AUTOMAÇÃO DE INVENTARIADO DE HARDWARE E REGISTRO DE USO

Aluno: David Mascarenhas Watkins – R.A.: 2031743/4

Orientador: MSc. Antônio José Gonçalves Pinto

Brasília – DF, Novembro de 2007

AUTOMAÇÃO DE INVENTARIADO DE HARDWARE E REGISTRO DE USO

por

David Mascarenhas Watkins

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

Brasília – DF, Novembro de 2007

Banca Examinadora

_____

Prof. MSc. Antônio José Gonçalves Pinto
Orientador

_____

Prof. MSc. Miguel Archanjo Jr.
Examinador

_____

Prof. MSc. Fabiano Mariath
Examinador

_____

Prof. MSc. Roberto Schaefer
Examinador

Agradecimentos

Ao meu pai, Jeffrey Micheal Watkins, que com seu amor pela família e dedicação ao trabalho sempre serviu como um modelo para mim.

A minha mãe, Rosita Mascarenhas Watkins, por sempre estar ao meu lado me dando apoio e por confiar em mim.

As minhas irmãs, Karla e Michelle Mascarenhas Watkins, pelo apoio que sempre me deram.

A todos os professores do curso de Engenharia da Computação, por terem me dado um base sólida de conhecimento.

Ao meu orientador Professor MSc. Antônio José Gonçalves Pinto, por me guiar no desenvolvimento desse projeto.

Ao Prof. MSc. Miguel Archanjo Jr., pela ajuda com o tema do trabalho e seus objetivos.

Ao meu primo Roberto Mascarenhas Braga, pelo apoio fundamental no desenvolvimento desse projeto e por sempre estar disposto a ajudar.

Aos meus colegas de turma, que fizeram da minha experiência universitária algo memorável.

Aos meus grandes amigos, em especial: Augusto Cavalcanti, Fabrício Vieira da Costa, Guilherme Perez, João Herman Sampaio Junior, Lucas Sampaio, Pedro Antunes, Shauan Leite e Tiago Rosa e Silva.

Sumário

LISTA DE FIGURAS ..... **VI**
LISTA DE TABELAS ..... **VII**
ÍNDICE DE SIGLAS E ABREVIATURAS ..... **VIII**
RESUMO ..... **X**
ABSTRACT ..... **XI**
1. CAPÍTULO 01 – INTRODUÇÃO **12**
1.1. MOTIVAÇÃO 12
1.2. OBJETIVOS 13
1.3. METODOLOGIA 14
1.4. ESTRUTURA DA MONOGRAFIA 15
2. CAPÍTULO 02 – GERENCIAMENTO DE REDES **17**
2.1. CONCEITOS GERAIS 17
2.2. FERRAMENTAS COMERCIALMENTE DISPONÍVEIS 18
2.3. TECNOLOGIAS PARA GERÊNCIA DE REDES 19
2.3.1. Secure Network Management Protocol *19*
2.3.2. Windows Management Instrumentation *22*
2.4. LINGUAGEM DE SCRIPT 25
2.4.1. C Sharp *26*
2.5. GERENCIADOR DE BANCOS DE DADOS COMPUTACIONAL 26
2.5.1. SQL Server 2005 ..... *27*
2.6. INTERFACE GRÁFICA DE USUÁRIO 28
2.6.1. ASP.NET *28*
3. CAPÍTULO 03 – INVENTARIADO AUTOMÁTICO DE HARDWARE **30**
3.1. INTRODUÇÃO 30
3.2. RAZÃO DO USO DA TECNOLOGIA ESCOLHIDA 31
3.2.1. Requisitos para execução do IAH *32*
3.3. FLUXO DE FUNCIONAMENTO 34
3.4. SCRIPT VISUAL C# 36
3.5. BANCO DE DADOS SQL SERVER 2005 40
3.6. INTERFACE WEB ASP.NET 42
3.7. PROBLEMAS E SOLUÇÕES ENCONTRADOS 46
3.8. APRESENTAÇÃO DO APLICATIVO 48
4. CAPÍTULO 04 – EXPERIMENTO E RESULTADOS **52**
4.1. INTRODUÇÃO 52
4.2. CONFIGURAÇÃO DO AMBIENTE DE TESTES 53

4.3. IDENTIFICAÇÃO DO PLANO DO EXPERIMENTO 55
4.4. CONFIGURAÇÃO DO AMBIENTE DO EXPERIMENTO 57
4.5. TESTES REALIZADOS 59
4.6. RESULTADOS OBTIDOS 60
5. CAPÍTULO 05 – CONCLUSÕES 63
5.1. SUGESTÕES PARA TRABALHOS FUTUROS 66
6. REFERÊNCIAS BIBLIOGRÁFICAS 68
7. APÊNDICE A – CÓDIGOS DE PROGRAMAÇÃO 71

LISTA DE FIGURAS

Figura 2.1 – O SNMP facilita a troca de informações entre ativos [CISCO, 2006] ……20
Figura 2.2 – Elementos de uma rede gerenciada por SNMP [CISCO, 2006]……...21
Figura 2.3 – Script usando WMI para identificar versão do *Microsoft Office* [MSDN, 2007b…… 22
Figura 2.4 – Relacionamento de componentes da arquitetura WMI [MSDN, 2007c]……..24
Figura 3.1 – Fluxo de aplicativo *Inventariado Automático de Hardware* (IAH)……33
Figura 3.2 – Início de um novo projeto no *Visual Studio 2005*…….... 35
Figura 3.3 – Adição de referência para o uso de WMI……36
Figura 3.4 – Variáveis globais usadas no *script*……37
Figura 3.5 – Métodos usados no *script*……...37
Figura 3.6 – Código do método *statscpu*……... 38
Figura 3.7 – Diagrama de classe do script criado……39
Figura 3.8 – Criação de um novo banco de dados……...40
Figura 3.9 – Nome de variáveis e tipo definidos em tabela de banco de dados……..41
Figura 3.10 – Criação de um novo *web site*……...42
Figura 3.11 – *Toolbox* e opções do sub-item *Login*……...43
Figura 3.12 – Seleção dos campos a serem usados……... 44
Figura 3.13 – *Login* de interface gráfica de usuário……..... 47
Figura 3.14 – Tela de opções…….. 48
Figura 3.15 – Relatório geral de uso por MAC……..48
Figura 3.16 – Detalhes de registro……...49
Figura 3.17 – Possíveis alterações de *hardware*……..50
Figura 4.1 – Configuração de rede de computador servidor……...53
Figura 4.2 – *Setup* do ambiente de experimento……..56
Figura 4.3 – Gráfico de quantidade de registros por duração de sessão……..60

LISTA DE TABELAS

Tabela 4.1 – Exemplo de dados apresentados no *Relatório Geral – Hardware*……..60

ÍNDICE DE SIGLAS E ABREVIATURAS

AR	Administrador de rede
C#	C Sharp
CIMOM	CIM Object Manager
DMBS	Database Management Systems
DMTF	Distributed Management Task Force
GUI	Graphical User Interface
HP	Hewlett-Packard
HTML	Hypertext Markup Language
IAH	Inventariado Automático de Hardware
IBM	International Business Machines
IETF	Internet Engineering Task Force
MAC	Media Access Control
MIBs	Management Information Base
NMSs	Network-management Systems
RAM	Ram Access Memory
SCOM	System Center Operation Manager
SNMP	Secure Network Management Protocol
SO	Sistema Operacional
SQL	Structured Query Language
SQLS	Microsoft SQL Server 2005
TCP/IP	Transmission Control Protocol/Internet Protocol
TI	Tecnologia da Informação
USB	Universal Serial Bus
VC#	Visual C Sharp
VS	Visual Studio

WBEM	Web-Based Enterprise Management
WC	WMI Consumer
WDM	Windows Driver Model
WI	WMI Infrastructure
WMI	Windows Management Instrumentation
WQL	Windows Query Language
WR	WMI Repository
WS	WMI Service
XML	Extensible Markup Language

RESUMO

Este trabalho apresenta um estudo que possibilita a criação de uma ferramenta com fins de administração de rede que gera remotamente, de maneira automatizada, um inventariado do parque computacional de computadores pessoais de uma dada rede, assim como informações de horários de uso de cada computador, permitindo que administradores de rede, através de uma interface gráfica de usuário, tenham fácil acesso a essas informações, auxiliando suas tarefas.

A ferramenta acima citada foi desenvolvida com o *Microsoft Visual Studio 2005* utilizando *Visual C#* para o desenvolvimento do *script*, ASP.NET para o desenvolvimento da interface gráfica de usuário e *Microsoft SQL Server 2005* para a criação do banco de dados responsável pelo armazenamento de dados coletados.

Palavras-chave: gerência de rede; automatização de tarefas; inventário de *hardware*.

ABSTRACT

This final paper presents a study that enables the creation of a network administration tool with the purpose of generating remotely and automatically a hardware inventory database of all personal computers in a given network as well as information regarding the usage of each individual computer allowing network administrators to have easy access to these information's through a graphical user interface.

The above mentioned tool was developed with *Microsoft Visual Studio 2005* using *Visual C#* for the development of the script, ASP.NET for the development of the graphical user interface and *Microsoft SQL Server 2005* for the creation of the database responsible for storing the collected data.

Key-Words: network managing; automatization of tasks; hardware inventory.

Capítulo 1 – Introdução

1.1 MOTIVAÇÃO

Nas últimas décadas, os avanços tecnológicos têm feito com que recursos computacionais se tornem mais acessíveis a todos. Com isso, empresas começaram a desenvolver parques computacionais e armazenarem seu bem mais valioso, a informação, em formato binário, não mais em papel. Esse crescimento exponencial do uso de computadores deu lugar a uma nova profissão: o administrador de rede (AR), profissional responsável por assegurar o bom funcionamento desses conglomerados de computadores.

As tarefas de um AR são as mais variadas e demandam muito esforço e tempo do mesmo. A proliferação permanente de recursos de tecnologia da informação (TI) leva a esforços administrativos altos, assim como à elevação constante no custo de manutenção de uma dada rede. Com essas premissas é possível verificar a necessidade de uma ferramenta que possa auxiliar um AR em suas tarefas que mais consomem tempo e fornecer uma visão mais profunda do uso de recursos, permitindo melhor planejamento estratégico.

Com o objetivo de agilizar e facilitar as tarefas executadas pelo AR, surgiu a necessidade de estudar o gerenciamento de redes com o intuito de desenvolver ferramentas que facilitem e agilizem as tarefas administrativas de forma eficaz.

1.2 OBJETIVOS

O trabalho desenvolvido tem como objetivo projetar um aplicativo administrativo que disponibilize informações de hardware e sessões de uso para suporte aos administradores de rede.

Os objetivos específicos do trabalho são:

a) Obter informações de *hardware* de computadores pessoais de uma dada rede, sempre que eles são ligados, gerando um inventariado dos mesmos automaticamente, incluindo as seguintes informações:

- Processador:
  - Fabricante; Família; Identificação; Número de série.
- Placa mãe:
  - Fabricante; Modelo; Número de série.
- Memória RAM:
  - Quantidade de pentes; Capacidade de memória *Random Access Memory* (RAM) total.
- Discos Rígidos:
  - Nome; Capacidade; Quantidade.
- Placa de rede:
  - Nome; Endereço *Media Access Control* (MAC).

- Teclado:
    - Tipo; Identificação.
- *Mouse*:
    - Tipo; Identificação.
- Monitor:
    - Tipo; Identificação; *Status*.
- *Drives* de CD:
    - Nome.

b) Verificar início e fim de cada sessão de uso de computador e disponibilizar essas informações ao AR, para que possa ser verificado o horário de uso de computadores em uma dada rede;

c) Desenvolver uma interface gráfica de usuário, para que o AR possa ter acesso às informações supracitadas com maior facilidade e agilidade;

1.3 METODOLOGIA

O desenvolvimento do projeto começou com uma ampla pesquisa bibliográfica para definir com quais tecnologias a implementação do projeto seria elaborada. Após isso, deu-se início à elaboração do aplicativo com o desenvolvimento do *script* responsável pela coleta de dados. Logo em seguida, foram criadas as tabelas para o armazenamento das informações em banco de dados e,

finalmente, foi desenvolvida uma interface gráfica de usuário para que a visualização dos dados obtidos fosse feita de maneira fácil e eficaz.

Terminada a fase de criação do aplicativo, deu-se início à fase de testes e experimentação. Foi utilizada uma rede de computadores montada especificamente com esse objetivo. Essa rede possibilitou verificar as alterações necessárias ao aplicativo para assegurar seu bom funcionamento.

Com os testes e experimentação do aplicativo finalizados, foi possível concluir a parte escrita da monografia utilizando os dados obtidos.

1.4 ESTRUTURA DA MONOGRAFIA

Este trabalho foi elaborado em cinco capítulos, incluindo a introdução. Segue uma breve descrição do que será exposto nos capítulos seguintes:

- *Capítulo 02 – Gerenciamento de Redes* - Traz diversos conceitos-chave acerca de gerenciamento de rede; informações sobre os principais produtos de gerenciamento de rede disponíveis no mercado; e informações sobre as tecnologias utilizadas no decorrer do desenvolvimento desse trabalho.
- *Capítulo 03 – Inventariado Automático de Hardware* – Elucida o desenvolvimento do aplicativo criado durante o projeto apresentando o porquê da tecnologia escolhida; os requisitos do aplicativo; seu fluxo

de funcionamento; problemas encontrados e a apresentação do aplicativo.

- *Capítulo 04 – Experimento e Resultados* - Mostra onde e como os testes e experimentação foram realizados e os resultados obtidos.
- *Capítulo 05 – Conclusões* - Apresenta as considerações finais do projeto com base nos resultados obtidos e sugestões para trabalhos futuros.

Capítulo 2 – Gerenciamento de Redes

2.1 CONCEITOS GERAIS

Um administrador de rede deve satisfazer as necessidades e requisitos do cliente, compreender a natureza do serviço prestado e os métodos adequados à administração de rede [SOUSA JR., 2005]. Essas três premissas sempre guiam um administrador de rede, seja em suas tarefas corriqueiras ou em seu planejamento de longo prazo.

Para obter êxito nos deveres supracitados, um administrador de rede conta com o auxílio de várias tecnologias e programas que permitem manter a rede disponível continuamente, com desempenho adequado, sustentado, estável e com controle e racionalização de custos [SOUSA JR., 2005]. Programas de gerenciamento de inventariado e ativos existem para eliminar o desconhecido. Eles incluem enormes bancos de dados para poder identificar correta e praticamente todos os variados tipos de *hardware* e *software* [GASKIN, 1999]. Grandes companhias multinacionais do ramo de TI já identificaram esse cenário no passado e desenvolveram ferramentas para suprir as necessidades do mesmo.

2.2 FERRAMENTAS COMERCIALMENTE DISPONÍVEIS

A seguir, serão descritos alguns exemplos de *softwares* de grandes empresas globais, responsáveis por gerenciamento de inventariado, entre outras funções:

- ***CiscoWorks Resource Manager Essentials***: Ferramenta de gerenciamento de rede projetada para diminuir chances de erros humano e eliminar várias tarefas manuais associadas com a manutenção de uma rede. Tem capacidade de gerenciamento de inventariado, gerenciamento de configuração de *devices* e serviço de auditoria de atualizações. Atualmente se encontra em sua versão 4.1 [CISCO, 2007].
- ***IBM Tivoli Configuration Manager***: Antigamente conhecido como *Tivoli Inventory*, essa ferramenta desenvolvida pela *International Business Machines* (IBM) inclui um módulo que permite que se faça uma varredura automática de um parque computacional, coletando informações de configuração de *hardware* e *software*. Essa ferramenta também auxilia um AR, pois permite impor adesão às regras de uma companhia mudando configurações de sistema, entre outras funcionalidades [IBM, 2007].
- ***HP OpenView Enterprise Discovery***: *Software* administrativo capaz de descobrir e inventariar automaticamente todos os ativos de TI de

uma empresa para que o AR possa saber exatamente o que está ocorrendo em sua rede. É também responsável por acompanhar de perto mudanças nas configurações de ativos de TI [HP, 2007].

- ***Novell ZENworks Asset Management***: Produto *add-on* do *Novell ZENworks Suíte* responsável por providenciar ao AR uma visão completa dos ativos de TI. Possui reconhecimento de *hardware* e *software*, além de fornecer informações vitais para a expansão da rede sobre o uso de ativos [NOVELL, 2007].

2.3 TECNOLOGIAS PARA GERÊNCIA DE REDES

2.3.1 Secure Network Management Protocol

O *Secure Network Management Protogol* (SNMP) foi introduzido pela primeira vez nos anos oitenta. O SNMP é o protocolo padrão de operações e manutenção para a internet e faz parte do *Transmission Control Protocol/Internet Protocol* (TCP/IP) tendo suas diretrizes definidas pela *Internet Engineering Task Force* (IETF) [SNMP, 2007] .

