Centro Universitário de Brasília – UniCEUB
Faculdade Tecnologia e Ciências Sociais Aplicadas –FATECS
Curso de Engenharia de Computação
Projeto Final

AUTOMAÇÃO DE INVENTÁRIO DE SOFTWARE

FERNANDO JOSÉ SILVESTRE DE FARIA FILHO
2021811/1

Brasília – DF, Junho de 2009

FERNANDO JOSÉ SILVESTRE DE FARIA FILHO

AUTOMAÇÃO DE INVENTÁRIO DE SOFTWARE

Monografia apresentada ao Curso de Engenharia da Computação, como requisito parcial para obtenção do grau de Engenharia de Computação.

Orientador: Prof. M.Sc. Antonio José Gonçalves Pinto

BRASÍLIA/DF

1º SEMESTRE DE 2009

AGRADECIMENTOS

Primeiramente a Deus, pela oportunidade que me foi dada de cursar o ensino superior, o que é um privilélio para poucos.

Aos meus pais, Fernando e Márcia, e a minha avó, Terezinha, por todo o incentivo e apoio recebidos. Especialmente a minha mãe e a minha avó que já faleceram e que sempre me serviram como modelo de amor e carinho.

A minha namorada, Larissa, por todo o seu apoio, carinho e compreensão em todas as vezes que foi preciso abdicar de um momento de lazer para que eu pudesse dar continuidade as atividades desse trabalho de conclusão de curso.

Aos meu amigos e companheiros de curso com os quais eu passei um grande período durante graduação e que sempre me ajudaram nos momentos de dificuldades.

A todos os meus colegas de trabalho pelo apoio e incentivo recebidos.

Ao meu professor orientador, M.Sc. Antonio José Gonçalves Pinto, por todo conhecimento repassado, por sua paciência e principalmente pela sua orientação na condução desse trabalho.

A todo o corpo docento do UniCEUB, por todo apoio e pelo conhecimento transmitido so longo do curso.

RESUMO

Esse projeto demonstra um estudo que possibilitou a criação de uma solução com o objetivo de auxiliar no processo de administração de redes. Tal ferramenta é responsável por fazer a geração automática do inventário de softwares de um parque computacional, assim como a validação de softwares instalados em uma determinada rede de computadores. Através de um interface de usuário o administrador de rede pode ter acesso a todas as informações coletadas de forma simples e rápida, facilitando suas tarefas. Todo o processo de desenvolvimento foi feito usando o *Microsoft Visual Studio 2008 Professional*, e como linguagem de programação foi escolhido o CSharp (C#). Para a interface de usuário (nesse caso uma interface web) foi utilizado o ASP.NET e como banco de dados, responsável pelo armazenamento das informações, foi usado o *Microsoft SQL Server 2008 Express*. Pôde-se perceber que houve um acerto na escolha das tecnologias e ferramentas utilizadas tornando possível a criação da solução proposta atendendo aos requisitos desejados.

Palavras-chave: administração de redes, gerência de redes, inventário de software.

ABSTRACT

This project demonstrates a study that enabled the creation of a tool with the aim of assisting in the administration of networks. This tool is responsible for doing the automatic generation of inventory of software of a computacional park, as well as the validation software installed on a network computers. Through a user interface, the network administrator can have access to all information collected from a simple and rapid way, facilitating their tasks.

The whole process of development was done using Microsoft Visual Studio 2008 Professional, and as the programming language was chosen Csharp (C #). For the user interface (in this case a web interface) was used as the ASP.NET, and for the database, used for storage of information, was used the Microsoft SQL Server 2008 Express.

Key-Words: Network Administration; Network Managing; Inventory of Software.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS ABREVIATUTAS

C# - C Sarp

SGBD - Sistema Gerenciador de Bancos de Dados

DMTF - Distributed Management Task Force

GUI - Graphical User Interface

HP - Hewlett-Packard

HTML - HyperText Markup Language

IBM - International Business Machines

MAC - Media Access Control

RAM - Random Access Memory

SO - Sistema Operacional

SQL - Structured Query Language

MSSQLS - Microsoft SQL Server

TCP/IP - Transmission Control Protocol/Internet Protocol

TI - Tecnologia da Informação

USB - Universal Serial Bus

VC# - Visual C Sharp

VS - Visual Studio

VS2008 - Visual Studio 2008

CAPÍTULO 1 – INTRODUÇÃO

1.1. MOTIVAÇÃO

O avanço da tecnologia dos últimos anos reduziu o custo dos recursos computacionais, os tornando mais acessíveis a todos. Também por isso as redes de computadores começaram a crescer e a se desenvolver. Com isso as empresas começaram a optar por armazenar as informações em formato eletrônico, deixando de lado o papel. Justamente por esse motivo, as profissões voltadas para administração de redes começaram a ter uma maior importância nas instituições, pois são esses profissionais os responsáveis pelo correto funcionamento dos parques computacionais [WATKINS, 2007].

Um administrador de redes possui uma série de responsabilidades e tarefas, como o bom funcionamento das redes de computadores, o que demanda muito tempo. Juntamente com o exponencial crescimento das redes, proporcionalmente aumentam os problemas, pois quanto maior é o número de ativos de uma rede maiores serão os esforços administrativos e os custos de manutenção. Por isso pode-se perceber a necessidade de uma ferramenta que auxilie nessa administração e que possa diminuir o tempo gasto nas atividades rotineiras do administrador de redes permitindo inclusive que ele tenha uma visão mais completa dos recursos e das necessidades do seu ambiente [WATKINS, 2007].

Surgiu então a necessidade de estudar o processo de gerenciamento e administração de redes de forma que fosse possível o desenvolvimento de ferramentas que pudessem ajudar os profissionais dessa área em suas tarefas rotineiras.

1.2. OBJETIVOS

1.2.1. Gerais

Esse projeto tem como objetivo principal a criação de uma solução que seja capaz de disponibilizar informações de softwares instalados nas máquinas de uma rede de computadores. Devem ser coletadas também informações referentes aos componentes de hardware.

1.2.2. Específicos

a) Geração de um inventário com os softwares instalados nos computadores de uma rede. Esse processo deverá ser feita no *startup* do sistema operacional. Serão armazenadas as seguintes informações:

- Nome.
- Versão.
- Fabricante.
- Data de instalação.
- Diretório de instalação.

b) Deverão ser coletadas informações referentes ao hardware das máquinas:

- **Processador:**
  - Modelo e Fabricante.
- **Placa Mãe:**

- Modelo e Fabricante.

- **Adaptador de rede:**
  - Modelo e Endereço MAC.
- **Disco rígido:**
  - Modelo e capacidade de armazenamento.

c) A ferramenta deverá ser capaz de validar os softwares instalados em cada coleta, informando o administrador de rede sempre que houver a presença de algum programa instalado. A definição dos software não permitidos será feita pelo usuário do sistema, através da interface web.

d) O sistema deverá possuir uma interface onde o administrador possa visualizar de forma simples e rápida todas as informações necessárias.

1.3. ESCOPO DO PROJETO

A ferramenta criada se destina a coletar as informações dos programas instalados e armazenar em um banco de dados, gerando assim um inventário de softwares. Essas informações contemplam nome do software, fabricante do software, data de instalação, versão do aplicativo instalado, diretório de instalação.

Apesar de se tratar de um inventário de software, informações referentes ao hardware também serão coletadas e algumas são de suma importância para o projeto como o endereço MAC, da placa de rede, por se tratar de um identificador único em um ambiente computacional e por isso também é usada no projeto como identificador dos dispositivos. As informações de hardware coletadas são: Processador (modelo e fabricante), Placa mãe (modelo e

fabricante), disco rígido (modelo e capacidade) e placa de rede (endereço MAC). No caso da placa de rede o endereço MAC será fundamental, pois será o identificador único das máquinas.

As informações referentes aos programas instalados serão confrontadas com uma segunda base de dados onde serão verificadas as existências de softwares não permitidos. A criação dessa base faz parte do escopo do projeto e a mesma será alimentada pelo próprio administrador da rede, ou seja, ele se torna o responsável por definir quais softwares não serão permitidos. Após a verificação os administradores serão alertados quanto da existência. O alerta se dará por meio de mensagens de correio eletrônico (e-mail).

1.4. ESTRUTURA DO TRABALHO

Esse trabalho está dividido em 6 capítulos, incluindo a introdução. Abaixo está uma pequena descrição do que será abordado nos próximos capítulos:

- Capítulo 2: *Apresentação do problema* - Mostra o problema que o projeto se propõe a confrontar.

- Capítulo 3: *Gerência de Redes* – Apresenta alguns importantes conceitos a respeito de gerenciamento de redes; traz informações sobre os principais produtos existentes no mercado; e por fim trata das tecnologias usadas para a implementação do projeto.

