interativa que permite aos humanos interagirem com servidores remotos. O *FTP* permite que o cliente espeficique o tipo e a representação dos dados armazenados. Por exemplo, o usuário pode especificar se um arquivo contém texto ou dados binários e se os arquivos de texto usam os conjuntos de caracteres *ASCII* ou *EBCDIC*. O *FTP* exige que os clientes se autorizem enviando o nome de usuário e uma senha para o servidor antes de requisitar transferência de arquivo. O servidor recusa acesso a clientes que não fornecerem um usuário e senha válidos.

2.5.3 Funcionalidades do protocolo FTP

Segundo Tittel (2002), o *FTP* opera como um modelo cliente-servidor. O sistema que aceita arquivos executa um serviço *FTP* e a máquina que envia e recupera arquivos executa um *software* de cliente *FTP*.

Conforme Vita (2005), durante uma sessão *FTP*, existirão duas conexões separadas no nível de transporte, uma entre os interpretadores do protocolo e outra entre os processos de transferência de dados. A primeira e a conexão de controle, que e bidirecional e baseada no protocolo *telnet*, utilizada para comandos relativos ao controle da transferência. A segunda e a conexão de dados, também bidirecional e é por ela que os dados são transferidos efetivamente. As duas conexões operam sobre o protocolo de transporte *TCP*.

A conexão de dados pode ser estabelecida entre um cliente e um servidor ou entre dois servidores diferentes, podendo a mesma realizar as diversas maneiras de operar os arquivos e diretórios remotamente, sempre por envios e respostas de

comando pré estabeledos nas regras *FTP* sendo todos em forma de *string*, conforme se pode entender melhor nas figuras 9.

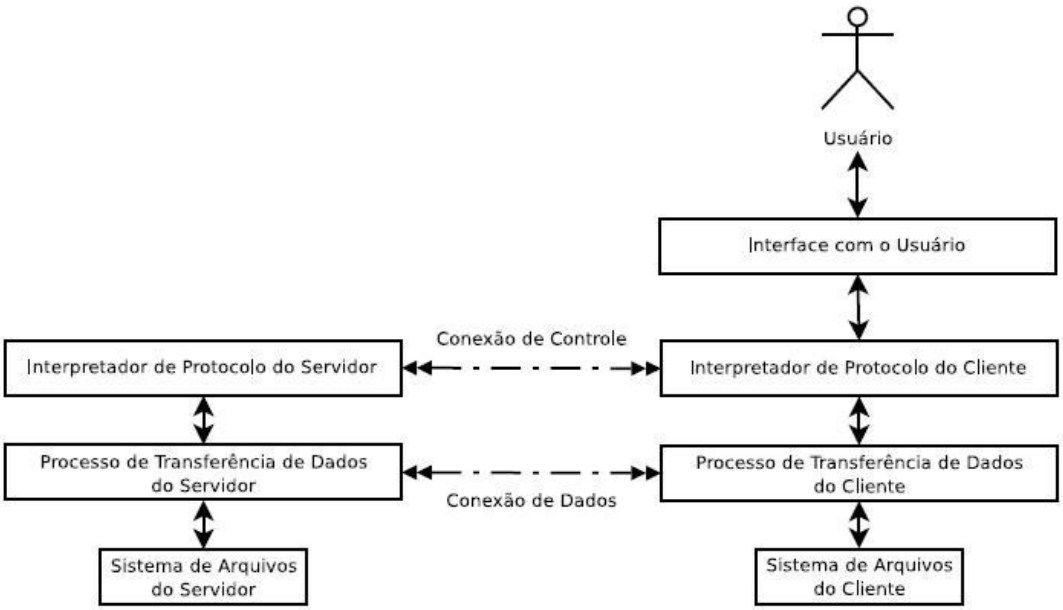

Figura 9: Entidades de uma sessão de *FTP* usual

Fonte: (VITA, 2005)

Na figura 10 que exibe a *shell* com as ações do usuário.

```
[mhoag@research mhoag]$ ftp ftp.blissranch.com
Connected to ftp.blissranch.com.
220 Service Ready for new User
Name (ftp.blissranch.com:root): mhoag
331 Password Needed for Login
Password:
230 User mhoag Logged in Successfully
Remote system type is NETWARE.
ftp> put motif
local: motif remote: motif
227 Entering Passive Mode (172,16,3,4,4,98)
150 Opening data connection
226 Transfer Complete
196965 bytes sent in 0.595 secs (3.2e+02 Kbytes/sec)
ftp>
```