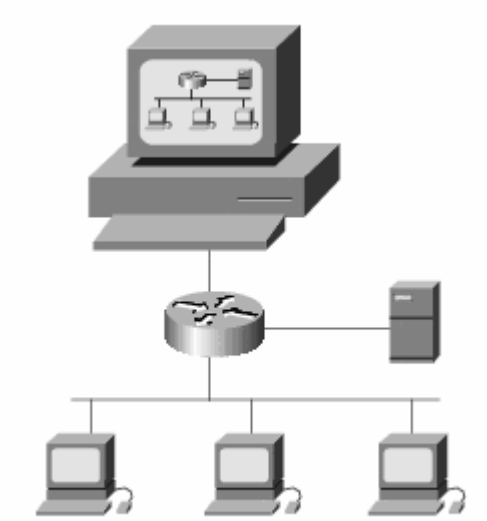


Figura 2.1 – O SNMP facilita a troca de informações entre ativos [CISCO, 2006].

O SNMP é um protocolo de nível de aplicação que facilita a troca de informações de gerenciamento entre variados ativos do parque computacional de uma empresa, como ilustra a figura 2.1. Tal protocolo permite que o administrador de rede gerencie- desempenho de rede, identifique e resolva problemas relacionados à ela e se planeje para futuras expansões [CISCO, 2006].

Uma rede gerenciada através de SNMP consiste de três componentes chave: ativos gerenciados, agentes e *Network-management Systems* (NMSs).

Um ativo gerenciado, também chamado de elementos de rede, é um nó da rede que contém um agente SNMP e que se encontra em uma rede gerenciada. Coletam e armazenam dados, além de disponibilizá-los aos NMSs através de SNMP. Também podem ser roteadores, *switches*, *hubs*, computadores ou impressoras [CISCO, 2006].

Um agente consiste de um módulo de *software* instalado em um ativo gerenciado. Ele possui conhecimento local de informações de gerenciamento e é

responsável por traduzir tais informações para um formato compatível com o SNMP. Agentes armazenam informações referentes a si próprio em *Management Information Base* (MIBs).

Finalmente, um NMSs executa uma aplicação que monitora e controla ativos gerenciados. Também é responsável por todo o processamento e memória necessários para o gerenciamento de rede. A figura 2.2 ilustra o relacionamento entre esses três componentes [CISCO, 2006].

Uma outra tecnologia existente de gerenciamento é o *Windows Management Instrumentation* (WMI), tecnologia desenvolvida pela *Microsoft*.

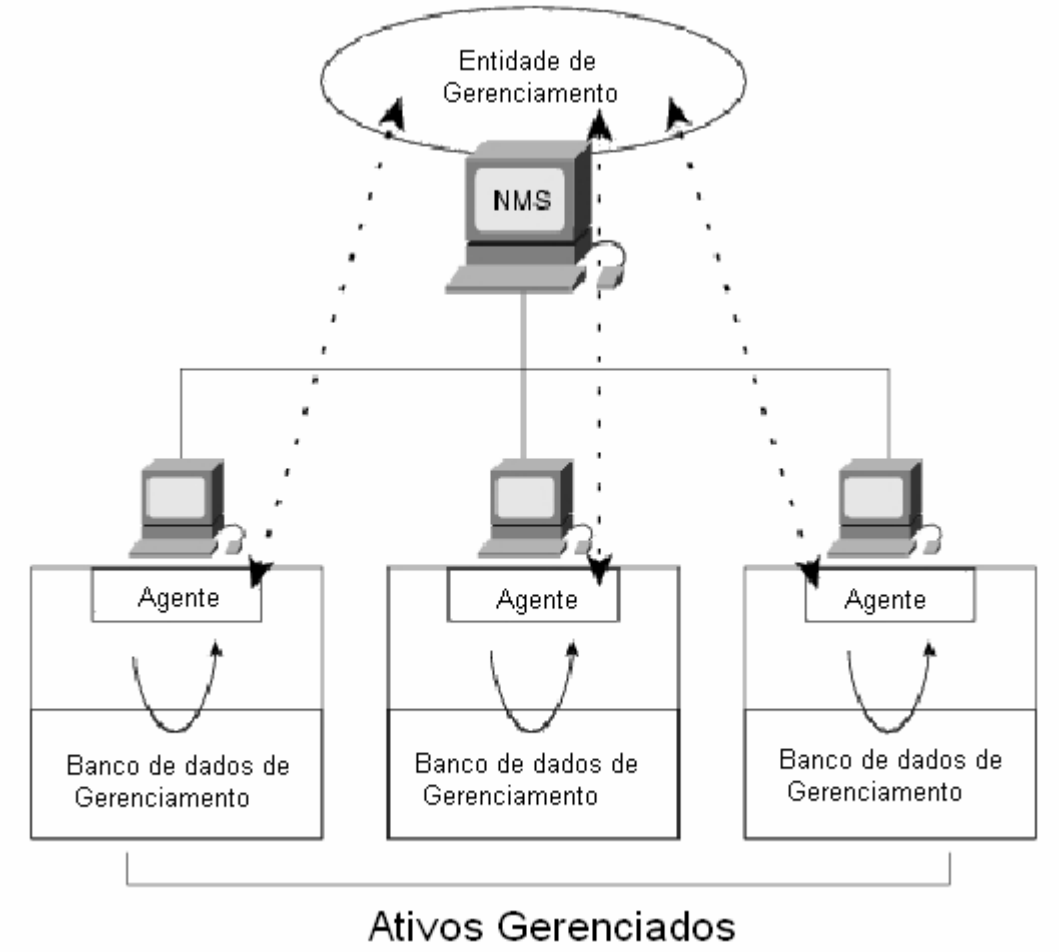


Figura 2.2 – Elementos de uma rede gerenciada por SNMP [CISCO, 2006].

2.3.2 Windows Management Instrumentation

O *Windows Management Instrumentation* (WMI), a implementação da *Microsoft* do *Web-Based Enterprise Management* (WBEM), desenvolvida pela *Distributed Management Task Force* (DMTF), é a infra-estrutura de gerenciamento de dados e operações para sistemas operacionais baseados em *Windows* [MICROSOFT, 2007a]. A infra-estrutura fornecida permite escrever *scripts* e aplicações para automatizar tarefas administrativas em computadores remotos e também fornecer dados para gerenciamento por parte de outros módulos de um dado sistema operacional (SO), como o *System Center Operation Manager* (SCOM) do *Windows* [MSDN, 2007a]. A figura 2.3 apresenta um código de *Visual Basic* usando WMI para obter informações a cerca da versão do *Microsoft Office* instalado em um computador.