- Capítulo 4: *Inventário de Softwares* – Mostra o desenvolvimento da solução durante o projeto, apresentando o porquê da escolha das tecnologias e ferramentas usadas; mostra os requisitos para o uso da ferramenta; demonstra o fluxo de funcionamento do

aplicativo; e finalmente os processos de desenvolvimento de forma detalhada.

- Capítulo 5: *Testes e Resultados* – Após a implementação, é nessa parte do trabalho que são apresentados os testes feitos com a ferramenta e quais foram os resultados obtidos.

- Capítulo 6: *Conclusão* – São colocadas todas as considerações finais do trabalho baseadas nos resultados dos testes obtidos e algumas sugestões de futuros trabalhos.

CAPÍTULO 2 – APRESENTAÇÃO DO PROBLEMA

Com a explosão dos avanços tecnológicos e conseqüentemente das redes de computadores, surgiu a necessidade de um gerenciamento mais preciso e eficiente sobre os ativos para que se possa manter o correto funcionamento do parque computacional e por conseqüência atender as expectativas dos usuários.

A administração de uma rede acaba se tornando complexa sem o auxílio de uma ferramenta de apoio que seja capaz de mostrar a realidade do ambiente, e isso pode influenciar diretamente em futuras tomadas de decisões, as vezes influenciando de forma negativa no planejamento das empresas [SOUSA JR, 2005].

Um grande problema hoje nas empresas é o controle sobre o que é instalado nos microcomputadores de sua rede. Mesmo com uma boa política de segurança e de controle de acesso não é raro o sucesso de alguns usuários na instalação de aplicativos não permitidos. Isso pode se tornar uma grande “dor de cabeça”, pois aquele programa instalado pode não apenas tirar a atenção e a produtividade do funcionário, mas também pode ter falhas de segurança que comprometam a rede de uma forma geral.

Baseado no que foi dito acima, notou-se a necessidade de criação de uma ferramenta que fosse capaz de dar suporte ao profissional da área de administração de redes, informando-o constantemente sobre as mudanças realizadas em seu parque, mudanças essas relacionadas aos softwares instalados em cada computador ativo. Através de uma solução como essa muito tempo e esforço dos profissionais, gastos com tarefas manuais para esses levantamentos, seriam poupados.

Esse projeto tem por finalidade principal confrontar essa maior dificuldade, criar uma solução através da qual o usuário possa ter acesso a

informações gerenciais dos computadores e que emita alertas sobre as situações não permitidas. A ferramenta deverá fazer uma coleta detalhada com as informações importantes e armazená-las em um banco de dados para posteriores visualizações e possíveis análises. Deverá ser possível também a configuração de parâmetros específicos que possam atender aos diversos tipos de usuários.

CAPÍTULO 3 – GERÊNCIA DE REDES

3.1. CONCEITOS

No início das redes de computadores o conceito de *Gerência de Redes* nem era comentado. Entretanto com o crescimento acelerado das redes a tarefa de gerenciar tornou-se um tanto quanto complexa [KUROSE; ROSS, 2003], e um administrador/gerente de redes deve, entre outras coisas, conhecer métodos adequados para a administração de redes [SOUSA JR., 2005].

Um administrador de redes atualmente pode contar com uma série de tecnologias (snmp, wbem, wmi, entre outras) e ferramentas (softwares de gerenciamento de inventário, por exemplo) que tornam mais simples a tarefa de administração, mantendo a rede sempre disponível, com o desempenho desejável, de forma estável e com um bom controle sobre os recursos [SOUSA JR., 2005]. Softwares de gerenciamento de inventário de rede possuem grandes bancos de dados o que torna possível a identificação de grande parte dos componentes de hardware e software.

Algumas empresas da área de Tecnologia da Informação já haviam identificado essa necessidade de controle e gerência dos componentes de uma rede, e por isso desenvolveram algumas ferramentas para suprir as necessidades [WATKINS, 2007].

3.2. FERRAMENTAS DISPONÍVEIS NO MERCADO

Abaixo serão citadas algumas ferramentas criadas com a intenção de gerenciar os componentes de uma rede:

- ***CACIC:*** É o resultado de um consórcio entre a Secretaria de Logística Tecnologia da Informação (SLTI) e a Empresa de Tecnologia e Informações da Previdência Social (DATAPREV). Foi desenvolvido pelo Escritório Regional da DATAPREV no Espírito Santo. É o primeiro software público do Governo Federal. A ferramenta é capaz de fornecer um diagnóstico preciso do parque computacional disponibilizando informações como o número de equipamentos, os tipos de softwares utilizados, configurações de hardware, entre outras coisas. O software também pode fornecer informações sobre e a localização física dos equipamentos, ampliando o controle do parque computacional e a segurança na rede. [CACIC, 2009].

- ***IBM Tivoli Configuration Manager:*** Evolução da ferramenta *Tivoli Inventory*, esse software criado pela IBM possui uma funcionalidade de busca automática na rede e com isso a coleta das informações sobre os componentes de hardware e software. Através dessa ferramenta é possível fazer uma varredura na rede criando um inventário de hardware e software do parque computacional. [IBM, 2009].

- ***HP OpenView Enterprise Discovery:*** Ferramenta para fazer inventário automaticamente dos ativos de uma rede. Permite o acompanhamento de qualquer mudança de configuração dos componentes. Esse software permite fazer a função de *discovery* e através das configurações de rede (endereço IP e máscara de rede)

é possível montar uma mapa com a topologia das redes e a criação de grupos de objetos gerenciáveis [HP, 2009].

- ***CiscoWorks Resource Manager Essentials:*** Software para gerenciamento de redes, criado com a intenção de diminuir as chances de falhas humanas eliminando uma série de tarefas manuais associadas a manutenção das redes. Por meio dessa ferramenta é possível fazer a gerência de inventário de hardware e software, bem como a gerência de configuração de dispositivos. [CISCO, 2009].

3.3. TECNOLOGIAS DE GERENCIA DE REDES

3.3.1. WEB-BASED ENTERPRISE MANAGEMENT

O Web-Based Enterprise Management (WBEM) é um conjunto de tecnologias padrões desenvolvido para a gerência e administração de redes através da web [DMTF, 2009b]. Originalmente surgiu de uma iniciativa de grandes empresas de Tecnologia da Informação (TI) em conjunto com Distributed Management Task Force (DMTF) com o intuito de prover informações gerenciais sobre os equipamentos e dispositivos em ambientes computacionais, principalmente os heterogêneos, ou seja, independente das plataformas utilizadas [DMTF, 2009b].

A idéia era a criação de um ambiente aberto de gerência onde os sistemas ou aplicações gerenciadas pudessem ter acesso e controle as informações trocadas entre os agentes de gerenciamento e os dispositivos gerenciados.

3.3.2. WINDOWS MANAGEMENT INSTRUMENTATION

O Windows Management Instrumentation é uma infra-estrutura de gerenciamento de dados e operações para sistemas operacionais Windows [MICROSOFT, 2009a]. Trata-se da implementação da Microsoft do WBEM, e através da qual é possível a criação de aplicações para a automatização de tarefas em equipamentos remotos e também o acesso a dados para outros aplicativos de gerência [WATKINS, 2007].

Para efetuar consultas ao WMI é usada a *Windows Query Language* (WQL), que é uma parte do *Structure Query Language* (SQL). As instruções do WQL que permitem acesso aos dados são similares às cláusulas do SQL (ex: insert, update, select e delete) [GEORGE, DELANO. 2006]..

A composição do WMI está de acordo com o estabelecido pelo DMTF, onde quatro elementos são fundamentais: *Managed Objects*, *WMI Providers* (WP), *WMI Consumers* (WC) e *WMI Infrastructure* (WI).

Managed Objects são dispositivos físicos ou lógicos que podem ser representados através de instancias de classes do WMI. Eles podem ser unidades de discos rígidos, adaptadores de rede, bancos de dados, sistemas operacionais, processos ou serviços [MICROSOFT, 2009c].

WMI Providers são os responsáveis por prover as informações sobre os *Managed Objects*. São eles que fazem os tratamentos das mensagens entre o WMI e os *Managed Objects*. O Windows possui de forma nativa uma série de WP's. A versão mais recente (Windows Vista) já conta com aproximadamente de 100 [MICROSOFT, 2009c]. Entre eles podemos citar o *Win32 Provider*, através do qual é possível ter acesso a informações sobre os SO's Windows bem como aos dispositivos de hardware e software [MICROSOFT, 2009e].

WMI Consumers são aplicativos ou scripts de gerenciamento que utilizam o *WMI*. Eles funcionam basicamente fazendo as requisições dos dados e o processando as informações [MICROSOFT, 2009d].

WMI Infrastructure é um componente do SO Windows e é formado por dois outros componentes: o *WMI Repository* (WR) e o *WMI Service* (WS).

- ***WMI Repository*:** É o repositório dos dados e é gerenciado pelo *Common Information Model Object Manager* (CIMOM). É o CIMOM o responsável por cuidar do tratamento da comunicação entre os WP's e as aplicações de gerência garantindo o correto

armazenamento e por conseqüência a integridade das informações [MICROSOFT, 2009d].