Figura 10: Console de saída de um cliente *FTP*

Fonte: (TITTEL, 2002)

Segundo Tittel (2002), para evitar confusão entre essas duas operações de transferência de controle e de arquivo, atribui-se a cada uma delas um número

diferente de porta de protocolo *IP*. A porta 21 é a porta de protocolo *IP* usada para a conexão de controle *FTP* e a porta 20 é usada para transferência de arquivo.

3 MATERIAIS E MÉTODOS

O presente estudo desenvolveu um serviço de *backup* via protocolo *FTP* que necessita de dispositivos como um computador com o sistema operacional *Ubuntu Linux* versão 9.10 instalado contendo o pacote *php-cli* versão 5.3.1 junto ao pacote *PEAR Daemon* para poder rodar a aplicação como serviço, o mesmo se comportando como cliente, com processador *Pentium 4 HT 2.8 gigahertz* da *Intel Inside* com 2048 *megabytes* de memória RAM e HD (disco rígido) de 1 *terabyte,* dois computadores para serem utilizados como servidores, sendo um deles com sistema operacional *Windows XP* e o segundo com sistema operacional *Linux Ubuntu versão 9.10,* ambos com a configuração de processador *Core 2 duo 2,1 gigahertz* da *Intel Inside* com 4096 *megabytes* de memória RAM e HD (disco rígido) de 500 *gigabytes,* estando na rede *WAN* e com servidor FTP instalado e configurado com as contas de usuário do *FTP* pré estabelecidas no arquivo de configuração do serviço junto a arquivos e diretórios sortidos, para que se possa realizar os testes.

Através do fluxograma na figura 11 pode-se observar o ambiente de testes da aplicação, onde a “Máquina 1” é responsável por rodar o serviço de backup. Pela internet é conectado a “Máquina 2” e na “Máquina 3”, as quais possues os servidores de *FTP* pré configurados pelo operador de acordo com a explicação.

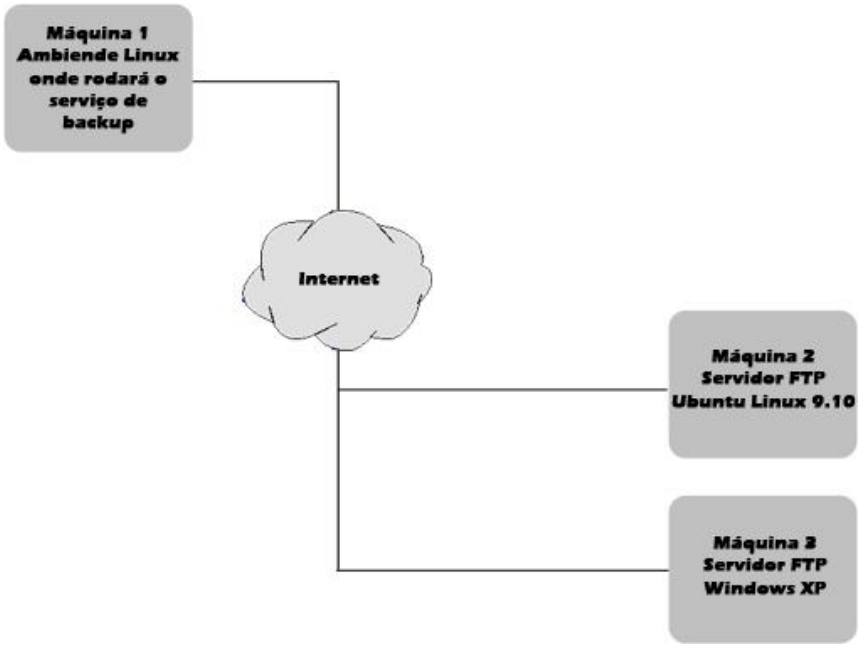

Figura 11 - Esquema do ambiente de testes

O *backup* foi incremental, pois a primeira sessão (*backup* total) baixou todos os arquivos e as próximas sessões baixarão apenas arquivos que foram alterados ou que sejam arquivos novos em relação ao último *backup* realizado, podendo ser incremental ou total.