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:" _
    & "{impersonationLevel=impersonate}!\\" _
    & strComputer & "\root\cimv2")
Set colSoftware = objWMIService.ExecQuery(_
    "Select * from Win32_Product " & _
    "Where IdentifyingNumber =" _
        & " '{90280409-6000-11D3-8CFE-0050048383C9}'")
For Each objItem in colSoftware
    Wscript.Echo "Name: " & objItem.Name
    Wscript.Echo "Version: " & objItem.Version
Next
```

Figura 2.3 – Script usando WMI para identificar versão do *Microsoft Office* [MSDN, 2007b].

Windows Query Language (WQL), subconjunto do *Structured Query Language* (SQL), é usado para fazer consultas ao WMI e permite verificação de

dados, eventos e esquemas. As consultas podem ser de dados, eventos ou esquemas. As instruções que permitem acesso a dados são semelhantes às cláusulas *insert, update* e *delete* do SQL.

A infra-estrutura do WMI é 100% compatível com as normas impostas pela DMTF. Existem quatro elementos chave: *Managed Objects, WMI Providers* (WP), *WMI Consumers* (WC) e *WMI Infrastructure* (WI).

Um *Managed Object* é um *hardware* ou *software* componente de sistema que é representado como uma instância de uma classe WMI. Ele pode ser um disco rígido, adaptador de rede, banco de dados, SO, processo ou serviço [MSDN. 2007d].

O *WMI Provider* fornece informações de gerenciamento sobe o *Managed Object* ao WMI, além de fazer o tratamento das mensagens do WMI ao *Managed Object*. Existe uma série de WP's previamente instalados em SO's *Windows* que contém classes com vários métodos, porém poucas propriedades. O *Windows NT* 4.0 contava com 15 *WMI Providers*, o *Windows 2000* com 29 WP's e atualmente o *Windows Vista* conta com quase 100 diferentes WP's [MSDN. 2007d].

Dois exemplos de *WMI Providers* são o *WDM Provider* e o *Win32 Provider*. O *WDM Provider* fornece informações de classe, instância, métodos e eventos de *drivers* de *hardware* com conformidade ao modelo *Windows Driver Model* (WDM). O *Win32 Provider* coleta e atualiza informações relevantes aos sistemas do *Windows*. Essas informações incluem atributos de um disco lógico e status de alguém periférico. O *Win32 Provider* funciona através de chamadas e *querys* ao sistema de registros do *Windows.*

WMI Consumer é um aplicativo de gerenciamento ou *script* que interage com a *WMI Infrastructure.* O usuário faz uma requisição de dados através do WC que

então enumera dados ou gera *querys.* O WC também e responsável pelo processamento da informação solicitada [MSDN. 2007d].

WMI Infrastructure é constituída de dois componentes principais: o *WMI Repository* (WR) e o *WMI Service* (WS).

- WR é um repositório de dados centralizado, gerenciado pelo *Common Information Model Object Manager* (CIMOM), um padrão aberto. CIMOM é um componente responsável por tratar a interação entre aplicativos de gerenciamento e os WP assegurando que dados serão armazenados de maneira uniforme independente de sua origem. Ele também dá permissão para acesso ao WR [MSDN. 2007d].
- Finalmente, o WS funciona como um intermediário entre os WP, o WR e os aplicativos de gerenciamento [MSDN. 2007d].

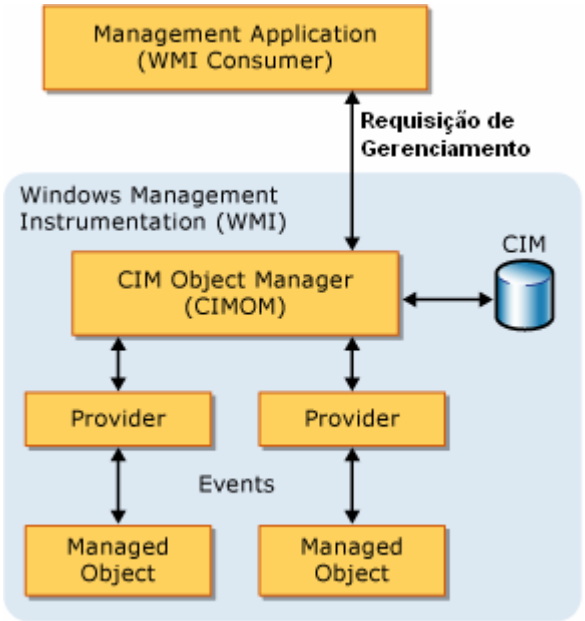


Figura 2.4 – Relacionamento de componentes da arquitetura WMI [MSDN, 2007c].

A figura 2.4 ilustra os relacionamentos entre um WC interagindo com o WR (que são gerenciados pelo CIMOM), os WP e os *Managed Objects*. O *Managed Object* tem seus dados de gerenciamento coletado pelo *WMI Provider*. Os dados são então armazenados em um *WMI Repository*, tendo seu gerenciamento feito por um CIMOM. Um *WMI Consumer* então faz a requisição de dados de gerenciamento ao *WMI Repository*. Toda as relações na estrutura do WMI são feitas através de *WMI Services.*

Windows Management Instrumentation Query Language (WQL) é um subconjunto do SQL com algumas mudanças de semântica. Ele é um tipo de linguagem de *query* dedicada ao WMI e desenhada para executar *querys* no repositório CIM com o objetivo de obter informações ou notificações de eventos.

2.4 LINGUAGEM DE SCRIPT

Linguagens de s*cript* são linguagens de programação usadas para controlar aplicações. *Scripts* são executados diretamente do seu código fonte, que normalmente são arquivos de texto que contém uma linguagem *markup* específica. Com isso, *scripts* são tratados como um programa distinto que executa de maneira independente de qualquer outra aplicação.

2.4.1 C Sharp

C Sharp (C#) é uma linguagem orientada a objetos, regulamentada pela *European Computer Manufacturer* Association (ECMA). C# teve seu desenvolvimento baseado em C e inclui aspectos de várias outras linguagens, como *Delphi* e *Java*, com ênfase em simplificação de linguagem, em ser moderna e em atender a propósitos-gerais. Um dos objetivos dela é promover suporte aos princípios de engenharia de *software* [ECMA, 2006].

Visual C Sharp (VC#) é a implementação da *Microsoft* da linguagem C# e faz parte do *Microsoft Visual Studio* (VS). Ele é baseado nas especificações ECMA/ISO da linguagem C# e foi desenhado para o desenvolvimento de uma ampla variedade de aplicações que rodam no *.NET Framework*. VC# inclui um editor de código completo, modelos de projetos, ferramentas de *design* e assistentes de códigos, entre outras funções [MSDN, 2007f].

VC# pode ser usado para coletar grandes quantidades de dados através do uso de WMI. Se existir uma necessidade desses dados serem armazenados, a melhor solução é através de um gerenciador de banco de dados computacional.

2.5 GERENCIADOR DE BANCO DE DADOS COMPUTACIONAL

Bancos de dados computacionais são coleções de dados estruturados que são armazenados em um computador. Esses dados são utilizados por programas ou

por usuários através de linguagens de *query* e são gerenciados por sistemas chamados de *Database Management Systems* (DBMS).

Um DBMS é composto por quatro itens-chave. Uma linguagem de modelagem é responsável por definir o esquema em que os bancos de dados serão armazenados de acordo com o modelo de dados do DBMS. Ele também deve ter estruturas de dados, composta por campos, arquivos e objetos, otimizadas para lidar com grandes quantidades de informações. Não só isso, existe também uma linguagem de *query* que permite que o administrador de banco de dados obtenha dados armazenados, faça análise deles e atualize campos de forma interativa. Finalmente, existe também um mecanismo de transação responsável por manter a integridade de dados [GEORGE, DELANO. 2006].

2.5.1 SQL Server 2005

Microsoft SQL Server 2005 (SQLS) é a implementação ANSI/ISO de linguagem de *query* estruturada da *Microsoft*. SQLS é responsável pelo gerenciamento dos maiores bancos de dados existentes, além de ser o DBMS mais usado no mundo. Isso se deve a sua alta disponibilidade, ferramentas de gerenciamento avançadas, tecnologia de segurança de ponta e alta escalabilidade. SQLS também é capaz de executar serviços de análise em seus bancos de dados, além de fornecer serviços de integração e relatório [MICROSOFT, 2007b].

Um banco de dados repleto de informações será extremamente valioso se um administrador puder ter fácil acesso e visualização clara dos dados inseridos nele. A melhor forma de se prover isso é através de uma interface gráfica de usuário.

2.6 INTERFACE GRÁFICA DE USUÁRIO

Interfaces gráficas de usuário, ou *Graphical User Interface* (GUI), são tipos de interface responsáveis pelo fornecimento de meios que facilitem a interação com computadores e dispositivos controlados por computadores. Elementos gráficos como ícones ou indicadores visuais são utilizados para facilitar a interação usuário/máquina e podem ser baseados em *prompts*, janelas, páginas *web* ou formulários.

2.5.1 ASP.NET

ASP.NET é uma tecnologia aberta desenvolvida pela *Microsoft* responsável pelo desenvolvimento de aplicativos *web* que permite usuários a construir *websites* dinâmicos, aplicações *web* e serviços *web* do tipo *Extensible Markup Language* (XML). Ele pode ser aplicado através do *Visual Studio*, software pago de desenvolvimento, ou através da versão gratuita *Visual Studio Express.* ASP.NET

pode ser usado para criar de pequenos sites até grandes aplicativos empresariais e funciona usando a plataforma .NET *framework* [MSDN, 2007g].

A .NET *framework* é executada sobre uma *Common Language Runtime* (CLR), ambiente de execução que independe de linguagem, através de uma interação com coletâneas de bibliotecas. Essas bibliotecas compõem o .NET *framework* [MSDN, 2007f].

Capítulo 3 – Inventariado Automático de Hardware

3. DESENVOLVIMENTO DO APLICATIVO

3.1 INTRODUÇÃO

O uso de “tecnologias e ferramentas para assegurar serviço de rede sob determinada condições ou restrições” é uma das principais definições de administração e gerência de redes [SOUSA JR., 2005, pg. 07]. Nos últimos anos, com os recursos computacionais se tornando mais acessíveis a cada dia, houve um grande crescimento de redes de computadores. Grandes ambientes corporativos implementaram enormes redes de computadores. Surgiu então a necessidade de se administrar esse ativo. AR’s são responsáveis por vários itens diferentes dentro de uma rede, por isso programas e aplicativos são usados para auxiliar suas tarefas.

O aplicativo desenvolvido neste capítulo, chamado de *Inventariado Automático de Hardware* (IAH) tem o intuito de auxiliar o AR em uma tarefa que, dependendo do tamanho da rede sob sua responsabilidade, pode consumir várias horas de trabalho. Através de *script*, banco de dados e interface gráfica de usuário, o AR terá disponibilizado a ele o inventariado de seu parque computacional, sendo que esse inventariado é atualizado todas as vezes que um computador é ligado. Não só isso, o AR terá acesso à informações sobre o horário de uso de cada computador.

Vale lembrar que o aplicativo desenvolvido não é uma solução completa para o AR e também não tem fins comerciais. O projeto visa a demonstrar que as tecnologias escolhidas para o desenvolvimento do mesmo, possibilitam a criação de tal solução e que os objetivos propostos podem ser atingidos através dessas tecnologias. Esse capítulo abordará os seguintes itens:

- O porquê determinadas tecnologias usadas foram escolhidas;
- Requisitos computacionais necessários para o desenvolvimento do aplicativo;
- Demonstração do fluxo de funcionamento do aplicativo;
- Desenvolvimento do *script* usando *Visual C;.*
- Desenvolvimento do banco de dados usando *SQL Server 2005;*
- Desenvolvimento da interface gráfica usando ASP.NET;
- Problemas e soluções encontrados, durante o desenvolvimento;
- Apresentação do aplicativo.

3.2 RAZÃO DO USO DA TECNOLOGIA ESCOLHIDA

WMI tem como objetivo prover um gerenciamento efetivo de computadores e servidores em redes corporativas permitindo monitoramento e controle desses ativos de maneira local e remota [MICROSOFT, 2007a]. Essas premissas se encaixam nos objetivos propostos permitindo que essa tecnologia auxilie o desenvolvimento do aplicativo.

Uma linguagem de script que pode ser utilizada para aplicação do WMI é o C# através do *Visual C#* (VC#). VC# pode ser usado através da versão *Visual C# 2005 Express Edition* ou através da solução completa da *Microsoft*, o *Visual Studio 2005*. Esse programa, *Visual Studio 2005*, foi escolhido pelo graduado para o desenvolvimento do *script* por ele possuir dele uma cópia autêntica.

Uma vez o *script* concluído, os dados obtidos devem ser armazenados em um banco de dados, portanto um DBMS deve ser utilizado. Devido às facilidades de integração disponíveis no *Visual Studio 2005* com o DBMS *SQL Server 2005* e o fato do graduado ter tido experiência prévia com esse DBMS em uma cadeira de seu curso, o DBMS *SQL Server 2005* foi escolhido.

Finalmente, um método para se desenvolver uma interface gráfica de usuário precisou ser definido. Como o *Visual Studio 2005* pode ser usado para implementar ASP.NET, essa ferramenta foi escolhida para o desenvolvimento da porção GUI do aplicativo aqui proposto.

Vale lembrar que os programas citados são amplamente difundidos no meio profissional, portanto a documentação disponível sobre eles é extremamente vasta.

3.2.1 Requisitos para execução do IAH

Para o correto desenvolvimento do aplicativo IAH, alguns requisitos devem ser observados e seguidos.

- Para o WMI [MICROSOFT, 2007a]:
    - Sistema operacional *Windows Vista*, *Windows Server 2008*, *Windows Server 2003*, *Windows XP, Windows Me* ou *Windows 2000; Windows NT Workstation 4.0 SP4, Windows 98* ou *Windows 95,* todos com *WMI CORE 1.5* instalado.
- Para o *Visual Studio 2005* [MICROSOFT, 2007d]
    - Processador de 600 MHz ou mais veloz; 192 MB de memória RAM ou mais; 1 GB de disco rígido disponível ou mais; Sistema operacional *Windows 2000 SP4, Windows XP SP2, Windows Server 2003* ou *Windows Vista; Drive* de CD ou DVD; Vídeo de 800x600, 256 cores ou melhor; *Mouse.*
- Para o *SQL Server 2005* [MICROSOFT, 2007b]:
    - Processador de 600 MHz *Pentium 3* (ou compatível) ou mais veloz; .NET *Framework* 2.0 instalado; Sistema operacional *Windows XP SP2, Windows 2000 Professional SP4, Windows 2000 Server SP4, Windows Server 2003, Windows Small Buisness Server SP1, Windows Vista Home Basic* ou melhor, *Windows XP Embedded SP2* ou W*indows Embedded for Point of Service SP2;* 192 MB de memória RAM ou mais; 350 MB de disco rígido disponível ou mais; *Drive* de CD ou DVD; Vídeo Super VGA 1.024x768 ou melhor; *Mouse; Internet Explorer 6.0 SP1.*
- Para o computador atuando como servidor:

- Atender requisitos do *SQL Server 2005;* Estar conectado à mesma rede dos computadores cliente.

- Para os computadores atuando como cliente:
  - .NET *Framework* 2.0; Estarem conectados à mesma rede do computador servidor; Nível de permissão de usuário alto para acesso a dados requisitados.

3.3 FLUXO DE FUNCIONAMENTO

Na figura 3.1 é apresentada uma visão geral do fluxo de funcionamento do aplicativo proposto.

Figura 3.1 – Fluxo de aplicativo *Inventariado Automático de Hardware* (IAH).

O *script* desenvolvido é responsável por gerar mensagem com dados do computador cliente. Esses dados são enviados ao servidor que faz o tratamento apropriado deles. Por último, o administrador da rede tem acesso a esses dados através de solicitações feitas por meio de uma interface gráfica de usuário.

Os passos enumerados abaixo se referem ao fluxo de funcionamento do aplicativo em um ambiente, onde os computadores cliente já estão configurados com o *script* e o servidor está configurado e ativado em um computador.

1º passo – O computador cliente é ligado. Quando o sistema operacional termina de carregar, o *script* é executado, coletando os dados requisitados e gerando uma mensagem para ser enviada ao servidor.

2º passo – A mensagem é enviada do computador cliente ao servidor Primeiro, é estabelecida uma conexão com o banco de dados através da verificação de usuário e senha. Após isso, é construída uma *string* que contém a *query* a ser executada no banco de dados. Esta *query* não contém o horário do fim da sessão, informação que só e atualizada quando o computador cliente é desligado.

3º passo – O DBMS *SQL Server* faz o tratamento da mensagem recebida de computadores cliente e armazena os dados enviados em tabelas pré-estruturadas.

4º passo – Por meio de um *web browser*, o administrador de rede tem acesso à interface gráfica de usuário desenvolvida, que solicita seu *login* e senha.

5º passo – *Login* e senha de administrador são inseridos e verificados.

6º passo – Se os dados fornecidos pelo administrador de rede conferem com os dados previamente armazenados, a conexão ao *script,* que acessa o banco de dados através de instruções SQL, é efetivada.

7º passo – Com os dados disponibilizados, a página *web* ASP.NET é criada.

3.4 SCRIPT VISUAL C#

Os passos enumerados abaixo constituem o processo de desenvolvimento da primeira parte do aplicativo proposto por esse projeto: o *script* em C# utilizando WMI, através da ferramenta *Visual Studio 2005*.

1º passo – Como fica demonstrado na figura 3.2, um novo projeto chamado *MonoDV* é iniciado utilizando a linguagem C# e o *template* de *Console Application*.

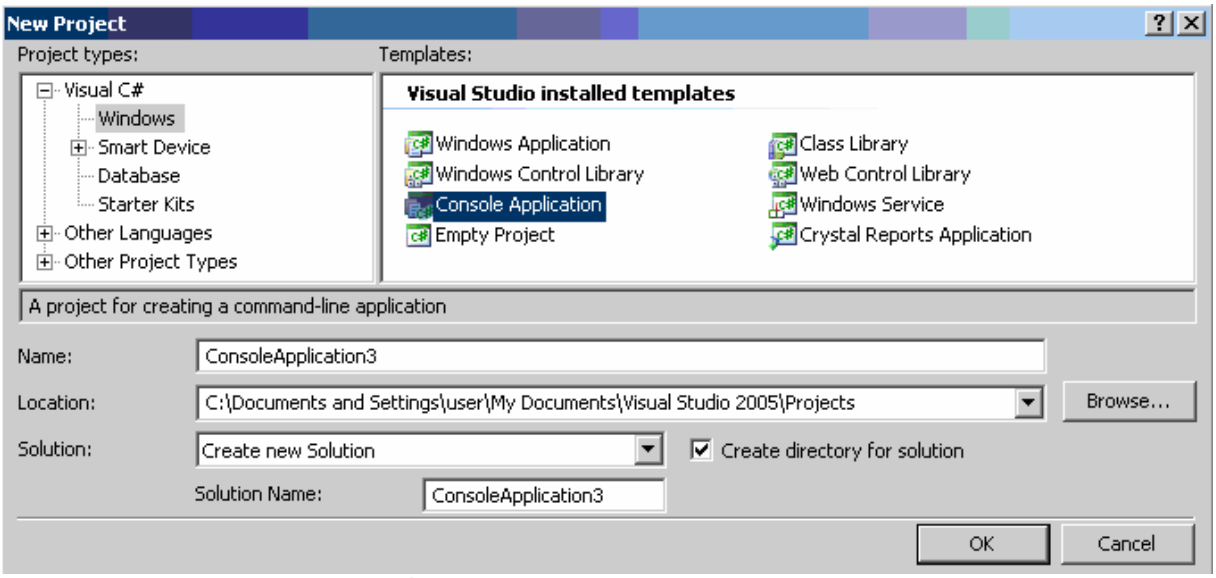


Figura 3.2 – Início de um novo projeto no *Visual Studio 2005*.

2º passo – É adicionada ao projeto *MonoDV a* referência .NET *System.Management,* que possibilita o uso de chamadas WMI no *script*, como ilustrado na figura 3.3.

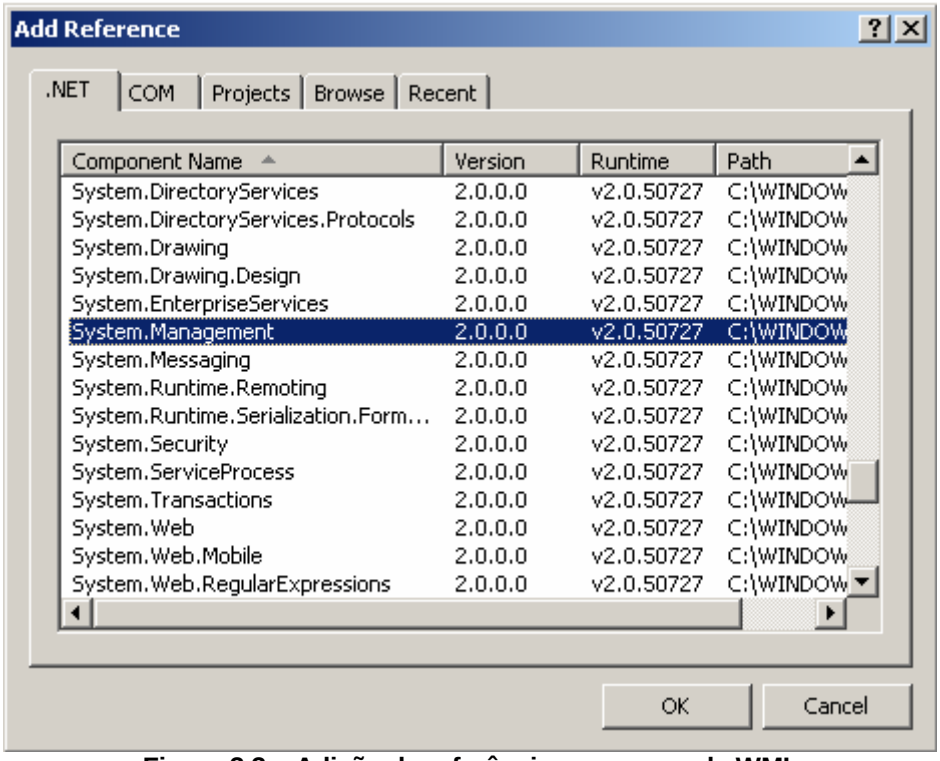


Figura 3.3 – Adição de referência para o uso de WMI.

3º passo – É criada a classe *MonoDV*.

4º passo – Todas as variáveis usadas no *script* são definidas de maneira global, conforme ilustrado na figura 3.4. Isso de dá necessário devido ao fato que as tarefas de coleta de dados foram divididas em métodos diferentes.

banco.cs | Start Page

ConsoleApplication1.MonoDV

```
public int rammb = (1024 * 1024);
public int rampentes = 0;
public Int64 totalram = 0;

public String hdmodelo = "";
public int hdmb = (1024 * 1024);
public int hds = 0;
public Int64 hdcap = 0;

public String cdnome = "";

public String mousefab = "";
public String mouseid = "";

public String keytipo = "";
public String keyid = "";

public String montipo = "";
public String monid = "";
public String monstat = "";

public String mac = "";
public String netnome = "";

public String cpufabricante = "";
public String cpuid = "";
public Int64 velocidade = 0;
public Int64 familia = 0;

public String mbfabricante = "";
public String mbmodelo = "";
public String mbsn = "";
```

Figura 3.4 – Variáveis globais usadas no *script*.

5º passo – São nomeados os métodos que serão usados no *script*, conforme a figura 3.5. É necessário instanciar a própria classe *MonoDV* e fazer referência a seus métodos secundários no método *Main*. A não-invocação de métodos estáticos diretamente é um requisito da orientação a objetos em C# [MARSHALL, 2006].

```
static void Main(string[] args)
{
    MonoDV mono = new MonoDV();
    mono.statsram();
    mono.statshd();
    mono.statscddrive();
    mono.statspointingdevice();
    mono.statskeyboard();
    mono.statsmonitor();
    mono.statsnetworkadapter();
    mono.statscpu();
    mono.statsmotherboard();
    mono.escreveBanco();
    mono.desliga();
}
```

Figura 3.5 – Métodos usados no *script*.

6º passo – O primeiro método, *statsram*, referente a dados sobre a memória RAM, é desenvolvido. A classe WMI *Win32_PhysicalMemory* é usada para a

obtenção de dados. Este passo é repetido para cada método, utilizando-se a classe WMI correspondente aos dados requeridos. A figura 3.6 demonstra o método *statscpu,* usado para a obtenção de dados sobre fabricante, família, identificação e número de série do processador.

