Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

ANDRÉ DA SILVA RODRIGUES

SISTEMA GERENCIADOR DE USUÁRIOS, GRUPOS, DIRETÓRIOS E PERMISSÕES PARA SERVIDOR DE ARQUIVO LINUX

Assis
2014

ANDRÉ DA SILVA RODRIGUES

SISTEMA GERENCIADOR DE USUÁRIOS, GRUPOS, DIRETÓRIOS E PERMISSÕES PARA SERVIDOR DE ARQUIVO LINUX

Trabalho de Conclusão de Curso apresentado ao
Instituto Municipal de Ensino Superior de Assis,
como requisito do Curso de Graduação.

Orientador: Prof. Esp. Fernando César de Lima
Área de Concentração: Informática

Assis
2014

FICHA CATALOGRÁFICA

RODRIGUES, André da Silva

Sistema Gerenciador de Usuários, Grupos, Diretórios e Permissões para Servidor de Arquivo Linux / André da Silva Rodrigues. Fundação Educacional do Município de Assis – FEMA – ASSIS, 2014.

57p.

Orientador: Fernando César de Lima.

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA.

1. Servidor **2.** Linux. **3.** SSH. **4.** Shell Script. **5.** Java. **6.** Acesso Remoto.

CDD: 001.6

Biblioteca FEMA.

SISTEMA GERENCIADOR DE USUÁRIOS, GRUPOS, DIRETÓRIOS E PERMISSÕES PARA SERVIDOR DE ARQUIVO LINUX

ANDRÉ DA SILVA RODRIGUES

Trabalho de Conclusão de Curso apresentado ao
Instituto Municipal de Ensino Superior de Assis,
como requisito do Curso de Graduação, analisado
pela seguinte comissão examinadora:

Orientador: Prof. Esp. Fernando Cesar de Lima
Analisador (1): Prof. Esp. Guilherme de Cleva Farto

Assis
2014

DEDICATÓRIA

Dedico este trabalho à minha família, amigos e todas as pessoas que tiveram paciência e me apoiaram em todos os momentos para que este trabalho fosse realizado.

AGRADECIMENTOS

Agradeço toda minha família, meus pais José Irineu Rodrigues e Luzia Cecilia da Silva Rodrigues, meu irmão Rafael da Silva Rodrigues, que sempre apoiam e dão todo o suporte para que meus sonhos se tornem realidade.

Ao professor, Fernando César de Lima pela orientação e pelo constante estimulo e ideias transmitidos durante o trabalho.

A todos os professores que contribuíram de forma competente para minha formação e crescimento pessoal.

Aos meus amigos de formação Vinicius de Paiva Dionizio, Ani Carolina Custódio, Emerson Carlos Ribeiro, Valdinei Aparecido de Oliveira e Iraí Corrêa Leite.

Por fim todos os colegas e pessoas que contribuíram de alguma forma para a minha formação.

“Se você quer ser bem sucedido, precisa ter dedicação total, buscar seu último limite e dar o melhor de si.”

Ayrton Senna (1960 - 1994)

RESUMO

Atualmente, torna-se cada vez mais comum instalação de servidores Linux em empresas de pequeno ou grande porte, até mesmo, em comércios que atuam em setores variados. Porém, servidores Linux geralmente são configurados por intermédio de comandos de textos, o que torna o processo de manutenção de pequenas rotinas como adicionar usuários, grupos e gerenciar permissões de arquivos e diretórios demorada, e feita somente por quem tem conhecimento avançado de Linux.

Este trabalho apresenta o processo do desenvolvimento de um sistema onde o administrador de servidor de arquivo Linux através de um sistema web pode administrar e gerenciar usuários, grupos e permissões de um ou mais servidor dentro de uma rede interna. Além disso, apresenta como a linguagem de programação Java pode acessar e enviar comando ao servidor via SSH, e o conceito de Shell Script usado para criar um comando para adicionar usuário de forma paralela ao modo usado normalmente em Linux.

Palavras-chave: Servidor; Linux; SSH; Shell Script; Java; Acesso Remoto.

ABSTRACT

Currently, installation of Linux servers in small or large companies has become much more common, even at stores that work with assorted sectors. However, Linux servers normally are the configured through text command, which makes the process of maintaining simple routines as add users, groups and manage permissions of files and directories time consuming difficult to manage, what means that only those who have advanced knowledge of Linux would be able to do it.

This study presents the process of developing a system where the administrator of Linux file server through a web system can administer and manage users, groups and permissions from one or more servers into an internal network. In addition, it presents how the Java programming language can access and send command to the server via SSH, and the Shell Script's concept used to create a command to add user in a parallel way to the mode usually used on Linux.

Keywords: Server; Linux; SSH; Shell Script; Java; Remote Access.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

SO	Sistema Operacional
JVM	Java Virtual Machine
SSH	Secure Shell
BASH	Bourne Again Shell
IDE	Integrated Development Environment
IBM	International Business Machines
SQL	Structure Query Language
DER	Diagrama de Entidade e Relacionamento
Usenet	Unix User Network

SUMÁRIO

1. INTRODUÇÃO

O servidor de arquivos tem a função de centralizar o armazenamento e compartilhamento de arquivos entre usuários de uma empresa ou residência agilizando o trabalho do usuário ou funcionário, ao invés de criar compartilhamento de arquivos e diretórios na própria máquina, o servidor de arquivos disponibiliza um diretório para esse usuário armazenar e compartilhar seus documentos.

Atualmente empresas de grande, médio e até de pequeno porte utilizam servidores de arquivos com o SO Linux em suas redes devido ao seu baixo custo, segurança e estabilidade. A empresa não precisa comprar licença, é livre de vírus e conhecido como um sistema capaz de funcionar anos sem falhas.

O profissional que administra servidor de arquivo Linux para adicionar e gerenciar usuários e grupos no sistema precisa ter conhecimento em comandos de texto Linux, igualmente para criação de diretório e administração de permissões de arquivos e diretórios. Essas configurações são feitas por linhas de comandos, gastam tempo e não tem a interface amigável, mas sim um terminal com fundo preto onde são executados os comandos.

Visando diminuir o tempo gasto e a dificuldade do manuseio de um servidor Linux, esse trabalho tem a finalidade de gerar uma ferramenta na qual o administrador de rede poderá cadastrar e fazer a gerência do servidor de arquivo da empresa no sistema através de uma interface web.

1.1 OBJETIVO

Esse trabalho de conclusão de curso objetiva o desenvolvimento de um software que possibilite o administrador de servidor de arquivos que utilize o SO Linux, adicionar e gerenciar usuários e diretórios, administrar permissões de acesso a arquivos e diretórios através de um aplicativo remoto.

1.2 JUSTIFICATIVA

A maioria das empresas possuem computadores como ferramentas de trabalho, quando surge a necessidade de obter mais computadores, para centralização e organização de arquivos e documentos dos funcionários, uma opção é adquirir um servidor de arquivo para essa tarefa, este servidor pode conter um SO Linux ou Microsoft Windows.

Se o servidor utilizar o SO Linux, o sistema resultante deste trabalho de conclusão de curso auxilia de forma eficiente o profissional responsável por gerenciar usuários, grupos, diretórios e permissões deste servidor, através de uma interface web. O profissional fica isento de gerar e executar linhas de comando para estas tarefas, consideradas simples para um especialista em SO Linux.

1.3 MOTIVAÇÃO

Quando uma empresa opta por ter um servidor Linux, geralmente são instalados somente terminais que executam comandos de textos e sem nenhuma interface gráfica. Então a motivação para fazer um trabalho de desenvolvimento de sistema, onde o administrador de um servidor Linux, por meio de um aplicativo web acessa remotamente o servidor para adicionar e gerenciar usuários e grupos no servidor, crie, move e remove diretórios, e gerencie todas as permissões de arquivos e diretórios desse servidor.

1.4 METODOLOGIA DE PEQUISA

Para o desenvolvimento do sistema gerenciador de usuários, grupos diretórios e permissões para servidor de arquivo Linux, foram feitos estudos de casos focados em uma empresa que presta serviço de instalação e gerenciamento de servidor Linux. Foram observados as configurações solicitadas pelos clientes para gerência de usuários, grupos, permissões de arquivos e diretórios em seus servidores Linux.

O trabalho foi dividido em 3 etapas:

- Na primeira etapa foi pesquisado como a linguagem de programação Java executa comando *shell* no terminal Linux, e como recebe retorno de erro e êxito.
- Na segunda etapa foi simulado no software *VirtualBox* uma pequena rede com servidor Linux Debian.
- Na Última etapa foi desenvolvido o sistema gerenciador de usuários, grupos, diretórios e permissões para servidor de arquivo Linux, e foram feitos testes para garantir o correto funcionamento do sistema.

1.5 ESTRUTURA DO TRABALHO

O trabalho está estruturado nos seguintes capítulos:

- ✓ **Capítulo 1 – Introdução**, apresenta os objetivos, motivações, justificativas e a metodologia de pesquisa.
- ✓ **Capítulo 2 – Estrutura do desenvolvimento do Sistema**, apresenta todas as tecnologias e ferramentas utilizados para o desenvolvimento do sistema.
- ✓ **Capítulo 3 – Sistema Operacional Linux**, apresenta todas as características do SO Linux utilizados para o desenvolvimento do sistema.
- ✓ **Capítulo 4 – Proposta do Trabalho**, apresenta todas as funcionalidades e objetivos propostos para o desenvolvimento do trabalho.
- ✓ **Capítulo 5 – Análise do Sistema**, apresenta o caso de uso, diagramas e as telas do sistema desenvolvido.
- ✓ **Capítulo 6 – Conclusão**, apresenta os resultados da pesquisa e do desenvolvimento do sistema.
- ✓ **Referências bibliográficas.**

2. ESTRUTURA DE DESENVOLVIMENTO DO SISTEMA

Neste capítulo são descritos ferramentas e tecnologias utilizadas para o desenvolvimento do sistema gerenciador de usuários, grupos, diretórios e permissões para servidor de arquivo Linux.