A conexão com a internet utilizada foi de 3 *megabits* por segundo, sendo ela realizada via cabo, do tipo dedicada e com fluxo síncrono de transmissão de dados, com tempo de resposta aos servidores de aproximadamente 20 milésimos de segundo.

A aplicação desenvolvida foi na linguagem *PHP* em modo de linha de comando (*CLI*), rodando como serviço que pode ser referenciado por *daemon* em ambiente *Linux,* e teve a finalidade de levar facilidade no desenvolvimento, proporcionando melhor manutenção no futuro e simplicidade na operação, já que a mesma traz ao operador o padrão de operação com as diversas aplcações e serviços dos sistemas operacionais da família *Linux*.

4 RESULTADOS OBTIDOS

No presente estudo foi levantado a viabilidade de usar a linguagem *PHP* em uma aplicação de *backup* remoto via *FTP* sendo a mesma multiplataforma, rodando como serviço e seguindo os padrões do mesmo em plataforma *Linux Ubuntu* versão 9.10.

Inicialmente pode-se observar na figura 12 o arquivo de configuração que é processado pela aplicação em cada início ou "*start*" da mesma, o qual está localizado no caminho "/etc/ftpbackup".

```
//// CONFIGURAÇOES ////

// caminho onde será armazenado os logs
logPath = /var/log/ftpbackup
backupPath = /home/bruno/backups/%s


//// ENTRADAS DE BACKUP ////

// midukan.com.br
host[1] = midukan.com.br
path[1] = /
user[1] = midukan
pass[1] = teste*pw

// maisdametade.com.br
host[2] = maisdametade.com.br
path[2] = /teste/
user[2] = maisdametade
pass[2] = teste*pw
```

Figura 12 - Arquivo de configuração da aplicação

É através deste que a aplicação obtem informações de suas procedências, as quais podem ser alteradas pelo usuário do sistema operacional.

Na figura 13, pode-se observar o momento em que a aplicação é configurada para operar como serviço, rodando como *daemon* e a criação da entrada no arquivo de inicialização do sistema operacional sendo este trecho de código necessário apenas na primeira inicialização da aplicação.

```
#!/usr/bin/php -q
<?php
// a cima, seleciona-se a shell em que rodará o script, no caso a do PHP

// inclui o pacote PEAR Daemon para que se possa executar o php como serviço
require_once "System/Daemon.php";

System_Daemon::setOption("appName", "ftpbackup");
System_Daemon::setOption("authorName", "Bruno C. Woelke");
System_Daemon::setOption("authorEmail", "bruno@midukan.com.br");
System_Daemon::setOption("appDescription", "FTP Backup");
echo System_Daemon::writeAutoRun();
```

Figura 13 - Trecho de código responsável pela configuração da aplicação como serviço

Através da figura 14 pode-se observar um trecho do código da aplicação que é responsável por varrer o diretório remoto via *FTP* de modo recursivo, disparando chamadas de funções da aplicação como a de baixar arquivo encontrado ou verificar data de modificação em determinados momentos. Esta função de árvore é considerada a principal do serviço.

```
// função recursiva que varre arquivos e diretórios seguindo a hierarquia
// realiza chamada das funções passadas por parametros quando um novo diretório é acessado
// e quando um arquivo é encontrado, elá manipula o array global backup_path
function arvore($func_dir, $func_file){
        global $backup_path;
        static $nivel = 0;
        foreach(lista(".") as $item){
                processar();
                if(alteradir($item)){
                        $nivel++;
                        $backup_path[] = $item;
                        $func_dir($item);
                        arvore($func_dir, $func_file);
                }
                else
                        $func_file($item);
        }
        if($nivel) {
                array_pop($backup_path);
                $nivel--;
                alteradir("..");
        }
}
```