```
private void statscpu()
{
    ManagementObjectSearcher busca = new ManagementObjectSearcher("Select * from Win32_Processor");
    foreach (ManagementObject result in busca.Get())
    {
        velocidade = Convert.ToInt64(result.GetPropertyValue("CurrentClockSpeed"));
        familia = Convert.ToInt64(result.GetPropertyValue("Family"));
        cpuid = Convert.ToString(result.GetPropertyValue("DeviceId"));
        cpufabricante = Convert.ToString(result.GetPropertyValue("Manufacturer"));
        ;
    }
    Console.WriteLine("Fabicante: {0}", cpufabricante);
    Console.WriteLine("Familia: {0}", familia);
    Console.WriteLine("Identificacao: {0}", cpuid);
    Console.WriteLine("Velocidade em Hz: {0}", velocidade);
    Console.WriteLine("");
}
```

Figura 3.6 – Código do método *statscpu*.

7º passo – A *string* de conexão com o banco de dados é desenvolvida. São inseridas informações sobre a localização do banco de dados, nome de usuário e senha para que a mensagem gerada pela *string* possa ser inserida no servidor.

8º passo – A *string* de dados recolhida é estruturada para que ela possa ser inserida nos campos corretos do banco de dados. Para sua construção, são concatenadas as estruturas de inserção de dados (cláusula *insert*) SQL com os valores das variáveis recolhidas.

9º passo – Finalmente, se gera o executável do *script* para que ele possa ser instalado nos computadores cliente. A figura 3.7 apresenta o diagrama de classe do *script* desenvolvido mostrando seus campos e métodos.

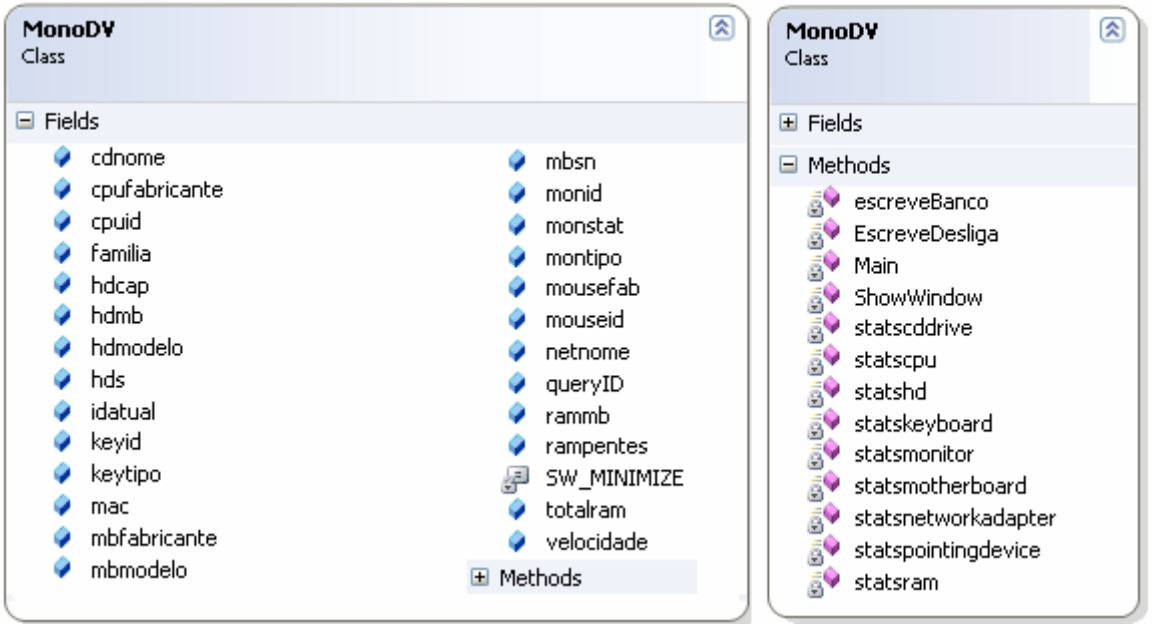


Figura 3.7 – Diagrama de classe do *script* criado

3.5 BANCO DE DADOS SQL SERVER 2005

Uma vez concluído o *script*, pode se dar início a criação da tabela do banco de dados. Isto foi relativamente simples, porque, durante a elaboração do *script*, as variáveis e seus tipos já haviam sido definidas. Os passos enumerados abaixo constituem o processo de desenvolvimento da segunda porção do aplicativo proposto por esse projeto: o banco de dados utilizando o DBMS *SQL Server 2005*.

1º passo – Como é demonstrado na figura 3.8, um novo projeto é iniciado definindo seus parâmetros básicos.

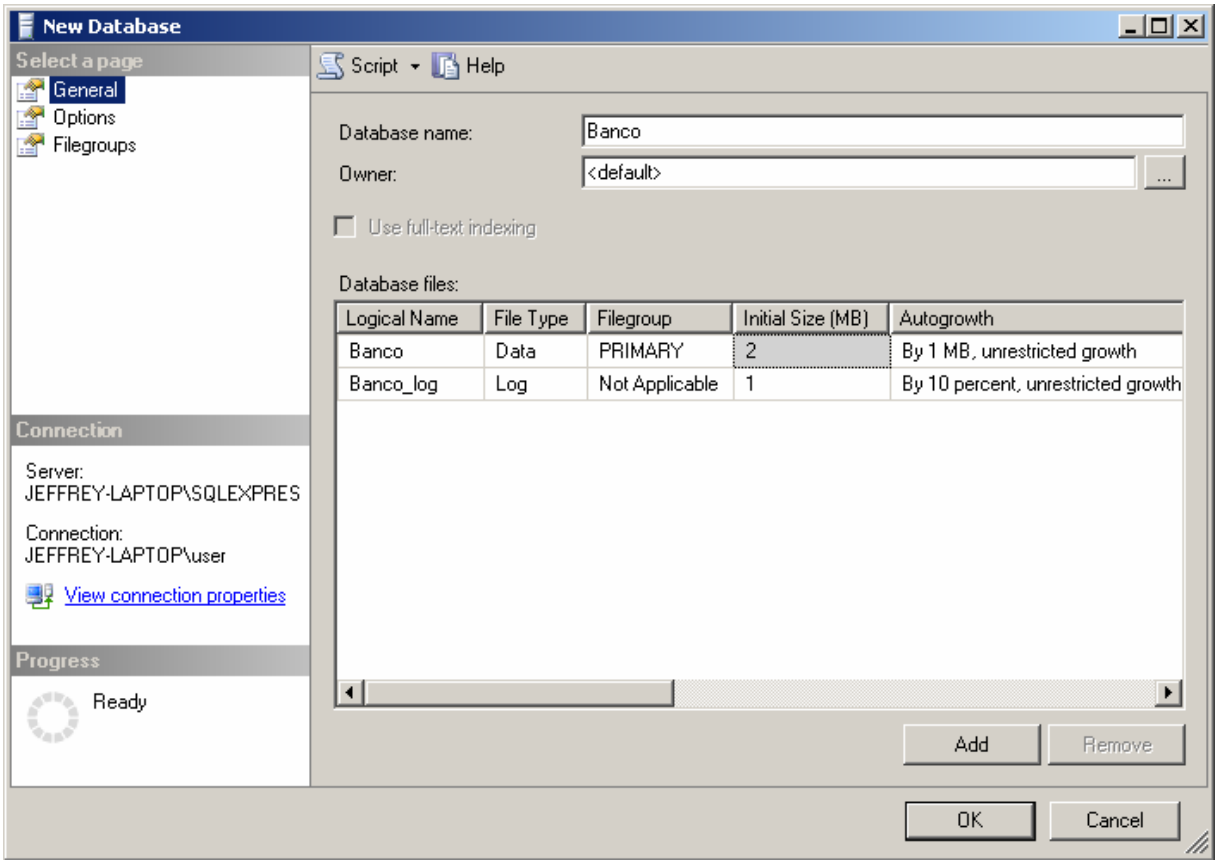


Figura 3.8 – Criação de um novo banco de dados.

2º passo – Ao clicar em *Security* e *Login*, adiciona-se o usuário e senha definidos no sétimo passo do desenvolvimento do *script*. A este usuário se concede todas as permissões.

3º passo – É criada a tabela de banco de dados *InfoClient*. Seu primeiro campo é um identificador com auto-incrementação que é usado como chave da entrada. As entradas seguintes mantêm a ordem do oitavo passo do desenvolvimento do *script,* onde a *string* de inserção de dados é construída. Conforme demonstrado na figura 3.9, o tipo de cada entrada respeita os definidos no quarto passo do desenvolvimento do *script*.

Table - dbo.InfoClient | Summary

Column Name	Data Type	Allow Nulls
id	int	☐
totalram	bigint	☑
rammb	int	☑
rampentes	int	☑
hdcap	bigint	☑
hdmb	bigint	☑
hdmodelo	varchar(MAX)	☑
hds	int	☑
cdnome	varchar(MAX)	☑
mousefab	varchar(MAX)	☑
mouseid	varchar(MAX)	☑
keytipo	varchar(MAX)	☑
keyid	varchar(MAX)	☑
montipo	varchar(MAX)	☑
monid	varchar(MAX)	☑
monstat	varchar(MAX)	☑
mac	varchar(MAX)	☑
netnome	varchar(MAX)	☑
cpufabricante	varchar(MAX)	☑
cpuid	varchar(MAX)	☑
velocidade	bigint	☑
familia	bigint	☑
mbfabricante	varchar(MAX)	☑
mbmodelo	varchar(MAX)	☑
mbsn	varchar(MAX)	☑
tempoliga	datetime	☑
tempodesliga	datetime	☑

Figura 3.9 – Nome de variáveis e tipo definidos em tabela de banco de dados.

4º passo – Por último, o *script* é executado em um computador com o servidor instalado para verificar, através de *query,* a correta inserção de dados de *hardware* nos campos apropriados da tabela gerada no terceiro passo do desenvolvimento do banco de dados.

3.6 INTERFACE WEB ASP.NET

Por último, foi desenvolvida a interface gráfica de usuário (GUI) para *web* do aplicativo. A primeira porção da GUI é responsável pela autenticação do usuário e as seguintes são responsáveis por fornecerem uma interface amigável para visualização dos dados recolhidos anteriormente. Seguem abaixo os passos tomados na última etapa de desenvolvimento do aplicativo:

1º passo – Ao clicar com o botão direito na solução *MonoDV*, foi adicionado um novo *web site* do tipo *ASP.NET Web Site* chamado *MonoWeb,* conforme a figura 3.10.

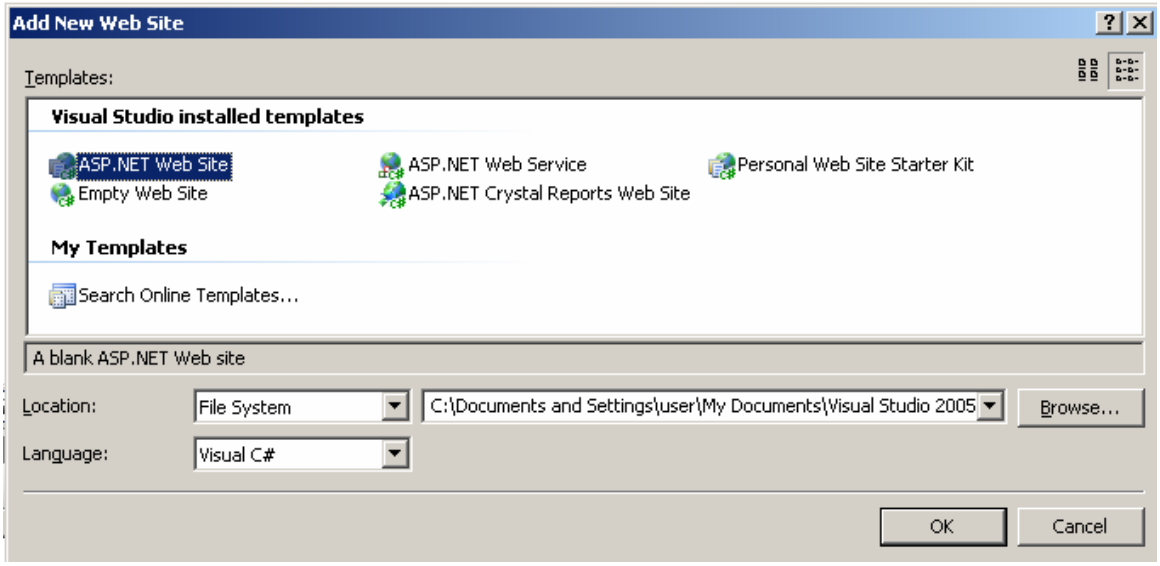


Figura 3.10 – Criação de um novo *web site.*

2º passo – É adicionado o primeiro item da interface gráfica. Usando a opção de novo item *Web Form*, é criado a primeira página chamada de *Default.aspx* para o *login* de usuário. Com isso, o programa *Visual Studio 2005* gera uma página estruturada no padrão *Hypertext Markup Language* (HTML) pronta para receber dados. Após isso, a visualização da área de trabalho é alterada de *Source* para *Design*, pois assim é possível adicionar componentes de acesso a dados. Conforme a figura 3.11, usando a ferramenta *Toolbox* e acessando as opções de *Login,* é arrastado para a área de trabalho o sub-item *Login.* Com o componente de *login* gerado, são então adicionados usuários à lista de permissão de acesso aos relatórios. Ao clicar na opção *Administer Website* do menu *Login Tasks*, uma página é aberta com as ferramentas de administração do *web site*. Na aba *Security,* são adicionados

os usuários que terão acesso à interface gráfica de usuário. No campo *DestinationPageUrl* é definido o destino do *login* com sucesso (*~/home.aspx*). Finalmente, caso o *login* não obtenha sucesso, é configurada uma mensagem de aviso no campo *FailureText*.

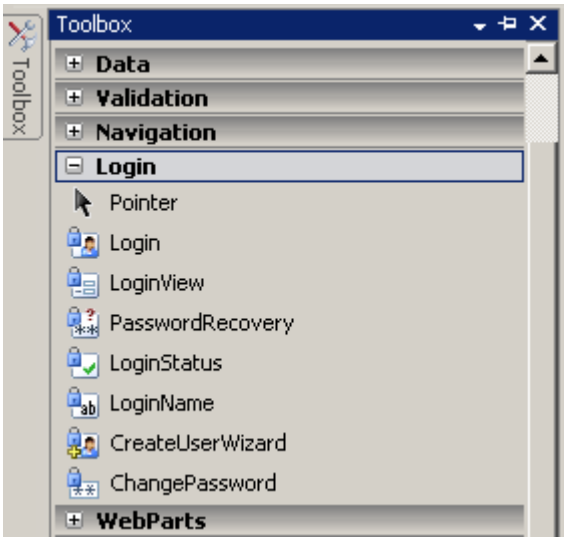


Figura 3.11 – *Toolbox* e opções do sub-item *Login.*

3º passo – Com o acesso de usuários definido, inicia-se a página principal da interface gráfica do usuário. Conforme as etapas iniciais do segundo passo, um novo *Web Form* é adicionado ao *MonoWeb* chamado *home.aspx*. Com a área de trabalho alterada para o modo *Design*, digita-se o nome da ferramenta (*Inventariado Automático de Hardware*) seguido das cinco funções principais do aplicativo (*Relatório Geral – Tempo, Relatório Geral – Hardware, Relatório por Máquina, Dados Gerais do Sistema, Possíveis Alterações de Hardware*). Cada uma das cinco opções é selecionada individualmente e ao apertar *ctrl*+L é adicionado o link *url* que cada opção tem como destino.

4º passo – A GUI do *Relatório Geral – Tempo* é desenvolvida. Então é adicionado um novo *Web Form* chamado *geral.aspx* responsável por mostrar todos os

registros por endereço MAC, tempo liga, tempo desliga, duração total de sessão e finalmente os detalhes do registro. Para tanto, primeiro é criado uma forma de acesso aos dados do banco, usando o componente *SqlDataSource*, sub-item do menu *Data* dentro do *Toolbox*. Com o *SqlDataSource* na área de *Design* é selecionada a opção *Configure Data Source* e é usada a mesma conexão definida no *script*. Em seguida são selecionados os campos usados na página, conforme ilustra a figura 3.12. Esta fase de desenvolvimento também permite *querys* personalizadas, tendo seus valores inseridos manualmente. As várias opções de *query* personalizadas incluem diferença de dois valores ou buscam todos os registros de um campo específico, entre outras funções. Em seguida é selecionada a forma de visualização dos dados requeridos. A opção *GridView* é selecionado do menu *Data* dentro do *Toolbox*. Com o *grid* inserido na área de trabalho, é atribuído a ele o *data source* configurado neste mesmo passo, para que os dados selecionados sejam visualizados no *grid*.

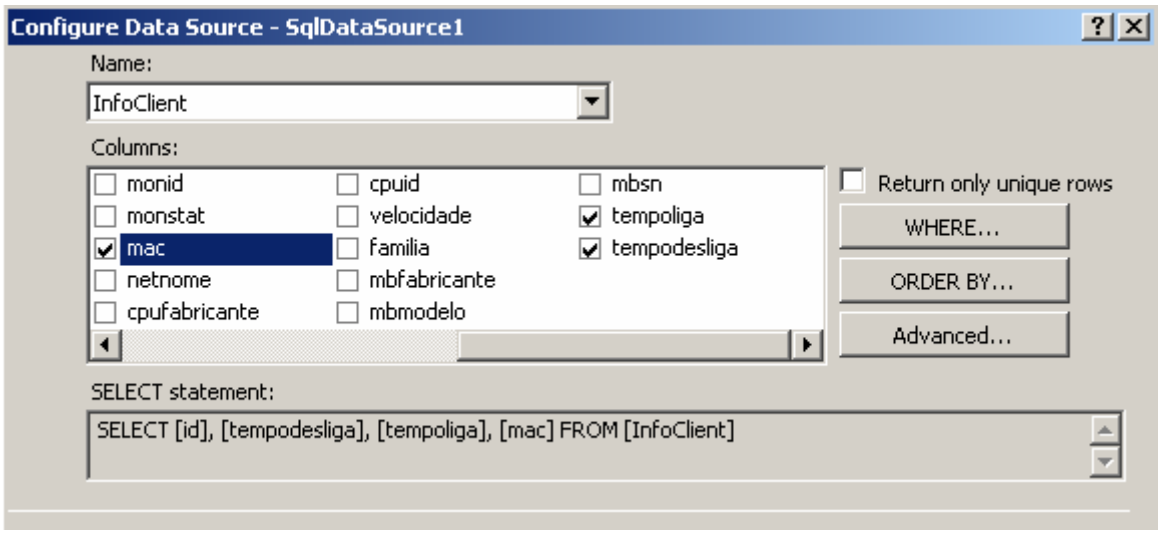


Figura 3.12 – Seleção dos campos a serem usados.

5º passo – O quarto passo é repetido para cada uma das quatro outras opções do *menu* principal. A única diferenciação entre o desenvolvimento de uma opção principal para a outra é o nome dado ao seu destino, os dados selecionados, as *querys* implementadas e a forma de visualização. Com isso, conclui-se a porção de interface gráfica de usuário.

3.7 PROBLEMAS E SOLUÇÕES ENCONTRADOS

No decorrer do desenvolvimento de cada etapa do aplicativo IAH alguns problemas surgiram. Segue-se lista de problemas encontrados e o que foi feito para contorná-los.

- Problema: Método *statshd* retorna valor inconstante sobre quantidade de discos rígido e capacidade total.
    - Solução: Através de requisição de nome de *drives* pôde-se concluir que o WMI considera *flash-drives* localizados em portas do tipo *Universal Serial Bus* (USB) discos rígidos fixo. Para evitar essa situação adicionou-se uma condição onde *drives* do tipo USB não serão selecionados.
- Problema: Método *statsnetworkadapter* não retornava dados consistentes, tais como retorno de dados em branco, endereços MAC variando e diferentes nomes de adaptadores de rede.

  - Solução: Foi verificado que a chamada WMI busca qualquer tipo de conexão existente no computador em questão. Adicionou-se uma exceção, onde somente a conexão ativa seria retornada para solucionar o problema.

- Problema: Dados enviados do computador cliente ao computador servidor gerava erro de inserção de dados no servidor.
  - Solução: A *string* gerada no oitavo passo do desenvolvimento do *script*, *strDados*, foi reestruturada através de referência cruzada entre os tipos de variável dos dados gerados com os tipos de variável dos campos da tabela *InfoClient*.
- Problema: Sempre que computador é inicializado, *script* é executado com sua janela maximizada.
  - Solução: Foi encontrado comando *SW_MINIMIZE* responsável por minimizar aplicação de console. Com isso, assim que o *script* é executado ele é minimizado.
- Problema: Nos primeiros testes realizados, mesmo que computador cliente visualizasse computador com o servidor instalado, o *script* não foi capaz de inserir dados em banco de dados.
  - Solução: Após cuidadosa análise das mensagens de erro geradas, conclui-se que problema era relacionado a níveis de permissão e segurança. Criando-se uma exceção no *firewall* do *Windows* para aplicativos acessando o *SQL Server 2005* o problema foi solucionado.

3.8 APRESENTAÇÃO DO APLICATIVO IAH

A seguir serão apresentadas as telas de funcionamento do aplicativo *Inventário Automático de Hardware* (IAH), bem como suas descrições e características.

Figura 3.13 – Login de interface gráfica de usuário

Como pode ser visto na figura 3.13, a primeira interação do usuário com o aplicativo se dá por uma tela que requisita do usuário um *login* e uma senha, para que o mesmo possa ter acesso ao aplicativo desenvolvido e as informações disponibilizadas por ele. Após o usuário ter inserido esses dados, uma verificação é feita com dados previamente armazenados, autorizando ou não o acesso do mesmo.

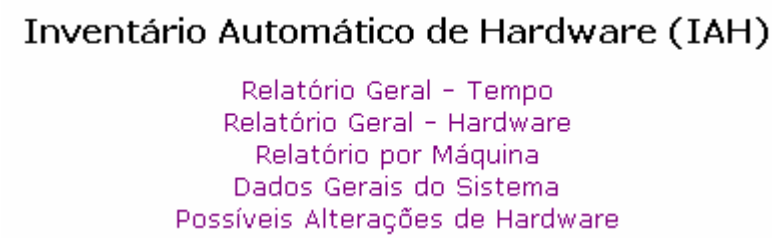


Figura 3.14 – Tela de opções

Caso o acesso do usuário seja liberado, ele se depara com a as opções apresentadas na figura 3.14. Cada uma das cinco opções (*Relatório Geral – Tempo; Relatório Geral – Hardware; Relatório por Máquina; Dados Gerais do Sistema; Possíveis Alterações de Hardware*) leva a uma página, cada uma delas com um conjunto de informações único.

Inventário Automático de Hardware (IAH) - Relatório Geral

	Endereço MAC	Tempo - Liga	Tempo - Desliga	Tempo Total	
>	0090F55D6F01	11/16/2007 3:20:54 AM	11/16/2007 3:40:01 AM	1147 segundo(s)	Detalhes do Registro
>	000FB0370852	11/16/2007 3:09:23 AM	11/16/2007 3:34:27 AM	1504 segundo(s)	Detalhes do Registro
>	000EA67A7EC3	11/16/2007 3:02:34 AM	11/16/2007 3:31:29 AM	1735 segundo(s)	Detalhes do Registro
>	000FB0661CC8	11/16/2007 3:02:29 AM	11/16/2007 3:21:38 AM	1149 segundo(s)	Detalhes do Registro
>	0016365A0BBE	11/16/2007 3:01:52 AM	11/16/2007 3:24:11 AM	1339 segundo(s)	Detalhes do Registro
>	000FB0370852	11/16/2007 2:45:57 AM	11/16/2007 3:02:13 AM	976 segundo(s)	Detalhes do Registro
>	0016365A0BBE	11/16/2007 2:42:25 AM	11/16/2007 2:58:58 AM	993 segundo(s)	Detalhes do Registro
>	000FB0661CC8	11/16/2007 2:42:15 AM	11/16/2007 2:58:44 AM	989 segundo(s)	Detalhes do Registro
>	0090F55D6F01	11/16/2007 2:41:40 AM	11/16/2007 3:01:57 AM	1217 segundo(s)	Detalhes do Registro
>	000EA67A7EC3	11/16/2007 2:41:21 AM	11/16/2007 2:55:41 AM	860 segundo(s)	Detalhes do Registro

1 2 3 4

Figura 3.15 – Relatório geral de uso por MAC