2.1 LINGUAGEM DE PROGRAMAÇÃO JAVA

Java é uma linguagem de programação orientada a objetos baseada em C++, financiada pela Sun Microsystems e desenvolvida por uma equipe de programadores liderada por James Gosling. A Sun pretendia desenvolver uma linguagem de programação que executassem seus programas independentemente das plataformas que seriam rodados, visando o mercado de eletrônicos. Porém os dispositivos eletrônicos não estavam se desenvolvendo tão rapidamente como a Sun imaginava, e o projeto passou por algumas dificuldades. Mas com a explosão da *Web* em 1993 a Sun viu a oportunidade de adicionar interatividade as página da *Web* e o projeto se reencontrou.

Segundo Deitel (2010, p.6):

> A Sun anunciou o Java formalmente em uma conferência do setor em maio de 1995. O Java chamou a atenção da comunidade de negócios por causa do enorme interesse na *Web*. O Java é agora utilizado para desenvolver aplicativos corporativos de grande porte, aprimorar a funcionalidade de servidores *Web* (os computadores que fornecem o conteúdo que vemos em nosso navegadores da Web), e fornecer aplicativos para dispositivos voltados para o consumo popular (como celulares, *pagers* e PDAs) e para muitos outros propósitos.

No dia 20 de Abril de 2009 a Oracle anunciou que comprou a Sun Microsystems por US$ 7 bilhões de Dólares, na época a comunidade *open source* e de desenvolvedores

Java ficaram preocupados com o futuro da linguagem. Porem há atualizações! Recentemente foi anunciada à versão Java 8 JDK (*Java Development Kit*) e continua sendo *open source*.

Java tornou-se popular rapidamente por sua portabilidade, possuí uma sintaxe simples, e por ser orientada a objetos possibilita ao programador melhor organização de seus projetos tanto no desenvolvimento, quanto na manutenção de erros do projeto. Também possui um Coletor de Lixo (*Garbage Collector*), que é um gerenciador de memória, sua função é recuperar área de memória inutilizada por programas para evitar problemas como sobrecarga de memória.

A Linguagem de Programação Java foi escolhida para esse trabalho de conclusão de curso por ser uma tecnologia forte, e por ter uma mobilidade para desenvolvimento de aplicação para qualquer plataforma ou dispositivo, é utilizado por milhões de desenvolvedores, e possui uma comunidade ativa que torna Java uma linguagem madura.

2.1.1 Funcionamento da JVM

Diferente de linguagem convencional como C++, que compila seu código fonte diretamente para linguagem nativa de máquina deixando pronta para execução do programa, Java compila seu código fonte com um arquivo de extensão ".java" através do javac, e faz verificações de erros. Se não houver erros ele gera o *Byte Code*, um arquivo de extensão ".class" que posteriormente é executado pela JVM como mostra a Figura 1.

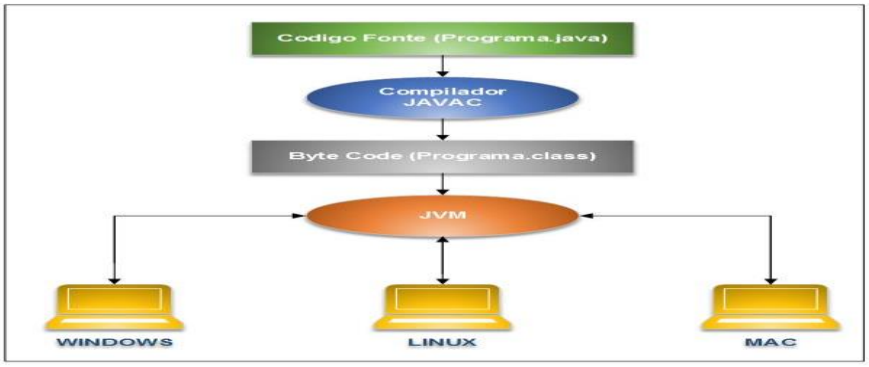

Figura 1 - Funcionamento da linguagem Java

O que torna Java uma linguagem de programação portátil é a sua capacidade de executar seu código fonte em qualquer plataforma, independentemente de seu sistema operacional, isso é possível devido a JVM, responsável por executar os programas para o usuário. Para o programa Java executar sem problema, cada tipo de plataforma ou SO tem sua própria JVM.

2.2 SSH

O SSH é o serviço que tem a função de acessar remotamente um computador ou servidor com SO Linux. Pode-se utilizar o terminal de comando *shell* ou ambiente gráfico deste computador ou servidor através de um cliente SSH. Utilizando o serviço SSH, um computador com o SO Windows pode acessar e transferir dados para uma máquina Linux. A conexão SSH é totalmente segura pois todo o tráfego de rede é criptografado pelo serviço.

Para fazer uma conexão ao ambiente *shell* do servidor Linux via SSH na máquina cliente, temos que informar qual o usuário remoto e o IP do servidor que será acessado. Como é exemplificado na Figura 2.

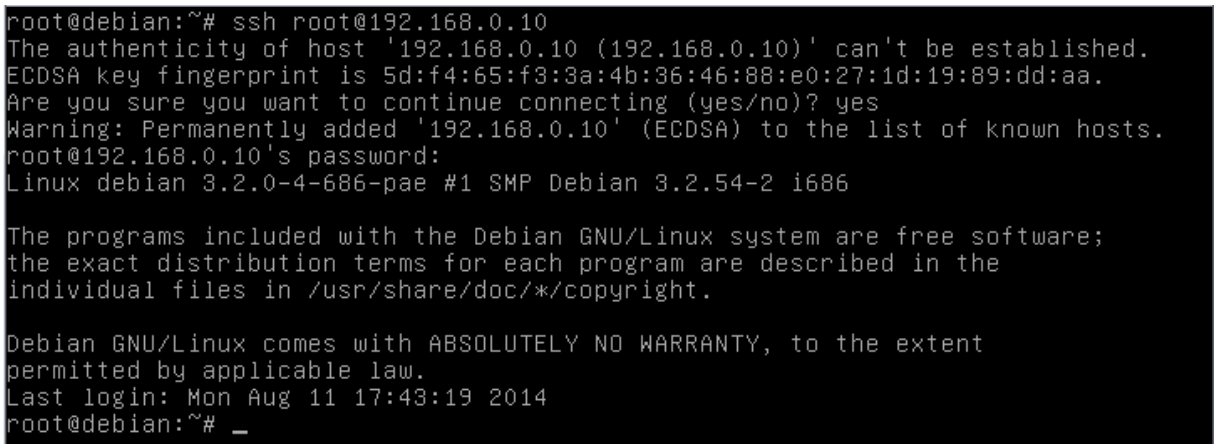

Figura 2 - Conexão remota via SSH

Para fazer a conexão de um cliente SSH com a máquina servidora 192.168.0.10 como usuário root, o comando utilizado é o "**# ssh root@192.168.0.10**", após ser executado, é mostrado na tela o *fingerprinter* e uma pergunta se o usuário deseja continuar com a conexão.

O *fingerprinter* é uma sequência de números hexadecimal que é atribuído ao servidor como identificador no momento da instalação do SSH. Quando é feita a primeira conexão pelo cliente esse identificador é armazenado localmente no diretório /home/usuário/known_hosts, no caso do *root*, em /root/known_hosts. Todas as conexões posteriores desse cliente, será verificado o *fingerprint* por questões de segurança.

2.2.1 Projeto sshj.jar

O projeto sshj.jar foi criado por Shikhar Brushan, um indiano que atualmente trabalha como engenheiro de software sênior na empresa *Etsy, inc,* localizada na cidade de Nova Iorque nos EUA.

Shikhar Brushan estudou *Jacobs University Bremen* no estado alemão de *Bremen*. O download da biblioteca pode ser baixado no *github* do autor, no *link* https://github.com/shikhar.

O projeto sshj.jar é uma biblioteca que através de um cliente SSH cria e gerencia conexão segura entre Java e servidor Linux, ele permite executar comandos remotos para o Shell BASH do Linux, e receber as linhas de respostas geradas pelo SO. A Código 1 mostra a classe de conexãoSSH implementada para este trabalho de conclusão de curso:

```java
package br.com.asrti.model.util;
import java.io.Closeable;
import java.io.IOException;
import java.security.PublicKey;
import net.schmizz.sshj.SSHClient;
import net.schmizz.sshj.transport.verification.HostKeyVerifier;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class ConexaoSSH {
	private final static Logger log =
LoggerFactory.getLogger(ConexaoSSH.class);
	private final static String root = "root";
	private static SSHClient ssh = null;

	public static SSHClient conectarSSH(String ipAddressServidor, String
senha) {
		ssh = new SSHClient();
		try {
			setupKeyVerifier(ssh);
			ssh.connect(ipAddressServidor);
			ssh.authPassword(root, senha);
		} catch (Exception e) {
			ssh = null;
			e.printStackTrace();
		}
		return ssh;
	}
	public static void fecharConexaoSSH() throws IOException {
		ssh.disconnect();
	}
	public static void fecharRecursos(Closeable resource) {
		try {
			if (resource != null) {
				resource.close();
			}
		} catch (IOException e) {
			log.error("Erro ao Fechar Recurso", e);
		}
	}
	private static void setupKeyVerifier(SSHClient ssh) {
		ssh.addHostKeyVerifier(new HostKeyVerifier() {
			@Override
			public boolean verify(String arg0, int arg1, PublicKey
arg2) {
				return true; // sem verificação
			}
		});
	}
}
```

Código 1 - Classe de ConexãoSSH

Segundo (Senaga, 2012), “o objeto da classe *SSHClient*, provê toda a infraestrutura necessária para realizar a conexão, autenticação e execução de comandos remotos via SSH através de métodos bem intuitivos”.

Na classe ConexaoSSH o método conectarSSH recebe como parâmetro as *strings* de endereço de IP e a senha do servidor, a primeira ação do método é instanciar o objeto da classe *SSHClient*.

No próximo passo o objeto SSH é passado como parâmetro para método *setupKeyVerifier*, onde é definido uma classe interna anônima que implementa a interface *HostKeyVerifier* para o método *addHostKeyVerifier*. Esse método diz ao objeto *SSHClient* para não fazer a verificação da existência do *fingerprint*, e retorna sempre a verificação como verdadeiro. Este método é necessário, pois se o sistema for executado através de uma máquina com SO Windows, ele não encontrará o diretório /home/usuário/known_hosts e dará erro na conexão remota.