Figura 14 - Trecho do código responsável por varrer os diretórios

Usando o comando “/etc/init.d/ftpbackup start” inícia-se a operação do serviço de *backup* via *FTP,* como pode-se observar na figura 15 a seguir. A partir deste momento a aplicação rodará em background e para que suas atividades sejam observadas e analizadas, a mesma gerará registros de atividade em um arquivo pré estabelecido no arquivo de configuração, no caso será o caminho “/var/log/ftpbackup.log”.

```
root@ubuntu:/home/bruno# /etc/init.d/ftpbackup start
 * Iniciando o serviço FTP Backup (ftpbackup)                    [ OK ]
root@ubuntu:/home/bruno#
```

Figura 15 - Linha de comando no momento da inicialização do serviço

Pode-se observar em seguida, através das ilustrações dos registros das atividades da aplicação onde foi simulado o trabalho do serviço para dois destinos diferentes, com alguns arquivos e diretórios em árvore de maneira sortida, sendo o primeiro destino o sistema operacional *Linux Ubuntu* versão 9.10 e o segundo destino o sistema operacional *Windows XP*.

Observando estes registros de atividades gerados pela aplicação na figura 16, vê-se que em poucos segundos foi possível varrer e copiar arquivos e diretórios remotos, no âmbito da internet, para um computador local simplesmente conectando em diferentes servidores usando diferentes contas de *FTP* em caminhos de diretórios raizes específicos, como mencionado anteriormente.

Pode ainda criar utilizar a aplicação em mais de uma raiz para um mesmo destino, bastando apenas criar mais entradas no arquivo de configuração da aplicação.

```
[17-11-2010 22:22:53] [LOCAL] Iniciado o serviço FTP Backup

[17-11-2010 22:22:53] [LOCAL] Iniciado o backup do host midukan.com.br no diretório /
[17-11-2010 22:22:53] [REMOTO] Encontrado diretório /diretorio_teste_1/
[17-11-2010 22:22:53] [LOCAL] Criando diretório /diretorio_teste_1/
[17-11-2010 22:22:54] [REMOTO] Acessando diretório /diretorio_teste_1/
[17-11-2010 22:22:54] [REMOTO] Encontrado arquivo /diretorio_teste_1/arquivo_teste_1.jpg [49.7 kbs]
[17-11-2010 22:22:55] [LOCAL] Baixando cópia do arquivo /diretorio_teste_1/arquivo_teste_1.jpg [0.39 s]
[17-11-2010 22:22:55] [REMOTO] Encontrado arquivo /diretorio_teste_1/arquivo_teste_2.jpg [1018 bytes]
[17-11-2010 22:22:55] [LOCAL] Baixando cópia do arquivo /diretorio_teste_1/arquivo_teste_2.jpg [0.02 s]
[17-11-2010 22:22:55] [REMOTO] Encontrado arquivo /diretorio_teste_1/arquivo_teste_3.jpg [106.1 kbs]
[17-11-2010 22:22:56] [LOCAL] Baixando cópia do arquivo /diretorio_teste_1/arquivo_teste_3.jpg [1 s]
[17-11-2010 22:22:56] [REMOTO] Encontrado diretório /diretorio_teste_2/
[17-11-2010 22:22:56] [LOCAL] Criando diretório /diretorio_teste_2/
[17-11-2010 22:22:56] [REMOTO] Encontrado arquivo /diretorio_teste_2/arquivo_teste_1.jpg [65.1 kbs]
[17-11-2010 22:22:57] [LOCAL] Baixando cópia do arquivo /diretorio_teste_2/arquivo_teste_1.jpg [0.51 s]
[17-11-2010 22:22:57] [REMOTO] Encontrado arquivo /diretorio_teste_2/arquivo_teste_2.jpg [488.6 kbs]
[17-11-2010 22:23:01] [LOCAL] Baixando cópia do arquivo /diretorio_teste_2/arquivo_teste_2.jpg [2.94 s]
[17-11-2010 22:22:01] [LOCAL] Terminado o backup do host midukan.com.br

[17-11-2010 22:22:01] [LOCAL] Iniciado o backup do host maisdametade.com.br no diretório /
[17-11-2010 22:22:01] [REMOTO] Encontrado arquivo /teste/teste.exe [0.23 kbs]
[17-11-2010 22:22:02] [LOCAL] Baixando cópia do arquivo /teste/teste.exe [0.15 s]
[17-11-2010 22:22:02] [REMOTO] Encontrado diretório /teste/dir1/
[17-11-2010 22:22:02] [LOCAL] Criando diretório /teste/dir1/
[17-11-2010 22:22:02] [REMOTO] Acessando diretório /teste/dir1/
[17-11-2010 22:22:02] [REMOTO] Encontrado arquivo /teste/dir1/arquivo1.exe [1.2 mbs]
[17-11-2010 22:22:07] [LOCAL] Baixando cópia do arquivo /teste/dir1/arquivo1.exe [5.7 s]
[17-11-2010 22:22:07] [REMOTO] Encontrado diretório /teste/dir2/
[17-11-2010 22:22:07] [LOCAL] Criando diretório /teste/dir2/
[17-11-2010 22:22:07] [REMOTO] Acessando diretório /teste/dir2/
[17-11-2010 22:22:07] [REMOTO] Encontrado arquivo /teste/dir2/arquivo2.exe [502.51 kbs]
[17-11-2010 22:22:10] [LOCAL] Baixando cópia do arquivo /teste/dir2/arquivo2.exe [2.78 s]
[17-11-2010 22:22:10] [REMOTO] Encontrado diretório /teste/dir2/dir3/
[17-11-2010 22:22:10] [LOCAL] Criando diretório /teste/dir2/dir3/
[17-11-2010 22:22:11] [REMOTO] Acessando diretório /teste/dir2/dir3/
[17-11-2010 22:22:11] [REMOTO] Encontrado arquivo /teste/dir2/dir3/arquivo3.jpg [184.53 kbs]
[17-11-2010 22:22:12] [LOCAL] Baixando cópia do arquivo /teste/dir2/dir3/arquivo3.jpg [1.09 s]
[17-11-2010 22:22:12] [LOCAL] Terminado o backup do host midukan.com.br

[17-11-2010 22:22:12] [DEBUG] Finalizado o serviço FTP Backup
```

