

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

**SISTEMA PARA AUTOMATIZAR O MONITORAMENTO DE
ROTEADORES DE UM PROVEDOR DE ACESSO**

JEAN VICTOR ZUNINO

BLUMENAU
2012

2012/2-16

JEAN VICTOR ZUNINO

**SISTEMA PARA AUTOMATIZAR O MONITORAMENTO DE
ROTEADORES DE UM PROVEDOR DE ACESSO**

Trabalho de Conclusão de Curso submetido à
Universidade Regional de Blumenau para a
obtenção dos créditos na disciplina Trabalho
de Conclusão de Curso II do curso de Ciência
da Computação — Bacharelado.

Prof. Miguel Alexandre Wisintainer - Orientador

**BLUMENAU
2012**

2012/2-16

SISTEMA PARA AUTOMATIZAR O MONITORAMENTO DE ROTEADORES DE UM PROVEDOR DE ACESSO

Por

JEAN VICTOR ZUNINO

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente: _____
Prof. Miguel Alexandre Wisintainer – Orientador, FURB

Membro: _____
Prof. Francisco Adell Péricas – FURB

Membro: _____
Prof. Mauro Marcelo Mattos – FURB

Blumenau, 03 de Dezembro de 2012

Dedico este trabalho a minha família e a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, pelo apoio e incentivo.

À minha namorada, pelo apoio, pelas cobranças e pela compreensão nos momentos em que não estive presente.

Aos meus amigos, pelas cobranças e compreensão nos momentos em que não estive presente.

Ao meu orientador, Miguel Alexandre Wisintainer, por ter acreditado na conclusão deste trabalho.

À professora Joyce Martins, que colaborou para a especificação do analisador léxico e sintático.

A mente que se abre a uma nova idéia jamais
voltará ao seu tamanho original.

Albert Einstein

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema *web* para automatizar o monitoramento de roteadores de um provedor de acesso. O sistema possui uma área de configuração, onde o usuário configura as mensagens que o sistema enviará para os roteadores. As mensagens são configuradas baseadas nos comandos que é possível executar por linha de comando, quando conectado no roteador através de terminal virtual telnet. Para realizar o monitoramento o sistema utiliza regras que são criadas pelo usuário. O sistema também fornece uma área para o usuário monitorar os roteadores de acordo com as mensagens e regras criadas.

Palavras-chave: Provedor de acesso. Mikrotik. Monitoramento. Rede wireless. Internet via rádio.

ABSTRACT

This paper presents the development of a web-based system to automate the monitoring of routers of an ISP. The system has a configuration area where the user configures the system to send messages to the routers. Messages are configured based on the commands you can run from the command line, when connected to the router with terminal virtual telnet. To perform the monitoring system uses rules that are created by the user. The system also provides an area for the user to monitor the routers according to the messages and rules created.

Keywords: Access provider. Mikrotik. Monitoring. Wireless network. Internet radio.

LISTA DE ILUSTRAÇÕES

Figura 1 - Monitoramento da tensão da bateria de uma RB433AH através de Telnet.....	18
Figura 2 - Protocolos no modelo TCP/IP inicial	20
Figura 3 – Uma solução para a RouterBoard ser alimentada com energia solar.....	23
Figura 4 – Foto da solução para a RouterBoard ser alimentada com energia solar	24
Quadro 1 - Detalhes da RB433AH.....	25
Figura 5 – Foto da RB433AH.....	25
Figura 6 – Tela inicial do GALS	28
Figura 7 – Tela de <i>login</i> da WinBox	30
Figura 8 - Monitoramento da tensão da bateria de uma RouterBoard, através da WinBox	30
Figura 9 – Monitoramento da eficiência da banda de transmissão em relação à banda máxima disponível no link (Overall Tx CCQ).....	31
Figura 10 – Tela de <i>login</i> da ferramenta Winbox.....	32
Figura 11 - Monitoramento da tensão da bateria de uma RouterBoard, através da Webfig	32
Figura 12 - Painel com as Funções criadas/alteradas na ferramenta The Dude	33
Figura 13 - Verificação da tensão.....	33
Figura 14 - Diagrama de casos de uso	36
Quadro 2 - UC01 - Manter equipamentos	37
Quadro 3 - UC02 - Manter responsáveis pelo monitoramento.....	38
Quadro 4 - UC03 - Manter regras.....	39
Quadro 5 – UC04 – Visualizar <i>status</i> dos equipamentos	40
Quadro 6 – UC05 – Configurar intervalo do monitoramento	40
Quadro 7 – UC06 – Monitorar propriedades.....	41
Quadro 8 - UC07 - Notificar responsável.....	42
Figura 15 – Diagrama de classes do pacote modelo	43
Figura 16 – Diagrama de classes do pacote dao	44
Figura 17 – Diagrama de classes do pacote service.....	45
Figura 18 – Diagrama de classes para o lado cliente.....	46
Figura 19 – Diagrama de sequência para abertura da tela manutenção de equipamentos.....	47
Figura 20 - Diagrama de sequência para o Sistema Monitorador dos Equipamentos	48
Figura 21 – Modelagem de dados conceitual do sistema	49

Figura 22 – Modelo de dados físico do sistema	49
Figura 23 – Arquitetura do sistema	51
Quadro 9 – Configuração do arquivo <code>persistence.xml</code>	53
Figura 24 – Método <code>save</code> da classe <code>AbstractDAO</code>	54
Figura 25 – Método <code>loadDAO</code> da classe <code>DAOFactory</code>	54
Figura 26 - Método <code>getEntityManagerFactory</code> da classe <code>EntityManagerFactoryPool</code>	55
Figura 27 – Método <code>buscarTodos</code> da classe <code>NetworkManagerJFacade</code>	56
Figura 28 – Método <code>save</code> da classe <code>NetworkManagerJFacade</code>	56
Figura 29 – Método <code>run</code> da <i>thread</i> <code>ExecutorMensagem</code>	57
Figura 30 – método <code>processarMensagens</code> da classe <code>ExecutorMensagem</code>	57
Figura 31 – método <code>processaMensagem</code> da classe <code>ExecutorMensagem</code>	58
Figura 32 – Método <code>getConnection</code> da classe <code>ConnectionPool</code>	59
Figura 33 - método <code>processaAcaoMensagem</code> da classe <code>ExecutorMensagem</code>	59
Figura 34 – Método <code>parser</code> da classe <code>ExecutorMensagem</code>	60
Figura 35 – Definições regulares utilizando a ferramenta GALS	61
Figura 36 – Definição dos <i>tokens</i> utilizando a ferramenta GALS.....	61
Figura 37 - Definição dos não terminais utilizando a ferramenta GALS.....	62
Figura 38 - Definição da gramática utilizando a ferramenta GALS.....	63
Figura 39 – <i>Tokens</i> reconhecidos para resposta da mensagem <i>system health print</i>	64
Figura 40 – Árvore sintática para resposta da mensagem <i>system health print</i>	64
Figura 41 – implementação do método <code>run</code> da classe <code>NotificadorResponsaveis</code>	65
Figura 42 – Método <code>validarSessao</code> da classe <code>JFlexService</code>	66
Figura 43 – Métodos <code>doLogin</code> e <code>doLogout</code> da classe <code>JFlexServiceLogin</code>	66
Figura 44 – Implementação de alguns métodos da classe <code>NetworkmanagerJFlexService</code>	67
Figura 45 – Construtor da classe <code>GridViewMdiWindow</code>	68
Figura 46 – Método <code>onCreateComplete</code> da classe <code>GridViewMdiWindow</code>	68
Figura 47 – Método <code>operationSucess</code> da classe <code>GridViewMdiWindow</code>	69
Figura 48 – Método <code>operationFailed</code> da classe <code>GridViewMdiWindow</code>	69
Figura 49 – Método <code>onCreateComplete</code> da classe <code>NetworkManagerJ</code>	70
Figura 50 - Código para o <i>layout</i> das telas de <i>login</i> e alterar senha	71
Figura 51 – Código para o <i>layout</i> do menu	71

Figura 52 – Trecho de código da classe <code>EquipamentoStatusGridView</code>	72
Figura 53 – Trecho de código da classe <code>EquipamentoStatusGridScript</code> para atualizar as colunas da tela de monitoramento	72
Quadro 10 – Linhas incluídas no arquivo <code>web.xml</code> para a configuração do BlazeDS	73
Quadro 11 – Linhas incluídas no arquivo <code>remoting-config.xml</code>	73
Figura 54 – Tela de <i>login</i>	74
Figura 55 – Falha no <i>login</i> para usuário ou grupo inativo	74
Figura 56 – Tela para alterar senha	75
Figura 57 – Tela do menu principal do sistema.....	75
Figura 58 – Tela de manutenção de grupo de usuários	76
Figura 59 – Painel para cadastrar novo grupo de usuário.....	77
Figura 60 – Tela de manutenção de usuários	77
Figura 61 – Painel para cadastrar novo usuário.....	78
Figura 62 – Tela de manutenção de equipamentos.....	79
Figura 63 – Painel para cadastrar novo equipamento.....	79
Figura 64 – Tela de manutenção de responsáveis pelo monitoramento	80
Figura 65 – Tela de manutenção de mensagens	80
Figura 66 – Tela de manutenção de regras	81
Figura 67 – Tela de visualização dos alarmes gerados pelo sistema.....	82
Figura 68 – Alertas gerados do tipo <code>Parâmetro da regra não encontrado</code>	83
Figura 69 – Tela de visualização de <i>status</i> com uma regra cadastrada no sistema	83
Figura 70 - Tela de visualização de <i>status</i> com duas regras cadastradas no sistema	84
Figura 71 – Atualizando tela de monitoramento	84
Figura 72 – Tela de configurações do sistema	85
Figura 73 – E-mails enviados para o responsável pelo monitoramento	85
Figura 74 – Tabelas criadas no banco de dados	87
Figura 75 – Transferência dos arquivos para a pasta <code>webapps</code> do Tomcat localizado na máquina do provedor de acesso	88
Figura 76 – Linha do arquivo <code>server.xml</code>	88
Figura 77 – Acessando o sistema no provedor de acesso pela porta 8083.....	89
Figura 78 – Cadastro de novo usuário no provedor de acesso	90
Figura 79 – Equipamentos no provedor de acesso Central Net.....	91
Figura 80 – Cadastro de novo equipamento	91

Figura 81 – Equipamentos cadastrados no provedor de acesso.....	92
Figura 82 – Cadastro da mensagem <i>system health print</i>	93
Figura 83 – Log do monitoramento do primeiro equipamento cadastrado	93
Figura 84 – Alterando nome da RouterBoard através da Winbox	94
Quadro 12 – Alteração na gramática para o parâmetro <i>voltage</i>	95
Figura 85 – Alertas gerados pelo erro de <i>parser</i> da mensagem	95
Figura 86 – Cadastro da regra para o parâmetro <i>voltage</i>	96
Figura 87 – Caixa de seleção mensagem e tipo validação permitem edição do texto selecionado	96
Figura 88 - Trecho do <i>log</i> do Tomcat para o monitoramento da regra cadastrada.....	97
Figura 89 – Monitoramento da tensão elétrica das <i>RouterBoards</i> no provedor de acesso.....	97
Figura 90 - Alertas gerados no monitoramento do parâmetro <i>voltage</i>	98

LISTA DE TABELAS

Tabela 1 – Equipamentos utilizados para a RouterBoard ser alimentada com energia solar ...	23
Tabela 2 – Descrição das tabelas	50
Tabela 3 – <i>Checklist</i> dos Requisitos	99
Tabela 4 - Tabela de avaliação do sistema pelos usuários	100
Tabela 5 – Comparação entre os trabalhos correlatos e o sistema desenvolvido neste trabalho	101

LISTA DE SIGLAS

ADSL - *Asymmetric Digital Subscriber Line*

AMF - *Action Message Format*

AP - *Access Point*

API - *Application Programming Interface*

BGP - *Border Gateway Protocol*

CDM – *Conceptual Data Model*

DAO – *Data Access Object*

EA - *Enterprise Architect*

HTTP - *HyperText Transfer Protocol*

IDE – *Integrated Development Environment*

IP - *Internet Protocol*

IPSEC - *Internet Protocol Security*

JDK – *Java Development Kit*

MDI - *Multiple Document Interface*

MPLS - *Multi Protocol Label Switching*

MVC – *Model View Controller*

OO – *Orientação à Objetos*

OSPF - *Open Shortest Path First*

PDM – *Physical Data Model*

PPTP - *Point-to-Point Tunneling Protocol*

QoS - *Quality of Service*

RADIUS - *Remote Authentication Dial In User Service*

RIA - *Rich Internet Applications*

RF - Requisitos Funcionais

RIP - *Routing Information Protocol*

RNF - Requisitos Não Funcionais

SDK - *Software Development Kit*

SNMP - *Simple Network Management Protocol*

SO – Sistema Operacional

TCP - *Transmission Control Protocol*

UML - *Unified Modeling Language*

URL - *Uniform Resource Locator*

VPN - *Virtual Private Network*

WDS - *Wireless Distribution System*

XML – *EXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO.....	17
1.1 OBJETIVOS DO TRABALHO	18
1.2 ESTRUTURA DO TRABALHO	19
2 FUNDAMENTAÇÃO TEÓRICA	20
2.1 TELNET	20
2.2 ROUTEROS	21
2.3 ROUTERBOARD	22
2.3.1 RB433AH.....	24
2.4 GERÊNCIA DE REDES	25
2.5 ANÁLISE LÉXICA E SINTÁTICA.....	26
2.6 GALS.....	27
2.7 ADOBE FLEX	28
2.8 TRABALHOS CORRELATOS.....	29
2.8.1 WINBOX	29
2.8.2 WEBFIG	31
2.8.3 THE DUDE.....	32
3 DESENVOLVIMENTO.....	34
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO.....	34
3.2 ESPECIFICAÇÃO	35
3.2.1 DIAGRAMA DE CASOS DE USO	35
3.2.2 Diagrama de classes	42
3.2.2.1 Diagramas de classes do lado servidor	42
3.2.2.1.1 Diagrama de classes do pacote modelo	42
3.2.2.1.2 Diagrama de classes do pacote dao	43
3.2.2.1.3 Diagrama de classes do pacote service.....	44
3.2.2.2 Diagrama de classes do lado cliente	45
3.2.3 Diagramas de sequência.....	46
3.2.4 Modelagem de dados.....	48
3.3 IMPLEMENTAÇÃO	50
3.3.1 Técnicas e ferramentas utilizadas.....	52
3.3.1.1 Implementação do lado servidor.....	52

3.3.1.1.1	Implementação da camada DAO.....	53
3.3.1.1.2	Implementação da classe NetworkmanagerJFacade.....	55
3.3.1.1.3	Implementação do Sistema Monitorador dos Equipamentos.....	56
3.3.1.1.4	Implementação do Sistema Notificador.....	65
3.3.1.1.5	Implementação da camada de serviço para o Flex.....	65
3.3.1.2	Implementação do lado cliente.....	67
3.3.1.2.1	Implementação da classe GridViewMdiWindow.....	67
3.3.1.2.2	Implementação da classe NetworkManagerJ.....	69
3.3.1.2.3	Implementação da tela de monitoramento.....	71
3.3.1.3	Configuração do BlazeDS.....	72
3.3.2	Operacionalidade da implementação.....	73
3.3.3	Tela de login.....	73
3.3.4	Menu principal.....	75
3.3.5	Manutenção de grupo de usuários.....	76
3.3.6	Manutenção de usuários.....	77
3.3.7	Manutenção de equipamentos.....	78
3.3.8	Manutenção de responsáveis pelo monitoramento.....	79
3.3.9	Manutenção de mensagens.....	80
3.3.10	Manutenção de regras.....	81
3.3.11	Visualização de alertas gerados pelo sistema.....	82
3.3.12	Tela de monitoramento.....	83
3.3.13	Tela de configurações do sistema.....	84
3.3.14	<i>E-mail</i> do Sistema Notificador.....	85
3.4	IMPLANTAÇÃO E TESTES DO SISTEMA NO PROVEDOR DE ACESSO.....	86
3.4.1	Implantação do sistema no provedor de acesso.....	86
3.4.2	Teste piloto.....	89
3.5	RESULTADOS E DISCUSSÃO.....	98
4	CONCLUSÕES.....	102
4.1	EXTENSÕES.....	103
	REFERÊNCIAS BIBLIOGRÁFICAS.....	104
	APÊNDICE A – Questionário de avaliação.....	108

1 INTRODUÇÃO

À medida que as redes de computadores foram crescendo, passaram a fazer parte do cotidiano das pessoas através de recursos e serviços que permitem uma maior interação entre os usuários e um conseqüente aumento de produtividade. Surgiram então serviços como correio eletrônico, transferência de arquivos, internet, aplicações multimídia, dentre outras, aumentando ainda mais a complexidade das redes. O mundo da interconexão de sistemas passou a conviver com a grande heterogeneidade de padrões, sistemas operacionais, equipamentos, etc. (PINHEIRO, 2006).

Considerando este quadro, Pinheiro (2006) informa que torna-se cada vez mais necessário o gerenciamento do ambiente de redes de computadores. Surge então a necessidade de buscar uma maneira consistente de realizar o gerenciamento de redes para manter toda esta estrutura funcionando de forma a atender às necessidades dos usuários e às expectativas dos administradores.

Paralelamente, o mercado da internet via rádio vem crescendo a cada ano, devido ao custo relativamente barato em relação a outros meios de acesso a internet. Este aumento vem ocorrendo principalmente nas pequenas e médias cidades, onde as redes *Asymmetric Digital Subscriber Line* (ADSL) não são contempladas a priori (CABRAL, 2007, p. 83). Isso faz com que sejam criados novos provedores de acessos nestes locais.

No provedor de acesso a internet via rádio Central Net, localizado na cidade de Major Gercino, são utilizados roteadores chamados RouterBoard (MIKROTIK, 2012a). Alguns destes equipamentos se localizam em torres altas de difícil acesso e são alimentados com baterias. As baterias são recarregadas com energia solar, utilizando um painel solar.

Acompanhando o monitoramento destes equipamentos, observou-se que o mesmo é lento, pois é possível monitorar apenas um equipamento por vez. Para cada equipamento que se deseja monitorar, é necessário fazer o *login*.

Durante o funcionamento da rede podem ocorrer diversos problemas com estes roteadores. Alguns destes problemas podem ser relacionados com: tensão da bateria, qualidade de conexão do cliente, ruído no sinal, etc.

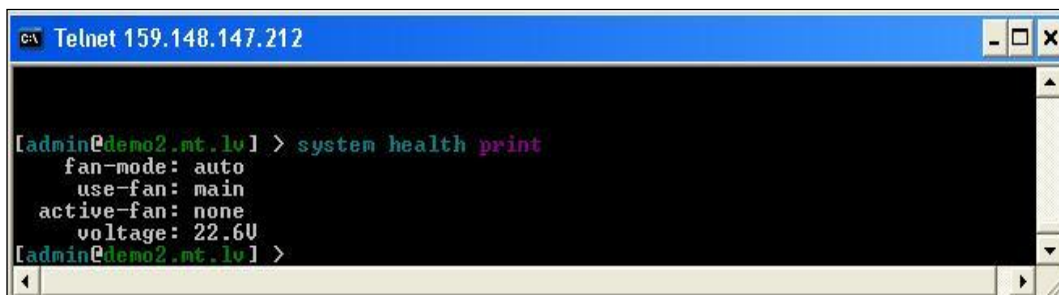
Diante do exposto, este trabalho relata o desenvolvimento de um sistema que possibilita monitorar os equipamentos de um provedor de acesso a internet via rádio que utiliza RouterBoard e seu Sistema Operacional (SO) RouterOS. Este monitoramento é realizado baseado nas três etapas do modelo clássico de gerenciamento de rede: coleta de

dados, diagnóstico e ação ou controle. Este sistema roda em um servidor e é acessado através de um navegador. O sistema faz o monitoramento dos equipamentos, de tempos em tempos, guardando um histórico de algumas propriedades e mantendo o *status* atual dos equipamentos para mostrar ao usuário na interface web. O sistema também atua sobre algumas situações que identifica como problemáticas baseado nas regras cadastradas pelo usuário.

Para fazer o monitoramento dos equipamentos o sistema conecta-se ao Mikrotik através de Telnet, envia mensagens, lê a resposta, faz um *parser* da resposta e realiza as validações necessárias dos parâmetros.

Entre os modelos disponíveis de RouterBoard, foi utilizado neste trabalho o modelo RB433AH (NOVANETWORK, 2012).

Na Figura 1 é apresentado um exemplo do monitoramento da tensão elétrica da bateria de uma RouterBoard modelo RB433AH, através de um terminal virtual Telnet. Neste exemplo, o parâmetro a ser monitorado seria *voltage* (tensão), que está com valor 22.6V. O usuário cadastra uma regra informando que o valor mínimo aceitável para a tensão é 10 volts. Quando a tensão baixar de 10 volts, o sistema gera uma notificação, registrando o ocorrido. Esta notificação é enviada por e-mail para as pessoas que devem ser notificadas.



```
ca Telnet 159.148.147.212
[admin@demo2.mt.lv] > system health print
fan-mode: auto
use-fan: main
active-fan: none
voltage: 22.6V
[admin@demo2.mt.lv] >
```

Figura 1 - Monitoramento da tensão da bateria de uma RB433AH através de Telnet

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é disponibilizar um sistema web para monitorar os roteadores RB433AH.

Os objetivos específicos do trabalho são:

- a) criar uma interface amigável para o monitoramento, permitindo monitorar mais de um roteador por vez;
- b) estabelecer comunicação e efetuar a troca de mensagens entre o sistema e a RB433AH, respeitando o manual do fabricante;

- c) atuar sobre as variáveis de controle, gerando alarmes e notificando via *e-mail* as pessoas responsáveis pelo monitoramento.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está dividido em capítulos que serão explicados a seguir.

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

O segundo capítulo apresenta a fundamentação teórica do trabalho e a apresentação dos trabalhos correlatos.

O terceiro capítulo apresenta os requisitos do sistema, sua especificação, implementação e resultados. Também é apresentada a operacionalidade do sistema.

O quarto capítulo aborda as conclusões, as contribuições e as sugestões para trabalhos futuros.

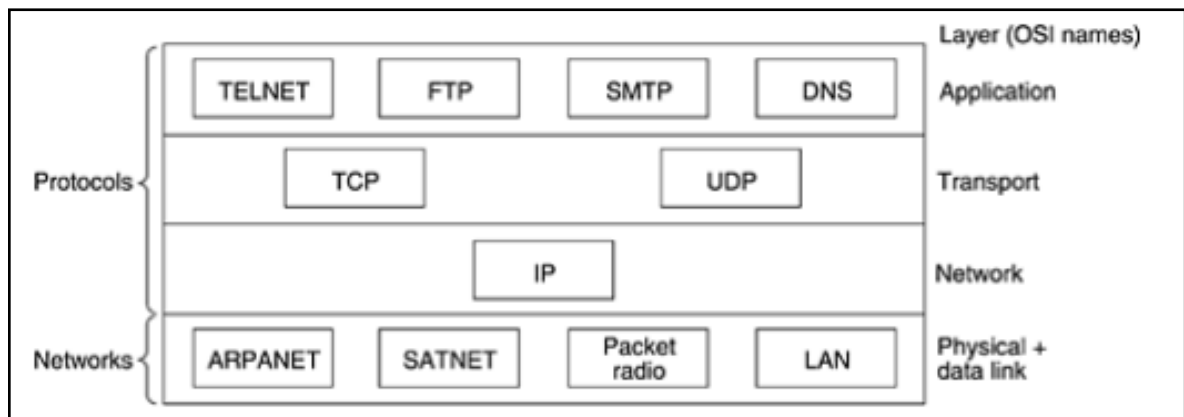
2 FUNDAMENTAÇÃO TEÓRICA

Abaixo são descritos conceitos, métodos, técnicas, tecnologias e ferramentas que serão empregadas na realização deste TCC.

2.1 TELNET

Segundo Torres (2007) o TCP/IP é um conjunto de protocolos distribuídos em quatro camadas. Cada tipo de programa se comunica com um protocolo de aplicação diferente, dependendo da finalidade do programa. O protocolo na camada de aplicação se comunica com um protocolo na camada de transporte, normalmente o TCP. O protocolo na camada de transporte se comunica com o protocolo da camada internet, o IP. O protocolo IP se comunica com o protocolo Interface de Rede.

Telnet é um protocolo de terminal virtual. Este protocolo está localizado dentro da camada de aplicação (Figura 2) do modelo TCP/IP. O protocolo de terminal virtual permite que um usuário de um computador conecte-se a uma máquina distante e trabalhe nela (TANENBAUM, 2003, p. 46).



Fonte: Tanenbaum (2003, p. 46).

Figura 2 - Protocolos no modelo TCP/IP inicial

2.2 ROUTEROS

O RouterOS é um SO desenvolvido em 1997, pela empresa Mikrotik para roteadores. Fornece uma grande estabilidade, controles e flexibilidade para todos os tipos de interfaces de dados e de roteamento (MIKROTIK, 2012a). RouterOS é um SO independente, baseado no *kernel* do Linux v2.6 (MIKROTIK, 2010).

O Mikrotik RouterOS ficou muito conhecido como Mikrotik pela comunidade de software livre e provedores de internet a rádio (DEUS, 2011).

O RouterOS é o sistema oficial pré-instalado nos minicomputadores RouterBoards, e que também pode ser instalado em um computador (plataforma x86) a fim de transformá-lo em um roteador avançado (LUCA, 2010, p. 51).

Segundo Rocha e Paiva (2011), RouterOS suporta dois tipos de roteamento: dinâmico e estático. No estático as rotas são criadas pelo usuário e no dinâmico as rotas são geradas automaticamente.

Os principais recursos do RouterOS são (MIKROTIK, 2010):

- a) roteamento estático;
- b) roteamento dinâmico com suporte aos protocolos *Routing Information Protocol* (RIP), *Open Shortest Path First* (OSPF), *Border Gateway Protocol* (BGP) e *Multi Protocol Label Switching* (MPLS);
- c) *firewall tasteful layer7* com filtro de pacotes *peer-to-peer*;
- d) controle de qualidade do serviço (*Quality of Service - QoS*);
- e) agrupamento de interfaces – *bonding/etherchannel*;
- f) *Wireless Distribution System* (WDS) e *Access Point (AP)* virtual;
- g) monitoramento *Simple Network Management Protocol* (SNMP);
- h) portal *hot spot* para autenticação via web por nome/senha;
- i) suporte a *Remote Authentication Dial In User Service* (RADIUS) e autenticação 802.1x;
- j) suporte a *Virtual Private Network* (VPN) – *Internet Protocol SECURITY* (IPSEC), *Point-to-Point Tunneling Protocol* (PPTP), entre outros.

2.3 ROUTERBOARD

RouterBoard é uma plataforma de hardware criada pela empresa MikroTik em 2002. São roteadores utilizados por provedores e grandes corporações ao redor do mundo (MIKROTIK, 2012a).

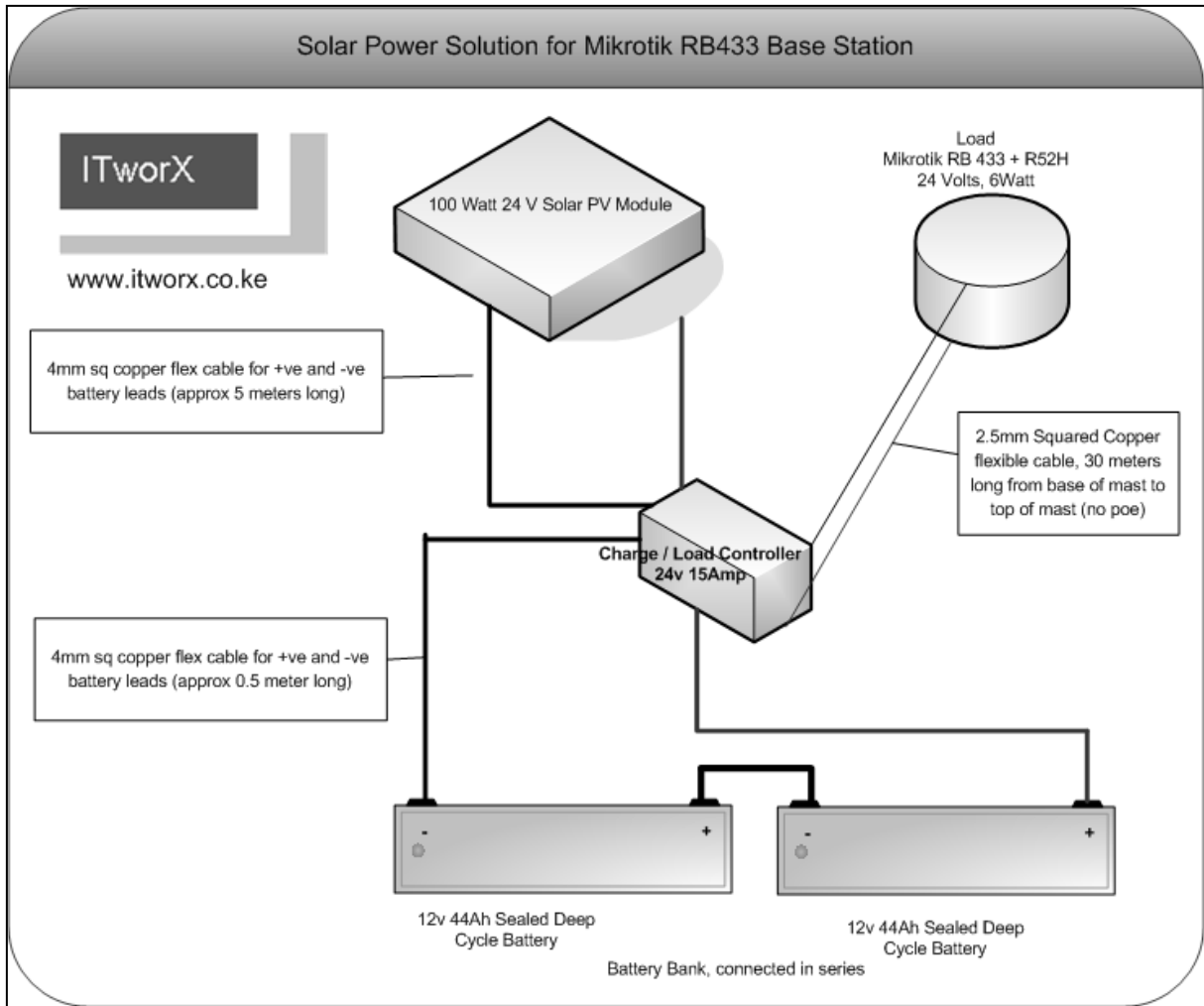
Seu formato e padrão fazem do seu uso mais adequado a aplicações *outdoor*, montado em caixas herméticas e instalados em torres ou quadros de função (IMASTERS, 2009).

Uma das formas de administrar estes roteadores é através de comandos no terminal virtual Telnet (MICHIGAN, 2011, p. 2).

Geralmente estes roteadores são alimentados sem qualquer ligação à rede elétrica municipal, utilizando baterias e painéis solares. Algumas razões pelas quais isso pode ser necessário são (PEDRO, 2011):

- a) não há fornecimento de eletricidade de rede onde se está construindo uma estação;
- b) a eletricidade da rede não é confiável onde se está colocando a estação e não se tem qualquer gerador automático;
- c) tem acesso à rede elétrica, mas acha que a energia solar é muitas vezes mais estável para o RouterBoard, podendo começar a reduzir os problemas;
- d) não se quer pagar pela energia, para o proprietário que está hospedando a estação.

Segundo Pedro (2011) o desenho apresentado na Figura 3 ilustra o projeto de uma solução para a RouterBoard ser alimentada com energia solar.



Fonte: Pedro (2011).

Figura 3 – Uma solução para a RouterBoard ser alimentada com energia solar

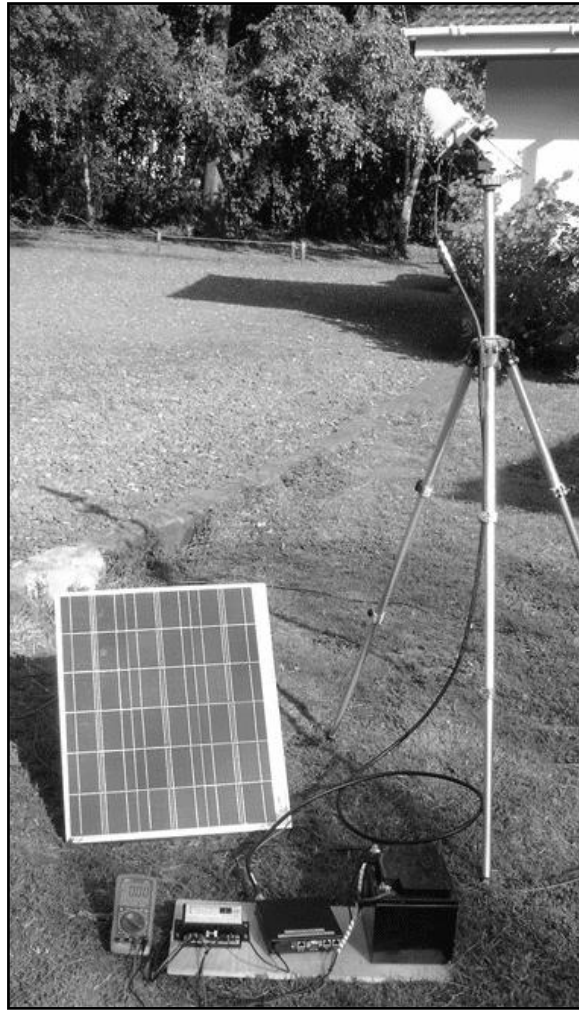
A lista de equipamentos utilizados na solução são apresentadas na Tabela 1.

Descrição do equipamento	Quantidade
Painel solar, 40 Watt, 12 Volt	2
Controlador de carga, 15Amp, 24 Volt	1
Ciclo de 44 ampères hora de profundidade, selada de chumbo-ácido, 12 Volt	2
Núcleo de cobre, 2,5 mm	30
Cabo de cobre single core, metade vermelho, metade preto	20
DC Plug Power (para ir para a RouterBoard)	1
Caixa de aço exterior para conter as baterias e controlador	1
Uma armação de aço para montar o painel	1
Muitas Abraçadeiras	?
Vários terminais de cabo e terminadores	?

Fonte: adaptado de Pedro (2011).

Tabela 1 – Equipamentos utilizados para a RouterBoard ser alimentada com energia solar

A Figura 4 apresenta a foto do resultado final da solução para a RouterBoard ser alimentada com energia solar.



Fonte: Pedro (2011).

Figura 4 – Foto da solução para a RouterBoard ser alimentada com energia solar
No item 2.3.1 é apresentado um modelo de RouterBoard, o RB433AH.

2.3.1 RB433AH

O RouterBoard RB433AH é um AP/roteador criado pela empresa MikroTik. Suas principais características são: processador de alto desempenho Atheros 680MHz, memória de 128 MB, 3 portas *fast* Ethernet 10/100 Mbit/s e RouterOS *level 5* (NOVANETWORK, 2012).

O Quadro 1 apresenta mais detalhes sobre a RB433AH.

CPU	Atheros AR7161 680MHz network processor
Memory	128MB DDR SDRAM onboard memory
Boot loader	RouterBOOT
Data storage	64MB onboard NAND memory chip and microSD
Ethernet	Three 10/100 Mbit/s Ethernet ports with Auto-MDI/X
miniPCI	Three MiniPCI Type IIIA/IIIB slots
Extras	Reset switch, Beeper
Serial port	One DB9 RS232C asynchronous serial port
LEDs	Power, NAND activity, 5 user LEDs
Power options	Power over Ethernet: 10..28V DC (except power over datalines). Power jack: 10..28V DC. Voltage monitor.
Dimensions	10.5 cm x 15 cm, 137 grams
Power consumption	~3W without extension cards, maximum - 25 W, 16W output to cards
Operating System	MikroTik RouterOS v3, Level5 license

Fonte: Mikrotik (2011).

Quadro 1 - Detalhes da RB433AH

A Figura 5 apresenta uma foto da RB433AH.