Após a verificação, o objeto *SSHClient* chama o método de conexão passando a s*tring* com o IP do servidor que será acessado remotamente, em seguida chama o método *authPassword* passando as *strings* com o usuário e senhas que acessará o servidor. No caso deste sistema o usuário sempre será o *root*, que tem permissão total de acesso. Por fim, o método conectarSSH retorna o objeto *SSHClient* com a conexão efetuada para quem o solicitou.

2.3 IDE ECLIPSE

O eclipse é um IDE (*Integrated Development Enviroment*) que foi desenvolvido pela empresa IBM (*international Business Machines*), a empresa investiu cerca de 40 milhões de dólares para o desenvolvimento de um IDE de código aberto e de grande usabilidade.

O eclipse foi desenvolvido em Java, é uma plataforma extensível e suporta o desenvolvimento de diversas linguagens de programação, como, C/C++, PHP, *ColdFusion, Python, Android* e principalmente Java, basta instalar o *plug-in* para desenvolver para a linguagem desejada.

2.4 BANCO DE DADOS MYSQL

MySQL é um sistema gerenciador de bando de dados relacional de código aberto, é baseado em comandos SQL (*Structure Query Language*), e usado na maioria da aplicações gratuitas para gerir suas bases de dados.

O MySQL *Workbench* é uma ferramenta gráfica de gerenciamento do banco de dados MySQL, que possibilita ao desenvolvedor de banco de dados gerenciar todos os bancos, tabelas e relacionamentos existente no seu MySQL.

Com essa ferramenta é possível desenhar DER (Diagrama de Entidade e Relacionamento), e a partir do projeto gerar e executar os scripts para a criação do banco de dados, tabelas e relacionamentos. Ou caso tenha um banco de dados pronto, a ferramenta possibilita fazer engenharia reversa e gerar o DER desse banco de dados.

3. SISTEMA OPERACIONAL LINUX

O sistema operacional Linux é um sistema baseado em *Unix*, *open source* (com código aberto) que não necessita comprar licença para uso, começou ser desenvolvido por Linus Torvalds no ano de 1991, através de e-mails na Usenet (*Unix User Network*) anunciou que estava criando um sistema operacional de código aberto e pedia sugestões aos usuários, no mesmo ano a primeira versão foi lançada, e ao passar dos anos aprimorada (PEREIRA, 2010).

Hoje podemos encontrar divers as distribuições Linux como Ubuntu, Red Hat, Slackware, Debian, Conectiva, SuSe, Fedora, Kurumim e Mandriva. Qualquer pessoa com conhecimento avançado pode desenvolver uma distribuição Linux, o mais comum é utilizar uma distribuição já existente como ponto de partida.

3.1 DEBIAN

Debian é uma distribuição que utiliza o *kernel* do SO Linux como base e centenas de aplicações pré-empacotadas. O projeto Debian são grupos de voluntários que focam em produzir um SO *open source* e composto inteiramente de softwares livres.

Segundo GARBEE (2002), o projeto Debian foi fundado por Ian Murdock em 16 de Agosto de 1993, ele pretendia que o Debian fosse uma distribuição criada abertamente como o Linux, esse projeto posteriormente passou nas mãos de vários líderes, e atualmente é liderado por Lucas Nussbaum, que assumiu em Abril de 2013.

A distribuição Debian conta com doze versões. A Tabela 1 mostra a versão, nome e ano do lançamento de cada distribuição.

VERSÃO	NOME	ANO
7.0	Wheezy	2013
6.0	Squeeze	2011
5.0	Lenny	2009
4.0	Etch	2007
3.1	Sarge	2005
3.0	Woody	2002
2.2	Potato	2000
2.1	Slink	1999
2.0	Hamm	1998
1.3	Bo	1997
1.2	Rex	1996
1.1	Buzz	1996

Tabela 1 - Versões da distribuição Debian

A primeira versão da distribuição Debian foi lançada em 1994 com à versão 0.9, a partir de 1996 as versões começaram ganhar nomes inspirados nos personagens da

animação infantil *Toy Story*. A versão atual foi lançada em 4 de maio de 2013 é a 7.0 – *Wheezy*. O ciclo de desenvolvimento das versões passam por três fazes, que são:

- *Unstable* (instável) os pacotes mais novos ou atualizados são mandado para essa distribuição onde recebe o nome Sid.
- *Testing* (teste) os pacotes migram semi-automaticamente da instável para teste através do gerente e de alguns scripts, para os pacotes se tornarem mais estáveis, e recebem o nome.
- *Stable* (Estável) onde os pacotes estão totalmente estáveis pronto para serem lançados, e recebem o número de versão (ex: 7.0).

Atualmente a versão teste é a Jessie 8.0.

3.2 USUARIO, GRUPOS E PERMISSÕES

Neste item são mostrados comandos para criar e administrar usuários, grupos e permissões no sistema operacional Linux.

3.2.1 Usuários

Quando um usuário é adicionado no sistema automaticamente ele recebe um número identificação (UID), e também é criado um diretório home e um grupo com o nome do usuário. Todas as informações dos usuários do sistema ficam armazenados nos arquivo *passwd* no diretório (/etc/passwd).

Para adicionar um usuário no sistema, é preciso estar logado como administrador (*root*). E o comando utilizado é “**# adduser ‘opção’ ‘usuário’**”.

Algumas opções disponíveis:

- --*disable-login* “usuário”, desativa solicitação de login do usuário.
- --*home* “caminho” “usuário”, define o caminho do diretório home do usuário.

- *--no-create-home* “usuário”, não cria o diretório home do usuário.
- *-ingroup* “grupo” “usuário”, cria um novo e adiciona em um grupo já existe.

A Figura 3 ilustra como é adicionado um usuário no sistema:

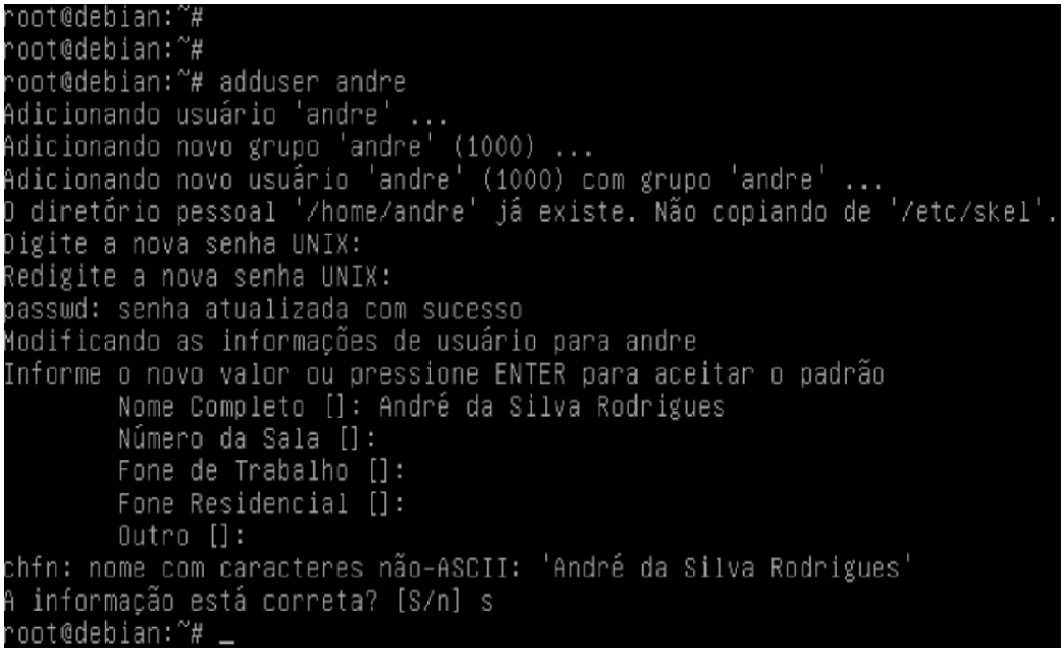

Figura 3 - Comando adduser

Logo após o comando “**# adduser ‘andre’**” ser executado o sistema pede para o administrador digitar e redigitar à senha do usuário, depois para digitar informações adicionais do usuário, que não são obrigatórios, e por último pede a confirmação das informações.

É possível alterar contas de usuários executando o comando “**# usermod ‘opção’ ‘usuário’**”, abaixo algumas opções disponíveis:

- -d “diretório” “usuário”, altera o caminho do diretório home do usuário.
- -d “diretório” –m “usuário”, altera o caminho do diretório home do usuário e move seu conteúdo para o novo diretório.
- -e “data” “usuário”, altera a data de expiração da conta do usuário, o data tem que estar com o seguinte formato “aaaa-mm-dd”.

- -l “novo usuário” “usuário”, altera o nome do usuário. Para alterar as informações adicionais (Nome Completo, Fone e etc.) basta digitar o comando “# chfn” “usuário”.

A alteração de senha do usuário pode ser feita pelo próprio, através do comando “**$ passwd**”, após o comando ser executado o sistema solicitará que digite senha atual, nova senha e confirmação da nova senha. O administrador (*root*) tem o privilégio de alterar à senha de qualquer usuário com o comando “**# passwd ‘usuário’**”, o sistema solicitará apenas que digite a nova senha e confirmação da nova senha.

Caso necessário, é possível excluir contas de usuários executando o comando “# **userdel ‘usuário’**”, a conta é deletada porém o diretório home e seu conteúdo é mantido, para excluir a conta e o home do usuário basta utilizar a opção “-r”, exemplo “# **userdel –r ‘usuário’**”.

3.2.2 Grupos

O grupo possibilita o administrador organizar de forma hierárquica e por categoria os usuários do sistema. Quando um grupo é adicionado no sistema automaticamente é criado um número de identificação (GID), todas as informações do grupo ficam armazenadas no arquivo *group* no diretório (/etc/group).

Para adicionar um grupo no sistema, é preciso estar logado como administrador (*root*). E o comando utilizado é “**# addgroup ‘grupo’**”.

É possível gerenciar grupos executando o comando “# **gpasswd ‘opção’ ‘grupo’**”. Sem “opção” este comando serve para criar ou alterar senha do grupo.

Alguns opções disponíveis:

- -a “usuário” “grupo”, adiciona usuário existente em um grupo.
- -d “usuário” “grupo”, remove usuário de um grupo.
- -A “usuário” “grupo”, coloca usuário como administrador do grupo.
- -r “grupo”, remove senha do grupo.

