ANÁLISE E GERENCIAMENTO REMOTO DE CLIENTES E SERVIDORES EM UMA REDE COMPUTACIONAL

Trabalho apresentado à
Banca examinadora do curso de
Engenharia da Computação da
FAET – Faculdade de Ciências Exatas e Tecnologia do
UNICEUB – Centro Universitário de Brasília
Como requisito parcial para obtenção do título de
Engenheiro da Computação

Aluno: Henrique Rodrigues da Silva - R.A.: 2001588/8
Orientador: MSc. Fabiano Mariath D`Oliveira

Brasília – DF, Junho de 2005

Agradecimentos

A Deus, que me concede forças e coragem para executá-lo.

Aos meus pais, pelo apoio.

Aos meus irmãos, pelo carinho.

A minha namorada, que esteve sempre ao meu lado me dando força e apoio.

Aos amigos de faculdade, por todos os momentos que passamos juntos.

Aos meus amigos de serviço, pelo incentivo nos momentos mais difíceis.

Ao meu orientador MSc. Fabiano Mariath D`Oliveira, que esteve rente comigo.

Ao coordenador do curso de Engenharia da Computação, MSc. Abiezer Amarília.

Ao MSc. Francisco Javier Obaldia, por sempre me apoiar e incentivar.

Aos demais professores do Curso de Engenharia da Computação, por todo o conhecimento e amizade dispensados a mim.

Aos meus grandes amigos em especial: Mateus Silva, Wolmer Andrade, Luciano Caixeta, Cezário Neto, Júlio César, Fernanda Sakamoto e Adriano Delfino.

A todos, que de certa forma, contribuíram para a realização deste projeto.

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

ÍNDICE DE SIGLAS E ABREVIATURAS

API	Application Program Interface
ASP	Active Server Pages
CIM	Common Information Model
CIMOM	CIM Object Manager
CMIS/CMIP	Common Management Information Service/Protocol
DCOM	Distributed Component Object Model
DHTML	Dynamic HTML
DMTF	Distributed Management Task Force
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Security
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
ITIL	IT Infrastructure Library
JScript	Java Script
JCP	Java Community Process 2.0
JSR	Java Specification Request
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MIB	Management Information Base
MOF	Managed Object Format
NDS	Novell Directory Service
ODBC	Open Database Connectivity
OSI	Open Systems Interconnection
RAM	Random Access Memory
SNMP	Simple Management Network Protocol
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
UCB	Universidade Católica de Brasília
UML	Unified Model Language

UnB	Universidade de Brasília
UniCEUB	Centro Universitário de Brasília
VB	Visual Basic
VBA	Visual Basic Application
VBScript	Visual Basic Script
WAN	Wide Area Network
WBEM	Web Based Enterprise Management
WQL	WMI Query Language
WSH	Windows Script Host
WMI	Windows Management Instrumentation
XML	Extensible Markup Language

RESUMO

Este trabalho apresenta o estudo de tecnologias que proporcionaram a criação de uma ferramenta de análise e gerenciamento de uma rede, remotamente, possibilitando verificar e alterar configurações existentes, além da execução de tarefas administrativas que facilitarão o dia-a-dia do administrador.

A ferramenta foi desenvolvida utilizando o *Windows Management Instrumentation* (WMI) da Microsoft, que é baseado no padrão *Web Based Enterprise Management* (WBEM) da *Distributed Management Task Force* (DMTF), além de utilizar *VBScript* e a linguagem de programação *VB .NET*, ambos da Microsoft.

Palavras-Chave: gerência de rede; administração remota; gerenciamento de tarefas; WBEM, WMI; VBScript; inventário de hardware e software.

ABSTRACT

This work shows the study of technologies that provided the designing of analyses and management remote network, make capable verify and change active configurations, beyond the execution of administrator tasks that make easier the administrator routine.

The tool have been designed using Windows Management Instrumentation (WMI) from Microsoft, which is based in Web Based Enterprise Management (WBEM) from Distributed Management Task Force (DMTF), beyond use VBScript and the programming language VB .NET, both from Microsoft.

Keywords: network management; remote administration; task management; WBEM; WMI; VBScript, hardware and software inventory.

Capítulo 1 - Introdução

1.1 MOTIVAÇÃO

Com a proliferação de computadores do tipo *desktop*, laptops e dispositivos portáteis, o custo real de manutenção de uma rede de computadores pessoais, devido ao esforço administrativo gerado, tem aumentado significativamente. Esse custo inclui todo o capital utilizado na aquisição de hardware e software, nas despesas de infra-estrutura, na capacitação de pessoas, na manutenção de rotina, nos custos de atualização de hardware e software, e nos gastos com suporte técnico [PEREIRA, 2001].

No intuito de obter uma noção clara do mercado nacional e verificar suas necessidades, foi realizada uma entrevista através de e-mail com 15 administradores de redes dos mais diferenciados ambientes possíveis (ver apêndice B), envolvendo redes de 25 computadores até 30000 computadores aproximadamente. Essa pesquisa foi realizada no segundo semestre de 2004.

Das 15 entrevistas realizadas foram obtidas 11 respostas, das quais é possível concluir que:

- A média de computadores por empresa é de 1758, excluindo a de menor número deles (25) e a de maior número (30000), obtendo assim, um dado mais preciso da média de computadores por empresa;

- 55% dos entrevistados possuem alguma ferramenta que faz inventário de hardware e software. 36% dos entrevistados não possuem ferramentas desse tipo e 9% dos entrevistados não souberam informar a respeito;

- 100% dos entrevistados consideraram importante ter o inventário de hardware e software do seu parque computacional atualizado;

- são gastos em média 25 minutos por computador para realizar o inventário de hardware e software dos computadores de sua rede;

- 100% dos entrevistados consideraram necessário ter uma ferramenta que faça o inventário de hardware e software de forma remota e automatizada em seu parque computacional;

- 100% dos entrevistados acreditam que, com a existência de uma ferramenta de inventário de hardware e software seria reduzido o tempo gasto na execução dessas tarefas em seu parque computacional;

- 100% dos entrevistados acreditam que uma ferramenta de inventário de hardware e software possa disponibilizar informações que sirvam como suporte para novos investimentos no parque computacional;

- 82% entrevistados consideraram importante ter uma ferramenta que possibilite desligar, reiniciar e/ou executar logoff em computadores remotos, enquanto 18% consideraram isso pouco relevante;

- 82% entrevistados acreditam que, com a existência de uma ferramenta que possibilite desligar, reiniciar e/ou executar logoff em computadores remotos, poderia reduzir o tempo gasto na execução dessas tarefas, enquanto 18% dos entrevistados acreditam não ser importante;

- 100% dos entrevistados consideram importante ter uma ferramenta que possibilite listar, parar ou iniciar serviços em computadores remotos;

- 100% dos entrevistados acreditam que, com a existência de uma ferramenta que possibilite listar, parar ou iniciar serviços em computadores remotos, poderia reduzir o tempo gasto na execução dessas tarefas.

Com os resultados obtidos nessa entrevista é possível verificar a necessidade de uma ferramenta de gerência de rede que contemple as tarefas de administração questionadas na mesma.

Como contribuição para agilizar e facilitar as tarefas executadas pelos administradores, surgiu a motivação de estudar o assunto de gerência de redes e projetar uma ferramenta que auxilie a execução dessas tarefas administrativas de forma eficaz e remota.

1.2 OBJETIVOS

O trabalho desenvolvido tem como objetivo projetar uma ferramenta de gerência que atenda às necessidades básicas de inventário de computadores em rede e disponibilize informações para os administradores que sirvam como suporte para novos investimentos no parque computacional, provendo informações dos hardwares de maneira prática e segura.

Os objetivos específicos do trabalho são:

a) obter dados físicos de um computador para fins de inventário de hardware em máquina local ou remota conectada à rede, especificando:

- Modelo;
- Processador;
- Memória RAM;
- Discos Existentes;
- Placas Existentes.

b) desligar e/ou reiniciar computador(es) conectados à rede;

c) pesquisar softwares existentes nos computadores clientes, para fins de inventário de software;

d) visualizar os serviços disponíveis e seus estados no sistema operacional de um computador local ou remoto, podendo parar ou iniciar um serviço desejado;

1.3 METODOLOGIA DA PESQUISA

Foi realizada uma entrevista de campo com administradores de redes, a qual serviu como motivação para a realização desse projeto. Com base nos questionamentos abordados foram traçados os objetivos desse projeto.

Para o desenvolvimento do projeto foi realizada uma pesquisa bibliográfica para obtenção do material teórico. Após a pesquisa bibliográfica e a realização da pesquisa em campo, deu-se início à monografia e ao planejamento de criação e desenvolvimento do protótipo de uma aplicação de gerência de rede que será utilizada para validar os objetivos aqui propostos.

Terminada a fase de criação da ferramenta, iniciou-se a fase de testes, utilizando um laboratório do UniCEUB e o ambiente computacional do Tribunal Superior Eleitoral - TSE, que possibilitaram a execução dos testes e ajustes necessários para obter-se o funcionamento desejado da ferramenta.

1.4 ESTRUTURA DA MONOGRAFIA

Este trabalho está disposto em capítulos, dos quais será apresentada uma breve descrição a seguir:

- O *Capítulo 2 – Referencial Teórico* traz diversos conceitos na área de gerência de redes, que elucida pontos chaves que serão de grande relevância para o bom entendimento do trabalho como um todo;

- O *Capítulo 3 – HRNet Gerenciamento* mostra a ferramenta criada durante o projeto, validando o estudo feito, demonstrando o procedimento utilizado para a criação, os testes executados, suas características e funcionalidades, os problemas percorridos e as soluções abordadas;

- Finalmente, no *Capítulo 4 – Conclusões* são feitas as considerações finais do projeto e as sugestões para trabalhos futuros.

Capítulo 2 – Referencial Teórico

Atualmente as redes locais de computadores têm crescido significativamente e conseqüentemente o esforço administrativo gerado se tornou proporcionalmente elevado [PEREIRA, 2001]. Este crescimento vem impulsionando o surgimento de diversas tecnologias para facilitar o trabalho exercido pelos administradores de rede e proporcionar um bom nível de satisfação dos usuários, aliado a um ambiente seguro. Dentre estas tecnologias surgem padrões de gerenciamento de redes e sistemas, as quais serão descritas a seguir.

2.1 TECNOLOGIAS PARA GERÊNCIA DE REDE

Existem hoje no mercado diversos sistemas de gerência corporativos convencionais conhecidos como *frameworks*. Esses *frameworks* são baseados em padrões abertos e utilizados em redes heterogêneas, provendo suporte aos diversos tipos de *hardware* e *software* existentes. Nos últimos anos, os *frameworks* passaram por grandes evoluções e amadurecimentos. Esse período foi marcado pela criação de diferentes tipos de funcionalidades e gerenciamento [CARVILHE, 2000].

Um dos padrões recomendados pela *Internet Engineering Task Force* (IETF) para a implementação de gerência de redes utilizando TCP/IP é o padrão *Simple Network Management Protocol* (SNMP). Da mesma forma, os *frameworks* de gerência de redes corporativa da *International Organization for Standardization* (ISO), com o protocolo *Common Management Information Service/Protocol* (CMIS/CMIP) têm sido utilizados nas redes OSI e nas grandes companhias de telecomunicações na última década. Com o crescimento da WEB reativou-se a iniciativa e o interesse nos *frameworks* pela *Distributed Management Task Force* (DMTF) para gerência de redes e sistemas [CARVILHE, 2000].

Com o surgimento de todos esses *frameworks*, a atuação sobre a gerência de redes de dados e telecomunicações tem crescido rapidamente. Adicionalmente, consórcios e fóruns internacionais estão desenvolvendo novos *frameworks* promovendo assim, o aumento da abrangência da gerência corporativa de redes computacionais [CARVILHE, 2000].

Tentando resolver esses problemas múltiplos agentes, protocolos, formatos de dados, e sistemas passaram a ser desenvolvidos e utilizados no mercado. Um dos protocolos mais conhecidos é o SNMP. Esse protocolo funciona com um esquema de comunicação “agente – servidor”, que pode ser configurado para relatar eventos chaves ocorridos nos agentes monitorados.

O gerenciamento do SNMP é feito através do servidor utilizando um *software* especial. Estes servidores possuem processos que se comunicam com os agentes emitindo comandos e obtendo repostas. Pode-se dizer que toda a inteligência fica nos servidores de gerenciamento [TANENBAUM, 1997].

Também foram criados diversos serviços de diretório para facilitar a administração da rede. O primeiro fabricante a elaborar um esquema de serviço de diretório foi a Novell com o *Novell Directory Service* (NDS). Do NDS partiu-se para o desenvolvimento de um padrão, o *Lightweight Directory Access Protocol* (LDAP), adotado em seguida pela Microsoft com o *Active Directory* e pela Oracle com o LDAP Oracle, e posteriormente pela própria Novell, melhorando o NDS [MICROSOFT, 2004] [NOVELL, 2000] [ORACLE, 2001].

Por fim, surgiu o *Web-Based Enterprise Management* (WBEM), um *framework* de gerência de redes baseada na WEB que cresceu rapidamente e conseguiu englobar diversas outras tecnologias de gerência de redes através de um único esquema de comunicação “agente – servidor”. A seguir será apresentada a tecnologia WBEM.

2.2 WBEM

Em 1996 foi criado o *framework* conhecido como *Web-Based Enterprise Management* (WBEM) e em 1998, passou a ser responsabilidade da *Distributed Management Task Force* (DMTF) [DMTF, 1999]. O WBEM surgiu através do interesse em gerência de redes por grandes empresas como: IBM, Cisco Systems, Compaq/HP, Intel, BMC Software e Microsoft [SACRAMENTO, 2003]. Essas empresas partiram do princípio que necessitariam de uma padronização de um esquema de gerência corporativa que trabalhasse com os novos padrões de mercado e os convencionais *frameworks* de gerência de redes.

O WBEM foi se desenvolvendo com a DMTF, e surgiu o *Common Information Model* (CIM), uma coleção de esquemas orientados a objetos para gerenciamento de informação, que pode ser considerada como uma das partes mais importantes do WBEM [SACRAMENTO, 2003]. O CIM também é padronizado pela DMTF e possibilita acessar o repositório de dados e ser gerenciado tanto local como remotamente [DMTF,1999].

O CIM é um modelo de dados que pretende representar de forma normalizada os mais diversos componentes de um sistema de informática, seguindo o conceito de um modelo orientado a objetos. Ele permite que outros modelos de informação sejam incluídos em sua estrutura de dados como por exemplo: *Simple Network Management Protocol* (SNMP) e *Common Management Information Protocol* (CMIP), criando assim uma extensão do esquema do CIM. [CARVILHE, 2000] [SACRAMENTO, 2003].

Outro elemento da arquitetura WBEM é o *Common Information Model Object Manager* (CIMOM), que possui uma linguagem própria para definição dos objetos gerenciados denominada de *Managed Object Format* (MOF) [DMTF, 1999]. O CIMOM pode ser chamado de gerente de objetos, pois é o responsável pelo controle dos objetos gerenciados, incluindo o armazenamento e o acesso aos dados no repositório central de dados [CARVILHE, 2000].

O WBEM possui um componente muito importante no processo utilizado em seu fluxo de dados, denominado provedores. Eles correspondem a um conjunto de processos que são responsáveis em fazer a comunicação entre os agentes de gerência de sistemas convencionais e os dispositivos gerenciados, obtendo assim, os valores dos objetos gerenciados [CARVILHE, 2000].

Segundo [CARVILHE, 2000] o *framework* WBEM define os seguintes provedores:

a) **provedores de propriedades**: retornam os valores de propriedade de objetos solicitados através de uma chave de identificação. Retorna informações em uma única instância;

b) **provedores de instância**: retornam valores de instância de objetos. Retornam informações da instância em uma única classe;

c) **provedores de classe**: retornam classes e instâncias. Controlam todas as classes;

d) **provedores de *namespace***: são capazes de gerenciar um espaço de nome CIMOM pré-definido. Controlam todos os espaços de nomes e instâncias de objetos.

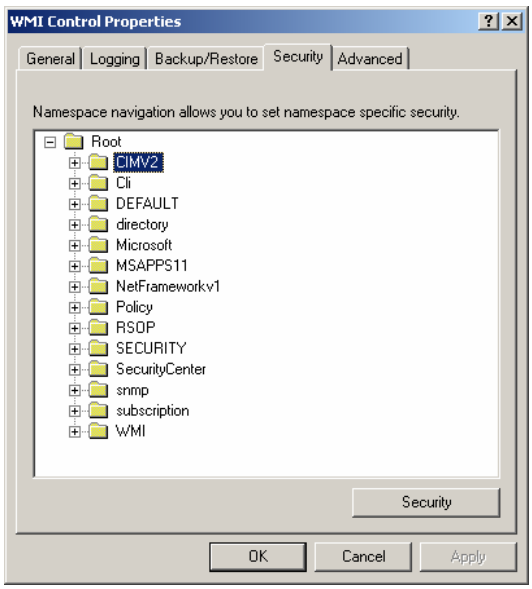

Figura 2.1 – Exemplo de Espaço de Nomes (*namespace*) disponíveis

Na Figura 2.1 é demonstrado os *namespace* disponíveis nos quais as aplicações de gerência podem se conectar para obter acesso às propriedades referentes aos objetos existentes em cada *namespace.* Os administradores podem configurar o nível de segurança que os clientes podem ter em cada *namespace* determinado [CASTRO, 2002].

A seguir será explicada a arquitetura do padrão WBEM.

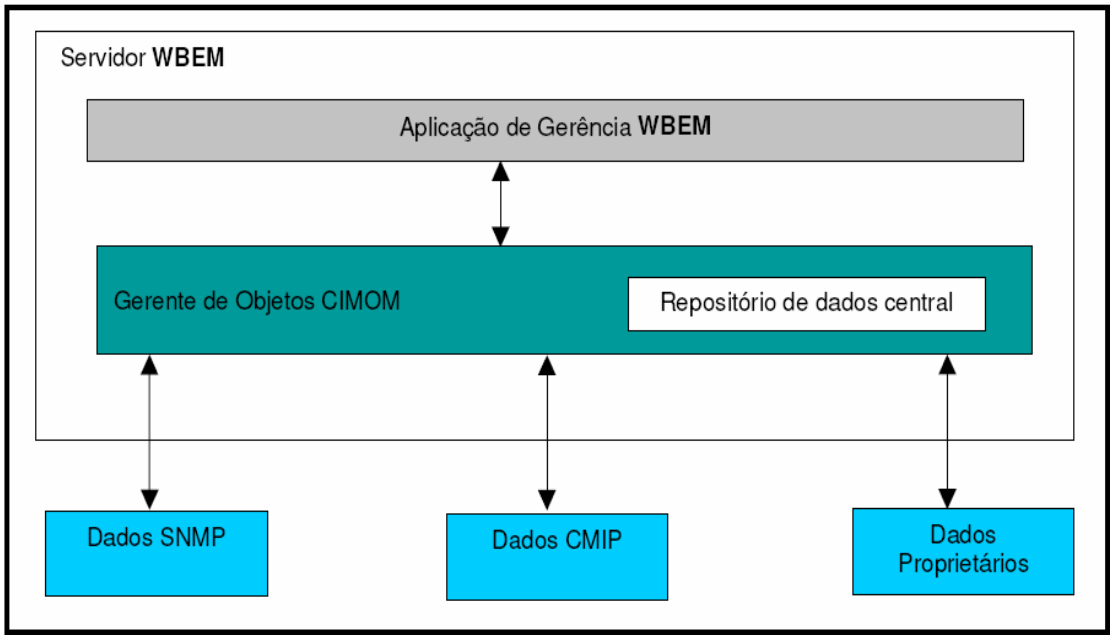

Figura 2.2 – Modelo do fluxo de dados do padrão WBEM [BRAZ, 2003, p.21].

Na Figura 2.2, pode-se observar como funciona o fluxo de dados no padrão WBEM. A aplicação de gerência, entra em contato com o CIMOM, o qual faz uma solicitação de informação sobre um determinado objeto gerenciado.

O CIMOM exercendo a sua função de gerenciador dos objetos, consulta ou atualiza a informação desejada no CIM. Caso essa informação seja alterada no objeto gerenciado, o provedor (SMNP, CMIP ou outros) é responsável por entregar ao CIMOM a informação atualizada, a qual é transformada em um formato padrão, ou seja, a linguagem MOF, para assim, serem armazenados no repositório de dados (CIM). Apenas os dados estáticos são armazenados no CIM, enquanto que os dinâmicos são consultados pelo CIMOM diretamente aos provedores.

O CIMOM pode aceitar todos os objetos dos *frameworks* de gerência de sistemas e de redes existentes, porém, antes de poderem ser utilizadas, assim como numa solicitação de uma aplicação, ele transforma todas as informações dos objetos gerenciados para sua própria linguagem e só então, armazena no repositório de dados (CIM). Além disso, o CIMOM é o grande responsável em prover as informações solicitadas pela aplicação de gerência [CASTRO, 2002].

Baseado no modelo de informação comum (CIM), o WBEM é um *framework* que estabelece padrões de infra-estrutura de gerência de rede, além de prover um meio padronizado de acesso a informações de vários hardwares e softwares através de um único sistema de gerência. Com a utilização dessa tecnologia é possível criar ferramentas para a redução da complexidade da gerência de redes e dos custos da empresa, contribuindo assim, para a redução do Custo Total de Propriedade (TCO)[1] [DMTF, 1999].

A grande inovação dessas tecnologias de gerência de redes é a possibilidade de realizar as tarefas de gerenciamento de qualquer ponto na rede, podendo até mesmo ser executada em diversas máquinas ao mesmo tempo. Esta inovação é um facilitador para a vida do administrador de rede, que deixa de perder tempo ao se locomover até a estação ou servidor desejado para realizar uma simples tarefa que pode ser executada remotamente [CARVILHE, 2000].