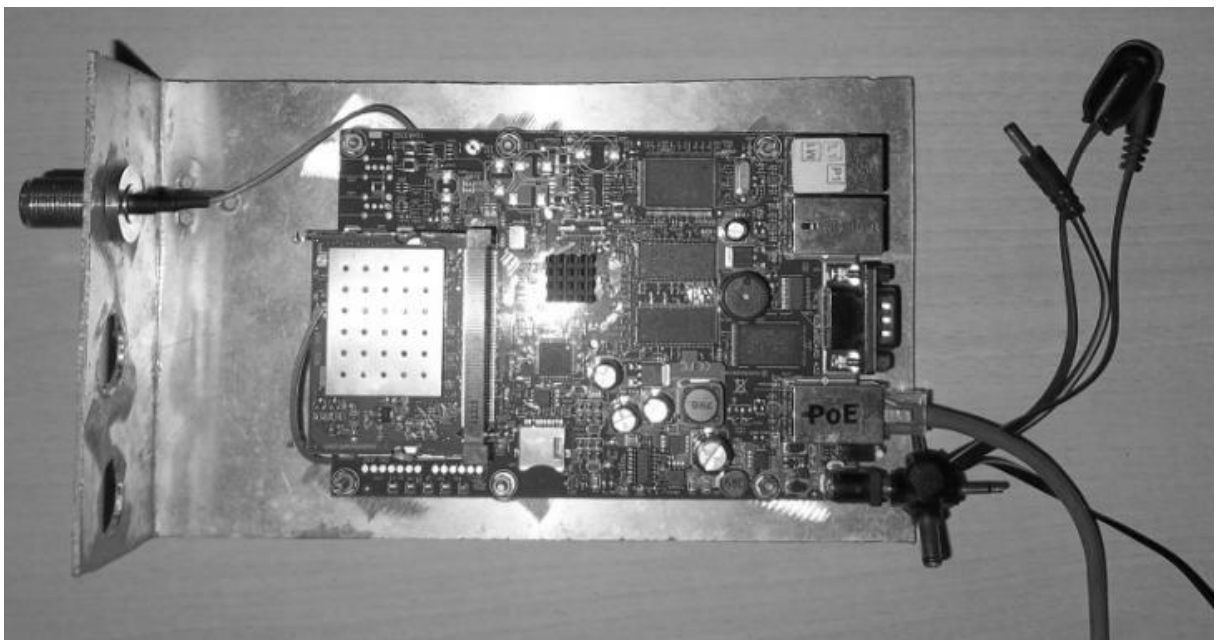


Figura 5 – Foto da RB433AH

2.4 GERÊNCIA DE REDES

Pinheiro (2006) afirma que o gerenciamento de rede pode ser definido como a coordenação de recursos materiais e/ou lógicos, assegurando, na medida do possível, confiabilidade, tempos de resposta aceitáveis e segurança das informações.

O modelo clássico de gerenciamento de rede pode ser resumido em três etapas (PINHEIRO, 2006):

- a) coleta de dados: um processo, em geral automático, que consiste de monitoração sobre os recursos gerenciados;
- b) diagnóstico: consiste no tratamento e análise realizados a partir dos dados coletados. O computador de gerenciamento executa uma série de procedimentos (por intermédio de um operador ou não) com o intuito de determinar a causa do problema representado no recurso gerenciado;
- c) ação ou controle: uma vez diagnosticado o problema, cabe uma ação, ou controle, sobre o recurso, caso o evento não tenha sido passageiro (incidente operacional).

Uma parte significativa do processo de gerenciamento baseia-se na aquisição de informações sobre a rede, sendo as mais importantes aquelas relativas a erros, falhas e outras condições excepcionais. Tais dados devem ser armazenados em forma bruta, sendo importante definir os valores aceitáveis como limiares de tolerância que, quando ultrapassados, determinam uma sinalização para pedir intervenção de um operador, ou o início de uma operação corretiva. Tais limites não são necessariamente absolutos, tais como a taxa de erros num enlace de dados, sendo necessário dispor de estatísticas de erros em função do tráfego existente (PINHEIRO, 2006).

Segundo Pinheiro (2006), monitoração e controle são os tipos mais básicos de tarefas de gerenciamento de uma rede. A monitoração consiste na observação periódica dos objetos gerenciados. A partir da monitoração, o gerente tem conhecimento do estado da rede e, desta forma, pode efetuar operações de controle sobre a mesma.

2.5 ANÁLISE LÉXICA E SINTÁTICA

Segundo Price e Toscani (2001, p. 7), análise léxica é uma das fases do processo de tradução de linguagens de programação. Seu objetivo principal é identificar sequências de caracteres que constituem unidades léxicas (“*tokens*”). O analisador léxico lê o texto, caractere a caractere, verificando se os caracteres lidos pertencem ao alfabeto da linguagem, identificando *tokens* e desprezando comentários e brancos desnecessários. Um *token* é formado por um par ordenado (valor, classe). A classe representa a natureza da informação contida em valor.

Segundo Price e Toscani (2001, p. 7-8), além da identificação dos *tokens*, o analisador léxico inicia a construção da tabela de símbolos e envia mensagens de erro caso identifique unidades léxicas não aceitas pela linguagem em questão.

Segundo Price e Toscani (2001, p. 9), análise sintática é uma das fases do processo de tradução de linguagens de programação. Seu objetivo principal é verificar se a estrutura gramatical do texto está correta. O analisador sintático identifica sequências de símbolos que constituem estruturas sintáticas, através de uma varredura ou “*parsing*” da representação interna (cadeia de *tokens*) do texto. O analisador sintático produz uma estrutura em árvore, chamada árvore de derivação, que exhibe a estrutura sintática do texto, resultante da aplicação das regras gramaticais da linguagem.

2.6 GALS

Segundo Gesser (2003, p. 4), o Gerador de Analisadores Léxicos e Sintáticos (GALS) é uma ferramenta para geração automática de analisadores léxicos e sintáticos. Foi desenvolvido para ser uma ferramenta com propósitos didáticos, mas com características profissionais e também pode ser utilizado tanto no auxílio aos alunos da cadeira de construção de compiladores quanto em outros projetos que necessitem processamento de linguagens.

A Figura 6 apresenta a tela inicial do GALS.

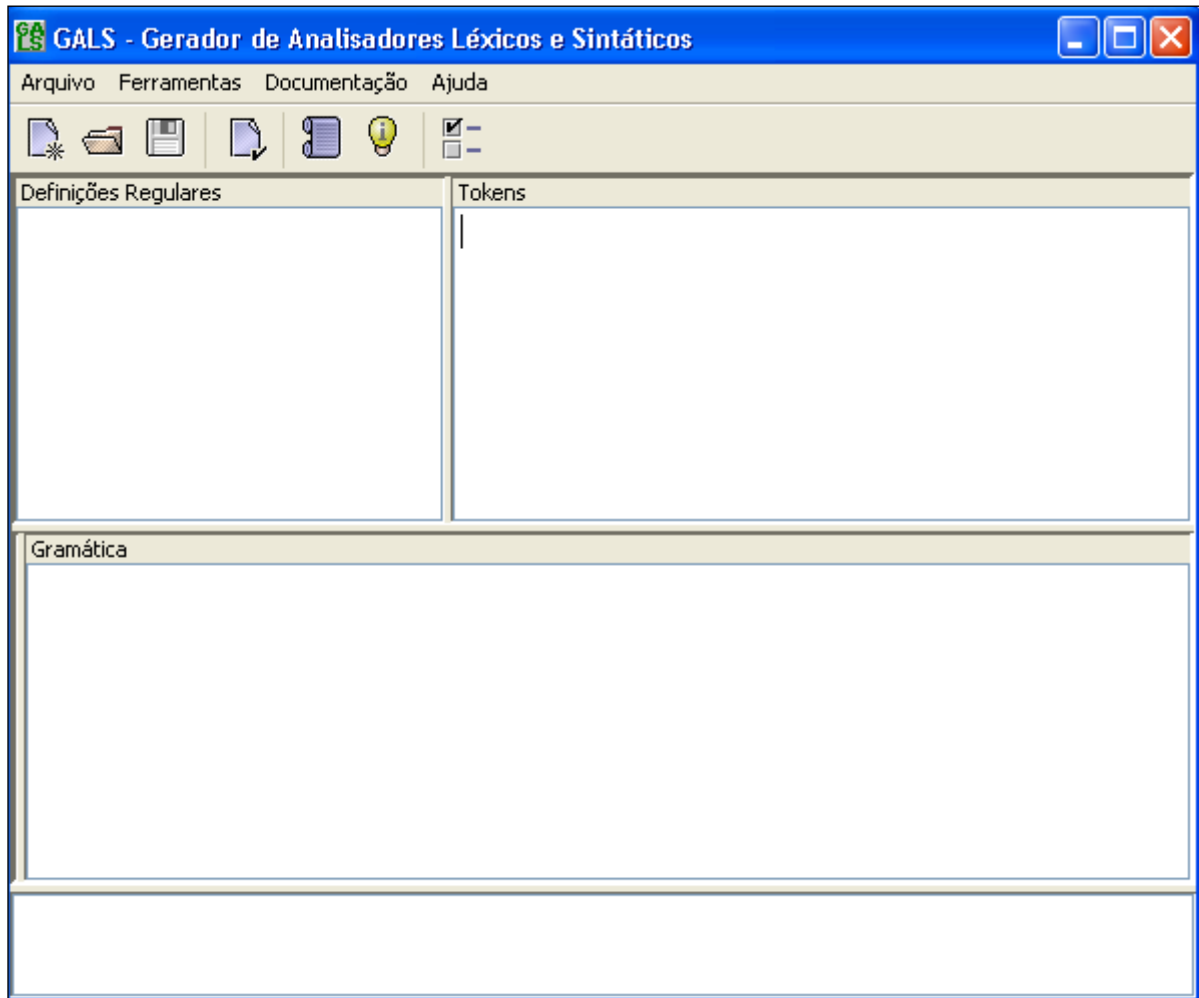


Figura 6 – Tela inicial do GALS

Segundo Gesser(2003, p. 40), na tela apresentada na Figura 6 são definidos aspectos léxicos e sintáticos de uma forma conjunta. Porém, dependendo da escolha que se faça, para gerar apenas o analisador léxico ou o sintático, o ambiente é rearranjado de modo adequado.

2.7 ADOBE FLEX

Segundo Adobe (2012e), Adobe Flex é um poderoso *framework* de código aberto que permite construir aplicações móveis para dispositivos iOS, Android, BlackBerry e Tablet OS, bem como aplicações para navegador e desktop utilizando o mesmo modelo de programação, ferramentas e base de código.

Para Adobe (2012d), Adobe Flex é a primeira solução e mais completa para *Rich Internet Application* (RIA), ela combina a capacidade de resposta e interatividade de

aplicativos *desktop* com o amplo alcance e facilidade de distribuição de aplicações *web*.

Segundo Busch e Koch (2009, p. 7), as aplicações *web* tradicionais possuem grandes limitações quanto à usabilidade e interatividade de suas interfaces. Entre as limitações estão a necessidade de recarregamento da página para obtenção de dados do servidor e comunicação síncrona. As RIAs oferecem interfaces com tempo de resposta menor, comunicação assíncrona, animações multimídia e aplicações mais robustas e ao mesmo tempo leves.

Flex possui diversos componentes acessíveis que aceleram o desenvolvimento de aplicações. Com estes componentes, os desenvolvedores podem criar RIAs rápida e facilmente, promovendo um elevado nível de acessibilidade (ADOBE, 2012d).

Segundo Schmitz (2011), ActionScript é a linguagem de programação utilizada juntamente com MXML para criar aplicações Flex. MXML é uma linguagem de marcação baseada em XML.

Para Schmitz (2011) a linguagem MXML é utilizada pelos desenvolvedores Flex principalmente para apresentar a interface do usuário, enquanto que, o ActionScript é utilizado para a lógica de negócio.

2.8 TRABALHOS CORRELATOS

Nesta seção são apresentadas três ferramentas que desempenham um papel semelhante ao trabalho proposto. No item 2.8.1 é apresentada a ferramenta Winbox (MIKROTIK, 2004), no item 2.8.2 a ferramenta Webfig (MIKROTIK, 2012d) e no item 2.8.3 a ferramenta The Dude (MIKROTIK, 2012e).

2.8.1 WINBOX

Winbox é uma interface gráfica para administração do Mikrotik, que funciona em Windows e Linux. Utiliza a porta TCP 8291 (MICHIGAN, 2011, p. 4).

Na Figura 7 é apresentada a tela inicial da Winbox, onde se informa as credenciais de acesso para uma RouterBoard.

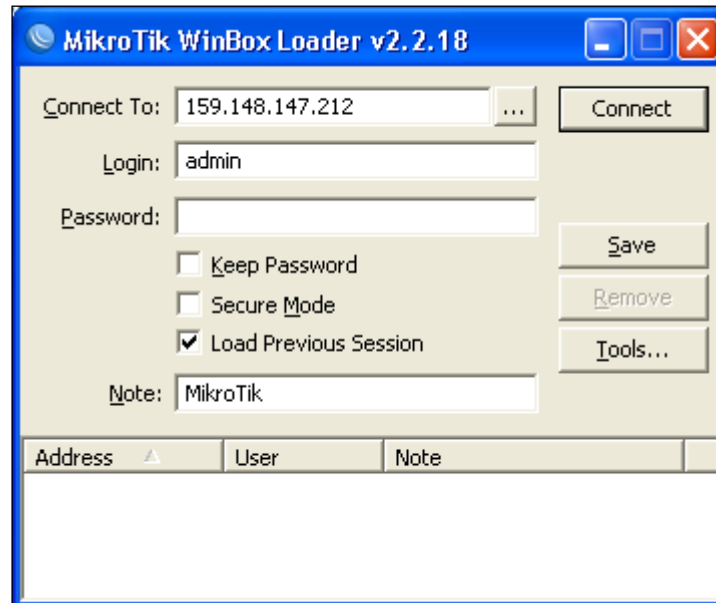


Figura 7 – Tela de *login* da WinBox

Na Figura 8 é apresentada uma das telas da Winbox, onde é possível monitorar a tensão da bateria para o equipamento que se está conectado.

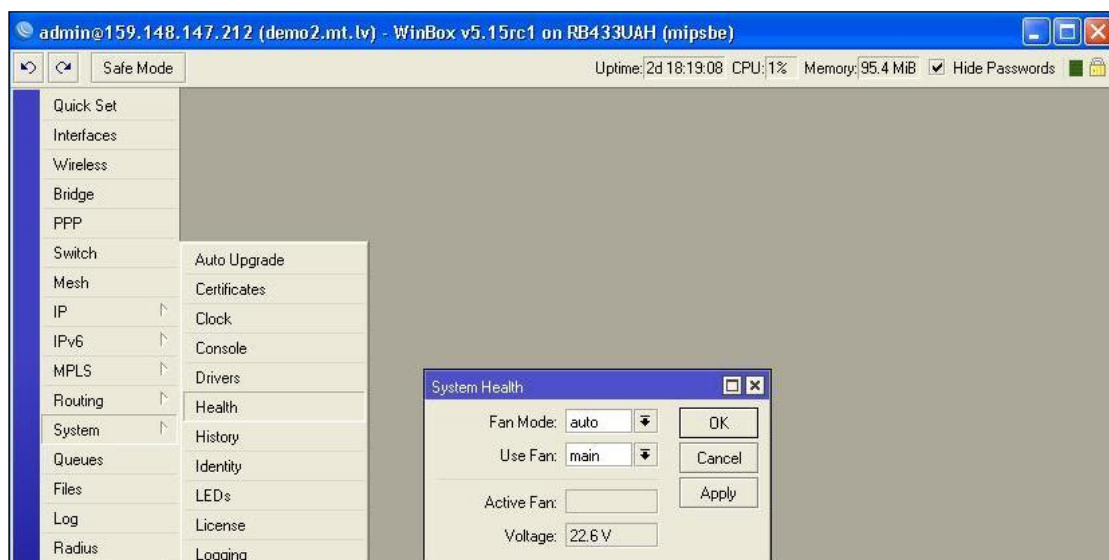


Figura 8 - Monitoramento da tensão da bateria de uma RouterBoard, através da WinBox

Na Figura 9 é apresentado o monitoramento da informação Overall Tx CCQ. Segundo Michigan (2011, p. 19) a informação Overall Tx CCQ é um valor em percentual que mostra a eficiência da banda de transmissão em relação à banda máxima disponível no link. Esse valor é calculado com base nos pacotes *wireless* que são retransmitidos no meio físico. Quanto mais retransmissões, menos a eficiência.

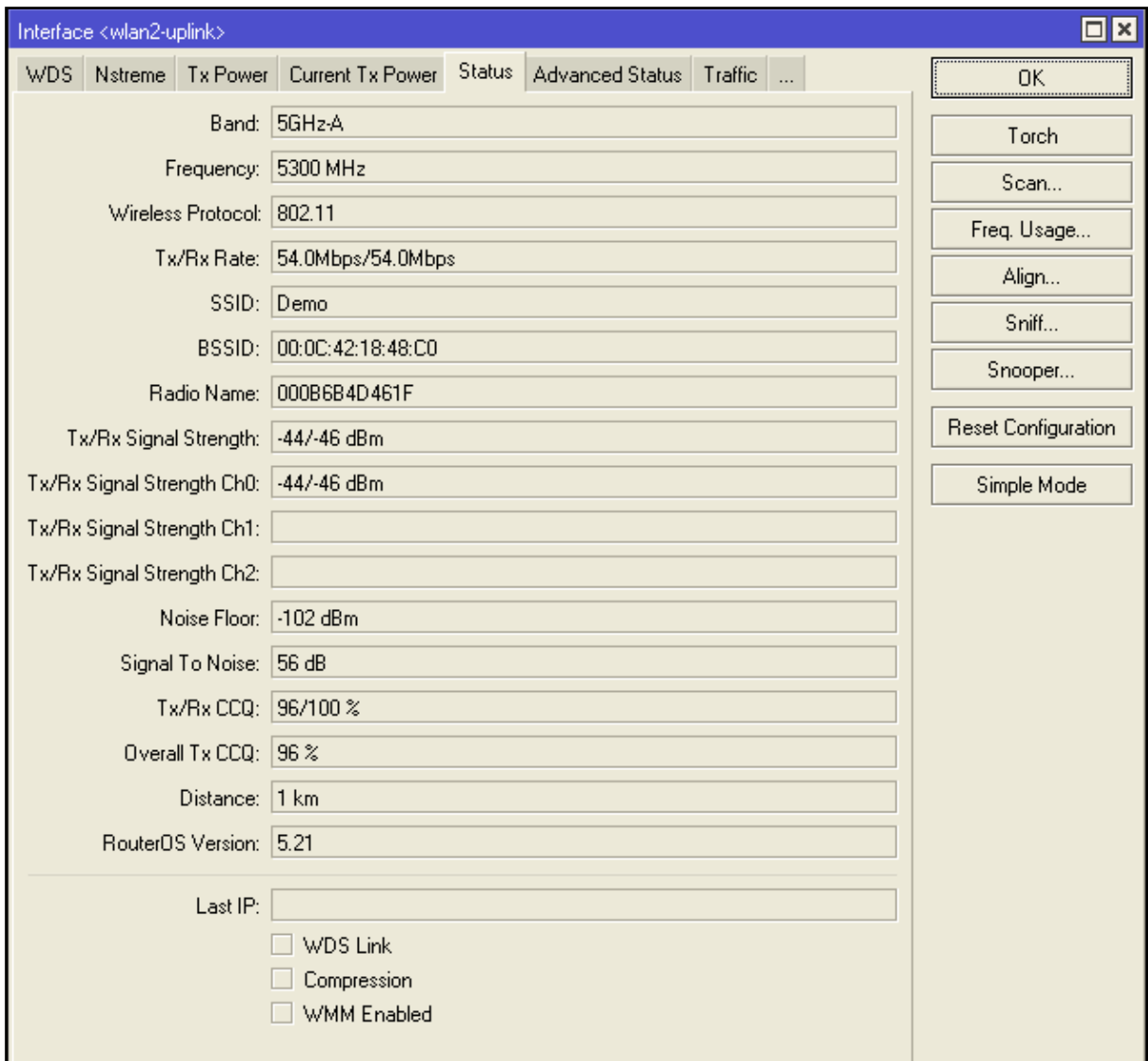


Figura 9 – Monitoramento da eficiência da banda de transmissão em relação à banda máxima disponível no link (Overall Tx CCQ)

2.8.2 WEBFIG

Segundo Rocha e Paiva (2011), Webfig é uma ferramenta Web de configuração do RouterOS que permite as mesmas funções dispostas na ferramenta Winbox e possui o *layout* muito parecido com Winbox. Para abrir sua interface tem-se que digitar o endereço IP do roteador no campo de endereços do navegador.

A Figura 10 apresenta a tela de *login* da ferramenta Webfig.

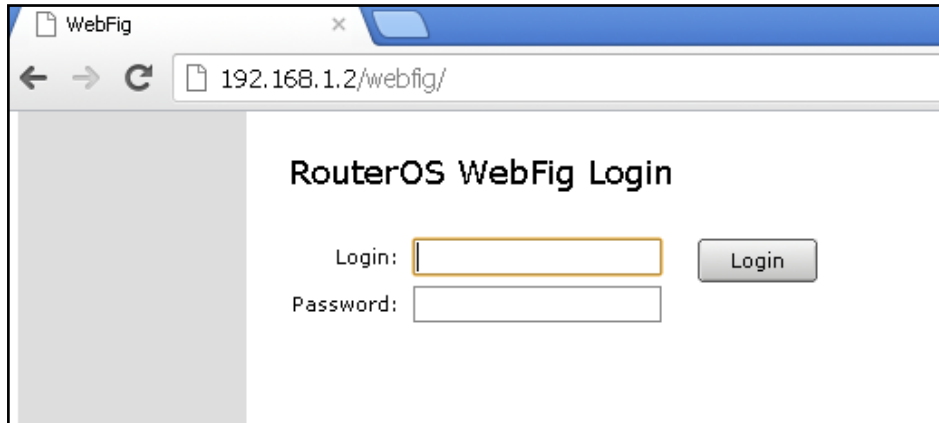


Figura 10 – Tela de *login* da ferramenta Winbox

Na Figura 11 é apresentado uma das telas da Webfig, onde é possível monitorar a tensão da bateria para o equipamento que se está conectando.



Figura 11 - Monitoramento da tensão da bateria de uma RouterBoard, através da Webfig

2.8.3 THE DUDE

The Dude é uma aplicação criada pela MikroTik. Ela permite controlar roteadores e alertar o administrador da rede em caso de algum serviço estar com problemas (MIKROTIK, 2010).

O painel de funções lista todas as funções que podem ser utilizadas. É permitido editar uma função ou escrever uma nova função (MIKROTIK, 2012b). Na Figura 12 é apresentado uma função que foi alterada e uma função que foi criada para buscar a tensão da bateria na RouterBoard.

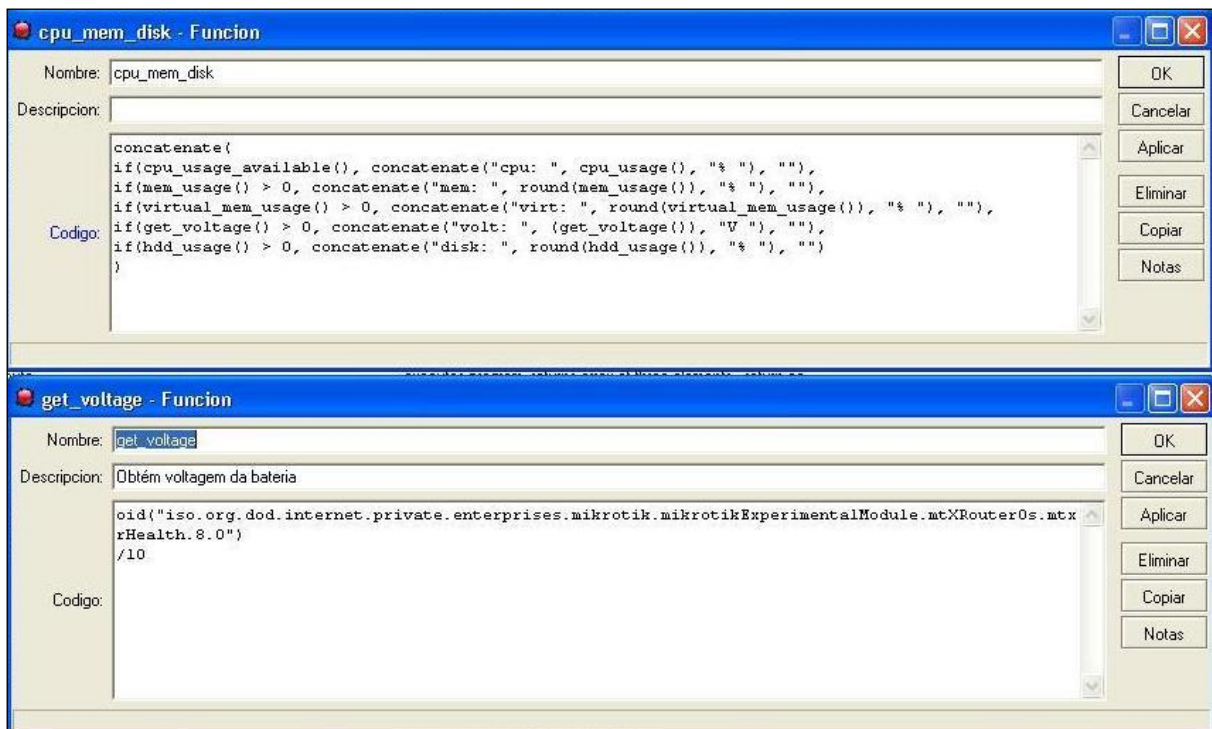


Figura 12 - Painel com as Funções criadas/alteradas na ferramenta The Dude

Há uma grande variedade de verificações dos serviços dos dispositivos pré-definidas (MIKROTIKI, 2012c). Na Figura 13 é apresentado a criação de uma verificação para monitorar a bateria e destacar os dispositivos que estiverem com a tensão menor ou igual a 11 volts.

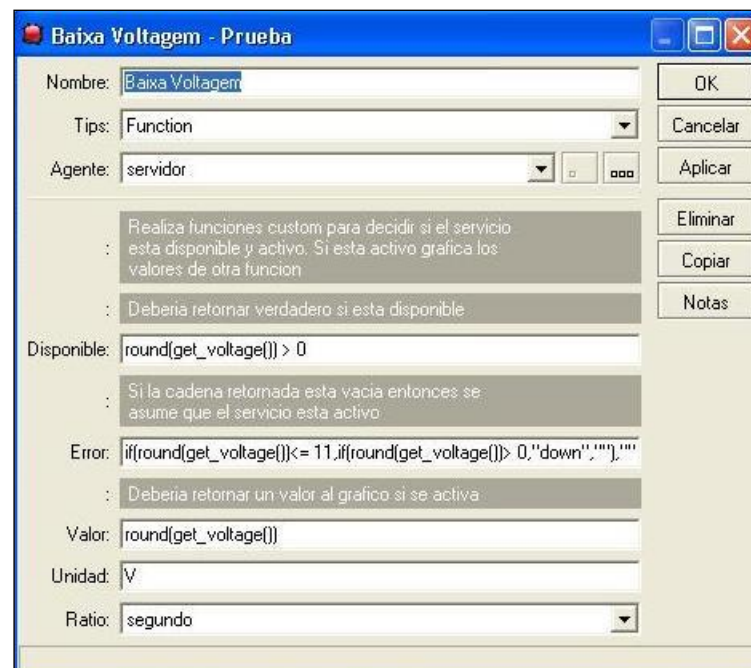


Figura 13 - Verificação da tensão

3 DESENVOLVIMENTO

Neste capítulo são apresentados os requisitos principais do problema a ser trabalhado, a especificação e a implementação do trabalho. Para finalizar são apresentados os resultados obtidos e uma comparação com os trabalhos correlatos.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O sistema possui os seguintes Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF):

- a) permitir o cadastro de equipamentos, informando marca, modelo, endereço *Internet Protocol (IP)*, *login* e senha para acessá-los (RF01);
- b) permitir o cadastro dos responsáveis pelo monitoramento. Ao efetuar ou editar o cadastro deve ser possível informar quais dispositivos são de responsabilidade do monitor em questão (RF02);
- c) permitir que o usuário informe o valor aceitável de algumas propriedades monitoradas nos equipamentos modelo RB433AH. Exemplos de valores são a tensão da bateria (em volts) com valor mínimo configurado em 11 e a qualidade do sinal enviada para os clientes (em %) com valor mínimo configurado em 90 (RF03);
- d) monitorar os equipamentos modelo RB433AH sobre a tensão da bateria e manter um histórico (RF04);
- e) monitorar os equipamentos modelo RB433AH sobre a qualidade do sinal enviada para os clientes e manter um histórico (RF05);
- f) enviar *e-mail* notificando os responsáveis pelo monitoramento, cujas propriedades monitoradas em um equipamento estejam fora dos valores aceitáveis. A notificação deve ser feita somente para os responsáveis pelo monitoramento do equipamento em questão (RF06);
- g) permitir que o usuário visualize no navegador a informação mais atual, descritas nos itens anteriores (RF07);
- h) permitir a configuração de quanto em quanto tempo será feito o monitoramento

(RF08);

- i) implementar o sistema web utilizando tecnologia *Java Enterprise Edition* (JEE) e o Flex (RNF01);
- j) ser compatível com os navegadores Google Chrome, Internet Explorer e Firefox (RNF02).

3.2 ESPECIFICAÇÃO

A especificação do sistema foi realizada com Orientação à Objetos (OO) utilizando a *Unified Modeling Language* (UML). Foram construídos os diagramas de casos de uso, classes, sequência e feito a modelagem de dados utilizando *Conceptual Data Model* (CDM) e *Physical Data Model* (PDM). Para auxiliar na construção dos diagramas foi utilizada a ferramenta *Enterprise Architect* (EA) e para auxiliar na modelagem de dados foi utilizada a ferramenta PowerDesigner 12.

3.2.1 DIAGRAMA DE CASOS DE USO

Um caso de uso (*Use Case* - UC) descreve a funcionalidade específica que um sistema deve desempenhar ou exibir, por meio da modelagem do diálogo que um usuário, um sistema externo ou outra entidade terá com o sistema a ser desenvolvido (PFLEEGER, 2004, p. 216). Segundo Bezerra (2002, p. 46), um caso de uso deve ser escrito na forma narrativa das interações que acontecem entre os elementos externos e o sistema.

Geralmente o caso de uso é apresentado como um desenho dos objetos envolvidos, mais um esboço resumido (o *script* do cenário) de como a função é realizada (PFLEEGER, 2004, p. 216). Segundo Bezerra (2002, p. 49), os casos de uso podem ser definidos de várias maneiras, uma delas pode ser através dos cenários.

Os atores identificados no sistema são:

- a) Usuário;
- b) Sistema Monitorador dos Equipamentos;
- c) Sistema Notificador.

O Usuário representa a pessoa que interage com a tela do sistema no navegador.

O Sistema Monitorador dos Equipamentos e o Sistema Notificador representam subsistemas do sistema desenvolvido que interagem com o sistema para realizar suas tarefas.

A Figura 14 apresenta o diagrama dos principais casos de uso do sistema.

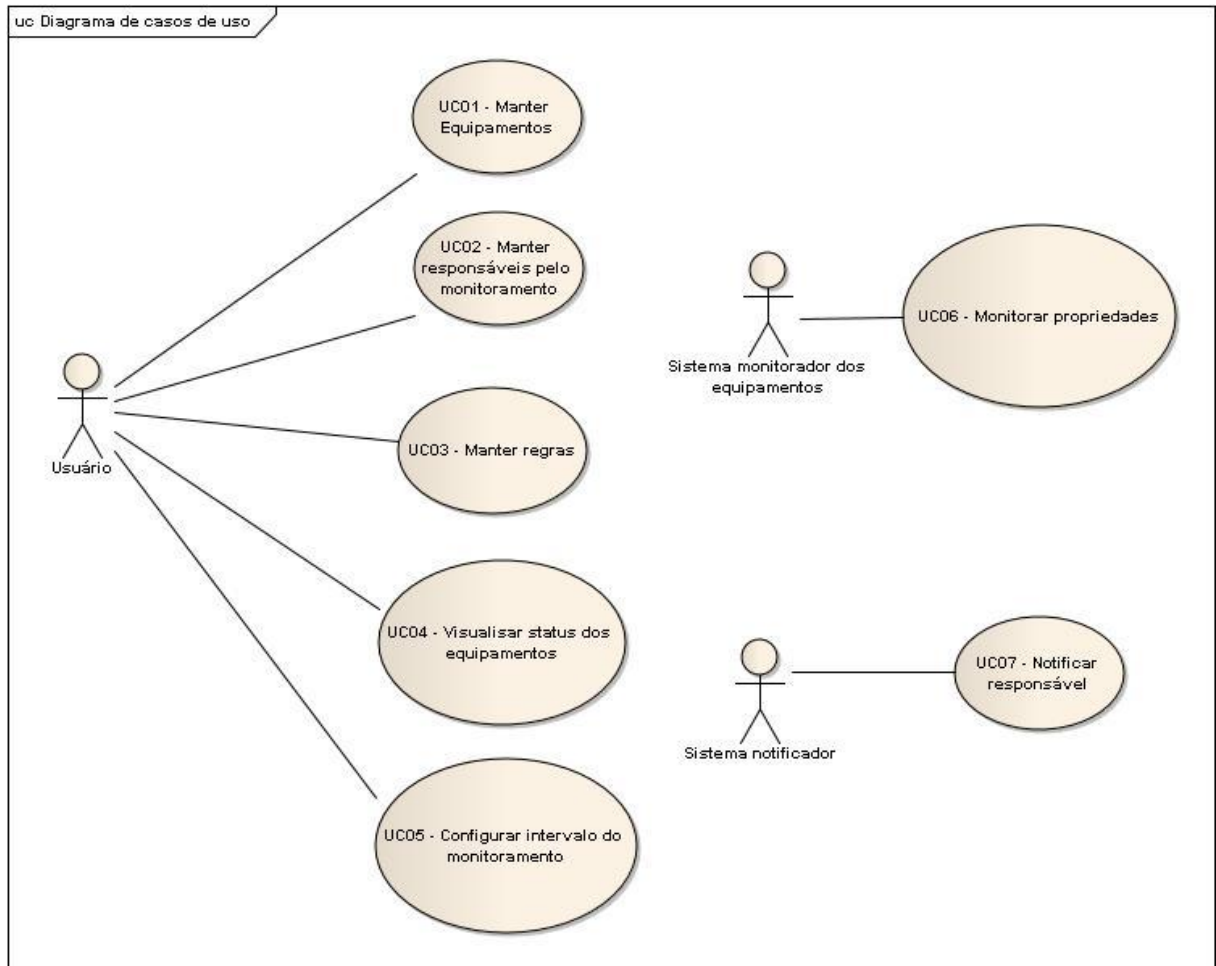


Figura 14 - Diagrama de casos de uso

No Quadro 2 é apresentado o *script* dos cenários do caso de uso manter equipamentos. Nesse caso de uso o Usuário abre a visualização dos equipamentos e pode optar por alterar, adicionar ou excluir cadastro. Este caso de uso está associado ao RF01.