- ***WMI Service*:** funciona como intermediário entre o *WMI Provider*, *WMI Repository* e as aplicações de gerenciamento [MICROSOFT, 2009d].

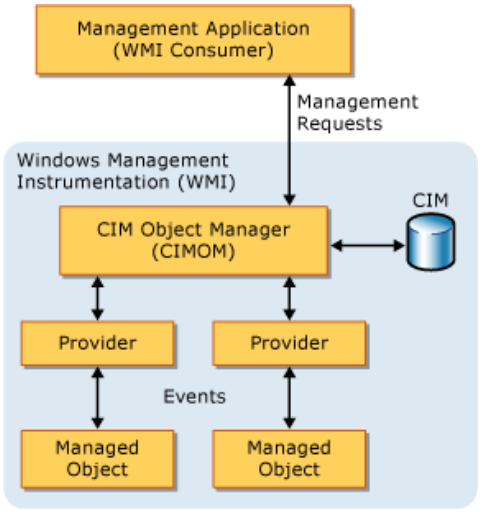


Figura 2. 1 – Interação entre os componentes do WMI

A figura 2.1 mostra como é o relacionamento entre um *WMI Consumer* e o *WMI Repository*, e os *WMI Providers* com os *Managed Objects*. As informações dos *Managed Objects* são coletadas pelo *WP* e armazenadas em um *WR*. O *WMI Consumer* faz uma requisição ao *WR*, e todas os relacionamentos na arquitetura do WMI são feitas por *WMI Services*.

3.4. SERVIÇOS DO WINDOWS

Em sistemas operacionais Windows, um serviço nada mais é do que um programa responsável pela execução de funções específicas e que seja acionado sem que haja necessariamente a intervenção do usuário. Ele pode ser configurado para iniciar automaticamente no *startup* do sistema operacional e rodar em segundo plano, ou pode ser iniciado manualmente quando necessário. Muitos serviços podem ser vistos na listagem de processos através da opção de gerenciamento de tarefas [MICROSOFT, 2009h]. Uma vez instalado, o serviço pode ser gerenciado através do *Services Control Manager* do Windows, presente no painel de controle do sistema operacional.

Um serviço passa por diversos estados internos em sua vida. Primeiro, o serviço é instalado no sistema no qual ele será executado. Depois disso ele deve ser iniciado o que permite que ele comece a funcionar. Ele pode ser iniciado através do *Services Control Manager* ou do *Server Explorer*. Uma vez em execução ele pode existir indefinidamente até que seja interrompido (parado, pausado ou desligamento do computador). Um serviço pode estar em um dos três estados básicos: Executando, Pausado, ou Parado [MICROSOFT, 2009h].

3.5. C SHARP

CShap é uma linguagem de programação orientada a objetos (OO) desenvolvida pela Microsoft para fazer parte da plataforma .Net. Ela foi baseada na linguagem C e possui várias características de outras linguagens, como C++, Java e Delphi. Pode-se dizer que é principal linguagem do *.NET Framework*. O principal criador foi o Anders Hejlsberg [ECMA, 2006].

A Microsoft criou sua própria implementação do C#, denominada de Visual C Sharp (VC#), que já é nativa nas versões mais recentes do Visual Studio. Foi desenvolvida para a criação de aplicações que rodem no *.NET Framework.*

Através do VC# é possível ter acesso a várias informações por meio do WMI já que existe um conjunto de classes preparadas no *.NET Framework.* Para um administrador de rede pode ser interessante armazenar as informações coletadas e para isso pode-se utilizar um banco de dados computacional.

3.6. BANCO DE DADOS COMPUTACIONAL

Um banco de dados pode ser comparado a um armário, um lugar para arquivamento de dados no formato eletrônico; ou seja, pode-se dizer que é um repositório feito para guardar uma coleção de arquivos de computador. Bancos de dados são conjuntos de registros estruturados e que organizados que possibilitam a produção de informação. Normalmente agrupa registros utilizáveis para um mesmo fim [DATE, 2004].

As informações armazenadas nos bancos de dados muito freqüentemente sofrem alterações e isso é controlado pelos sistemas de bancos de dados. Esses são basicamente sistemas para manutenção de registros [DATE, 2004]. As alterações podem ser classificadas como:

- Consulta de registros;
- Criação de novos registros;
- Atualização de registros;
- Exclusão de registros.

Os sistemas de bancos de dados, também chamados de *Database Management Systems* (DBMS) ou *Sistemas de Gerenciamento de Bancos de Dados* (SGBD), são compostos por quatros itens principais: uma linguagem de modelagem que é a responsável por definir a maneira que os bancos de dados serão armazenados pelo SGBD; uma estrutura de dados composta por campos, arquivos e objetos preparados para trabalhar com grande quantidade de informações; uma linguagem de query para que os usuários tenham acesso às informações gravadas; e por fim deve possuir uma forma de efetuar transações e com isso garantir a integridade dos dados [GEORGE, DELANO. 2006].

3.6.1. MICROSOFT SQL SERVER 2008

O *Microsoft Sql Server* é um SGBD criado pela Microsoft. Atualmente sua versão mais atual é a *Microsoft Sql Server 2008*. Com essa nova versão é possível ter uma plataforma de dados confiável e produtiva, que permite a execução de aplicações críticas e que ao mesmo tempo diminua o tempo e o custo para o desenvolvimento [MICROSOFT, 2009f].

Um banco de dados para a gerência de redes é fundamental, entretanto nem sempre os SGBD's possuem interfaces amigáveis para a visualização das informações, ou mesmo não possuem ferramentas suficientes para criação de bons relatórios. Com isso muitas vezes se faz necessária a criação de interfaces customizadas para os usuários.

3.7. INTERFACE GRÁFICA

No mundo da informática, interface gráfica é o meio pelo qual os usuários podem interagir com dispositivos digitais através de elementos

gráficos como ícones, menus e outros elementos visuais. Tais elementos podem ser baseados em *prompts*, janelas, páginas de internet, formulários entre outros [WATKINS, 2007].

Em microcomputadores o controle das interfaces geralmente é feito através de mouses e teclados. Assim o usuário pode interagir com os elementos gráficos desejados e com isso obter o resultado desejado.

Pode-se dizer que os grandes sistemas revolucionários em relação a interface gráfica foram o *Apple Lisa* da Apple Computer lançado em 1983, e o *Windows 3.0* da Microsoft lançado em 1990.

3.7.1. ASP.NET

O ASP.NET é uma tecnologia gratuita que permite aos desenvolvedores a criação de websites dinâmicos. Pode ser usado para o desenvolvimento desde pequenas soluções até grandes aplicações corporativas. Para isso basta que o usuário tenha o .*Net Framework* e o *Visual Web Developer* [ASP.NET, 2009a].

O .*Net Framework* é um conjunto de bibliotecas que interage com o *Common Language Runtime* (CLR), um ambiente de execução que independe da linguagem utilizada [ASP.NET, 2009a].

CAPÍTULO 4 – INVENTÁRIO DE SOFTWARE

4.1 INTRODUÇÃO

Nos últimos anos houve um enorme crescimento das redes de computadores, pois os recursos computacionais se tornaram mais acessíveis. No mercado corporativo grandes redes foram criadas e/ou aumentadas. Entretanto, à medida que o número de elementos de cresce, a administração se torna mais e mais complexa. Por isso os administradores de rede sentiram a necessidade da utilização de ferramentas (programas e aplicativos de computador) para o auxílio na administração.

A ferramenta criada nesse projeto, e descrita nesse capítulo, denominada de *Automação de Inventário de Software* foi criado com o objetivo de auxiliar os administradores de rede em uma tarefa que nem sempre eles podem ter o controle. Através de um serviço, de um banco de dados e de uma interface gráfica ele poderá ter acesso a informações referentes ao hardware e ao software instalado em cada componente de seu parque computacional. E essas informações serão sempre atualizadas quando os equipamentos foram ligados.

É importante ressaltar que a solução aqui desenvolvida não é feita para fins comerciais. O propósito é demonstrar que é possível o desenvolvimento de uma aplicação usando as tecnologias escolhidas para o projeto. O capítulo abordará os seguintes itens:

- O motivo da escolha das tecnologias envolvidas;
- Os requisitos computacionais para o desenvolvimento;
- Fluxo de funcionamento da aplicação;
- Desenvolvimento:
    - Criação do banco de dados;
    - Criação do serviço;
    - Criação da interface gráfica.

4.2. TECNOLOGIAS ESCOLHIDAS

O *WMI* é uma tecnologia para acesso a informações gerenciais em um ambiente corporativo. Através dele é possível fazer o monitoramento e controle dos ativos de uma rede de forma local ou remota [MICROSOFT, 2009b]. Com isso a tecnologia se encaixa nos requisitos para o desenvolvimento do projeto.