A primeira opção leva ao relatório geral por tempo, conforme ilustrado na figura 3.15. Aqui o usuário encontra uma tabela que mostra, por registro, o endereço MAC de um computador, o horário que ele foi ligado, o horário que ele foi desligado, a

duração da sessão e os detalhes do registro. Os detalhes do registro são acessados clicando nas palavras “*Detalhes de Registro*” e levam a uma página similar à apresentada na figura 3.16.

Inventário Automático de Hardware (IAH) - Detalhes de Registro

id	31	**Status do Monitor**	OK
RAM Total	536870912	**Endereço MAC**	000EA67A7EC3
Total de Pentes	1	**Nome Adaptador de Rede**	NVIDIA nForce MCP Networking Adapter
Modelo do HD	Maxtor 6Y080L0	**Fabricante do Processador**	AuthenticAMD
Quantidade de HDs	1	**ID do Processador**	CPU0
Nome do Drive de CD	HL-DT-ST CD-ROM GCR-8523B	**Velocidade CPU**	1491
Fabricante Mouse	Microsoft	**Família CPU**	29
ID do Mouse	ACPI\PNP0F13\3&13C0B0C5&0	**Fabricante Placa-mãe**	ASUSTeK Computer INC.
Tipo de Teclado	Aperfeiçoado (101 ou 102 teclas)	**Modelo placa-mãe**	A7N8X-X
ID do Teclado	ACPI\PNP0303\3&13C0B0C5&0	**No. serial Placa-mãe**	xxxxxxxxxx
Tipo de Monitor	Samsung	**Horário - Liga**	11/16/2007 3:02:34 AM
ID do Monitor	SyncMaster 753DF(T)/ 783DF(T), MagicSyncMaster AQ17DF	**Horário - Desliga**	11/16/2007 3:31:29 AM

Figura 3.16 – Detalhes de Registro

O *link* seguinte, *Relatório Geral – Hardware*, leva a uma tabela estruturada da mesma maneira que os dados são inseridos, apresentando todos os campos armazenados no banco de dados.

O *link Relatório por Máquina* leva a uma página com todos os diferentes endereços MAC existentes no banco de dados. Ao clicar em qualquer um dos endereços MAC disponíveis, uma tela com todos os registros daquele MAC é disponibilizada. Endereço MAC é utilizado como identificador e agrupador no IAH devido ao fato dele ser único para cada computador.

O quarto *link, Dados Gerias do Sistema*, apresenta ao usuário duas informações sobre os registros armazenados no banco de dados. Primeiro, o tempo

médio em minutos de cada sessão e em seguida a quantidade total de sessões registradas.

Por último, o link *Possíveis Alterações de Hardware* apresenta uma tela com todos os endereços MAC presentes no banco de dados e mais 4 colunas, *Memória RAM, Capacidade de HD*, *ID CPU* e *ID Placa Mãe*, como pode ser observado na figura 3.17. Caso exista uma ou mais alterações de dados em qualquer uma dessas quatro colunas, uma mensagem de alerta, específica por MAC e tipo de *hardware*, é mostrada no campo afetado ao usuário para conscientizá-lo de alterações de *hardware*.

Inventário Automático de Hardware (IAH) - Possíveis Alterações de Hardware

Endereço MAC	Memória RAM	Capacidade do HD	ID CPU	ID Placa Mãe
000EA67A7EC3	OK	OK	OK	OK
000FB0370852	OK	OK	OK	OK
000FB0661CC8	OK	OK	OK	OK
0016365A0BBE	OK	OK	OK	OK
0090F55D6F01	OK	OK	OK	OK

Figura 3.17 – Possíveis alterações de *hardware*

Capítulo 4 – Experimento e Resultados

4. EXPERIMENTO E RESULTADOS

4.1 INTRODUÇÃO

Para verificar a funcionalidade do aplicativo IAH apresentado no capítulo anterior, testes foram feitos entres dois computadores. Um computador foi configurado como computador servidor de banco SQL, enquanto outro foi configurado como computador cliente. Esse cenário de testes foi usado para corrigir qualquer *bug* no programa, assim como para verificar seu correto funcionamento.

Após essa primeira fase de testes, o aplicativo já se encontrava em fase de experimentação. Seis computadores foram configurados em uma rede, sendo um computador servidor SQLS e outros cinco funcionando como cliente. Sob esse cenário de experimentação, uma série de rotinas foram executadas, possibilitando a coleta de dados e a obtenção de resultados acerca do desempenho e funcionamento do aplicativo.

4.2 CONFIGURAÇÃO DO AMBIENTE DE TESTES

O cenário de testes foi constituído de dois computadores, um como computador cliente e outro como servidor. Seguem-se as configurações básicas de hardware de cada um:

- Computador servidor:
  - Fabricante Dell; Processador Core 2 Duo T5600 1.83GHz; 1024 MB de memória Ram; 55 GB de disco rígido.
- Computador cliente:
  - Fabricante HP; Processador Athlon XP 2500+ 1.85GHz; 512 MB de memória RAM; 40 GB de disco rígido.

Ambos os computadores do cenário de testes atendem a todos os requisitos de sistema, em termos de *hardware* e *software*, para funcionarem como computador servidor ou cliente.

Abaixo, detalhadamente, são apontadas as ações tomadas para a configuração e execução deste cenário de testes;

1º passo – Ambos os computadores usados foram configurados em uma mesma rede para que o computador cliente pudesse enviar com sucesso informações ao computador servidor. A configuração de rede do computador servidor respeita o

sétimo passo de desenvolvimento do *script,* onde é definida a localização por IP do servidor. A figura 4.1 ilustra as configurações de rede do computador servidor.

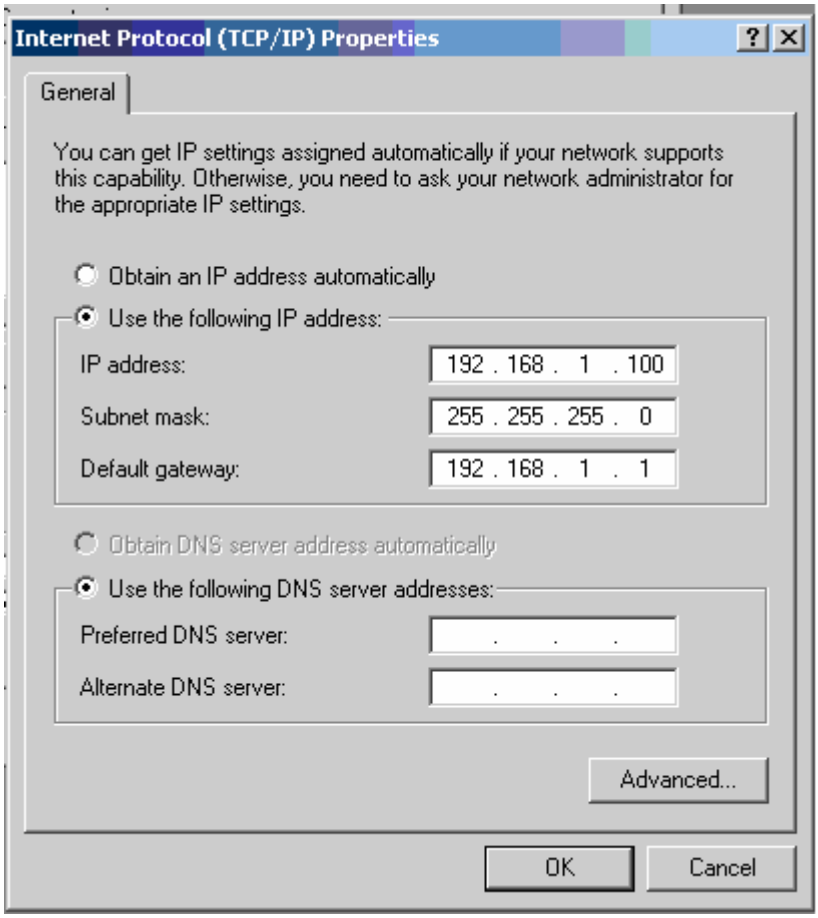


Figura 4.1 – Configuração de rede de computador servidor.

2º passo – O computador servidor tem o banco de dados iniciado e uma *query* é feita à tabela *InfoClient* para a verificação dos dados previamente inseridos. Através da *query "truncate table InfoClient*" a tabela *InfoClient* foi reiniciada para a verificação de que novas entradas estão sendo escritas pelo computador cliente.

3º passo – É instalado no computador cliente o executável do *script* e o .NET *Framework* 2.0 para a correta comunicação com o banco de dados.

4º passo – Com ambos os computadores configurados foi executada uma rotina de liga/desliga no computador cliente para a execução do *script* e a inserção de dados no computador servidor.

5º passo – Verificado que os passos acima foram executados com sucesso, foi dado início a uma nova rotina de testes. O computador cliente foi então submetido a uma série de rotinas de liga/desliga, com tempos de sessões variáveis. Além disso, um comparativo de uso de recursos foi efetuado para verificar o efeito do *script* no desempenho do computador cliente.

6º passo – Após os testes realizados, comprovou-se a correta inserção de informações em banco de dados através da *query* "*select * from InfoClient*" no DBMS.

7º passo – Finalmente, observou-se a correta inserção de dados em banco de dados através do uso de pagina *web*, criada na terceira etapa do desenvolvimento do aplicativo.

4.3 IDENTIFICAÇÃO DO PLANO DO EXPERIMENTO

Uma vez a fase de testes concluída, pôde-se dar início a fase de experimentação. Essa nova fase consiste em montar uma rede privada com seis computadores utilizando um *8 Port NWay Switch* da marca *Encore*. Destes seis computadores um foi configurado como servidor enquanto os outros cinco como computadores cliente.

De acordo com o explicitado no primeiro objetivo desse trabalho, os seguintes itens de *hardware* foram coletados:

- Processador:
    - Fabricante; Família; Identificação; Número de série.
- Placa mãe:
    - Fabricante; Modelo; Número de série.
- Memória RAM:
    - Quantidade de pentes; Capacidade de memória RAM total.
- Discos Rígidos:
    - Nome; Capacidade; Quantidade.
- Placa de rede:
    - Nome; Endereço MAC.
- Teclado:
    - Tipo; Identificação.
- *Mouse*:
    - Tipo; Identificação.
- Monitor:
    - Tipo; Identificação; *Status*.
- *Drives* de CD:
    - Nome.

Também foi armazenado o início e o fim de cada sessão de uso de computadores clientes, como proposto no segundo objetivo deste trabalho. Finalmente, a correta inserção dos dados mencionados acima foi verificada através de

uma interface gráfica de usuário, como proposto no terceiro e último objetivo deste trabalho.

4.4 CONFIGURAÇÃO DO AMBIENTE DO EXPERIMENTO

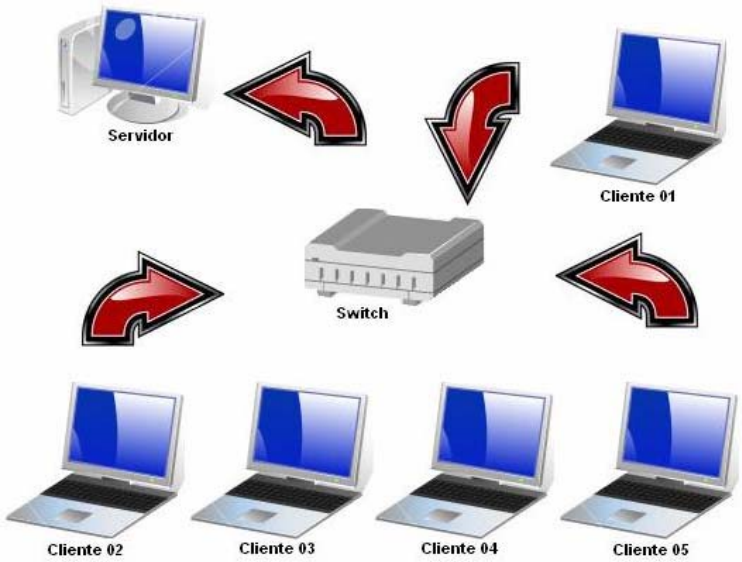


Figura 4.2 – *Setup* do ambiente de experimento

Conforme a figura 4.2, o cenário de experimento foi constituído de seis computadores, sendo um servidor e outros cinco clientes. Segue abaixo configuração básica de hardware de cada um:

- Computador servidor:
    - *Laptop* fabricante Dell; Processador Core 2 Duo T5600 1.83GHz; 1024 MB de memória Ram; 55 GB de disco rígido.

- Computador cliente 01:
    - *Laptop* fabricante Toshiba; Processador Celeron M 1.3 GHz; 256 MB de memória RAM; 37 GB de disco rígido.
- Computador cliente 02:
    - *Laptop* fabricante Toshiba; Processador Celeron M 1.3 GHz; 256 MB de memória RAM; 37 GB de disco rígido.
- Computador cliente 03:
    - *Laptop* fabricante HP; Processador Dual-Core U1400 1.20 GHz; 1024 MB de memória RAM; 75 GB de disco rígido.
- Computador cliente 04:
    - *Laptop* fabricante Positivo; Processador Core 2 Duo T5300 1.73 GHz; 2048 MB de memória RAM; 120 GB de disco rígido.
- Computador cliente 05:
    - *Desktop* montado; Processador Athlon XP 1800+ 1.49GHz; 512 MB de memória RAM; 77 GB de disco rígido.

O computador servidor atende os requisitos de *hardware* e *software* para ser computador servidor ou cliente. Entretanto, ele foi utilizado, somente, como computador cliente devido ao fato de que se sua sessão fosse encerrada (computador reiniciado) os outros computadores clientes perderiam a capacidade de se conectar ao servidor.

Todos os outros cinco computadores atendiam aos requisitos de *hardware* para serem computadores clientes. A única exigência para serem compatíveis com os

requisitos de *software* do aplicativo IAH para um computador cliente era ter o .NET *Framework 2.0* instalados.

4.5 TESTES REALIZADOS

Seguem-se abaixo as ações tomadas para a configuração e execução do cenário de experimentação;

1º passo – Todos os seis computadores usados foram configurados em uma mesma rede para que os cinco computadores cliente pudessem enviar com sucesso informações ao computador servidor. A configuração de rede do computador servidor deve respeitar o sétimo passo de desenvolvimento do *script*, onde é definida a localização por IP do servidor.

2º passo – O computador servidor teve seu banco de dados iniciado e uma *query* foi feita à tabela *InfoClient* para verificação de dados previamente inseridos. Usando a *query* "*truncate table InfoClient*" a tabela *InfoClient* foi reiniciada para o experimento.

3º passo – Foi instalado nos cinco computadores cliente o executável do *script* e o .NET *Framework* 2.0 para a correta comunicação com o banco de dados. Em todos os cinco computadores cliente também foi alterado o local do atalho do *script* para a pasta de programas executados durante o *startup* do sistema operacional.

4º passo – Foi executada uma rotina de liga/desliga por computador cliente para a execução do *script* e a inserção de dados em tabela. Com isso, a correta comunicação de cada computador cliente com o servidor pôde ser observada.

5º passo – Deu-se inicio então a uma rotina de testes liga/desliga simulando várias sessões de usuários de durações variadas..

6º passo – Após os testes realizados foi verificada a correta inserção de dados em banco de dados através de *query* em DBMS. A query "*select * from InfoClient"* retorna todos as entradas e valores inseridos na tabela, confirmando o funcionamento do aplicativo.

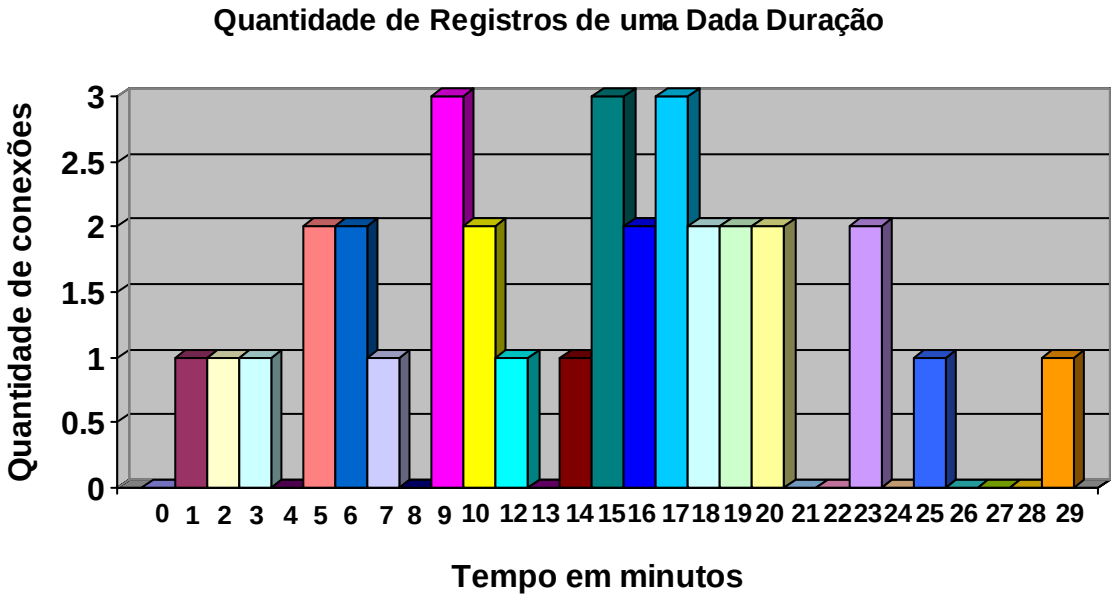
7º passo – Finalmente, a correta inserção de dados foi também observada através do uso de GUI desenvolvida na terceira etapa do aplicativo.

4.6 RESULTADOS OBTIDOS

Durante as duas horas e quinze minutos de testes, foram computados um total de trinta e três registros dos cinco computadores ligados na rede montada.

O computador cliente 01, com terminação MAC xx:52, teve sete registros. O computador cliente 02, com terminação MAC xx:C8, teve também sete registros. O computador cliente 03, com terminação MAC xx:BE, teve nove registros. O computador cliente 04, com terminação MAC xx:01, teve seis registros e finalmente, o computador cliente 05, com terminação MAC xx:C3, teve quatro registros.

Os tempos de conexões variaram de dois a vinte e nove minutos. A figura 4.2 demonstra a quantidade de registros por duração de sessão (em minutos). Os *Dados Gerais do Sistema* retornaram a informação que o tempo médio das conexões foi de 14 minutos.



.Figura 4.3 – Gráfico de quantidade de registros por duração de sessão

O *Relatório Geral – Hardware* retornou 858 campos de informações sem nenhum erro ou *delay*, sendo 792 desses campos informações de *hardware*. Essas informações, coletadas através de WMI, são apresentadas conforme a tabela 4.1.

	id	totalram	rammb	rampentes	hdcap	hdmb	hdmodelo	hds	cdnome