Ator	Usuário
Cenário Principal	1. O Usuário solicita a visualização dos cadastros de equipamentos.
	2. O sistema abre a tela de visualização dos equipamentos.
	3. O Usuário pode optar por adicionar, excluir ou editar cadastro.
	4. O Usuário fecha a tela de visualização dos equipamentos.
Adicionar Cadastro	No passo 3 do Cenário Principal, caso o Usuário escolha adicionar:
	3.1. O Usuário clica no botão Adicionar.
	3.2. O sistema abre a tela para informar os dados do novo equipamento.
	3.3. O Usuário informa os dados do novo equipamento.
	3.4. O Usuário clica no botão OK.
	3.5. O sistema valida as informações do novo equipamento.
	3.6. O sistema registra o novo equipamento.
	3.7. Vai para o passo 3 do Cenário Principal.
Editar Cadastro	No passo 3 do Cenário Principal, caso o Usuário escolha editar:
	3.1. O Usuário clica no botão editar do equipamento.
	3.2. O sistema abre a tela para alterar os dados do equipamento.
	3.3. O Usuário altera os dados do novo equipamento.
	3.4. O Usuário clica no botão Salvar.
	3.5. O sistema valida as informações do equipamento.
	3.6. O sistema grava a alteração.
	3.7. Vai para o passo 3 do Cenário Principal.
Excluir Cadastro	No passo 3 do Cenário Principal, caso o Usuário escolha excluir:
	3.1. O Usuário clica no botão excluir do equipamento.
	3.2. O sistema abre uma caixa de diálogo perguntando se o Usuário deseja excluir o equipamento.
	3.3. O Usuário pode optar por sim ou não.
	3.4. Vai para o passo 3 do Cenário Principal.
Confirmar Exclusão	No passo 3.3 do cenário Excluir Cadastro, caso o Usuário escolha sim:
	3.3.1. O Usuário clica no botão Sim.
	3.3.2. O sistema fecha a caixa de diálogo.
	3.3.2. O sistema exclui o equipamento.
	3.3.4. Vai para o passo 3 do Cenário Principal.
Não Confirmar Exclusão	No passo 3.3 do cenário Excluir Cadastro, caso o Usuário escolha não:
	3.3.1. O Usuário clica no botão Não.
	3.3.2. O sistema fecha a caixa de diálogo.
	3.3.3. Vai para o passo 3 do Cenário Principal.
Exceção 01	No passo 3.5 do cenário Adicionar Cadastro, caso algum dado for inválido:
	3.5.1. Informa ao usuário que existem dados inválidos.
	3.5.2. Vai para o passo 3.3 do cenário Adicionar Cadastro.
Exceção 02	No passo 3.5 do cenário Editar Cadastro, caso algum dado for inválido:
	3.5.1. O sistema informa ao Usuário sobre os dados inválidos.
	3.5.2. Vai para o passo 3.3 do cenário Editar Cadastro.

Quadro 2 - UC01 - Manter equipamentos

No Quadro 3 é apresentado o *script* dos cenários do caso de uso manter responsáveis pelo monitoramento. Nesse caso de uso o Usuário abre a visualização dos responsáveis pelo monitoramento e pode optar por alterar, adicionar ou excluir cadastro. Este caso de uso está

associado ao RF02.

Ator	Usuário
Cenário Principal	1. O Usuário solicita a visualização dos responsáveis.
	2. O sistema abre a tela de visualização solicitada.
	3. O Usuário pode optar por adicionar, excluir ou editar cadastro.
	4. O Usuário fecha a tela de visualização dos responsáveis.
Adicionar Cadastro	No passo 3 do Cenário Principal, caso o Usuário escolha adicionar:
	3.1. O Usuário clica no botão adicionar.
	3.2. O sistema abre a tela para informar os dados.
	3.3. O Usuário informa nome e e-mail do novo responsável.
	3.4. O Usuário associa os equipamentos para o novo responsável.
	3.5. O Usuário clica no botão OK.
	3.6. O sistema valida as informações do novo responsável.
	3.7. O sistema registra o novo responsável.
	3.9. Vai para o passo 3 do Cenário Principal.
Editar Cadastro	No passo 3 do Cenário Principal, caso o Usuário escolha editar:
	3.1. O Usuário clica no botão editar do responsável.
	3.2. O sistema abre a tela para alterar os dados do responsável.
	3.3. O Usuário altera os dados do responsável e clica no botão Salvar.
	3.4. O sistema valida as informações do responsável.
	3.6. Vai para o passo 3 do Cenário Principal.
Excluir Cadastro	No passo 3 do Cenário Principal, caso o Usuário escolha excluir:
	3.1. O Usuário clica no botão excluir do responsável.
	3.2. O sistema exibe um alerta para confirmar exclusão.
	3.4. Vai para o passo 3 do Cenário Principal.
Confirmar Exclusão	No passo 3.3 do cenário Excluir Cadastro, caso o Usuário escolha sim:
	3.3.1. O Usuário clica no botão Sim.
	3.3.2. O sistema fecha a caixa de diálogo e exclui o responsável.
Não Confirmar Exclusão	No passo 3.3 do cenário Excluir Cadastro, caso o Usuário escolha não:
	3.3.1. O Usuário clica no botão Não.
	3.3.2. O sistema fecha a caixa de diálogo.
Exceção 01	No passo 3.6 do cenário Adicionar Cadastro, caso algum dado for inválido:
	3.5.1. Informa ao usuário que existem dados inválidos.
Exceção 02	No passo 3.4 do cenário Editar Cadastro, caso algum dado for inválido:
	3.5.1. O sistema informa ao Usuário sobre os dados inválidos.

Quadro 3 - UC02 - Manter responsáveis pelo monitoramento

No Quadro 4 é apresentado o *script* dos cenários do caso de uso manter regras. Nesse caso de uso o Usuário abre a visualização das regras e pode optar por alterar, adicionar ou excluir cadastro. Este caso de uso está associado ao RF03.

Ator	Usuário
Cenário Principal	1. O Usuário solicita a visualização das regras.
	2. O sistema abre a tela de visualização solicitada.
	3. O Usuário pode optar por adicionar, excluir ou editar cadastro.
	4. O Usuário fecha a tela de visualização das regras.
Adicionar Cadastro	No passo 3 do Cenário Principal, caso o Usuário escolha adicionar:
	3.1. O Usuário clica no botão Adicionar.
	3.2. O sistema abre a tela para informar os dados.
	3.3. O Usuário seleciona a qual mensagem esta regra pertence.
	3.3. O Usuário informa nome do parâmetro, tipo de validação e valor do parâmetro.
	3.4. O Usuário clica no botão Salvar.
	3.5. O sistema valida as informações da nova regra.
	3.6. O sistema registra a nova regra.
3.9. Vai para o passo 3 do Cenário Principal.	
Editar Cadastro	No passo 3 do Cenário Principal, caso o Usuário escolha editar:
	3.1. O Usuário clica no botão editar da regra.
	3.2. O sistema abre a tela para alterar os dados da regra.
	3.3. O Usuário altera os dados da regra e clica no botão Salvar.
	3.5. O sistema valida as informações da regra.
	3.6. O sistema grava a alteração da regra.
3.7. Vai para o passo 3 do Cenário Principal.	
Excluir Cadastro	No passo 3 do Cenário Principal, caso o Usuário escolha excluir:
	3.1. O Usuário clica no botão excluir da regra.
	3.2. O sistema exibe um alerta para confirmar exclusão.
	3.3. O Usuário pode optar por sim ou não.
3.4. Vai para o passo 3 do Cenário Principal.	
Confirmar Exclusão	No passo 3.3 do cenário Excluir Cadastro, caso o Usuário escolha a opção sim:
	3.3.1. O Usuário clica no botão Sim.
	3.3.2. O sistema fecha a caixa de diálogo e exclui a regra.
	3.3.4. Vai para o passo 3 do Cenário Principal.
Não Confirmar Exclusão	No passo 3.3 do cenário Excluir Cadastro, caso o Usuário escolha a opção não:
	3.3.1. O Usuário clica no botão Não.
	3.3.2. O sistema fecha a caixa de diálogo.
3.3.3. Vai para o passo 3 do Cenário Principal.	
Exceção 01	No passo 3.5 do cenário Adicionar Cadastro, caso algum dado for inválido:
	3.5.1. Informa ao usuário que existem dados inválidos.
	3.5.2. Vai para o passo 3.3 do cenário Adicionar Cadastro.
Exceção 02	No passo 3.5 do cenário Editar Cadastro, caso algum dado for inválido:
	3.5.1. O sistema informa ao Usuário sobre os dados inválidos.
	3.5.2. Vai para o passo 3.3 do cenário Editar Cadastro.

Quadro 4 - UC03 - Manter regras

No Quadro 5 é apresentado o *script* dos cenários do caso de uso visualizar status dos equipamentos. Nesse caso de uso o Usuário abre a tela de monitoramento e monitora os

valores dos parâmetros, correspondentes as regras que foram cadastradas. Este caso de uso está associado ao RF01. Este caso de uso está associado ao RF07.

Ator	Usuário
Cenário Principal	1. O Usuário solicita a visualização do monitoramento.
	2. O sistema abre a tela de monitoramento.
	3. O sistema verifica se existem regras cadastradas.
	4. O sistema mostra os dados para o Usuário de acordo com as regras cadastradas.
	5. O Usuário monitora os equipamentos.
	4. O Usuário fecha a tela de monitoramento.
Exceção 01	No passo 3 do Cenário Principal, caso o sistema não encontrar regras cadastradas:
	3.1. O sistema informa ao usuário que não existem regras cadastradas.
	3.2. O sistema fecha a tela de monitoramento.

Quadro 5 – UC04 – Visualizar *status* dos equipamentos

No Quadro 6 é apresentado o *script* dos cenários do caso de uso configurar intervalo do monitoramento. Nesse caso de uso o Usuário configura de quanto em quanto tempo será enviado as mensagens para os equipamentos. Este caso de uso está associado ao RF08.

Ator	Usuário
Cenário Principal	1. O Usuário solicita a tela de configuração.
	2. O sistema abre a tela de configuração.
	3. O Usuário informa as propriedades do sistema
	4. O Usuário clica no botão Salvar.
	5. O sistema valida as informações.
	6. O sistema grava as informações.
	7. O Usuário fecha a tela de configuração.
Exceção 01	No passo 5 do Cenário Principal, caso o sistema invalidar as informações:
	3.1. O sistema informa ao usuário que existem informações inválidas.
	3.2. Vai para o passo 3 do Cenário Principal.

Quadro 6 – UC05 – Configurar intervalo do monitoramento

No Quadro 7 é apresentado o *script* dos cenários do caso de uso monitorar propriedades. Nesse caso de uso o Sistema Monitorador dos Equipamentos envia mensagens para os equipamentos, e realiza o monitoramento dos valores de cada parâmetro das regras que foram cadastradas. Este caso de uso está associado aos RF04 e RF05.

Ator	Sistema Monitorador dos Equipamentos
Cenário Principal	1. O Sistema Monitorador dos Equipamentos solicita ao sistema as mensagens habilitadas para serem enviadas.
	2. O sistema retorna para o Sistema Monitorador dos Equipamentos uma lista de mensagens habilitadas para serem enviadas.
	3. O Sistema Monitorador dos Equipamentos verifica se existem mensagens na lista.
	4. O Sistema Monitorador dos Equipamentos aguarda um tempo.
Monitorar Propriedades	No passo 3 do Cenário Principal, caso o Sistema Monitorador dos Equipamentos observar que a lista possui mensagens:
	1.1. O Sistema Monitorador dos Equipamentos solicita ao sistema os equipamentos habilitados para enviar as mensagens.
	1.2. O sistema retorna para o Sistema Monitorador dos Equipamentos uma lista de equipamentos habilitados.
	1.3. O Sistema Monitorador dos Equipamentos verifica se existem equipamentos na lista.
Enviar Mensagens	No passo 1.3 do cenário Monitorar Propriedades, caso o Sistema Monitorador dos Equipamentos encontre equipamentos, para cada mensagem:
	1.1.1. O Sistema Monitorador dos Equipamentos envia a mensagem para cada equipamento.
	1.1.2. O Sistema Monitorador dos Equipamentos aguarda a resposta da mensagem para cada equipamento.
	1.1.3. O Sistema Monitorador dos Equipamentos verifica a estrutura da resposta da mensagem, realizando o <i>parser</i> da mensagem.
	1.1.4. O Sistema Monitorador dos Equipamentos verifica se a mensagem possui regras associada.
	1.1.5. Vai para o passo 1 do Cenário Principal.
Conferir Regras	No passo 1.1.4 do cenário Enviar Mensagens, caso o Sistema Monitorador dos Equipamentos encontre regras para a
	1.1.4.1. O Sistema Monitorador dos Equipamentos confere os parâmetros da mensagem com as regras do sistema.
Exceção 01	No passo 1.1.3 do cenário Enviar Mensagens, caso a estrutura da mensagem não for válida:
	1.1.3.1. O Sistema Monitorador dos Equipamentos gera uma notificação com erro no <i>parser</i> da mensagem.
Exceção 02	No passo 1.1.4.1 do cenário Conferir Regras, caso o Sistema monitorador dos equipamentos encontre parâmetros com o valor que não confere com as regras:
	1.1.4.1.1. O Sistema monitorador dos equipamentos gera uma notificação informando o parâmetro, equipamento e valor inaceitável.

Quadro 7 – UC06 – Monitorar propriedades

No Quadro 8 é apresentado o *script* dos cenários do caso de uso notificar responsável. Nesse caso de uso o Sistema Notificador envia e-mail para os responsáveis pelo monitoramento, informando-os sobre as notificações geradas. Este caso de uso está associado

ao RF06.

Ator	Sistema Notificador
Cenário Principal	1. O Sistema Notificador solicita ao sistema as notificações pendentes de serem enviadas.
	2. O sistema retorna para o Sistema Notificador uma lista de notificações.
	3. O Sistema Notificador verifica se a lista possui notificações.
	4. O Sistema Notificador aguarda um tempo.
Enviar Notificações	No passo 3 do Cenário Principal, caso o Sistema Notificador encontre notificações na lista:
	1.1. O Sistema Notificador envia <i>e-mail</i> para os responsáveis pelo monitoramento que estão associados com os equipamentos de cada notificação.

Quadro 8 - UC07 - Notificar responsável

3.2.2 Diagrama de classes

A apresentação da estrutura de classes do sistema foi dividida em pequenas seções. As classes foram divididas em dois grandes grupos, que são as classes do lado servidor (Java) e as classes do lado cliente (Flex). As classes do servidor e do cliente também possuem subseções, separadas por pacote.

3.2.2.1 Diagramas de classes do lado servidor

Nesta seção são apresentadas as principais classes do sistema que ficam do lado servidor. O diagrama de classes foi separado em partes que são apresentadas a seguir.

3.2.2.1.1 Diagrama de classes do pacote `modelo`

A Figura 15 apresenta o diagrama de classes do pacote `modelo` para o lado servidor. Para simplificar o diagrama não são apresentados os métodos, são apresentados apenas os atributos de cada classe.

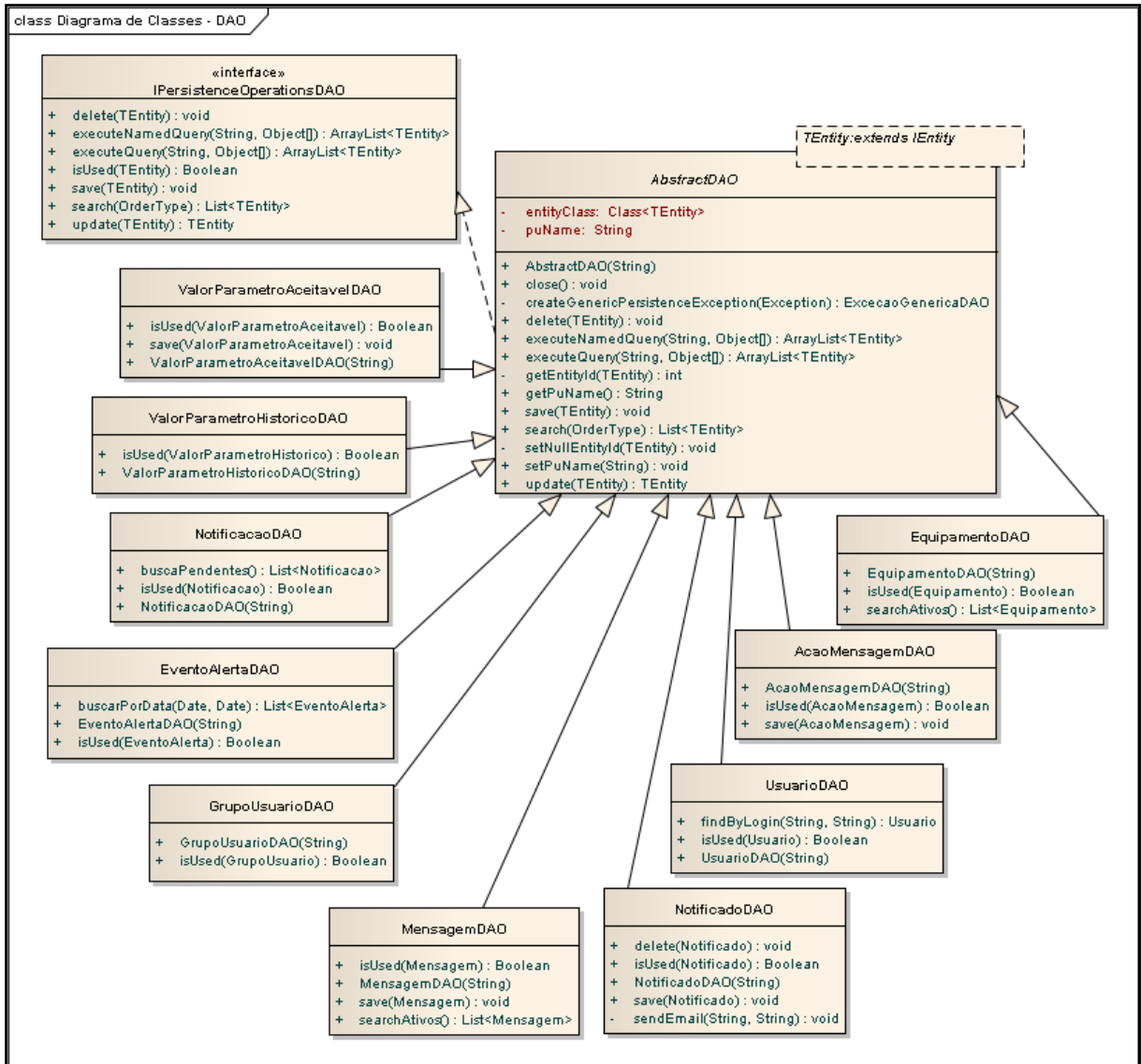


Figura 16 – Diagrama de classes do pacote dao

3.2.2.1.3 Diagrama de classes do pacote service

A Figura 17 apresenta o diagrama de classes do lado servidor para as classes de serviço do Flex, do Sistema Notificador e do Sistema Monitorador dos Equipamentos.

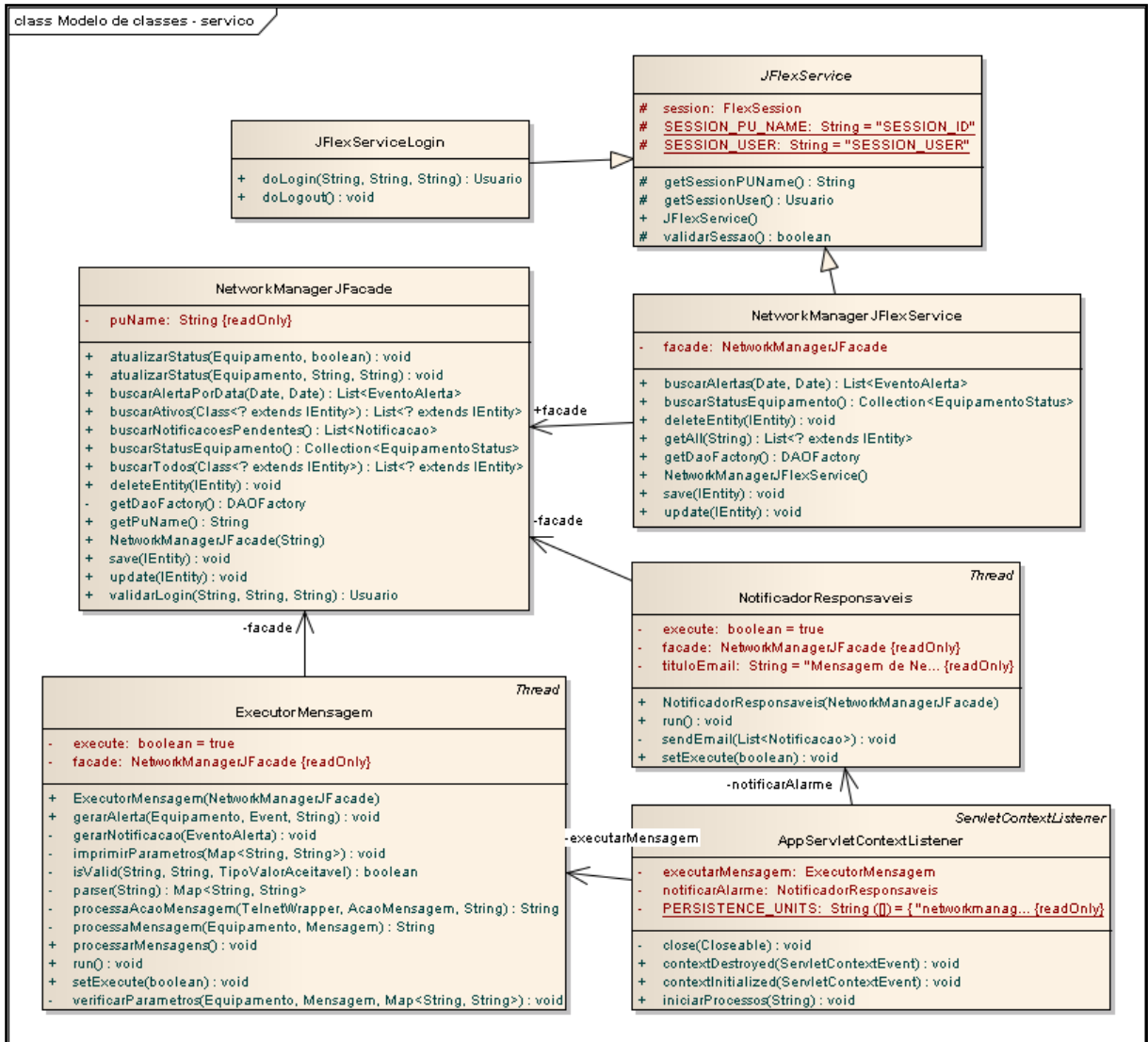


Figura 17 – Diagrama de classes do pacote service

3.2.2.2 Diagrama de classes do lado cliente

Nesta seção são apresentadas as principais classes do sistema que ficam do lado cliente.

A Figura 18 apresenta o diagrama de classes do lado cliente.

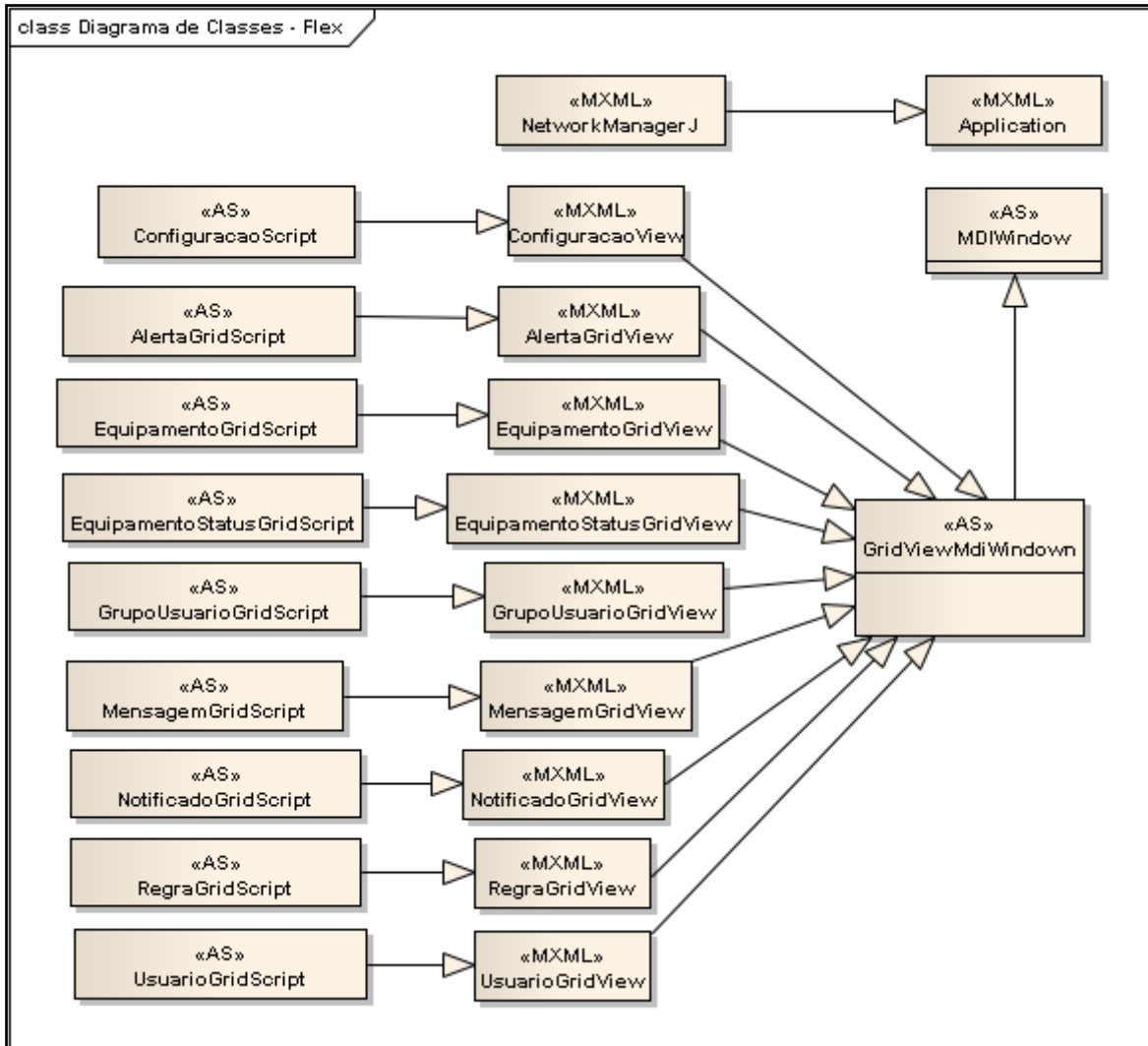


Figura 18 – Diagrama de classes para o lado cliente

3.2.3 Diagramas de sequência

Segundo Pfleeger (2004, p. 221), diagramas de sequência mostram como as mensagens fluem de um objeto para outro, formalizando as descrições informais dos eventos nos requisitos.

Nesta seção são apresentados os diagramas de sequência para as principais funcionalidades do sistema.

O diagrama de sequência na Figura 19 apresenta a sequência na troca de mensagens entre os objetos quando o Usuário abre a tela de manutenção de equipamentos.

A sequência inicia quando o Usuário clica no botão Equipamentos, localizado no menu principal do sistema, e a classe NetworkManajerJ cria a classe EquipamentoGriScript, que é a tela de manutenção de equipamentos. Após a classe

`EquipamentoGridScript` ser criada ela faz uma chamada assíncrona para o servidor, através de *Action Message Format (AMF)*¹, chamando o método `getAll` do objeto `RemoteObject`, passando como parâmetro o caminho da classe `Equipamento` no Java. O servidor recebe a mensagem, processa e retorna de forma assíncrona uma lista com os equipamentos cadastrados. O objeto `RemoteObject` dispara um evento quando a resposta chega no cliente, esse evento é processado pela classe `EquipamentoGridScript` no método `getAllSucess`, que recebe o evento como parâmetro. Por fim é feita a atualização dos dados da tela com a informação de lista dos equipamentos que está no evento e mostrado para o Usuário.

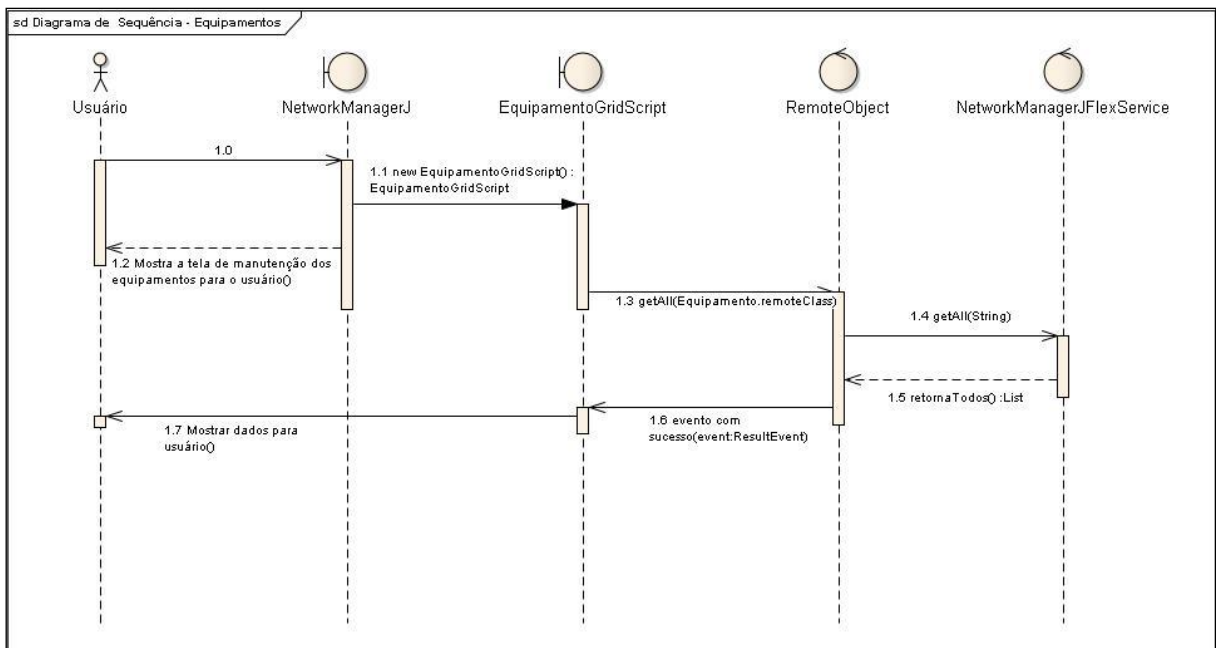


Figura 19 – Diagrama de sequência para abertura da tela manutenção de equipamentos

O diagrama de sequência na Figura 20 apresenta a sequência na troca de mensagens entre os objetos para quando o Sistema Monitorador dos Equipamentos monitora os equipamentos de acordo com as mensagens e regras cadastradas no sistema.

¹ Segundo Santos (2012, p. 1), AMF possibilita a troca de mensagens binárias entre cliente e servidor utilizando o protocolo *HyperText Transfer Protocol (HTTP)*.

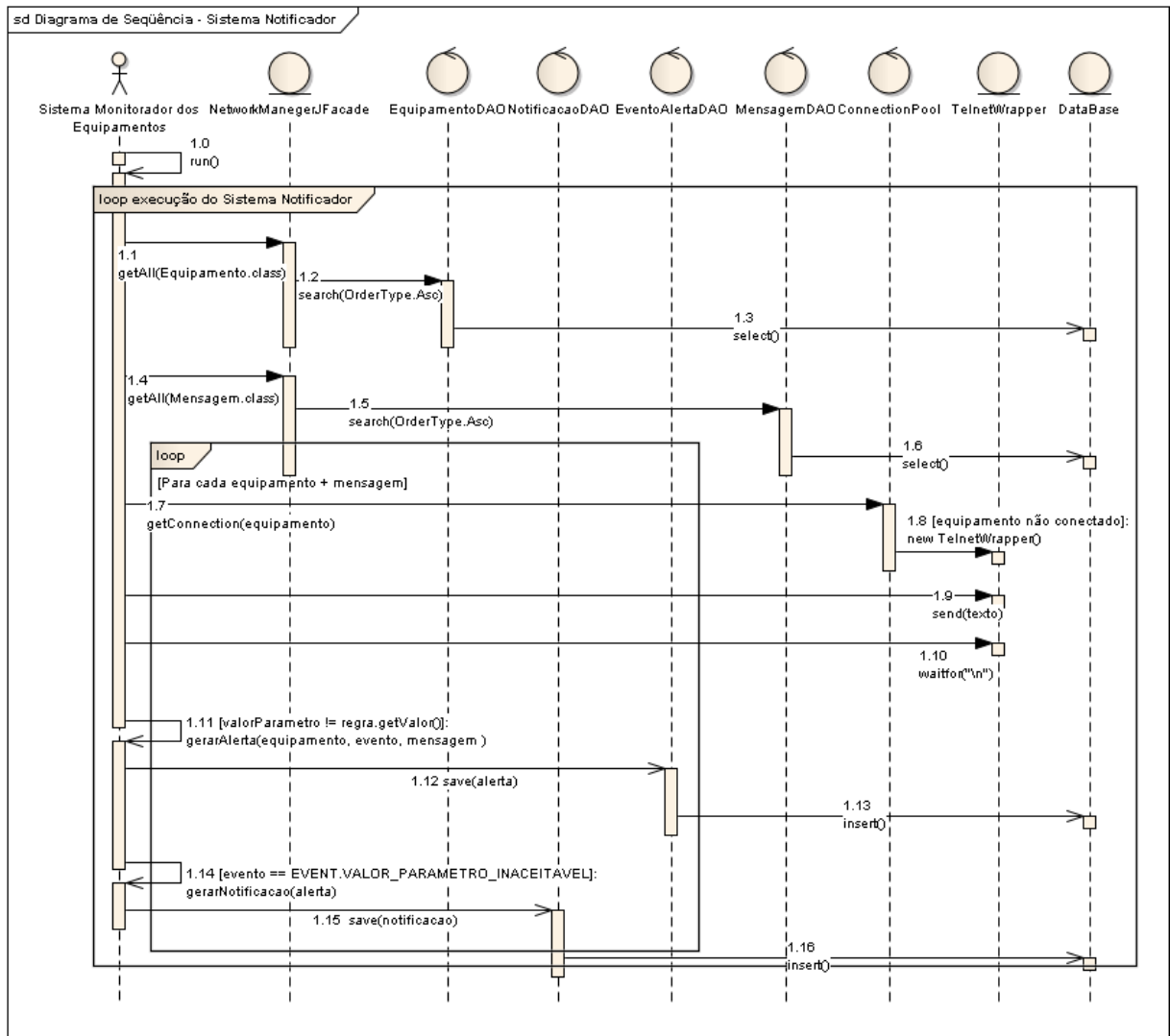


Figura 20 - Diagrama de seqüência para o Sistema Monitorador dos Equipamentos

3.2.4 Modelagem de dados

A Figura 21 apresenta a modelagem de dados utilizando CDM.

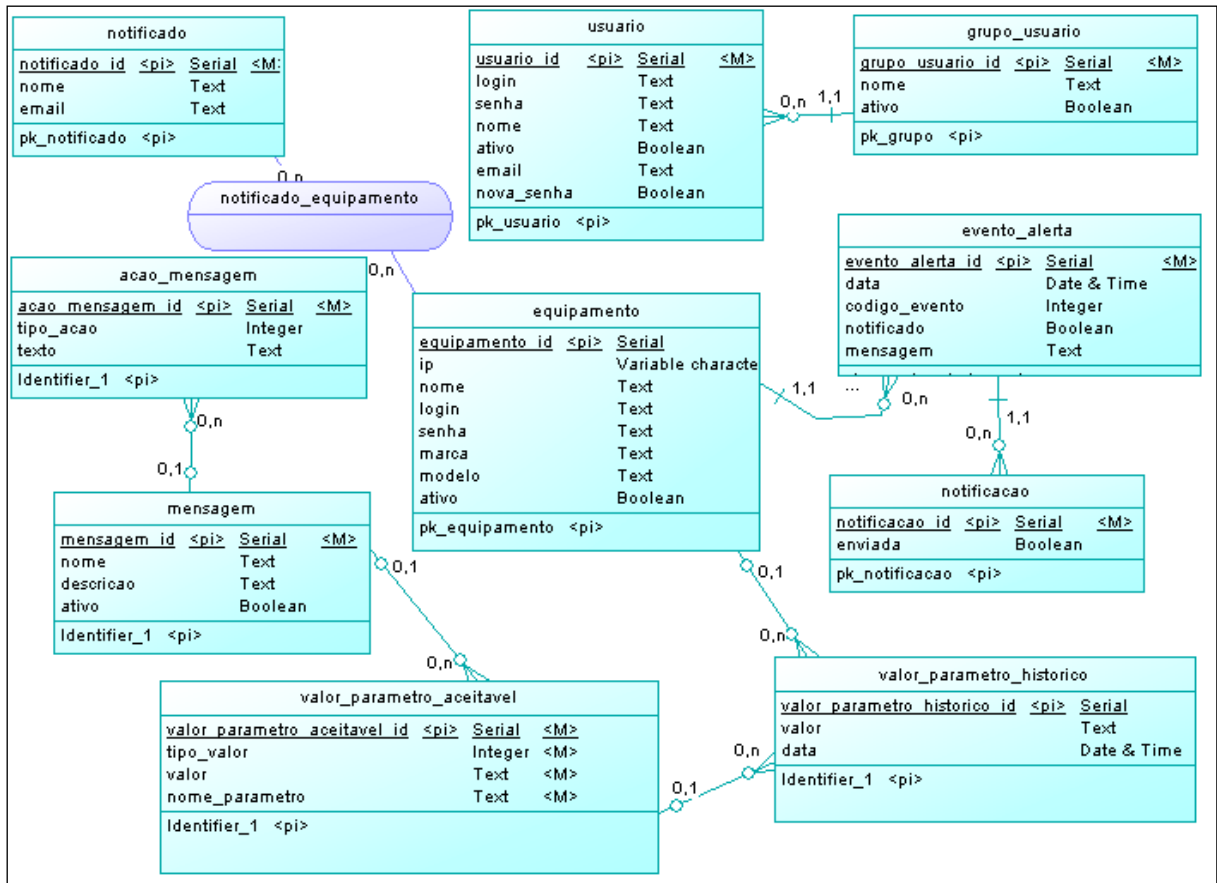


Figura 21 – Modelagem de dados conceitual do sistema

A partir do modelo conceitual foi gerado o PDM, utilizando a ferramenta PowerDesigner 12. A Figura 22 apresenta o PDM.