A linguagem de programação escolhida para a criação do serviço foi o C Sharp (C#) por meio do Visual C# (VC#). O VC# pode ser utilizado pelo *Visual Studio* 2005 (VS 2005) ou 2008 (VS2008), ou através das versões gratuitas: *Visual C# Express* 2005 ou 2008. Para o desenvolvimento do projeto foi usada a versão do Visual Studio 2008 Professional.

O serviço criado para a obtenção das informações precisa gravá-las em um banco de dados e por isso deve ser usado um SGBD. Pelas várias facilidades de integração com o VS2008 e pela familiaridade do graduando com o produto, foi escolhido o *SQL Server 2008 Express*.

Para a interface gráfica foi usado novamente o VS2008, pois tal ferramenta também foi projetada para trabalhar com ASP.NET.

É importante deixar claro que todas as tecnologias e ferramentas aqui utilizadas são amplamente utilizadas e difundidas no mercado profissional e por isso há uma enorme quantidade de documentação sobre elas [WATKINS, 2007]. Além disso, um fator determinante para a escolha das tecnologias escolhidas é a familiaridade do autor com as mesmas devido a sua experiência profissional.

4.2.1. REQUISITOS COMPUTACIONAIS

Para que a solução funcione de forma adequada, alguns requisitos computacionais devem ser seguidos.

- Para o funcionamento do *WMI* [MICROSOFT, 2009a]:
  - Sistemas operacionais 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 2008:
  - Processador de 1.6 GHz, 384 MB de memória RAM, monitor com a resolução 1024x768 e disco rígido de 5400 RPM; Sistemas operacionais Microsoft Windows XP, Microsoft Windows Server 2003 ou Windows Vista.

- Para o SQL Server 2008 Express [MICROSOFT, 2009g]:
  - Computador com processador Intel ou compatível com 1 GHz ou mais rápido (recomendável 2 GHz ou mais rápido. Apenas um único processador é suportado).

Mínimo de 256 MB de RAM (recomendável 1 GB ou mais), 1 GB de espaço livre no disco rígido; Sistemas operacionais Windows Server 2003 Service Pack 2; Windows Server 2008; Windows Vista; Windows Vista Service Pack 1; Windows XP Service Pack 2; Windows XP Service Pack 3.

- Para o servidor (computador atuando como servidor):
    - As mesmas configurações para o SQL Server 2008 e estar ligado na mesma rede dos computadores que estão atuando como clientes.

- Para os clientes (máquinas ligadas a rede e atuando como clientes):
    - .NET Framework 2.0. Precisam estar conectados na mesma rede do computador que está atuando como servidor.

4.3. FLUXO DA SOLUÇÃO

Na figura 4.1 está representada uma visão do fluxo de funcionamento da solução:

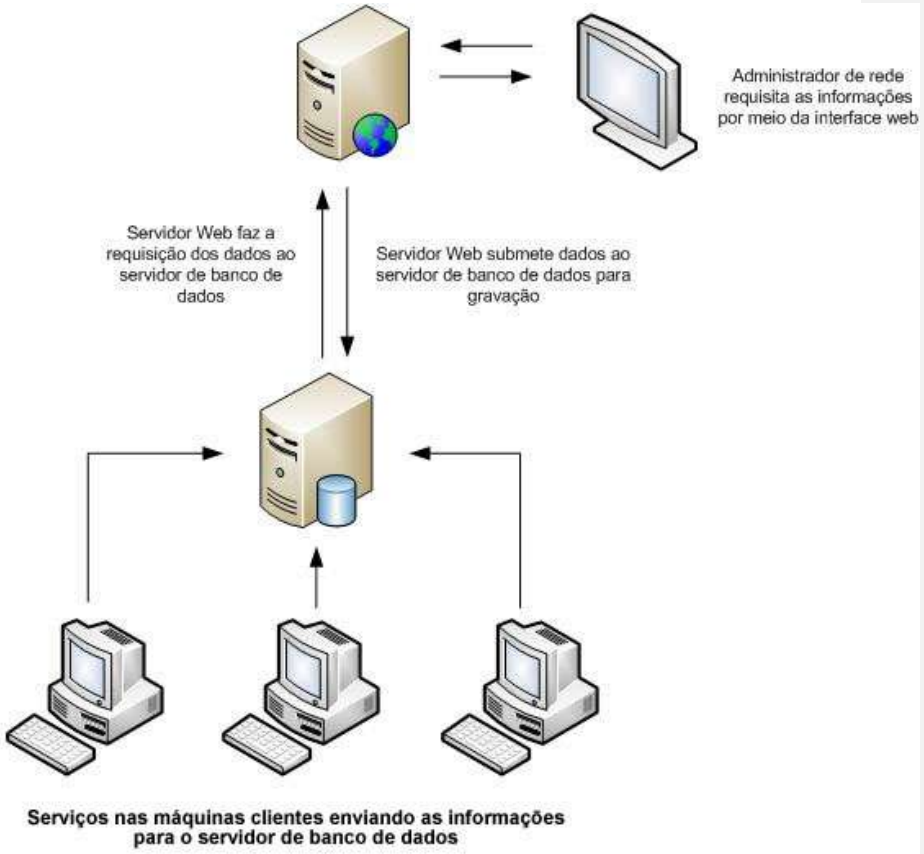


Figura 4. 1 – Fluxo de funcionamento da solução

O serviço criado é responsável por buscar as informações nas máquinas clientes e enviar esses dados para o servidor de bando de dados. No servidor as

informações são tratadas e armazenadas para posterior análise do administrador através da interface web.

Abaixo estão listados e numeradas as etapas do fluxo, onde os clientes já estão configurados com o serviço e o servidor configurado e ativo em um computador.

1º Etapa: O cliente é ligado. Juntamente com a inicialização do sistema operacional o serviço automaticamente é iniciado e com isso já fazendo a coletas das informações e enviando para o servidor de banco de dados.

2º Etapa: Após o envio das informações, é feita uma conexão com o banco de dados por meio da verificação de usuário e senha. Após a verificação, é executada uma query no bando de dados que faz a verificação, tratamento e gravação das informações em tabelas já pré-estruturadas.

3º Etapa: Através de um navegador de internet, o administrador de rede tem acesso a interface gráfica desenvolvida, onde são solicitados seu *login* e *senha*.

4º Etapa: Os dados de login e senha são submetidos e caso exista confiram com os dados já previamente armazenados, o acesso as informações é concedido através de instruções SQL e mostrador pela interface.

4.4. SERVIÇO DO WINDOWS

Abaixo serão listadas as etapas para a criação do serviço do Windows que é o responsável por coletar as informações dos clientes e fazer o envio e gravação no banco de dados. Esse serviço foi feito no Visual Studio 2008, usando a linguagem C#. O acesso as informações é permitido por meio do *WMI*.

1º Etapa – Criação de um novo projeto chamado *WindowsServiceMonografia* utilizando a linguagem C# e usando o *template Windows Service*.

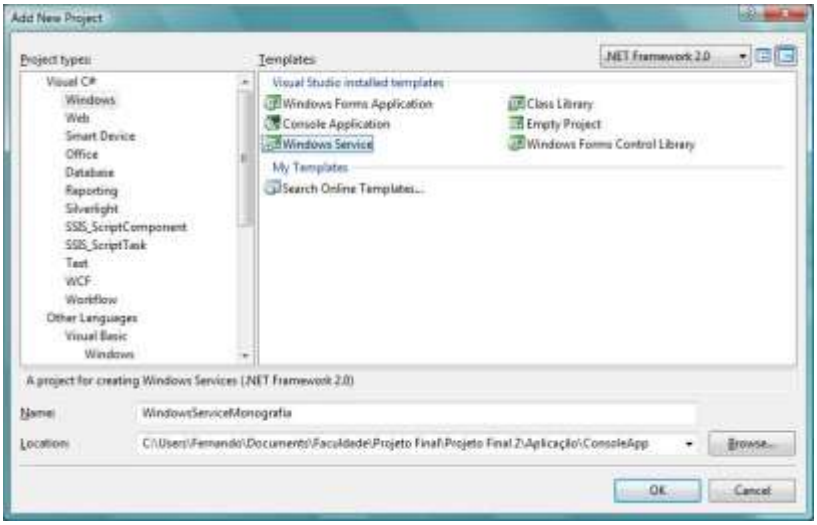


Figura 4. 2 – Criação de um novo projeto do tipo *Windows Service* usando o Visual C#.

2º Etapa – No projeto *WindowsServiceMonografia* é adicionada a referência ao *namespace System.Management*, através do qual o uso do *WMI* é possível.