>	33	2147483648	1048576	2	120031511040	1048576	SAMSUNG HM121HI ATA Device	1	MATSHITA DVD-RAM UJ-850S ATA Device
>	32	268435456	1048576	1	40007761920	1048576	HTS424040M9AT00	1	MATSHITA UJDA760 DVD/CDRW
>	31	536870912	1048576	1	81956689920	1048576	Maxtor 6Y080L0	1	HL-DT-ST CD-ROM GCR-8523B
>	30	268435456	1048576	1	40007761920	1048576	HTS424040M9AT00	1	MATSHITA UJDA760 DVD/CDRW

Tabela 4.1 – Exemplo de dados apresentados no *Relatório Geral - Hardware*

Finalmente, os campos que informam possíveis alterações de *hardware* retornaram informações corretas sobre mudanças de hardware nos computadores utilizados no experimento.

Capítulo 5 – Conclusões

O IAH foi criado com três objetivos: fornecer um inventariado de *hardware* automaticamente que é atualizado toda vez que um computador é ligado, verificar o início e o fim de cada sessão de uso de computadores e disponibilizar essas informações ao AR e, finalmente, disponibilizar uma interface gráfica de usuário (GUI), para que às informações coletadas possam ser acessadas com maior facilidade e agilidade.

O desenvolvimento do aplicativo foi dividido em três etapas. Seu início se deu com a criação de um *script* responsável pela coleta de dados de *hardware*, horário de uso e conexão com banco de dados. Após isso, foi desenvolvida uma tabela para o armazenamento dos dados obtidos e finalmente foi criada uma GUI.

O aplicativo supracitado empregou soluções *Microsoft.* *Microsoft Visual Studio 2005* foi usado para o *script* com tecnologia WMI e, também, para a GUI com tecnologia ASP.NET, enquanto o *Microsoft SQL Server 2005* (SQLS) foi utilizado para a confecção da tabela e armazenamento de dados. Com a conclusão do aplicativo, deu-se inicio a uma fase de testes e experimentações que levaram a uma série de resultados que permitiram uma análise da funcionalidade de uso.

Ao final dos testes realizados, pôde-se observar que as tecnologias usadas foram apropriadas à criação de tal aplicativo. A tecnologia WMI pode ser facilmente aplicada através do *Visual C#* (VC#) para a obtenção dos dados necessários, que foram armazenados, corretamente, sem nenhuma falha e o VC# também provou ser

apto para o desenvolvimento de uma GUI capaz de suprir as necessidades do aplicativo proposto.

O VC# aliado ao WMI gera uma importante ferramenta para criação de aplicativos que auxiliam um administrador de rede. Os diferentes dados que o WMI, juntamente com um aplicativo desenvolvido em VC#, consegue obter de um computador permite a criação de um banco de dados com informações de configurações de computadores, alterações de *hardware* e registros de sessões de uso que podem ser vitais a um AR.

Ao final dos testes, com todos os dados coletados, pode-se concluir que a obtenção de dados de inventariado foi um sucesso. Em um período de tempo de duas horas e quinze minutos 792 campos de informações de *hardware* foram coletados de cinco computadores diferentes, sendo todos os campos armazenados na tabela do banco de dados sem nenhuma falha. Isso significa uma informação a cada 10.2 segundos sem nenhum erro e qualquer mão-de-obra adicional para a coleta de dados.

Testes realizados comprovaram que a execução do *script* teve um efeito quase nulo na carga de trabalho do processador. Observou-se apenas uma pequena sobrecarga de processamento gerada no início do *script*, contudo, com decorrer do tempo, essa carga caiu quase à zero. Contudo, com o decorrer do tempo, essa sobrecarga caiu quase a zero mostrando que quanto mais a longa a conexão, mais próxima a zero a carga do script se apresentará no processador.

Outra conclusão satisfatória foi o êxito na obtenção dos tempos de cada sessão de uso. O intervalo de tempo de verificação pelo servidor se um computador

cliente está em uso foi de aproximadamente três segundos, fazendo que os dados obtidos gerassem informações satisfatórias.

Finalmente, a interface gráfica de usuário (GUI) cumpriu sua tarefa sem nenhuma falha. Os dados foram apresentados de uma forma limpa e objetiva, sem gerar nenhum trabalho desnecessário ao administrador de rede. A GUI de *login* provou ser funcional, permitindo, apenas, que usuários cadastrados tivessem acessos aos cinco *links* da tela principal da GUI, cada um com seu conjunto de informações únicas e de fácil interpretação. Cada sessão de uso pôde ter seu tempo de início, final e tempo total de sessão verificada, bem como cada registro ter suas informações de *hardware* visualizadas com facilidade.

O desenvolvimento em etapas desse aplicativo também provou ser um sucesso. A criação do aplicativo em etapas permitiu a correção de erros em fases, promovendo um maior entendimento e a obtenção de um produto final funcional.

Para o desenvolvimento desse trabalho, desde a concepção do tema até a conclusão do aplicativo, várias matérias ministradas no curso *Engenharia da Computação* foram de grande valia. Dentre elas, destacam-se:

- Tópicos Avançados de Redes, Redes I e II: o conhecimento do funcionamento de uma rede e as responsabilidades de um administrador de redes permitiu o desenvolvimento da proposta do tema além de noções de tecnologias que deviam ser empregadas no desenvolvimento do aplicativo.
- Estágio Profissional: auxiliou no desenvolvimento de relatórios, documentação e toda e qualquer parte escrita deste trabalho.

- Desempenho e Segurança de Sistemas: apresentou conceitos de segurança da informação, utilizados no aplicativo para proteger a integridade e acesso limitado às informações coletadas.
- Banco de Dados I: permitiu a escolha de um DBMS apropriado ao trabalho além de apresentar conceitos chave para o desenvolvimento do banco de dados e tabela utilizados.
- Estrutura de dados, Linguagem Técnica de Programação I e II: introduziu conceitos de programação crucias para o desenvolvimento do *script*.
- Arquitetura de Computadores: forneceu conhecimentos acerca de componentes de computadores, entre outras informações valiosas, que auxiliaram na escolha de dados coletados pelo *script* para consulta.
- Engenharia de Programas: técnicas de desenvolvimento de um *software* puderam ser facilmente importadas para o desenvolvimento do aplicativo aqui proposto facilitando seu andamento e permitindo seu sucesso.

5.1 SUGESTÕES PARA TRABALHOS FUTUROS

No decorrer do desenvolvimento do aplicativo algumas possibilidades de melhorias e expansão do mesmo surgiram, infelizmente não puderam ser aplicadas

devido ao curto período de tempo existente para desenvolvimento. Seguem abaixo sugestões de trabalhos futuros.

- O WMI pode fornecer dezenas de informações dos mais variados campos sobre um computador com um SO *Windows* instalado. Explorando mais a fundo o potencial do WMI, várias outras funcionalidades podem ser adicionadas ao aplicativo, como por exemplo, a verificação de softwares instalados via o *Windows Installer*, tornando o aplicativo mais rico.
- O desenvolvimento de uma GUI que irá tratar os dados obtidos de uma forma mais interativa e dinâmica utilizando conhecimentos de estatística, permitindo a geração de gráficos, tabelas, esquemas e previsões de uso baseado em dados previamente coletados.
- O aplicativo desenvolvido é um *Console Application*, por tanto é visível ao usuário. Modificando o *script* para se tornar um *Windows Service* possibilitará que o aplicativo rode de maneira oculta. O mesmo banco de dados e GUI podem ser facilmente aproveitados.
- O WMI é uma tecnologia exclusiva de SO *Windows*. A idéia central desse trabalho pode ser aproveitada para o desenvolvimento de um aplicativo similar que possa ser executado em qualquer plataforma, assim ampliando o leque de aplicação do mesmo.

Referências Bibliográficas

CISCO. Cisco Systems, Inc. **Simple Network Management Protocol (SNMP)**. 12 out. 2006. Disponível em: <http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/snmp.htm>. Acesso em: 22 set. 2007.

CISCO. Cisco Systems, Inc. **CiscoWorks Resource Manager Essentials**. 15 jun. 2007. Disponível em: <http://www.cisco.com/en/US/products/sw/cscowork/ ps2073/>. Acesso em: 22 set. 2007.

DOBSON, Rick. **Beginning SQL Server 2005 Express Edition Database Applications with Visual Basic Express and Visual Web Developer Express From Novice to Professional**. Nova Iorque, Nova Iorque, EUA; Springer-Verlag New York, Inc., 2006

ECMA. European Computer Manufacturer Association. **C# Language Specification**. Jun. 2006. Disponível em: <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf>. Acesso em: 2 out. 2007

GASKIN, James. Review: Network inventory tools, Inventory you can count on. **Network World**, 31 mai. 1999. Disponível em: <http://www.networkworld.com/reviews/0531rev.html>. Acesso em: 4 set. 2007

GEORGE, Rajesh; DELANO, Lance. **Wrox's SQL Server™ 2005 Express Edition Starter Kit**. Indianápolis, Indiana, EUA; Wiley Publishing, Inc., 2006

HART, Chris; KAUFFMAN, John; SUSSMAN, David; ULLMAN, Chris. **Beginning ASP.NET 2.0 with C#**. Indianápolis, Indiana, EUA; Wiley Publishing, Inc., 2006

HP. Hewlett-Packard Development Company, L.P. **HP Enterprise Discovery software**. Abr. 2007. Disponível em: <https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-15-43^1340_4000_100>. Acesso em: 15 set. 2007.

IBM. International Business Machines Corp. **IBM – IBM Tivoli Configuration Manager**. 16 jan. 2007. Disponível em: <http://www-306.ibm.com/software/tivoli/products/config-mgr/index.html>. Acesso em: 14 set. 2007.

JONES, Allen; MACDONALD, Matthew; RAJAN, Rakesh. **Visual C# 2005: A Problem-Solution Approach**. Nova Iorque, Nova Iorque, EUA; Springer-Verlag New York, Inc., 2006

MACDONALD, Matthew. **Beginning ASP.NET 2.0 in C# 2005: From Novice to Professional**. Nova Iorque, Nova Iorque, EUA; Springer-Verlag New York, Inc., 2006

MARSHALL, Donis. **Programming Microsoft Visual C# 2005 – The Language**. Washington, EUA; Microsoft Press, 2006

MICROSOFT. Microsoft Corporation. **Microsoft SQL Server 2005: SQL Server 2005**. 2007b. Disponível em: <http://www.microsoft.com/sql/prodinfo/overview/default.mspx>. Acesso em: 27 set. 2007.

MICROSOFT. Microsoft Corporation. **WMI – Windows Management Instrumentation**. 2007a. Disponível em: <http://www.microsoft.com/whdc/system/pnppwr/wmi/default.mspx>. Acesso em: 26 set. 2007.

MICROSOFT. Microsoft Corporation. **Visual Studio Hardware Requirements**. 2007b. Disponível em: <http://msdn2.microsoft.com/en-us/library/4c26cc39 (VS.80).aspx>. Acesso em: 04 out. 2007.

MICROSOFT. Microsoft Corporation. **Microsoft SQL Server: SQL Server 2005 Express Edition System Requirements**. 2007c. Disponível em: <http://www.microsoft.com/sql/editions/express/sysreqs.mspx>. Acesso em: 26 set. 2007.

MSDN. Microsoft Developer Network. **Windows Management Instrumentation (Windows)**. 11 jan. 2007a. Disponível em: <http://msdn2.microsoft.com/en-us/library/aa394582.aspx>. Acesso em: 26 set. 2007.

MSDN. Microsoft Developer Network. **WMI Tasks: Computer Software (Windows)**. 11 jan. 2007b. Disponível em: <http://msdn2.microsoft.com/en-us/library/aa394588.aspx>. Acesso em: 26 set. 2007.

MSDN. Microsoft Developer Network. **Using WMI to Deliver ASP.NET Health Monitioring Events**. 2007c. Disponível em: <http://msdn2.microsoft.com/en-us/library/ms178709.aspx>. Acesso em: 26 set. 2007.

MSDN. Microsoft Developer Network. **WMI Architecture (Windows)**. 2007d. Disponível em: <http://msdn2.microsoft.com/en-us/library/aa394553.aspx>. Acesso em: 26 set. 2007.

MSDN. Microsoft Developer Network. **Visual C#**. 2007e. Disponível em: <http://msdn2.microsoft.com/en-us/library/kx37x362(VS.80).aspx>. Acesso em: 20 set. 2007.

MSDN. Microsoft Developer Network. **Introduction to the C# Language and the .NET Framework.** 2007f. Disponível em: <http://msdn2.microsoft.com/en-us/library/z1zx9t92(VS.80).aspx>. Acesso em: 21 set. 2007.

MSDN. Microsoft Developer Network. **ASP.NET Developer Center.** 2007g. Disponível em: <http://msdn2.microsoft.com/en-us/asp.net/default.aspx>. Acesso em: 27 set. 2007.

NOVELL. Novell, Inc. **Novell ZENworks Asset Management**. 2007. Disponível em: <http://www.novell.com/products/zenworks/assetmanagement/>. Acesso em: 15 set. 2007.

SNMP. SNMP Research International, Inc. **SNMP Research**. 2007. Disponível em: <http://www.snmp.com/>. Acesso em: 21 set. 2007.

SOUSA JR., Rafael Timóteo de. Universidade de Brasília. **Administração e Gerêrencia de Rede**, Brasília, 5 abr. 2005. Disponível em anexo.

Apêndice A – Códigos de Programação

Coleta de dados