2.3 IMPLEMENTAÇÕES DO WBEM

De acordo com [BRAZ, 2003] [DMTF, 1999] existem diversas implementações do WBEM. A seguir será feito um breve comentário sobre algumas delas, a saber: *WMI*, *WBEMServices*, *OpenPegasus* e *OpenWBEM.*

[1] TCO refere-se ao custo administrativo associado com aquisição de hardware e software, implementação e configuração, atualizações de hardware e software, treinamento, manutenção e suporte técnico [MICROSOFT, 2000].

WMI: implementação proposta pela Microsoft, inclui um repositório de dados compatível com o CIM e o gerenciador de objetos CIMOM [MICROSOFT, 1999]. A comunicação entre uma aplicação gerente e o CIMOM é realizada utilizando a arquitetura de objetos distribuídos (DCOM) da Microsoft. No item 2.4 será detalhada a implementação WMI [BRAZ, 2003].

WBEMServices: implementação Java do WBEM em código aberto, seja para aplicações comerciais ou não-comerciais. O projeto consiste de API, aplicações clientes, aplicações servidoras, além de algumas ferramentas. As API são baseadas na *Java Specification Request* (JSR) e submetidas à aprovação da *Java Community Process* (JCP) 2.0. [WBEMSERVICES, 2004].

Possui ferramentas como o CIMOM, repositório de dados com alguns arquivos MOF já adicionados, compilador MOFCOMP, entre outras e suporte aos protocolos HTTP e RMI para transferência de informação. Dentre as desvantagens destaca-se a necessidade de implementação dos provedores, que suprem os dados da base de objetos [BRAZ, 2003] [CASTRO, 2002].

OpenPegasus: implementação em C++ do WBEM desenvolvido em código aberto pelo *OpenGroup.* Abrange todos os componentes do WBEM. Portável para diversas plataformas, entre elas, *Microsoft Windows*, *Linux* e a maioria das versões *UNIX*. [OPENPEGASUS, 2004].

OpenWBEM: implementação em código aberto do WBEM desenvolvida em C++ e com extensões em JAVA, comercial ou não-comercial. Realiza operações do CIM sobre HTTP ou HTTPS, possui diversos módulos de autenticação, entre eles: *SIMPLE, PAM e Digest Authentication.* Possui um compilador MOF e uma biblioteca WQL para realizar buscas no repositório de dados. [BRAZ, 2003] [OPENWBEM, 2004].

Quadro 2.1 – Comparativo de implementações do WBEM. Adaptado de [CHOI, 2004].

	WMI	OpenPegasus	WBEMServices	OpenWBEM
Licenciamento	Microsoft	MIT open source license	SISSL v1.2	BSD Style License
Desenvolvedores	Microsoft & EMC	OpenGroup	Sun Microsystems Inc	Caldera International Inc
Plataforma	Windows 98, NT, ME, 2000, XP, 2003	Linux, AIX, HPUX, Windows	Indiferente	Linux, Caldera Open Sun Solaris
Documentação	Excelente	Boa	Boa	Ruim
Implementação	Excelente	Muito Boa	Muito Boa	Muito Boa
Fornecedor	Microsoft	SBLIM	Inside API	SBLIM

No quadro 2.1 é apresentada a comparação entre alguns exemplos de implementações do WBEM. No próximo item desse capítulo será detalhada a implementação da Microsoft, o *Windows Management Instrumentation* (WMI).

2.4 WINDOWS MANAGEMENT INTRUMENTATION – WMI

O WMI é um componente do sistema operacional Microsoft Windows, implementado e desenvolvido pela Microsoft baseado na padronização do WBEM. Ele surgiu para ajudar no desafio encontrado pelos administradores em gerenciar as redes e sistemas de suas empresas. Através dele é possível com uma simples consulta ao repositório de dados, obter e filtrar dados requeridos, podendo esses estarem locais ou remotos [MSDN TRAINING, 2001].

O WMI também oferece uma variedade de interfaces de programação como C/C++, ODBC, Microsoft Visual Basic, HTML entre outros que podem ser usadas por desenvolvedores para customizar aplicações de gerência [MICROSOFT, 2005].

Essa tecnologia é uma infra-estrutura de gerência de redes que suporta a sintaxe do modelo de dados CIM, possui uma interface de programação comum, além do gerenciador de objetos CIMOM [MICROSOFT, 1999]. Esses serviços e os dados de gerência são acessados através da interface de programação *Common Object Model* (COM) ou através de scripts [JONES, 2004].

Os administradores da rede podem utilizar-se do WMI através de scripts para automatizar as tarefas administrativas. O WMI pode se integrar com componentes do Windows como o serviço de diretório *Active Directory*, permitindo uma experiência de gerência de redes a níveis complexos de forma simples [MSDN TRAINING, 2001].

Como exemplo, através da WMI é possível estabelecer padrões para acessar e compartilhar informações de gerenciamento e notificações de eventos de uma rede, podendo controlar e monitorar os componentes do sistema sejam softwares ou hardwares [CASTRO, 2002].

2.4.1 Arquitetura WMI

A arquitetura WMI de acordo com a Microsoft [MICROSOFT, 2000] provê:

a) acessar e monitorar, comandar e controlar qualquer objeto gerenciável através de uma interface comum;

b) ter um modelo consistente de status e configuração de um sistema operacional Windows;

c) utilizar a COM *Application Programming Interface* (API) que fornece um único ponto de acesso a toda informação gerenciável;

d) capacitar os serviços de gerência de dois computadores diferentes se comunicarem. Esta aproximação pode facilitar bastante à criação de processos integrados para soluções de gerência;

e) possibilitar a flexibilidade e a expansão, podendo assim estender modelos de informações para suportar novos dispositivos e aplicações, apenas escrevendo novos módulos de códigos chamados WMI *providers*;

f) obter uma poderosa arquitetura de eventos que permite identificar as informações de gerência, podendo agregar, comparar ou associar com outras informações de gerência, tanto locais como remotas;

g) possuir uma rica linguagem de pesquisa que permite fazer consultas detalhadas nos modelos de informação;

h) ter uma API na qual pode-se criar aplicações gerenciáveis utilizando scripts escritos em diversas linguagens como VBScript e JScript.

A arquitetura da tecnologia do WMI consiste em [MICROSOFT, 2000]:

a) **aplicações de gerenciamento**: são aplicações que requisitam e processam informações de objetos gerenciados armazenados no repositório de dados. Estes dados são acessados através de uma requisição ao CIMOM utilizando métodos descritos na API do WMI;

b) **infra-estrutura de gerenciamento**: inclui o CIMOM e o repositório de dados. O CIMOM é responsável pela ligação entre as aplicações de gerenciamento e os provedores de dados;

c) **objetos gerenciados**: são os objetos físicos ou lógicos modelados através do CIM, que podem ser gerenciados. Por exemplo, um objeto

pode ser tanto um hardware, como um cabo, quanto um software, como uma aplicação de banco de dados;

d) **aplicações gerenciadas**: são as aplicações ou serviços do Windows que usa ou processa as informações originadas dos objetos gerenciáveis. As aplicações gerenciáveis podem acessar as informações dos objetos de gerência fazendo uma requisição ao CIMOM através de um dos métodos da WMI API;

e) **provedores WMI**: atuam como mediadores entre o CIMOM e os objetos gerenciados. Quando o CIMOM recebe uma solicitação de uma aplicação de gerenciamento para um dado que não está presente no repositório ou de um evento que não é suportado, este encaminha a requisição para um provedor WMI. Provedores suprem dados e notificações de eventos dos objetos gerenciados específicos de seu domínio.

Alguns provedores *(providers)* disponíveis [MASTON, 1999]:

a) ***Active Directory Provider***: Atua como uma porta de comunicação para todas as informações armazenadas no serviço *Active Directory*. Permite acessar, utilizando apenas uma API, informações tanto do WMI como do *Active Directory*;

b) ***Windows Installer Provider***: Permite o controle completo das instalações dos softwares através do *Windows Installer* utilizando WMI. Além de prover informações de qualquer software instalado através desta tecnologia.

c) ***Performance Counter Provider***: Utilizado para acessar o contador de desempenho usado para computar os valores mostrados da ferramenta do monitor do sistema. Qualquer contador de performance instalado do sistema operacional será visível através deste *provider*;

d) ***Registry Provider***: Permite criar, ler e escrever nas chaves de registro do Windows. Eventos do WMI podem ser gerados quando chaves específicas do registro forem alteradas;

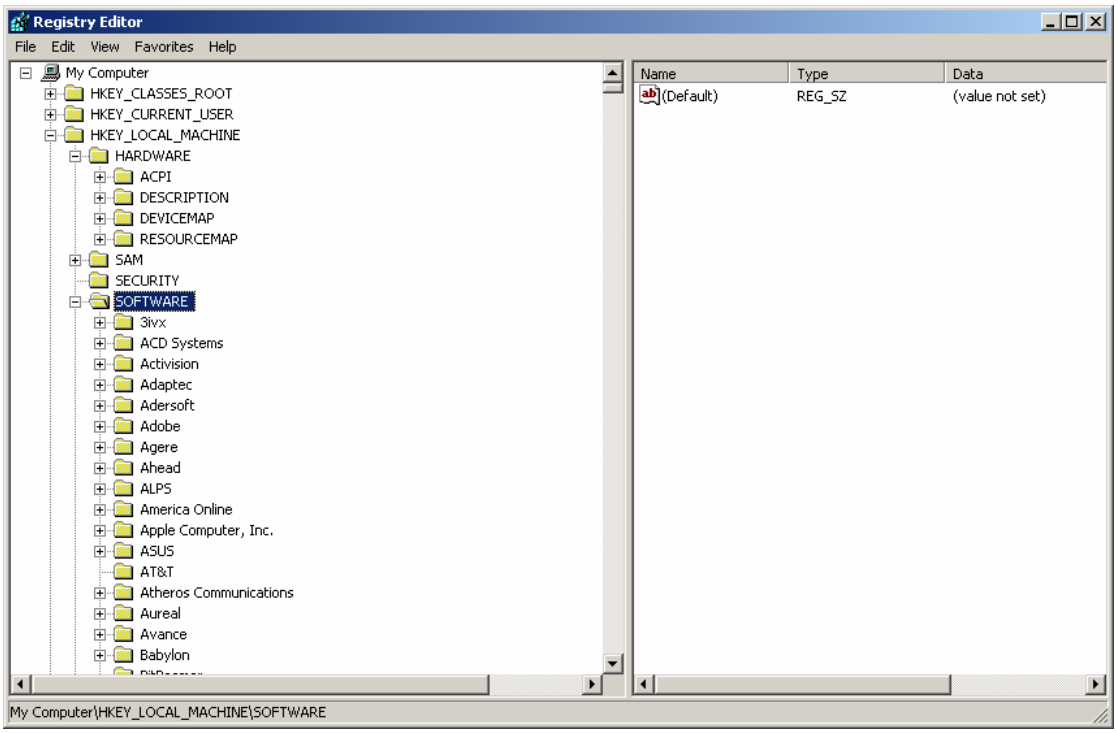

Figura 2.3 – Tela do Registro do Windows

Na Figura 2.3 é possível ver um exemplo de atuação de um *provider*. Utilizando o *Registry Provider* do WMI é possível fazer leitura, alteração ou criação de chaves no registro do Windows.

e) ***SNMP Provider***: Atua como uma porta de comunicação para usar o SNMP para gerenciamento dos dispositivos e sistemas. As variáveis de objetos SNMP MIB podem ser lidas e escritas. SNMP pode ser automaticamente mapeado para os eventos do WMI;

f) ***Event Log Provider***: Fornece acesso aos dados e notificações de eventos do log de eventos do Windows;

g) ***Win32 Provider***: Fornece informações sobre o sistema operacional, sistema computadorizado, dispositivos periféricos, sistema de arquivos e informações de segurança;

h) ***WDM Provider***: Fornece informações de baixo nível do modelo de drivers do Windows (WDM) para dispositivos de entrada do usuário, dispositivos de armazenamento, interfaces de rede, e portas de comunicação;

i) ***View Provider***: Permite agregar ou construir novas classes em cima das classes já existentes, fazer filtros para obter apenas as informações de interesse, combinar múltiplas classes em uma única classe ou ainda, agregar dados de várias máquinas em uma única visão.

A tecnologia WMI também dá suporte a *providers* customizados criados por terceiros, os quais podem ser usados para requisição de serviços relacionados ao gerenciamento de objetos de ambientes específicos [MICROSOFT, 2000].

O processo executável que provê todas as funcionalidades do WMI chama-se WinMgmt.exe. O serviço WMI (WinMgmt.exe) é responsável por manipular solicitações dos clientes, trocar informações com os provedores e gerenciar o repositório de dados [MARTINSSON, 2002].

Este serviço é exposto aos programadores por intermédio de duas API, sendo que, para linguagem C/C++ é utilizada a API COM, que é o acesso de mais baixo nível ao WMI que se pode solicitar. Já, para linguagens de scripts que reconhecem automação, é utilizada a API Scripting, que é um empacotador (*wrapper*) em torno da API COM [MARTINSSON, 2002].

2.4.2 WMI QUERY LANGUAGE

A *WMI Query Language* (WQL) é a linguagem utilizada para fazer consultas ao WMI. Através dessas consultas é possível levantar dados de diversos tipos no ambiente gerenciado, como também explorá-los e utilizá-los conforme a necessidade.

A WQL é um subconjunto do SQL (*Structured Query Language*, linguagem de consultas estruturada). Diferentemente do SQL tradicional, a WQL é uma linguagem de consultas que não foi concebida para atualizar, eliminar ou inserir dados [MICROSOFT, 2005a].

A WQL, segundo [MARTINSSON, 2002], suporta três tipos de consultas:

a) **consultas de dados**: usada pelas aplicações de gerência que precisam selecionar associações de dados e instâncias sobre instâncias;

b) **consultas de evento**: usada pelas aplicações de gerência que implementam a manipulação de eventos. Os eventos são tratados dentro da aplicação de gerência, a qual recebe notificações de alterações como a criação, a eliminação ou a modificação de um banco de dados. Essas consultas registram na aplicação de gerência notificações de eventos ocorridos;

c) **consultas de esquema**: similar às consultas de dados, porém estas retornam meta-dados ao invés de instâncias. Por exemplo: consultar os processos que estejam consumindo mais memória do que um valor determinado.

```
strComputer = inputbox("Nome", "Computador analisado")
if strComputer = "" then
      strComputer = "."
end if
Set objWMIService = GetObject("winmgmts:" & _
"{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")
Set Disco_C = objWMIService.ExecQuery _
   ("Select * from Win32_LogicalDisk Where DeviceID = 'C:'")
For Each objDisco in Disco_C
   Wscript.Echo "O espaço livre na partição C: é " & objDisco.FreeSpace
Next
```

Figura 2.4 – Exemplo de código WQL

Na Figura 2.4 está representado o código para verificação do espaço disponível na partição “C:” de um determinado computador, podendo esse, ser local ou remoto. Na execução do mesmo, será apresentada ao usuário uma janela na qual ele informará o computador desejado e receberá então a notificação da informação solicitada, podendo assim, utilizar a informação para o fim desejado.

2.5 XML

O XML é considerado uma linguagem de grande evolução na internet. Essa linguagem é uma especificação técnica desenvolvida pela *World Wide Web Consortium* (W3C), entidade responsável pela definição da área gráfica da internet, para superar as limitações do HTML, que é o padrão das páginas da Internet [XML, 2004].

A linguagem XML é baseada em texto e definida como o formato universal para dados estruturados. Esses dados consistem em tabelas, desenhos, parâmetros de configuração, etc. A linguagem então, define regras que permitem escrever esses documentos de forma que sejam adequadamente visíveis ao computador [XML, 2004].

```
<?xml version="1.0" encoding="iso-8859-15" ?>
<Resultado>
    <Computador>
        <Nome>NTBHENRIQUE</Nome>
        <Fabricante>Hewlett-Packard</Fabricante>
        <Modelo>HP Pavilion dv1000 (PM054UA#ABA)</Modelo>
        <Processadores>Intel(R) Pentium(R) M processor 1.60GHz</Processadores>
        <MemoriaFisica>478 MB</MemoriaFisica>
    </Computador>
</Resultado>
```

Figura 2.5 – Exemplo de arquivo XML

Na Figura 2.5 é representada a estrutura de um arquivo XML.

Capítulo 3 - HRNet Gerenciamento

3.1 INTRODUÇÃO

A administração de um ambiente lógico e físico corporativo tornou-se uma tarefa complexa nos ambientes de redes atuais. Essa complexidade exige dos administradores de rede, alta capacidade de absorção das novas tecnologias em um curto período de tempo. A Microsoft, objetivando simplificar esse processo, tentou através do WMI englobar uma gama de agentes de diversos fabricantes em uma única interface de gerenciamento mais amigável [MSDN TRAINING, 2001].

Para demonstrar o funcionamento dessas tecnologias, foi desenvolvida uma aplicação de gerência remota, utilizando a implementação do WBEM proposta pela Microsoft, o WMI. Vale ressaltar que o protótipo da aplicação não tem o intuito de ser uma aplicação de gerência de rede completa e comercial, mas sim, uma aplicação que valide a possibilidade de utilização dos padrões de gerência aqui apresentados, bem como os objetivos propostos no projeto.

A aplicação de gerência que foi desenvolvida para validar o projeto proposto chama-se HRNet Gerenciamento. No decorrer desse capítulo serão apresentadas informações relativas a aplicações de mercado criadas utilizando a tecnologia WMI, os fatores que levaram a escolha da tecnologia WMI, os requisitos necessários para o projeto, os procedimentos utilizados durante a implementação e os testes realizados.

3.2 FERRAMENTAS DE MERCADO DESENVOLVIDAS A PARTIR DA TECNOLOGIA WMI

Para a criação da ferramenta proposta nesse projeto, foi feito um trabalho de pesquisa para levantar as características e funcionalidades de ferramentas existentes no mercado que utilizam a tecnologia WMI.

Essa pesquisa serviu para verificar as principais necessidades dos ambientes de redes atuais, podendo assim, nortear quais funcionalidades teriam maior proveito no dia-a-dia de um administrador de rede.

Quadro 3.1 – Comparativo de ferramentas do mercado que utilizam tecnologia WMI

Características	Sandra 2005 Enterprise	MonitAPP	Network Inventory Manager 3.0.11.5	Domain Administration Tool 3.2	Clarus Evolution 2.0	Everest Corporate Edition	Hyena 6.3
Inventário de Hardware	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Inventário de Software	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Desligar e reiniciar computadores remotos	Não	Não	Não	Sim	Sim	Não	Sim
Listar, parar e iniciar serviços	Parcial	Parcial	Não	Não	Sim	Parcial	Sim
Gerenciamento de contas de usuário	Não	Não	Não	Sim	Não	Não	Sim
Preço (em US$)	80,00	8,95	195,00	99,00	30,00	29,95	269,00

Com os dados levantados das ferramentas de mercado, juntamente com as necessidades expressas pelos administradores de rede entrevistados, foram determinadas as funcionalidades básicas da ferramenta, citadas abaixo:

- Inventário de Hardware;
- Inventário de Software;
- Desligar e reiniciar computadores remotos;
- Listar, parar e iniciar serviços.

Em todas as entrevistas realizadas foi observada a necessidade da execução das funcionalidades remotamente e em grandes escalas, pois a execução das mesmas, localmente, ou mesmo em poucas máquinas, não justifica a necessidade de se ter uma ferramenta específica para tal uso.

3.3 FATORES QUE LEVARAM A ESCOLHA DA TECNOLOGIA WMI

Como visto no capítulo 2, apesar de todas as funcionalidades já agregadas ao SNMP, ele não atende a todas as necessidades de gerência de redes. Similarmente, o padrão *Desktop Management Interface* (DMI) foi projetado para

trabalhar somente com os dados relativos a hardware, através de protocolos desenvolvidos pelos fabricantes, também não solucionando todas as necessidades de gerência de redes.

Outras tecnologias projetadas para facilitar as tarefas de gerenciamento de redes foram as ferramentas de serviço de diretório, que assim como o SNMP e DMI não conseguem, de forma isolada, operar toda a gerência de uma rede.

A grande vantagem do WMI é sua atuação como tradutor, escondendo a complexidade de tecnologias múltiplas (DMI, SNMP, CMIP, *Windows Driver Model* (WDM), Microsoft Win32), que utilizam o acesso uniforme a todos os dados de sistema e de hardware suportados, através de uma interface comum.

O acesso aos dados de gerência utilizando WMI, dispensa conhecimentos específicos em diferentes tipos de interfaces, protocolos e tipos de dados utilizados, o que facilita em muito o desenvolvimento de uma ferramenta de gerência, que possa recolher essas informações e apresentá-las de uma forma simples para os administradores de rede.

A seguir serão apresentadas algumas vantagens na utilização do WMI:

- Coleção de tecnologias baseada no padrão internacional WBEM;
- Ampla documentação, tanto de material teórico como prático (scripts) que podem ser encontrados na Internet;
- Fácil implementação, possui suporte a VB, VBA, VBScript, JScript, ASP, DHTML, etc.;
- Modular, utiliza implementação orientada a objeto;
- Altamente integrada e utilizada com todos os produtos Microsoft;
- Redução no tempo e aumento na qualidade de execução das tarefas administrativas;
- Diminuição do esforço administrativo gerado ao executar as tarefas cotidianas;
- Provê suporte ao acesso de dados remoto, sem a necessidade de criar uma aplicação que faça a conexão remota;

- Esquema de dados escalar e consistente, permitindo que novos tipos de informações gerenciáveis, criadas por terceiros, possam ser adicionadas ao modelo de dados do WMI de forma consistente.

3.4 REQUISITOS OBSERVADOS

Para a execução da aplicação de gerência, de forma satisfatória, de acordo com o estudo sobre WMI, alguns requisitos são necessários.