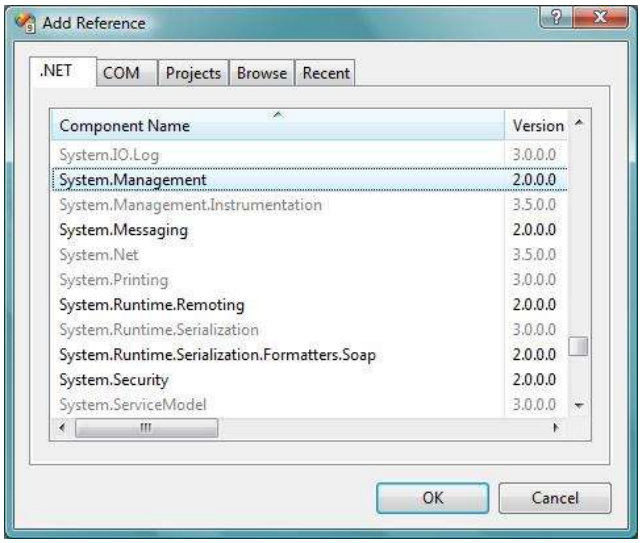


Figura 4. 3 - Adicionando a referência ao namespace System.Management para o uso do WMI.

3º Etapa – É criada a classe *Machine*. Nessa classe são declaradas as propriedades que representam as informações a serem coletadas através do *WMI*. As figuras 4.4 e 4.5 mostram as propriedades.

```
// ***************************
// propriedades do processador
// ***************************
private string cpuModel;
public string CpuModel
{
    get { return cpuModel; }
    set { cpuModel = value; }
}

private string cpuManufacturer;
public string CpuManufacturer
{
    get { return cpuManufacturer; }
    set { cpuManufacturer = value; }
}

// ***************************
// propriedades do HD
// ***************************
private string hdModel;
public string HdModel
{
    get { return hdModel; }
    set { hdModel = value; }
}

private double hdCapacity;
public double HdCapacity
{
    get { return hdCapacity; }
    set { hdCapacity = value; }
}
```

```
// ***************************
// propriedades da placa mãe
// ***************************
private string motherBoardModel;
public string MotherBoardModel
{
    get { return motherBoardModel; }
    set { motherBoardModel = value; }
}

private string motherBoardManufacturer;
public string MotherBoardManufacturer
{
    get { return motherBoardManufacturer; }
    set { motherBoardManufacturer = value; }
}

// ***********************************
// propriedades do adaptador de redes
// ***********************************
private string ntAdapterName;
public string NtAdapterName
{
    get { return ntAdapterName; }
    set { ntAdapterName = value; }
}

private string ntAdapterMAC;
public string NtAdapterMAC
{
    get { return ntAdapterMAC; }
    set { ntAdapterMAC = value; }
}
```

Figura 4. 4 - Mostra a definição das propriedades de hardware

```
private string[] softwareCaption;
public string[] SoftwareCaption
{
    get { return softwareCaption; }
    set { softwareCaption = value; }
}
private string[] softwareInstallDate;
public string[] SoftwareInstallDate
{
    get { return softwareInstallDate; }
    set { softwareInstallDate = value; }
}
private string[] softwareInstallLocation;
public string[] SoftwareInstallLocation
{
    get { return softwareInstallLocation; }
    set { softwareInstallLocation = value; }
}
private string[] softwareVendor;
public string[] SoftwareVendor
{
    get { return softwareVendor; }
    set { softwareVendor = value; }
}
private string[] softwareVersion;
public string[] SoftwareVersion
{
    get { return softwareVersion; }
    set { softwareVersion = value; }
}
```

Figura 4. 5 – Mostra a criação das propriedades referentes aos softwares instalados

4º Etapa – É criada a classe *MachineLocal*. Essa classe é herdada da classe *Machine* e implementa os métodos de busca das informações usando o *WMI*. Já na instanciação da classe os métodos são invocados devido a sobrecarga do construtor.

```
/*
 * Modificando o construtor padrão da classe
 */
public MachineLocal()
{
    this.getProcessor();
    this.getMotherBoard();
    this.getHD();
    this.getNetworkAdapter();
    this.getSoftwareList();
}
```

Figura 4. 6 – Modificando o construtor padrão da classe para poder invocar os métodos de busca ao instanciar a classe.

5º Etapa – São desenvolvidos os métodos para a obtenção das informações das máquinas clientes. O primeiro a ser criado é o responsável pela obtenção das informações referentes ao processador e chamado de *getProcessor*. Para tal é usada a classe *Win32_Processor* do *WMI*. Para todos os métodos essa

etapa se repete, mas usando as devidas classes para a aquisição dos dados necessários. A figura 4.7 mostra a implementação do método *getMotherBoard* que é o responsável por trazer as informações da placa mãe (modelo e fabricante).

```
public override void getMotherBoard()
{
    try
    {
        ManagementObjectSearcher searcher = new ManagementObjectSearcher(Machine.motherBoardQuery);
        foreach (ManagementObject result in searcher.Get())
        {
            this.MotherBoardModel = result.GetPropertyValue("Product").ToString();
            this.MotherBoardManufacturer = result.GetPropertyValue("Manufacturer").ToString();
        }
    }
    catch (Exception erro)
    {
        throw new Exception(erro.Message);
    }
}
```

Figura 4. 7 – Método getMotherBoard, responsável por trazer as informações referentes a placa mãe.

6º Etapa – Após a obtenção das informações necessárias, é preciso fazer a gravação no banco de dados, então foi criada uma camada (um conjunto de classes) para efetuar esse processo de armazenamento. Em desenvolvimento de softwares é considerada uma boa prática trabalhar com desenvolvimento em camadas, onde é possível separar toda a parte da lógica do negócio, da parte de bancos de dados e da parte de apresentação, que pode ser uma interface gráfica de usuário. Para esse projeto, a camada de acesso a dados é composta por três classes: *ItemDB*, *ColecaoItemDB* e *DatabaseActions*. As operações com o banco são feitas de fato na última.

7º Etapa – Após a camada de dados, é criada a camada de negócio. É nessa parte que está a inteligência da aplicação. A partir de dados recebidos é possível a execução de métodos de validação por exemplo. Através de entradas dadas por um usuário pode ser feito o processamento e saída de outras informações. Para esse projeto a camada de negócio é formada por duas classes: *InsertMachine* e *VerifySoftwareList*. A primeira é responsável por fazer a interação com a camada de acesso a dados e possui os métodos para validação e

inserção das informações na base de dados. A segunda é responsável por validar os softwares instalados nas máquinas. Caso haja algum programa instalado que não seja permitido, ela é a responsável por emitir o alerta ao administrador da rede.

8º Etapa – Por fim, é adicionado ao projeto um instalador. É por causa dele que será possível fazer a instalação do serviço nas máquinas clientes. Basta abrir o serviço, pelo Visual Studio, e adicionar o instalador.

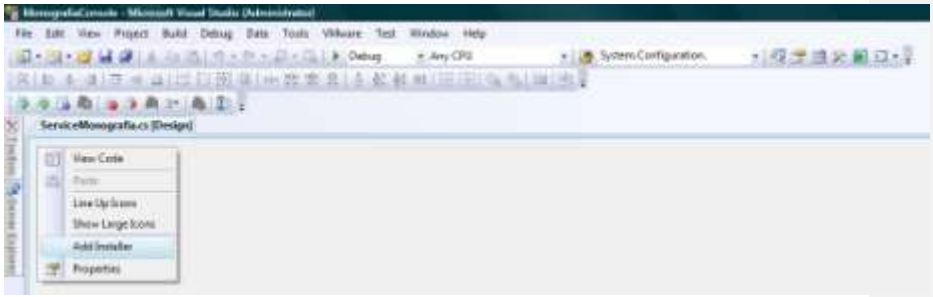


Figura 4. 8 – Inserindo o instalador do service através do visual studio.

A figura 4.9 demonstra o digrama de classes que compõe o serviço, com suas propriedades e métodos.

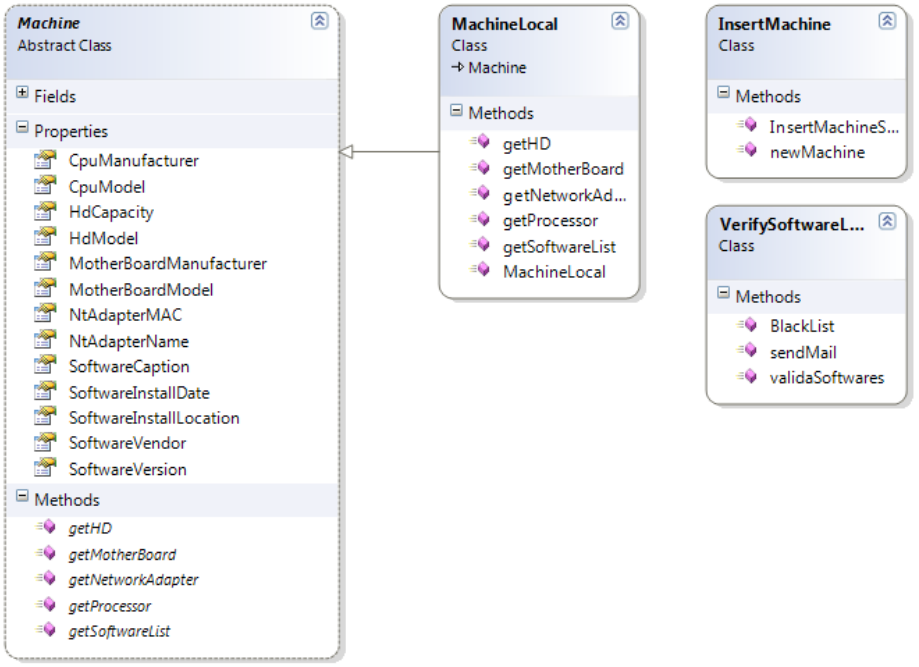


Figura 4. 9 – Diagrama de classes do serviço gerada pelo Visual Studio 2008

4.5. BASE DE DADOS

Após a criação do serviço, pode se começar a criação do banco de dados que irá armazenar as informações coletadas. Abaixo são enumeradas as etapas para o desenvolvimento do banco de dados usando o SGBD SQL Server 2008.