```
using System;
using System.IO;
using System.Management;
using System.Management.Instrumentation;
using System.Data.SqlClient;
using System.Threading;

namespace Monografia
{
    class MonoDV
    {
        //variaveis globais devido a tarefa de coleta em métodos diferentes
        public int rammb = (1024 * 1024);
        public int rampentes = 0;
        public Int64 totalram = 0;

        public String hdmodelo = "";
        public int hdmb = (1024 * 1024);
        public int hds = 0;
        public Int64 hdcap = 0;

        public String cdnome = "";

        public String mousefab = "";
        public String mouseid = "";

        public String keytipo = "";
        public String keyid = "";

        public String montipo = "";
        public String monid = "";
```

```
        public String monstat = "";

        public String mac = "";
        public String netnome = "";

        public String cpufabricante = "";
        public String cpuid = "";
        public Int64 velocidade = 0;
        public Int64 familia = 0;

        public String mbfabricante = "";
        public String mbmodelo = "";
        public String mbsn = "";

        public string idatual = "";
        public String queryID = "";

        private const int SW_MINIMIZE = 6;
        [System.Runtime.InteropServices.DllImport("user32.dll")]
        private static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);


        [STAThread]

        static void Main(string[] args)
        {
            //minimizar janela


            IntPtr                          winHandle                          =
System.Diagnostics.Process.GetCurrentProcess().MainWindowHandle;
            ShowWindow(winHandle, SW_MINIMIZE);

            MonoDV mono = new MonoDV();
            mono.statsram();
            mono.statshd();
            mono.statscddrive();
            mono.statspointingdevice();
            mono.statskeyboard();
```

```
            mono.statsmonitor();
            mono.statsnetworkadapter();
            mono.statscpu();
            mono.statsmotherboard();
            mono.escreveBanco();

            Thread t = new Thread(new ThreadStart(mono.EscreveDesliga));
            t.Start();
        }
//coleta de informações sobre memória RAM
        private void statsram()
        {
            ManagementObjectSearcher busca = new
ManagementObjectSearcher("Select * from Win32_PhysicalMemory");
            foreach (ManagementObject result in busca.Get())
            {
                totalram +=
Convert.ToInt64(result.GetPropertyValue("Capacity"));
                rampentes++;
            }
            Console.WriteLine("RAM Total: {0}", totalram / rammb);
            Console.WriteLine("Qtd de Pentes: {0}", rampentes);
            Console.WriteLine("");
        }
//coleta de informações sobre discos rígidos
        private void statshd()
        {
            //Dados HD - nao pega USB
            ManagementObjectSearcher busca = new
ManagementObjectSearcher("SELECT * FROM Win32_DiskDrive WHERE
InterfaceType!='USB'");
//commando Interface!=’USB’ faz com que não seja coletadas dados sobre
drives USB
            foreach (ManagementObject result in busca.Get())
            {
                hdcap += Convert.ToInt64(result.GetPropertyValue("Size"));

                hds++;
```

```
                hdmodelo                                                =
Convert.ToString(result.GetPropertyValue("Model"));
                Console.WriteLine("Nome do Disco Rigido: {0}", hdmodelo);
            }
            Console.WriteLine("Capacidade de Armazenamento de Disco Rigido:
{0}", hdcap / hdmb);
            Console.WriteLine("Qtd de HDs: {0}", hds);
            Console.WriteLine("");
        }
//coleta de informações sobre drive de CD
        private void statscddrive()
        {
            ManagementObjectSearcher           busca           =           new
ManagementObjectSearcher("SELECT * FROM Win32_CDROMDrive");
            foreach (ManagementObject result in busca.Get())
            {
                cdnome = Convert.ToString(result.GetPropertyValue("Name"));
            }
            Console.WriteLine("Nome do drive de CD: {0}", cdnome);
            Console.WriteLine("");
        }
//coleta de informações sobre Mouse
        private void statspointingdevice()
        {
            ManagementObjectSearcher           busca           =           new
ManagementObjectSearcher("SELECT * FROM Win32_PointingDevice");
            foreach (ManagementObject result in busca.Get())
            {
                mousefab                                                 =
Convert.ToString(result.GetPropertyValue("Manufacturer"));
                mouseid                                                  =
Convert.ToString(result.GetPropertyValue("DeviceId"));
            }
            Console.WriteLine("Fabricante Mouse: {0}", mousefab);
            Console.WriteLine("Identificacao: {0}", mouseid);
            Console.WriteLine("");
        }
//coleta de informações sobre teclado
        private void statskeyboard()
```

```
        {
            ManagementObjectSearcher busca = new
ManagementObjectSearcher("SELECT * FROM Win32_Keyboard");
            foreach (ManagementObject result in busca.Get())
            {
                keytipo = Convert.ToString(result.GetPropertyValue("Name"));
                keyid =
Convert.ToString(result.GetPropertyValue("DeviceId"));
            }
            Console.WriteLine("Tipo de Teclado: {0}", keytipo);
            Console.WriteLine("Identificacao: {0}", keyid);
            Console.WriteLine("");
        }
//coleta de informações sobre monitor
        private void statsmonitor()
        {
            ManagementObjectSearcher busca = new
ManagementObjectSearcher("SELECT * FROM Win32_DesktopMonitor");
            foreach (ManagementObject result in busca.Get())
            {
                montipo =
Convert.ToString(result.GetPropertyValue("MonitorManufacturer"));
                monid =
Convert.ToString(result.GetPropertyValue("MonitorType"));
                monstat =
Convert.ToString(result.GetPropertyValue("Status"));
            }
            Console.WriteLine("Tipo de Monitor: {0}", montipo);
            Console.WriteLine("Identificacao: {0}", monid);
            Console.WriteLine("Status: {0}", monstat);
            Console.WriteLine("");
        }
//coleta de informações sobre adaptador de rede
        private void statsnetworkadapter()
        {
            ManagementObjectSearcher busca = new
ManagementObjectSearcher("SELECT * FROM Win32_NetworkAdapter WHERE
NetConnectionStatus=2");
```

```
//NetConnectionStatus=2 faz com que informações só sejam coletadas de
adaptadores de rede com conexão ativa
            foreach (ManagementObject result in busca.Get())
            {
                mac                                                    =
Convert.ToString(result.GetPropertyValue("MACAddress"));
                netnome = Convert.ToString(result.GetPropertyValue("Name"));
            }
            Console.WriteLine("Endereco MAC: {0}", mac);
            Console.WriteLine("Nome: {0}", netnome);
            Console.WriteLine("");
        }
//coleta de informações sobre processador
        private void statscpu()
        {
            ManagementObjectSearcher          busca          =          new
ManagementObjectSearcher("Select * from Win32_Processor");
            foreach (ManagementObject result in busca.Get())
            {
                velocidade                                             =
Convert.ToInt64(result.GetPropertyValue("CurrentClockSpeed"));
                familia                                                =
Convert.ToInt64(result.GetPropertyValue("Family"));
                cpuid                                                  =
Convert.ToString(result.GetPropertyValue("DeviceId"));
                cpufabricante                                          =
Convert.ToString(result.GetPropertyValue("Manufacturer"));
                ;
            }
            Console.WriteLine("Fabicante: {0}", cpufabricante);
            Console.WriteLine("Familia: {0}", familia);
            Console.WriteLine("Identificacao: {0}", cpuid);
            Console.WriteLine("Velocidade em Hz: {0}", velocidade);
            Console.WriteLine("");
        }
//coleta de informações sobre placa mãe
        private void statsmotherboard()
        {
```

```
            ManagementObjectSearcher busca = new
ManagementObjectSearcher("SELECT * FROM Win32_BaseBoard");
            foreach (ManagementObject result in busca.Get())
            {
                mbfabricante =
Convert.ToString(result.GetPropertyValue("Manufacturer"));
                mbmodelo =
Convert.ToString(result.GetPropertyValue("Product"));
                mbsn =
Convert.ToString(result.GetPropertyValue("SerialNumber"));
            }
            Console.WriteLine("Fabricante Placa Mae: {0}", mbfabricante);
            Console.WriteLine("Modelo: {0}", mbmodelo);
            Console.WriteLine("Numero de Serie: {0}", mbsn);
            Console.WriteLine("");

        }
//conexão com tabela SQL
        private void escreveBanco()
        {
            //string de conexao
            SqlConnection Conexao = new SqlConnection(@"Network
Library=DBMSSOCN;Data Source=192.168.1.100\SQLEXPRESS;database=monodv;User
id=david;Password=1234;");

            try
            {
                Conexao.Open();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }


            String strDados = "INSERT INTO InfoClient values (" + totalram +
"," + rammb + "," + rampentes + "," + hdcap + "," + hdmb + ",'" + hdmodelo +
"'," + hds + ",'" + cdnome + "','" + mousefab + "','" + mouseid + "','" +
keytipo + "','" + keyid + "','" + montipo + "','" + monid + "','" + monstat
```

```
+ "','" + mac + "','" + netnome + "','" + cpufabricante + "','" + cpuid +
"'," + velocidade + "," + familia + ",'" + mbfabricante + "','" + mbmodelo +
"','" + mbsn + "',GETDATE(),GETDATE())";

            try
            {
                SqlCommand query = new SqlCommand(strDados, Conexao);
                query.ExecuteNonQuery();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
            Console.WriteLine("{0}", strDados);


            try
            {
                SqlDataReader leitor = null;
                SqlCommand comando = new SqlCommand("select top 1 id from
InfoClient order by id desc", Conexao);
                leitor = comando.ExecuteReader();
                while (leitor.Read())
                {
                    idatual = leitor["id"].ToString();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }

        }
//permite verificação do fim de sessão
        private void EscreveDesliga()
        {
            SqlConnection Conexao = new SqlConnection(@"Network
Library=DBMSSOCN;Data Source=192.168.1.100\SQLEXPRESS;database=monodv;User
id=david;Password=1234;");
```

```
            try
            {
                Conexao.Open();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
            while (true)
            {


                try
                {
                    SqlCommand query = new SqlCommand("update InfoClient set
tempodesliga = GETDATE() where id = " + idatual, Conexao);
                    query.ExecuteNonQuery();
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.ToString());
                }
                Console.WriteLine("ID do Registro: {0}", idatual);
                Thread.Sleep(3000);

            }
        }

    }
}
```

Interface gráfica de usuário

Default.aspx responsável por *Login* de usuário

```
<%@ Page Language="C#" AutoEventWireup="true"    CodeFile="Default.aspx.cs"
Inherits="_Default" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
    <div style="text-align: center">
        <strong><span       style="font-size:       small;       font-family:
Verdana">Inventário Automático de Hardware (IAH)</span></strong><br />
        <br />
        <asp:Login     ID="Login1"     runat="server"     BackColor="#F7F6F3"
BorderColor="#E6E2D8"          BorderPadding="4"          BorderStyle="Solid"
BorderWidth="1px"     DestinationPageUrl="~/home.aspx"     FailureText="Login
inválido. Por favor tente novamente." Font-Names="Verdana" Font-Size="0.8em"
ForeColor="#333333"                                  PasswordLabelText="Senha:"
PasswordRequiredErrorMessage="Senha        é        campo        requerido."
RememberMeText="Lembrar-me"                       UserNameLabelText="Usuário:"
UserNameRequiredErrorMessage="Usuário é campo requerido." Width="296px">
            <TitleTextStyle   BackColor="#5D7B9D"   Font-Bold="True"   Font-
Size="0.9em" ForeColor="White" />
            <InstructionTextStyle Font-Italic="True" ForeColor="Black" />
            <TextBoxStyle Font-Size="0.8em" />
            <LoginButtonStyle    BackColor="#FFFBFF"    BorderColor="#CCCCCC"
BorderStyle="Solid" BorderWidth="1px"
                Font-Names="Verdana"   Font-Size="0.8em"   ForeColor="#284775"
/>
        </asp:Login>

    </div>
    </form>
</body>
</html>
```

home.aspx tela com os cinco links principais

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="home.aspx.cs"
Inherits="home" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="estilo.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
    <div style="text-align: center">
        <span style="font-size: small"><strong>Inventário Automático de
Hardware (IAH)</strong></span><br />
        <br />
        <a href="geral.aspx">Relatório Geral - Tempo</a><br />
        <a href="completo.aspx">Relatório Geral - Hardware<br />
        </a>
        <a href="form_maquina.aspx">Relatório por Máquina</a><br />
        <a href="dados.aspx">Dados Gerais do Sistema</a><br />
        <a href="alertas.aspx">Possíveis Alterações de Hardware</a><br />
    </div>
    </form>
</body>
</html>
```

geral.aspx Relatório Geral por MAC com tempo liga, desliga, duração de sessão e detalhes de registro

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="maquina.aspx.cs"
Inherits="maquina" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
```

```
    <title>Untitled Page</title>
    <link href="estilo.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <strong><span  style="font-size:  small">Inventário  Automático  de
Hardware (IAH) - Relatório Geral<br />
        </span></strong><br />
        <asp:SqlDataSource         ID="SqlDataSource1"         runat="server"
ConnectionString="<%$ ConnectionStrings:MonoDVConnectionString %>"
            SelectCommand="select
mac,tempoliga,tempodesliga,cast(datediff(second,tempoliga,tempodesliga)    as
varchar)+ ' segundo(s)' as tempototal,id from InfoClient order by id desc">
        </asp:SqlDataSource>