Abaixo estão listados os principais requisitos funcionais e não funcionais da aplicação de gerência:

- Rodar em sistema operacional Windows Me/2000/XP/2003 (WMI nativo) ou Windows 95/98/NT4 com WMI (Core) 1.5 e WSH 5.6 instalados e estar com o serviço WMI iniciado;
- Ter permissão de administrador nas máquinas remotas para obter todas as informações desejadas através do acesso aos dados ao WMI;
- Rodar em computadores com fonte ATX;
- Coletar informações das classes WMI em computadores remotos;
- Realizar um inventário básico de hardware e software, no intuito de demonstrar as funcionalidades da ferramenta;
- Obter acesso ao controlador de domínio da rede, com a finalidade de pesquisar os computadores existentes na rede através do *provider Active Directory*;
- Executar a verificação dos computadores disponíveis no momento da execução da ferramenta;
- Coletar informações nas classes do WMI referente aos serviços nas máquinas clientes e seus estados, podendo esses, serem alterados;
- Não possuir qualquer aplicativo (*firewall, antispyware)* que possa bloquear a execução dos scripts em VBScript.
- Ter o Microsoft *framework* .NET instalado no computador que possui a aplicação de gerência (não é necessário o *framework* .NET nos computadores clientes).

3.5 FLUXO DE FUNCIONAMENTO DA APLICAÇÃO DE GERÊNCIA

Na Figura 3.1 será apresentada uma visão geral do fluxo de funcionamento da aplicação de gerência HRNet.

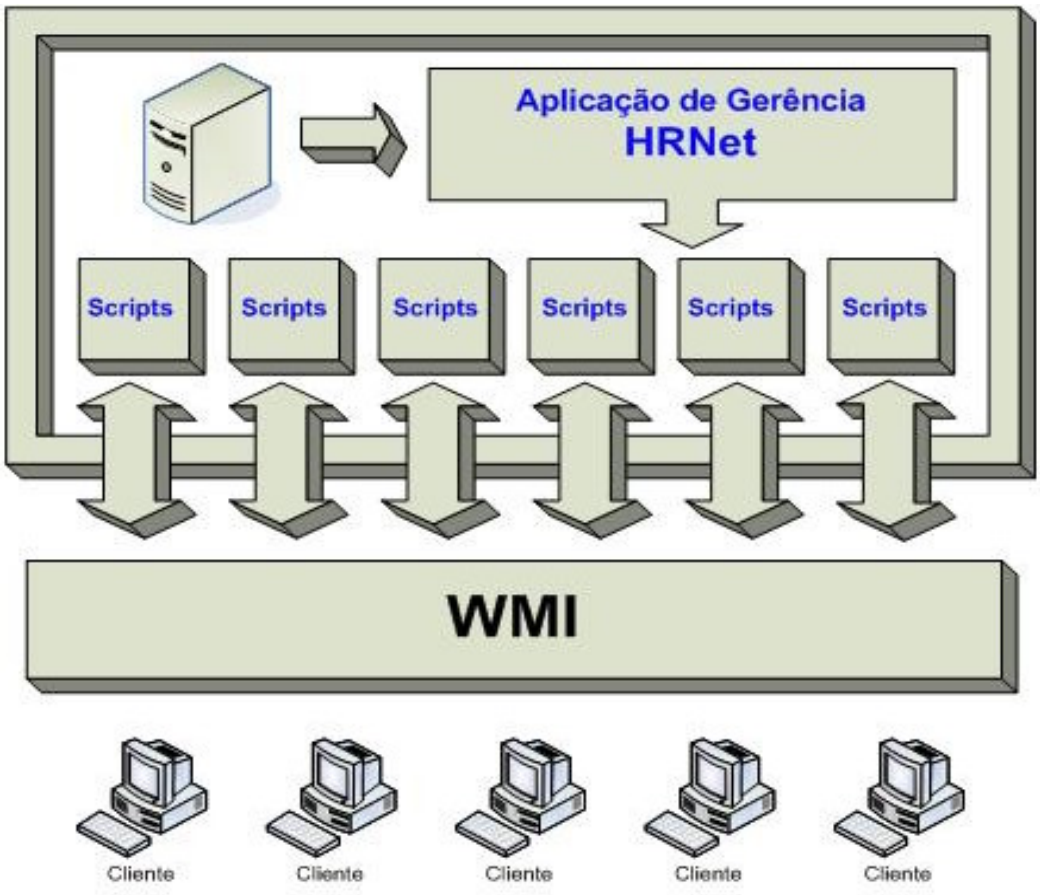

Figura 3.1 – Fluxo do funcionamento da aplicação de gerência HRNet

Através da interface gráfica do HRNet Gerenciamento são feitas chamadas para a execução dos scripts. Eles são responsáveis por dispararem consultas ao WMI, o qual está localizado na máquina destino. O WMI por sua vez, retorna os dados da pesquisa para serem devidamente tratados na aplicação de gerência.

3.5.1 Fluxo Geral do Funcionamento

Os passos abaixo se referem ao processo macro de funcionamento da ferramenta HRNet Gerenciamento.

1º passo – A aplicação de gerência HRNet Gerenciamento faz uma chamada a um Script VBS que tem a função de comunicar com o serviço de diretório do Windows através do *provider Active Directory* do WMI.

2º passo – O Script VBS através de uma consulta (*query)* do WMI (WQL) faz acesso às bases de dados do serviço de diretórios do Windows e retorna os nomes de todos os computadores existentes na rede. Esse mesmo script faz ainda uma verificação, utilizando um comando do próprio sistema operacional denominado *PING,* em todos os computadores da rede, verificando assim as máquinas que estão ligadas, e gera um arquivo XML apenas com os nomes das máquinas disponíveis no momento.

3º passo – O arquivo XML com o nome dos computadores disponíveis é gravado na máquina que está sendo executada a aplicação de gerência para ser lido e apresentado ao administrador através da aplicação HRNet. O arquivo XML funciona para aplicação como se fosse um banco de dados das informações consultadas no WMI local ou retomo.

4º passo – A aplicação HRNet lista todos os computadores possíveis de acesso através do arquivo gerado em XML e disponibiliza as funcionalidades que podem ser executadas nas máquinas listadas.

5º passo – O administrador seleciona o(s) computador(es) desejado(s) e clica na função desejada (ex.: Inventário).

6º passo – A aplicação de gerência novamente fará uma chamada ao Script VBS responsável pela função requisitada.

7º passo – O Script VBS por sua vez executará os procedimentos necessários de acordo com a função desejada e retornará as informações requisitadas para um novo arquivo no formato XML.

8º passo – O arquivo no formato XML novamente será armazenado na máquina que está executando a aplicação de gerência.

9º passo – A aplicação de gerência, HRNet Gerenciamento, lê os dados contidos no arquivo XML e apresenta, de forma amigável, ao usuário da aplicação.

3.5.2 Fluxo das Funcionalidades

Na Figura 3.2 será apresentado o diagrama de caso de uso, que exemplifica a interação do usuário com a aplicação de gerência. As funcionalidades existentes foram criadas, dada a necessidade dos administradores de rede e das análises das comparações de funcionalidades estabelecidas entre as ferramentas de mercado no subitem 3.2 desse mesmo capítulo.

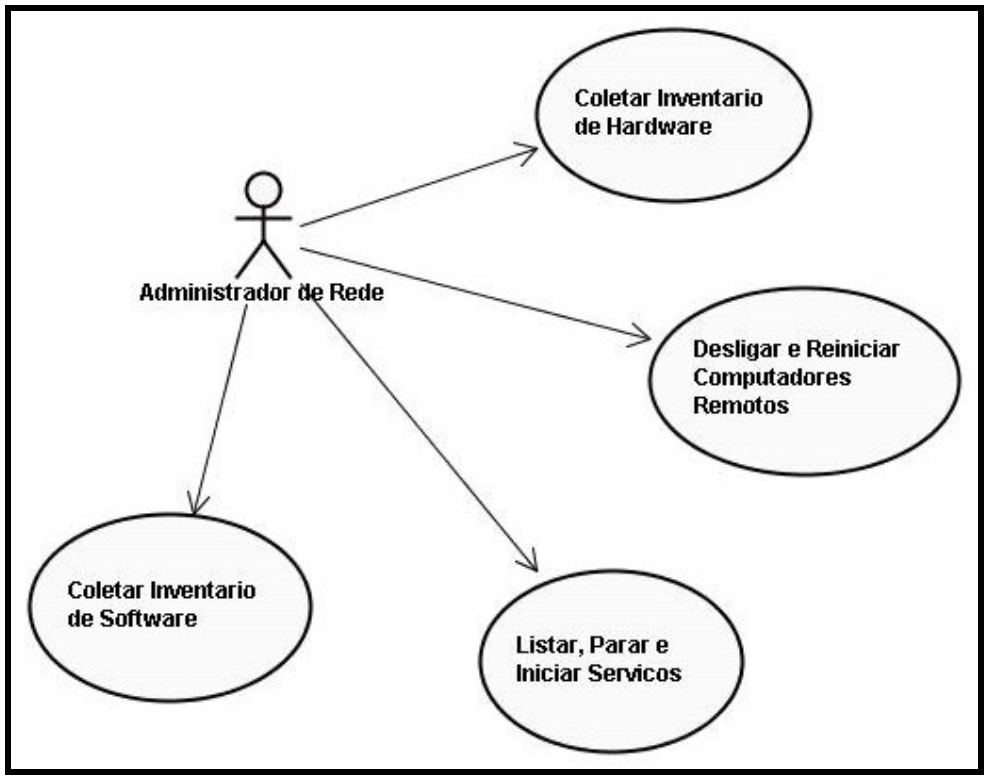

Figura 3.2 – Diagrama de Caso de Uso[2]

[2] Diagrama de caso de uso é um elemento da UML (Unified Modeling Language) que exibe as interações entre atores e casos de uso, ou seja, as funcionalidades do sistema. A UML é uma linguagem para especificação, documentação, visualização e desenvolvimento de sistemas orientados a objetos. Sintetiza os principais métodos existentes, sendo considerada uma das linguagens mais expressivas para modelagem de sistemas orientados a objetos. Por meio de seus diagramas é possível representar sistemas de softwares sob diversas perspectivas de visualização. Facilita a comunicação de todas as pessoas envolvidas no processo de desenvolvimento de um sistema - gerentes, coordenadores, analistas, desenvolvedores - por apresentar um vocabulário de fácil entendimento [RUMBAUGH, 2000].

3.5.2.1 Coletar Inventário de Hardware

O inventário de hardware consiste em obter os dados físicos relativos aos computadores desejados.

1º passo – A aplicação de gerência HRNet Gerenciamento faz uma chamada ao script Inventario.vbs, passando como parâmetro, os nomes dos computadores nos quais deve ser executado o inventário de hardware.

2º passo – O script Inventario.vbs executa uma conexão ao *namespace* (conforme visto no capítulo 2) padrão do WMI no computador remoto.

3º passo – O script Inventario.vbs executa uma consulta através da linguagem WQL utilizando o *provider Win32*, que faz uma pesquisa no CIM dos seguintes itens referentes ao computador:

- Nome;
- Fabricante;
- Modelo;
- Processadores;
- Memória Física;
- Discos Rígidos;
- Placas de Rede;
- Placas de Vídeo;
- Placas de Som.

4º passo – O script Inventario.vbs cria um arquivo no padrão XML chamado Inventario.xml localizado no computador de gerência com os dados obtidos pela consulta executada ao CIM.

5º passo – A aplicação HRNet aguarda até que o arquivo Inventario.xml termine de ser escrito.

6º passo – A aplicação HRNet executa a leitura do arquivo Inventario.xml e exibe os dados obtidos ao usuário através da janela Inventário.

3.5.2.2 Coletar Inventário de Software

O inventário de software consiste em obter os dados referentes aos softwares instalados nos computadores desejados.

1º passo – A aplicação de gerência HRNet Gerenciamento faz uma chamada ao script Softwares.vbs, passando como parâmetro, os nomes dos computadores nos quais devem ser executado o inventário de software.

2º passo – O script Softwares.vbs executa uma conexão ao *namespace* padrão do WMI no computador remoto.

3º passo – O script Softwares.vbs executa uma consulta através da linguagem WQL utilizando o *provider Win32*, que faz uma pesquisa no CIM referente aos softwares instalados através do *Windows Installer*.

4º passo – O script Softwares.vbs cria um arquivo no padrão XML chamado Softwares.xml localizado no computador de gerência com os dados obtidos pela consulta executada ao CIM.

5º passo – A aplicação HRNet aguarda até que o arquivo Softwares.xml termine de ser preenchido.

6º passo – A aplicação HRNet executa a leitura do arquivo Softwares.xml e exibe os dados obtidos ao usuário através da janela Softwares Instalados.

3.5.2.3 Desligar e Reiniciar computadores remotos

Desligar e reiniciar computadores remotos é a possibilidade de, através da aplicação de gerência HRNet, o usuário conseguir desligar ou reiniciar um computador que esteja distante de si.

1º passo – A aplicação de gerência HRNet Gerenciamento faz uma chamada ao script DesligarReiniciar.vbs, passando como parâmetro, os nomes dos computadores que devem ser desligados ou reiniciados e o parâmetro de valor 12 para desligar ou 6 para reiniciar, conforme o desejado.

2º passo – O script DesligarReiniciar.vbs executa uma conexão ao *namespace* padrão do WMI no computador remoto.

3º passo – O script DesligarReiniciar.vbs executa um processo através da linguagem WQL utilizando o *provider Win32* e o objeto *Win32_OperatingSystem* configurando o parâmetro *Win32Shutdown(12)* para desligar o computador ou *Win32Shutdown(6)* para reiniciá-lo.

4º passo – O computador remoto é desligado ou reiniciado de acordo com a opção desejada.

3.5.2.4 Listar, parar e iniciar serviços.

1º passo – A aplicação de gerência HRNet Gerenciamento faz uma chamada ao script ListaServicos.vbs, passando como parâmetro, o nome dos computadores nos quais devem ser listados os serviços.

2º passo – O script ListaServicos.vbs executa uma conexão ao *namespace* padrão do WMI no computador remoto.

3º passo – O script ListaServicos.vbs executa uma consulta através da linguagem WQL utilizando o *provider Win32*, que faz uma pesquisa no CIM dos seguintes itens referente aos serviços:

- Nome;
- Estado;
- Status;
- Modo inicial;
- Usuário.

4º passo – O script ListaServicos.vbs cria um arquivo no padrão XML chamado ListaServicos.xml localizado no computador de gerência com os dados obtidos pela consulta executada ao CIM.

5º passo – A aplicação HRNet aguarda até que o arquivo ListaServicos.xml termine de ser preenchido.

6º passo – A aplicação HRNet executa a leitura do arquivo ListaServicos.xml e exibe os dados obtidos ao usuário através da janela Serviços.

7º passo – O usuário seleciona o serviço desejado e clica na opção Iniciar Serviço ou Parar Serviço, de acordo com sua necessidade.

- É importante lembrar que não é possível parar alguns serviços devido à preservação da estabilidade do sistema operacional além do que, para iniciar um serviço, o mesmo não pode estar em modo desabilitado.

8º passo – A aplicação de gerência HRNet Gerenciamento faz uma chamada ao script Servico.vbs, passando como parâmetro o nome do computador e o serviço pré-selecionado.

9º passo – O script Servico.vbs executa uma conexão ao *namespace* padrão do WMI no computador remoto.

10º passo – O script Servico.vbs executa um processo através da linguagem WQL utilizando o *provider Win32* e o objeto *Win32_Service* passando como parâmetro *StartService()* para iniciar o serviço ou *StopService()* para parar o serviço conforme opção pré-selecionada pelo usuário.

11º passo – O serviço é iniciado ou parado conforme desejado.

3.5.3 Problemas e soluções encontrados no desenvolvimento da ferramenta

A seguir serão apresentados os principais problemas encontrados no decorrer do desenvolvimento do projeto:

1. Formato de dados gerado

Problema:

- Na integração das funcionalidades dos scripts com a interface gráfica desenvolvida em VB .NET, tentou-se a princípio utilizar arquivos de resposta no formato texto (TXT), porém, com o decorrer do projeto viu-se que não seria a melhor opção para alcançar os objetivos desejados e propostos nesse trabalho.

Solução:

- Com isso, como opção para suprir às necessidades do projeto, foi utilizado o padrão XML (*Extensible Mark-up Language)*, por ser um padrão de mercado altamente utilizado e compatível com diversas linguagens, o qual é gerado em tempo de execução, com os dados obtidos através da execução dos scripts em linguagem VBScript.

2. Listagem de computadores disponíveis

Problema:

- No início do projeto foi desenvolvida a funcionalidade que permitia ao usuário digitar o computador desejado e executar as funcionalidades no computador indicado. Porém, durante o desenvolvimento e realização dos testes da ferramenta, verificou-se que essa opção não era viável para uma rede corporativa, em função do grande número de computadores existentes na mesma. Assim, o administrador da rede teria que saber o nome de todas as máquinas e poderia executar as funcionalidades em apenas uma máquina de cada vez, tornando a ferramenta pouco usual.

Solução:

- Para resolver os problemas relacionados à listagem de computadores disponíveis, foi criada uma funcionalidade que através do *provider Active Directory* do WMI, faz conexão ao serviço de diretórios do Windows, coleta o nome de todos os computadores do domínio e gera um arquivo no formato XML no computador de gerência para que possa ser lido pela ferramenta HRNet.

- Com isso, o problema inicial foi resolvido, porém, outro problema surgiu. Nesta etapa já era possível ter conhecimento de todas as máquinas do domínio e executar uma funcionalidade em mais de uma máquina por vez. No entanto, ainda não era possível saber quais os computadores que estariam disponíveis naquele momento. Isso poderia causar um erro ao mandar executar uma funcionalidade em um computador que não estivesse disponível, ou seja, desligado naquele momento.

- Para isso, foi desenvolvida outra funcionalidade que utiliza a lista gerada com todos os computadores do domínio e executa um comando

do próprio sistema operacional chamado PING, o qual retorna os computadores que respondem no momento da execução e gera uma nova lista em XML, apenas com os computadores possíveis de serem utilizados. Esse arquivo é lido pela ferramenta HRNet que exibe em uma janela a lista dos computadores disponíveis a serem executadas as funcionalidades desejadas.

- Após a resolução dos problemas acima, foi feito ainda, a caráter de desempenho, uma funcionalidade que contempla as duas soluções vistas. Nessa funcionalidade, os computadores são pesquisados no serviço de diretórios e é feita a verificação da disponibilidade dos mesmos, tornando assim, o processo mais rápido e eficiente.

3. Consultar Inventário de Software

Problema:

- Não é possível coletar informações de todos os softwares instalados no computador através do WMI. O inventário de software feito pelo WMI, coleta apenas informações dos softwares instalados através do *Windows Installer*.

4. Seleção de domínio

Problema:

- No início do projeto não houve preocupação quanto ao domínio que seria executada a ferramenta. Com isso a mesma tornou-se restritiva, pois, apesar de todas as funcionalidades serem dinâmicas, só podia ser utilizada em um único domínio. No entanto, na etapa final do projeto foi detectada a necessidade de expansão da ferramenta para torná-la dinâmica e possibilitar sua execução no domínio desejado.

Solução:

- Foi implementada uma função que possibilita ao usuário informar o domínio no qual deseja trabalhar, podendo assim, facilmente, utilizar a ferramenta de forma mais abrangente.

3.6 APRESENTAÇÃO DA APLICAÇÃO HRNET GERENCIAMENTO

Conforme visto no subitem 3.4 desse capítulo, alguns requisitos devem ser observados para obter o funcionamento da ferramenta de forma desejável.

A seguir serão demonstradas as telas de funcionamento da ferramenta, bem como suas descrições e características. Para utilizar a ferramenta basta iniciar o aplicativo HRNet.

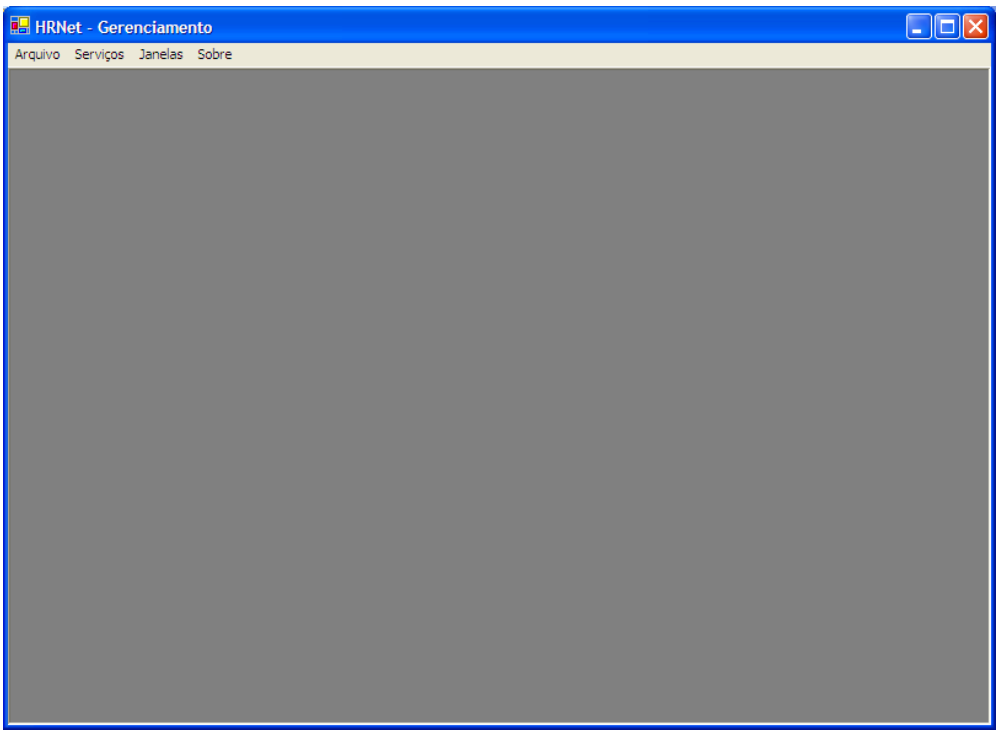

Figura 3.3 – Tela Principal do HRNet Gerenciamento

Na Figura 3.3 é demonstrada a tela principal do HRNet na qual o usuário pode interagir com a ferramenta. Nela é possível escolher a funcionalidade desejada a ser executada.