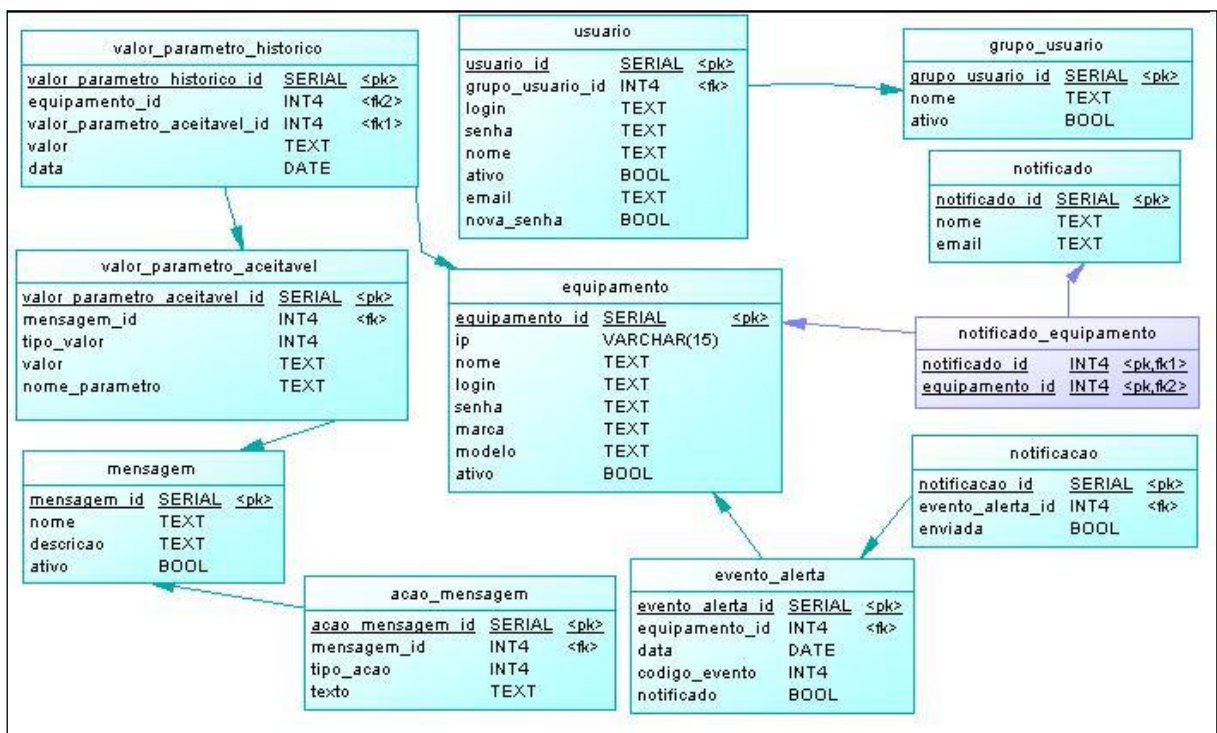


Figura 22 – Modelo de dados físico do sistema

A Tabela 2 apresenta a descrição de cada tabela.

Tabela	Descrição
usuario	Tabela para registrar os usuários cadastrados no sistema.
grupo_usuario	Tabela para registrar os grupos de usuários.
notificado	Tabela para registrar os responsáveis por equipamentos.
equipamento	Tabela para registrar os equipamentos.
notificado_equipamento	Tabela para realizar o mapeamento de muitos para muitos entre os registros de equipamentos e responsáveis por equipamentos.
evento_alerta	Tabela para registrar os alertas gerados pelo sistema.
notificacao	Tabela para registrar as notificações geradas pelo sistema, sobre um alerta de um parâmetro inválido.
mensagem	Tabela para armazenar as mensagens que devem ser enviadas para os equipamentos ativos, para monitorá-los.
acao_mensagem	Tabela para armazenar as ações de cada mensagem, informando ao sistema como ele deve processar cada mensagem.
valor_parametro_aceitavel	Tabela para armazenar os valores aceitáveis de cada parâmetro.
valor_parametro_historico	Tabela para armazenar o histórico do monitoramento de cada parâmetro.

Tabela 2 – Descrição das tabelas

3.3 IMPLEMENTAÇÃO

A implementação do sistema foi realizada seguindo a estrutura cliente/servidor. A Figura 23 apresenta a arquitetura do sistema.

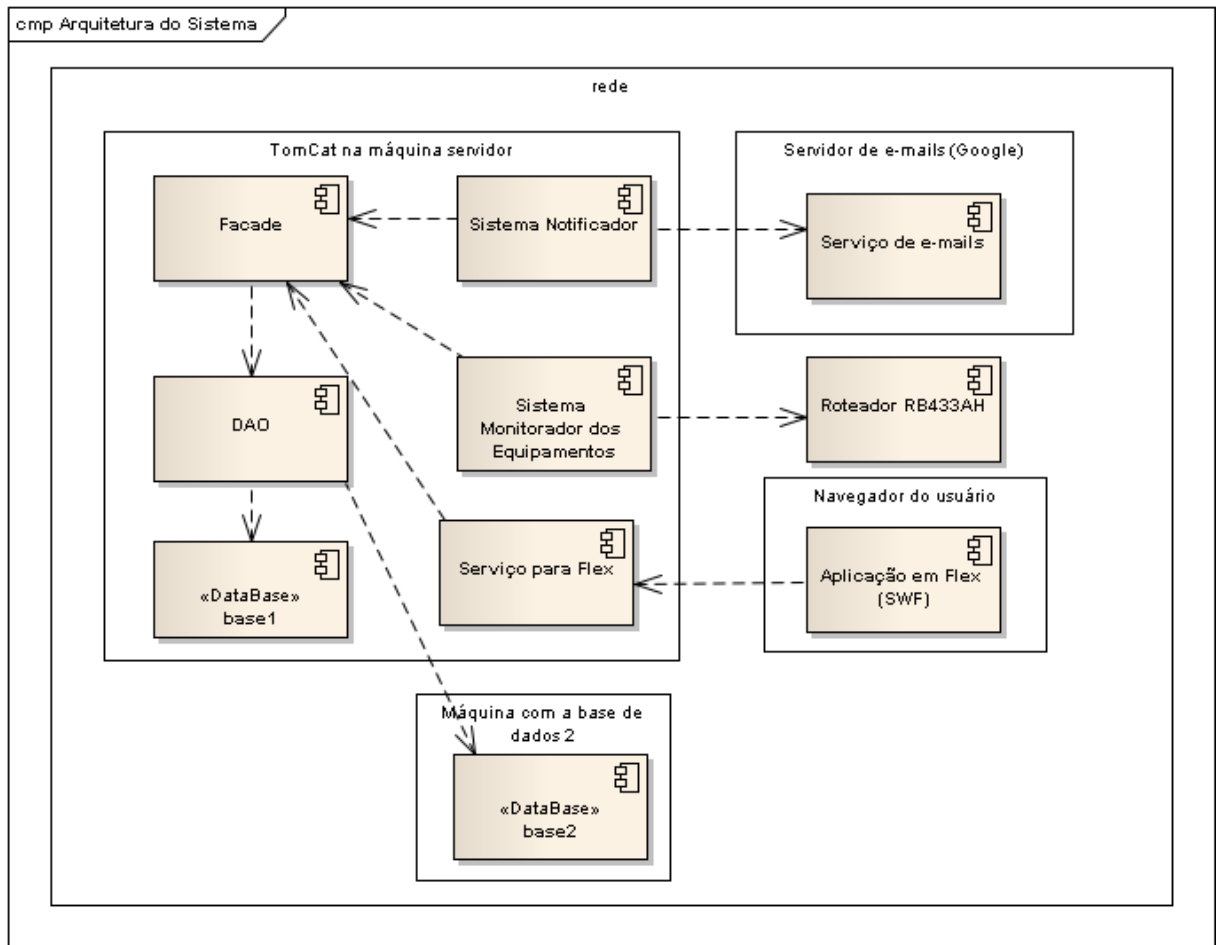


Figura 23 – Arquitetura do sistema

A interface com usuário é uma aplicação em Flex que faz requisições ao servidor através das classes de serviço para o Flex.

O Sistema Monitorador dos Equipamentos se comunica com a RouterBoard.

O Sistema Notificador se conecta ao servidor de *e-mails* para enviar *e-mail* para os responsáveis pelos equipamentos.

A camada de serviço para o Flex, o Sistema Monitorador dos Equipamentos e o Sistema Notificador dependem do *Facade*. A camada *Facade* acessa a camada DAO do sistema. A camada DAO do sistema manipula os dados na base de dados correspondente a base de dados que o usuário está logado ou a base de dados utilizada pelo subsistema. Na próxima seção será apresentado mais detalhes sobre o *Facade* e a camada DAO.

A seguir são apresentadas as técnicas, ferramentas utilizadas e a operacionalidade da implementação do sistema.

3.3.1 Técnicas e ferramentas utilizadas

Foi utilizado o paradigma da OO em toda a implementação do sistema em conjunto com os padrões de projeto *Data Access Object* (DAO), *Facade*, *Factory*, *Object Pool* e *Model View Controller* (MVC).

Segundo Tacla (2010, p. 2), o objetivo principal do padrão MVC é organizar uma aplicação em três camadas para aumentar a flexibilidade e o reuso. Para Tacla (2010, p. 3) o padrão MVC possui três tipos de objetos, que são:

- a) *model* – são objetos do domínio da aplicação;
- b) *view* – são objetos da apresentação;
- c) *controller* – são objetos que definem como a interface do usuário reage a uma entrada do usuário.

O padrão MVC foi aplicado neste trabalho separando os fontes do sistema em pacotes para dividir os objetos do domínio do sistema, das telas e das classes de controle.

A implementação foi realizada pensando em obter um baixo acoplamento e alta coesão. Segundo Kung, Lopes, Moreira, Silveira, Silveira e Steppat (2012, p. 80) acoplamento é o quanto dois elementos estão amarrados entre si e quanto as alterações no comportamento de um afetam o de outro e alta coesão é uma importante e boa prática de programação que garante que a responsabilidade dos componentes sejam relacionadas e façam sentido juntas.

Para a implementação do sistema, utilizou-se no lado servidor a linguagem de programação Java com a *Java Development Kit* (JDK) versão 1.7 (ORACLE, 2012), através da IDE Eclipse e o framework Adobe Flex (ADOBE, 2012e) com o *Flex Software Development Kit* (SDK) versão 4.6.0 (ADOBE, 2012c), através da IDE Adobe Flash Builder 4.6 (ADOBE, 2012a) no lado cliente. Para o servidor web foi utilizado o Tomcat versão 7 (APACHE, 2012) e o banco de dados PostgreSQL (POSTGRESQL, 2012) versão 8.4.

A seguir é apresentado os detalhes da implementação do lado servidor, do lado cliente do sistema e a configuração do BlazeDS para a comunicação cliente/servidor.

3.3.1.1 Implementação do lado servidor

Nesta seção é apresentado a implementação das principais classes no lado servidor. A seguir, nas subseções, são apresentados os detalhes para a implementação da camada DAO, do

Facade, dos subsistemas Sistema Monitorador dos Equipamentos e Sistema Notificador e da camada.

3.3.1.1.1 Implementação da camada DAO

Para a implementação da camada DAO utilizou-se o framework Hibernate.

O Quadro 9 apresenta a configuração do arquivo `persistence.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
  <persistence-unit name="networkmanagerj"
    transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <properties>
      <property name="hibernate.archive.autodetection"
value="class"/>
      <property name="hibernate.connection.driver_class"
value="org.postgresql.Driver" />
      <property name="hibernate.connection.url"
value="jdbc:postgresql://localhost:5432/networkmanagerj" />
      <property name="hibernate.connection.username"
value="postgres" />
      <property name="hibernate.connection.password"
value="netdb" />
      <property name="hibernate.dialect"
value="org.hibernate.dialect.PostgreSQLDialect" />
      <property name="hibernate.cache.provider_class"
value="org.hibernate.cache.NoCacheProvider" />
      <property name="connection.pool_size" value="2" />
      <property name="hibernate.show_sql" value="true" />
    </properties>
  </persistence-unit>
</persistence>
```

Quadro 9 – Configuração do arquivo `persistence.xml`

No arquivo `persistence.xml` é utilizado *EXtensible Markup Language* (XML) para configurar as conexões com o banco de dados. Foi mapeada uma bases de dados, onde a unidade de persistência com nome `networkmanagerj` mapeia a base de dados `networkmanagerj`.

Na camada DAO foi utilizado os padrões de projeto DAO, Factory e Object Pool.

O padrão DAO foi aplicado neste trabalho para centralizar a camada que é responsável pela persistência dos dados, sendo a única camada do sistema que conhece como os dados são persistidos.

A Figura 24 apresenta o método `save` da classe `AbstractDAO`.

```

144 @Override
145 public void save(TEntity entity) throws ExcecaoGenericaDAO {
146     EntityManager em = null;
147     try {
148         em = EntityManagerFactoryPool.getEntityManagerFactory(getPuName())
149             .createEntityManager();
150         em.getTransaction().begin();
151         setNullEntityId(entity);
152         em.persist(entity);
153         em.getTransaction().commit();
154     } catch (Exception ex) {
155         if (em != null)
156             if (em.getTransaction().isActive())
157                 em.getTransaction().rollback();
158
159         throw (ExcecaoGenericaDAO) this.createGenericPersistenceException(
160             ex).initCause(ex);
161     } finally {
162         if (em != null)
163             em.close();
164     }
165 }

```

Figura 24 – Método save da classe AbstractDAO

O padrão Factory foi aplicado na camada DAO deste trabalho para criar o dao responsável por operações *Create Read Update Delete* (CRUD) de cada classe do modelo. Resultando assim em uma maneira abstrata de realizar as operações de CRUD.

A Figura 25 apresenta o método loadDAO da classe DAOFactory.

```

22 private IPersistenceOperationsDAO<IEntity> loadDAO(Class<? extends IEntity> entityClass)
23     throws ExcecaoGenericaDAO {
24     Class<IEntity> clazz = null;
25     try {
26         clazz = (Class<IEntity>) Class.forName("com.networkmanagerj.dao."
27             + entityClass.getSimpleName() + "DAO");
28         Constructor<?> constructor = clazz.getConstructor(String.class);
29
30         IPersistenceOperationsDAO<IEntity> dao = (IPersistenceOperationsDAO<IEntity>) constructor
31             .newInstance(this.puName);
32         if (dao == null)
33             throw new ClassNotFoundException();
34
35         return dao;
36     } catch (Exception e) {
37         throw new ExcecaoGenericaDAO(
38             "Erro ao carregar DAO.\n"
39             + e.getMessage());
40     }
41 }

```

Figura 25 – Método loadDAO da classe DAOFactory

O método loadDAO cria uma instância de uma classe dao considerando que a classe está no pacote `com.networkmanagerj.dao` e possui o nome da classe formado pelo nome da entidade e com o sufixo DAO. Por exemplo, o método loadDAO pode ser chamado com o parâmetro `Equipamento.class` e retornar uma instância da classe `EquipamentoDAO`.

Segundo Cabus e Vidal (2012), o padrão de projeto *Object Pool* tem o objetivo de reduzir o tempo e custo das instanciações, reaproveitando objetos, melhorar a performance e o

controle sobre os recursos. Para Cabus e Vidal (2012) um uso comum do padrão Object Pool são nas aplicações com bancos de dados, onde a criação das conexões com o banco é custosa, consumindo muito processamento e tempo.

O padrão *object pool* foi aplicado na camada DAO deste trabalho para melhorar a performance e o controle dos recursos do banco de dados.

A Figura 26 apresenta o método `getEntityManagerFactory` da classe `EntityManagerFactoryPool` para a criação do objeto `EntityManagerFactory` criando apenas uma instância para cada nome da unidade de persistência.

```

388 public static EntityManagerFactory getEntityManagerFactory(String puName) {
39   EntityManagerFactory emf = emfList.get(puName);
40   if (emf == null) {
41     System.out.println("Criando EntityManager para " + puName);
42     try {
43       emf = Persistence.createEntityManagerFactory(puName);
44       System.out.println("\n**** EntityManagerFactory criado, puName:" + puName);
45     } catch (Exception e) {
46       System.err.println("\n*****Erro ao criar o EntityManagerFactory." + e.getCause().getMessage());
47     }
48     emfList.put(puName, emf);
49   }
50   return emf;
51 }
52 }
53 }

```

Figura 26 - Método `getEntityManagerFactory` da classe `EntityManagerFactoryPool`

3.3.1.1.2 Implementação da classe `NetworkmanagerJFacade`

A classe `NetworkmanagerJFacade` faz parte da aplicação do padrão de projeto Facade neste trabalho.

Segundo Godói (2009), o padrão de projeto *facade* reduz a complexidade de uma *Application Programming Interface* (API), liberando acesso a métodos de alto nível encapsulando os demais, produz uma interface comum e simplificada, pode encapsular uma ou mais interfaces mal projetadas em uma mais concisa, reduz drasticamente o acoplamento entre as camadas do projeto e torna o código mais manutenível na medida em que as classes de visualização e negócio forem aumentando em quantidade.

No caso deste trabalho o padrão Facade foi utilizado para reduzir o acoplamento entre a camada DAO, os subsistemas e as classes de serviços do Flex, centralizar o acesso à camada DAO do sistema e facilitar a manutenção do código na medida em que o número de classes aumenta.

Conforme foi definido na arquitetura do sistema, apenas a classe `NetworkmanagerJFacade` conhece a camada DAO do sistema. Os subsistemas e a camada de

serviço do Flex dependem da classe `NetworkManagerJFacade` para acessar camada DAO do sistema.

A Figura 27 apresenta o método `buscarTodos` da classe `NetworkManagerJFacade`.

```

65 public List<? extends IEntity> buscarTodos(Class<? extends IEntity> clazze)
66     throws ExcecaoGenericaDAO {
67     IPersistenceOperationsDAO<? extends IEntity> dao = getDaoFactory().createDAO(clazze);
68     List<? extends IEntity> search = dao.search(OrderType.Asc);
69     return search;
70 }

```

Figura 27 – Método `buscarTodos` da classe `NetworkManagerJFacade`

A Figura 28 apresenta a função `save` da classe `NetworkManagerJFacade`.

```

90 public void save(IEntity object) throws Exception {
91     IPersistenceOperationsDAO dao = getDaoFactory().createDAO(object.getClass());
92     dao.save(object);
93 }

```

Figura 28 – Método `save` da classe `NetworkManagerJFacade`

Pode-se observar na Figura 27 e na Figura 28 que os métodos `buscarTodos` e `save` chamam o método `createDAO` da fábrica de DAO, passando a classe do objeto que se deseja salvar ou buscar todos os registros para obter o DAO responsável pelo objeto em questão. Após obter o DAO, é feito a chamada do método desejado para o DAO.

O método `getDaoFactory` é um método privado, que chama o método estático `getInstance` da classe `DAOFactory` passando o nome da unidade de persistência (atributo `puName`) como parâmetro.

3.3.1.1.3 Implementação do Sistema Monitorador dos Equipamentos

Para a implementação do Sistema Monitorador dos Equipamentos foi criado a classe `ExecutorMensagem`. A classe `ExecutorMensagem` é uma *thread*, iniciada junto com o servidor de aplicação TomCat na classe `AppServletContextListener`.

A Figura 29 apresenta o método `run` da classe `ExecutorMensagem`.


```

60 @Override
61 public void run() {
62     while (execute) {
63         try {
64             processarMensagens();
65             final String puName = facade.getPuName();
66             VariaveisConfiguracao conf = VariaveisConfiguracao.getConfig(puName);
67             Thread.sleep(conf.getIntervaloProcessarMensagens());
68         } catch (Exception e) {
69             execute = false;
70         }
71     }
72     System.out.println("Finalizando Executor mensagem");
73 }

```

Figura 29 – Método run da *thread* ExecutorMensagem

A Thread ExecutorMensagem fica executando até que o TomCat seja finalizado. Como é ilustrado na Figura 29, a linha 64 invoca o método processarMensagens e a linha 67 aguarda um tempo para processar as mensagem novamente. O tempo que deve aguardar para processar novamente as mensagens é uma variável da classe VariaveisConfiguracao. O servidor possui um cache de instâncias da classe VariaveisConfiguracao, uma instância para cada unidade de persistência. Pode-se observar na linha 66 que o cache fica na própria classe VariaveisConfiguracao e é obtido com o método estático getConfig informando o nome da unidade de persistência no parâmetro.

A Figura 30 apresenta a implementação do método processarMensagens da classe ExecutorMensagem.

```

84 public void processarMensagens() throws ExcecaoGenericaDAO {
85     List<Equipamento> listaEquipamentosTemp = (List<Equipamento>) facade
86         .buscarAtivos(Equipamento.class);
87     List<Mensagem> listaMensagensTemp = (List<Mensagem>) facade
88         .buscarAtivos(Mensagem.class);
89     for (Equipamento equipamento : listaEquipamentosTemp) {
90         for (Mensagem mensagem : listaMensagensTemp) {
91             try {
92                 String returnMessage = processaMensagem(equipamento, mensagem);
93                 facade.atualizarStatus(equipamento, true);
94                 Map<String, String> params = parser(returnMessage);
95                 verificarParametros(equipamento, mensagem, params);
96             } catch (IOException e) {
97                 facade.atualizarStatus(equipamento, false);
98                 gerarAlerta(equipamento, Event.ERRO_PROCESSANDO_MENSAGEM,
99                     "Erro ao processar mensagem " + e.getMessage());
100             } catch (Exception e) {
101                 gerarAlerta(equipamento, Event.ERRO_PROCESSANDO_MENSAGEM,
102                     "Erro ao executar mensagem " + e.getMessage());
103             }
104         }
105     }
106 }

```

Figura 30 – método processarMensagens da classe ExecutorMensagem

Na linha 85 da classe ExecutorMensagem busca-se todos os equipamentos ativos e na

linha 87 todas as mensagens ativas. Em seguida, itera-se pelos equipamentos e para cada equipamento itera-se pelas mensagens. Na linha 92 é chamado o método `processaMensagem`, o retorno desse método é a resposta da RouterBoard para a mensagem que foi processada. Na linha 93 é atualizado o status do equipamento para on-line, chamando o método `atualizaStatus` da classe `NetworkManagerJFacade`. Na linha 94 é chamado o método `parser`, que retorna um mapa com os nome de parâmetros e os valores de cada parâmetro. Na linha 95 é chamado o método `verificarParametros`, passando como parâmetros o equipamento, a mensagem e o mapa de parâmetros.

Na Figura 31 é apresentado o método `processaMensagem` da classe `ExecutorMensagem`.

```

239 private String processaMensagem(Equipamento equipamento, Mensagem mensagem)
240     throws IOException {
241     List<AcaoMensagem> acoes = mensagem.getAcoes();
242     if (acoes.isEmpty()) {
243         throw new IllegalStateException("Mensagem não possui ação");
244     }
245     TelnetWrapper telnetWrapper = ConnectionPool.getConnection(equipamento);
246     String messageText = null;
247     for (AcaoMensagem acaoMensagem : acoes) {
248         messageText = processaAcaoMensagem(telnetWrapper, acaoMensagem,
249             messageText);
250     }
251     return messageText;
252 }

```

Figura 31 – método `processaMensagem` da classe `ExecutorMensagem`

No método `processaMensagem` é feito uma conferência se a mensagem possui ações para serem processadas. Na linha 245 obtém-se a conexão do equipamento representado pela classe `TelnetWrapper` e na linha 248 chama-se o método `processaAcaoMensagem` para processar cada ação da mensagem.

Para obter a conexão na linha 245 foi utilizado o padrão de projeto *object pool*. Ao invés de cada vez que for enviar uma mensagem criar a conexão, conectar e realizar o *login*, a classe `ConnectionPool` realiza esta operação apenas uma vez.

A Figura 32 apresenta o método `getConnection` da classe `ConnectionPool`.

```

3 import java.io.IOException;
4 import java.util.HashMap;
5 import java.util.Map;
6 import com.networkmanagerj.model.Equipamento;
7 import de.mud.telnet.TelnetWrapper;
8
9 public class ConnectionPool {
10
11     private static final Map<Integer, TelnetWrapper> POOL = new HashMap<Integer, TelnetWrapper>();
12
13     public static TelnetWrapper getConnection(Equipamento equipamento)
14         throws IOException {
15         TelnetWrapper telnetWrapper = POOL.get(equipamento.getId());
16         if (telnetWrapper == null) {
17             telnetWrapper = new TelnetWrapper();
18             telnetWrapper.setLocalEcho(true);
19             telnetWrapper.connect(equipamento.getIp(), 23);
20             telnetWrapper.login(equipamento.getLogin(), equipamento.getSenha());
21             telnetWrapper.waitFor("] >");
22             POOL.put(equipamento.getId(), telnetWrapper);
23         }
24         return telnetWrapper;
25     }
26 }

```

Figura 32 – Método `getConnection` da classe `ConnectionPool`

Nota-se que a implementação do método `getConnection` procura a instância de um objeto da classe `TelnetWrapper` para o equipamento desejado na variável `POOL` que é um mapa, caso não encontre cria um novo `TelnetWrapper`, conecta para o IP do equipamento e na porta padrão do telnet (23), realiza o *login* com o *login* e senha que estão no objeto equipamento, aguarda o texto `] >` ser recebido e adiciona o objeto criado na variável `POOL`.

A classe `TelnetWrapper` faz parte da biblioteca Java Telnet Api (JTA) (JUGEL e MEISSNER, 2009). A biblioteca JTA foi utilizada para realizar a comunicação com a RouterBoard através do protocolo telnet.

A Figura 33 apresenta o método `processaAcaoMensagem` da classe `ExecutorMensagem`.

```

245 private String processaAcaoMensagem(TelnetWrapper tw, AcaoMensagem acao,
246     String textActual) throws IOException {
247     if (acao.getTipoAcao() == TipoAcaoMensagem.ENVIAR.ordinal()) {
248         tw.send(acao.getTexto());
249         textActual = tw.waitFor("\n");
250     } else if (acao.getTipoAcao() == TipoAcaoMensagem.AGUARDAR.ordinal()) {
251         textActual = tw.waitFor(acao.getTexto());
252     } else if (acao.getTipoAcao() == TipoAcaoMensagem.IGNORAR_TEXTO
253         .ordinal()) {
254         textActual = textActual.replaceAll(acao.getTexto(), "");
255     }
256     return textActual;
257 }

```

Figura 33 - método `processaAcaoMensagem` da classe `ExecutorMensagem`

Uma ação de uma mensagem pode ser enviar um texto para o equipamento, aguardar um texto ser recebido ou ignorar um texto. Quando a ação é enviar, na linha 248 é chamado o método `send` da classe `TelnetWrapper` passando o texto da mensagem como parâmetro e na

linha 249 é chamado o método `waitfor` passando `\n` como parâmetro e o retorno é guardado na variável `textActual`, ou seja, após enviar algo espera a próxima linha e guarda o texto na variável `textActual`. Quando a ação é aguardar, na linha 251 é chamado o método `waitfor` passando o texto da ação como parâmetro e o retorno é guardado na variável `textActual`. Quando a ação é ignorar um texto, na linha 254 é chamado o método `replaceAll` da variável `textActual`, passando o texto da ação e uma `String` vazia como parâmetros, que fará toda parte do texto da variável `textActual` que seja igual ao texto da ação ser eliminado.

A Figura 34 apresenta o método `parser` da classe `ExecutorMensagem`.

```

108 private Map<String, String> parser(String message) throws LexicalError,
109     SyntaticError, SemanticError {
110     Sintatico sintatico = new Sintatico();
111     Semantico s = new Semantico();
112     Lexico lexico = new Lexico(message);
113     sintatico.parse(lexico, s);
114     Token token = null;
115     lexico.setPosition(0);
116     Map<String, String> params = new HashMap<String, String>();
117     String lastName = null;
118     String lastValue = null;
119     while ((token = lexico.nextToken()) != null) {
120         if (token.getId() > Constants.t_numero) {
121             lastName = token.getLexeme();
122             lexico.nextToken();// Ignora ":"
123             token = lexico.nextToken();
124             lastValue = token.getLexeme();
125             params.put(lastName, lastValue);
126         }
127     }
128     return params;
129 }

```

Figura 34 – Método `parser` da classe `ExecutorMensagem`

O método `parser` recebe como parâmetro uma resposta da `RouterBoard` para alguma mensagem e utiliza as classes geradas pelo GALS para fazer o *parse*.

Para fazer o *parser* foram utilizadas duas fases do processo de tradução de linguagens de programação: análise léxica e análise sintática. O analisador léxico faz a separação do texto (que representa o conteúdo recebido pelo roteador) em símbolos. Na análise sintática, o analisador sintático analisa a sequência de símbolos, transformando o conteúdo em uma estrutura de dados.

As definições dos analisadores léxico e sintático foram feitas para a mensagem *system health print* e *interface wireless monitor*. A Figura 35 apresenta a definição das expressões regulares utilizando a ferramenta GALS.

```

Definições Regulares
minúscula: [a-z]
maiuscula: [A-Z]
dígito: [0-9]
letra: {minúscula} | {maiuscula}
parte_mac: ({letra} | {dígito}) ({letra} | {dígito})

```

Figura 35 – Definições regulares utilizando a ferramenta GALS

A Figura 36 apresenta a definição dos *tokens* utilizando a ferramenta GALS.

```

Tokens
"ghz-a"
"MHZ"
"Mbps"
"dBm"
"%"
"dB"
"V"
"["
"]"
">"
":"
mac : {parte_mac} (":" {parte_mac})+
numero: "-"? {dígito}+ (\. {dígito}+)? " "?
nome: ({letra}) ({letra}|{dígito} | "-")*
email : {letra} ({minúscula} | {maiuscula}|{dígito})* "@" ({letra} | {dígito} | ".")+
fanmode = nome: "fan-mode"
usefan = nome: "use-fan"
activefan = nome: "active-fan"
voltage = nome: "voltage"
status = nome: "status"
band = nome: "band"
frequency = nome: "frequency"
wireless_protocol = nome: "wireless-protocol"
tx_rate = nome: "tx-rate"
rx_rate = nome: "rx-rate"
ssid = nome: "ssid"
bssid = nome: "bssid"
radio_name = nome: "radio-name"
signal_strength = nome: "signal-strength"
signal_strength_ch0 = nome: "signal-strength-ch0"
tx_signal_strength = nome: "tx-signal-strength"
tx_signal_strength_ch0 = nome: "tx-signal-strength-ch0"
noise_floor = nome: "noise-floor"
signal_to_noise = nome: "signal-to-noise"
tx_ccq = nome: "tx-ccq"
rx_ccq = nome: "rx-ccq"
p_throughput = nome: "p-throughput"
overall_tx_ccq = nome: "overall-tx-ccq"
authenticated_clients = nome: "authenticated-clients"
current_distance = nome: "current-distance"
wds_link = nome: "wds-link"
bridge = nome: "bridge"
: "-- [q quit|C-z pause|down]"
: [\s\t\n\r]
: "[k"
: "[]"

```

Figura 36 – Definição dos *tokens* utilizando a ferramenta GALS

A Figura 37 apresenta a definição dos não terminais utilizando a ferramenta GALS.

```
Não Terminais
<msg>
<msg_list>

<parametro>

<parametro_shp>
<shp_fanmode>
<shp_usefan>
<shp_activefan>
<shp_voltage>

<parametro_iwm>
<iwm_status>
<iwm_band>
<iwm_frequency>
<iwm_wirelessprotocol>
<iwm_txrate>
<iwm_rxrate>
<iwm_ssid>
<iwm_bssid>
<iwm_radioname>
<iwm_radioname2>
<iwm_signal_strength>
<iwm_signal_strength_ch0>
<iwm_tx_signal_strength>
<iwm_tx_signal_strength_ch0>
<iwm_noise_floor>
<iwm_signal_to_noise>
<iwm_tx_ccq>
<iwm_rx_ccq>
<iwm_p_throughput>
<iwm_overall_tx_ccq>
<iwm_authenticated_clients>
<iwm_current_distance>
<iwm_wds_link>
<iwm_bridge>

<linha_digitar_comando>
```

Figura 37 - Definição dos não terminais utilizando a ferramenta GALS

A Figura 38 apresenta a definição da gramática utilizando a ferramenta GALS.