Para alterar o nome de um grupo é através do comando "# **groupmod –n 'novo grupo' 'grupo'**". E caso necessário excluir um grupo basta executar o comando "# **groupdel 'grupo'**".

3.2.3 Permissões

Segundo Alecrim (2010), as permissões de acesso são um dos aspectos mais importantes do SO Linux, tem a função de proteger arquivos e diretórios de usuário ou programa que não tem a permissão para acessa-los, evitando que arquivos de configurações importantes para o funcionamento do SO Linux sejam alterados por usuário comum.

Para melhor entendimento a Figura 4 mostra uma lista de arquivos e diretórios e suas permissões.

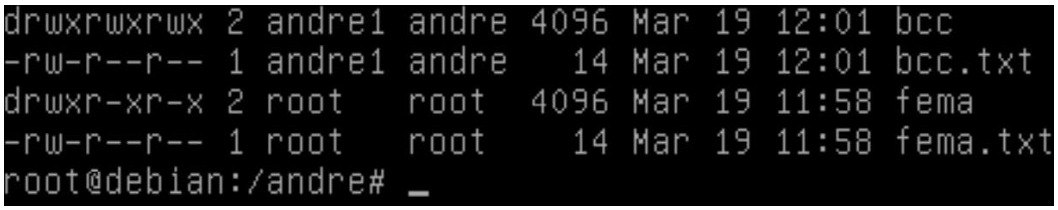

Figura 4 - Exemplo de permissões

O primeiro item que aparece em cada linha é (-rw- r-- r--) que é o modo em que o Linux apresenta as permissões de arquivos e diretórios, o primeiro caractere define o tipo do arquivo, onde tem um "d" significa que é um diretório se for "-" isso indica que é um arquivo comum.

No diretório bcc da Figura 4, nota-se que a permissão é (drwx rwx rwx), que são três blocos de três caracteres que são preenchidos com "rwx", lembrando que o primeiro caractere é o tipo do arquivo. Abaixo o significado do "rwx":

- r - é permissão de leitura.
- w - é permissão de escrita.
- x – é permissão de execução de programa

O primeiro bloco são as permissões do proprietário do diretório, o segundo são permissões do grupo ao qual o usuário pertence, e o terceiro são permissões para os demais usuários.

Para alterar a permissão de um arquivo ou diretório basta executar o comando "# **chmod 'permissão' 'diretório / arquivo'**". A Tabela 2 mostra a representação decimal e binaria das permissões "rwx".

DECIMAL	BINÁRIO	PERMISSÃO
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Tabela 2 - Permissões

Utilizando a Tabela 2 para executar o comando chmod, e alterar a permissão do diretório bcc que está configurado (drwx rwx rwx) para (drwx r-x ---), utiliza-se para cada bloco "rwx" um número decimal representando uma tipo de permissão, por exemplo "**# chmod 750 'bcc'**". Como mostra a Figura 5.

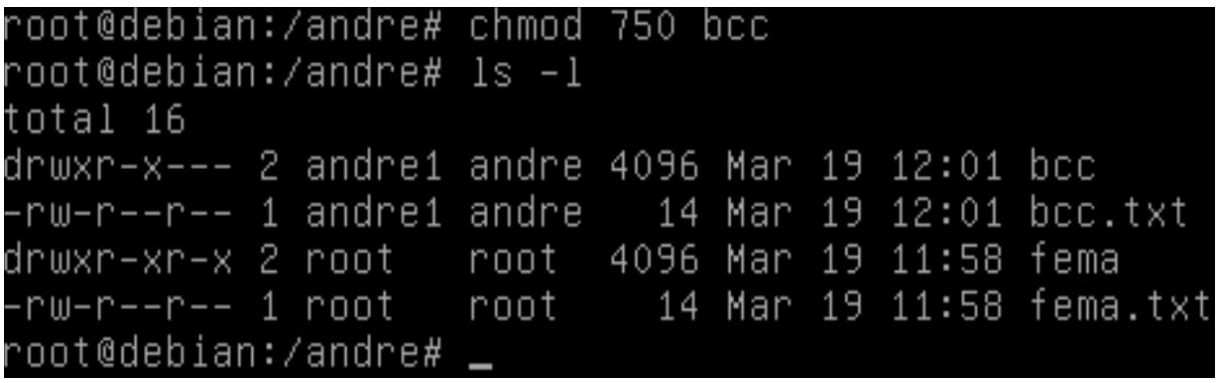

Figura 5 - Exemplo de comando chmod

O decimal 7 manteve as permissões de leitura, escrita e execução “rwx” para o dono do arquivo, o decima 5 alterou a permissão de escrita “r-x” para o grupo do usuário, e o decimal 0 alterou todas as permissões “---” para outros usuários deixando-os sem acesso ao diretório.

3.3 SHELL

Shell é um programa de usuário que oferece uma interface personalizável para seu sistema, onde são executados as linhas de comando em modo texto no “interprete de comandos” no Linux e Unix, e também serve de ambiente para execução de outros programas. Existe vários tipos de *shell* como:

- *Bourne Shell* (sh) é o mais antigo foi por muitos anos o Shell padrão do Unix.
- *Korn Shell* (ksh) possui todas as característica do sh com mais funções agregadas.
- *C shell* (csh) mais utilizados em ambientes BSD e *Xenix* seus comandos é bem similar à da linguagem C.
- *Bourne Again Shell* (bash) o mais utilizado atualmente por ser o *shell* padrão do Linux, seus comandos possui características do C *shell*.

Além desses existem outros *Shells*, e cada um possui suas próprias características, um único sistema pode conter vários *Shells* instalados paralelamente.

Para melhor entendimento do funcionamento do *shell* a Figura 6 mostra como funciona o SO Linux.

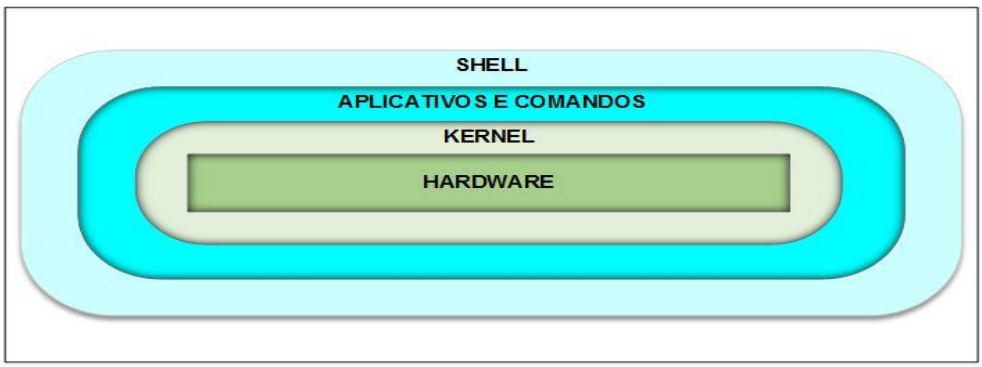

Figura 6 - Gráfico de funcionamento de Linux

O gráfico mostra que o nível mais baixo é o hardware, que é formado pelos componentes físicos de um computador, o nível seguinte é o *kernel*, o núcleo do Linux responsável pela gerência e funcionamento correto do *hardware*. Aplicativos e comandos, é o nível responsável por executar aplicativos desenvolvidos para realizar tarefas utilizando o *kernel*, por fim a camada do *shell*, que é interface entre o usuário e o SO, todas as interações homem maquina passa por ela.

3.4 SHELL SCRIPT

Shell Script é uma linguagem de programação de *scripts* utilizada por sistemas operacionais, cada SO tem sua linguagem de *script* e característica própria dependendo do interpretador de comando. O interpretador utilizado neste trabalho é o BASH que é disponível na distribuição Linux Debian.

Um *script* é um arquivo executável e em seu conteúdo existe um conjunto de instruções e funções executados em sequência pelo interpretador BASH. Arquivo em lote do Windows (bach) é outro exemplo de *script*.

Os arquivos de scripts *shell* BASH são identificados através da extensão ".sh", por exemplo, "arquivo.sh". Outra característica que identifica um script é a instrução descrita na primeira linha do arquivo que indica qual será o processador de comandos, por exemplo, "#!/bin/bash" indica o caminho do BASH.

A programação s*hell script* geralmente é utilizada para criar rotinas. O administrador do SO Linux pode criar pequenos programas, por exemplo, criar rotinas para automatizar *backups* de arquivos existentes no HD de um computador para um HD externo, criar configurações do próprio SO em que está rodando e até mesmo criar comando para rodar essas rotinas.

3.4.1 Scripts de usuários

Para adicionar usuário através de comando Linux foi mostrado neste capítulo no item 3.2.1, que após executar o comando "# **adduser usuário**", é necessário informar a

senha e repeti-la para autenticação, em seguida a informação pessoal de usuário é opcional.

O Código 2 apresentado, é o arquivo de *shell script* asr_adduser.sh que após ser executado, cria e executa o comando "**asr_adduser 'usuário' 'senha'**", que obrigatoriamente recebe os parâmetros de *username* e senha respectivamente.

```
#!/bin/bash
prog = "$0"
USERNAME = $1
PASSWORD = $2

instalado =  $(ls /usr/bin/ |grep -sow $(basename "$0")) > /dev/null 2>&1
if [ -z $instalado ]; then
      ln "$prog" /usr/bin/asr_adduser
      echo "ASR - Rodando a primeira vez"
else
      echo "ASR - Rodando o programa..."
fi

if [ -z $USERNAME ] || [ -z $PASSWORD ]; then
      echo "ASR - Usuário ou senha não informardo"
else
      LOGIN=$(echo "$USERNAME" |tr [[:upper:]] [[:lower:]])
      unset USERNAME
      USERNAME=$(cat /etc/passwd | grep -siow $LOGIN) > /dev/null 2>&1
      if [ -z $USERNAME ]; then

                  echo "ASR - Usuário < $LOGIN > está disponível"

            adduser $LOGIN --gecos "$LOGIN, , , , " --disabled-password

            (echo $PASSWORD; echo $PASSWORD) | passwd $LOGIN
      else
            echo "ASR - Usuário < $LOGIN > já existe"
      fi
fi
```

Código 2 - Shell Script asr_adduser.sh

No código acima as variáveis prog, *USERNAME*, *PASSWORD* e instalado, recebem parâmetros no momento da execução do *script*:

- Variável prog = "$0", retorna o nome do arquivo que é utilizado para rodar este programa.