1º Etapa – Através do *SQL Server Management Studio* crie um novo banco de dados com os parâmetros básicos. A figura 4.10 ilustra o processo.

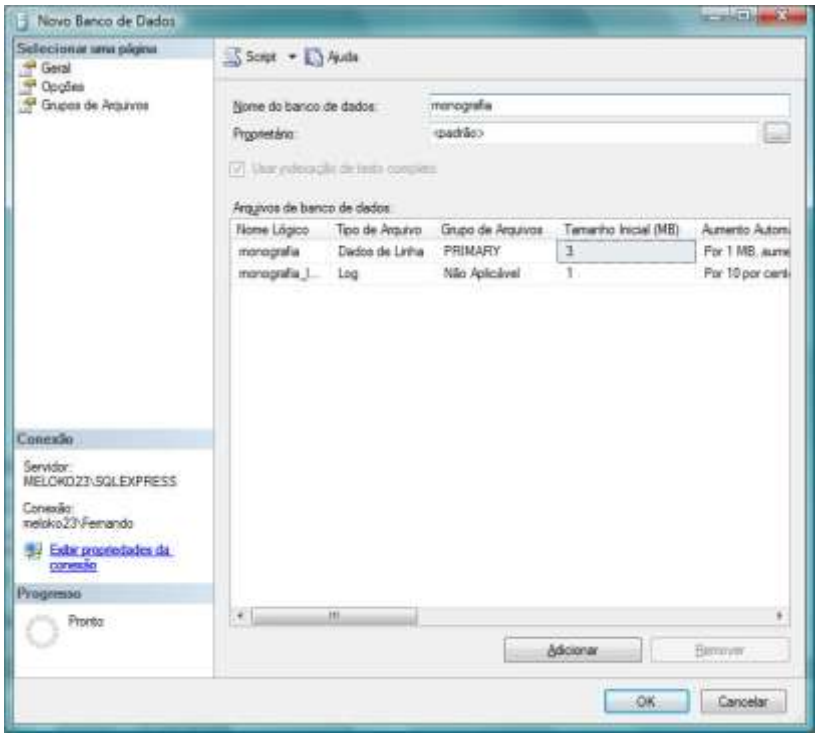


Figura 4. 10 – Criação do banco de dados pelo SQL Server Management Studio

2º Etapa – Criação de um usuário que tenha permissão de escrita e leitura na base recém-criada. Para isso é preciso ir através do menu *Pesquisador de Objetos*, e nas opções *Segurança → Logons*, fazer a inclusão do usuário.

3º Etapa – São criadas as tabelas que irão armazenar as informações coletadas. Para esse projeto, foram criadas três tabelas: *Machine*, *SoftwareList* e *SoftwareBlackList*. As duas primeiras são responsáveis por guardar os dados referentes às coletas feitas pelo serviço, ou seja, as informações das máquinas clientes. A terceira é possui uma lista, criada pelo administrador, com os softwares não permitidos. A inclusão de um software nessa lista varia de acordo com a política de segurança adotada por cada administrador de rede. As figuras 4.11 e 4.12 mostram a estrutura das tabelas e o relacionamento entre elas.

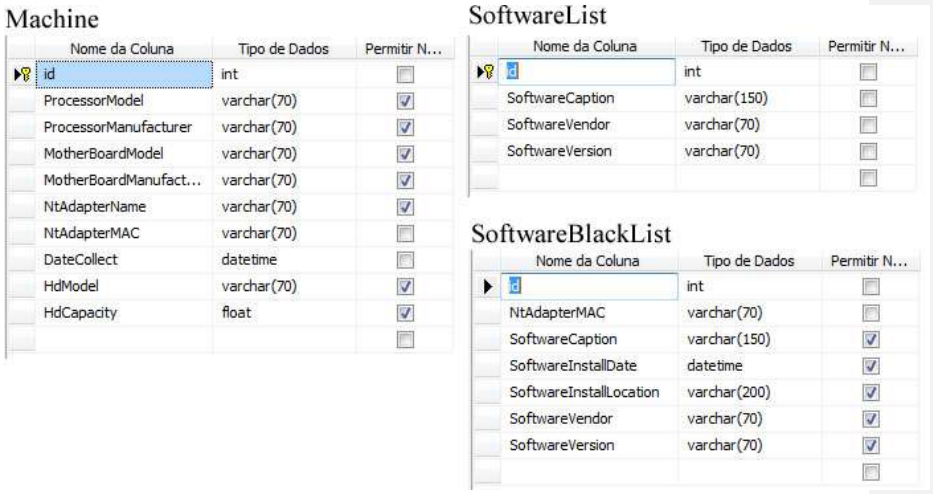


Figura 4. 11 – Estrutura das tabelas e a nomenclatura dos campos

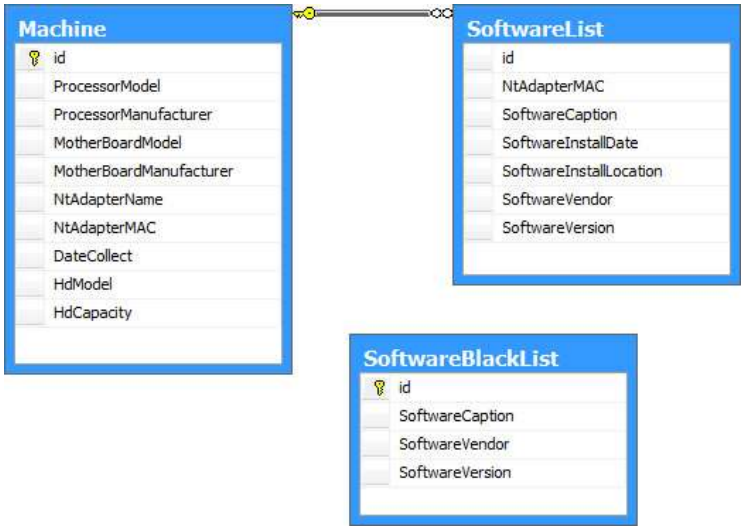


Figura 4. 12 – Relacionamento entre as tabelas no banco de dados

4º Etapa – São desenvolvidos procedimentos armazenados (*stored procedures*) que são responsáveis pelas operações como banco de dados (INSERT, UPDATE, DELETE e SELECT).

5º Etapa – O serviço e instalado em uma máquina na mesma rede que o servidor e é iniciado para verificar a correta inserção das informações nas tabelas apropriadas.

4.6. INTERFACE WEB

Enfim foi criada a interface gráfica de usuário. Desenvolvida em ASP.NET, pode ser dividida em 3 partes. A primeira é a autenticação dos usuários do sistema, usando campos de usuário e senha. A segunda é a responsável por tratar as informações guardadas no banco de dados e apresentá-las de forma amigável aos usuários do sistema. Por fim, a última é a responsável por gerenciar os softwares não permitidos. Através dela os usuários podem inserir, editar ou mesmo apagar softwares da lista. Abaixo seguem as etapas do desenvolvimento.

1º Etapa – Através do Visual Studio 2008, foi adicionada à solução uma ASP.NET Web Application chamada de *MonografiaWEB*, como ilustra a figura 4.13.