No menu **Arquivo** existe a opção **Sair** para encerrar e fechar a execução da aplicação. No menu **Serviços** é possível selecionar as opções de **Listar Computadores, Listar Serviços, Listar Softwares e Listar Hardware**. Já no menu **Janelas** é possível selecionar o tipo de visualização preferida **(cascata/horizontal/vertical)** e ainda, alterar entre as janelas abertas até o momento. Por fim, no menu **Sobre,** possui informações referentes ao graduando, ao orientador e ao projeto.

No intuito de disponibilizar uma opção a mais e tornar a ferramenta apropriada para uso em qualquer domínio, foi implementada uma função que possibilita o usuário informar o domínio no qual deseja pesquisar as máquinas existentes como é possível ver na Figura 3.4.

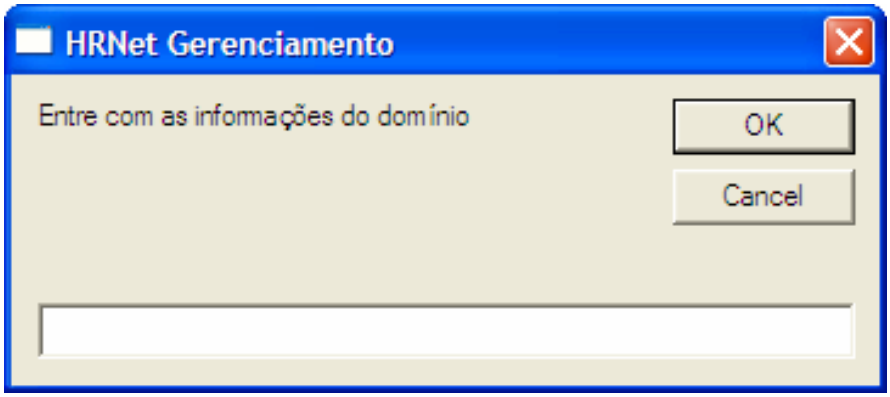

Figura 3.4 – Tela de Informação do Domínio

Ao executar a ferramenta é preciso primeiramente selecionar o domínio no qual deseja se conectar. Após a seleção do domínio o usuário deve listar os computadores disponíveis no domínio pré-selecionado e selecionar o(s) computador(es) aos quais deseja se conectar para executar a funcionalidade escolhida.

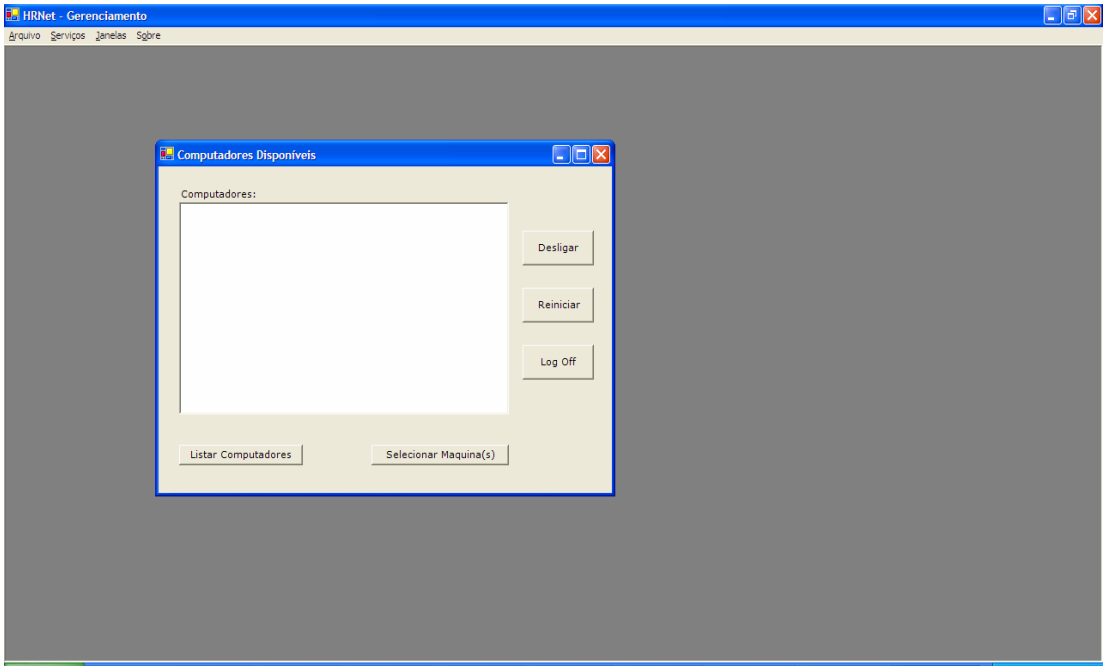

Figura 3.5 – Tela de Listagem dos Computadores

Na Figura 3.5 é apresentada a tela de listagem de computadores disponíveis. Para acessar essa tela o usuário deve clicar em **Serviços** e posteriormente em **Listar Computadores.**

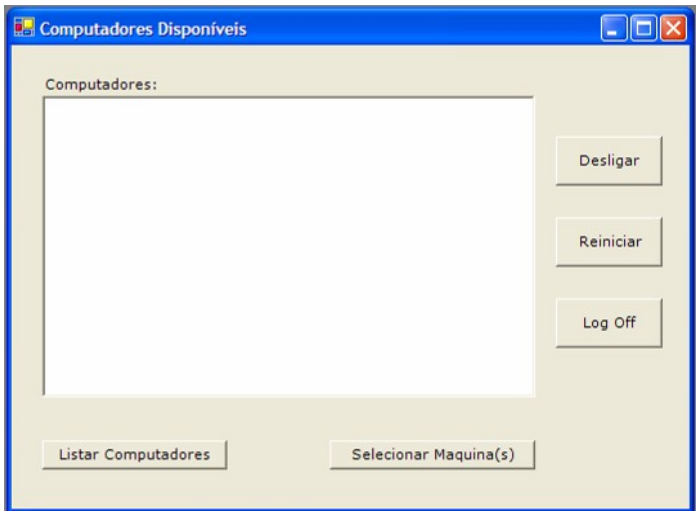

Figura 3.6 – Tela de Computadores Disponíveis

Na Figura 3.6 é mostrada a tela de Computadores Disponíveis. O usuário deve clicar em **Listar Computadores** para obter a lista de computadores disponíveis. Após a listagem dos computadores, o usuário seleciona o(s)

computador(es) em que deseja executar alguma das funcionalidades disponíveis na ferramenta e clica em **Selecionar Máquina(s)**.

Para desligar, reiniciar ou executar logoff no(s) computador(es) desejado(s), basta selecioná-lo(s) e clicar na opção referente.

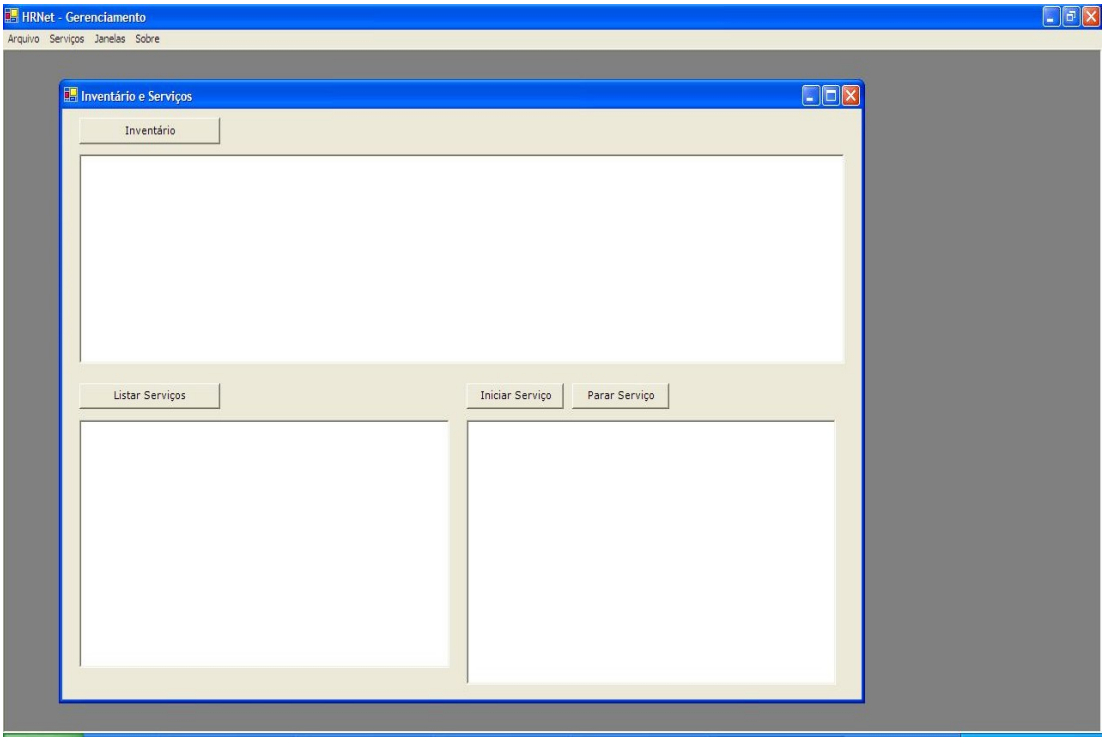

Figura 3.7 – Tela de Inventário e Serviços

Na Figura 3.7 é apresentada a tela de Inventário e Serviços na qual o usuário pode coletar o inventário de hardware dos computadores pré-selecionados clicando em **Inventário.** Listar os serviços e seus estados clicando em **Listar Serviços**, e ainda iniciar ou parar um serviço de acordo com sua necessidade.

Para iniciar um serviço, basta selecionar o serviço desejado que esteja com o estado parado e clicar em **Iniciar Serviço** e para parar um serviço, é necessário apenas selecionar o serviço desejado que esteja com o estado iniciado e clicar em **Parar Serviço**.

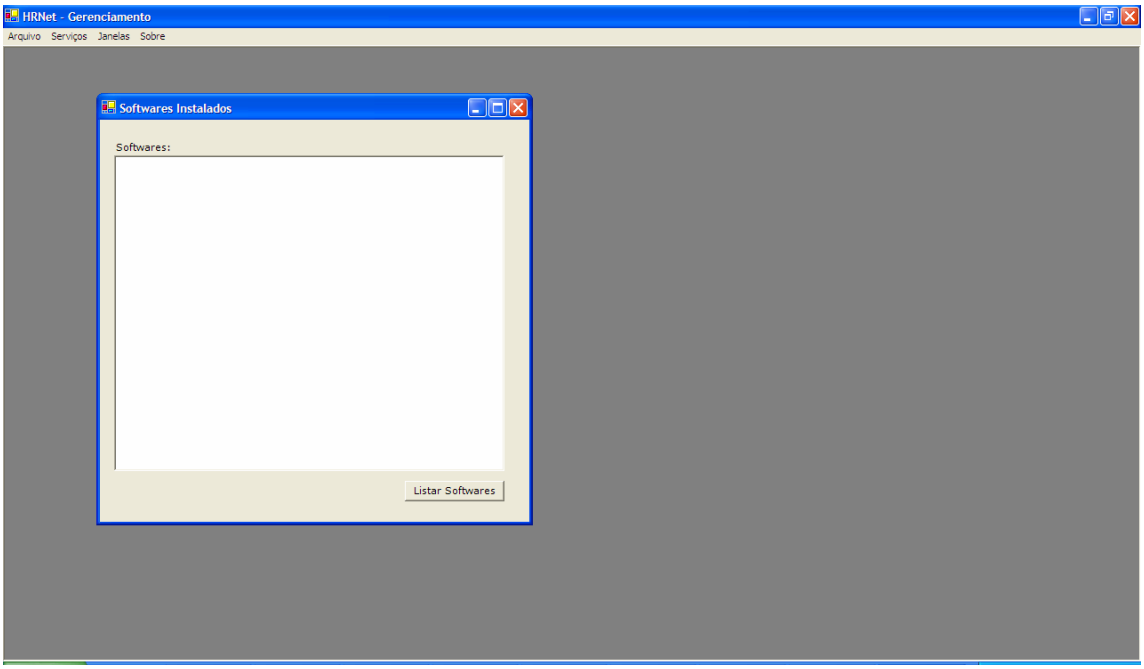

Figura 3.8 – Tela de Softwares Instalados

Na Figura 3.8 é mostrada a tela de Softwares Instalados. Nessa tela, basta clicar em Listar Softwares para que seja obtido o Inventário de Software do(s) computador(es) pré-selecionados.

3.7 HOMOLOGAÇÃO E RESULTADOS OBTIDOS

A homologação da ferramenta e os resultados obtidos foram baseados no modelo de plano de testes IEEE 829 [IEEE,1998].

3.7.1 Identificação do Plano de Testes

Plano de testes da ferramenta HRNet Gerenciamento.

3.7.2 Itens de Teste

- Desligar computadores;
- Reiniciar computadores;
- Coletar informações de hardware;

- Coletar informações de software;
- Listar Serviço;
- Iniciar Serviço;
- Parar Serviço.

3.7.3 Riscos Potenciais

- Não possuir *hardware* e *softwares* necessários para a execução dos testes;
- Não possuir permissão de acesso como administrador nos computadores remotos;
- Ter qualquer tipo de *hardware* ou *software* (*firewall, antispyware, antivírus)* bloqueando o acesso aos clientes ou a execução de scripts;
- Não possuir acesso ao serviço de diretórios para coletar o nome dos computadores existentes na rede;
- Não ter o serviço WMI iniciado no computador remoto.

3.7.4 Abordagem de Teste (Estratégia)

- Realização de testes no ambiente de rede do Tribunal Superior Eleitoral – TSE;
- Realização de testes em 5 computadores no laboratório 8001 do UniCEUB;
- Execução dos testes em diferentes combinações de *hardware* e *software*;
- Utilização de técnica de teste de desempenho;
- Utilização da ferramenta de análise de desempenho do Microsoft Windows;
- Coletar as seguintes métricas de utilização de hardware:

    - Processador;
    - Memória Ram;
    - Disco Rígido.

- Realização de testes do tempo gasto na execução de cada funcionalidade existente, de acordo com o número de computadores nos quais será executada tal funcionalidade.

3.7.5 Critérios de Sucesso ou Falha

Serão considerados critérios de sucesso quando:

- Os resultados requeridos forem obtidos com sucesso;
- Não houver erro durante o processo de execução da funcionalidade;
- Todas as funcionalidades existentes forem executadas com êxito.

Serão considerados critérios de falha quando:

- Não conseguir comunicação com os computadores remotos;
- Ocorrerem erros na ferramenta que impeçam a execução das funcionalidades;
- Houver execução inadequada das funcionalidades existentes.

3.7.6 Artefatos Entregues

- Documento de plano de testes baseado no IEEE 829;
- Informações dos testes realizados;
- Tabela com os tempos obtidos na execução das funcionalidades;
- Problemas encontrados e ações corretivas.

3.7.7 Necessidades do Ambiente de Teste

- Disponibilidade de computadores ligados em rede;
- Computadores com fonte ATX;
- Existência do serviço de diretório do Windows;
- Computadores remotos inseridos no domínio existente;
- Computadores com serviço WMI iniciado;
- *Microsoft framework .NET* instalado no computador gerente;
- Sistema operacional Windows 2000, XP ou 2003.

3.7.8 Treinamentos e Alocações Necessários

- Não serão necessários treinamentos e alocações para os testes e utilização da ferramenta.

3.7.9 Responsabilidades dos Testes

- É responsabilidade do Professor Fernando do UniCEUB disponibilizar 5 computadores com fonte ATX no laboratório 8001 e possibilitar a utilização dos mesmos para a execução dos testes;
- É responsabilidade do técnico Márcio do UniCEUB prover o acesso às dependências do laboratório 8001;
- É responsabilidade do Setor de Micro Informática (SMI) do TSE disponibilizar os computadores utilizados durante os testes.
- É responsabilidade do graduando, Henrique Rodrigues, preparar e realizar os testes nos ambientes utilizados para teste, tanto no UniCEUB como no TSE.

3.7.10 Configuração do Ambiente de Homologação

Para a execução dos testes com a ferramenta HRNet foram utilizados os seguintes equipamentos:

Equipamento 1

Hardware:

- Processador Pentium Intel Centrino 1.6 GHz;
- 512 MB de memória ram;
- 60 GB de disco rígido.

Softwares:

- Sistema operacional Microsoft Windows Server 2003 Enterprise Edition com Service Pack 1;
- Microsoft Visual Studio .NET;
- Microsoft Bloco de Notas;
- HRNet Gerenciamento.

Equipamento 2

Hardware:

- Processador Pentium IV 2.66 GHz;
- 512 MB de memória RAM;
- 40 GB de disco rígido.

Softwares:

- Sistema operacional Microsoft Windows XP Professiona com Service Pack 2l;
- Microsoft Visual Studio .NET;
- Microsoft Bloco de Notas;
- HRNet Gerenciamento.

3.7.11 Resultados Obtidos

Com a utilização do parque computacional existente no TSE, foi possível realizar testes em vários computadores simultâneos tornando o processo de homologação da ferramenta preciso e confiante. Além dos testes executados no TSE, também foram realizados testes no UniCEUB no intuito de demonstração para a banca examinadora.

Na tabela 1 é possível verificar o tempo gasto na execução de cada funcionalidade de acordo com o número de computadores em execução. Esses testes foram realizados no ambiente do TSE.

Tabela 3.1 – Relação de funcionalidades (tempo em segundos x computadores)

	1 Computador	5 Computadores	50 Computadores	100 Computadores
Inventário de Hardware	5,4	25,5	270,9	525,3
Inventário de Software	5,7	26,1	283,4	551,6
Desligar Computadores	28,2	43,0	72,6	85,3
Reiniciar Computadores	31,2	51,8	79,2	88,4
Listar Serviços	5,5	25,8	277,8	538,6
Parar Serviço	5,9	-	-	-
Iniciar Serviço	5,8	-	-	-

Analisando os resultados obtidos na tabela 3.1, de acordo com os testes realizados, verificou-se que as funcionalidades: Inventário de Hardware, Inventário de Software e Listar Serviços, são executadas aproximadamente em progressão aritmética, na qual a cada computador adicionado ao teste, são acrescidos por volta de 5 segundos. Resumidamente é possível concluir que, cada computador demora aproximadamente 5 segundos para a execução completa de cada uma dessas funcionalidades.

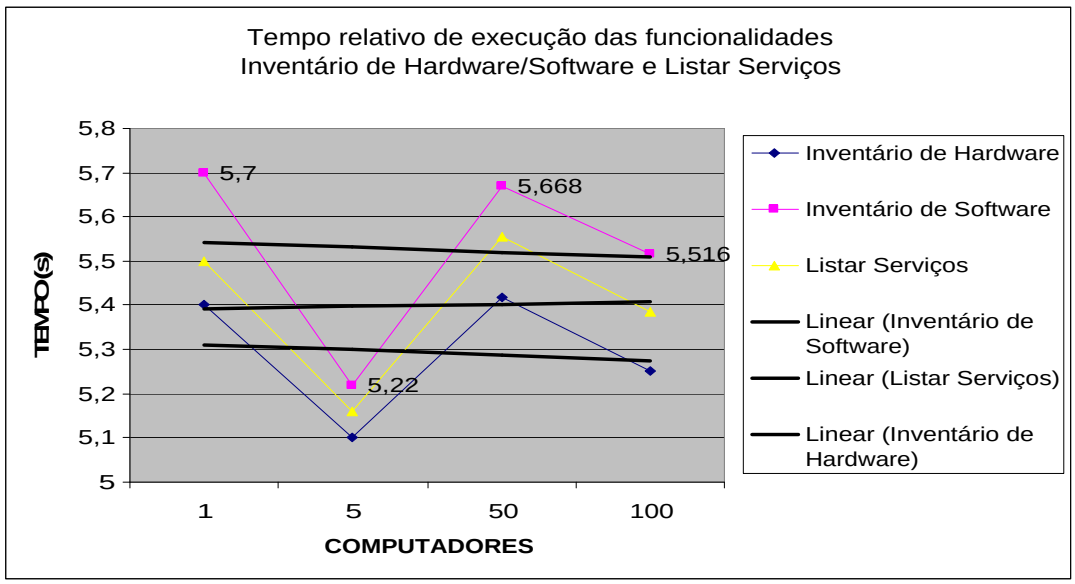

Figura 3.9 - Tempo relativo de execução das funcionalides de inventário e listar serviços

Com a análise da figura a cima, foi possível obter uma conclusão mais precisa da estabilidade da ferramenta, na qual verificou-se que a ferramenta mantem-se estável tanto para a execução das funcionalidades citadas em 1 computador como para 100 computadores. Também foi notado, com a linha de tendência linear utilizada no gráfico, que o tempo relativo (tempo de execução total dividido pelo número de computadores) na execução dessas funcionalidades foi estável independente do número de computadores utilizados nos testes.

Analisando ainda os tempos obtidos nos testes foi possível observar que as funcionalidades Desligar Computadores e Reiniciar Computadores são executadas proporcionalmente à velocidade do computador remoto. Os computadores testados que possuíam *hardware* atuais, executaram essas funcionalidades com maior rapidez, enquanto os que possuíam *hardwares* ultrapassados demoraram um tempo maior.