- Variável USERNAME = $1, recebe primeiro parâmetro digitado no comando asr_adduser.
- Variável PASSWORD = $2, recebe o segundo parâmetro digitado no comando asr_adduser.
- Variável instalado, Linha 6, recebe o resultado de uma busca para verificar se existe no diretório /usr/bin/ instalado o comando gerado com a execução desse script.

No teste entre as linhas sete e doze, o *script* verifica se a variável instalado é nula, se a condição for verdadeira ela gera um *link* do *script* cujo o nome é armazenado na variável prog no diretório /usr/bin/asr_adduser, e esse *script* passa a ser executado com o comando "**# asr_adduser 'usuário', 'senha'**".

Após verificar se esse *script* está instalado no servidor Linux, o bloco de código entre as linhas quatorze e trinta verifica se as variáveis USUARIO e *PASSWORD* são nulas. Se a condição for falsa a variável *LOGIN* na linha dezessete recebe o conteúdo da variável USUARIO com tratamento de caixa baixa. Após ser limpa a variável *USERNAME* recebe o resultado de uma busca para verificar se usuário já existe no arquivo /etc/passwd. Se a variável for nula indica que está disponível e na linha vinte e quatro o usuário é adicionado ao Linux e na linha vinte e seis sua senha é atribuída.

A Figura 7 mostra como é executado o comando "**# asr_adduser 'usuário' 'senha'**" localmente.

```
root@debian:/home/andre# asr_adduser andre andre123
ASR - Rodando o programa...
ASR - Usuario < andre > esta disponivel
Adicionando usuário 'andre' ...
Adicionando novo grupo 'andre' (1000) ...
Adicionando novo usuário 'andre' (1000) com grupo 'andre' ...
Criando diretório pessoal '/home/andre' ...
Copiando arquivos de '/etc/skel' ...
Digite a nova senha UNIX:Redigite a nova senha UNIX:passwd: senha atualizada com
 sucesso
```

Figura 7 - Funcionamento comando asr_adduser

Podemos ver que após o comando "**# asr_adduser 'andre' 'andre123'**" ser executado automaticamente o usuário é adicionado ao Linux sem pedir mais informações.

O Código 3 mostra que além do *script* para adicionar o usuário no Linux, também foi desenvolvido o arquivo *shell script* asr_passwd.sh, que executa e cria o comando "**# asr_passwd 'usuario' 'novaSenha'**" para alterar a senha do usuário.

```
#!/bin/bash
prog = "$0"
USERNAME = $1
PASSWORD = $2

instalado =  $(ls /usr/bin/ |grep -sow $(basename "$0")) > /dev/null 2>&1
if [ -z $instalado ]; then
      ln "$prog" /usr/bin/asr_passwd
      echo "ASR - Rodando a primeira vez"
else
      echo "ASR - Rodando o programa..."
fi

if [ -z $USERNAME ] || [ -z $PASSWORD ]; then
      echo "ASR - Usuário ou senha não informardo"
else
      LOGIN=$(echo "$USERNAME" |tr [[:upper:]] [[:lower:]])
      (echo $PASSWORD; echo $PASSWORD) | passwd $LOGIN
      echo "ASR – Senha do usuário <$LOGIN> alterada com sucesso"
fi
```

Código 3 - Shell Script asr_passwd.sh

Seu funcionamento é basicamente o mesmo do *script* de adicionar usuário, a diferença é que o segundo parâmetro recebe a nova senha do usuário, e o comando gerado por ele é o "**# asr_passwd 'usuário' 'novaSenha'**".

Estes comandos são utilizados pela classe ShellRemoto, no método addUsuario no Java para adicionar ou alterar senha de usuário no servidor Linux remotamente, sem a obrigação de redigitar a senha, eliminando o uso da tecla *enter* que é constantemente pressionada a cada informação pedida ao adicionar ou editar localmente um usuário com o comando "**# adduser 'usuário'**" ou "**# passwd 'usuario'**".

Para que o sistema gerenciador de usuários, grupo, diretórios e permissões para servidor de arquivo Linux funcione de forma correta, estes *scripts* são requisitos obrigatórios nos servidores Linux que serão gerenciados.

4. PROPOSTA DE TRABALHO

As principais funcionalidades do sistema consiste em adicionar e gerenciar usuários, grupos e permissões de arquivos e diretórios remotamente, acessando um ou mais servidores Linux dentro da rede através de interface *web*. Qualquer computador ou sistema operacional é capaz de gerenciar os servidores.

4.1 CADASTRO DE SERVIDOR

O cadastro de servidor é a tela onde o administrador cadastrará os servidores que serão gerenciados remotamente pelo sistema. Após o cadastro do servidor, o sistema automaticamente sincroniza os usuários e grupos existentes no servidor e os cadastram no banco de dados.

O campo nome representa o *hostname* ou apelido do servidor. O administrador do sistema poderá apenas selecionar o nome do servidor para fazer qualquer interação com o sistema, os demais campos são preenchidos com configurações importantes para o acesso remoto via SSH. No campo senha o administrador é obrigado informar a senha *root* do servidor, pois todos os acessos remotos feitos pelo sistema será como usuário *root*.

4.2 CONEXÃO COM SERVIDOR LINUX

As conexões com os servidores são feitos através dos componentes *combo Box* e botão de conectar, onde são listados os servidores cadastrados no sistema. Como mostra a Figura 8.

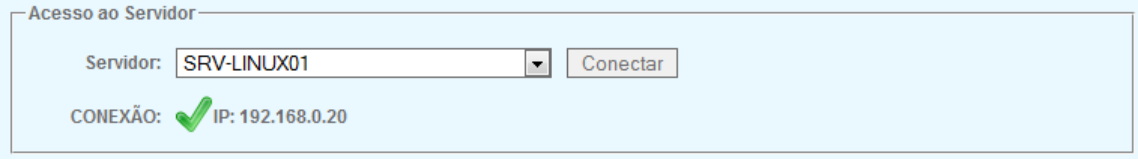

Figura 8 - Componente de conexão com servidor

Após selecionar o servidor que será conectado remotamente e clicar em conectar, se o servidor estiver em funcionamento na rede, ele retorna uma mensagem de conectado e libera os restantes dos componentes da tela para conclusão da operação, senão ele mostra um erro e bloqueia todos os componentes da tela. As telas que possuem esses componentes são: cadastro de usuário, cadastro de grupo, gerenciar permissões e gerenciar grupos.

4.3 CADASTRO DE USUÁRIO

O cadastro do usuário é dividido em quatro partes: o acesso ao servidor onde o administrador conecta o servidor ao sistema, as informações pessoais e de contatos, para manter o registro do usuário em banco de dados, e informações de login, que são passadas remotamente para adicionar o usuário no servidor Linux.

Os campos *username* e senha na área de informações de *login*, são utilizados pelo usuário para logar no servidor. Todos os dados são registrados em banco de dados. O Código 4 mostra o método addUsuario da classe shellRemoto para adicionar o usuário no servidor Linux.

```
1  public void addUsuario(Usuario usu, Servidor serv)
2                      throws ConnectionException, TransportException,
3  IOException {
4
5       try {
6       ssh = ConexaoSSH.conectarSSH(serv.getIp(), serv.getSenha());
7       sessao = ssh.startSession();
8       String linha = null;
9       if (usu.getUsername() != null && usu.getSenha() != null) {
10           Command cmd = sessao.exec("asr_adduser " + usu.getUsername()
11              + " " + usu.getSenha());
12           bf = new BufferedReader(new InputStreamReader(
13              cmd.getInputStream()));
14
15           while ((linha = bf.readLine()) != null) {
16                System.out.println(linha);
17           }
18      } else {
19            System.out
20        .println("GSAL:SHELLREMOTO - Usuário ou senha não informados");
21      }
22      } catch (IOException e) {
23            e.printStackTrace();
24      } finally {
```

```
25            ConexaoSSH.fecharRecursos(sessao);
26            ConexaoSSH.fecharRecursos(bf);
27            ConexaoSSH.fecharConexaoSSH();
28        }
29    }
```

Código 4 - Método addUsuario

4.4 CADASTRO DE GRUPOS

O cadastro do grupo tem duas partes: o acesso ao servidor onde o administrador conecta o servidor ao sistema e as informações onde tem somente dois campos, nome do grupo e observações sobre o grupo.

O campo nome é obrigatório e é utilizado para adicionar remotamente o grupo ao servidor. O campo de observações é onde o administrador descreve os objetivos do grupo criado. Todos os dados são registrados em banco de dados.

O Código 5 mostra o método addGrupo da classe shellRemoto para adicionar o grupo no servidor Linux.

```
1  public void addGrupo(Grupo grp, Servidor serv) throws ConnectionException,
2                      TransportException, IOException {
3      try {
4           ssh = ConexaoSSH.conectarSSH(serv.getIp(), serv.getSenha());
5           sessao = ssh.startSession();
6           String linha = null;
7           if (grp.getNome() != null) {
8                Command cmd = sessao.exec("addgroup " + grp.getNome());
9                bf = new BufferedReader(new InputStreamReader(
10               cmd.getInputStream()));
11               while ((linha = bf.readLine()) != null) {
12                    System.out.println(linha);
13               }
14          } else {
15               System.out.println("GSAL:SHELLREMOTO - Nome do grupo não
16 informado");
17          }
18     } catch (IOException e) {
19           e.printStackTrace();
20     } finally {
21          ConexaoSSH.fecharRecursos(sessao);
22          ConexaoSSH.fecharRecursos(bf);
23          ConexaoSSH.fecharConexaoSSH();
24     }
25 }
```

Código 5 - Método addGrupo

4.5 GERENCIAMENTO DE GRUPOS

O gerenciamento de grupos é dividido em três partes: o acesso ao servidor, onde o administrador conecta o servidor ao sistema, o acesso ao grupo, que é desbloqueado após o retorno positivo da conexão com o servidor, e as tabelas Usuário, Adicionar usuário e Usuários do grupo.