```

Gramática
<msg> ::= <parametro> <msg_list> ;
<msg_list> ::= f |
           <parametro> <msg_list> | <linha_digitar_comando>;
<parametro> ::= <parametro_shp> | <parametro_iwm>;
<parametro_shp> ::= <shp_fanmode> | <shp_usefan> | <shp_activefan> |
                   <shp_voltage>;
<shp_fanmode> ::= fanmode ":" nome ;
<shp_usefan> ::= usefan ":" nome ;
<shp_activefan> ::= activefan ":" nome ;
<shp_voltage> ::= voltage ":" numero "V";

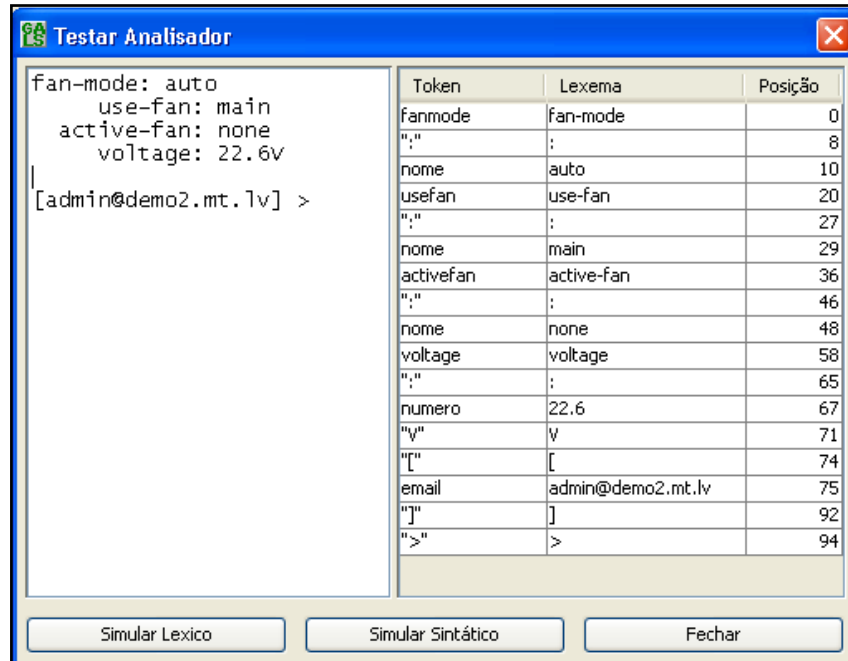
<parametro_iwm> ::= <iwm_status> | <iwm_band> | <iwm_frequency> |
                   <iwm_wirelessprotocol> | <iwm_txrate> | <iwm_rxrate>|
                   <iwm_ssid>|<iwm_bssid> |<iwm_radioname> |<iwm_signal_strength> |
                   <iwm_signal_strength_ch0> | <iwm_tx_signal_strength>|
                   <iwm_tx_signal_strength_ch0> |<iwm_noise_floor>|<iwm_signal_to_noise> |
                   <iwm_tx_ccq> |<iwm_rx_ccq> |<iwm_p_throughput>|<iwm_overall_tx_ccq>|
                   <iwm_authenticated_clients> |<iwm_current_distance>|<iwm_wds_link>|
                   <iwm_bridge>;
<iwm_status> ::= status ":" nome;
<iwm_band> ::= band ":" numero "ghz-a";
<iwm_frequency> ::= frequency ":" numero "MHz";
<iwm_wirelessprotocol> ::= wireless_protocol ":" numero;
<iwm_txrate> ::= tx_rate ":" numero "Mbps";
<iwm_rxrate> ::= rx_rate ":" numero "Mbps";
<iwm_ssid> ::= ssid ":" nome;
<iwm_bssid> ::= bssid ":" mac;
<iwm_radioname> ::= radio_name ":" <iwm_radioname2> ;
<iwm_radioname2> ::= nome | numero nome;
<iwm_signal_strength> ::= signal_strength ":" numero "dBm";
<iwm_signal_strength_ch0> ::= signal_strength_ch0 ":" numero "dBm";
<iwm_tx_signal_strength> ::= tx_signal_strength ":" numero "dBm";
<iwm_tx_signal_strength_ch0> ::= tx_signal_strength_ch0 ":" numero "dBm";
<iwm_noise_floor> ::= noise_floor ":" numero "dBm";
<iwm_signal_to_noise> ::= signal_to_noise ":" numero "dB";
<iwm_tx_ccq> ::= tx_ccq ":" numero "%";
<iwm_rx_ccq> ::= rx_ccq ":" numero "%";
<iwm_p_throughput> ::= p_throughput ":" numero;
<iwm_overall_tx_ccq> ::= overall_tx_ccq ":" numero "%";
<iwm_authenticated_clients> ::= authenticated_clients ":" numero;
<iwm_current_distance> ::= current_distance ":" numero;
<iwm_wds_link> ::= wds_link ":" nome;
<iwm_bridge> ::= bridge ":" nome;

<linha_digitar_comando> ::= "[" email "]" ">";

```

Figura 38 - Definição da gramática utilizando a ferramenta GALS

A Figura 39 apresenta os *tokens* reconhecidos pelo analisador léxico utilizando como texto de entrada a mesma resposta de mensagem da Figura 1 – a resposta para a mensagem *system health print*.



Token	Lexema	Posição
fanmode	fan-mode	0
":"	:	8
nome	auto	10
usefan	use-fan	20
":"	:	27
nome	main	29
activefan	active-fan	36
":"	:	46
nome	none	48
voltage	voltage	58
":"	:	65
numero	22.6	67
"v"	v	71
"["	[74
email	admin@demo2.mt.lv	75
"]"]	92
">"	>	94

Figura 39 – Tokens reconhecidos para resposta da mensagem *system health print*

A Figura 40 apresenta a árvore sintática gerada pelo analisador sintático utilizando como texto de entrada a mesma resposta de mensagem da Figura 1.

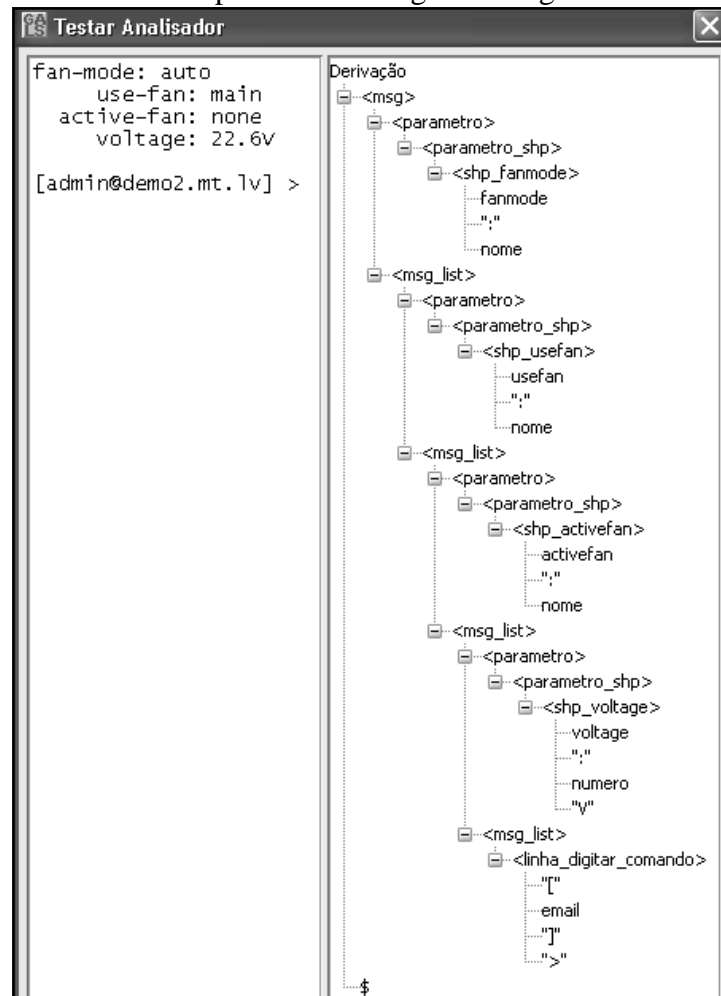


Figura 40 – Árvore sintática para resposta da mensagem *system health print*

3.3.1.1.4 Implementação do Sistema Notificador

Para a implementação do Sistema Notificador foi criada a classe `NotificadorResponsaveis`. A classe `NotificadorResponsaveis` também é uma `Thread`, iniciada junto com o servidor de aplicação TomCat.

A Figura 41 apresenta a implementação do método `run` da classe `NotificadorResponsaveis`.

```

41     while (execute) {
42         try {
43             List<Notificacao> notificacoes = facade
44                 .buscarNotificacoesPendentes();
45             if (notificacoes.size() > 0) {
46                 sendEmail(notificacoes);
47             }
48             final String puName = facade.getPuName();
49             VariaveisConfiguracao conf = VariaveisConfiguracao.getConfig(puName);
50             Thread.sleep(conf.getIntervaloSendEmail());
51         } catch (Exception e) {
52             System.out.println("Erro ao enviar notificações: "+e.getMessage());
53             e.printStackTrace();
54             execute = false;
55         }
56     }

```

Figura 41 – implementação do método `run` da classe `NotificadorResponsaveis`

Nas linhas 43 e 44 da classe `NotificadorResponsaveis` é solicitado para o `facade` as notificações pendentes para serem enviadas, na linha 45 é realizado a verificação se existem notificações para serem enviadas, se existir chama o método para enviar as notificações na linha 46 passando as notificações pendentes como parâmetro.

3.3.1.1.5 Implementação da camada de serviço para o Flex

A implementação da camada de serviço para o Flex envolve a implementação das classes `JFlexService`, `JFlexServiceLogin` e `NetworkManagerJFlexService`.

A classe `JFlexService` é uma classe abstrata que guarda a sessão do usuário e também possui um método para validar a sessão.

A Figura 42 apresenta o método `validarSessao` da classe `JFlexService`. O método retorna `true` quando a sessão é válida e `false` quando a sessão é inválida.

```

13 protected boolean validarSessao() {
14     this.session = FlexContext.getFlexSession();
15     if (!this.session.isValid()) {
16         return false;
17     }
18     if (this.session.getAttribute(SESSION_USER) == null) {
19         return false;
20     }
21     if (this.session.getAttribute(SESSION_PU_NAME) == null) {
22         return false;
23     }
24     return true;
25 }

```

Figura 42 – Método validarSessao da classe JFlexService

A classe JFlexServiceLogin é uma subclasse da classe JFlexService, responsável por receber no servidor as chamadas de *login* e *logout* do usuário no sistema.

A Figura 43 apresenta os métodos doLogin e doLogout da classe JFlexServiceLogin.

```

18 public Usuario doLogin(String puName, String login, String senha)
19     throws Exception {
20     Usuario usuario = new NetworkmanagerJFacade(puName).validarLogin(
21         puName, login, senha);
22     this.session.setAttribute(SESSION_USER, usuario);
23     this.session.setAttribute(SESSION_PU_NAME, puName);
24     return usuario;
25 }
26
27 public void doLogout()
28 {
29     if (this.validarSessao())
30     {
31         this.session.removeAttribute(SESSION_PU_NAME);
32         this.session.removeAttribute(SESSION_USER);
33         this.session.invalidate();
34     }
35 }

```

Figura 43 – Métodos doLogin e doLogout da classe JFlexServiceLogin

Pode-se observar na Figura 43 que na linha 20 é chamado o método validarLogin da classe NetworkManagerJFacade informando os parâmetros puName, login e senha. Se retornou um usuário o *login* é válido e portanto é adicionado o usuário e o nome da unidade de persistência utilizada na sessão, caso o *login* não for válido a chamada do método validarLogin lança uma exceção.

A Figura 44 apresenta alguns métodos da classe NetworkmanagerJFlexService.

```

46 public void save(IEntity object) throws Exception {
47     facade.save(object);
48 }
49
50 public void update(IEntity object) throws Exception {
51     facade.update(object);
52 }
53
54 public void deleteEntity(IEntity object) throws Exception {
55     facade.deleteEntity(object);
56 }
57
58 public Collection<EquipamentoStatus> buscarStatusEquipamento()
59     throws Exception {
60     Collection<EquipamentoStatus> lista = facade.buscarStatusEquipamento();
61     return lista;
62 }
63
64 public List<EventoAlerta> buscarAlertas(Date dataInicio, Date dataFim)
65     throws ExcecaoGenericaDAO {
66     List<EventoAlerta> lista = facade.buscarAlertaPorData(dataInicio,
67         dataFim);
68     return lista;
69 }

```

Figura 44 – Implementação de alguns métodos da classe NetworkmanagerJFlexService

A classe NetworkmanagerJFlexService é responsável por receber qualquer requisição do lado cliente, diferente de login e logout. Para qualquer requisição que depende do banco de dados é invocado um método da classe NetworkManagerJFacade.

3.3.1.2 Implementação do lado cliente

Nesta seção é apresentada a implementação das principais classes no lado cliente e a configuração para a comunicação entre o cliente e servidor. Será apresentada a implementação da classe GridViewMdiWindow, o código de algumas telas do sistema e como foi configurado o BlazeDS.

3.3.1.2.1 Implementação da classe GridViewMdiWindow

A classe GridViewMdiWindow foi implementada utilizando a linguagem ActionScript. A classe GridViewMdiWindow possui a maior parte dos controles de todas as telas, fazendo as telas ficarem com um comportamento padrão.

A implementação da classe GridViewMdiWindow facilitou a criação de cada tela de

cadastro, pois basta estender a classe e a maioria das funcionalidades da tela já estavam prontas.

Pode-se observar no diagrama de classes para o lado cliente, na Figura 18, que todas as telas do sistema são subclasses da classe `GridViewMdiWindow`.

A classe `GridViewMdiWindow` estende a classe `MDIWindow`. A classe `MDIWindow` faz parte da biblioteca FlexMDI (NEGREIROS, 2010). Para trabalhar com janelas *Multiple Document Interface* (MDI), foi adicionado a biblioteca FlexMDI no projeto.

A Figura 45 apresenta o construtor da classe `GridViewMdiWindow`, onde é criado o `RemoteObject` e adicionado as escutas de eventos.

```

46 public function GridViewMdiWindow()
47 {
48     super();
49     addEventListener(FlexEvent.CREATION_COMPLETE, onCreateComplete);
50     remoteObject = new RemoteObject("service");
51     remoteObject.showBusyCursor=true;
52     remoteObject.getOperation("save").addEventListener(ResultEvent.RESULT, operationSucess);
53     remoteObject.getOperation("update").addEventListener(ResultEvent.RESULT, operationSucess);
54     remoteObject.getOperation("deleteEntity").addEventListener(ResultEvent.RESULT, operationSucess);
55     remoteObject.getOperation("getAll").addEventListener(ResultEvent.RESULT, operationSucess);
56     remoteObject.getOperation("save").addEventListener(FaultEvent.FAULT, operationFailed);
57     remoteObject.getOperation("update").addEventListener(FaultEvent.FAULT, operationFailed);
58     remoteObject.getOperation("deleteEntity").addEventListener(FaultEvent.FAULT, operationFailed);
59     remoteObject.getOperation("getAll").addEventListener(FaultEvent.FAULT, operationFailed);
60 }

```

Figura 45 – Construtor da classe `GridViewMdiWindow`

A Figura 46 apresenta o método `onCreateComplete` que é executado após a janela ser criada (quando o evento `FlexEvent.CREATION_COMPLETE` for disparado).

```

114 protected function onCreateComplete(event:FlexEvent):void{
115     var obj:DisplayObject = getChildByNameAux(BT_ADD);
116     if(obj!=null){
117         obj.addEventListener(MouseEvent.CLICK, btAddOnClick);
118     }
119     obj = getChildByNameAux(BT_REFRESH);
120     if(obj!=null){
121         obj.addEventListener(MouseEvent.CLICK, btRefreshOnClick);
122     }
123     obj = getChildByNameAux(BT_CANCEL);
124     if(obj!=null){
125         obj.addEventListener(MouseEvent.CLICK, btCancelOnClick);
126     }
127
128     obj = getChildByNameAux(TW_CAD_NEW);
129     if(obj!=null){
130         TitleWindow(obj).addEventListener(CloseEvent.CLOSE, twOnClose);
131         obj.x = (screen.width - obj.width)/2;
132         obj.y = (screen.height - obj.height)/2;
133
134         var btSave:DisplayObject = getChildByNameAux(BT_SAVE);
135         if(btSave!=null){
136             Button(btSave).addEventListener(MouseEvent.CLICK, validate);
137         }
138         obj.visible= false;
139     }
140     this.addEventListener(NetGridEvents.ITEM_GRID_DELETE, this.gridItemDeleteEventHandler);
141     this.addEventListener(NetGridEvents.ITEM_GRID_EDIT, this.gridItemEditEventHandler);
142     loadDataProvider();
143 }

```

Figura 46 – Método `onCreateComplete` da classe `GridViewMdiWindow`

Nota-se que o método `onCreateComplete` procura alguns componentes com nomes armazenados em variáveis estáticas e adiciona nesses componentes escutas de eventos. As subclasses da classe `GridViewMdiWindow` é quem possuem os componentes procurados.

A Figura 47 apresenta o método `operationSucess` que é executado quando a aplicação Flex recebe uma resposta do servidor com sucesso (quando o evento `ResultEvent.RESULT` for disparado).

```

186 protected function operationSucess(event:ResultEvent):void{
187     if(RemotingMessage(event.token.message).operation=="save"){
188         Alert.show(MESSAGE_SAVE_SUCESS);
189     }else if(RemotingMessage(event.token.message).operation=="update"){
190         Alert.show(MESSAGE_UPDATE_SUCESS);
191     }else if(RemotingMessage(event.token.message).operation=="deleteEntity"){
192         Alert.show(MESSAGE_DELETE_SUCESS);
193         loadDataProvider();
194         return;
195     }else if(RemotingMessage(event.token.message).operation=="getAll"){
196         getAllSucess(event);
197         return;
198     }
199     closeTw();
200     clearForm();
201     loadDataProvider();
202 }

```

Figura 47 – Método `operationSucess` da classe `GridViewMdiWindow`

A Figura 48 apresenta o método `operationFailed` que é executado quando a aplicação Flex recebe uma resposta do servidor com falha (quando o evento `FaultEvent.FAULT` for disparado).

```

204 protected function operationFailed(event:FaultEvent):void{
205     var errorMessage:ErrorMessage = event.message as ErrorMessage;
206     if(RemotingMessage(event.token.message).operation=="deleteEntity"){
207         Alert.show(errorMessage.rootCause.message, "Erro ao excluir dados!");
208     }else if(RemotingMessage(event.token.message).operation=="save"){
209         Alert.show(errorMessage.rootCause.message, "Erro ao salvar dados!");
210     }else if(RemotingMessage(event.token.message).operation=="update"){
211         Alert.show(errorMessage.rootCause.message, "Erro ao alterar dados!");
212     }else if(RemotingMessage(event.token.message).operation=="getAll"){
213         Alert.show(errorMessage.rootCause.message, "Erro ao buscar dados!");
214     }
215 }

```

Figura 48 – Método `operationFailed` da classe `GridViewMdiWindow`

3.3.1.2.2 Implementação da classe `NetworkManagerJ`

A classe `NetworkManagerJ` foi implementada utilizando as linguagens MXML e ActionScript. Ela estende a classe `Application`.

A classe `NetworkManagerJ` possui o *layout* da tela de login e do menu principal do sistema.

A Figura 49 apresenta o método `onCreateComplete` que é executado após a aplicação ser carregada no navegador (quando o evento `FlexEvent.CREATION_COMPLETE` for disparado).

```

64 public function onCreateComplete():void
65 {
66     currentState="login";
67     panel.x = (screen.width - panel.width)/2;
68     panel.y = (screen.height - panel.height)/2;
69     panelInfo.x = (screen.width - panelInfo.width)-20;
70     remoteO = new RemoteObject("service_login");
71     remoteO.getOperation("doLogin").addEventListener(ResultEvent.RESULT,sucessLogin);
72     remoteO.getOperation("doLogin").addEventListener(FaultEvent.FAULT,faultLogin);
73     remoteO.getOperation("doLogout").addEventListener(ResultEvent.RESULT,logout);
74     remoteO.getOperation("doLogout").addEventListener(FaultEvent.FAULT,logout);
75     btOk.addEventListener(MouseEvent.CLICK,btOkOnClick);
76     btCancelar.addEventListener(MouseEvent.CLICK,btCancelOnClick);
77     btUsuarios.addEventListener(MouseEvent.CLICK,btAddMdiOnClick);
78     btGrupoUsuarios.addEventListener(MouseEvent.CLICK,btAddMdiOnClick);
79     btEquipamentos.addEventListener(MouseEvent.CLICK,btAddMdiOnClick);
80     btNotificados.addEventListener(MouseEvent.CLICK,btAddMdiOnClick);
81     btEquipamentoStatus.addEventListener(MouseEvent.CLICK,btAddMdiOnClick);
82     btMensagens.addEventListener(MouseEvent.CLICK,btAddMdiOnClick);
83     btRegras.addEventListener(MouseEvent.CLICK,btAddMdiOnClick);
84     btConfig.addEventListener(MouseEvent.CLICK,btAddMdiOnClick);
85     btAlertas.addEventListener(MouseEvent.CLICK,btAddMdiOnClick);
86     btSair.addEventListener(MouseEvent.CLICK,btAddMdiOnClick);
87 }

```

Figura 49 – Método `onCreateComplete` da classe `NetworkManagerJ`

A Figura 50 apresenta o trecho de código da classe `NetworkManagerJ` para o layout da tela de *login* e a tela para alterar senha.

```

196 <s:Panel id="panel" visible="true" x="283" y="202" width="334"
197     height="136" title.login="Login"
198     title.novaSenha="Alterar Senha"
199     visible.menu="false"
200     x.novaSenha="203" y.novaSenha="131" height.novaSenha="201">
201 <s:Label x="10" y="15" width="86" height="20" text="Id de Acesso:"
202     textAlign="right"/>
203 <s:Label x="10" y="40" width="86" height="20" text="Login:" textAlign="right"
204     x.novaSenha="10" y.novaSenha="40"/>
205 <s:Label x="10" y="65" width="86" height="20" text="Senha:" textAlign="right"
206     x.novaSenha="10" y.novaSenha="67" text.novaSenha="Senha Atual:"/>
207 <s:TextInput id="txtPuName" x="107" y="10" height="20" text="networkmanagerj"/>
208 <s:TextInput id="txtLogin" x="107" y="35" height="20"
209     x.novaSenha="107" y.novaSenha="37"/>
210 <s:TextInput id="txtSenha" x="107" y="60" width="128" height="20"
211     displayAsPassword="true" x.login="107" y.login="62"
212     x.novaSenha="107" y.novaSenha="64"/>
213 <s:Label includeIn="novaSenha" x="11" y="94" width="86" height="20"
214     text="Nova Senha:" textAlign="right"/>
215 <s:TextInput id="txtNovaSenha" includeIn="novaSenha" x="108" y="92"
216     width="128" height="20" displayAsPassword="true"/>
217 <s:Label includeIn="novaSenha" x="11" y="122" width="86" height="33"
218     text="Confirmar Nova Senha:" textAlign="right"/>
219 <s:TextInput id="txtConfirmarSenha" includeIn="novaSenha" x="108"
220     y="127" width="128" height="20" displayAsPassword="true"/>
221 <s:Button id="btOk" x="250" y="10" label="OK"/>
222 <s:Button id="btCancelar" x="249" y="35" label="Cancelar"
223     x.novaSenha="249" y.novaSenha="38"/>
224 </s:Panel>

```

Figura 50 - Código para o *layout* das telas de *login* e alterar senha

A Figura 51 apresenta o trecho de código da classe `NetworkManagerJ` para o *layout* do menu.

```

179 <mx:HBox width="100%" height="30" borderColor="#F53F3F" horizontalAlign="center"
180     verticalAlign="middle">
181 <s:Button id="btUsuarios" label="Usuários" tooltip="Manutenção de Usuários"/>
182 <s:Button id="btGrupoUsuarios" label="Grupo de Usuários"
183     tooltip="Manutenção de Grupo de Usuários"/>
184 <s:Button id="btEquipamentos" label="Equipamentos" tooltip="Manutenção de Equipamentos"/>
185 <s:Button id="btNotificados" label="Notificados" tooltip="Manutenção de Notificados"/>
186 <s:Button id="btEquipamentoStatus" label="Status"
187     tooltip="Visualizar Status de Equipamentos"/>
188 <s:Button id="btMensagens" label="Mensagens" tooltip="Manutenção de Mensagens"/>
189 <s:Button id="btRegras" label="Regras" tooltip="Configurar Regras"/>
190 <s:Button id="btAlertas" label="Alertas" tooltip="Visualizar Alertas"/>
191 <s:Button id="btConfig" label="Configurações" tooltip="Alterar Configurações"/>
192 <s:Button id="btSair" label="Sair"/>
193 </mx:HBox>

```

Figura 51 – Código para o *layout* do menu

3.3.1.2.3 Implementação da tela de monitoramento

A classe `EquipamentoStatusGridView` é a classe que possui o *layout* da tela de monitoramento. A figura apresenta o trecho de código na classe `EquipamentoStatusGridView` para a grid da tela de monitoramento.

```

<s:Button id="btRefresh" x="6" y="5" label="Atualizar" name="{BT_REFRESH}" />
<mx:DataGrid left="5" right="5" id="grid" x="0" y="30"
  width="100%" height="100%" dataProvider="{dataProvider}">
  <mx:columns>
    <mx:DataGridColumn dataField="equipamento" labelFunction="getNomeEquipamento"/>
    <mx:DataGridColumn dataField="online" headerText="on-line"
      labelFunction="{labelFunctionOnline}" width="50"/>
  </mx:columns>
</mx:DataGrid>

```

Figura 52 – Trecho de código da classe EquipamentoStatusGridView

A classe EquipamentoStatusGridScript é a classe que possui os controles da tela de monitoramento. A Figura 53 apresenta o método `recriarGrid` da classe EquipamentoStatusGridScript, utilizado para incluir as colunas na grid da tela de monitoramento de acordo com as regras cadastradas.

```

59 protected function recriarGrid():void{
60     var colunas:Array = new Array();
61     for each(var col:DataGridColumn in colunasFixas){
62         colunas.push(col);
63     }
64     var count:int = 0;
65     for each(var regra:ValorParametroAceitavel in regras){
66         var colunaNova: DataGridColumn = new DataGridColumn();
67         colunaNova.headerText = regra.nomeParametro;
68         colunaNova.labelFunction= getTipo;
69         colunas.push(colunaNova);
70     }
71     grid.columns = colunas;
72 }

```

Figura 53 – Trecho de código da classe EquipamentoStatusGridScript para atualizar as colunas da tela de monitoramento

3.3.1.3 Configuração do BlazeDS

Para configurar o BlazeDS foi alterado o arquivo `web.xml`, criado uma pasta `flex` dentro da pasta `WEB-INF`, adicionado os arquivos do BlazeDS nessa pasta e alterado o arquivo `remoting-config.xml`.

Segundo Elrom, Schulze e Tiwari (2011, p. 280-281) no BlazeDS um servlet é o gerenciador central de todas as comunicações e invocações de serviços. No arquivo `web.xml` é realizado a configuração desse *servlet* e o arquivo de configuração que ele deve ler.

O Quadro 10 apresenta as linhas incluídas no arquivo `web.xml` para a configuração do *servlet* do BlazeDS.


```

<servlet>
  <servlet-name>MessageBrokerServlet</servlet-name>
  <display-name>MessageBrokerServlet</display-name>
  <servlet-class>flex.messaging.MessageBrokerServlet</servlet-class>
  <init-param>
    <param-name>services.configuration.file</param-name>
    <param-value>/WEB-INF/flex/services-config.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

```

Quadro 10 – Linhas incluídas no arquivo `web.xml` para a configuração do BlazeDS

Segundo Elrom, Schulze e Tiwari (2011, p. 294) o arquivo `remoting-config.xml` é um dos arquivos componentes de `services-config.xml`, que configura o *servlet message broker* e permite que os serviços de dados entreguem a funcionalidade esperada.

O Quadro 11 apresenta as linhas incluídas no arquivo `remoting-config.xml`.