Figura 16 – Registros da varredura que realiza o *backup* total

Em seguida, é possível observar a segunda interação da aplicação nos servidores *FTP*, a qual realiza o processo de backup incremental que baixou apenas

os arquivos com data de modificação diferentes dos arquivos que foram baixados pela última vez na figura 17.

```
[17-11-2010 22:22:12] [DEBUG] Finalizado o serviço FTP Backup

[17-11-2010 22:23:09] [LOCAL] Iniciado o serviço FTP Backup

[17-11-2010 22:23:09] [LOCAL] Iniciado o backup do host midukan.com.br no diretório /
[17-11-2010 22:23:09] [REMOTO] Encontrado diretório /diretorio_teste_1/
[17-11-2010 22:23:09] [LOCAL] Diretório não criado, diretório /diretorio_teste_1/ já existe
[17-11-2010 22:23:10] [REMOTO] Acessando diretório /diretorio_teste_1/
[17-11-2010 22:23:10] [REMOTO] Encontrado arquivo /diretorio_teste_1/arquivo_teste_1.jpg [49.7 kbs]
[17-11-2010 22:23:10] [LOCAL] Arquivo existe porém diferente, baixando cópia do arquivo /diretorio_teste_1/arquivo_teste_1.jpg [0.44 s]
[17-11-2010 22:23:11] [REMOTO] Encontrado arquivo /diretorio_teste_1/arquivo_teste_2.jpg [1022 bytes]
[17-11-2010 22:23:11] [LOCAL] Arquivo existe porém diferente, baixando cópia do arquivo /diretorio_teste_1/arquivo_teste_2.jpg [0.02 s]
[17-11-2010 22:23:11] [REMOTO] Encontrado arquivo /diretorio_teste_1/arquivo_teste_3.jpg [106.1 kbs]
[17-11-2010 22:23:11] [LOCAL] Arquivo /diretorio_teste_1/arquivo_teste_3.jpg existe e não foi modificado
[17-11-2010 22:23:11] [REMOTO] Encontrado diretório /diretorio_teste_2/
[17-11-2010 22:23:11] [LOCAL] Diretório não criado, diretório /diretorio_teste_2/ já existe
[17-11-2010 22:23:12] [REMOTO] Encontrado arquivo /diretorio_teste_2/arquivo_teste_1.jpg [65.16 kbs]
[17-11-2010 22:23:12] [LOCAL] Arquivo existe porém diferente, baixando cópia do arquivo /diretorio_teste_1/arquivo_teste_1.jpg [0.59 s]
[17-11-2010 22:23:12] [REMOTO] Encontrado arquivo /diretorio_teste_2/arquivo_teste_2.jpg [488.6 kbs]
[17-11-2010 22:23:13] [LOCAL] Arquivo /diretorio_teste_2/arquivo_teste_2.jpg existe e não foi modificado
[17-11-2010 22:23:13] [LOCAL] Terminado o backup do host midukan.com.br

[17-11-2010 22:23:13] [LOCAL] Iniciado o backup do host maisdametade.com.br no diretório /
[17-11-2010 22:23:13] [REMOTO] Encontrado arquivo /teste/teste.exe [0.58 kbs]
[17-11-2010 22:23:14] [LOCAL] Arquivo existe porém diferente, baixando cópia do arquivo /teste/teste.exe [0.33 s]
[17-11-2010 22:23:14] [REMOTO] Encontrado diretório /teste/dir1/
[17-11-2010 22:23:14] [LOCAL] Diretório não criado, /teste/dir1/ já existe
[17-11-2010 22:23:15] [REMOTO] Acessando diretório /teste/dir1/