Assim que o servidor selecionado é conectado, a tabela Usuário é carregada com os usuários do servidor, a combo grupo é carregada com os grupos pertencente ao servidor selecionado. Quando um grupo é selecionado a tabela Usuários do grupo é carregada com os usuários existentes neste grupo.

Para adicionar um usuário na tabela Adicionar usuário, basta escolher um usuário na tabela Usuário e dar duplo clique no usuário escolhido, e para remover um usuário da tabela Usuário do grupo, somente com duplo *click* no usuário escolhido. Após as modificações dos grupos, ao clicar no botão gravar o sistema sincroniza com o servidor e registra as alterações no banco de dados.

O Código 6 mostra o método gerenciarUsrGrupo da classe shellRemoto para o gerenciamento de usuários dos grupos no servidor Linux.

```
1  public String gerenciarUsrGrupo(Servidor serv, Grupo grp, Usuario usr,
2  Integer addDel) throws IOException {
3       String msg = "";
4       try {
5            String linha = "";
6            String linhaFinal = "";
7            ssh = ConexaoSSH.conectarSSH(serv.getIp(), serv.getSenha());
8            sessao = ssh.startSession();
9            Command cmd = null;
10           if (addDel == 0) {
11                cmd = sessao.exec("addgroup " + usr.getUsername() + " "
12                + grp.getNome());
13                msg = "Usuário " + usr.getUsername() + " entrou no grupo "
14                + grp.getNome();
15           } else {
16                cmd = sessao.exec("gpasswd -d " + usr.getUsername() + " "
17                + grp.getNome());
18                 msg = "Usuário " + usr.getUsername() + " saiu do grupo "
19                + grp.getNome();
20           }
21           bf = new BufferedReader(new
22 InputStreamReader(cmd.getInputStream()));
23           while ((linha = bf.readLine()) != null) {
```

```
24                  linhaFinal += linha + "\n";
25              }
26              System.out.println(linhaFinal);
27          } catch (IOException e) {
28              e.printStackTrace();
29          } finally {
30              ConexaoSSH.fecharRecursos(sessao);
31              ConexaoSSH.fecharRecursos(bf);
32              ConexaoSSH.fecharConexaoSSH();
33          }
34              return msg;
35  }
```

Código 6 - Método gerenciarUrsGrupo

4.6 GERENCIAMENTO DE PERMISSÕES

O gerenciamento de permissões é dividido em três partes: o acesso ao servidor onde o administrador conecta o servidor ao sistema, o acesso ao usuário que é desbloqueado após o retorno positivo da conexão, e uma lista de diretórios e arquivos que é composto por um cabeçalho e corpo com componentes *combo box* e botões.

Ao selecionar e conectar um servidor a *combo box* de usuário é carregada com os usuários do servidor. Ao selecionar um usuário na *combo box* é carregado uma lista com todos os diretórios e arquivos do diretório *Home* do usuário selecionado. Cada item da lista tem o nome do diretório ou arquivo no cabeçalho, seu corpo é composto pelas informações como tipo, dono, grupo e uma série de combo box representando as permissões do dono, grupo e outros, do arquivo ou diretórios.

As opções das combo box de dono, grupo e outros são:

- Opção Negado, sem permissões acesso ao arquivo ou diretório.
- Opção Leitura, pode apenas ler o arquivo e não alterar, ou ver o conteúdo do diretório.
- Opção Leitura e Alteração, pode ler o arquivo e alterar seu conteúdo e total acesso ao diretório

O administrador pode alterar o tipo de permissão de qualquer um dos item da lista e posteriormente clicar no botão gravar, o sistema dispara um comando de alteração de

permissão para o servidor. Essas alterações não registram em banco de dados apenas modifica o servidor.

Se o tipo do item for um diretório, o administrador através do botão Entrar pode navegar dentro desse diretório, ele dispara um comando para o servidor e traz uma nova lista com diretórios e arquivos existentes no diretório escolhido.

O Código 7 mostra o método permissões da classe shellRemoto para o gerenciamento de permissões no servidor Linux.

```
1  public String permissoes(String permissao, Servidor serv)
2  throws IOException {
3       try {
4          ssh = ConexaoSSH.conectarSSH(serv.getIp(), serv.getSenha());
5          sessao = ssh.startSession();
6          Command cmd = sessao.exec("chmod " + permissao);
7          bf = new BufferedReader(new InputStreamReader(cmd.getInputStream()));
8
9        } catch (IOException e) {
10         e.printStackTrace();
11       } finally {
12          ConexaoSSH.fecharRecursos(sessao);
13          ConexaoSSH.fecharRecursos(bf);
14         ConexaoSSH.fecharConexaoSSH();
15       }
16      return "Permissão foi Alterada";
17 }
```

Código 7 - Método permissões

5. ANÁLISE DO SISTEMA

Os tópicos seguintes mostram análises e diagramas gerados para o desenvolvimento do sistema e da pesquisa.

5.1 CASO DE USO

A Figura 9 ilustra o caso de uso que mostra de forma simples algumas das funcionalidades do sistema, através do ator administrador e seus casos de usos com as funções que terá disponível no sistema.

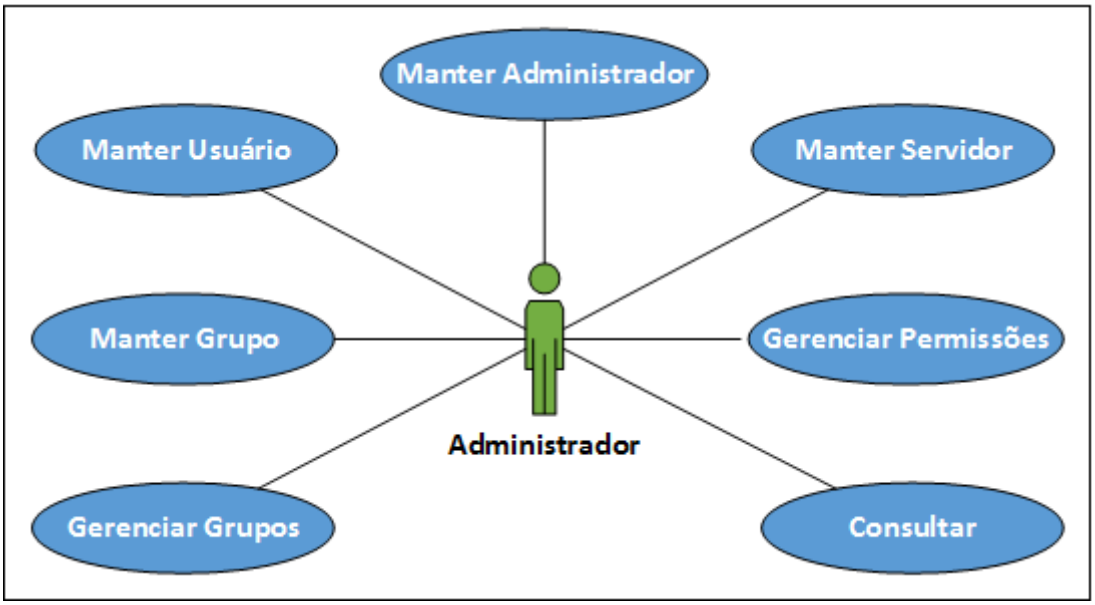

Figura 9 - Caso de uso

5.1.1 Caso de uso Manter Servidor

A Tabela 3 mostra a documentação do caso de uso Manter Servidor.

Nome do caso de uso	Manter Servidor
Ator Principal	Administrador
Resumo	Esse caso de uso descreve as etapas percorrida para cadastrar um servidor Linux no sistema.
Pré-condições	O servidor Linux deve existir na rede.
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Logar no sistema	
	2. Apresentar Menu Principal
3. Selecionar opção Cadastros	
	4. Apresentar Menu de Cadastros
5. Selecionar opção Cadastrar Servidor	
	6. Apresentar Formulário de Cadastro
7. Preencher Formulário	
	8. Validar campos
9. Clicar em gravar	
	10. Gravar Servidor no banco de dados

Tabela 3 - Caso de uso Manter Servidor

5.1.2 Caso de uso Manter Usuário

A Tabela 4 mostra a documentação do caso de uso Manter Usuário.

Nome do caso de uso	Manter Usuário
Ator Principal	Administrador
Resumo	Esse caso de uso descreve as etapas percorrida para cadastrar um usuário no servidor Linux e no sistema.
Pré-condições	O servidor Linux deve existir na rede.
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Logar no sistema	
	2. Apresentar Menu Principal
3. Selecionar opção Cadastros	
	4. Apresentar Menu de Cadastros
5. Selecionar opção Cadastrar Usuário	
	6. Apresentar Formulário de Cadastro
7. Conectar Servidor	
	8. Sincronizar Usuários e desbloquear campos
9. Preencher Formulário	
	10. Validar campos
11. Clicar em gravar	

	12. Gravar Usuário no banco de dados e adicionar ao servidor Linux

Tabela 4 - Caso de uso Manter Usuário

5.1.3 Caso de uso Manter Grupo

A Tabela 5 mostra a documentação do caso de uso Manter Grupo.

Nome do caso de uso	**Manter Grupo**
Ator Principal	Administrador
Resumo	Esse caso de uso descreve as etapas percorrida para cadastrar um grupo no servidor Linux e no sistema.
Pré-condições	O servidor Linux deve existir na rede.
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Logar no sistema	
	2. Apresentar Menu Principal
3. Selecionar opção Cadastros	
	4. Apresentar Menu de Cadastros
5. Selecionar opção Cadastrar Grupo	
	6. Apresentar Formulário de Cadastro
7. Conectar Servidor	
	8. Sincronizar Grupos e desbloquear campos
9. Preencher Formulário	
	10. Validar campos
11. Clicar em gravar	
	12. Gravar Grupo no banco de dados e adicionar ao servidor Linux

Tabela 5 - Caso de uso Manter Grupo

5.1.4 Caso de uso Manter Administrador

A Tabela 6 mostra a documentação do caso de uso Manter Administrador.

Nome do caso de uso	**Manter Administrador**
Ator Principal	Administrador
Resumo	Esse caso de uso descreve as etapas percorrida para cadastrar um Administrador no sistema.