```

<destination id="service_login">
  <properties>
<source>com.networkmanagerj.service.flex.JFlexServiceLogin</source>
  </properties>
</destination>
<destination id="service">
  <properties>
<source>com.networkmanagerj.service.flex.NetworkManagerJFlexService</source>
  </properties>
</destination>

```

Quadro 11 – Linhas incluídas no arquivo `remoting-config.xml`

3.3.2 Operacionalidade da implementação

Nesta seção é apresentada a operacionalidade da implementação do trabalho. A seguir são apresentadas algumas telas do sistema.

3.3.3 Tela de login

A Figura 54 apresenta a tela de *login* do sistema, apresentada ao usuário quando o usuário acessa no navegador a *Uniform Resource Locator* (URL) do sistema que é `http://localhost:8080/NetworkManagerJ/NetworkManagerJ.html`. Nela o usuário informa o *login*, senha e o identificador de acesso. O identificador de acesso corresponde ao nome da unidade de persistência configurado no arquivo `persistence.xml`. Baseado no nome da unidade de persistência o sistema saberá em qual base de dados o usuário deseja conectar-se.

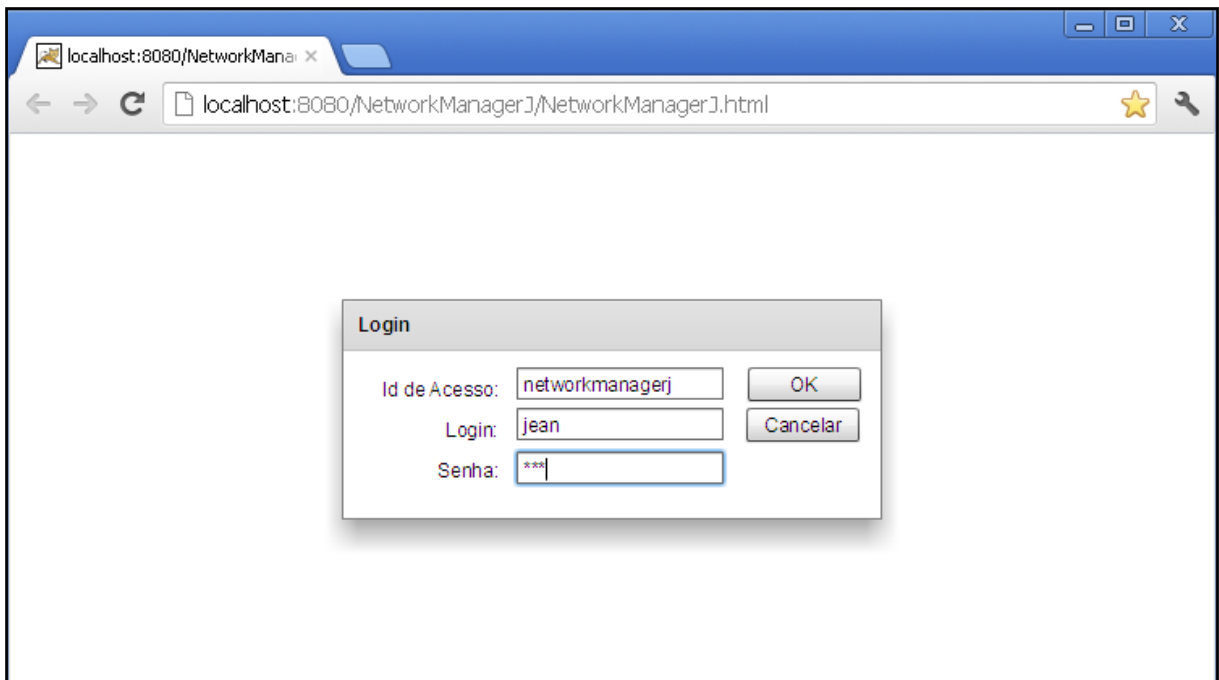


Figura 54 – Tela de login

Ao clicar no botão OK, caso o usuário ou seu grupo de usuário estiver inativo é apresentada uma mensagem informando que o usuário ou o grupo de usuário está inativo. A Figura 55 ilustra o caso.

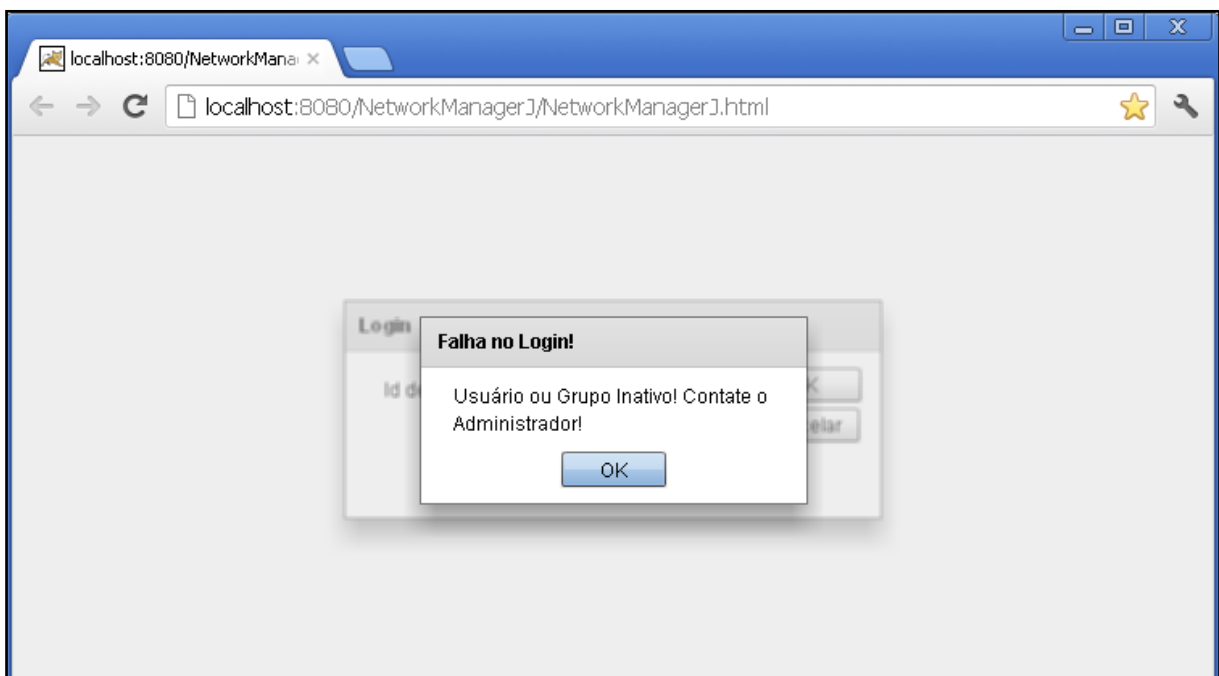


Figura 55 – Falha no login para usuário ou grupo inativo

Ao clicar no botão OK, caso o atributo novaSenha do usuário estiver com valor verdadeiro o sistema altera o layout da tela de login para o usuário realizar a troca de senha. O layout da tela para realizar a troca de senha é apresentado na Figura 56.

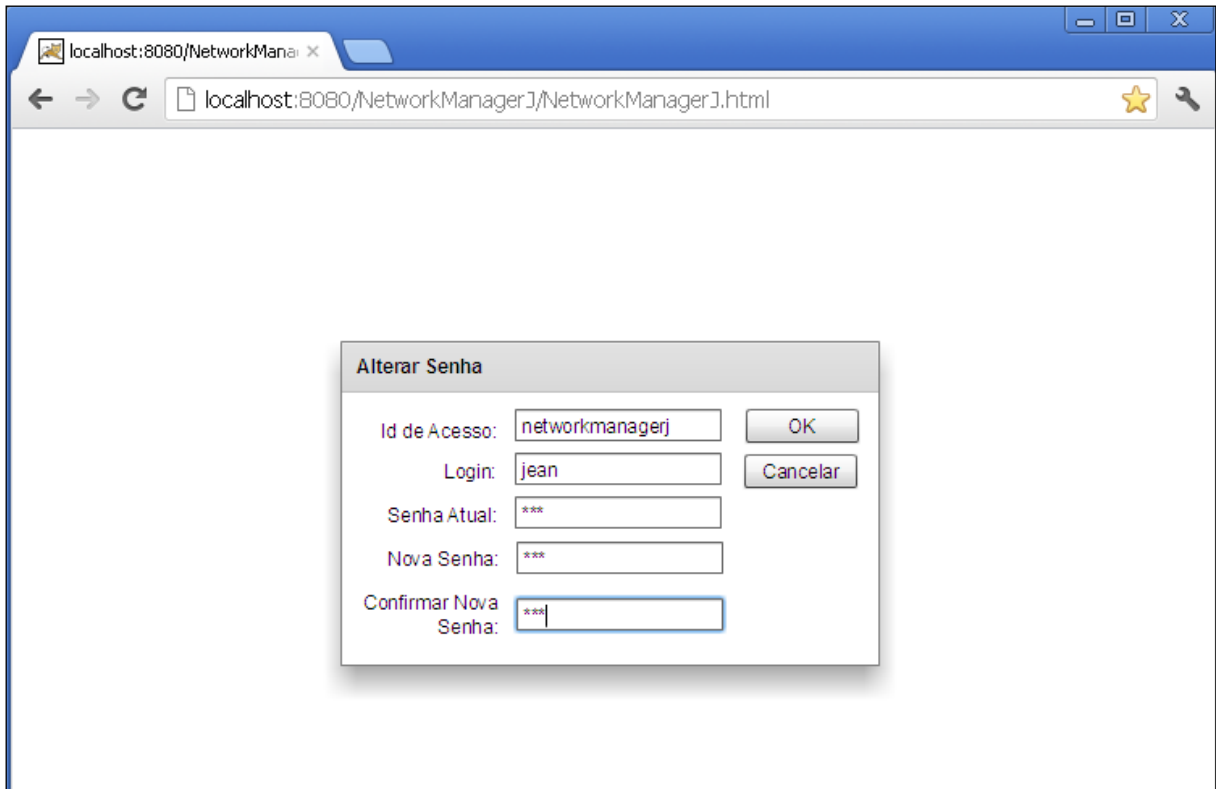


Figura 56 – Tela para alterar senha

3.3.4 Menu principal

Após realizar o *login*, a tela apresentada para o usuário é o menu principal. A Figura 57 apresenta o menu principal do sistema.

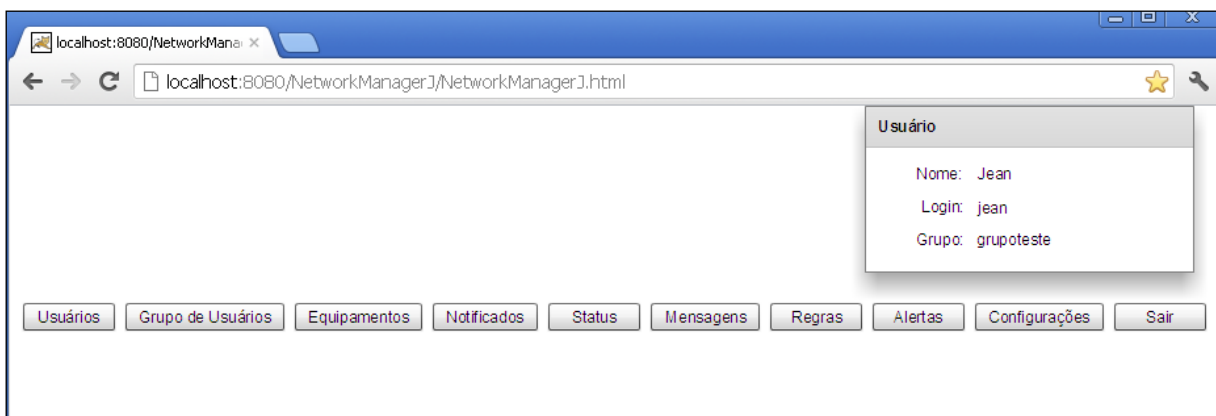


Figura 57 – Tela do menu principal do sistema

3.3.5 Manutenção de grupo de usuários

Quando o usuário clica no botão Grupo de Usuários, que está localizado no menu principal, o sistema abre a tela de manutenção de grupo de usuários, apresentando os grupos de usuários já cadastrados. A Figura 58 apresenta a tela de manutenção de grupo de usuários.

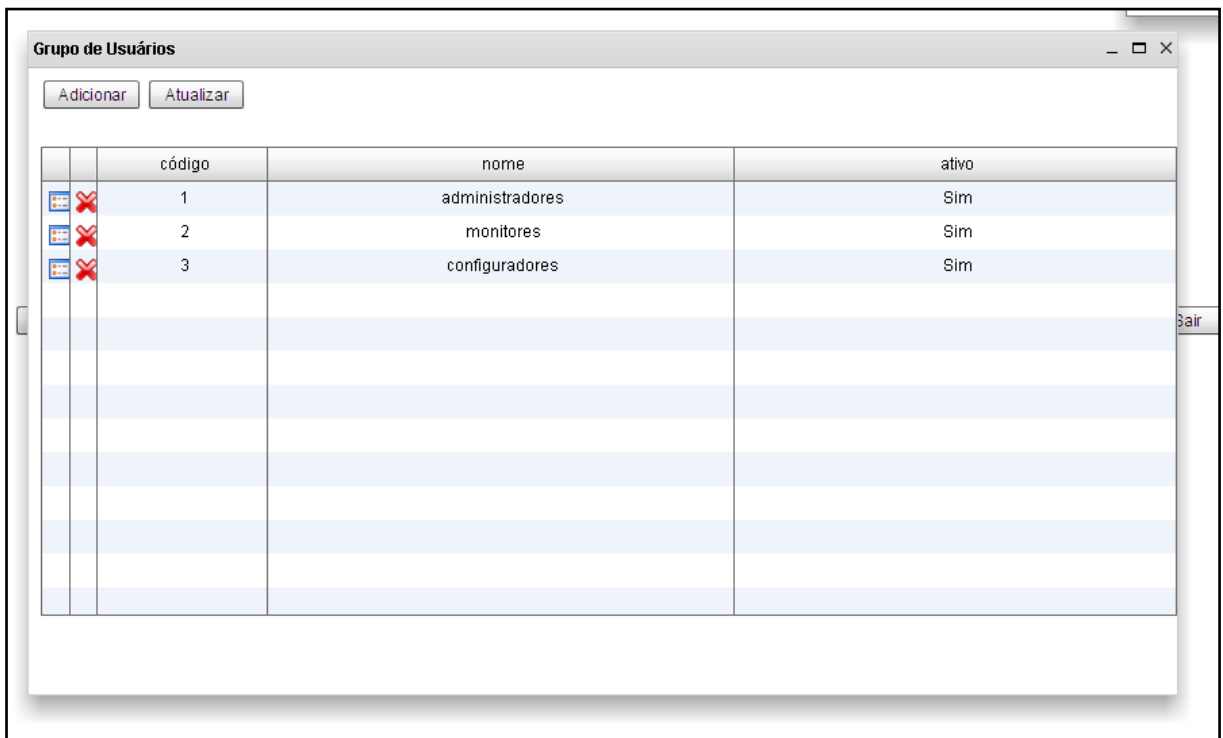




Figura 58 – Tela de manutenção de grupo de usuários

Na tela de manutenção de grupo de usuários o usuário pode observar os registros cadastrados, editar um registro clicando na imagem , excluir um registro clicando na imagem , adicionar um novo registro clicando no botão Adicionar ou Atualizar os dados clicando no botão Atualizar. Todas as telas de manutenção de cadastro seguem esse padrão.

Após abrir a tela de manutenção de grupo de usuários, quando o usuário clica no botão Adicionar o sistema abre um painel sobre a tela para cadastrar um novo grupo de usuário. A Figura 59 apresenta a tela para cadastrar um novo grupo de usuário.

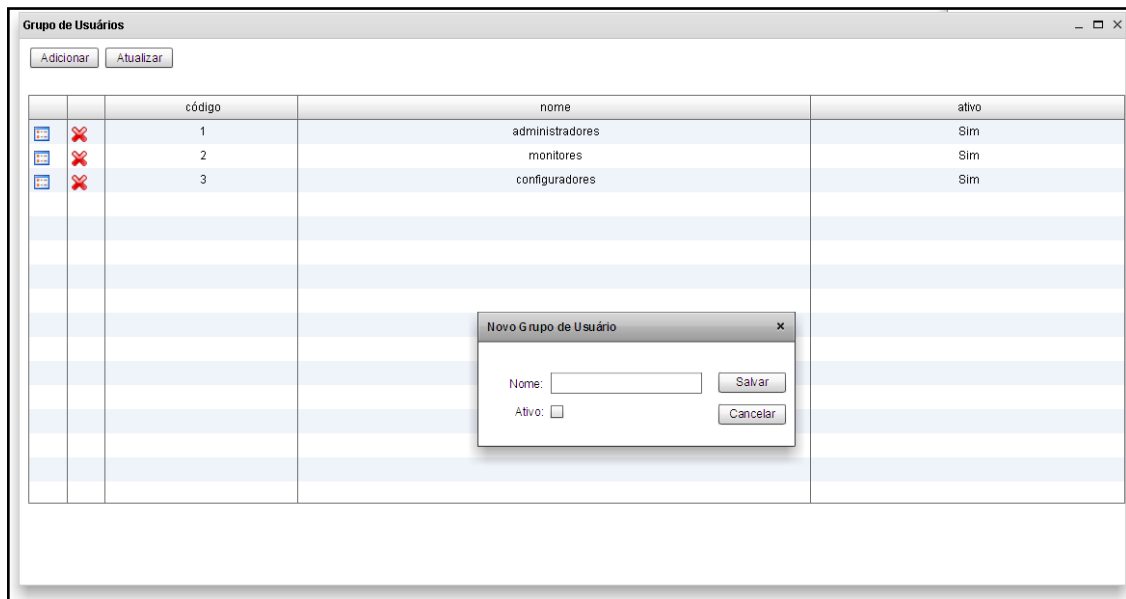


Figura 59 – Painel para cadastrar novo grupo de usuário

3.3.6 Manutenção de usuários

Quando o usuário clica no botão **Usuários**, que está localizado no menu principal, o sistema abre a tela de manutenção de usuários, apresentando os usuários já cadastrados. A Figura 60 apresenta a tela de manutenção de usuários.

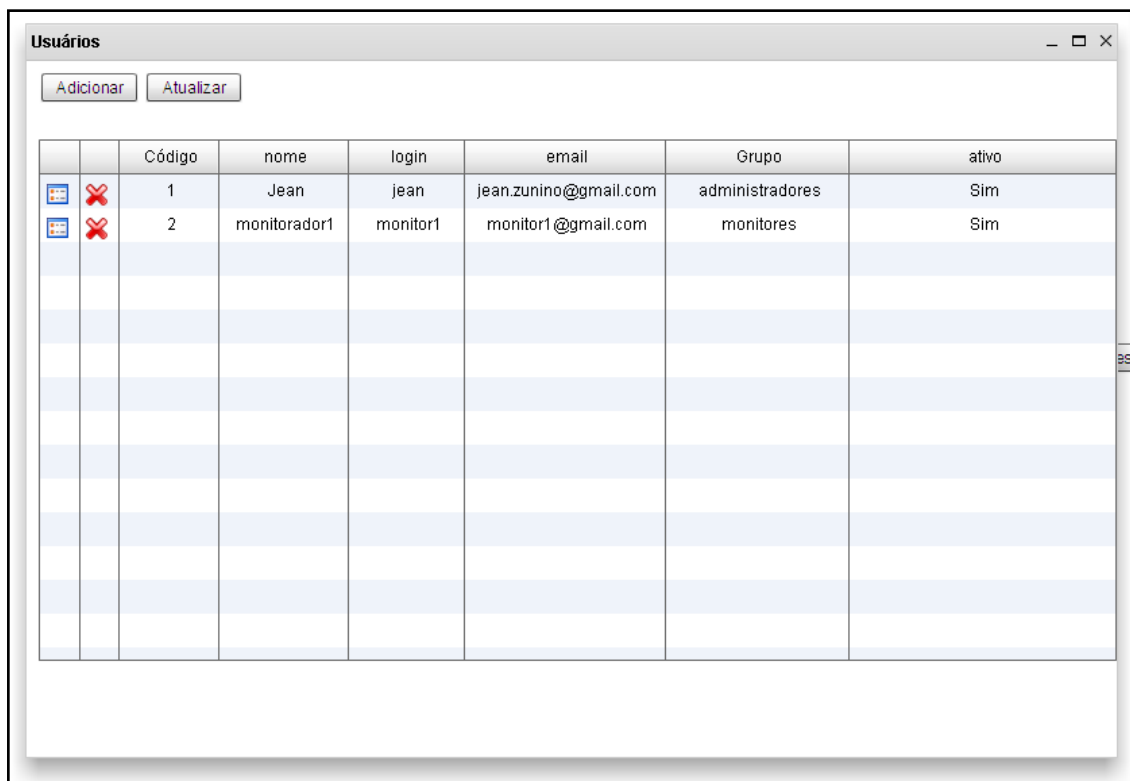


Figura 60 – Tela de manutenção de usuários

Após abrir a tela de manutenção de usuários, quando o usuário clica no botão Adicionar o sistema abre um painel sobre a tela para cadastrar um novo usuário. A Figura 61 apresenta a tela para cadastrar um novo usuário.

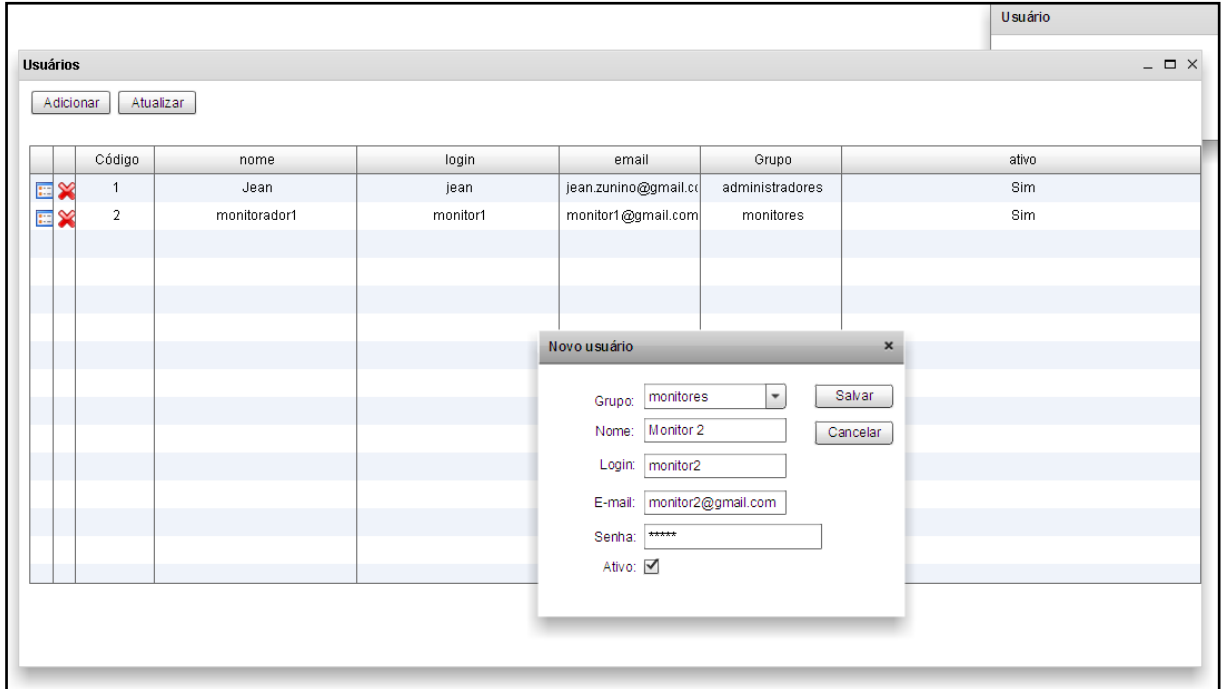


Figura 61 – Painel para cadastrar novo usuário

3.3.7 Manutenção de equipamentos

Quando o usuário clica no botão Equipamentos, que está localizado no menu principal, o sistema abre a tela de manutenção de equipamentos. A Figura 62 apresenta a tela de manutenção de equipamentos.

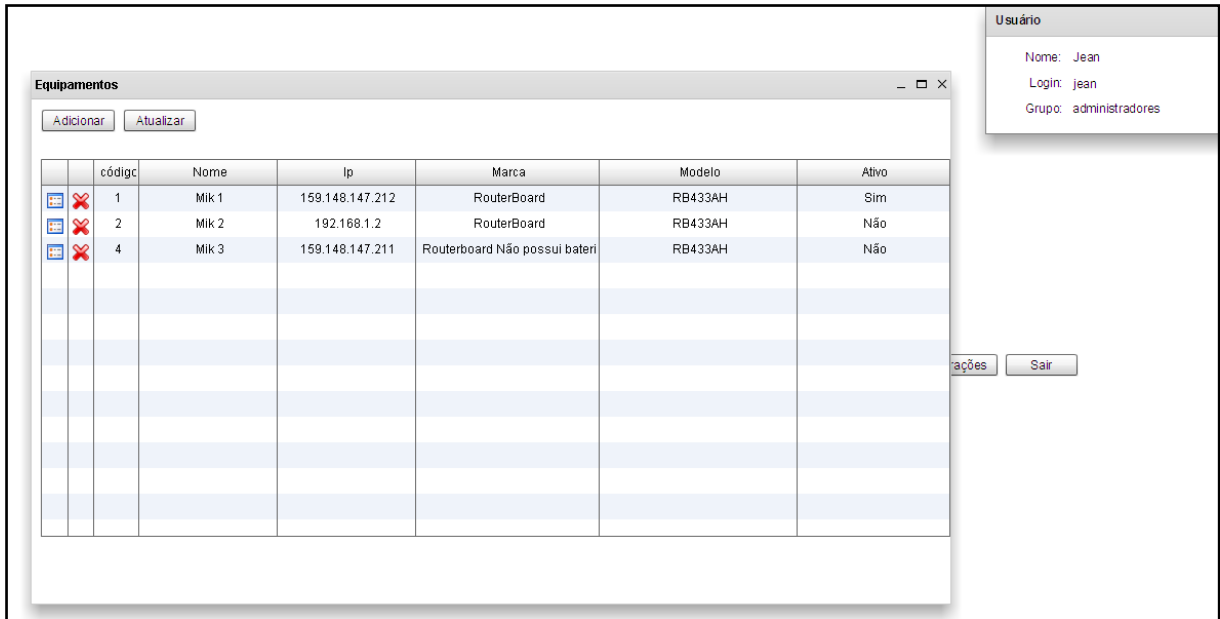


Figura 62 – Tela de manutenção de equipamentos

Após abrir a tela de manutenção de equipamentos, quando o usuário clica no botão Adicionar o sistema abre um painel sobre a tela para cadastrar um novo equipamento. A Figura 63 apresenta a tela para cadastrar um novo usuário.

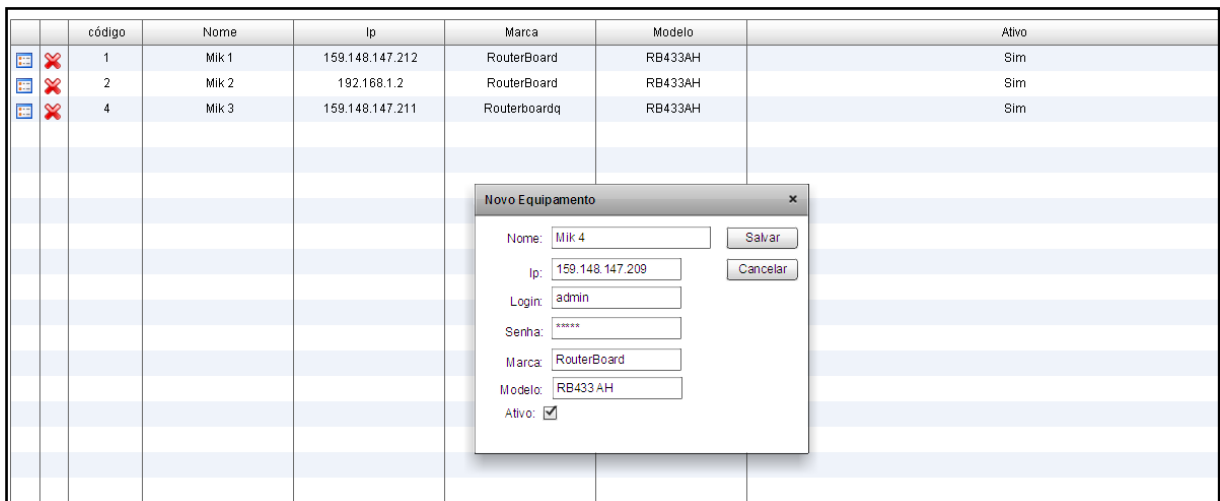


Figura 63 – Painel para cadastrar novo equipamento

3.3.8 Manutenção de responsáveis pelo monitoramento

Quando o usuário clica no botão Notificados, que está localizado no menu principal, o sistema abre a tela de manutenção de responsáveis pelo monitoramento, que são as pessoas que serão notificadas por *e-mail* quando algum parâmetro não estiver de acordo com sua regra. A Figura 64 apresenta a tela de manutenção de responsáveis pelo monitoramento.

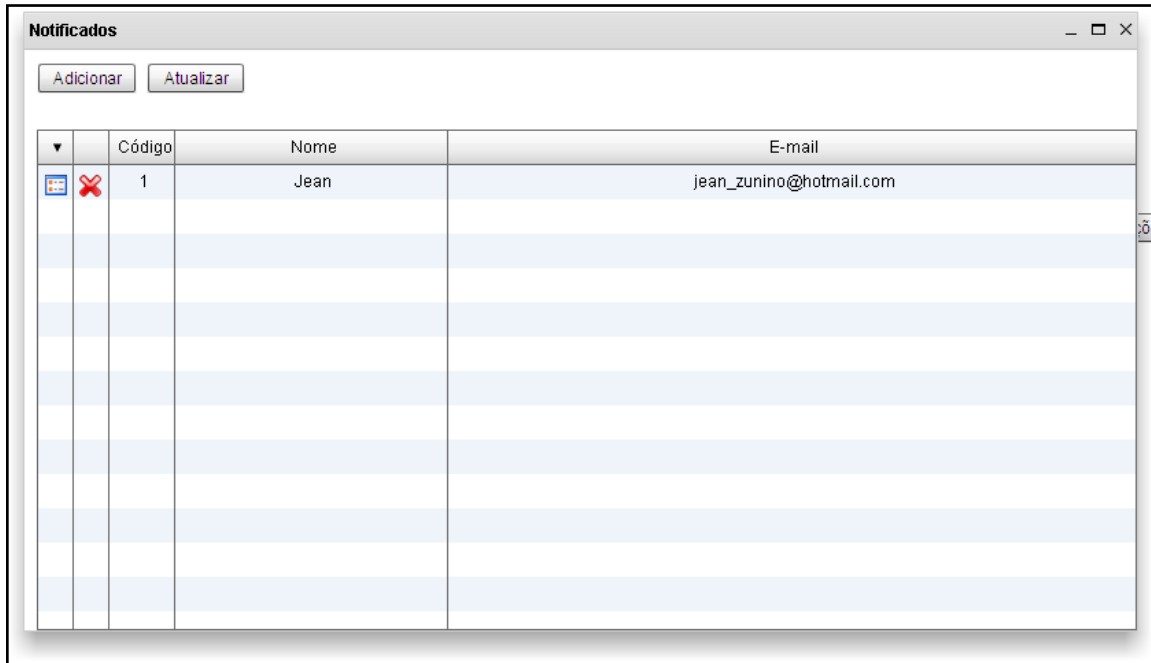


Figura 64 – Tela de manutenção de responsáveis pelo monitoramento

3.3.9 Manutenção de mensagens

Quando o usuário clica no botão *Mensagens*, que está localizado no menu principal, o sistema abre a tela de manutenção de mensagens. A Figura 65 apresenta a tela de manutenção de mensagens.

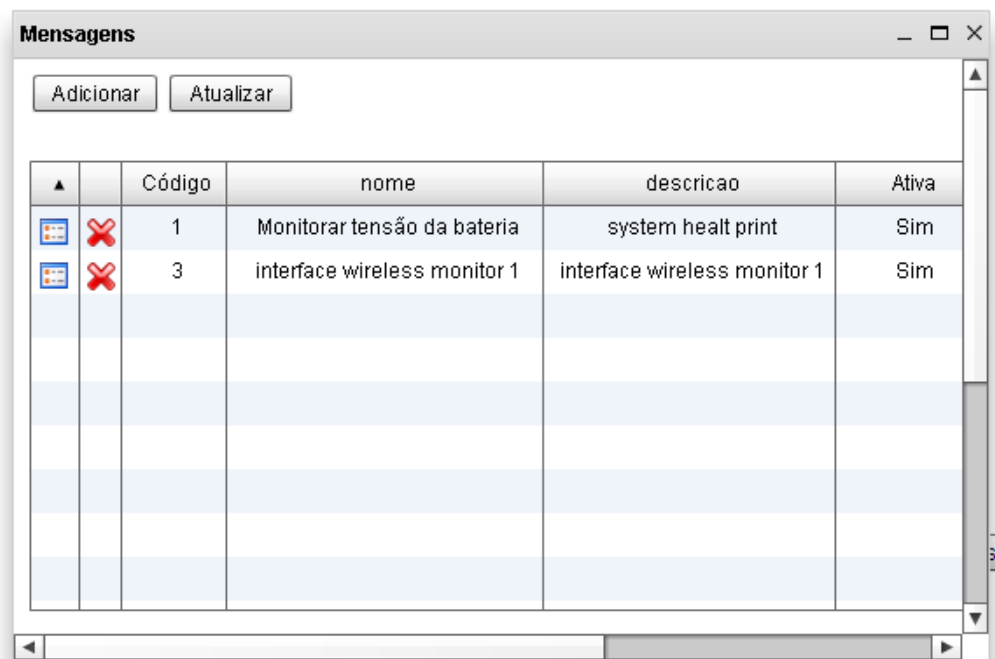


Figura 65 – Tela de manutenção de mensagens

Para cadastrar uma nova mensagem, é necessário que a resposta da Router Board para esta nova mensagem esteja compatível com a definição do GALS, apresentada na seção 3.3.1.1.3.

3.3.10 Manutenção de regras

Quando o usuário clica no botão `Regras`, que está localizado no menu principal, o sistema abre a tela de manutenção de regras. A Figura 66 apresenta a tela de manutenção de regras com duas regras cadastradas.

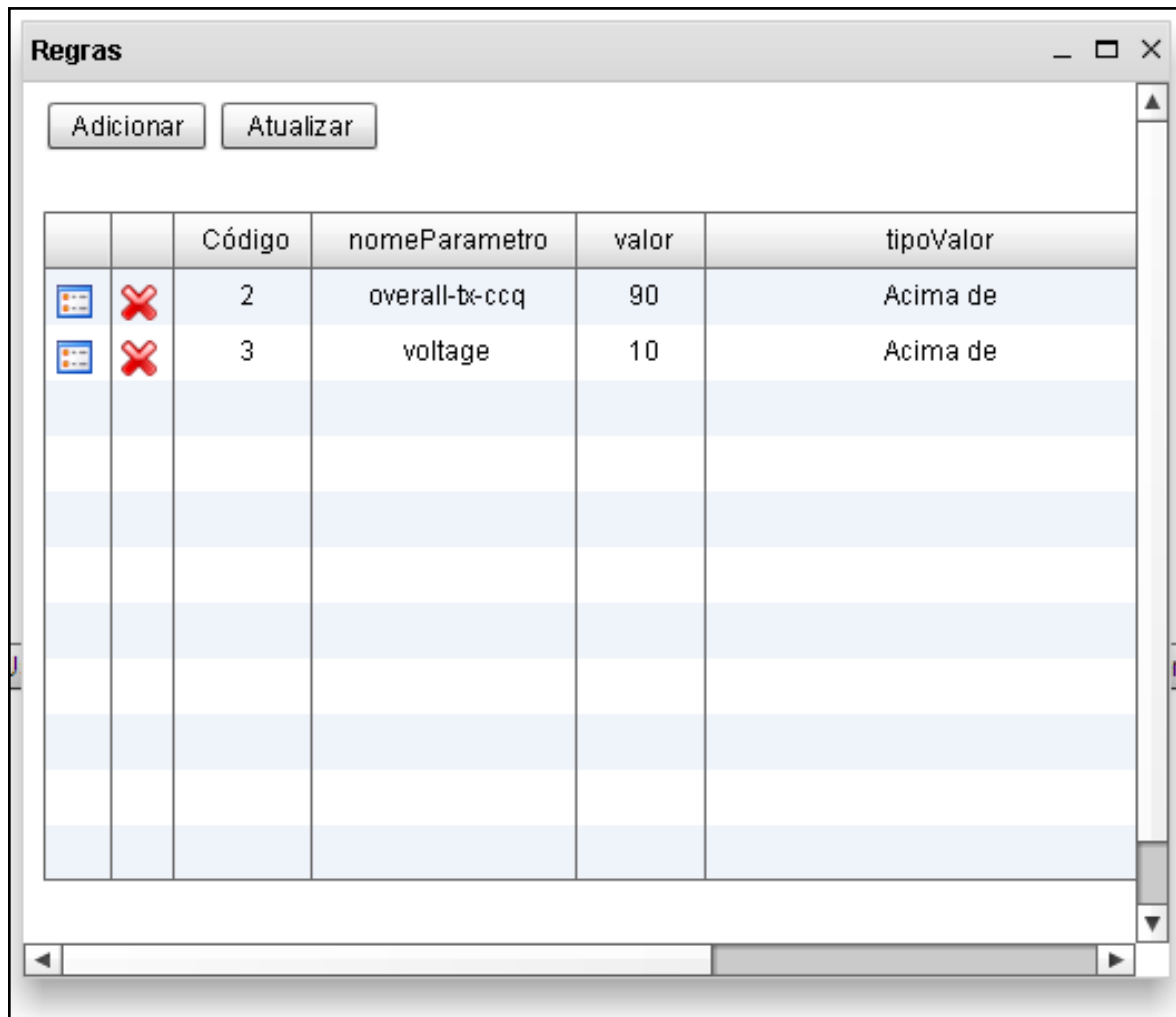
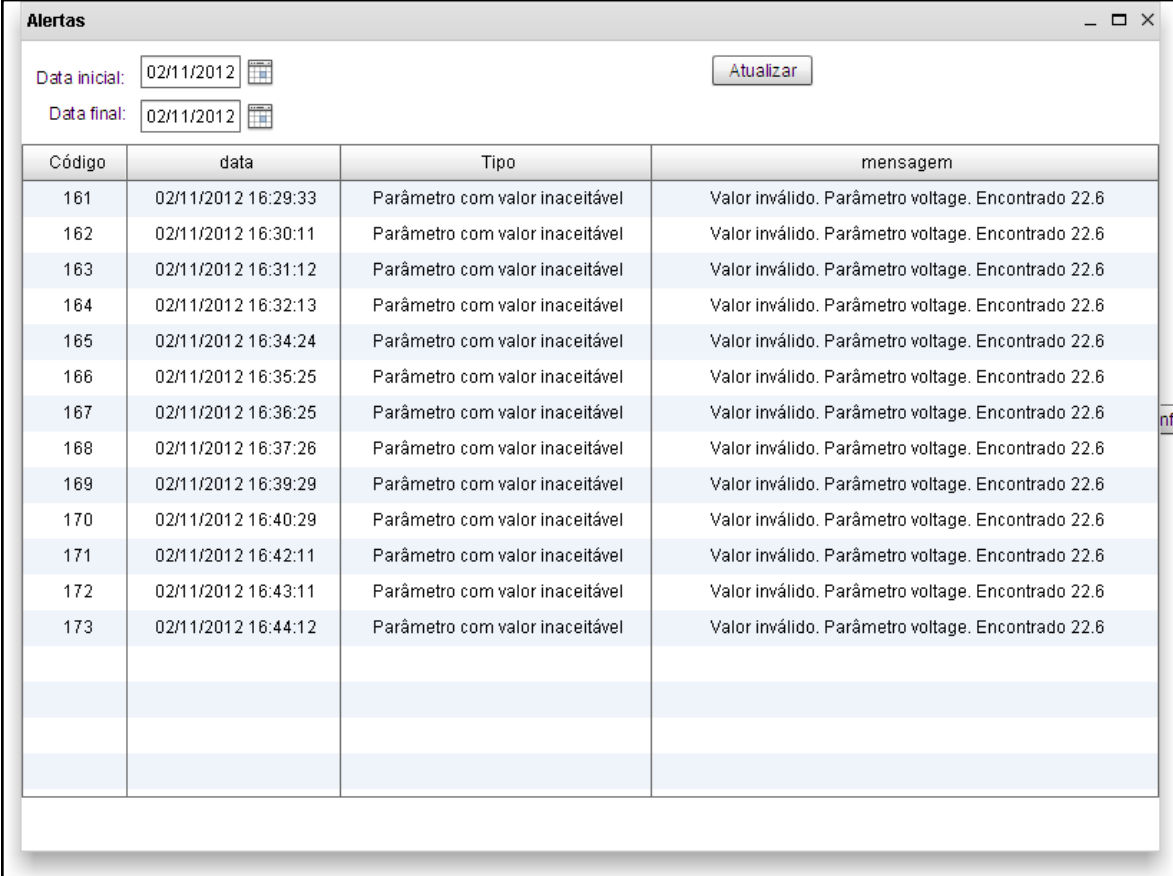


Figura 66 – Tela de manutenção de regras

3.3.11 Visualização de alertas gerados pelo sistema

Quando o usuário clica no botão *Alertas*, que está localizado no menu principal, o sistema abre a tela de visualização de alertas apresentando os alertas gerados no dia atual. A Figura 67 apresenta a tela de visualização de alertas.



Código	data	Tipo	mensagem
161	02/11/2012 16:29:33	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6
162	02/11/2012 16:30:11	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6
163	02/11/2012 16:31:12	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6
164	02/11/2012 16:32:13	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6
165	02/11/2012 16:34:24	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6
166	02/11/2012 16:35:25	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6
167	02/11/2012 16:36:25	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6
168	02/11/2012 16:37:26	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6
169	02/11/2012 16:39:29	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6
170	02/11/2012 16:40:29	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6
171	02/11/2012 16:42:11	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6
172	02/11/2012 16:43:11	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6
173	02/11/2012 16:44:12	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 22.6

Figura 67 – Tela de visualização dos alarmes gerados pelo sistema

Na tela de visualização de alertas o usuário pode fazer um filtro dos alertas pela data de geração do alerta alterando a data inicial e final e clicando no botão *Atualizar*.

Quando é cadastrado uma regra para uma mensagem e o nome do parâmetro não é encontrado na resposta da mensagem recebida pela RouterBoard o sistema gera um alarme do tipo *Parâmetro da regra não encontrado*. A Figura 68 apresenta exemplos de alertas gerados desse tipo.

Código	data	Tipo	mensagem
3075	11/11/2012 08:18:53	Parâmetro da regra não encontrado	Parâmetro não encontrado na mensagem: mensagem: Monitorar tensão da bateria, parâmetro: voltage, \\'
3077	11/11/2012 08:20:57	Parâmetro da regra não encontrado	Parâmetro não encontrado na mensagem: mensagem: Monitorar tensão da bateria, parâmetro: voltage, \\'
3079	11/11/2012 08:23:01	Parâmetro da regra não encontrado	Parâmetro não encontrado na mensagem: mensagem: Monitorar tensão da bateria, parâmetro: voltage, \\'
3081	11/11/2012 08:25:03	Parâmetro da regra não encontrado	Parâmetro não encontrado na mensagem: mensagem: Monitorar tensão da bateria, parâmetro: voltage, \\'
3083	11/11/2012 15:43:44	Parâmetro da regra não encontrado	Parâmetro não encontrado na mensagem: mensagem: Monitorar tensão da bateria, parâmetro: voltage, \\'
3085	11/11/2012 15:45:56	Parâmetro da regra não encontrado	Parâmetro não encontrado na mensagem: mensagem: Monitorar tensão da bateria, parâmetro: voltage, \\'
3088	11/11/2012 15:52:38	Parâmetro da regra não encontrado	Parâmetro não encontrado na mensagem: mensagem: Monitorar tensão da bateria, parâmetro: voltage, \\'
3090	11/11/2012 15:54:55	Parâmetro da regra não encontrado	Parâmetro não encontrado na mensagem: mensagem: Monitorar tensão da bateria, parâmetro: voltage, \\'
3092	11/11/2012 15:57:18	Parâmetro da regra não encontrado	Parâmetro não encontrado na mensagem: mensagem: Monitorar tensão da bateria, parâmetro: voltage, \\'
3094	11/11/2012 15:59:30	Parâmetro da regra não encontrado	Parâmetro não encontrado na mensagem: mensagem: Monitorar tensão da bateria, parâmetro: voltage, \\'
3073	11/11/2012 08:16:50	Parâmetro da regra não encontrado	Parâmetro não encontrado na mensagem: mensagem: Monitorar tensão da bateria, parâmetro: voltage, \\'

Figura 68 – Alertas gerados do tipo Parâmetro da regra não encontrado

3.3.12 Tela de monitoramento

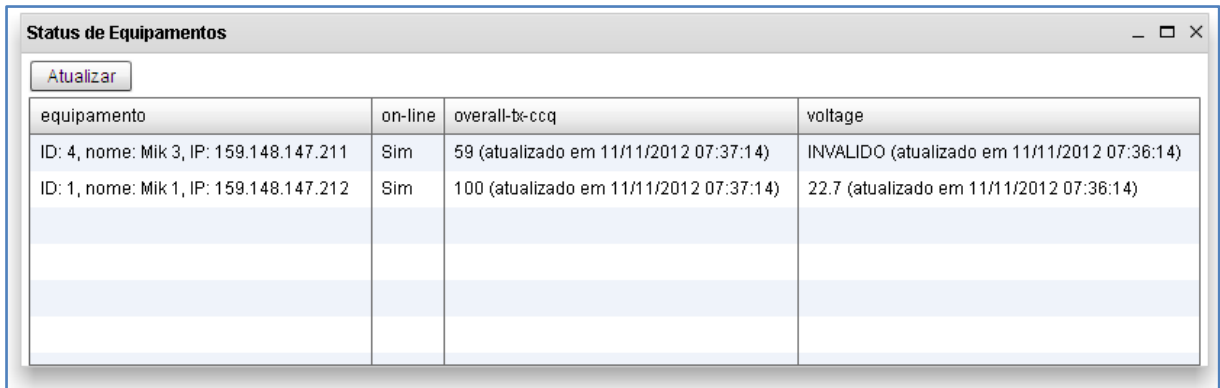
Quando o usuário clica no botão *Status*, que está localizado no menu principal, o sistema abre a tela de visualização de *status* de equipamentos apresentando o status dos equipamentos que estão sendo monitorados.

A tabela de informações é criada de forma dinâmica, sendo as duas primeiras colunas fixas (*equipamento* e *on-line*) e as outras são criadas em tempo de execução, de acordo com as regras cadastradas. A Figura 69 apresenta a tela de visualização de *status* em um momento que se tinha apenas uma regra cadastrada.

equipamento	on-line	voltage
ID: 2, nome: Mik 2, IP: 192.168.1.2	Sim	18 (atualizado em 03/11/2012 01:46:11)
ID: 1, nome: Mik 1, IP: 159.148.147.212	Sim	22.6 (atualizado em 03/11/2012 01:46:11)

Figura 69 – Tela de visualização de *status* com uma regra cadastrada no sistema

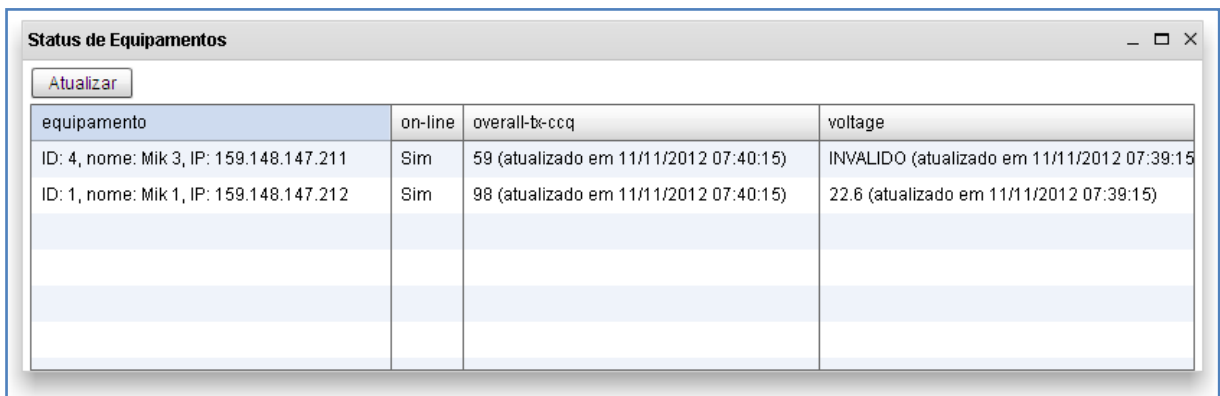
A Figura 70 apresenta a tela de visualização de *status* após o cadastro de uma segunda regra. A RouterBoard de IP 159.148.147.211 é do modelo RB433, esse modelo não possui a função de monitoramento da tensão elétrica da bateria, pois isso seu valor na coluna *voltage* aparece como *INVALIDO*.



equipamento	on-line	overall-bx-ccq	voltage
ID: 4, nome: Mik 3, IP: 159.148.147.211	Sim	59 (atualizado em 11/11/2012 07:37:14)	INVALIDO (atualizado em 11/11/2012 07:36:14)
ID: 1, nome: Mik 1, IP: 159.148.147.212	Sim	100 (atualizado em 11/11/2012 07:37:14)	22.7 (atualizado em 11/11/2012 07:36:14)

Figura 70 - Tela de visualização de *status* com duas regras cadastradas no sistema

A Figura 71 apresenta a tela de visualização de *status* atualizada depois de três minutos. É possível perceber em cada célula das colunas *overall-tx-ccq* e *voltage* que a data de atualização foi atualizada.



equipamento	on-line	overall-bx-ccq	voltage
ID: 4, nome: Mik 3, IP: 159.148.147.211	Sim	59 (atualizado em 11/11/2012 07:40:15)	INVALIDO (atualizado em 11/11/2012 07:39:15)
ID: 1, nome: Mik 1, IP: 159.148.147.212	Sim	98 (atualizado em 11/11/2012 07:40:15)	22.6 (atualizado em 11/11/2012 07:39:15)

Figura 71 – Atualizando tela de monitoramento

3.3.13 Tela de configurações do sistema

Quando o usuário clica no botão *Configurações*, que está localizado no menu principal, o sistema abre a tela de configuração do sistema apresentando as configurações atuais.

A Figura 72 apresenta a tela de configurações do sistema.

Configuração

E-mail para Sistema Notificador (G-mail):

Senha do e-mail:

Intervalo de tempo para o monitoramento (em Minutos):

Intervalo de tempo para notificação (em Minutos):

Figura 72 – Tela de configurações do sistema

3.3.14 E-mail do Sistema Notificador

A Figura 73 apresenta o formato do e-mail enviado pelo Sistema Notificador para os responsáveis pelo monitoramento. Algumas informações foram apagadas por questões de privacidade.

[Redacted Name] 10 nov (2 dias atrás)

Abaixo segue os alarmes com parâmetros que possuem valores inaceitáveis:

Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.6
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.6
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.6
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7
 Nome: Mik 1 Ip: 159.148.147.212 motivo: Valor inválido. Parâmetro voltage. Encontrado 22.7

Figura 73 – E-mails enviados para o responsável pelo monitoramento

3.4 IMPLANTAÇÃO E TESTES DO SISTEMA NO PROVEDOR DE ACESSO

Em Novembro de dois mil e doze foi realizado a implantação do sistema e um teste piloto no provedor de acesso Central Net. Esta seção está dividida em duas partes, a primeira descreve como foi a implantação do sistema e a segunda descreve como foram realizados os testes.

3.4.1 Implantação do sistema no provedor de acesso

Para realizar a implantação do sistema foram seguidos os seguintes passos:

- a) realizado conexão remota utilizando o TeamViewer 7 com uma das máquinas do provedor de acesso;
- b) realizado a instalação e configuração do Tomcat versão 7, o banco de dados PostgreSQL versão 8.4 e a JDK versão 7 na máquina do provedor de acesso;
- c) criado uma base de dados no PostgreSQL chamada `networkmanagerj` e executado o *script* para a criação das tabelas (o *script* foi gerado pela ferramenta PowerDesign, baseado no PDM);
- d) feito *deploy* dos arquivos do sistema na pasta `webapps` do servidor de aplicação Tomcat;
- e) criado atalhos na área de trabalho para o arquivo `startup.bat` do Tomcat e para a Uniform Resource Locator (URL) do sistema.
- f) iniciado o Tomcat;
- g) *login* no sistema utilizando usuário padrão do sistema (*login* `jean` e senha `123`);
- h) criar um *help* para cada tela do sistema.

A Figura 74 apresenta o resultado da execução do `script` para a criação das tabelas na base de dados `networkmanagerj`, onde foram criadas 11 tabelas.

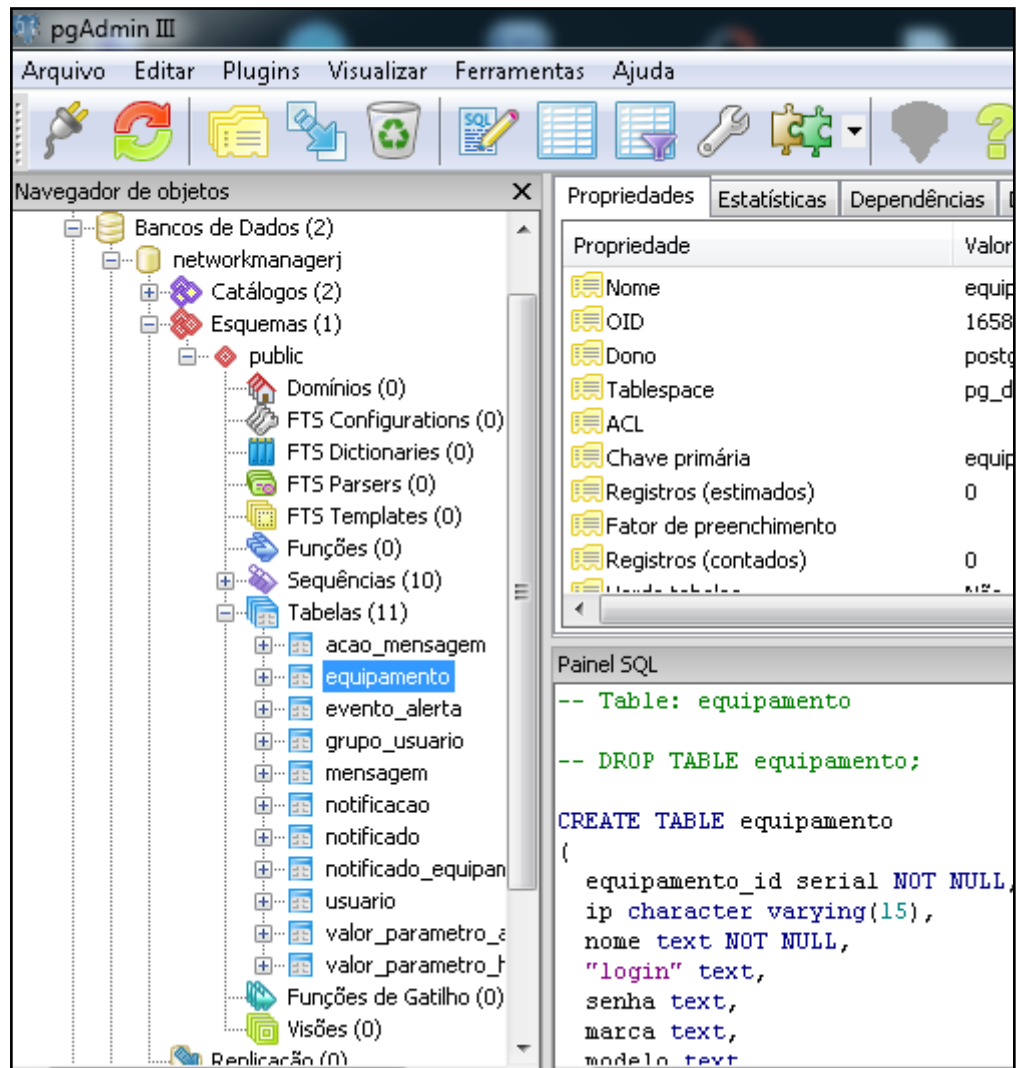


Figura 74 – Tabelas criadas no banco de dados

A Figura 75 apresenta um *print screen* da tela no momento da transferência dos arquivos para a pasta `webapps` do Tomcat localizado na máquina do provedor de acesso.

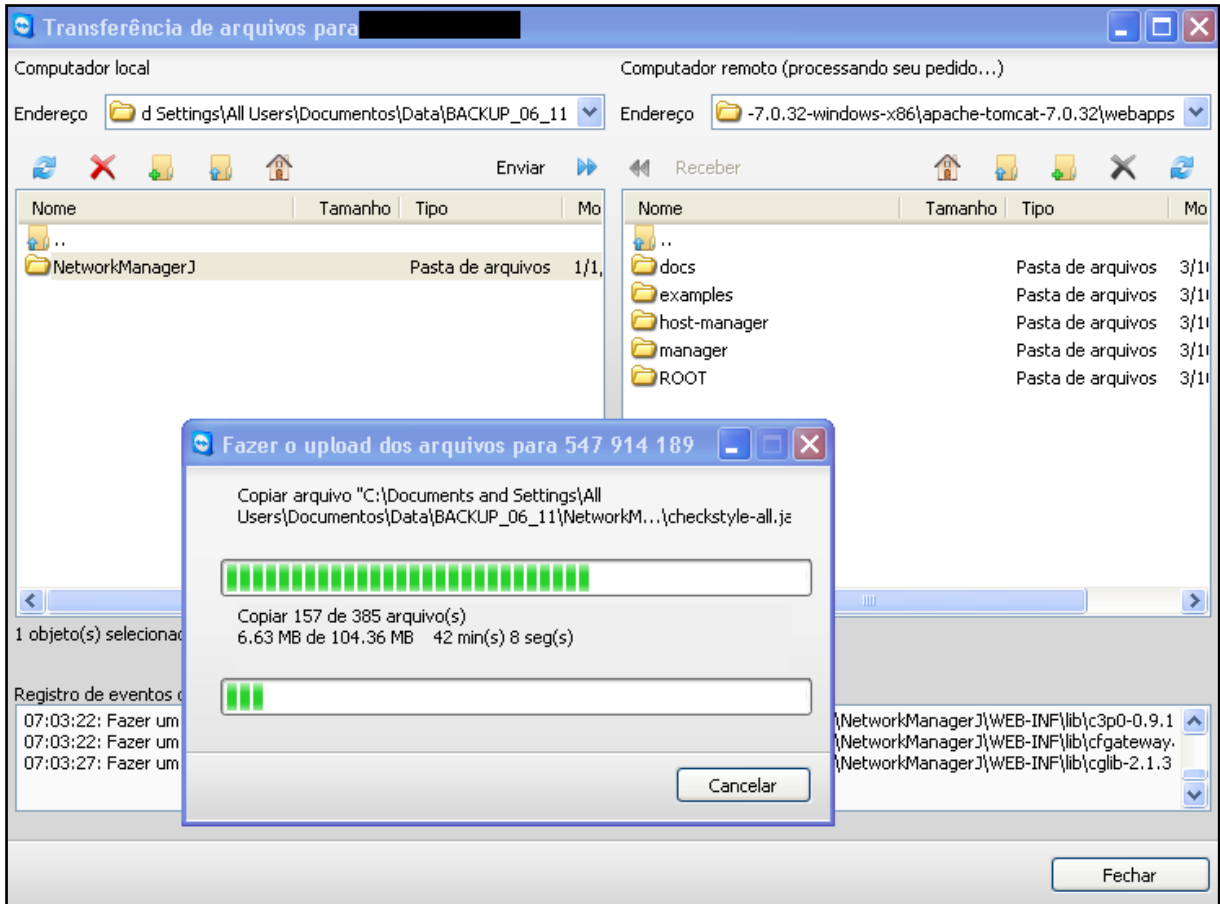


Figura 75 – Transferência dos arquivos para a pasta `webapps` do Tomcat localizado na máquina do provedor de acesso

Na implantação do sistema os problemas começaram a surgir no passo descrito no item g. Ao tentar acessar a URL do sistema, a página não era encontrada. Ao acessar a URL `http://localhost:8080`, percebeu-se que não era o Tomcat que estava rodando na porta 8080, mas sim uma aplicação Web do banco de dados PostgreSQL que foi instalado.

Segundo Valim (2011), a porta padrão do Tomcat é 8080 e pode ser alterada seguindo os seguintes passos:

- localizar a pasta `conf` nos diretórios de instação do Tomcat;
- localizar e abrir o arquivo `server.xml` que está na pasta `conf`;
- editar o valor da propriedade `port` da linha ilustrada na Figura 76.

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8180 -->
<Connector port="8080" maxHttpHeaderSize="8192"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" redirectPort="8443" acceptCount="100"
connectionTimeout="20000" disableUploadTimeout="true" />
```