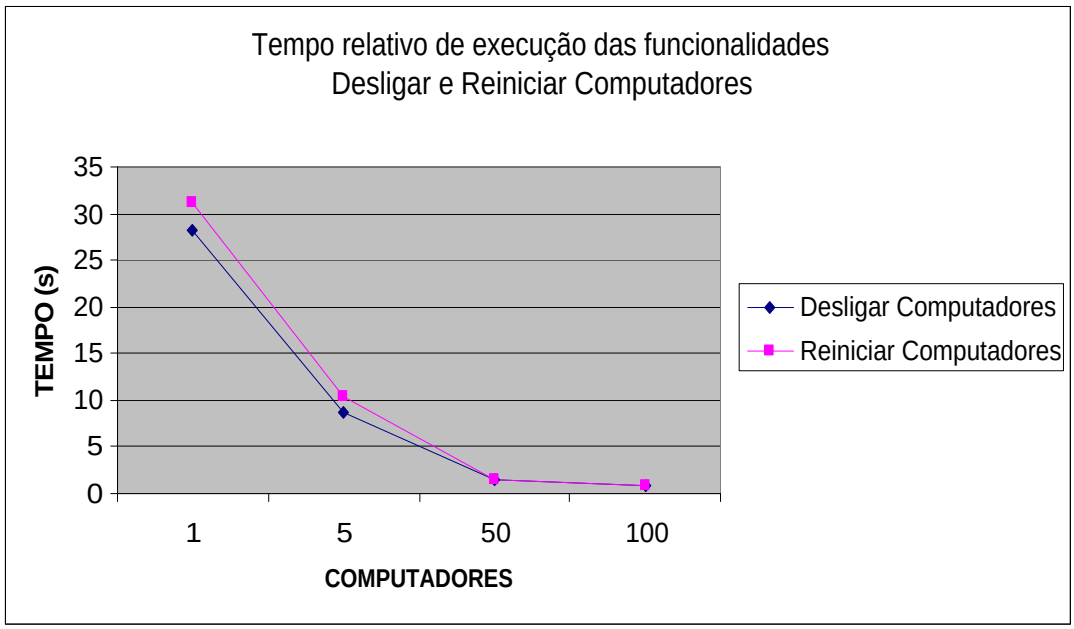

Figura 3.10 - Tempo relativo na execução das funcionalidades desligar/reiniciar computador

Também é possível verificar na Figura 3.10, que o tempo relativo é reduzido consideravelmente à medida que o número de computadores aumenta, chegando a um valor inferior a 1 segundo por computador na execução dessas funcionalidades na execução com 100 computadores.

Para as funcionalidades Parar Serviço e Iniciar Serviço, só é possível executar os testes em um computador por vez, pois é necessário selecionar o computador e o serviço desejado para que seja executada a funcionalidade. Esse processo demorou em média 5,85 segundos para ser executado.

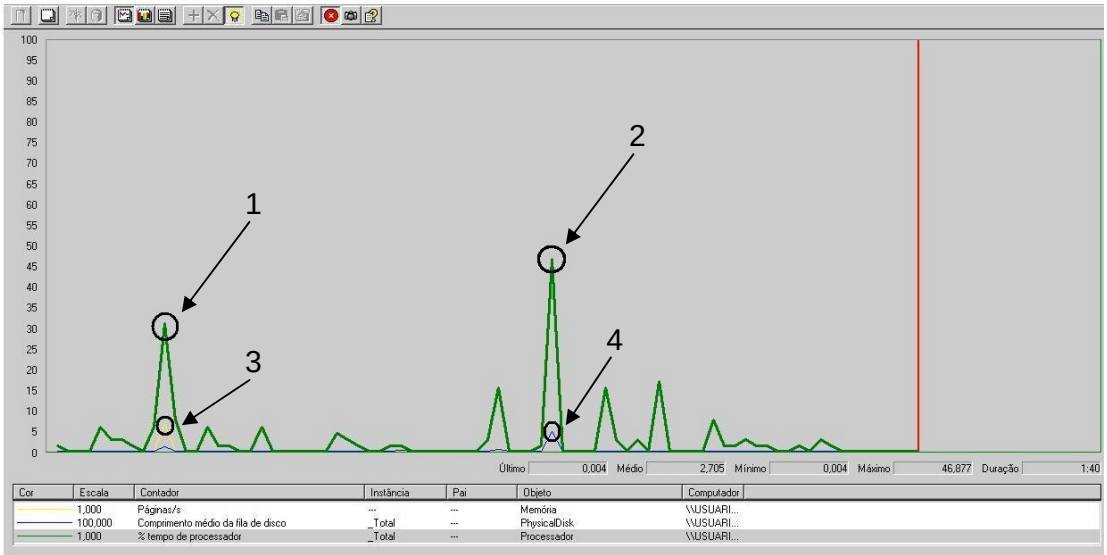

Figura 3.11 – Utilização (%) do processador, memória e disco (Inventário de Hardware)

Na Figura 3.11 é possível verificar quantos por cento é utilizado de processamento, de paginação da memória e da fila do disco rígido durante a execução da funcionalidade de inventário de hardware para 5 computadores. Como se pode verificar, existem quatro picos na figura, sendo, dois picos de processamento (ver 1 e 2), um pico de paginação de memória (ver 3) e um pico da fila do disco rígido (ver 4).

O pico 1 (processamento) ocorre simultaneamente ao pico 3 (paginação de memória), provavelmente no momento em que a aplicação de gerência executa os procedimentos de chamada ao script solicitado para a execução da funcionalidade nos computadores remotos.

Após o período em que a aplicação aguarda a resposta dos computadores remotos, ocorre o pico 2 (processamento) simultaneamente ao pico 4 (fila do disco rígido), provavelmente no momento em que os dados recebidos são processados e armazenados no disco rígido do computador de gerência.

Depois dos picos principais (2 e 4), ocorre provavelmente, o processamento das últimas etapas da aplicação de gerência para a apresentação dos dados solicitados pelo usuário.

Na análise de desempenho da aplicação de gerência é possível concluir que a mesma não necessita de processamento elevado, ou seja, o computador com a aplicação de gerência não ficará sobrecarregado durante o processo de execução das funcionalidades.

3.7.12 Problemas Encontrados e Ações Corretivas

1. Permissão e bloqueio

   Problema:

   - Nos testes realizados um dos problemas mais comuns foi em relação a permissões de acesso de administrador nos computadores remotos e softwares instalados que bloqueavam a execução dos scripts.

   Solução:

   - Nesses casos, quando possível, foi dada a permissão de acesso ao administrador nos computadores remotos e/ou desabilitados os softwares que bloqueavam a execução dos scripts.

2. Parar e Iniciar Serviço

   Problema:

   - As funcionalidades Parar Serviço e Iniciar Serviço só foram executadas com êxito no computador local, ou seja, nos computadores remotos essas funcionalidades não foram executadas com sucesso.

Solução:

- Os scripts Parar Serviço e Iniciar Serviço foram modificados, pois havia um erro de lógica na parte de conexão remota, assumindo então, o computador local como computador selecionado a ser executado a função desejada. Resolvendo o problema na conexão remota, tais funcionalidades foram testadas e executadas de forma esperada. Assim, através da ferramenta HRNet, foi possível executar essas funções com sucesso.

Capítulo 4 - Conclusões

Este projeto foi de grande valia para a vida acadêmica do graduando. Com certeza as cadeiras cursadas durante o curso de Engenharia da Computação do UniCEUB, proporcionaram ao graduando conhecimentos amplos e específicos, que possibilitaram a elaboração do projeto aqui apresentado.

O sucesso do desenvolvimento e conclusão do projeto, assim como os problemas enfrentados e as soluções encontradas, encorajam o graduando a exercer a profissão de Engenheiro da Computação com maior responsabilidade, empenho e dedicação, tendo clareza das dificuldades que poderão surgir ao longo de sua carreira, ao mesmo tempo em que, poderá experimentar a satisfação pessoal e profissional ao concluir um curso de tal envergadura.

Com o estudo feito para a execução do projeto, verificou-se a existência de diversas implementações de gerência de rede baseadas no padrão WBEM. O WMI, implementação proposta pela Microsoft, mostrou-se muito útil para o gerenciamento de estações que utilizam sistemas operacionais da Microsoft.

A utilização de scripts e da implementação WMI, traz grandes benefícios e desempenho na execução de tarefas administrativas de rede. Pois, sendo esta baseada no padrão WBEM, o qual unifica diversos padrões de gerenciamento (SNMP, DMI, CMIP), se consegue prover soluções completas de gerência local e remota para ambiente Windows, utilizando uma única interface de gerenciamento. Com isso é possível obter uma redução no esforço administrativo e no TCO da empresa.

Pôde-se observar também, que a utilização do WMI, assim como as outras tecnologias discutidas nessa monografia e utilizadas na elaboração do projeto, foram consideradas satisfatórias para o intuito do projeto, pois possibilitaram a validação dos objetivos propostos pelo mesmo, atendendo assim, às necessidades expressas pelos administradores nas entrevistas realizadas.

Uma restrição verificada no decorrer do projeto deve-se ao fato da possibilidade de gerenciamento se dar apenas em computadores clientes que estejam rodando o serviço WMI. Este serviço só existe em computadores com sistema operacional Microsoft Windows, impossibilitando a execução da ferramenta HRNet em outros sistemas operacionais.

Além disso, a ferramenta exige que se tenha instalado no computador de gerência, o *framework* .NET, disponível apenas para o sistema operacional Microsoft Windows. Não é necessária a instalação do *framework* .NET nos computadores clientes.

Para a realização desse projeto observou-se a importância das disciplinas ministradas no curso de Engenharia da Computação do UniCEUB. Dentre elas pode-se destacar:

- Redes de Computadores, Tópicos Avançados de Rede I e II: foram a base desse projeto; pois, além de dar uma visão geral sobre redes de computadores, viabilizaram o conhecimento das técnicas de gerenciamento aqui utilizadas;

- LTP1, LTP2 e Estrutura de Dados: forneceu subsídios para o raciocínio lógico na confecção dos scripts e programas da ferramenta HRNet Gerenciamento;

- Engenharia de Programas: contribuiu para o entendimento dos processos de desenvolvimento de software, UML e testes da ferramenta;

- Introdução à Engenharia: balizou os conhecimentos necessários para a confecção de um trabalho de Engenharia da Computação;

- Sistemas Operacionais: proporcionou conhecimento necessário ao entendimento dos mecanismos do S.O. e suas APIs que são chamadas pelo WMI;

- Probabilidade e Estatística: determinou a escolha, confecção e interpretação dos gráficos e dados utilizados ao longo do trabalho;

- Bancos de Dados: apesar de não ter sido utilizado no modelo relacional por este trabalho, abre possibilidades para persistência dos XML em trabalhos futuros;

- Arquitetura de Computadores: permitiu conhecimento base para escolha dos itens a serem monitorados nos testes da ferramenta, como CPU, memória e disco.

4.1 TRABALHOS FUTUROS

No decorrer da elaboração do projeto, algumas contribuições para trabalhos futuros foram vistos como interessantes e serão apresentadas a seguir:

4.1.1 Providers Customizados

A tecnologia WMI provê suporte a *providers* customizados criados por terceiros, os quais podem ser usados para requisição de serviços relacionados ao gerenciamento de objetos de ambientes específicos.

4.1.2 Implementações WBEM

Utilizar as funcionalidades apresentadas neste projeto para criar uma aplicação de gerência para ambientes heterogêneos, utilizando uma das implementações do WBEM.

4.1.3 Outras Funcionalidades

Incorporar novas funcionalidades a este projeto, possibilitando, por exemplo, a instalação remota de *softwares* ou a alteração de configurações existentes.

4.1.4 Autenticação de Usuários

Criar um método de autenticação do usuário para acesso ao serviço WMI, provendo assim, a possibilidade de informar o usuário com permissão administrativa no computador remoto.

4.1.5 Inventário de Software

Criar um módulo que faça o inventário de todos os *softwares* instalados no computador, pois o WMI só dá suporte a fazer o inventário dos *softwares* instalados através do *Windows Installer*, como visto no capítulo 2.

Referências Bibliográficas

[BRAZ, 2003] BRAZ JUNIOR, Edson Luis da Silva. Protótipo de um software para gerência de sistemas baseado no padrão WBEM utilizando o WMI. 2003. 63 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

[CARVILHE, 2000] CARVILHE, José Luís Vieira. **A utilização de tecnologias WEB em sistemas de gerência corporativa**. 2000. 102 f. Monografia (Especialização em Sistemas Distribuídos) – Pontifícia Universidade Católica do Paraná, Curitiba.

[CASTRO, 2002] CASTRO, João Carlos. **Gerenciamento de sistemas baseado no padrão WBEM**. 2002. 71 f. Monografia (Tecnólogo em Processamento de Dados) – Universidade Tiradentes, Aracaju.

[CHOI, 2004] CHOI, MI-Jung. Design of a WBEM-based Management System for Ubiquitous Computing Servers. Dept. of Computer Science and Engineering, POSTECH: 2001. Disponível em: <http://www.dmtf.org/education/academicallian ce/mjchoi_2004.pdf>. Acessado em: 13 de abril de 2005.

[DMTF, 1999] **Distributed Management Task Force**, 1999. Disponível em: <http://www.dmtf.org/>. Acessado em: 24 de novembro de 2004.

[IEEE, 1998] **IEEE 829 Test Plan**, University of Toronto at Scarborough. Disponível em:<http://www.utsc.utoronto.ca/~rosselet/cscd08/ref05/ieee829.html>. Acessado em: 03 de junho de 2005.

[JONES, 2004] JONES, Don. **Managing Windows® with VBScript and WMI**, Pearson Education Inc, Boston: Addison Wesley, 2004.

[MARTISSON, 2002] MARTINSSON, Tobias. **Desenvolvendo scripts XML e WMI para o Microsoft SQL Server 2000**. São Paulo: Makron Books, 2002.

[MASTON, 1999] MATSON, Michael. **Managing Windows with WMI**, Microsoft Corporation, 1999. Disponível em: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwmi/html/mngwmi.asp>. Acessado em: 20 de agosto de 2004.

[MICROSOFT, 1999] MICROSOFT, Microsoft Corporation. **Learn-WMI,** 1999. Disponível em: <http://www.microsoft.com/downloads/release.asp?releaseid=12570>. Acessado em: 23 de agosto de 2004.

[MICROSOFT, 2000] **Windows Management Instrumentation and Simple Network Protocol Management**, Microsoft Corporation, 2000. Disponível em: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwmi/html/wmiandsnpm.asp> . Acessado em: 25 de agosto de 2004.

[MICROSOFT, 2004] **Active Directory Overview**, Microsoft Corporation, 2004. Disponível em:<http://www.microsoft.com/technet/prodtechnol/windowsserver2003/library/ServerHelp/7c981583-cf41-4e6c-b1f6-5b8863475ede.mspx>.
Acessado em: 17 de setembro de 2004.

[MICROSOFT, 2005] **Platform SDK: Windows Management Instrumentation, About–WMI**. Microsoft Corporation, 2005. Disponível em: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/about_wmi.asp>. Acessado em: 25 de agosto de 2004.

[MICROSOFT, 2005a] **Platform SDK: Windows Management Instrumentation, Query Language Support**. Microsoft Corporation, 2005. Disponível em: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/query_language_support.asp>. Acessado em: 29 de agosto de 2004.

[MSDN TRAINING, 2001] MSDN TRAINING. **Scripting Microsoft® Windows® Management Instrumentation**, course 2439A. Microsoft Official Curriculum, 2001.

[NOVELL, 2000] **Novell NDS edirectory Overview**, 2000. Disponível em: <http://support.novell.com/techcenter/articles/dnd20000305.html>. Acessado em: 17 de setembro de 2004.

[OPENPEGASUS, 2004] **OpenPegasus**, 2004. Disponível em: <http://www.openpegasus.org/>. Acessado em: 05 de março de 2005.

[OPENWBEM, 2004] **OpenWBEM project**, 2004. Disponível em: <http://www.openwbem.org/>. Acessado em: 05 de março de 2005.

[ORACLE, 2001] **OID – Oracle Internet Directory , 2001**. Disponível em: <http://www.oracle.com/oramag/oracle/01-jan/o11o8i.html>. Acessado em: 17 de setembro de 2004.

[PEREIRA, 2001] PEREIRA, Mateus Casanova. **Administração e gerência de computadores.** 2001**.** 106 f. Dissertação (Mestrado em Ciências da Computação) - Universidade Federal de Santa Catarina, Florianópolis.

[RUMBAUGH, 2000] RUMBAUGH, James - BOOCH, Grady - JACOBSON, Ivar - **UML - Guia do Usuário** - Campus - 2000 1ª Edição

[SACRAMENTO, 2003] SACRAMENTO, Paulo Jorge Ferraz de Menezes. **WBEM e WMI: Noções e Estrutura de uma aplicação**, Departamento de Engenharia Informática, Universidade de Coimbra - 3030 Coimbra, Portugal, 2003.

[STALLINGS, 1999] STALLINGS, William. **SNMP, SNMPv2, SNMPv3, and RMON 1 and 2.** Boston: Addison-Wesley, 1999. xv, 619 p.

[TANENBAUM, 1997] TANENBAUM Andrew S. **Rede de Computadores**. Ed. Campus 5 ed. Rio de Janeiro, 1997.

[WBEM SERVICES, 2004] **WBEM Services**, 2004. Disponível em: <http://wbemservices.sourceforge.net/>. Acessado em: 05 de março de 2005.

[XML, 2004] **Extensible Markup Language (XML) 1.0 (Third Edition)**, 2004. Disponível em: <http://www.w3.org/TR/REC-xml/>. Acessado em: 12 de fevereiro de 2005.

Outras referências utilizadas para a confecção do quadro comparativo de ferramentas do mercado que utilizam tecnologia WMI:

Software **Clarus Evolution 2.0**. Disponível em: <http://www.cyrga.com.br/>. Acessado em: 08 de abril de 2005.

Software **Domain Administration Tool 3.2**. Disponível em: <http://www.pukka.com/default.asp>. Acessado em: 08 de abril de 2005.

Software **Everest Corporate Edition**. Disponível em: <http://www.lavalys.com/>. Acessado em: 07 de abril de 2005.

Software **Hyena 6.3**. Disponível em: <http://www.systemtools.com/hyena/>. Acessado em: 07 de abril de 2005.

Software **MoniAPP**. Disponível em: <http://monitapp.serprest.pt/>. Acessado em: 07 de abril de 2005.

Software **Network Inventory Manager 3.0.11.5**. Disponível em: <http://www.microforge.net/index.html>. Acessado em: 07 de abril de 2005.

Software **Sandra 2005 Enterprise**. Disponível em: <http://www.sisoftware.co.uk/>. Acessado em: 08 de abril de 2005.