Pré-condições	
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Logar no sistema	
	2. Apresentar Menu Principal
3. Selecionar opção Cadastros	
	4. Apresentar Menu de Cadastros
5. Selecionar opção Cadastrar Administrador	
	6. Apresentar Formulário de Cadastro
7. Preencher Formulário	
	8. Validar campos
9. Clicar em gravar	
	10. Gravar Administrador no banco de dados

Tabela 6 - Caso de uso Manter Administrador

5.1.5 Caso de uso Gerenciar Permissões

A Tabela 7 mostra a documentação do caso de uso Gerenciar Permissões.

Nome do caso de uso	**Gerenciar Permissões**
Ator Principal	Administrador
Resumo	Esse caso de uso descreve as etapas percorrida para gerenciar permissões de diretório e arquivos no servidor Linux.
Pré-condições	O servidor Linux deve existir na rede.
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Logar no sistema	
	2. Apresentar Menu Principal
3. Selecionar opção Servidor	
	4. Apresentar Menu de Servidor
5. Selecionar opção Permissões	
	6. Apresentar Formulário de Gerenciamento
7. Conectar Servidor	
	8. Sincronizar Usuários e desbloquear campos
9. Selecionar Usuário	
	10. Carregar lista de diretórios e arquivos do diretório home do usuário
11. Alterar Permissões	

	12. Validar campos
13. clicar em gravar	
	14. Alterar permissão no servidor Linux

Tabela 7 - Caso de uso Gerenciar Permissões

5.1.6 Caso de uso Gerenciar Grupos

A Tabela 8 mostra a documentação do caso de uso Gerenciar Grupos.

Nome do caso de uso	**Gerenciar Grupos**
Ator Principal	Administrador
Resumo	Esse caso de uso descreve as etapas percorrida para gerenciar grupos do servidor Linux.
Pré-condições	O servidor Linux deve existir na rede.
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Logar no sistema	
	2. Apresentar Menu Principal
3. Selecionar opção Servidor	
	4. Apresentar Menu de Servidor
5. Selecionar opção Grupos	
	6. Apresentar Formulário de Gerenciamento
7. Conectar Servidor	
	8. Sincronizar Usuários e desbloquear campos
9. Selecionar Grupo	
	10. Carregar tabelas de usuários do servidor e dos grupo
11. Adicionar usuário ao grupos	
	12. Validar campos
13. clicar em gravar	
	14. Adicionar usuários ao grupo no servidor Linux

Tabela 8 - Caso de uso Gerenciar Grupos

5.1.7 Caso de uso Consultar

A Tabela 9 mostra a documentação do caso de uso Gerenciar Consultar.

Nome do caso de uso	Consultar
Ator Principal	Administrador
Resumo	Esse caso de uso descreve as etapas percorrida para consultas
Pré-condições	O servidor Linux deve existir na rede.
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Logar no sistema	
	2. Apresentar Menu Principal
3. Selecionar opção Consultas	
	4. Apresentar filtro de consultas
5. Clicar Consultar	
	6. Apresentar resultado da consulta

Tabela 9 - Caso de uso Consultar

5.2 DIAGRAMA DE CLASSE

Neste item a Figura 10 apresentada o diagrama das classes utilizadas no desenvolvimento do sistema.

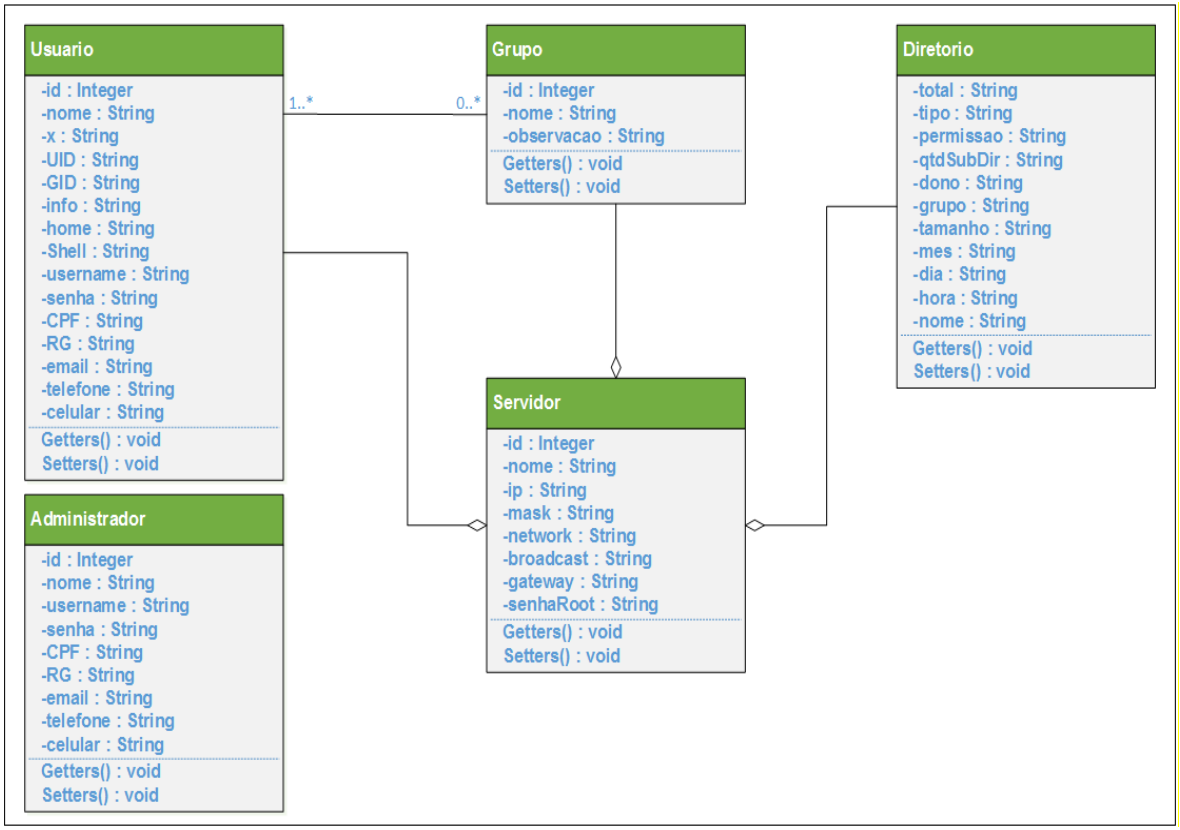

Figura 10 - Diagrama de classe

5.3 MAPA WEB DO SISTEMA

Na Figura 11 é apresentado um diagrama que representa a navegação e os níveis das páginas web ou telas do sistema.

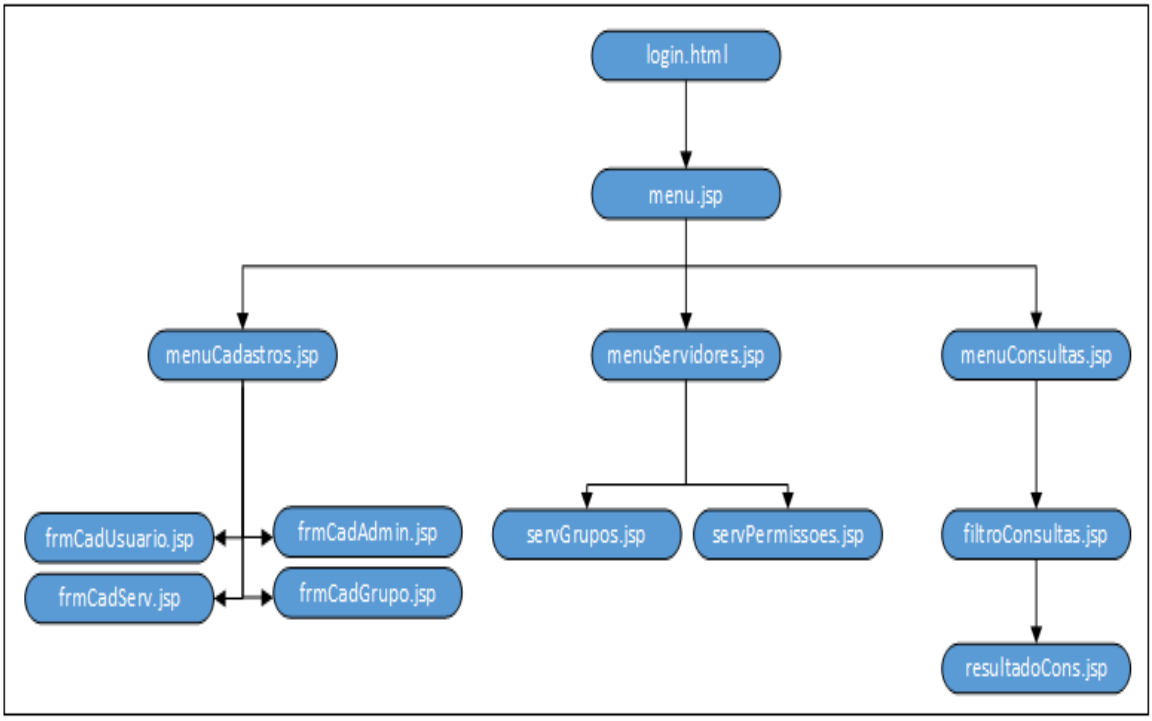

Figura 11 - Mapa web do sistema

5.4 TELAS DO SISTEMA

Neste item é apresentado as imagens das principais telas do sistema gerenciador de usuário, grupos, diretórios e permissões para servidor de arquivo Linux.

5.4.1 Tela de Cadastro de servidor

A Figura 12 - apresenta a tela de cadastro de servidor.

Figura 12 - Tela cadastro de servidor

5.4.2 Tela de cadastro de usuário

A Figura 13 - apresenta a tela de cadastro de usuário.