Figura 76 – Linha do arquivo `server.xml`

Após identificar este problema foi alterada a porta padrão do Tomcat para a porta 8083. A Figura 77 apresenta o sistema sendo acessado pela porta 8083.

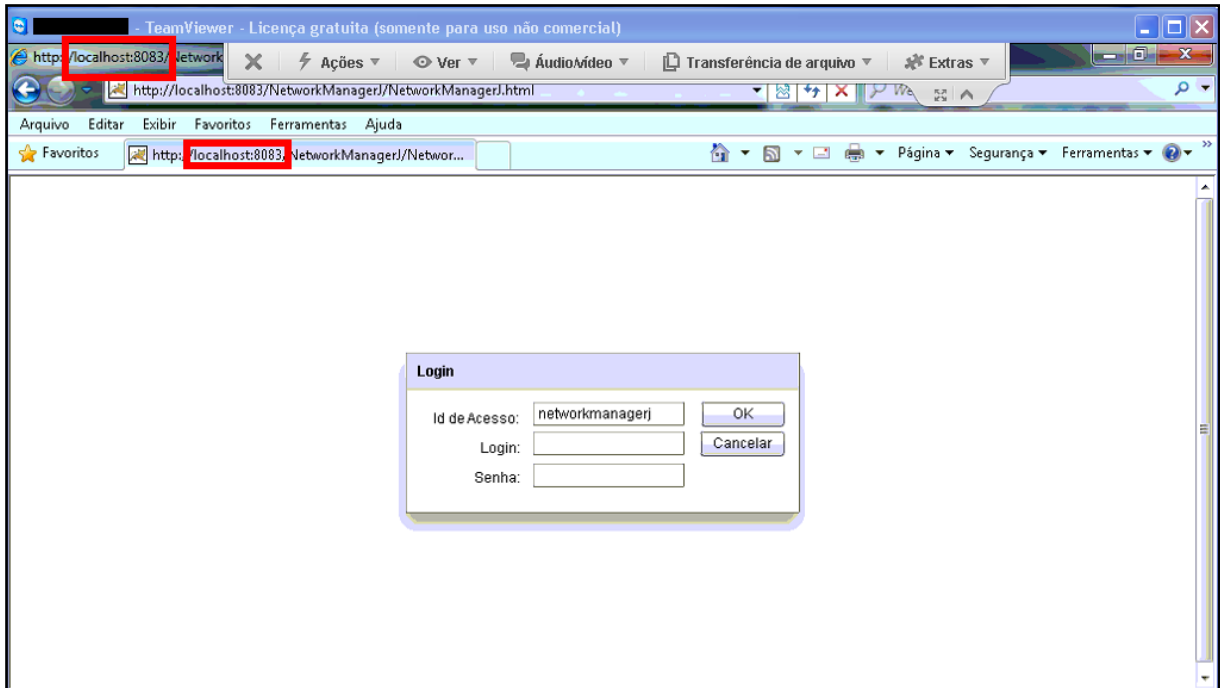


Figura 77 – Acessando o sistema no provedor de acesso pela porta 8083

3.4.2 Teste piloto

Segundo Pfleeger (2004, p. 338) um teste piloto instala o sistema, experimentalmente, para os usuários utilizarem o sistema como se tivesse sido instalado permanentemente. Para Pfleeger (2004, p. 338) os testes-piloto têm como base o trabalho diário do sistema para testar todas as funções.

O teste piloto no provedor de acesso Central net tem o principal objetivo de avaliar a usabilidade das principais funcionalidades do sistema em campo. Para testar a usabilidade utilizou-se um usuário não avançado (Cesar) sem nenhum conhecimento do sistema acompanhado por um usuário avançado (Jean) que conhece o sistema.

Para realizar os testes foi seguido um formulário de procedimentos que deveriam ser executados (primeira parte do Apêndice A).

Após realizar o *login* com *login* e senha padrão do sistema, foi solicitado ao usuário Cesar abrir a tela de manutenção de usuários e cadastrar um novo usuário.

Observando o usuário Cesar (através da conexão remota) realizar o que foi solicitado, foi possível perceber que mesmo sem ter nenhum contato anterior com o sistema, ele conseguiu localizar a tela de manutenção de usuários e cadastrar um novo usuário sem nenhum problema.

A Figura 78 apresenta o momento em que o usuário Cesar estava cadastrando seu usuário no sistema.

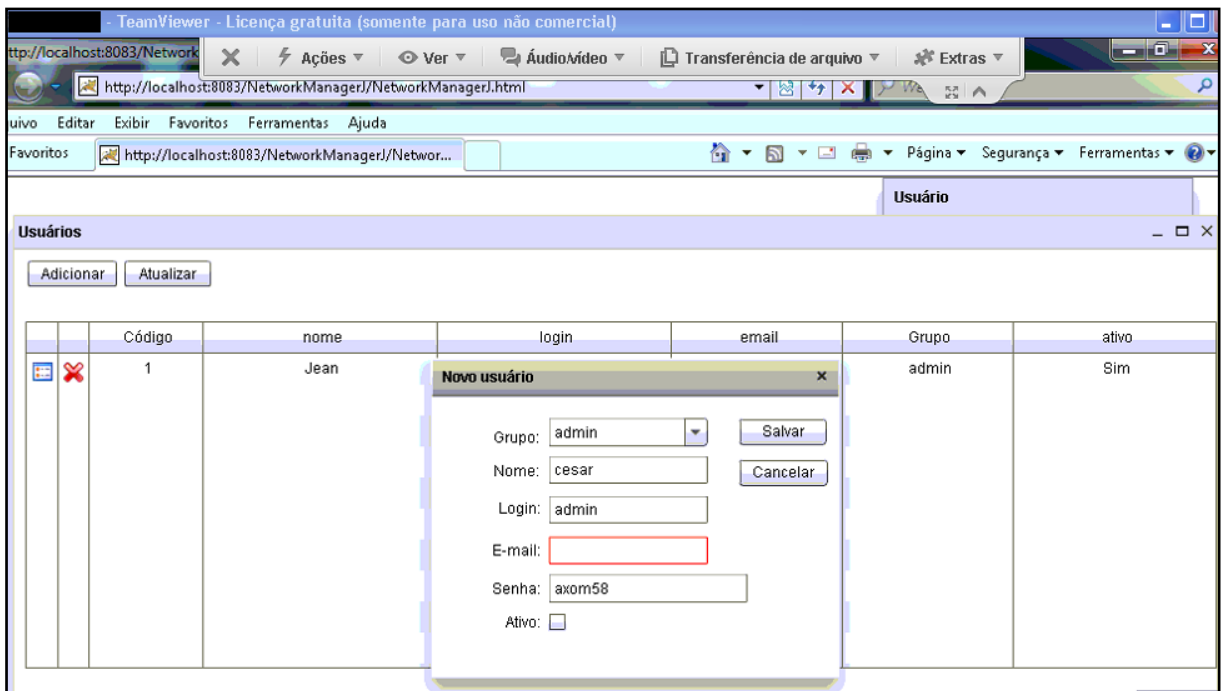



Figura 78 – Cadastro de novo usuário no provedor de acesso

Para o cadastro de novo usuário foi sugerido que a senha aparecesse no formato ****, igual a tela de login, para não aparecer a senha do usuário. A alteração foi realizada, conforme pode ser visto na seção 3.3.6.

Após o cadastro de usuário foi solicitado ao usuário Cesar abrir a tela de manutenção de equipamentos e cadastrar as RouterBoards modelo RB433AH do provedor de acesso.

Para identificar as RouterBoards Cesar abriu a Winbox e clicou no ícone . A Figura 79 apresenta o momento em que Cesar identificava as RouterBouters modelo RB433AH para serem cadastradas.

MAC Address	IP Address	Identity	Version	Board ...
00:0C:42:3F:7F:9B	192.168.20.23	TORRE BOA ESP...	3.25	RB433AH
00:0C:42:52:59:79	192.168.20.19	MORRO LINK	3.22	RB433AH
00:0C:42:5E:15:B1	192.168.20.13	RB MORRO SALTO	3.25	RB433AH
00:0C:42:5E:16:02	192.168.20.12	COLONIA SALTO...	3.25	RB433AH
00:0C:42:69:11:FE	192.168.20.2	CASA RAMAL CE...	4.10	RB411
00:0C:42:69:15:6F	192.168.20.3	RB HORATORIO	4.11	RB433AH
00:0C:42:6A:A4:9F	192.168.20.4	TORRE MAURINO	4.11	RB433AH
00:0C:42:68:4A:09	192.168.20.18	RB MORRO GALI...	3.22	RB433
00:0C:42:98:95:7E	192.168.20.42	MikroTik	4.14	RB433
00:0C:42:AF:F7:CB	192.168.20.48	RB/TORRE VOLT...	5.5	RB433AH
00:15:6D:67:A4:B9	192.168.20.21	RB TORRE PINH...	4.14	RB433AH
00:15:6D:67:AB:9E	192.168.20.43	RB MORRO SAU...	4.11	RB433AH
00:15:6D:68:32:9A	192.168.20.20	LINK BOA ESP /P...	4.11	RB433AH
00:15:6D:94:69:F9	192.168.20.24	MWF BARRAGE...	4.11	RB411
00:60:97:83:8B:4C	192.168.1.254	Roteador	5.16	x86

Figura 79 – Equipamentos no provedor de acesso Central Net.

Observando o usuário Cesar realizar o cadastro, foi possível perceber que ele conseguiu localizar a tela de manutenção de equipamentos e cadastrar um novo equipamento sem nenhum problema.

A Figura 80 apresenta o momento em que o usuário Cesar estava cadastrando um de seus equipamentos.

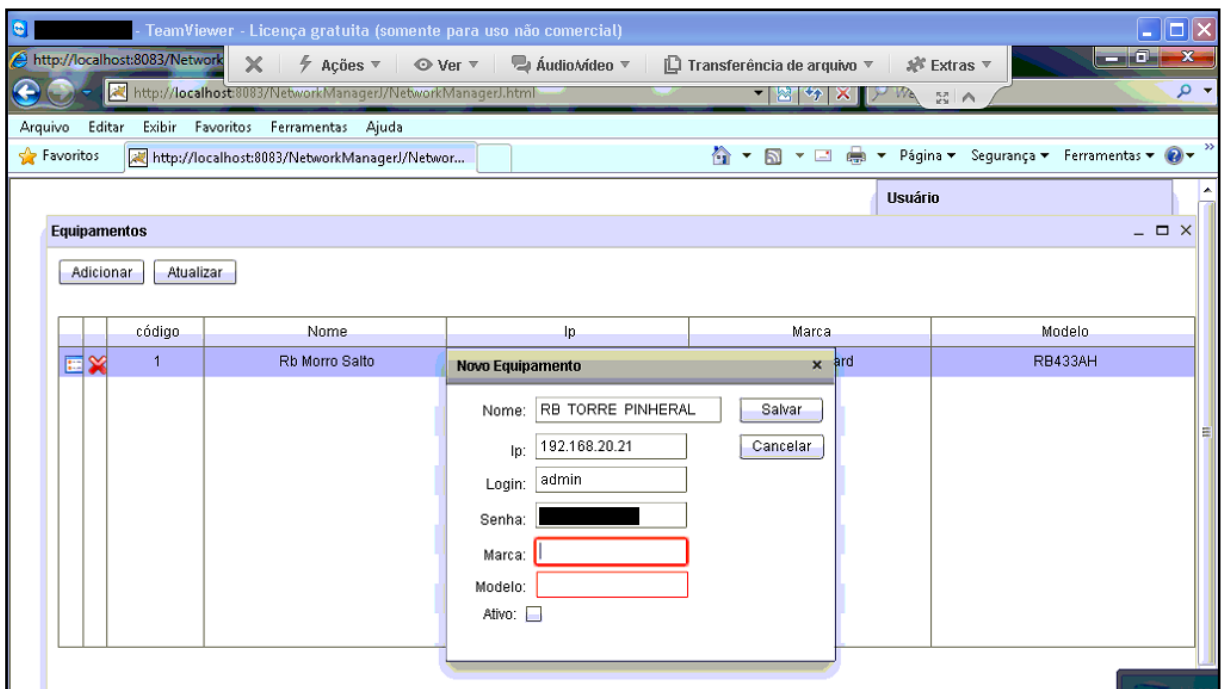


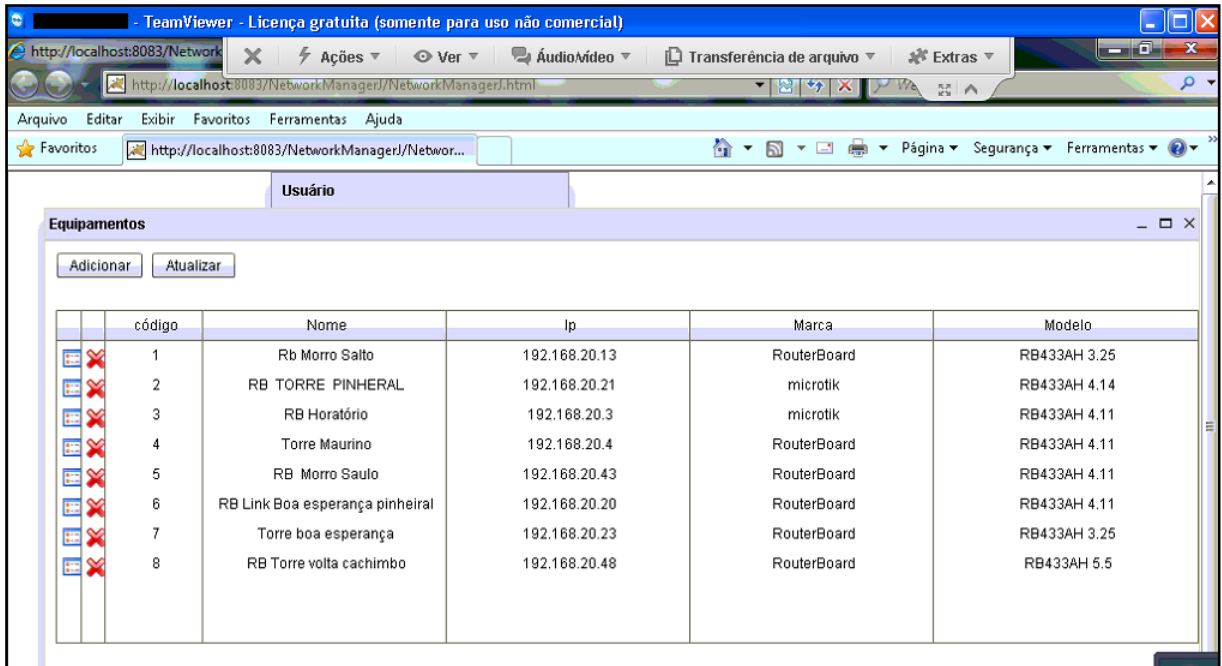
Figura 80 – Cadastro de novo equipamento

Para a tela de visualização de equipamento foi sugerido adicionar uma coluna informando se o equipamento está ativo. A alteração foi realizada, conforme foi apresentado

na seção 3.3.7.

Para o cadastro de novo equipamento também foi sugerido que a senha aparecesse no formato ****. A alteração foi realizada, conforme pode ser visto na seção 3.3.7.

Foram cadastrados oito equipamentos. A Figura 81 apresenta os equipamentos cadastrados no provedor de acesso.



	código	Nome	Ip	Marca	Modelo
	1	Rb Morro Salto	192.168.20.13	RouterBoard	RB433AH 3.25
	2	RB TORRE PINHERAL	192.168.20.21	microtik	RB433AH 4.14
	3	RB Horatório	192.168.20.3	microtik	RB433AH 4.11
	4	Torre Maurino	192.168.20.4	RouterBoard	RB433AH 4.11
	5	RB Morro Saulo	192.168.20.43	RouterBoard	RB433AH 4.11
	6	RB Link Boa esperança pinheiral	192.168.20.20	RouterBoard	RB433AH 4.11
	7	Torre boa esperança	192.168.20.23	RouterBoard	RB433AH 3.25
	8	RB Torre volta cachimbo	192.168.20.48	RouterBoard	RB433AH 5.5

Figura 81 – Equipamentos cadastrados no provedor de acesso

Após o cadastro dos equipamentos foi realizado o cadastro da mensagem para monitorar a tensão da bateria. Esse cadastro foi realizado pelo usuário Jean, pois foi necessário um usuário avançado com um conhecimento de como funciona o envio das mensagens. A Figura 82 apresenta o cadastro da mensagem *system health print*.

Código Ação	Tipo ação	Texto
0	Enviar	system health print
0	Aguardar] >

Figura 82 – Cadastro da mensagem *system health print*

Após o usuário Jean cadastrar a mensagem, o sistema iniciou o monitoramento dos equipamentos cadastrados. A Figura 83 apresenta um trecho do log do servidor de aplicação Tomcat.

```

****Logando em Id 1, Ip: 192.168.20.13
Aguardando ] > de Id 1, Ip: 192.168.20.13

****LOGADO em Id 1, Ip: 192.168.20.13
Processou ação: tipo ação: 0 texto: system health print
Texto: [admin@RB MORRO SA
LT01 > system health print[admin@RB MORRO SALT01 > system health print

Processou ação: tipo ação: 1 texto: ] >
Texto: fan-mode: auto
       use-fan: main
       active-fan: none
       voltage: 12
[admin@RB MORRO SALT01 >

```

Figura 83 – Log do monitoramento do primeiro equipamento cadastrado

Ao iniciar o monitoramento foram encontrados dois problemas no *parser* da mensagem.

O primeiro problema identificado está relacionado com o nome de cada RouterBoard e a definição dos analisares léxico e sintático para o *parser* da mensagem. Conforme

apresentado na seção 3.3.1.1.3, a definição do GALS possui o *token* email e o não terminal `<linha_digitar_comando>` que é formado pela sequência `"[" email "]" ">`. A última linha que aparece na Figura 83 deveria ser interpretada como um não terminal `<linha_digitar_comando>`, mas isso não ocorria pois possuía espaço em alguns locais onde não deveria.

Para corrigir este problema foi conectado em cada RouterBoard utilizando o Winbox, acessado o menu `System/Identity` e alterado o nome de cada RouterBoard para não possuir espaços. A Figura 84 apresenta o momento em que se estava alterando o nome da RouterBoard.

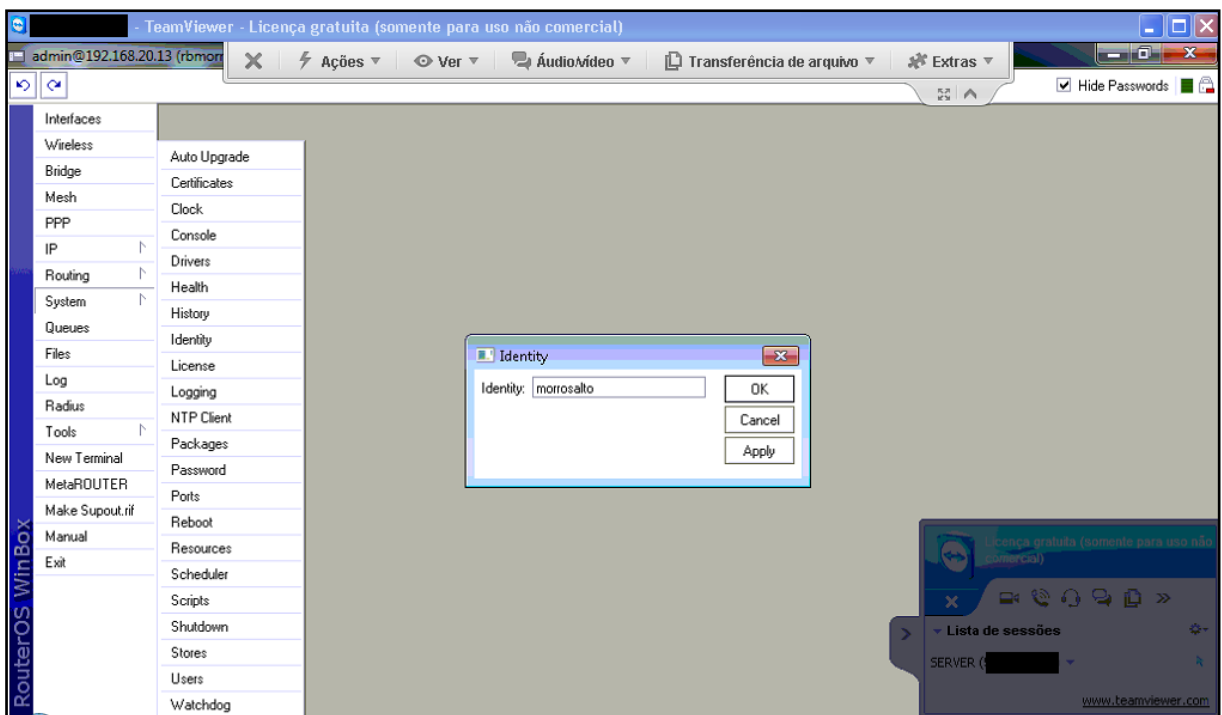


Figura 84 – Alterando nome da RouterBoard através da Winbox

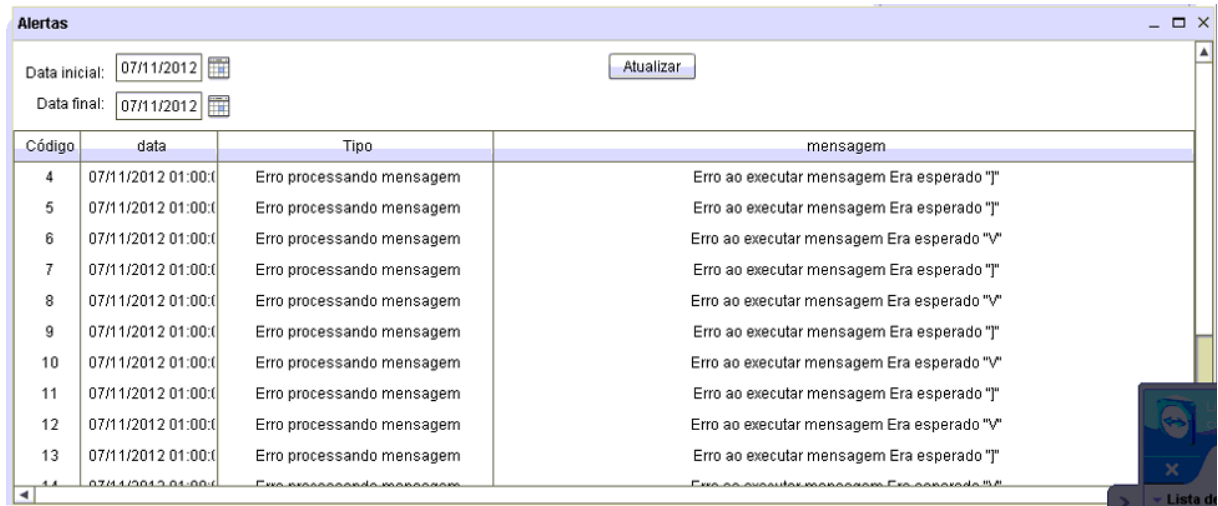
O segundo problema identificado está relacionado com o valor do parâmetro `voltage`. No momento da implementação deste trabalho foi utilizado Mikrotiks com versões iguais ou maiores que 5.22, no provedor de acesso todos as RouterBoards estavam em versões anteriores. As RouterBoards com versão 3.25 respondiam a mensagem `system health print` com o valor do parâmetro `voltage` sem o `v` atrás do número, fazendo gerar uma exceção no analisador sintático.