[17-11-2010 22:23:15] [REMOTO] Encontrado arquivo /teste/dir1/arquivo1.exe [1.2 mbs]
[17-11-2010 22:23:15] [LOCAL] Arquivo /teste/dir1/arquivo1.exe existe existe e não foi modificado
[17-11-2010 22:23:15] [REMOTO] Encontrado diretório /teste/dir2/
[17-11-2010 22:23:15] [LOCAL] Diretório não criado, /teste/dir2/ já existe
[17-11-2010 22:23:16] [REMOTO] Acessando diretório /teste/dir2/
[17-11-2010 22:23:17] [REMOTO] Encontrado arquivo /teste/dir2/arquivo2.exe [502.51 kbs]
[17-11-2010 22:23:17] [LOCAL] Arquivo /teste/dir2/arquivo2.exe existe e não foi modificado
[17-11-2010 22:23:17] [REMOTO] Encontrado diretório /teste/dir2/dir3/
[17-11-2010 22:23:17] [LOCAL] Diretório não criado, /teste/dir2/dir3/ já existe
[17-11-2010 22:23:17] [REMOTO] Acessando diretório /teste/dir2/dir3/
[17-11-2010 22:23:18] [REMOTO] Encontrado arquivo /teste/dir2/dir3/arquivo3_renomeado.jpg [184.53 kbs]
[17-11-2010 22:23:20] [LOCAL] Baixando cópia do arquivo /teste/dir2/dir3/arquivo3_renomeado.jpg [0.92 s]
[17-11-2010 22:23:20] [REMOTO] Encontrado diretório /teste/dir2/dir4/
[17-11-2010 22:23:20] [LOCAL] Criando diretório /teste/dir2/dir4/
[17-11-2010 22:23:20] [LOCAL] Terminado o backup do host midukan.com.br

[17-11-2010 22:22:20] [LOCAL] Iniciado o serviço FTP Backup
```

Figura 17 - Registros da varredura que realiza o *backup* incremental

Pode-se assim concluir que os arquivos e diretórios foram copiados com êxito e estão devidamente em seus caminhos na árvore de diretórios e arquivos conforme estavam nos servidores de *FTP*. Assim, de acordo com os registros das figuras 16 e 17, comprova-se que as cópias de *backup* estão de acordo com as regras de *backup* incremental, a qual é a metedologia de *backup* apropriada e selecionada para este estudo.

Levando em consideração que alguns arquivos foram modificados para que o procedimento de verificação dos arquivos modificados nos servidores remotos do serviço fosse testado e que os mesmos fossem copiados, um rápida alteração no conteúdo do mesmo foi realizada manualmente, simulando um alteração de um usuário comum.

O tempo necessário para realizar todo o processo de *backup* junto ao tamanho dos arquivos foi obtido em um teste realizado com 2 sessões sendo o *backup* total

na primeira varredura com o total de 19 segundos, contendo 5 diretórios e 9 arquivos com o total de 4.533 *kbytes* e posteriormente o incremental na segunda varredura com o total de 11 segundos criando 1 diretório extra e copiando 6 arquivos modificados ou novos com o total de 399 *kbytes*, somando um total de 30 segundos e 4.933 *kbytes* nas 2 sessões.