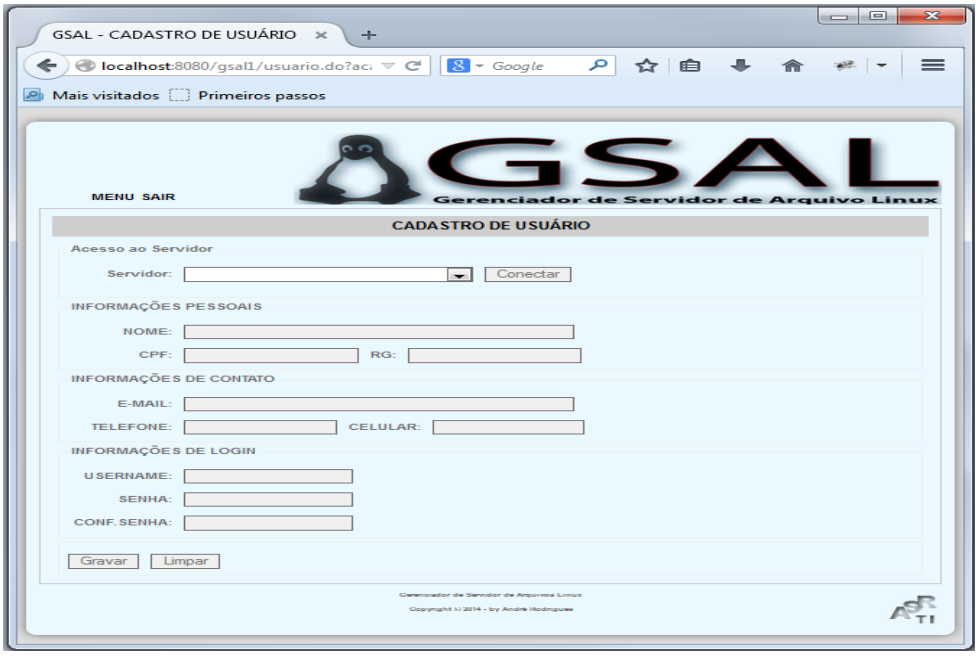

Figura 13 - Tela cadastro de usuário

5.4.3 Tela de cadastro de grupo

A Figura 14 - apresenta a tela de cadastro de grupo.

Figura 14 - Tela cadastro de Grupo

5.4.4 Tela de gerenciamento de grupos

A Figura 15 - apresenta a tela de gerenciamento de grupos.

Figura 15 - Tela gerenciar grupos

5.4.5 Tela de gerenciamento de permissões

A Figura 16 - apresenta a tela de gerenciamento de permissões.

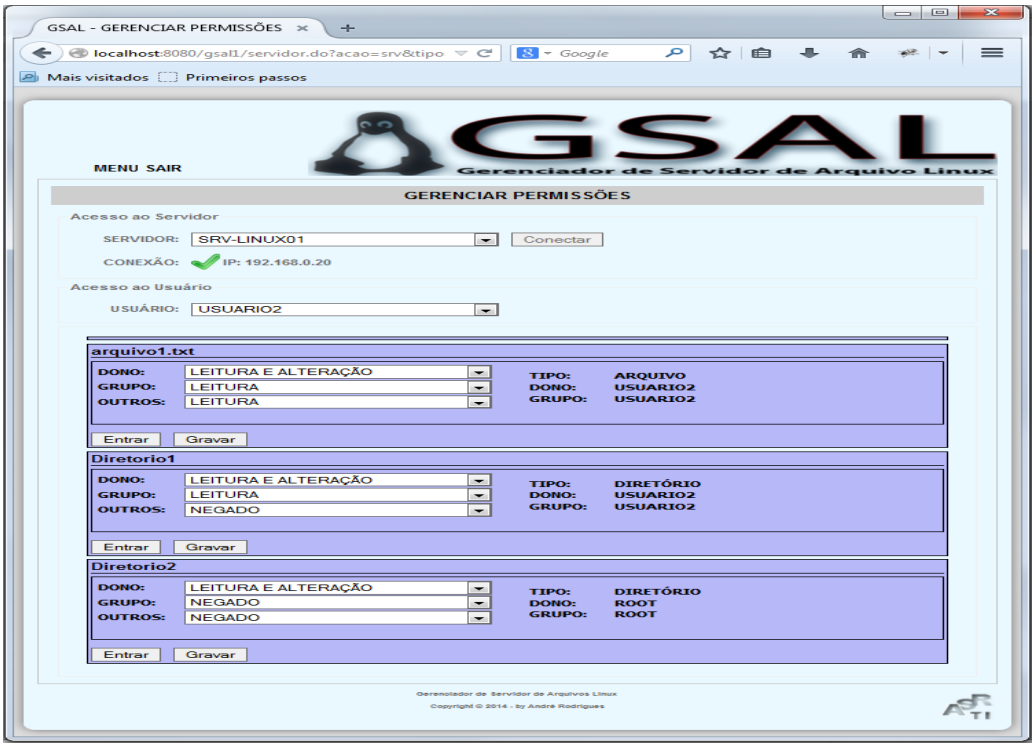

Figura 16 - Tela gerenciar permissões

5.5 RECURSOS

Neste item são apresentados recursos utilizados para o desenvolvimento do trabalho, como, de hardwares, softwares e bibliográficos.

5.5.1 Hardware

- Notebook Itautec *InfoWay*.
- Processador: Intel Core i5-450M 2.4 GHz.
- Memória RAM: 4 GB DDR 3.
- HD: 500 GB.
- Placa de Vídeo: 1 GB ATI *Mobility Radeon* HD4650.

5.5.2 Software

- Sistema Operacional Microsoft: Windows 7 Home Premium.
- Sistema Operacional Linux: Ubuntu e Debian.
- Linguagem de Programação: Java.
- Ferramenta de Desenvolvimento: IDE Eclipse Kepler.
- Sistema Gerenciador Bando de Dados: MySQL *Workbench*.
- Máquina Virtual: Oracle VirtualBox.
- Ferramenta: ArgoUML.

6. CONCLUSÃO

Durante o desenvolvimento do trabalho foram apresentados conceitos e ferramentas que possibilitam implementar sistemas em Java que se conectam ao SO Linux através de acesso remoto, apresentando como executar comandos gerados através de métodos em Java em SO Linux que utiliza o BASH como *shell*.

Foi estudado a biblioteca sshj.jar e mostrado a classe de conexão SSH assim como os códigos para adicionar usuários, grupos, gerencia de grupos e permissões. Também foi estudado o *shell script* e mostrado o código gerado para criar o comando para adicionar usuário no servidor Linux de forma alternativa ao comando padrão do Linux.

O resultado foi o desenvolvimento o Sistema Gerenciador de Usuários, Grupos, Diretórios e Permissões para Servidor de Arquivo Linux, que possibilita ao administrador de servidores Linux maior agilidade na gerencia de usuários, grupos e permissões de arquivos e diretórios. Através de uma interface web pode-se gerenciar um ou mais servidores remotamente apenas com alguns cliques sem a necessidade de executar comando de texto.

Como trabalhos futuros, o objetivo é desenvolver módulos de monitoramento e gerenciamento de processos que estão rodando no servidor Linux, e a configuração automática do servidor Samba, que permite ao usuário acessar e compartilhar arquivos do servidor Linux em uma máquina Windows.

7. REFERÊNCIAS

ALECRIM, Emerson. **Criando e gerenciando usuários no GNU/Linux** – disponível em: < http://www.infowester.com/usuarioslinux.php>. Acesso em: 30 jan. 2014.

ALECRIM, Emerson. **Entendendo e usando permissões no Linux** – disponível em: < http://www.infowester.com/linuxpermissoes.php>. Acesso em: 02 fev. 2014.

ALMEIDA, Ronieri. **Introdução ao novo MySQL Workbench** – disponível em: < http://www.devmedia.com.br/introducao-ao-novo-mysql-workbench/25939>. Acesso em: 15 fev. 2014.

BRLINK. ***Servidor de Arquivos –*** *Disponível em:* <http://www.brlink.com.br/s/linux/servidor-linux/servidor-de-arquivos>. Acesso em: 15 out. 2013.

DEITEL, Paul; DEITEL, Harvey. **Java: Como programar**. 8. Ed. Tradução de Edson Furmankiewicz. Editora Pearson, 2010.

FERREIRA, Rubens. E. **Linux: Guia do Administrador do Sistema.** 2. Ed. Editora Novatec, 2008.

FILHO, João Eriberto. M. **Descobrindo o Linux entendo o sistema operacional GNU/Linux.** 3. Ed. Editora Novatec, 2012.

GALLARDO, David; ANISZCZYK, Chris Introdução à Plataforma Eclipse – disponível em: <https://www.ibm.com/developerworks/br/library/os-eclipse-platform/>. Acessado em: 03 Mar. 2014.

GARBEE, Bdale. **Uma Breve História do Debian** – Tradução de: Michelle Ribeiro Disponível em: <https://www.debian.org/doc/manuals/project-history/ch-intro.pt.html>. Acessado em: 10 Mar. 2014.

GONÇALVES, Edson. **Dominando Eclipse**. Ed 1. Editora Ciência Moderna, 2006.
JÚNIOR, Waldemar. S. **VirtualBox** 1.Ed. Editora CDTC – Centro de Difusão de Tecnologia e Conhecimento, 2008.

LIMA, Priscila S.N. **Evolução e Principais Características do IDE Eclipse**. 2010. 8pg. Dissertação - Departamento de Ciência da Computação – (Universidade Federal de Goiás), GO, Catalão, 2010.

MARAN, Fábio. **Alterando o dono/grupo de um arquivo** – Disponível em: <http://www.vivaolinux.com.br/artigo/Leia-grave-e-execute?pagina=3>. Acesso em: 02 fev. 2014.

MORIMOTO, Carlos. E. **Linux Redes e Servidores**. 2. ED. Editora GDH Press e Sul Editores, 2006.

PEREIRA, Ana Paula. **A História do Linux -** *Disponível em:* <http://www.tecmundo.com.br/sistema-operacional/4228-a-historia-do-linux.htm>. Acesso 25 out. 2013.

SENAGA, Marcelo. **Executando Shell Scripts utilizando Java** – Disponível em: < http://www.devmedia.com.br/executando-shell-scripts-utilizando-java/26495>. Acessado em 20 Jul. 2014.

SILVA, Gleydson. M. **Guia Foca GNU/Linux** – Disponível em: <http://www.guiafoca.org/?page_id=14>. Acessado em: 10 Out. 2013.

SOUZA, João Paulo. C. **MySQL** 1.Ed. Editora CDTC – Centro de Difusão de Tecnologia e Conhecimento, 2006.

VIRGILIO, Jeferson. **O Que é Shell Script** – Disponível em: < http://www.vivaolinux.com.br/artigo/O-que-e-Shell-Script >. Acessado em: 05 Ago. 2014.