    </div>
        <asp:GridView              ID="GridView1"              runat="server"
AutoGenerateColumns="False" CellPadding="4"
            DataSourceID="SqlDataSource1"  ForeColor="#333333"  Width="780px"
AllowPaging="True" AllowSorting="True">
            <FooterStyle         BackColor="#507CD1"         Font-Bold="True"
ForeColor="White" />
            <Columns>
                <asp:CommandField  CancelText="Cancelar"  DeleteText="Deletar"
EditText="Editar" SelectText="&gt;"
                    ShowSelectButton="True" />
                <asp:HyperLinkField              DataNavigateUrlFields="mac"
DataNavigateUrlFormatString="~/maquina.aspx?mac={0}"
                    DataTextField="mac" HeaderText="Endere&#231;o MAC" />
                <asp:BoundField  DataField="tempoliga"  HeaderText="Tempo  -
Liga" SortExpression="tempoliga" />
                <asp:BoundField DataField="tempodesliga" HeaderText="Tempo -
Desliga" SortExpression="tempodesliga" />
                <asp:BoundField   DataField="tempototal"   HeaderText="Tempo
Total" ReadOnly="True" SortExpression="tempototal" />
                <asp:HyperLinkField               DataNavigateUrlFields="id"
DataNavigateUrlFormatString="~/detalhes.aspx?id={0}"       ShowHeader="False"
Target="_self" Text="Detalhes do Registro" />
```

```
            </Columns>
            <RowStyle BackColor="#EFF3FB" />
            <EditRowStyle BackColor="#2461BF" />
            <SelectedRowStyle      BackColor="#D1DDF1"      Font-Bold="True"
ForeColor="#333333" />
            <PagerStyle         BackColor="#2461BF"         ForeColor="White"
HorizontalAlign="Center" />
            <HeaderStyle         BackColor="#507CD1"         Font-Bold="True"
ForeColor="White" />
            <AlternatingRowStyle BackColor="White" />
        </asp:GridView>
    </form>
</body>
</html>
```

completo.aspx Relatório complete com todos os dados escritos no banco de dados

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="completo.aspx.cs"
Inherits="completo" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="estilo.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <strong><span style="font-size: small">Inventário Automático de
Hardware (IAH) - Possíveis
            Alterações de Hardware<br />
            <br />
            <asp:SqlDataSource      ID="SqlDataSource1"      runat="server"
ConnectionString="<%$ ConnectionStrings:MonoDVConnectionString %>"
```

```
                SelectCommand="SELECT * FROM [InfoClient] ORDER BY [id]
desc"></asp:SqlDataSource>
        </span></strong>

    </div>
        <asp:GridView ID="GridView1" runat="server" AllowPaging="True"
AllowSorting="True"
            CellPadding="4"                    DataSourceID="SqlDataSource1"
ForeColor="#333333">
            <FooterStyle         BackColor="#5D7B9D"         Font-Bold="True"
ForeColor="White" />
            <Columns>
                <asp:CommandField SelectText="&gt;" ShowSelectButton="True"
/>
            </Columns>
            <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
            <EditRowStyle BackColor="#999999" />
            <SelectedRowStyle      BackColor="#E2DED6"      Font-Bold="True"
ForeColor="#333333" />
            <PagerStyle         BackColor="#284775"         ForeColor="White"
HorizontalAlign="Center" />
            <HeaderStyle         BackColor="#5D7B9D"         Font-Bold="True"
ForeColor="White" />
            <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
        </asp:GridView>
    </form>
</body>
</html>
```

form_maquina.aspx Lista de todos os endereços MAC registras em banco de dados

```
<%@           Page           Language="C#"           AutoEventWireup="true"
CodeFile="form_maquina.aspx.cs" Inherits="form_maquina" %>

<!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head runat="server">
    <title>Untitled Page</title>
    <link href="estilo.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:SqlDataSource         ID="SqlDataSource1"         runat="server"
ConnectionString="<%$ ConnectionStrings:MonoDVConnectionString %>"
            SelectCommand="select          distinct          mac          from
InfoClient"></asp:SqlDataSource>

    </div>
        <asp:GridView              ID="GridView1"              runat="server"
AutoGenerateColumns="False" CellPadding="4"
            DataSourceID="SqlDataSource1"                 ForeColor="#333333"
GridLines="None" Width="290px">
            <FooterStyle         BackColor="#5D7B9D"         Font-Bold="True"
ForeColor="White" />
            <Columns>
                <asp:HyperLinkField               DataNavigateUrlFields="mac"
DataNavigateUrlFormatString="~/maquina.aspx?mac={0}"
                    DataTextField="mac" HeaderText="Endere&#231;os MAC" />
            </Columns>
            <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
            <EditRowStyle BackColor="#999999" />
            <SelectedRowStyle      BackColor="#E2DED6"      Font-Bold="True"
ForeColor="#333333" />
            <PagerStyle         BackColor="#284775"         ForeColor="White"
HorizontalAlign="Center" />
            <HeaderStyle         BackColor="#5D7B9D"         Font-Bold="True"
ForeColor="White" />
            <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
        </asp:GridView>
    </form>
</body>
</html>
```

maquina.aspx Tela com todos os registros de um MAC específico

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="maquina.aspx.cs"
Inherits="maquina" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="estilo.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <strong><span style="font-size: small">Inventário Automático de
Hardware (IAH) - Relatório Geral<br />
        </span></strong><br />
        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:MonoDVConnectionString %>"
            SelectCommand="SELECT [id], [mac], [tempoliga], [tempodesliga]
FROM [InfoClient] WHERE ([mac] = @mac) order by [id] desc">
            <SelectParameters>
                <asp:QueryStringParameter Name="mac" QueryStringField="mac"
Type="String" />
            </SelectParameters>
        </asp:SqlDataSource>

    </div>
        <asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False" CellPadding="4"
            DataSourceID="SqlDataSource1" ForeColor="#333333" Width="662px"
AllowPaging="True" AllowSorting="True">
            <FooterStyle BackColor="#507CD1" Font-Bold="True"
ForeColor="White" />
            <Columns>
                <asp:CommandField SelectText="&gt;" ShowSelectButton="True"
/>
                <asp:HyperLinkField DataNavigateUrlFields="mac"
DataNavigateUrlFormatString="~/maquina.aspx?mac={0}"
                    DataTextField="mac" HeaderText="Endere&#231;o MAC" />
                <asp:BoundField DataField="tempoliga" HeaderText="Tempo -
Liga" SortExpression="tempoliga" />
                <asp:BoundField DataField="tempodesliga" HeaderText="Tempo -
Desliga" SortExpression="tempodesliga" />
                <asp:HyperLinkField DataNavigateUrlFields="id"
DataNavigateUrlFormatString="~/detalhes.aspx?id={0}" ShowHeader="False"
Target="_self" Text="Detalhes do Registro" />
            </Columns>
            <RowStyle BackColor="#EFF3FB" />
            <EditRowStyle BackColor="#2461BF" />
            <SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True"
ForeColor="#333333" />
            <PagerStyle BackColor="#2461BF" ForeColor="White"
HorizontalAlign="Center" />
            <HeaderStyle BackColor="#507CD1" Font-Bold="True"
ForeColor="White" />
            <AlternatingRowStyle BackColor="White" />
```

```
        </asp:GridView>
    </form>
</body>
</html>
```

dados.aspx Informações sobre duração média de sessão e número total de registros

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="dados.aspx.cs" Inherits="dados" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="estilo.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
    <div style="text-align: center">
        <strong><span style="font-size: small">Inventário Automático de Hardware (IAH)</span></strong><asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:MonoDVConnectionString %>"
            SelectCommand="select avg(datediff(minute,tempoliga,tempodesliga)) as media from InfoClient">
        </asp:SqlDataSource>
        <br />
          
        <asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False" CellPadding="4"
            DataSourceID="SqlDataSource1" ForeColor="#333333" GridLines="None" Height="4px"
            Width="252px"
OnPageIndexChanging="DetailsView1_PageIndexChanging">
            <FooterStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
            <CommandRowStyle BackColor="#E2DED6" Font-Bold="True" />
```

```
            <EditRowStyle BackColor="#999999" />
            <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
            <PagerStyle         BackColor="#284775"         ForeColor="White"
HorizontalAlign="Center" />
            <Fields>
                <asp:BoundField      DataField="media"      HeaderText="Tempo
m&#233;dio de uso (minutos):" ReadOnly="True"
                    SortExpression="media" />
            </Fields>
            <FieldHeaderStyle BackColor="#E9ECF1" Font-Bold="True" />
            <HeaderStyle         BackColor="#5D7B9D"         Font-Bold="True"
ForeColor="White" />
            <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
        </asp:DetailsView>
        <br />
        <asp:SqlDataSource         ID="SqlDataSource2"         runat="server"
ConnectionString="<%$ ConnectionStrings:MonoDVConnectionString %>"
            SelectCommand="select  count(id)  as  'Total  de  Registros:'  from
InfoClient"></asp:SqlDataSource>
           
        <asp:DetailsView  ID="DetailsView2"  runat="server"  CellPadding="4"
DataSourceID="SqlDataSource2"
            ForeColor="#333333"        GridLines="None"        Height="11px"
Width="252px">
            <FooterStyle         BackColor="#5D7B9D"         Font-Bold="True"
ForeColor="White" />
            <CommandRowStyle BackColor="#E2DED6" Font-Bold="True" />
            <EditRowStyle BackColor="#999999" />
            <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
            <PagerStyle         BackColor="#284775"         ForeColor="White"
HorizontalAlign="Center" />
            <FieldHeaderStyle BackColor="#E9ECF1" Font-Bold="True" />
            <HeaderStyle         BackColor="#5D7B9D"         Font-Bold="True"
ForeColor="White" />
            <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
        </asp:DetailsView>

    </div>
    </form>
```

```
</body>
</html>
```

alertas.aspx Página que informa qualquer alteração de *hardware*

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="maquina.aspx.cs" Inherits="maquina" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="estilo.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <strong><span style="font-size: small">Inventário Automático de Hardware (IAH) - Possíveis
            Alterações de Hardware</span></strong><br />
        <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:MonoDVConnectionString %>"
            SelectCommand="SELECT mac, &#13;&#10; ram = CASE&#13;&#10; WHEN count(distinct(totalRam)) = 1 THEN 'OK'&#13;&#10; ELSE 'Alerta'&#13;&#10; END&#13;&#10; &#13;&#10;&#9;&#9;,hdcap = CASE&#13;&#10; WHEN count(distinct(hdcap)) = 1 THEN 'OK'&#13;&#10; ELSE 'Alerta'&#13;&#10; END&#13;&#10;&#13;&#10; ,cpuid = CASE&#13;&#10; WHEN count(distinct(cpuid)) = 1 THEN 'OK'&#13;&#10; ELSE 'Alerta'&#13;&#10; END&#13;&#10;&#13;&#10; ,mbsn = CASE&#13;&#10; WHEN count(distinct(mbsn)) = 1 THEN 'OK'&#13;&#10; ELSE 'Alerta'&#13;&#10; END&#13;&#10;&#13;&#10;&#13;&#10;FROM InfoClient group by mac">
        </asp:SqlDataSource>
        <br />
```

```
    </div>
        <asp:GridView   ID="GridView1"   runat="server"   AllowPaging="True"
AllowSorting="True"
            AutoGenerateColumns="False"                        CellPadding="4"
DataSourceID="SqlDataSource1" ForeColor="#333333"
            Width="662px">
            <FooterStyle         BackColor="#507CD1"         Font-Bold="True"
ForeColor="White" />
            <Columns>
                <asp:HyperLinkField               DataNavigateUrlFields="mac"
DataNavigateUrlFormatString="~/maquina.aspx?mac={0}"
                    DataTextField="mac" HeaderText="Endere&#231;o MAC" />
                <asp:BoundField   DataField="ram"   HeaderText="Mem&#243;ria
RAM" ReadOnly="True" SortExpression="ram" />
                <asp:BoundField  DataField="hdcap"  HeaderText="Capacidade  do
HD" ReadOnly="True" SortExpression="hdcap" />
                <asp:BoundField   DataField="cpuid"   HeaderText="ID   CPU"
ReadOnly="True" SortExpression="cpuid" />
                <asp:BoundField   DataField="mbsn"   HeaderText="ID   Placa
M&#227;e" ReadOnly="True" SortExpression="mbsn" />
            </Columns>
            <RowStyle BackColor="#EFF3FB" />
            <EditRowStyle BackColor="#2461BF" />
            <SelectedRowStyle      BackColor="#D1DDF1"      Font-Bold="True"
ForeColor="#333333" />
            <PagerStyle         BackColor="#2461BF"         ForeColor="White"
HorizontalAlign="Center" />
            <HeaderStyle         BackColor="#507CD1"         Font-Bold="True"
ForeColor="White" />
            <AlternatingRowStyle BackColor="White" />
        </asp:GridView>
    </form>
</body>
</html>
```

detalhes.aspx Página com todos os dados de *hardware* de um registro específico

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="detalhes.aspx.cs"
Inherits="Default2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="estilo.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
    <div align="left">
        <strong><span style="font-size: small">Inventário Automático de
Hardware (IAH) - Detalhes
            de Registro</span></strong><br />
        <br />
        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:MonoDVConnectionString %>"
            SelectCommand="SELECT * FROM [InfoClient] WHERE ([id] = @id)">
            <SelectParameters>
                <asp:QueryStringParameter Name="id" QueryStringField="id"
Type="Int32" />
            </SelectParameters>
        </asp:SqlDataSource>
        <asp:DetailsView ID="DetailsView1" runat="server"
AutoGenerateRows="False" CellPadding="4"
            DataKeyNames="id" DataSourceID="SqlDataSource1"
ForeColor="#333333" GridLines="None"
            Height="50px" Width="401px">
            <FooterStyle BackColor="#507CD1" Font-Bold="True"
ForeColor="White" />
            <CommandRowStyle BackColor="#D1DDF1" Font-Bold="True" />
            <EditRowStyle BackColor="#2461BF" />
            <RowStyle BackColor="#EFF3FB" />
            <PagerStyle BackColor="#2461BF" ForeColor="White"
HorizontalAlign="Center" />
            <Fields>
```

```
                <asp:BoundField       DataField="id"       HeaderText="id"
InsertVisible="False" ReadOnly="True"
                    SortExpression="id" />
                <asp:BoundField  DataField="totalram"  HeaderText="RAM  Total"
SortExpression="totalram" />
                <asp:BoundField  DataField="rampentes"  HeaderText="Total  de
Pentes" SortExpression="rampentes" />
                <asp:BoundField  DataField="hdmodelo"  HeaderText="Modelo  do
HD" SortExpression="hdmodelo" />
                <asp:BoundField  DataField="hds"  HeaderText="Quantidade  de
HDs" SortExpression="hds" />
                <asp:BoundField DataField="cdnome" HeaderText="Nome do Drive
de CD" SortExpression="cdnome" />
                <asp:BoundField  DataField="mousefab"  HeaderText="Fabricante
Mouse" SortExpression="mousefab" />
                <asp:BoundField DataField="mouseid" HeaderText="ID do Mouse"
SortExpression="mouseid" />
                <asp:BoundField   DataField="keytipo"   HeaderText="Tipo   de
Teclado" SortExpression="keytipo" />
                <asp:BoundField DataField="keyid" HeaderText="ID do Teclado"
SortExpression="keyid" />
                <asp:BoundField   DataField="montipo"   HeaderText="Tipo   de
Monitor" SortExpression="montipo" />
                <asp:BoundField DataField="monid" HeaderText="ID do Monitor"
SortExpression="monid" />
                <asp:BoundField  DataField="monstat"  HeaderText="Status  do
Monitor" SortExpression="monstat" />
                <asp:BoundField   DataField="mac"   HeaderText="Endere&#231;o
MAC" SortExpression="mac" />
                <asp:BoundField     DataField="netnome"     HeaderText="Nome
Adaptador de Rede" SortExpression="netnome" />
                <asp:BoundField                     DataField="cpufabricante"
HeaderText="Fabricante do Processador"
                    SortExpression="cpufabricante" />
                <asp:BoundField    DataField="cpuid"    HeaderText="ID    do
Processador" SortExpression="cpuid" />
                <asp:BoundField                         DataField="velocidade"
HeaderText="Velocidade CPU" SortExpression="velocidade" />
```

```
                <asp:BoundField                          DataField="familia"
HeaderText="Fam&#237;lia CPU" SortExpression="familia" />
                <asp:BoundField                      DataField="mbfabricante"
HeaderText="Fabricante Placa-m&#227;e" SortExpression="mbfabricante" />
                <asp:BoundField    DataField="mbmodelo"    HeaderText="Modelo
placa-m&#227;e" SortExpression="mbmodelo" />
                <asp:BoundField   DataField="mbsn"   HeaderText="No.   serial
Placa-m&#227;e" SortExpression="mbsn" />
                <asp:BoundField                          DataField="tempoliga"
HeaderText="Hor&#225;rio - Liga" SortExpression="tempoliga" />
                <asp:BoundField                      DataField="tempodesliga"
HeaderText="Hor&#225;rio - Desliga" SortExpression="tempodesliga" />
            </Fields>
            <FieldHeaderStyle BackColor="#DEE8F5" Font-Bold="True" />
            <HeaderStyle         BackColor="#507CD1"         Font-Bold="True"
ForeColor="White" />
            <AlternatingRowStyle BackColor="White" />
        </asp:DetailsView>
        </div>
    </form>
</body>
</html>
```