Para corrigir este problema foi alterado a especificação do GALS, alterando a gramática e os não terminais para aceitar o valor do parâmetro `voltage` com o `v` ou sem o `v`. O Quadro 12 apresenta a alteração da gramática (apresentada na Figura 38) modificando o não terminal `<shp_voltage>` e criando um novo não terminal `<shp_voltage_unid>`.

```
<shp_voltage> ::= voltage ":" numero <shp_voltage_unid>;
<shp_voltage_unid> ::= 'V' | "V";
```

Quadro 12 – Alteração na gramática para o parâmetro *voltage*

Quando ocorreram estes dois erros no *parser* da mensagem o sistema gerou alertas que foram visualizados na tela de visualização de alertas. A Figura 85 apresenta a tela de visualização de alertas e alguns dos alertas gerados.



Código	data	Tipo	mensagem
4	07/11/2012 01:00:00	Erro processando mensagem	Erro ao executar mensagem Era esperado "]"
5	07/11/2012 01:00:00	Erro processando mensagem	Erro ao executar mensagem Era esperado "]"
6	07/11/2012 01:00:00	Erro processando mensagem	Erro ao executar mensagem Era esperado "V"
7	07/11/2012 01:00:00	Erro processando mensagem	Erro ao executar mensagem Era esperado "]"
8	07/11/2012 01:00:00	Erro processando mensagem	Erro ao executar mensagem Era esperado "V"
9	07/11/2012 01:00:00	Erro processando mensagem	Erro ao executar mensagem Era esperado "]"
10	07/11/2012 01:00:00	Erro processando mensagem	Erro ao executar mensagem Era esperado "V"
11	07/11/2012 01:00:00	Erro processando mensagem	Erro ao executar mensagem Era esperado "]"
12	07/11/2012 01:00:00	Erro processando mensagem	Erro ao executar mensagem Era esperado "V"
13	07/11/2012 01:00:00	Erro processando mensagem	Erro ao executar mensagem Era esperado "]"
14	07/11/2012 01:00:00	Erro processando mensagem	Erro ao executar mensagem Era esperado "]"

Figura 85 – Alertas gerados pelo erro de *parser* da mensagem

Após corrigir os erros identificados foi solicitado ao usuário Cesar abrir a tela de manutenção de regras e cadastrar uma nova regra para o parâmetro *voltage* da mensagem *system health print*.

Observando o usuário Cesar realizar o cadastro da regra, foi possível perceber que ele conseguiu localizar a tela de manutenção de regras e cadastrar uma nova regra, porém surgiram algumas dúvidas devido a falta de conhecimento de como funciona o sistema. Essas dúvidas foram sanadas com o usuário Jean.

Em um outro momento foi solicitado ao usuário Cesar para alterar a regra cadastrada anteriormente para ter um valor diferente. Nesse momento o usuário Cesar não teve dúvidas de como utilizar o sistema: ele abriu a tela de manutenção de regras, clicou no ícone de edição da regra, alterou o valor e clicou em salvar.

A Figura 86 apresenta o momento em que o usuário Cesar estava cadastrando a regra para o parâmetro *voltage* da mensagem *system health print*.

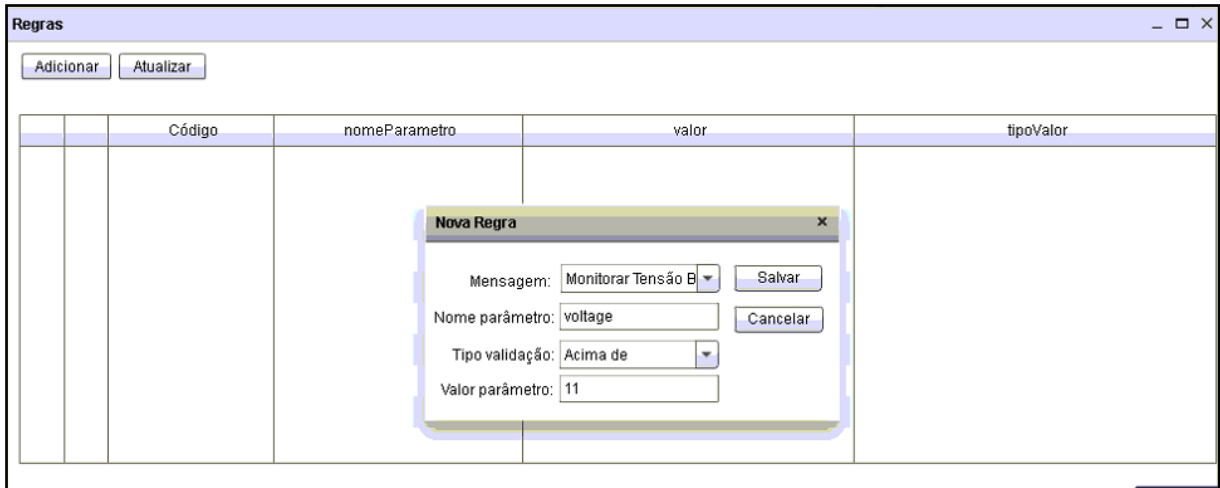


Figura 86 – Cadastro da regra para o parâmetro voltage

Para o cadastro de regras foi sugerido alterar a caixa de seleção *Mensagem* e a caixa de seleção *Tipo validação* para não permitir a edição do campo selecionado. A Figura 87 apresenta o momento em que observava-se o usuário Cesar, sem ter conhecimento da tela, editar a caixa de seleção colocando o nome da mensagem. Esta sugestão também se encontra na seção 4.1.

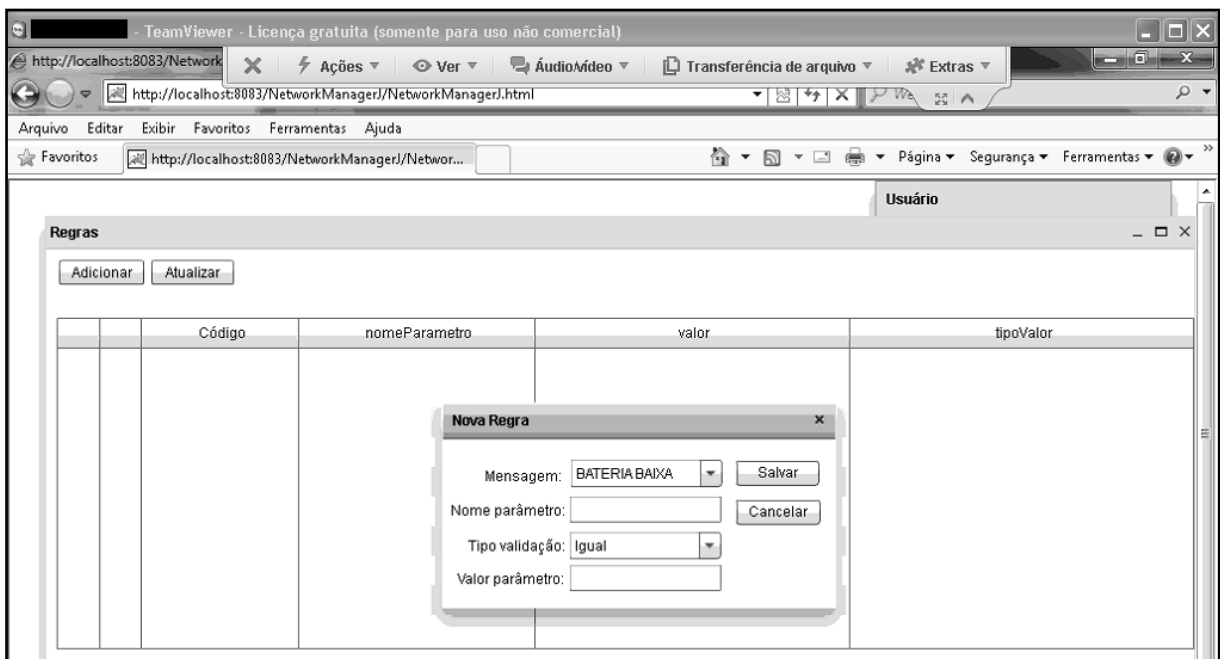


Figura 87 – Caixa de seleção mensagem e tipo validação permitem edição do texto selecionado

Após o cadastro da regra foi possível observar no log do Tomcat os logs realizados pelo sistema para o monitoramento da regra cadastrada. A Figura 88 apresenta o trecho do log do Tomcat para o monitoramento da regra cadastrada.


```

Processou abdo: tipo acao: 0 texto: system health print
Texto: [admin@rbtorrepinheira
rall > system health print [admin@rbtorrepinheira] > system health print

Processou abdo: tipo acao: 1 texto: 1 >
Texto: fan-mode: auto
       use-fan: main
       active-fan: none
       voltage: 12.20
[admin@rbtorrepinheira] >
Processou abdo: tipo acao: 2 texto: RB MORRO
Texto: fan-mode: auto
       use-fan: main
       active-fan: none
       voltage: 12.20
[admin@rbtorrepinheira] >
Conferindo parâmetros da mensagem nome: Monitorar Tensão Bateria
Conferindo valor mensagem: Monitorar Tensão Bateria, parâmetro: voltage, valor e
sperado: Acima de: 11

```

Figura 88 - Trecho do log do Tomcat para o monitoramento da regra cadastrada

Após o cadastro da regra foi solicitado ao usuário Cesar abrir a tela de status dos equipamentos e identificar a tensão elétrica de cada equipamento cadastrado.

Observando o usuário Cesar abrir a tela para o monitoramento, foi possível perceber que mesmo sem nunca ter acessado esta tela, ele conseguiu localizá-la e identificar a tensão elétrica de cada equipamento cadastrado.

A Figura 89 apresenta o momento em que o usuário Cesar estava monitorando a tensão elétrica dos equipamentos cadastrados.



equipamento	on-line	voltage
ID: 5, nome: RB 43, IP: 192.168.20.43	Sim	11.7 (atualizado em 08/11/2012 01:50:26)
ID: 8, nome: RB Torre volta cachimbo, IP: 192.168.20.48	Sim	13.4 (atualizado em 08/11/2012 01:50:27)
ID: 6, nome: RB 20, IP: 192.168.20.20	Sim	12 (atualizado em 08/11/2012 01:50:27)
ID: 3, nome: RB Horatário, IP: 192.168.20.3	Sim	13.1 (atualizado em 08/11/2012 01:50:26)
ID: 4, nome: Torre Maurino, IP: 192.168.20.4	Sim	13.6 (atualizado em 08/11/2012 01:50:25)
ID: 2, nome: RB TORRE PINHERAL, IP: 192.168.20.21	Sim	12.2 (atualizado em 08/11/2012 01:50:26)

Figura 89 – Monitoramento da tensão elétrica das *RouterBoards* no provedor de acesso

Para a tela de monitoramento foi sugerido que a tela atualizasse automaticamente a cada 30 segundos. Esta sugestão também se encontra na seção 4.1.

Após o monitoramento foi solicitado ao usuário Cesar abrir a tela de visualização dos alertas e identificar os alertas gerados pelo sistema.

Observando o usuário Cesar realizar o que foi solicitado, foi possível perceber que mesmo sem nunca ter acessado esta tela, ele conseguiu localizá-la e associar os alarmes com a regra cadastrada, porém não foi possível associar o alerta ao equipamento. Por isso foi sugerido adicionar o nome do equipamento na mensagem de erro ou criar uma coluna na *grid* para o equipamento.

A Figura 90 apresenta o momento em que o usuário Cesar estava monitorando os alertas gerados pelo sistema com um filtro para o dia 08/11/2012.

ID	Timestamp	Description	Details
274	08/11/2012 00:49:	Erro processando mensagem	Erro ao executar mensagem Era esperado "]"
275	08/11/2012 00:57:	Erro processando mensagem	Erro ao executar mensagem Era esperado "]"
276	08/11/2012 01:19:	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 11.7
277	08/11/2012 01:19:	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 12
278	08/11/2012 01:20:	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 11.7
279	08/11/2012 01:20:	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 12
280	08/11/2012 01:21:	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 11.7
281	08/11/2012 01:21:	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 12
282	08/11/2012 01:22:	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 11.7
283	08/11/2012 01:22:	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 12
284	08/11/2012 01:23:	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 11.7
285	08/11/2012 01:23:	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 12
286	08/11/2012 01:24:	Parâmetro com valor inaceitável	Valor inválido. Parâmetro voltage. Encontrado 11.7

Figura 90 - Alertas gerados no monitoramento do parâmetro *voltage*

No provedor de acesso não foi possível realizar testes com a mensagem *interface wireless monitor* devido a versão das RouterBoards serem mais antigas do que as utilizadas nos testes no momento da implementação. Ao enviar as mensagens para as RouterBoards cada versão respondia de forma diferente. Cesar informou que o provedor de acesso está se organizando para atualizar as versões das RouterBoards.

3.5 RESULTADOS E DISCUSSÃO

Os resultados obtidos no desenvolvimento deste sistema são apresentados nesta seção. Os requisitos levantados foram todos atendidos, conforme apresentado na Tabela 3.

Requisito (descrição resumida)	Atendido
Permitir o cadastro de equipamentos.	Sim
Permitir o cadastro dos responsáveis pelo monitoramento.	Sim
Permitir que o usuário informe o valor mínimo aceitável de algumas propriedades monitoradas nos equipamentos modelo RB433AH.	Sim
Monitorar os equipamentos modelo RB433AH sobre a tensão da bateria e manter um histórico.	Sim
Monitorar os equipamentos modelo RB433AH sobre a qualidade do sinal enviada para os clientes e manter um histórico.	Sim
Enviar <i>e-mail</i> notificando os responsáveis pelo monitoramento, cujas propriedades monitoradas em um equipamento estejam fora dos valores aceitáveis.	Sim
Permitir que o usuário visualize no navegador a informação mais atual, descritas nos itens anteriores.	Sim
Permitir a configuração de quanto em quanto tempo deve ser realizado o monitoramento.	Sim
Implementar o sistema web utilizando tecnologia <i>Java Enterprise Edition</i> (JEE) e o Flex.	Sim
Ser compatível com os navegadores Google Chrome, Internet Explorer e Firefox.	Sim

Tabela 3 – *Checklist* dos Requisitos

Na proposta deste trabalho levantou-se o requisito sobre configuração de quanto em quanto tempo deve ser realizado o monitoramento definindo que o tempo de intervalo deveria ser por mensagem, mas ao implementar percebeu-se que não era possível, pois poderia ocorrer do sistema tentar enviar duas mensagens para o mesmo equipamento ao mesmo tempo.

Durante os testes foram envolvidos cinco usuários e distribuído um formulário contendo tarefas que o usuário deveria realizar no sistema (primeira parte do Apêndice A). Estas tarefas tinham o objetivo de direcionar o usuário no teste. Ao final, foi apresentado um formulário com algumas questões em relação ao uso do sistema (segunda parte do Apêndice A).

Os usuários envolvidos nos testes responderam as perguntas do questionário. Em cada questão, os usuários assinalaram o grau de avaliação da ferramenta em termos de "Concordo

totalmente", "Concordo parcialmente", "Discordo parcialmente" e "Discordo totalmente". A partir das respostas obtidas na avaliação foi formulado a Tabela 4.

Perguntas / Critérios de avaliação	Concordo totalmente	Concordo parcialmente	Discordo parcialmente	Discordo totalmente
1. É fácil localizar a tela para cadastrar novo usuário	100%			
2. É fácil localizar a tela para cadastrar novo equipamento	100%			
3. É fácil localizar a tela para cadastrar nova regra	100%			
4. É fácil localizar a tela para visualizar o status dos equipamentos	100%			
5. É fácil localizar a tela para visualizar os alertas gerados	100%			
6. É fácil cadastrar novo usuário	100%			
7. É fácil cadastrar um novo equipamento	100%			
8. É fácil cadastrar nova regra	80%	20%		
9. É fácil localizar a tensão da bateria de um equipamento na tela de status dos equipamentos	100%			
10. É fácil localizar a qualidade de conexão de um equipamento na tela de status dos equipamentos	100%			
11. É fácil de identificar o motivo de um alerta gerado pelo sistema na tela de alertas gerados		100%		

Tabela 4 - Tabela de avaliação do sistema pelos usuários

A partir dos resultados das questões 1 à 5 da Tabela 4 é possível perceber que os usuários não tiveram dificuldade para localizar as telas relacionadas com cada tarefa.

Com relação às respostas das questões 6 e 7 da Tabela 4 é possível perceber que nenhum usuário teve problemas em relação ao cadastro de usuário e de grupo de usuário e todos concordam que é fácil realizar um novo cadastro de usuário ou de grupo de usuário.

Com relação às respostas da questão 7 da Tabela 4 é possível perceber que praticamente todos os usuários concordam que é fácil cadastrar um nova regra. Conforme mencionado na seção 3.4.2, durante os testes no provedor de acesso surgiram dúvidas de como utilizar a tela de cadastro de regra. Lucimara também teve dúvidas ao utilizar a tela, que tiveram que ser sanadas para continuar o teste. Mas em um segundo momento ambos não tiveram dúvidas em como utilizar a tela de manutenção de regras.

Com relação às respostas das questões 9 e 10 da Tabela 4 é possível perceber que nenhum usuário teve dúvidas em relação a tela de *status* dos equipamentos.

Com relação às respostas das questões 11 do da Tabela 4 é possível perceber que todos os usuários, acharam fácil identificar o motivo de um alerta gerado pelo sistema, mas que existem algumas melhorias que podem ser feitas, principalmente quando o erro é de *parser*.

A Tabela 5 apresenta uma comparação entre os trabalhos correlatos e o sistema desenvolvido neste trabalho. A partir dessa tabela é possível perceber as características comuns e as principais diferenças da ferramenta desenvolvida.

Funcionalidade/Característica	Este trabalho	Winbox	Webfig	The Dude
Monitoramento da tensão elétrica	X	X	X	X
Monitoramento da qualidade do sinal	X	X	X	X
Ambiente WEB	X		X	
Monitoramento de vários roteadores em uma mesma tela	X			X
Envio de e-mail para responsáveis pelo monitoramento do equipamento quando tensão da bateria estiver abaixo de um valor estipulado	X			
Envio de e-mail para responsáveis pelo monitoramento do equipamento quando a qualidade do sinal para clientes estiver abaixo de um valor estipulado.	X			
Cadastro de responsáveis por equipamento	X			
Tela de visualização de alertas gerados	X			
Monitora tensão da bateria e a qualidade do sinal de qualquer versão de RouterBoard que de suporte		X	X	X
O usuário não precisa manipular scripts do sistema para monitorar tensão elétrica e a qualidade do sinal	X	X	X	

Tabela 5 – Comparação entre os trabalhos correlatos e o sistema desenvolvido neste trabalho

Apesar do The Dude exigir que o usuário faça manipulação de *scripts*, pode-se concluir que todos possuem monitoramento da tensão elétrica e da qualidade do sinal.

Pode-se observar na Tabela 5 que apenas este trabalho possui a tela de visualização de alertas, o cadastro de responsáveis e o envio de e-mail para os responsáveis sobre a tensão da bateria ou a qualidade do sinal estiverem abaixo de um valor estipulado.

Pode-se concluir também que apenas este trabalho e o Webfig possuem uma característica muito importante que é ser um ambiente WEB.

Uma característica deste trabalho, que foi comentada na seção de testes do provedor de acesso e que deve ser melhorada é que o sistema está limitado para monitorar algumas versões de RouterBoard, o que não acontece com os trabalhos correlatos. Essa melhoria foi adicionada nas sugestões de extensões deste trabalho.

Outra característica muito importante é o monitoramento de vários equipamentos em apenas uma tela. Isto é possível apenas com este trabalho e com a ferramenta The Dude, porém a configuração no The Dude é muito mais complicada conforme apresentada na seção 2.8.3.

4 CONCLUSÕES

O desenvolvimento do sistema foi voltado para o monitoramento da RouterBoard, com objetivo de automatizar o processo de monitoramento das propriedades `voltage` e `overall-tx-cq`, encontradas, respectivamente, nas mensagens *system health print* e *interface wireless monitor* e criar uma tela amigável para o monitoramento.

Os objetivos principais deste trabalho foram alcançados. Foi construído um sistema web que possui uma interface amigável para o monitoramento, permitindo monitorar mais de um roteador por vez. O servidor possui um processo que é responsável por estabelecer comunicação com os roteadores cadastrados e monitorá-los, mantendo o *status* atual em memória e gerando alarmes quando identifica que o valor de algum parâmetro é inválido. O servidor também possui outro processo responsável por buscar os alarmes que foram gerados por causa de um parâmetro inválido e enviá-los para as pessoas responsáveis pelo monitoramento.

Os requisitos mencionados na seção 13.1 foram todos atendidos. O requisito descrito no item h da seção 13.1 precisou ser alterado durante a implementação pois percebeu-se que seu detalhamento estava incorreto, conforme já mencionado na seção 3.3.14.

Apesar deste trabalho não contribuir com nenhuma inovação na área de gerência de redes, ele produziu um sistema de monitoramento que pode ser utilizado em provedores de acesso a internet via rádio que utilizem RouterBoard. Além disso, o sistema possui capacidade de crescimento, não só para o monitoramento da RouterBoard, mas também para qualquer equipamento que suporte o acesso pelo terminal virtual telnet.

Os testes em campo apresentados na seção 3.4.2, mostraram que o sistema facilita o monitoramento dos roteadores da Mikrotik. Mas também mostrou que para o bom funcionamento do sistema seria necessário utilizar todas as RouterBoards com o mesmo modelo e atualizar o RouterOS de cada RouterBoard. O usuário Cesar reconheceu que o monitoramento de seus equipamentos com o sistema desenvolvido neste trabalho é mais rápido do que o monitoramento realizado com os trabalhos correlatos, ele não precisa fazer *login* em cada RouterBoard e consegue visualizar o status de todos os equipamentos em apenas um tela.

O framework Adobe Flex contribuiu para a criação das telas do sistema de maneira rápida e produtiva.

4.1 EXTENSÕES

As sugestões de extensões deste trabalho estão listadas a seguir:

- a) atualização automática da tela de monitoramento a cada trinta segundos
- b) atualização automática da tela de alertas a cada trinta segundos;
- c) emitir som na tela de monitoramento quando algum valor não estiver de acordo com as regras;
- d) construir um gráfico de linha baseado no status dos dispositivos, sobre algum parâmetro que possui valor numérico;
- e) alterar definição do GALS para possibilitar o monitoramento com novas mensagens;
- f) criar uma tela para envio de comandos para a RouterBoard;
- g) analisar a possibilidade de alterar a especificação do GALS e gerar suas classes em tempo de execução, utilizando as regras cadastradas pelo usuário;
- h) implementar um cadastro de permissão por tela para cada grupo de usuários. Um exemplo seria um grupo de usuário chamado monitor, só ter permissão para abrir a tela de monitoramento e visualizar os alarmes gerados pelo sistema;
- i) utilizar criptografia para salvar a senha do usuário no banco de dados;
- j) adicionar um campo endereço no equipamento e utilizar a API do Google Maps para Flex para mostrar no mapa onde o equipamento está localizado;
- k) melhorar o filtro das informações na tela de visualização de alarmes, adicionando filtro por hora e por equipamento;
- l) adicionar filtro das informações nas telas de cadastros;
- m) fazer o cadastro das mensagens ser independente da especificação do GALS (evitando de ter que alterar a especificação do GALS manualmente e gerar as classes do GALS novamente);
- n) trocar a forma de comunicação entre o sistema e os roteadores para o protocolo de comunicação SNMP;
- o) criar sistema de *logs* em arquivo para mensagens de erros, alertas, informações, etc.;
- p) não permitir o usuário alterar o texto dos campos de seleção (ComboBox).

REFERÊNCIAS BIBLIOGRÁFICAS

ADOBE. **Adobe**: download free trial version Adobe Flash Builder 4.6. [S.l.], 2012a. Disponível em: <https://www.adobe.com/cfusion/tdrc/index.cfm?product=flash_builder>. Acesso em: 23 out. 2012.

_____. **BlazeDS**. [S.l.], 2012b. Disponível em: <<http://sourceforge.net/adobe/blazeds/wiki/Home/>>. Acesso em: 23 out. 2012.

_____. **Download Adobe Flex SDK**. [S.l.], 2012c. Disponível em: <<http://www.adobe.com/devnet/flex/flex-sdk-download.html>>. Acesso em: 23 out. 2012.

_____. **Flex accessibility**. [S.l.], 2012d. Disponível em: <<http://www.adobe.com/accessibility/products/flex/>>. Acesso em: 23 out. 2012.

_____. **What is Flex?** [S.l.], 2012e. Disponível em: <<http://www.adobe.com/products/flex.html>>. Acesso em: 23 out. 2012.

APACHE. **Apache Tomcat**. [S.l.], 2012. Disponível em: <<http://tomcat.apache.org/download-70.cgi>>. Acesso em: 23 out. 2012.

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. Rio de Janeiro: Campus, 2002.

BUSCH, Marianne; KOCH, Nora. **Rich internet applications: state-of-the-art**. München, 2009. Disponível em: <http://uwe.pst.ifi.lmu.de/publications/maewa_rias_report.pdf>. Acesso em: 31 out. 2012.

CABRAL, Túlio E. **Provedor completo com Mikrotik**: tudo o que você precisa. Salvador, 2007. Disponível em: <<http://www.youblisher.com/p/148764-Manual-Mikrotik>>. Acesso em: 05 abr. 2012.

CABUS, Shankar; VIDAL, Henrique. **Object pool**. [S.l.], 2012. Disponível em: <http://www.csi.uneb.br/padroes_de_projetos/object_pool.html>. Acesso em: 3 nov. 2012.

DEUS, Jackson E. **O que é Mikrotik RouterOS?** [S.l.], 2011. Disponível em: <<http://jacksonezidio.blogspot.com.br/2011/02/o-que-e-mikrotik-routeros.html>>. Acesso em: 07 abr. 2012.

ELROM, Elad; SCHULZE, Charlie; TIWARI, Shashank. **Flex 4: avançado**. São Paulo: Novatec, 2011.

GESSER, Carlos E. **Gals**: gerador de analisadores léxicos e sintáticos. Florianópolis, 2003. Disponível em: <<http://pt.scribd.com/doc/45414071/Gals>>. Acesso em: 29 out. 2012.

GODÓI, Fábio. **O padrão facade aplicado**. [S.l.], 2009. Disponível em: <<http://www.devmedia.com.br/o-padrao-facade-aplicado/12683>>. Acesso em: 29 out. 2012.

IMASTERS. **Wireless com Mikrotik - parte 07 - configurando RouterBoard e Mikrotik como roteador wireless**. [S.l.], 2009. Disponível em: <http://imasters.com.br/artigo/13242/reds/wireless_com_mikrotik_-_configurando_routerboard_e_mikrotik_como_roteador_wireless>. Acesso em: 07 abr. 2012.

JUGEL, Matthias L.; MEISSNER, Marcus. **JTA: Telnet/SSH for the JAVA(tm) platform**. [S.l.], 2009. Disponível em: <<http://javatelnets.org/space/start>>. Acesso em: 3 nov. 2012.

KUNG, Fabio; et al. **Introdução à arquitetura e design de software: uma visão sobre a plataforma Java**. Rio de Janeiro: Elsevier, 2012. 257 p.

LUCA, Vicente. **Implantação e gerenciamento de uma rede sem fio nos domínios de um campus universitário**. Lavras, 2010. Disponível em: <www.bcc.ufla.br/monografias/2010/VICENTE_DE_LUCA.pdf>. Acesso em: 07 abr. 2012.

MICHIGAN. **Introdução ao Mikrotik**. [S.l.], 2011. Disponível em: <<http://www.michigan.com.br/downloads/mikrotik/Introducao-Mikrotik.pdf>>. Acesso em: 07 abr. 2012.

MIKROTIK. **Winbox**. [S.l.], 2004. Disponível em: <<http://www.mikrotik.com/testdocs/ros/2.9/guide/winbox.php>>. Acesso em: 07 abr. 2012.

_____. **Mikrotik RouterOS**. [S.l.], 2010. Disponível em: <http://www.mikrotik.com/pdf/what_is_routeros.pdf>. Acesso em: 07 abr. 2012.

_____. **About us**. [S.l.], 2012a. Disponível em: <<http://routerboard.com/about/>>. Acesso em: 06 abr. 2012.

_____. **Routerboard 433ah**. [S.l.], 2011. Disponível em: <<http://routerboard.com/pdf/73/rb433ah.pdf>>. Acesso em: 04 nov. 2012.

_____. **Manual: The Dude/functions**. [S.l.], 2012b. Disponível em: <http://wiki.mikrotik.com/wiki/Manual:The_Dude/Functions>. Acesso em: 15 abr. 2012.

_____. **Manual: The Dude/probes**. [S.l.], 2012c. Disponível em: <http://wiki.mikrotik.com/wiki/Manual:The_Dude/Probes>. Acesso em: 15 abr. 2012.

_____. **Manual: Webfig**. [S.l.], 2012d. Disponível em: <<http://wiki.mikrotik.com/wiki/Manual:Webfig>>. Acesso em: 15 abr. 2012.

_____. **MikroTik routers and wireless**. [S.l.], 2012e. Disponível em: <<http://www.mikrotik.com/thedude.php>>. acesso em: 03 nov. 2012.

NEGREIROS, Waelson. **Flex: trabalhando com janelas MDI com FlexMDI**. [S.l.], 2010. Disponível em: <<http://waelson.com.br/2010/08/03/rapidinhaflex-trabalhando-com-janelas-mdi-com-mdiflex/>>. Acesso em: 03 out. 2012.

NOVANETWORK. **RouterBoard Mikrotik RB433AH**. [S.l.], 2012. Disponível em: <<http://www.novanetwork.com.br/produtos/det/rb433ah/8/mikrotik/placas-routerboard/routerboard-mikrotik/routerboard-mikrotik-rb433ah.php>>. Acesso em: 08 abr. 2012.

ORACLE. **Java SE development kit 7 downloads**. [S.l.], 2012. Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_gerenciamento_de_redes_de_computadores.php>. Acesso em: 10 out. 2012.

PEDRO, Bruno P. **Como montar um painel solar e ligar um Mikrotik no sistema: energia solar para Mikrotik**. [S.l.], 2011. Disponível em: <<http://consultoriawireless.blogspot.com.br/2011/07/como-montar-um-painel-solar-e-ligar-um.html>>. Acesso em: 07 abr. 2012.

PFLEEGER, Shari L. **Engenharia de software: teoria e prática**. 2. ed. São Paulo: Prentice Hall, 2004. 535 p.

PINHEIRO, José M. S. **Gerenciamento de redes de computadores: uma breve introdução**. [S.l.], 2006. Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_gerenciamento_de_redes_de_computadores.php>. Acesso em: 07 abr. 2012.

POSTGRESQL. **PostgreSQL: downloads**. [S.l.], 2012. Disponível em: <<http://www.postgresql.org/download/>>. Acesso em: 23 out. 2012.

PRICE, Ana M. A.; TOSCANI, Simão S. **Implementação de linguagens de programação: compiladores**. 2. ed. Porto Alegre: Sagra Luzzatto, 2001. 195 p.

ROCHA, Frederico C.; PAIVA, Giselle S. **Prática de roteamento utilizando Routerboard Mikrotik**. Vila Velha, jun. 2011. Disponível em: <http://www.adjutojunior.com.br/tcc/Pratica_de_Roteamento_Utilizando_RouterBoard_Mikrotik_Frederico_Chacara_Rocha_Giselle_Salvador_de_Paiva.pdf>. Acesso em: 08 abr. 2012.

SANTOS, Juraci A. **Integrando Flash Builder 4 com o Eclipse**. Goiânia, 2012. Disponível em: <http://www.gojava.org/files/artigos/java_flex.pdf>. Acesso em: 23 out. 2012.

SCHMITZ, Daniel. **10 coisas que um bom programador Flex deve saber**. [S.l.], 2011. Disponível em: <<http://www.flex.etc.br/traducoes/10-coisas-que-um-bom-programador-flex-deve-saber/>>. Acesso em: 03 out. 2012.

TACLA, Cesar A. **Design patterns parte 3: padrão MVC**. Curitiba, 2010. Disponível em: <<http://www.dainf.ct.utfpr.edu.br/~tacla/DesignPatterns/0030-JavaDP-MVC.pdf>>. Acesso em: 3 nov. 2012.

TANENBAUM, Andrew S. **Redes de computadores**. 4. ed. Rio de Janeiro: Campus, 2003. 968 p.

TORRES, Gabriel. **Como o protocolo TCP/IP funciona**: parte 1. [S.l.], 2007. Disponível em: <<http://www.clubedohardware.com.br/artigos/1351>>. Acesso em: 04 nov. 2012.

VALIM, Rafael F. **Mudando a porta do Tomcat**. [S.l.], 2011. Disponível em: <<http://www.alltroniks.com.br/en/2011/05/mudando-a-porta-do-tomcat/>>. Acesso em: 04 nov. 2012.

APÊNDICE A – Questionário de avaliação

Data de realização da avaliação: / /2012.

PRIMEIRA PARTE:

A primeira parte do questionário descreve os procedimentos que o usuário deve fazer para avaliar o sistema.

Etapa 1: Realize os procedimentos conforme descritos a seguir:

1. acesse o sistema, será apresentada a tela inicial do sistema;
2. abrir a tela de manutenção de usuários e cadastrar um novo usuário;
3. abrir a tela de manutenção de equipamentos e cadastrar uma RouterBoard modelo RB433AH;
4. abrir a tela de manutenção de regras e cadastrar uma nova regra para o parâmetro *voltage* da mensagem *system health print* informando que o valor deve ser acima de 11 Volts;
5. abrir a tela de status dos equipamentos e identificar a tensão elétrica de cada equipamento cadastrado;
6. abrir a tela de visualização dos alertas e identificar os alertas gerados pelo sistema;

SEGUNDA PARTE:

A segunda etapa é um questionário de avaliação do sistema.

As respostas deverão ser feitas na tabela abaixo. Você deve responder preenchendo uma das alternativas.

Perguntas / Critérios de avaliação	Concordo totalmente	Concordo parcialmente	Discordo parcialmente	Discordo totalmente
1. É fácil localizar a tela para cadastrar novo usuário				
2. É fácil localizar a tela para cadastrar novo equipamento				
3. É fácil localizar a tela para cadastrar nova regra				
4. É fácil localizar a tela para visualizar o status dos equipamentos				
5. É fácil localizar a tela para visualizar os alertas gerados				
6. É fácil cadastrar novo usuário				
7. É fácil cadastrar uma novo equipamento				
8. É fácil cadastrar nova regra				
9. É fácil localizar a tensão da bateria de um equipamento na tela de status dos equipamentos				
10. É fácil localizar a qualidade de conexão de um equipamento na tela de status dos equipamentos				
11. É fácil de identificar o motivo de um alerta				

Qual é a sua opinião sobre cada tela quanto ao seu uso e funcionalidades?
(Utilize o verso da folha para responder)

Obrigado pela sua participação!