Apêndice A – Códigos de Programação

frmHRNet.vb

```
'**************************************************************************
'Nome: frmHRNet.vb
'Autor: Henrique Rodrigues
'Descricao: Formulario principal da ferramenta, atraves dele e possivel
'abrir os demais formularios existentes.
'**************************************************************************
Public Class frmHRNet
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form
Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents mnuSobre As System.Windows.Forms.MenuItem
    Friend WithEvents mnuPrincipal As System.Windows.Forms.MainMenu
    Friend WithEvents mnuArquivo As System.Windows.Forms.MenuItem
    Friend WithEvents mnuServicos As System.Windows.Forms.MenuItem
    Friend WithEvents mnuSair As System.Windows.Forms.MenuItem
    Friend WithEvents mnuAjuda As System.Windows.Forms.MenuItem
    Friend WithEvents mnuListaComputadores As
System.Windows.Forms.MenuItem
    Friend WithEvents mnuListaSoftwares As System.Windows.Forms.MenuItem
    Friend WithEvents mnuJanelas As System.Windows.Forms.MenuItem
    Friend WithEvents mnuListaServicos As System.Windows.Forms.MenuItem
    Friend WithEvents mnuHorizontal As System.Windows.Forms.MenuItem
    Friend WithEvents mnuVertical As System.Windows.Forms.MenuItem
    Friend WithEvents mnuCascata As System.Windows.Forms.MenuItem
    Friend WithEvents MenuItem1 As System.Windows.Forms.MenuItem
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
```

```
        Me.mnuPrincipal = New System.Windows.Forms.MainMenu
        Me.mnuArquivo = New System.Windows.Forms.MenuItem
        Me.mnuSair = New System.Windows.Forms.MenuItem
        Me.mnuServicos = New System.Windows.Forms.MenuItem
        Me.mnuListaComputadores = New System.Windows.Forms.MenuItem
        Me.mnuListaServicos = New System.Windows.Forms.MenuItem
        Me.mnuListaSoftwares = New System.Windows.Forms.MenuItem
        Me.mnuJanelas = New System.Windows.Forms.MenuItem
        Me.mnuCascata = New System.Windows.Forms.MenuItem
        Me.mnuHorizontal = New System.Windows.Forms.MenuItem
        Me.mnuVertical = New System.Windows.Forms.MenuItem
        Me.mnuAjuda = New System.Windows.Forms.MenuItem
        Me.mnuSobre = New System.Windows.Forms.MenuItem
        Me.MenuItem1 = New System.Windows.Forms.MenuItem
        '
        'mnuPrincipal
        '
        Me.mnuPrincipal.MenuItems.AddRange(New
System.Windows.Forms.MenuItem() {Me.mnuArquivo, Me.mnuServicos,
Me.mnuJanelas, Me.mnuAjuda})
        '
        'mnuArquivo
        '
        Me.mnuArquivo.Index = 0
        Me.mnuArquivo.MenuItems.AddRange(New
System.Windows.Forms.MenuItem() {Me.mnuSair})
        Me.mnuArquivo.Text = "&Arquivo"
        '
        'mnuSair
        '
        Me.mnuSair.Index = 0
        Me.mnuSair.Text = "Sai&r"
        '
        'mnuServicos
        '
        Me.mnuServicos.Index = 1
        Me.mnuServicos.MenuItems.AddRange(New
System.Windows.Forms.MenuItem() {Me.mnuListaComputadores,
Me.mnuListaServicos, Me.mnuListaSoftwares, Me.MenuItem1})
        Me.mnuServicos.Text = "&Serviços"
        '
        'mnuListaComputadores
        '
        Me.mnuListaComputadores.Index = 0
        Me.mnuListaComputadores.Text = "Listar &Computadores"
        '
        'mnuListaServicos
        '
        Me.mnuListaServicos.Index = 1
        Me.mnuListaServicos.Text = "Listar &Serviços"
        '
        'mnuListaSoftwares
        '
        Me.mnuListaSoftwares.Index = 2
        Me.mnuListaSoftwares.Text = "Listar S&oftwares"
        '
        'mnuJanelas
        '
        Me.mnuJanelas.Index = 2
        Me.mnuJanelas.MdiList = True
```

```
        Me.mnuJanelas.MenuItems.AddRange(New
System.Windows.Forms.MenuItem() {Me.mnuCascata, Me.mnuHorizontal,
Me.mnuVertical})
        Me.mnuJanelas.Text = "&Janelas"
        '
        'mnuCascata
        '
        Me.mnuCascata.Index = 0
        Me.mnuCascata.Text = "Organizar &cascata"
        '
        'mnuHorizontal
        '
        Me.mnuHorizontal.Index = 1
        Me.mnuHorizontal.Text = "Organizar &horizontal"
        '
        'mnuVertical
        '
        Me.mnuVertical.Index = 2
        Me.mnuVertical.Text = "Organizar &vertical"
        '
        'mnuAjuda
        '
        Me.mnuAjuda.Index = 3
        Me.mnuAjuda.MenuItems.AddRange(New
System.Windows.Forms.MenuItem() {Me.mnuSobre})
        Me.mnuAjuda.Text = "S&obre"
        '
        'mnuSobre
        '
        Me.mnuSobre.Index = 0
        Me.mnuSobre.Text = "&Sobre"
        '
        'MenuItem1
        '
        Me.MenuItem1.Index = 3
        Me.MenuItem1.Text = "Listar &Hardware"
        '
        'frmHRNet
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(6, 14)
        Me.ClientSize = New System.Drawing.Size(800, 533)
        Me.Font = New System.Drawing.Font("Verdana", 8.25!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.IsMdiContainer = True
        Me.Menu = Me.mnuPrincipal
        Me.Name = "frmHRNet"
        Me.Text = "HRNet - Gerenciamento"

    End Sub

#End Region

    Private Sub mnuSobre_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuSobre.Click
        Dim frm As New frmAbout
        frm.MdiParent = Me
        frm.Show()
    End Sub

    Private Sub mnuSair_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuSair.Click
```

```
        Application.Exit()
    End Sub

    Private Sub mnuListaComputadores_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles mnuListaComputadores.Click
        Dim frm As frmListaComputadores
        frm = frmListaComputadores.InstanciarFormulario()
        frm.MdiParent = Me
        frm.Show()
    End Sub

    Private Sub mnuListaServicos_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles mnuListaServicos.Click
        Dim frm As frmServicos
        frm = frmServicos.InstanciarFormulario()
        frm.MdiParent = Me
        frm.Show()
    End Sub

    Private Sub mnuListaSoftwares_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles mnuListaSoftwares.Click
        Dim frm As frmListaSoftwares
        frm = frmListaSoftwares.InstanciarFormulario()
        frm.MdiParent = Me
        frm.Show()
    End Sub

    Private Sub mnuCascata_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles mnuCascata.Click
        Me.LayoutMdi(MdiLayout.Cascade)
    End Sub

    Private Sub mnuHorizontal_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles mnuHorizontal.Click
        Me.LayoutMdi(MdiLayout.TileHorizontal)
    End Sub

    Private Sub mnuVertical_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles mnuVertical.Click
        Me.LayoutMdi(MdiLayout.TileVertical)
    End Sub

    Private Sub frmHRNet_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    End Sub
End Class
```

frmListaComputadores.vb

```
'**************************************************************************
'Nome: frmListaComputadores.vb
'Autor: Henrique Rodrigues
'Descricao: Formulário de listagem dos computadores disponiveis, onde e
'possível selecionar os computadores que deseja executar uma
'funcionalidade do sistema. Nesse formulário ainda e possível desligar,
'reiniciar ou executar logoff em um computador selecionado.
'**************************************************************************
Public Class frmListaComputadores
    Inherits System.Windows.Forms.Form
    Public NomeComputador As String
```

```
    Private Shared formulario As frmListaComputadores

#Region " Windows Form Designer generated code "

    Private Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form
Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents ListBox_ListaComputadores As
System.Windows.Forms.ListBox
    Friend WithEvents BtnListaComputadores As System.Windows.Forms.Button
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents btnSelecionarMaquina As System.Windows.Forms.Button
    Friend WithEvents BtnDesligar As System.Windows.Forms.Button
    Friend WithEvents BtnReiniciar As System.Windows.Forms.Button
    Friend WithEvents BtnLogOff As System.Windows.Forms.Button
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Dim resources As System.Resources.ResourceManager = New
System.Resources.ResourceManager(GetType(frmListaComputadores))
        Me.ListBox_ListaComputadores = New System.Windows.Forms.ListBox
        Me.BtnListaComputadores = New System.Windows.Forms.Button
        Me.Label1 = New System.Windows.Forms.Label
        Me.btnSelecionarMaquina = New System.Windows.Forms.Button
        Me.BtnDesligar = New System.Windows.Forms.Button
        Me.BtnReiniciar = New System.Windows.Forms.Button
        Me.BtnLogOff = New System.Windows.Forms.Button
        Me.SuspendLayout()
        '
        'ListBox_ListaComputadores
        '
        Me.ListBox_ListaComputadores.AccessibleDescription =
resources.GetString("ListBox_ListaComputadores.AccessibleDescription")
        Me.ListBox_ListaComputadores.AccessibleName =
resources.GetString("ListBox_ListaComputadores.AccessibleName")
        Me.ListBox_ListaComputadores.Anchor =
CType(resources.GetObject("ListBox_ListaComputadores.Anchor"),
System.Windows.Forms.AnchorStyles)
```

```
        Me.ListBox_ListaComputadores.BackgroundImage =
CType(resources.GetObject("ListBox_ListaComputadores.BackgroundImage"),
System.Drawing.Image)
        Me.ListBox_ListaComputadores.ColumnWidth =
CType(resources.GetObject("ListBox_ListaComputadores.ColumnWidth"),
Integer)
        Me.ListBox_ListaComputadores.Dock =
CType(resources.GetObject("ListBox_ListaComputadores.Dock"),
System.Windows.Forms.DockStyle)
        Me.ListBox_ListaComputadores.Enabled =
CType(resources.GetObject("ListBox_ListaComputadores.Enabled"), Boolean)
        Me.ListBox_ListaComputadores.Font =
CType(resources.GetObject("ListBox_ListaComputadores.Font"),
System.Drawing.Font)
        Me.ListBox_ListaComputadores.HorizontalExtent =
CType(resources.GetObject("ListBox_ListaComputadores.HorizontalExtent"),
Integer)
        Me.ListBox_ListaComputadores.HorizontalScrollbar =
CType(resources.GetObject("ListBox_ListaComputadores.HorizontalScrollbar"
), Boolean)
        Me.ListBox_ListaComputadores.ImeMode =
CType(resources.GetObject("ListBox_ListaComputadores.ImeMode"),
System.Windows.Forms.ImeMode)
        Me.ListBox_ListaComputadores.IntegralHeight =
CType(resources.GetObject("ListBox_ListaComputadores.IntegralHeight"),
Boolean)
        Me.ListBox_ListaComputadores.ItemHeight =
CType(resources.GetObject("ListBox_ListaComputadores.ItemHeight"),
Integer)
        Me.ListBox_ListaComputadores.Location =
CType(resources.GetObject("ListBox_ListaComputadores.Location"),
System.Drawing.Point)
        Me.ListBox_ListaComputadores.Name = "ListBox_ListaComputadores"
        Me.ListBox_ListaComputadores.RightToLeft =
CType(resources.GetObject("ListBox_ListaComputadores.RightToLeft"),
System.Windows.Forms.RightToLeft)
        Me.ListBox_ListaComputadores.ScrollAlwaysVisible =
CType(resources.GetObject("ListBox_ListaComputadores.ScrollAlwaysVisible"
), Boolean)
        Me.ListBox_ListaComputadores.SelectionMode =
System.Windows.Forms.SelectionMode.MultiSimple
        Me.ListBox_ListaComputadores.Size =
CType(resources.GetObject("ListBox_ListaComputadores.Size"),
System.Drawing.Size)
        Me.ListBox_ListaComputadores.TabIndex =
CType(resources.GetObject("ListBox_ListaComputadores.TabIndex"), Integer)
        Me.ListBox_ListaComputadores.Visible =
CType(resources.GetObject("ListBox_ListaComputadores.Visible"), Boolean)
        '
        'BtnListaComputadores
        '
        Me.BtnListaComputadores.AccessibleDescription =
resources.GetString("BtnListaComputadores.AccessibleDescription")
        Me.BtnListaComputadores.AccessibleName =
resources.GetString("BtnListaComputadores.AccessibleName")
        Me.BtnListaComputadores.Anchor =
CType(resources.GetObject("BtnListaComputadores.Anchor"),
System.Windows.Forms.AnchorStyles)
        Me.BtnListaComputadores.BackgroundImage =
CType(resources.GetObject("BtnListaComputadores.BackgroundImage"),
System.Drawing.Image)
```

```
        Me.BtnListaComputadores.Dock =
CType(resources.GetObject("BtnListaComputadores.Dock"),
System.Windows.Forms.DockStyle)
        Me.BtnListaComputadores.Enabled =
CType(resources.GetObject("BtnListaComputadores.Enabled"), Boolean)
        Me.BtnListaComputadores.FlatStyle =
CType(resources.GetObject("BtnListaComputadores.FlatStyle"),
System.Windows.Forms.FlatStyle)
        Me.BtnListaComputadores.Font =
CType(resources.GetObject("BtnListaComputadores.Font"),
System.Drawing.Font)
        Me.BtnListaComputadores.Image =
CType(resources.GetObject("BtnListaComputadores.Image"),
System.Drawing.Image)
        Me.BtnListaComputadores.ImageAlign =
CType(resources.GetObject("BtnListaComputadores.ImageAlign"),
System.Drawing.ContentAlignment)
        Me.BtnListaComputadores.ImageIndex =
CType(resources.GetObject("BtnListaComputadores.ImageIndex"), Integer)
        Me.BtnListaComputadores.ImeMode =
CType(resources.GetObject("BtnListaComputadores.ImeMode"),
System.Windows.Forms.ImeMode)
        Me.BtnListaComputadores.Location =
CType(resources.GetObject("BtnListaComputadores.Location"),
System.Drawing.Point)
        Me.BtnListaComputadores.Name = "BtnListaComputadores"
        Me.BtnListaComputadores.RightToLeft =
CType(resources.GetObject("BtnListaComputadores.RightToLeft"),
System.Windows.Forms.RightToLeft)
        Me.BtnListaComputadores.Size =
CType(resources.GetObject("BtnListaComputadores.Size"),
System.Drawing.Size)
        Me.BtnListaComputadores.TabIndex =
CType(resources.GetObject("BtnListaComputadores.TabIndex"), Integer)
        Me.BtnListaComputadores.Text =
resources.GetString("BtnListaComputadores.Text")
        Me.BtnListaComputadores.TextAlign =
CType(resources.GetObject("BtnListaComputadores.TextAlign"),
System.Drawing.ContentAlignment)
        Me.BtnListaComputadores.Visible =
CType(resources.GetObject("BtnListaComputadores.Visible"), Boolean)
        '
        'Label1
        '
        Me.Label1.AccessibleDescription =
resources.GetString("Label1.AccessibleDescription")
        Me.Label1.AccessibleName =
resources.GetString("Label1.AccessibleName")
        Me.Label1.Anchor = CType(resources.GetObject("Label1.Anchor"),
System.Windows.Forms.AnchorStyles)
        Me.Label1.AutoSize =
CType(resources.GetObject("Label1.AutoSize"), Boolean)
        Me.Label1.Dock = CType(resources.GetObject("Label1.Dock"),
System.Windows.Forms.DockStyle)
        Me.Label1.Enabled = CType(resources.GetObject("Label1.Enabled"),
Boolean)
        Me.Label1.Font = CType(resources.GetObject("Label1.Font"),
System.Drawing.Font)
        Me.Label1.Image = CType(resources.GetObject("Label1.Image"),
System.Drawing.Image)
```

```
        Me.Label1.ImageAlign =
CType(resources.GetObject("Label1.ImageAlign"),
System.Drawing.ContentAlignment)
        Me.Label1.ImageIndex =
CType(resources.GetObject("Label1.ImageIndex"), Integer)
        Me.Label1.ImeMode = CType(resources.GetObject("Label1.ImeMode"),
System.Windows.Forms.ImeMode)
        Me.Label1.Location =
CType(resources.GetObject("Label1.Location"), System.Drawing.Point)
        Me.Label1.Name = "Label1"
        Me.Label1.RightToLeft =
CType(resources.GetObject("Label1.RightToLeft"),
System.Windows.Forms.RightToLeft)
        Me.Label1.Size = CType(resources.GetObject("Label1.Size"),
System.Drawing.Size)
        Me.Label1.TabIndex =
CType(resources.GetObject("Label1.TabIndex"), Integer)
        Me.Label1.Text = resources.GetString("Label1.Text")
        Me.Label1.TextAlign =
CType(resources.GetObject("Label1.TextAlign"),
System.Drawing.ContentAlignment)
        Me.Label1.Visible = CType(resources.GetObject("Label1.Visible"),
Boolean)
        '
        'btnSelecionarMaquina
        '
        Me.btnSelecionarMaquina.AccessibleDescription =
resources.GetString("btnSelecionarMaquina.AccessibleDescription")
        Me.btnSelecionarMaquina.AccessibleName =
resources.GetString("btnSelecionarMaquina.AccessibleName")
        Me.btnSelecionarMaquina.Anchor =
CType(resources.GetObject("btnSelecionarMaquina.Anchor"),
System.Windows.Forms.AnchorStyles)
        Me.btnSelecionarMaquina.BackgroundImage =
CType(resources.GetObject("btnSelecionarMaquina.BackgroundImage"),
System.Drawing.Image)
        Me.btnSelecionarMaquina.Dock =
CType(resources.GetObject("btnSelecionarMaquina.Dock"),
System.Windows.Forms.DockStyle)
        Me.btnSelecionarMaquina.Enabled =
CType(resources.GetObject("btnSelecionarMaquina.Enabled"), Boolean)
        Me.btnSelecionarMaquina.FlatStyle =
CType(resources.GetObject("btnSelecionarMaquina.FlatStyle"),
System.Windows.Forms.FlatStyle)
        Me.btnSelecionarMaquina.Font =
CType(resources.GetObject("btnSelecionarMaquina.Font"),
System.Drawing.Font)
        Me.btnSelecionarMaquina.Image =
CType(resources.GetObject("btnSelecionarMaquina.Image"),
System.Drawing.Image)
        Me.btnSelecionarMaquina.ImageAlign =
CType(resources.GetObject("btnSelecionarMaquina.ImageAlign"),
System.Drawing.ContentAlignment)
        Me.btnSelecionarMaquina.ImageIndex =
CType(resources.GetObject("btnSelecionarMaquina.ImageIndex"), Integer)
        Me.btnSelecionarMaquina.ImeMode =
CType(resources.GetObject("btnSelecionarMaquina.ImeMode"),
System.Windows.Forms.ImeMode)
        Me.btnSelecionarMaquina.Location =
CType(resources.GetObject("btnSelecionarMaquina.Location"),
System.Drawing.Point)
        Me.btnSelecionarMaquina.Name = "btnSelecionarMaquina"
```

```
        Me.btnSelecionarMaquina.RightToLeft =
CType(resources.GetObject("btnSelecionarMaquina.RightToLeft"),
System.Windows.Forms.RightToLeft)
        Me.btnSelecionarMaquina.Size =
CType(resources.GetObject("btnSelecionarMaquina.Size"),
System.Drawing.Size)
        Me.btnSelecionarMaquina.TabIndex =
CType(resources.GetObject("btnSelecionarMaquina.TabIndex"), Integer)
        Me.btnSelecionarMaquina.Text =
resources.GetString("btnSelecionarMaquina.Text")
        Me.btnSelecionarMaquina.TextAlign =
CType(resources.GetObject("btnSelecionarMaquina.TextAlign"),
System.Drawing.ContentAlignment)
        Me.btnSelecionarMaquina.Visible =
CType(resources.GetObject("btnSelecionarMaquina.Visible"), Boolean)
        '
        'BtnDesligar
        '
        Me.BtnDesligar.AccessibleDescription =
resources.GetString("BtnDesligar.AccessibleDescription")
        Me.BtnDesligar.AccessibleName =
resources.GetString("BtnDesligar.AccessibleName")
        Me.BtnDesligar.Anchor =
CType(resources.GetObject("BtnDesligar.Anchor"),
System.Windows.Forms.AnchorStyles)
        Me.BtnDesligar.BackgroundImage =
CType(resources.GetObject("BtnDesligar.BackgroundImage"),
System.Drawing.Image)
        Me.BtnDesligar.Dock =
CType(resources.GetObject("BtnDesligar.Dock"),
System.Windows.Forms.DockStyle)
        Me.BtnDesligar.Enabled =
CType(resources.GetObject("BtnDesligar.Enabled"), Boolean)
        Me.BtnDesligar.FlatStyle =
CType(resources.GetObject("BtnDesligar.FlatStyle"),
System.Windows.Forms.FlatStyle)
        Me.BtnDesligar.Font =
CType(resources.GetObject("BtnDesligar.Font"), System.Drawing.Font)
        Me.BtnDesligar.Image =
CType(resources.GetObject("BtnDesligar.Image"), System.Drawing.Image)
        Me.BtnDesligar.ImageAlign =
CType(resources.GetObject("BtnDesligar.ImageAlign"),
System.Drawing.ContentAlignment)
        Me.BtnDesligar.ImageIndex =
CType(resources.GetObject("BtnDesligar.ImageIndex"), Integer)
        Me.BtnDesligar.ImeMode =
CType(resources.GetObject("BtnDesligar.ImeMode"),
System.Windows.Forms.ImeMode)
        Me.BtnDesligar.Location =
CType(resources.GetObject("BtnDesligar.Location"), System.Drawing.Point)
        Me.BtnDesligar.Name = "BtnDesligar"
        Me.BtnDesligar.RightToLeft =
CType(resources.GetObject("BtnDesligar.RightToLeft"),
System.Windows.Forms.RightToLeft)
        Me.BtnDesligar.Size =
CType(resources.GetObject("BtnDesligar.Size"), System.Drawing.Size)
        Me.BtnDesligar.TabIndex =
CType(resources.GetObject("BtnDesligar.TabIndex"), Integer)
        Me.BtnDesligar.Text = resources.GetString("BtnDesligar.Text")
        Me.BtnDesligar.TextAlign =
CType(resources.GetObject("BtnDesligar.TextAlign"),
System.Drawing.ContentAlignment)
```

```
        Me.BtnDesligar.Visible =
CType(resources.GetObject("BtnDesligar.Visible"), Boolean)
        '
        'BtnReiniciar
        '
        Me.BtnReiniciar.AccessibleDescription =
resources.GetString("BtnReiniciar.AccessibleDescription")
        Me.BtnReiniciar.AccessibleName =
resources.GetString("BtnReiniciar.AccessibleName")
        Me.BtnReiniciar.Anchor =
CType(resources.GetObject("BtnReiniciar.Anchor"),
System.Windows.Forms.AnchorStyles)
        Me.BtnReiniciar.BackgroundImage =
CType(resources.GetObject("BtnReiniciar.BackgroundImage"),
System.Drawing.Image)
        Me.BtnReiniciar.Dock =
CType(resources.GetObject("BtnReiniciar.Dock"),
System.Windows.Forms.DockStyle)
        Me.BtnReiniciar.Enabled =
CType(resources.GetObject("BtnReiniciar.Enabled"), Boolean)
        Me.BtnReiniciar.FlatStyle =
CType(resources.GetObject("BtnReiniciar.FlatStyle"),
System.Windows.Forms.FlatStyle)
        Me.BtnReiniciar.Font =
CType(resources.GetObject("BtnReiniciar.Font"), System.Drawing.Font)
        Me.BtnReiniciar.Image =
CType(resources.GetObject("BtnReiniciar.Image"), System.Drawing.Image)
        Me.BtnReiniciar.ImageAlign =
CType(resources.GetObject("BtnReiniciar.ImageAlign"),
System.Drawing.ContentAlignment)
        Me.BtnReiniciar.ImageIndex =
CType(resources.GetObject("BtnReiniciar.ImageIndex"), Integer)
        Me.BtnReiniciar.ImeMode =
CType(resources.GetObject("BtnReiniciar.ImeMode"),
System.Windows.Forms.ImeMode)
        Me.BtnReiniciar.Location =
CType(resources.GetObject("BtnReiniciar.Location"), System.Drawing.Point)
        Me.BtnReiniciar.Name = "BtnReiniciar"
        Me.BtnReiniciar.RightToLeft =
CType(resources.GetObject("BtnReiniciar.RightToLeft"),
System.Windows.Forms.RightToLeft)
        Me.BtnReiniciar.Size =
CType(resources.GetObject("BtnReiniciar.Size"), System.Drawing.Size)
        Me.BtnReiniciar.TabIndex =
CType(resources.GetObject("BtnReiniciar.TabIndex"), Integer)
        Me.BtnReiniciar.Text = resources.GetString("BtnReiniciar.Text")
        Me.BtnReiniciar.TextAlign =
CType(resources.GetObject("BtnReiniciar.TextAlign"),
System.Drawing.ContentAlignment)
        Me.BtnReiniciar.Visible =
CType(resources.GetObject("BtnReiniciar.Visible"), Boolean)
        '
        'BtnLogOff
        '
        Me.BtnLogOff.AccessibleDescription =
resources.GetString("BtnLogOff.AccessibleDescription")
        Me.BtnLogOff.AccessibleName =
resources.GetString("BtnLogOff.AccessibleName")
        Me.BtnLogOff.Anchor =
CType(resources.GetObject("BtnLogOff.Anchor"),
System.Windows.Forms.AnchorStyles)
```

```
        Me.BtnLogOff.BackgroundImage =
CType(resources.GetObject("BtnLogOff.BackgroundImage"),
System.Drawing.Image)
        Me.BtnLogOff.Dock = CType(resources.GetObject("BtnLogOff.Dock"),
System.Windows.Forms.DockStyle)
        Me.BtnLogOff.Enabled =
CType(resources.GetObject("BtnLogOff.Enabled"), Boolean)
        Me.BtnLogOff.FlatStyle =
CType(resources.GetObject("BtnLogOff.FlatStyle"),
System.Windows.Forms.FlatStyle)
        Me.BtnLogOff.Font = CType(resources.GetObject("BtnLogOff.Font"),
System.Drawing.Font)
        Me.BtnLogOff.Image =
CType(resources.GetObject("BtnLogOff.Image"), System.Drawing.Image)
        Me.BtnLogOff.ImageAlign =
CType(resources.GetObject("BtnLogOff.ImageAlign"),
System.Drawing.ContentAlignment)
        Me.BtnLogOff.ImageIndex =
CType(resources.GetObject("BtnLogOff.ImageIndex"), Integer)
        Me.BtnLogOff.ImeMode =
CType(resources.GetObject("BtnLogOff.ImeMode"),
System.Windows.Forms.ImeMode)
        Me.BtnLogOff.Location =
CType(resources.GetObject("BtnLogOff.Location"), System.Drawing.Point)
        Me.BtnLogOff.Name = "BtnLogOff"
        Me.BtnLogOff.RightToLeft =
CType(resources.GetObject("BtnLogOff.RightToLeft"),
System.Windows.Forms.RightToLeft)
        Me.BtnLogOff.Size = CType(resources.GetObject("BtnLogOff.Size"),
System.Drawing.Size)
        Me.BtnLogOff.TabIndex =
CType(resources.GetObject("BtnLogOff.TabIndex"), Integer)
        Me.BtnLogOff.Text = resources.GetString("BtnLogOff.Text")
        Me.BtnLogOff.TextAlign =
CType(resources.GetObject("BtnLogOff.TextAlign"),
System.Drawing.ContentAlignment)
        Me.BtnLogOff.Visible =
CType(resources.GetObject("BtnLogOff.Visible"), Boolean)
        '
        'frmListaComputadores
        '
        Me.AccessibleDescription =
resources.GetString("$this.AccessibleDescription")
        Me.AccessibleName = resources.GetString("$this.AccessibleName")
        Me.AutoScaleBaseSize =
CType(resources.GetObject("$this.AutoScaleBaseSize"),
System.Drawing.Size)
        Me.AutoScroll = CType(resources.GetObject("$this.AutoScroll"),
Boolean)
        Me.AutoScrollMargin =
CType(resources.GetObject("$this.AutoScrollMargin"), System.Drawing.Size)
        Me.AutoScrollMinSize =
CType(resources.GetObject("$this.AutoScrollMinSize"),
System.Drawing.Size)
        Me.BackgroundImage =
CType(resources.GetObject("$this.BackgroundImage"), System.Drawing.Image)
        Me.ClientSize = CType(resources.GetObject("$this.ClientSize"),
System.Drawing.Size)
        Me.Controls.Add(Me.BtnLogOff)
        Me.Controls.Add(Me.BtnReiniciar)
        Me.Controls.Add(Me.BtnDesligar)
        Me.Controls.Add(Me.btnSelecionarMaquina)
```

```
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.BtnListaComputadores)
        Me.Controls.Add(Me.ListBox_ListaComputadores)
        Me.Enabled = CType(resources.GetObject("$this.Enabled"), Boolean)
        Me.Font = CType(resources.GetObject("$this.Font"),
System.Drawing.Font)
        Me.Icon = CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
        Me.ImeMode = CType(resources.GetObject("$this.ImeMode"),
System.Windows.Forms.ImeMode)
        Me.Location = CType(resources.GetObject("$this.Location"),
System.Drawing.Point)
        Me.MaximumSize = CType(resources.GetObject("$this.MaximumSize"),
System.Drawing.Size)
        Me.MinimumSize = CType(resources.GetObject("$this.MinimumSize"),
System.Drawing.Size)
        Me.Name = "frmListaComputadores"
        Me.RightToLeft = CType(resources.GetObject("$this.RightToLeft"),
System.Windows.Forms.RightToLeft)
        Me.StartPosition =
CType(resources.GetObject("$this.StartPosition"),
System.Windows.Forms.FormStartPosition)
        Me.Text = resources.GetString("$this.Text")
        Me.ResumeLayout(False)

    End Sub

#End Region

    Public Shared Function InstanciarFormulario() As frmListaComputadores

        If (formulario Is Nothing) Then
            formulario = New frmListaComputadores
        End If

        If (formulario.IsDisposed) Then
            formulario = New frmListaComputadores
        End If

        Return formulario
    End Function

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnListaComputadores.Click
        Dim res As New System.Xml.XmlDocument
        Dim noFilho, noFilho2 As Xml.XmlNode
        Dim elemento As Xml.XmlElement
        Dim processo As New Process
        Dim arquivoEmUso As Boolean

        System.IO.File.Delete((Application.StartupPath &
"\..\Resultados\ListaComputadores.xml"))

        processo.StartInfo.FileName = Application.StartupPath &
"\..\Scripts\ListaComputadores2.vbs"
        processo.Start()
        arquivoEmUso = True

        Do Until (System.IO.File.Exists(Application.StartupPath &
"\..\Resultados\ListaComputadores.xml"))
            System.Threading.Thread.Sleep(5000)
        Loop
```

```
        ' Fica nesse loop até que o arquivo não esteja mais sendo usado,
isto é,
        ' somente fica acessível quando todas as inforamções já foram
salvas
        While arquivoEmUso
            Try
                res.Load(Application.StartupPath &
"\..\Resultados\ListaComputadores.xml")
                arquivoEmUso = False
                elemento = res.DocumentElement()
            Catch ex As Exception
                System.Threading.Thread.Sleep(2000)
            End Try
        End While

        ListBox_ListaComputadores.Items.Clear()
        ListBox_ListaComputadores.Sorted = True

        For Each noFilho In elemento.ChildNodes
            ListBox_ListaComputadores.Items.Add(noFilho.InnerText)
        Next
    End Sub

    Private Sub btnSelecionarMaquina_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnSelecionarMaquina.Click
        Dim nomes As String = ""

        For Each nome As Object In
ListBox_ListaComputadores.SelectedItems
            nomes &= nome & ";"
        Next

        clsAuxiliar.Maquinas = (nomes.Substring(0, nomes.Length -
1)).Split(";"c)

        For Each str As String In clsAuxiliar.Maquinas
            MessageBox.Show(str)
        Next

    End Sub

    Private Sub ListBox_ListaComputadores_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
ListBox_ListaComputadores.SelectedIndexChanged

    End Sub

    Private Sub frmListaComputadores_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

    End Sub

    Private Sub BtnDesligar_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnDesligar.Click
        Dim processo As New Process

        For Each nomeMaquina As String In clsAuxiliar.Maquinas
            processo.StartInfo.FileName = Application.StartupPath &
"\..\Scripts\DesligaComputadores.vbs"
            processo.StartInfo.Arguments = nomeMaquina
            processo.Start()
```

```
        Next
    End Sub

    Private Sub BtnReiniciar_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnReiniciar.Click
        Dim processo As New Process

        For Each nomeMaquina As String In clsAuxiliar.Maquinas
            processo.StartInfo.FileName = Application.StartupPath &
"\..\Scripts\ReiniciaComputadores.vbs"
            processo.StartInfo.Arguments = nomeMaquina
            processo.Start()
        Next
    End Sub

    Private Sub BtnLogOff_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnLogOff.Click
        Dim processo As New Process

        For Each nomeMaquina As String In clsAuxiliar.Maquinas
            processo.StartInfo.FileName = Application.StartupPath &
"\..\Scripts\Logoff.vbs"

            processo.StartInfo.Arguments = nomeMaquina
            processo.Start()
        Next

    End Sub

End Class
```