5 CONCLUSÃO

Dos resultados pode-se concluir que a linguagem *PHP* obteve bom desempenho, efetuando os *backups* com velocidade consideravelmente boa de acordo os as hipóteses, que citava o *FTP* como obstáculo na performance da linguagem em si, devido ao tempo de resposta na rede *WAN* junto ao grande número de requisições e respostas necessárias para interagir com cada arquivo, que por fim, podem possuir tamanhos que possam levar vários segundos para serem copiados, porém, isso acaba sendo reduzido em muito quando a conexão é feita pela rede *LAN*, que apresenta quase em todos casos velocidade de transmissão de dados maiores que a da *WAN,* podendo chegar por exemplo a 1000 *megabits* por segundos, acredita assim que mesmo com esta velocidade ampliada, o *PHP* ainda continua apresentando alto desempenho, porém, para ser comprovado será necessário outros testes.

Este serviço realizou com sucesso todo o processo de *backup* dos arquivos das contas de *FTP* cadastradas em um arquivo de configuração e gerou um arquivo de *log* para que as ações podessem ser analisadas.

Dentre os servidores remotos, foi possível conectar e realizar as cópias de *backup* com sucesso, dentro da necessidade esperada em relação ao quesito multi-plataforma.

Conclui-se neste projeto de pesquisa que a empregação da linguagem *PHP* no tipo de aplicação estudada foi vista como viável, e que a mesma não apresentou problemas de performance, onde o ideal para este propósito é utilizar linguagens de baixo nível e/ou compiladas. Apesar do foco principal da pesquisa ter sido comprovado, pode-se também observar as vantagens de se utilizar a linguagem *PHP* neste determiando caso, os quais são a rápida curva de aprendizagem, velocidade de desenvolveimento e a facilidadede de manutenção, todos trazendo reduções de custos, tornando em muitos dos casos viável a sua implementação para este tipo de serviço, principalmente em pequenas empresas.

REFERÊNCIAS

ADONAI, C. **PHP & MySQL – Guia Introdutório.** 3ª Ed. São Paulo: Sabbag, 2005.

ALECRIM, E. **A história do Tux, o pinguim-símbolo do Linux,** 2004. Disponível em: <http://www.infowester.com/tux.php> Acessado em: 5 jun. 2010.

ALIENAÇÃO, I. **Assista Online: Shell LINUX**. Disponível em: <http://informaticaealienacao.blogspot.com/2010/04/assista-online-shell-linux.html> Acessado em: 6 jun. 2010.

ALVAREZ, M. A. **O que é PHP,** 2004. Disponível em: <http://www.criarweb.com/artigos/202.php> Acessado em: 4 jun. 2010.

ANCORADOR. **FTP – Como surgiu e é utilizado até hoje,** 2010. Disponível em: <http://www.ancorador.com.br/internet/ftp-como-surgiu-por-que-e-utilizado-ate-hoje> Acessado em: 3 jun. 2010.

ARAÚJO, H. **A História do PHP,** 2006. Disponível em: <http://www.htmlstaff.org/ver.php?id=327> Acessado em: 5 jun. 2010.

BARROS, G. **Backup em Fita-DAT com TAR,** 2009. Disponível em: <http://gmbarros.wordpress.com/2009/01/30/backup-em-fita-dat-com-tar> Acessado em: 5 jun. 2010.

BUYENS, J. **Aprendendo MySQL & PHP.** Tradução Lavio Pareschi. São Paulo: Makron, 2002.

CATOZE, G. **Usuários no Linux,** 2008. Disponível em: <http://catoze.wordpress.com/2008/11/21/usuarios-no-linux> Acessado em: 4 jun. 2010.

COMER, D. **Interligação de Redes com TCP/IP.** Tradução Daniel Vieira. Vol. 1 São Paulo: Elsevier, 2006.

CONECTIVA. **Guia do Usuário do Conectiva Linux**. Disponível em: <http://www.conectiva.com/doc/livros/online/9.0/usuario/conceitos.html> Acessado em: 5 jun. 2010.