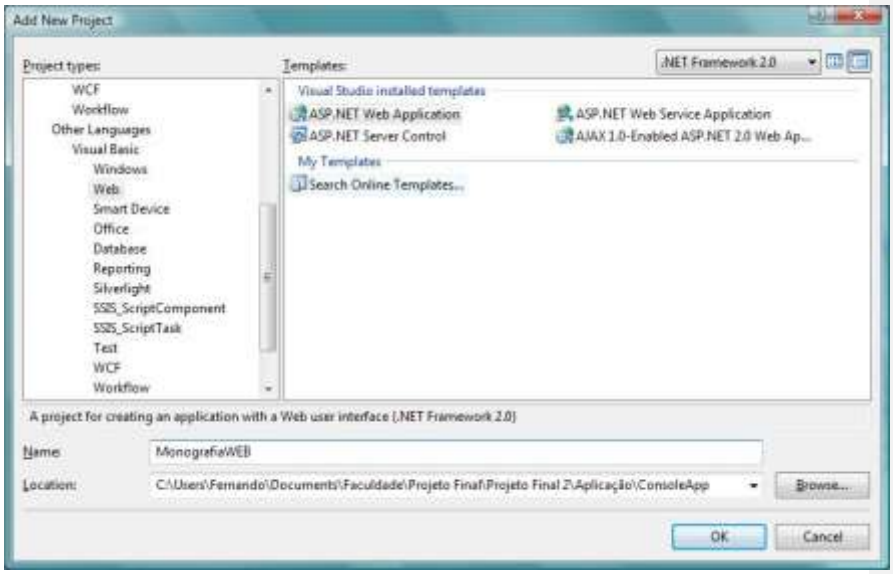


Figura 4. 13 – Criação da ASP.NET Web Application

2º Etapa – Com a aplicação criada é então adicionado um novo *web form*, o qual será responsável pela autenticação dos usuários. O Visual Studio então gera esse componente que foi chamado de *login.aspx*, estruturado no padrão *Hypertext Markup Language* (HTML). Após isso, abre-se a página e é alterado o modo de visualização de *Source* para *Design*, através do qual é possível fazer a inserção dos componentes de acesso a dados. Através da barra de ferramentas (*Toolbox*), na área de Login é selecionado o item *Login* e jogado para a área de trabalho da página. Com o item devidamente adicionado à página, são dadas as permissões de acesso à aplicação. Para isso, basta acessar a opção *ASP.NET Configuration* no menu superior (*Project* → *ASP.NET Configuration*). Feito isso, é aberta uma página de administração da aplicação. Na aba de segurança são adicionados os usuários que deverão ter permissão. Voltando ao componente de login, são configuradas algumas propriedades para a funcionalidade. No campo *DestinationPageUrl* é definido para onde será redirecionada a página em caso de sucesso na autenticação (Home.aspx). Caso o login não seja feito com sucesso, é configurada a mensagem de falha na propriedade *FailureText* (Sua tentativa de logon não pode ser efetuada. Por favor tente novamente.).

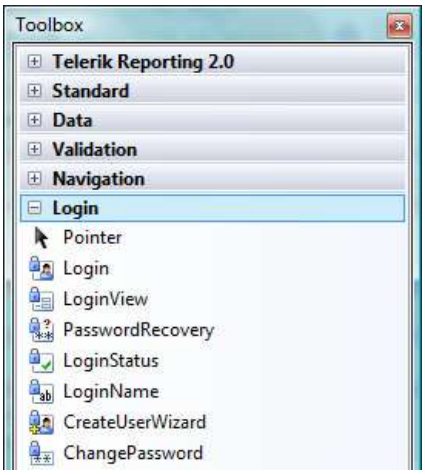


Figura 4. 14 – ToolBox com as opções de ações de login.

3º Etapa – Após a criação da página de login, são criadas as interfaces que darão acesso as informações. A primeira página a ser criada é a *Home.aspx* e para isso basta acessar o menu *Solution Explorer* do VS2008 e escolher a opção para adicionar um novo *webform*. Com a página criada é preciso fazer os links com as outras interfaces e para isso é usado um componente de menu do ASP.NET.

Figura 4. 15 – ToolBox que possui o componente de menu usado.

Depois da inclusão do componente de menu o mesmo foi configurado adicionando um item para cada interface: *Inventário*, *Softwares não Permitidos*, *Registrar softwares não permitidos* e *Pesquisa*. Para cada item adicionado ao

menu foram criados *webforms* que contém a lógica para o correto tratamento das informações.

Para o primeiro item do menu, foi adicionado um *webform* chamado *Inventário.aspx*, que é responsável por listar todas as coletas realizadas, exibindo as seguintes informações: processador (modelo e fabricante), placa mãe (modelo e fabricante), placa de rede (endereço MAC e modelo), disco rígido (modelo e capacidade) e a data que foi feita a coleta. E para a exibição dos dados foi usado um controle do ASP.NET já preparado para isso, o *gridview*, presente na *Toolbox* do Visual Studio. Após a seleção do controle é preciso configurá-lo, e a primeira coisa a ser feita é a conexão com o banco de dados, que pode ser realizada através de outro componente pronto, o *SqlDataSource*. Esse componente também precisa configurado e para a conexão é preciso acessar a opção *Configure Data Source*. Através disso pode-se definir a string de conexão e em seguida a forma de acesso aos dados (através de instruções SQL diretas ou usando procedimentos armazenados). Para esse projeto foram usadas apenas *stored* procedures, pois possui algumas vantagens (facilidade de manutenção, menor tráfego na rede e maior segurança das informações). Feito isso, são escolhidos os campos serão exibidos pelo *gridview*. Por fim é feita no *gridview* a referência ao *SqlDataSource* e os dados estão prontos para a exibição na interface. As figuras 4.16 e 4.17 ilustram a configuração do *gridview* e a configuração do *SqlDataSource*.

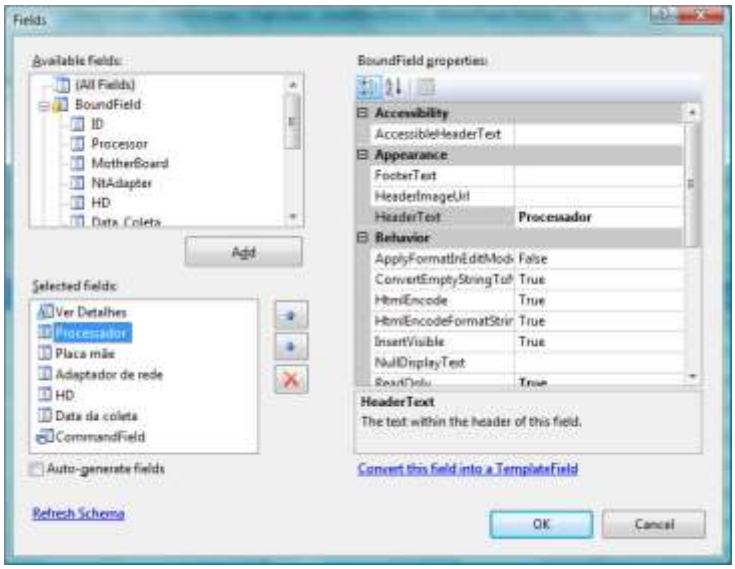


Figura 4. 16 – Configuração do controle Gridview.

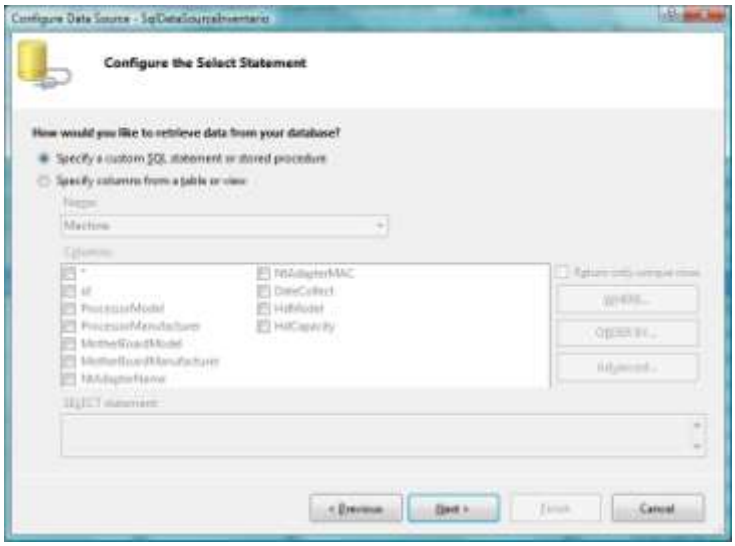


Figura 4. 17 – Configuração do componente SqlDataSource.

4º Etapa – O próximo item do menu a ser configurado é o de software não permitidos. Para isso adiciona-se um novo *webform* chamado *blacklist.aspx*. Para esse item repete-se a 3ª etapa sendo adaptados apenas os dados selecionados (isso inclui as *stored procedures* que fazem as buscas das informações) e a forma como os dados serão mostrados. É incluído também um link para a página da inserção de novos softwares.

5º Etapa – Na sequência é criada a página de inclusão de softwares não permitidos. Mais uma vez adiciona-se um *webform* denominado *RegisterSoftwares.aspx*. Nessa página são incluídos três componentes TextBox, do ASP.NET, que são referentes aos seguintes campos: Nome do software, Versão do software e Fabricante do software. É adicionado também um botão que é responsável por fazer a ação de inserção dos dados no banco. A figura 4.18 demonstra a tela de inserção.