clsauxiliar.vb

```
'*************************************************************************
'Nome: clsauxiliar.vb
'Autor: Henrique Rodrigues
'Descricao: Classe criada para armazenar os nomes dos computadores
'selecionados
'*************************************************************************
Public Class clsAuxiliar

    Private Shared listaMaquinas() As String

    Private Sub New()
    End Sub

    Public Shared Property Maquinas() As String()
        Get
            Return listaMaquinas
        End Get

        Set(ByVal Value As String())
            listaMaquinas = Value
        End Set

    End Property

End Class
```

ListaComputadores.VBS

```
'**********************************************************************
'Nome: ListaComputadores.vbs
'Autor: Henrique Rodrigues
'Descricao: Script utilizando WMI que conecta no servico de diretórios do
'Windows, obtem a listagem dos conputadores existentes no domínio pré-
'selecionado e gera um arquivo XML com esses nomes para posterior
'utilizacao pela ferramenta HRNet.
'**********************************************************************

ON ERROR RESUME NEXT

Dim infoDominio
Set oFSO = CreateObject("Scripting.FileSystemObject")
Set ArquivoSaida =
oFSO.CreateTextFile("..\Resultados\ListaComputadores.xml", True)
ArquivoSaida.WriteLine ("<?xml version=""1.0"" encoding=""iso-8859-15""
?><Resultado>")

infoDominio = InputBox("Entre com as informações do domínio", "HRNet
Gerenciamento")

Const ADS_SCOPE_SUBTREE = 2
Set objConnection = CreateObject("ADODB.Connection")
Set objCommand =   CreateObject("ADODB.Command")
objConnection.Provider = "ADsDSOObject"
objConnection.Open "Active Directory Provider"
Set objCOmmand.ActiveConnection = objConnection
objCommand.CommandText = _
    "Select Name from 'LDAP://" & infoDominio & "' " _
        & "where objectClass='computer'"
objCommand.Properties("Page Size") = 1000
objCommand.Properties("Timeout") = 30
objCommand.Properties("Searchscope") = ADS_SCOPE_SUBTREE
objCommand.Properties("Cache Results") = False
Set objRecordSet = objCommand.Execute

' Se aconteceu um erro, garante que o arquivo XML será fechado
' e sai da aplicação para evitar um loop eterno
if (Err.Number <> 0) then
    WScript.Echo "Errorcode: " & _
         Err.Number & " = " & Err.Description

    ArquivoSaida.WriteLine ("</Resultado>")
    ArquivoSaida.Close

    WScript.quit

end if

objRecordSet.MoveFirst

Do Until objRecordSet.EOF
    ArquivoSaida.WriteLine ("<Computador>" &
objRecordSet.Fields("Name").Value & "</Computador>")
    objRecordSet.MoveNext
Loop

ArquivoSaida.WriteLine ("</Resultado>")

ArquivoSaida.Close
```

DesligaComputadores.VBS

```
'**********************************************************************
'Nome: DesligaComputadores.vbs
'Autor: Henrique Rodrigues
'Descricao: Script utilizando WMI para desligar os computadores
'**********************************************************************

Const val = 5
On Error Resume Next

dim strComputerShutDown

strComputerShutDown = wscript.arguments(0)

  Set objWMIService = GetObject("winmgmts:" _
  & "{impersonationLevel=impersonate,(Shutdown)}!\\" &
strComputerShutDown & "\root\cimv2")
  Set colOperatingSystems = objWMIService.ExecQuery _
  ("Select * from Win32_OperatingSystem")

  For Each objOperatingSystem in colOperatingSystems
    ObjOperatingSystem.Win32Shutdown(val)
        Next
```

ReiniciaComputadores.VBS

```
'**********************************************************************
'Nome: ReiniciaComputadores.vbs
'Autor: Henrique Rodrigues
'Descricao: Script utilizando WMI para reiniciar os computadores
'**********************************************************************

Const val = 6
On Error Resume Next

dim strComputerShutDown

strComputerShutDown = wscript.arguments(0)

  Set objWMIService = GetObject("winmgmts:" _
  & "{impersonationLevel=impersonate,(Shutdown)}!\\" &
strComputerShutDown & "\root\cimv2")
  Set colOperatingSystems = objWMIService.ExecQuery _
  ("Select * from Win32_OperatingSystem")

  For Each objOperatingSystem in colOperatingSystems
    ObjOperatingSystem.Win32Shutdown(val)
  Next
```

Logoff.VBS

```
'**********************************************************************
'Nome: Logoff.vbs
'Autor: Henrique Rodrigues
'Descricao: Script utilizando WMI executar logoff nos computadores
'**********************************************************************


Const val = 4
On Error Resume Next
```

```
dim strComputerShutDown

strComputerShutDown = wscript.arguments(0)

  Set objWMIService = GetObject("winmgmts:" _
  & "{impersonationLevel=impersonate,(Shutdown)}!\\" &
strComputerShutDown & "\root\cimv2")
  Set colOperatingSystems = objWMIService.ExecQuery _
  ("Select * from Win32_OperatingSystem")

  For Each objOperatingSystem in colOperatingSystems
    ObjOperatingSystem.Win32Shutdown(val)
  Next
```

frmListaSoftwares.vb

```
'**************************************************************************
'Nome: frmListaSoftwares.vb
'Autor: Henrique Rodrigues
'Descricao: Formulario de inventario dos softwares
'**************************************************************************


Public Class frmListaSoftwares
    Inherits System.Windows.Forms.Form

    Private Shared formulario As frmListaSoftwares

#Region " Windows Form Designer generated code "

    Private Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form
Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents ListBox_ListaSoftwares As
System.Windows.Forms.ListBox
    Friend WithEvents BtnListaSoftwares As System.Windows.Forms.Button
```

```
    Friend WithEvents Label1 As System.Windows.Forms.Label
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.ListBox_ListaSoftwares = New System.Windows.Forms.ListBox
        Me.BtnListaSoftwares = New System.Windows.Forms.Button
        Me.Label1 = New System.Windows.Forms.Label
        Me.SuspendLayout()
        '
        'ListBox_ListaSoftwares
        '
        Me.ListBox_ListaSoftwares.Anchor =
CType((((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Bottom) _
                    Or System.Windows.Forms.AnchorStyles.Left) _
                    Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
        Me.ListBox_ListaSoftwares.Location = New System.Drawing.Point(16,
40)
        Me.ListBox_ListaSoftwares.Name = "ListBox_ListaSoftwares"
        Me.ListBox_ListaSoftwares.Size = New System.Drawing.Size(248,
225)
        Me.ListBox_ListaSoftwares.TabIndex = 0
        '
        'BtnListaSoftwares
        '
        Me.BtnListaSoftwares.Anchor =
CType((System.Windows.Forms.AnchorStyles.Bottom Or
System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
        Me.BtnListaSoftwares.Location = New System.Drawing.Point(152,
272)
        Me.BtnListaSoftwares.Name = "BtnListaSoftwares"
        Me.BtnListaSoftwares.Size = New System.Drawing.Size(112, 23)
        Me.BtnListaSoftwares.TabIndex = 1
        Me.BtnListaSoftwares.Text = "Listar Softwares"
        '
        'Label1
        '
        Me.Label1.Location = New System.Drawing.Point(16, 24)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(100, 16)
        Me.Label1.TabIndex = 2
        Me.Label1.Text = "Softwares:"
        '
        'frmListaSoftwares
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(6, 14)
        Me.ClientSize = New System.Drawing.Size(292, 318)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.BtnListaSoftwares)
        Me.Controls.Add(Me.ListBox_ListaSoftwares)
        Me.Font = New System.Drawing.Font("Verdana", 8.25!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.Name = "frmListaSoftwares"
        Me.Text = "Softwares Instalados"
        Me.ResumeLayout(False)

    End Sub

#End Region
```

```
    Public Shared Function InstanciarFormulario() As frmListaSoftwares

        If (formulario Is Nothing) Then
            formulario = New frmListaSoftwares
        End If

        If (formulario.IsDisposed) Then
            formulario = New frmListaSoftwares
        End If

        Return formulario
    End Function

    Private Sub BtnListaSoftwares_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BtnListaSoftwares.Click
        Dim res As New System.Xml.XmlDocument
        Dim noFilho, noFilho2 As Xml.XmlNode
        Dim elemento As Xml.XmlElement
        Dim processo As New Process
        Dim arquivoEmUso As Boolean


        ListBox_ListaSoftwares.Items.Clear()
        'ListBox_ListaSoftwares.Sorted = True

        For Each nomeMaquina As String In clsAuxiliar.Maquinas

            ListBox_ListaSoftwares.Items.Add("")

ListBox_ListaSoftwares.Items.Add("**************************************
*************")
            ListBox_ListaSoftwares.Items.Add("**********    " &
nomeMaquina & "    ***********")

ListBox_ListaSoftwares.Items.Add("**************************************
*************")
            ListBox_ListaSoftwares.Items.Add("")

            ' Deleta o arquivo para não pegar uma versão desatualizada
            System.IO.File.Delete((Application.StartupPath &
"\..\Resultados\ListaSoftwares.xml"))

            processo.StartInfo.FileName = Application.StartupPath &
"\..\Scripts\ListaSoftwares2.vbs"
            processo.StartInfo.Arguments = nomeMaquina
            processo.Start()
            arquivoEmUso = True

            Do Until (System.IO.File.Exists(Application.StartupPath &
"\..\Resultados\ListaSoftwares.xml"))
                System.Threading.Thread.Sleep(5000)
            Loop

            ' Fica nesse loop até que o arquivo não esteja mais sendo
usado, isto é,
            ' somente fica acessível quando todas as inforamções já foram
salvas
            While arquivoEmUso
                Try
                    res.Load(Application.StartupPath &
"\..\Resultados\ListaSoftwares.xml")
                    arquivoEmUso = False
```

```
                    elemento = res.DocumentElement()
                Catch ex As Exception
                    System.Threading.Thread.Sleep(2000)
                End Try
            End While

            For Each noFilho In elemento.ChildNodes
                For Each noFilho2 In noFilho.ChildNodes
                    ListBox_ListaSoftwares.Items.Add(noFilho2.InnerText)
                Next
            Next

        Next
    End Sub

    Private Sub frmListaSoftwares_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

    End Sub
End Class
```