DANASH, A. **Dominando o Linux "A Bíblia".** Tradução João Eduardo Nóbrega Tortello. São Paulo: Makron, 2000.

DEXTRA. **PHP, a língua franca da Web,** 2010. Disponível em: <http://www.dextra.com.br/empresa/artigos/php.htm> Acessado em: 5 jun. 2010.

FIALHO, M. **Guia Essencial do Backup.** São Paulo: Digerati, 2007.

FILHO, F. J. S. **Conceitos básicos sobre Linux,** 2000. Disponível em: <http://equipe.nce.ufrj.br/adriano/c/apostila/gtk/html/linux.html> Acessado em: 5 jun. 2010.

GUILLET, P. **Comparação entre os sistemas Linux e Windows,** 2006. Disponível em: <http://tecnologia.uol.com.br/ultnot/2006/12/18/ult2870u228.jhtm> Acessado em: 4 jun. 2010.

LNCC. **Perguntas & Respostas**. 2010. Disponível em: <http://queroquero.lncc.br/~licht/ist/so2/Linux/servidores.backup.porque.html>. Acesso em: 28 Abr. 2010.

LYRA, M. **Segurança e Auditoria em Sistemas de Informação.** Rio de Janeiro: Ciência Moderna, 2008.

MAFFEIS, C. **Linux e suas Soluções.** São Paulo: Campus, 2010.

MENDES, G. **Segurança na Gestão de Informações Corporativas**. Rio de Janeiro, 2009. Disponível em:

<http://www.aedb.br/seget/artigos07/1351_ultima%20versao.pdf>. Acesso em: 02 Abr. 2010.

MORIMOTO, C. **Entendendo e Dominando o Linux.** 3 ª Ed. São Paulo: Digerati, 2004.

NCE. **Conceitos básicos sobre Linux**. Disponível em: <http://equipe.nce.ufrj.br/adriano/programacao/privado3/basicos.ppt> Acessado em: 5 jun. 2010.

OLIVEIRA, N. A. C. **Sistemas Operativos**. Disponível em: <http://www.ensinodigital.com/sistema_operativo_linux.pdf> Acessado em: 6 jun. 2010.

PARIHAR, M. **TCP/IP – A Bíblia.** Tradução Daniel Vieira. Vol. 1 São Paulo: Elsevier, 2002.

PHP Group. **PHP: hypertext preprocessor.**, 2010. Disponível em: <http://www.php.net/manual/pt_BR/intro-whatcando.php>. Acesso em: 02 mai. 2010.

PHP Group. **PHP: hypertext preprocessor.**, 2010. Disponível em: <http://www.php.net/>. Acesso em: 02 mai. 2010.

PINHEIRO, J. M. S. **Políticas de Backup Coorporativo,** 2006. Disponível em: <http://www.malima.com.br/article_read.asp?id=334> Acessado em: 4 jun. 2010.

SANTOS, C. **Firewall Linux IPTABLES**. Rio de Janeiro, 2004. Disponível em: <http://www.clubedasredes.eti.br/rede0023.htm>. Acesso em: 13 mai. 2010.

SILVA, J. E. **História do GNU/Linux: 1965 assim tudo começou!,** 2006. Disponível em: <http://www.vivaolinux.com.br/artigo/Historia-do-GNU-Linux-1965-assim-tudo-comecou> Acessado em: 5 jun. 2010.

SOARES, W. **PHP 5 – Conceitos, programação e integração com banco de dados.** 4 ª Ed. Rio de Janeiro: Érica, 2007.

SOUZA, S. **FTP: O que é?,** 2006. Disponível em: <http://www.htmlstaff.org/ver.php?id=982> Acessado em: 3 jun. 2010.

TAYLOR, D; ARMSTRONG, J. **Aprenda em 24 horas UNIX.** Tradução Fremen Servios Especiais. Rio de Janeiro: Campus, 2000.

TITTEL, E. **Redes de Computadores.** São Paulo: Artmed, 2002.

VITA, J. P. R. **Funcionamento do Protocolo FTP,** 2005. Disponível em: <http://nschechter.net/ftp.pdf> Acessado em: 4 jun. 2010.