ListaSoftwares.VBS

```
'********************************************************************
'Nome: ListaSoftwares.vbs
'Autor: Henrique Rodrigues
'Descricao: Script utilizando WMI para obter a listagem dos softwares de
'um computador local ou remoto e geracao de um arquivo XML com os dados
'obtidos para posterior utilizacao pela ferramenta HRNet.
'********************************************************************

dim strComputer

' Se aconteceu algum erro, mostra para o usuário o erro e continua
executando todos os passos,
' desta forma o arquivo será concluído e o handle liberado.

ON ERROR RESUME NEXT

strComputer = WScript.Arguments(0)

if (TRIM(strComputer) = "") THEN
      strComputer = "."
END IF

Set oFSO = CreateObject("Scripting.FileSystemObject")
Set ArquivoSaida =
oFSO.CreateTextFile("..\Resultados\ListaSoftwares.xml", True)
Set objWMIService = GetObject("winmgmts:" _
    & "{impersonationLevel=impersonate}!\\" & strComputer &
"\root\cimv2")

if (Err.Number <> 0) then
    WScript.Echo "Errorcode: " & _
         Err.Number & " = " & Err.Description

end if

ArquivoSaida.WriteLine ("<?xml version=""1.0"" encoding=""iso-8859-15""
?><Resultado>")
```

```
Set colSettings = objWMIService.ExecQuery("Select * from Win32_Product")

if (Err.Number <> 0) then
    WScript.Echo "Errorcode: " & _
         Err.Number & " = " & Err.Description

end if

For each objItem in colSettings

ArquivoSaida.WriteLine ("<Software><Nome>" & objItem.Caption &
"</Nome></Software>")
Next

ArquivoSaida.WriteLine ("</Resultado>")

ArquivoSaida.Close
```

frmServicos.vb

```
'************************************************************************
'Nome: frmServicos.vb
'Autor: Henrique Rodrigues
'Descricao: Formulario dos seguintes servicos: Invetario de Hardware,
'Lista Serviços, Para Servico, Inicia Serviço.
'************************************************************************
Public Class frmServicos
    Inherits System.Windows.Forms.Form

    Private Shared formulario As frmServicos

#Region " Windows Form Designer generated code "

    Private Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form
Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents Button1 As System.Windows.Forms.Button
```

```
    Friend WithEvents ListBox1 As System.Windows.Forms.ListBox
    Friend WithEvents Button2 As System.Windows.Forms.Button
    Friend WithEvents ListBox2 As System.Windows.Forms.ListBox
    Friend WithEvents Button4 As System.Windows.Forms.Button
    Friend WithEvents Button5 As System.Windows.Forms.Button
    Friend WithEvents Button6 As System.Windows.Forms.Button
    Friend WithEvents trView As System.Windows.Forms.TreeView
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.Button1 = New System.Windows.Forms.Button
        Me.ListBox1 = New System.Windows.Forms.ListBox
        Me.Button2 = New System.Windows.Forms.Button
        Me.ListBox2 = New System.Windows.Forms.ListBox
        Me.Button4 = New System.Windows.Forms.Button
        Me.Button5 = New System.Windows.Forms.Button
        Me.Button6 = New System.Windows.Forms.Button
        Me.trView = New System.Windows.Forms.TreeView
        Me.SuspendLayout()
        '
        'Button1
        '
        Me.Button1.Location = New System.Drawing.Point(16, 8)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New System.Drawing.Size(128, 23)
        Me.Button1.TabIndex = 0
        Me.Button1.Text = "Inventário"
        '
        'ListBox1
        '
        Me.ListBox1.Anchor =
CType(((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Left) _
                    Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
        Me.ListBox1.Location = New System.Drawing.Point(16, 40)
        Me.ListBox1.Name = "ListBox1"
        Me.ListBox1.Size = New System.Drawing.Size(696, 186)
        Me.ListBox1.TabIndex = 1
        '
        'Button2
        '
        Me.Button2.Location = New System.Drawing.Point(16, 240)
        Me.Button2.Name = "Button2"
        Me.Button2.Size = New System.Drawing.Size(128, 23)
        Me.Button2.TabIndex = 2
        Me.Button2.Text = "Listar Serviços"
        '
        'ListBox2
        '
        Me.ListBox2.Anchor =
CType(((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Bottom) _
                    Or System.Windows.Forms.AnchorStyles.Left),
System.Windows.Forms.AnchorStyles)
        Me.ListBox2.Location = New System.Drawing.Point(16, 272)
        Me.ListBox2.Name = "ListBox2"
        Me.ListBox2.Size = New System.Drawing.Size(336, 225)
        Me.ListBox2.TabIndex = 3
        '
        'Button4
        '
        Me.Button4.Anchor = System.Windows.Forms.AnchorStyles.Bottom
```

```
        Me.Button4.Location = New System.Drawing.Point(624, 240)
        Me.Button4.Name = "Button4"
        Me.Button4.Size = New System.Drawing.Size(88, 24)
        Me.Button4.TabIndex = 6
        Me.Button4.Text = "Nome"
        Me.Button4.Visible = False
        '
        'Button5
        '
        Me.Button5.Location = New System.Drawing.Point(368, 240)
        Me.Button5.Name = "Button5"
        Me.Button5.Size = New System.Drawing.Size(88, 23)
        Me.Button5.TabIndex = 7
        Me.Button5.Text = "Iniciar Serviço"
        '
        'Button6
        '
        Me.Button6.Location = New System.Drawing.Point(464, 240)
        Me.Button6.Name = "Button6"
        Me.Button6.Size = New System.Drawing.Size(88, 23)
        Me.Button6.TabIndex = 8
        Me.Button6.Text = "Parar Serviço"
        '
        'trView
        '
        Me.trView.Anchor = CType((((System.Windows.Forms.AnchorStyles.Top
Or System.Windows.Forms.AnchorStyles.Bottom) _
                    Or System.Windows.Forms.AnchorStyles.Left) _
                    Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
        Me.trView.ImageIndex = -1
        Me.trView.Location = New System.Drawing.Point(368, 272)
        Me.trView.Name = "trView"
        Me.trView.SelectedImageIndex = -1
        Me.trView.Size = New System.Drawing.Size(336, 232)
        Me.trView.TabIndex = 9
        '
        'frmServicos
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.ClientSize = New System.Drawing.Size(728, 517)
        Me.Controls.Add(Me.trView)
        Me.Controls.Add(Me.Button6)
        Me.Controls.Add(Me.Button5)
        Me.Controls.Add(Me.Button4)
        Me.Controls.Add(Me.ListBox2)
        Me.Controls.Add(Me.Button2)
        Me.Controls.Add(Me.ListBox1)
        Me.Controls.Add(Me.Button1)
        Me.Name = "frmServicos"
        Me.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Inventário e Serviços"
        Me.ResumeLayout(False)

    End Sub

#End Region

    Public Shared Function InstanciarFormulario() As frmServicos

        If (formulario Is Nothing) Then
```

```
            formulario = New frmServicos
        End If

        If (formulario.IsDisposed) Then
            formulario = New frmServicos
        End If

        Return formulario
    End Function

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim res As New System.Xml.XmlDocument
        Dim noFilho, noFilho2 As Xml.XmlNode
        Dim elemento As Xml.XmlElement
        Dim processo As New Process
        Dim arquivoEmUso As Boolean

        ListBox1.Items.Clear()

        For Each nomeMaquina As String In clsAuxiliar.Maquinas

            ListBox1.Items.Add("")

ListBox1.Items.Add("**************************************************"
)
            ListBox1.Items.Add("************    " & nomeMaquina & "
*************")

ListBox1.Items.Add("**************************************************"
)
            ListBox1.Items.Add("")

            System.IO.File.Delete((Application.StartupPath &
"\..\Resultados\Inventario.xml"))

            processo.StartInfo.FileName = Application.StartupPath &
"\..\Scripts\Inventario.vbs"
            processo.StartInfo.Arguments = nomeMaquina
            processo.Start()
            arquivoEmUso = True

            Do Until (System.IO.File.Exists(Application.StartupPath &
"\..\Resultados\Inventario.xml"))
                System.Threading.Thread.Sleep(5000)
            Loop

            ' Fica nesse loop até que o arquivo não esteja mais sendo
usado, isto é,
            ' somente fica acessível quando todas as inforamções já foram
salvas
            While arquivoEmUso
                Try
                    res.Load(Application.StartupPath &
"\..\Resultados\Inventario.xml")
                    arquivoEmUso = False
                    elemento = res.DocumentElement()

                Catch ex As Exception
                    System.Threading.Thread.Sleep(2000)
                End Try
            End While
```

```
            For Each noFilho In elemento.ChildNodes
                For Each noFilho2 In noFilho.ChildNodes
                    ListBox1.Items.Add(noFilho2.Name & ": " &
noFilho2.InnerText)
                Next
                ListBox1.Items.Add("")
            Next
        Next
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        ListarServicos()
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
        MsgBox(trView.SelectedNode.Text)
    End Sub

    Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button5.Click
        Dim processo As New Process
        'Dim res As New System.Xml.XmlDocument
        'Dim noFilho, noFilho2 As Xml.XmlNode
        'Dim elemento As Xml.XmlElement
        'Dim noArvore As TreeNode

        processo.StartInfo.FileName = Application.StartupPath &
"\..\Scripts\IniciaServico.vbs"
        processo.StartInfo.Arguments = """" & trView.SelectedNode.Text &
""""
        processo.Start()

        System.Threading.Thread.Sleep(5000)

        ListarServicos()


    End Sub

    Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button6.Click
        Dim processo As New Process

        processo.StartInfo.FileName = Application.StartupPath &
"\..\Scripts\ParaServico.vbs"
        processo.StartInfo.Arguments = """" & trView.SelectedNode.Text &
""""
        processo.Start()

        System.Threading.Thread.Sleep(5000)

        ListarServicos()
    End Sub

    Private Sub ListarServicos()

        Dim res As New System.Xml.XmlDocument
        Dim noFilho, noFilho2 As Xml.XmlNode
        Dim elemento As Xml.XmlElement
```

```
        Dim processo As New Process
        Dim noArvore, noRaiz As TreeNode
        Dim arquivoEmUso As Boolean

        ListBox2.Items.Clear()
        trView.Nodes.Clear()

        For Each nomeMaquina As String In clsAuxiliar.Maquinas

            ListBox2.Items.Add("")

ListBox2.Items.Add("**************************************************"
)
            ListBox2.Items.Add("**********    " & nomeMaquina & "
**********")

ListBox2.Items.Add("**************************************************"
)
            ListBox2.Items.Add("")

            System.IO.File.Delete((Application.StartupPath &
"\..\Resultados\Servicos.xml"))

            processo.StartInfo.FileName = Application.StartupPath &
"\..\Scripts\ListaServicos2.vbs"
            processo.StartInfo.Arguments = nomeMaquina
            processo.Start()
            arquivoEmUso = True

            Do Until (System.IO.File.Exists(Application.StartupPath &
"\..\Resultados\Servicos.xml"))
                System.Threading.Thread.Sleep(5000)
            Loop

            ' Fica nesse loop até que o arquivo não esteja mais sendo
usado, isto é,
            ' somente fica acessível quando todas as inforamções já foram
salvas
            While arquivoEmUso
                Try
                    res.Load(Application.StartupPath &
"\..\Resultados\Servicos.xml")
                    arquivoEmUso = False
                    elemento = res.DocumentElement()

                Catch ex As Exception
                    System.Threading.Thread.Sleep(2000)
                End Try
            End While

            For Each noFilho In elemento.ChildNodes
                For Each noFilho2 In noFilho.ChildNodes
                    ListBox2.Items.Add(noFilho2.Name & ": " &
noFilho2.InnerText)
                Next
                ListBox2.Items.Add("")
            Next

            noRaiz = New TreeNode(nomeMaquina)
            trView.Nodes.Add(noRaiz)

            For Each noFilho In elemento.ChildNodes
```

```
                ' Pega o nome do serviço que é o primeiro elemento filho
                noArvore =
noRaiz.Nodes.Add(noFilho.FirstChild().InnerText)
                For Each noFilho2 In noFilho.ChildNodes
                    If (noFilho2.Name <> "Nome") Then
                        noArvore.Nodes.Add(noFilho2.Name & ": " &
noFilho2.InnerText)
                    End If
                Next
            Next

        Next

    End Sub

    Private Sub ListBox2_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ListBox2.SelectedIndexChanged

    End Sub

    Private Sub ListBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ListBox1.SelectedIndexChanged

    End Sub

    Private Sub trView_AfterSelect(ByVal sender As System.Object, ByVal e
As System.Windows.Forms.TreeViewEventArgs)

    End Sub

    Private Sub frmServicos_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load

    End Sub
End Class
```

ListaServicos.VBS

```
'**********************************************************************
'Nome: ListaServicos.vbs
'Autor: Henrique Rodrigues
'Descricao: Script utilizando WMI para obter a listagem dos servicos e
'seus estados de um computador remoto e geração de um arquivo XML para
'posterior utilizacao pela ferramenta HRNet.
'**********************************************************************
dim strComputer

' Se aconteceu algum erro, mostra para o usuário o erro e continua
executando todos os passos,
' desta forma o arquivo será concluído e o handle liberado.
ON ERROR RESUME NEXT

'strComputer = InputBox("Entre com o nome da máquina", "HR.NET")
strComputer = WScript.Arguments(0)
'msgbox(strComputer)

if (TRIM(strComputer) = "") THEN
      strComputer = "."
END IF
```

```
Set oFSO = CreateObject("Scripting.FileSystemObject")
Set ArquivoSaida = oFSO.CreateTextFile("..\Resultados\Servicos.xml",
True)
Set objWMIService = GetObject("winmgmts:" _
    & "{impersonationLevel=impersonate}!\\" & strComputer &
"\root\cimv2")

if (Err.Number <> 0) then
    WScript.Echo "Errorcode: " & _
         Err.Number & " = " & Err.Description

end if

ArquivoSaida.WriteLine ("<?xml version=""1.0"" encoding=""iso-8859-15""
?><Resultado>")

Set colSettings = objWMIService.ExecQuery("Select * from Win32_Service")

if (Err.Number <> 0) then
    WScript.Echo "Errorcode: " & _
         Err.Number & " = " & Err.Description

end if

For each objItem in colSettings

ArquivoSaida.WriteLine ("<Servico><Nome>" & objItem.DisplayName &
"</Nome>" & _
            "<Estado>" & objItem.State & "</Estado>" & _
            "<Status>" & objItem.Status & "</Status>" & _
            "<Inicializacao>" & objItem.StartMode & "</Inicializacao>" &
_
            "<Usuario>" & objItem.StartName & "</Usuario>" & _
            "</Servico>")
Next

ArquivoSaida.WriteLine ("</Resultado>")

ArquivoSaida.Close
```

Inventario.VBS:

```
'**********************************************************************
'Nome: Inventario.vbs
'Autor: Henrique Rodrigues
'Descricao: Script utilizando WMI para obter o inventario de hardware dos
'conputadores romotos e geracao de um arquivo XML com os dados obtidos
'para posterior utilizacao pela ferramenta HRNet.
'**********************************************************************

dim strComputer

' Se aconteceu algum erro, mostra para o usuário o erro e continua
executando todos os passos,
' desta forma o arquivo será concluído e o handle liberado.
ON ERROR RESUME NEXT

strComputer = Wscript.Arguments(0)
```

```
if (TRIM(strComputer) = "") THEN
      strComputer = "."
END IF

Set oFSO = CreateObject("Scripting.FileSystemObject")
Set ArquivoSaida = oFSO.CreateTextFile("..\Resultados\Inventario.xml",
True)
Set objWMIService = GetObject("winmgmts:" _
    & "{impersonationLevel=impersonate}!\\" & strComputer &
"\root\cimv2")

ArquivoSaida.WriteLine ("<?xml version=""1.0"" encoding=""iso-8859-15""
?><Resultado><Computador>")

Set colSettings = objWMIService.ExecQuery("Select * from
Win32_ComputerSystem")

For each objItem in colSettings
      sysNome = objItem.Caption
      sysFabricante = objItem.Manufacturer
      sysModelo = objItem.Model
      sysNumProcessadores = objItem.NumberofProcessors
      sysMemoriaFisica = Round((objItem.TotalPhysicalMemory / 1024) /
1024) & " MB"
Next

            ArquivoSaida.WriteLine ("<Nome>" & sysNome & "</Nome>" & _
            "<Fabricante>" & sysFabricante & "</Fabricante>"& _
            "<Modelo>" & sysModelo & "</Modelo>" & _
            "<NumProcessadores>" & sysNumProcessadores &
"</NumProcessadores>")

Set colSettings = objWMIService.ExecQuery("Select * from
Win32_Processor")

For each objItem in colSettings
            ArquivoSaida.WriteLine ("<Processadores>" & objItem.Name &
"</Processadores>")
Next

ArquivoSaida.WriteLine ("<MemoriaFisica>" & sysMemoriaFisica &
"</MemoriaFisica></Computador>")

Set colSettings = objWMIService.ExecQuery("Select * from
Win32_DiskDrive")

For each objItem in colSettings

            ArquivoSaida.WriteLine ("<HardDisk><HD>"&
objItem.InterfaceType & "</HD>" & _
            "<Modelo>" &  objItem.Caption & "</Modelo>" & _
            "<Capacidade>" & Round(objItem.Size / 1000000000) & "
GB</Capacidade>" & _
            "</HardDisk>")

Next

Set colSettings = objWMIService.ExecQuery("Select * from
Win32_NetworkAdapterConfiguration WHERE IPEnabled = True")

For each objItem in colSettings
```

```
            ArquivoSaida.WriteLine ("<PlacaRede><Rede>"&
objItem.Description & "</Rede>" & _
            "<MAC>" &  objItem.MACAddress & "</MAC>" & _
            "</PlacaRede>")

Next

Set colSettings = objWMIService.ExecQuery("Select * from
Win32_VideoController")

For each objItem in colSettings

            ArquivoSaida.WriteLine ("<PlacaVideo><Video>"&
objItem.Caption & " - " & ((objItem.AdapterRAM/1024)/1024) & "
MB</Video>" & _
            "</PlacaVideo>")

Next

Set colSettings = objWMIService.ExecQuery("Select * from
Win32_SoundDevice")

For each objItem in colSettings

            ArquivoSaida.WriteLine ("<PlacaSom><Som>"& objItem.Name &
"</Som>" & _
            "</PlacaSom>")

Next

ArquivoSaida.WriteLine ("</Resultado>")

      ArquivoSaida.Close
```

frmSobre.vb

```
'**************************************************************************
'Nome: frmSobre.vb
'Autor: Henrique Rodrigues
'Descricao: Formulario com informacoes sobre o autor.
'**************************************************************************
Public Class frmAbout
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
```

```
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form
Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents Button1 As System.Windows.Forms.Button
    Friend WithEvents Label3 As System.Windows.Forms.Label
    Friend WithEvents Label4 As System.Windows.Forms.Label
    Friend WithEvents Label5 As System.Windows.Forms.Label
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.Label1 = New System.Windows.Forms.Label
        Me.Label2 = New System.Windows.Forms.Label
        Me.Button1 = New System.Windows.Forms.Button
        Me.Label3 = New System.Windows.Forms.Label
        Me.Label4 = New System.Windows.Forms.Label
        Me.Label5 = New System.Windows.Forms.Label
        Me.SuspendLayout()
        '
        'Label1
        '
        Me.Label1.Font = New System.Drawing.Font("Verdana", 18.0!,
CType((System.Drawing.FontStyle.Bold Or
System.Drawing.FontStyle.Underline), System.Drawing.FontStyle),
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label1.Location = New System.Drawing.Point(8, 16)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(648, 32)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "HRNet - Gerenciamento"
        Me.Label1.TextAlign =
System.Drawing.ContentAlignment.MiddleCenter
        '
        'Label2
        '
        Me.Label2.Font = New System.Drawing.Font("Verdana", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.Label2.Location = New System.Drawing.Point(136, 248)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(384, 16)
        Me.Label2.TabIndex = 1
        Me.Label2.Text = "Aluno: Henrique Rodrigues da Silva"
        Me.Label2.TextAlign =
System.Drawing.ContentAlignment.MiddleCenter
        '
        'Button1
        '
        Me.Button1.Font = New System.Drawing.Font("Verdana", 8.25!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.Button1.Location = New System.Drawing.Point(288, 312)
        Me.Button1.Name = "Button1"
```

```
        Me.Button1.TabIndex = 2
        Me.Button1.Text = "Fechar"
        '
        'Label3
        '
        Me.Label3.Font = New System.Drawing.Font("Verdana", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.Label3.Location = New System.Drawing.Point(56, 72)
        Me.Label3.Name = "Label3"
        Me.Label3.Size = New System.Drawing.Size(532, 56)
        Me.Label3.TabIndex = 3
        Me.Label3.Text = "Ferramenta desenvolvida para validar o projeto
de graduação de Engenharia da Comp" & _
        "utação no UniCEUB"
        Me.Label3.TextAlign =
System.Drawing.ContentAlignment.MiddleCenter
        '
        'Label4
        '
        Me.Label4.Font = New System.Drawing.Font("Verdana", 14.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.Label4.Location = New System.Drawing.Point(128, 144)
        Me.Label4.Name = "Label4"
        Me.Label4.Size = New System.Drawing.Size(408, 88)
        Me.Label4.TabIndex = 4
        Me.Label4.Text = "Análise e Gerência de clientes e servidores de
uma rede computacional"
        Me.Label4.TextAlign =
System.Drawing.ContentAlignment.MiddleCenter
        '
        'Label5
        '
        Me.Label5.Font = New System.Drawing.Font("Verdana", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.Label5.Location = New System.Drawing.Point(136, 272)
        Me.Label5.Name = "Label5"
        Me.Label5.Size = New System.Drawing.Size(400, 23)
        Me.Label5.TabIndex = 5
        Me.Label5.Text = "Orientador: Fabiano Mariath"
        Me.Label5.TextAlign =
System.Drawing.ContentAlignment.MiddleCenter
        '
        'frmAbout
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.ClientSize = New System.Drawing.Size(656, 350)
        Me.Controls.Add(Me.Label5)
        Me.Controls.Add(Me.Label4)
        Me.Controls.Add(Me.Label3)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.Label1)
        Me.Name = "frmAbout"
        Me.Text = "Sobre"
        Me.ResumeLayout(False)
    End Sub
#End Region
```

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Me.Close()
    End Sub

End Class
```

Apêndice B – Entrevistas

Por questão de sigilo os dados de sua empresa não serão exibidos.

Srs. Administradores,

Estou realizando uma pesquisa para o projeto final, Análise e Gerenciamento Remoto de Clientes e Servidores em uma Rede Computacional, como requisito parcial para a obtenção do título de Engenheiro da Computação realizado no UniCEUB.

No intuito de obter dados referentes às necessidades existentes nos ambientes de rede atuais, gostaria de solicitar a cooperação dos senhores para responder algumas questões a seguir:

Nome:

E-mail:

Empresa:

1. Quantos computadores aproximadamente existem na rede de sua empresa?

2. Existe atualmente em sua empresa, alguma ferramenta que faça inventário de hardware e software?

   ☐ Sim ☐ Não ☐ Não sei opinar

3. Você considera importante ter o inventário de hardware de sua rede atualizado?

   ☐ Sim ☐ Não ☐ Não sei opinar

4. Você considera importante ter o inventário de software de sua rede atualizado?

   ☐ Sim ☐ Não ☐ Não sei opinar

5. Quanto tempo aproximadamente é gasto para fazer o inventário de cada computador em sua rede?

6. Você considera necessária a existência de uma ferramenta que faça o processo de inventário de hardware e software de forma remota e automatizada?

   ☐ Sim ☐ Não ☐ Não sei opinar

7. Você acredita que com a existência de uma ferramenta de inventário de hardware e software, seria reduzido o tempo gasto na execução dessas tarefas?

   ☐ Sim ☐ Não ☐ Não sei opinar

8. Você acredita que uma ferramenta de inventário de hardware e software, possa disponibilizar informações que sirvam como suporte para novos investimentos no parque computacional?

   ☐ Sim ☐ Não ☐ Não sei opinar

9. Você considera importante ter uma ferramenta que possibilite desligar, reiniciar e/ou executar logoff em computadores remotos?

   ☐ Sim ☐ Não ☐ Não sei opinar

10. Você acredita que com a existência de uma ferramenta que possibilite desligar, reiniciar e/ou executar logoff em computadores remotos, seria reduzido o tempo gasto na execução dessas tarefas?

    ☐ Sim ☐ Não ☐ Não sei opinar

11. Você considera importante ter uma ferramenta que possibilite listar, parar ou iniciar serviços em computadores remotos?

☐ Sim ☐ Não ☐ Não sei opinar

12. Você acredita que com a existência de uma ferramenta que possibilite listar, parar ou iniciar serviços em computadores remotos, seria reduzido o tempo gasto na execução dessas tarefas?

☐ Sim ☐ Não ☐ Não sei opinar

This document was created with Win2PDF available at http://www.daneprairie.com.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.