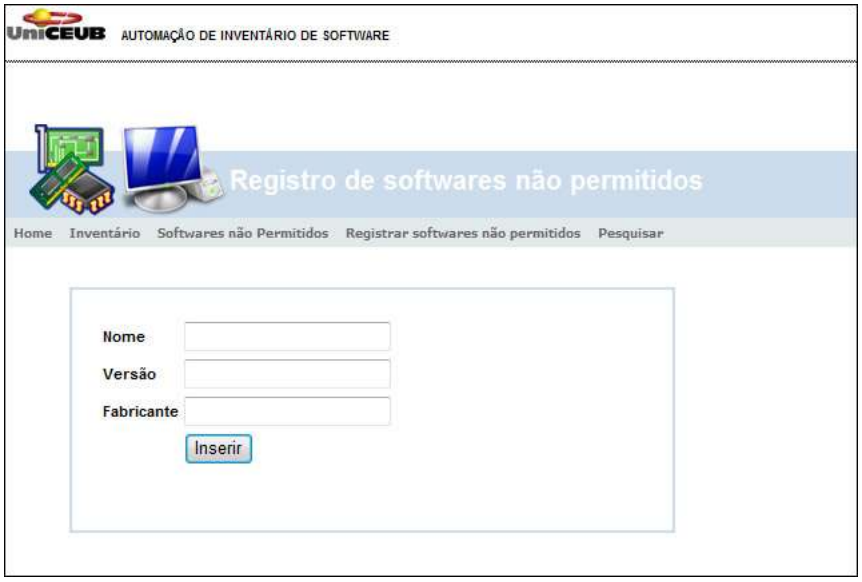


Figura 4. 18 – Página de inserção de softwares não permitidos

6º Etapa – O último item do menu é a opção de pesquisa. Para isso é criada uma nova página, chamada de *Pesquisa.aspx*. Nesse novo *webform* são adicionados um campo TextBox, usado para colocar as informações que serão

buscadas, e um botão que será responsável por fazer a ação de pesquisa das informações. O resultado da pesquisa é mostrado em um gridview, e segue o mesmo padrão descrito na etapa nº 3. Com isso é concluída a parte do projeto que se refere a interface de usuário.

CAPÍTULO 5 – TESTES E RESULTADOS

5.1. INTRODUÇÃO

Para poder atestar que o software criado no capítulo anterior faz o que foi proposto no escopo do projeto, foi montado um ambiente simples de testes com dois computadores: um atuando como servidor de bando de dados e servidor web (para a interface de usuário) e o outro sendo usado como cliente, onde foi instalado o serviço do Windows para fazer a busca, validação e gravação das informações.

Foram efetuados os testes nesse simples cenário e verificou-se que o aplicativo funciona como o planejado. Com isso um ambiente mais complexo foi montado para que testes mais completos pudessem ser feitos. Para esses testes foram configurados cinco (5) computadores: um sendo o servidor (de banco de dados e servidor web) e os outros quatro como sendo maquinas clientes. Após a instalação do serviço nos computadores clientes as informações foram coletadas e foi possível analisar os dados para poder avaliar o desempenho da ferramenta.

5.2. AMBIENTES DE TESTES

5.2.1. AMBIENTE SIMPLES

Para um primeiro cenário, mais simples, foram usados dois computadores, um atuando como servidor e outro como cliente. Abaixo são listadas as configurações dos mesmos:

- Servidor:
    - Notebook da marca NEC, processados Core 2 Duo T5500 1.7 GHz, 2 GB de memória RAM e 160 GB de disco rígido.
- Cliente:
    - Desktop da marca CTIS, processados Core 2 Duo T5600 2.0 GHz, 3 GB de memória RAM e 320 GB de disco rígido.

Os dois microcomputadores supracitados atendem aos requisitos de hardware e software para a realização dos testes. Abaixo são descritas as etapas para configuração e execução do experimento.

1ª Etapa: o primeiro passo é a inclusão dos dois computadores em uma mesma rede. Isso se faz necessário para que o computador cliente consiga enviar as informações que serão gravadas no banco de dados.

2ª Etapa: É instalado no computador cliente o .Net Framework 2.0 para a instalação do serviço do Windows. Como framework instalado é feita a instalação do serviço.

3ª Etapa: Com as etapas acima concluídas com sucesso, é dado início a procedimento de testes propriamente dito. Para isso, o computador usado como cliente foi submetido a uma série de procedimentos de ligar/desligar para que a rotina de inserção de dados pudesse ser executada.

4ª Etapa: Após a etapa três (3) foi possível notar que as informações estavam sendo inseridas de forma correta na base de dados e isso pode ser verificado fazendo uma seleção (usando uma instrução SELECT do SQL) na tabela responsável por guardar as informações.

5ª Etapa: Por fim, foi acessada a interface web e foi possível notar que os dados estavam corretamente armazenados no banco de dados.

5.2.2. AMBIENTE DE TESTES COMPLETO

Após a execução dos testes simples, um ambiente mais elaborado é montado para a execução de testes mais completos. Para esse novo cenário são usadas cinco (5) máquinas, uma sendo o servidor (de banco de dados e web) e as outras quatro (4) atuando como computadores clientes. Assim como no cenário anterior, todos os microcomputadores precisam estar configurados em uma mesma rede para a correta transmissão de dados entre eles.

Segue abaixo a configuração dos computadores usados para os testes:

- Servidor:
    - Notebook da marca NEC, processados Core 2 Duo T5500 1.7 GHz, 2 GB de memória RAM e 160 GB de disco rígido.

- Clientes:
    - 01 - Desktop da marca CTIS, processador Core 2 Duo 2.0 GHz, 3 GB de memória RAM e 320 GB de disco rígido.

- 02 - Desktop da marca MEGAWARE, processador Core 2 Duo 2.2 GHz, 2 GB de memória RAM e 250 GB de disco rígido.

- 03 - Notebook da marca CCE, processador Core 2 Duo 1.6 GHz, 1 GB de memória RAM e 120 GB de disco rígido.

- 04 - Desktop da marca DELL, processador Intel Pentium 4 1.8 GHz, 1 GB de memória RAM e 80 GB de disco rígido.

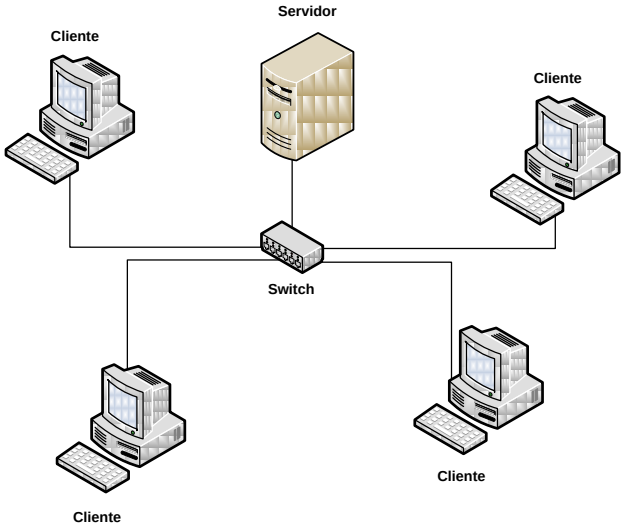


Figura 5. 1 – Configuração do ambiente de testes

Todos os computadores usados atendiam aos requisitos mínimos para a execução de suas tarefas específicas (computadores clientes e servidor). Todos os requisitos foram previamente especificados no capítulo anterior.

5.3. REALIZAÇÃO DOS TESTES

Abaixo serão listadas as etapas para a configuração e execução do cenário de testes;

1ª Etapa: Todos os computadores usados nos testes devem ser adicionados em uma mesma rede de computadores para que as informações dos clientes possam ser enviadas corretamente ao servidor.

2ª Etapa: Para que as informações gravadas previamente através dos primeiros testes não interferissem na esperimentação, as tabelas responsáveis por guardar as informações das coletas foram reiniciadas através do comando *truncate*, do SQL. Então dois comandos foram executados, uma para cada tabela: *truncate table Machine* e *truncate table SoftwareList.*

3ª Etapa: Em todas as máquinas utilizadas como clientes foram instalados o *.Net Framework* 2.0 e o serviço do Windows responsável pela coleta das informações. O serviço foi configurado como modo de inicialização automático, o que deixa a execução do programa transparente aos usuários dos computadores e permite que o processo ocorra durante o *startup* do sistema operacional.

4ª Etapa: Todos os computadores clientes foram submetidos a procedimentos de liga/desliga para que o serviço pudesse ser iniciado e com isso os dados fossem gravados no banco de dados. Foi possível também verificar a correta validação dos softwares instalados nas máquinas, de acordo com a listagem de softwares não permitidos. O alerta ao administrador do sistema foi enviado sempre que verificada a presença na lista de algum programa não permitido

5ª Etapa: Foi verificada também a correta inserção das informações nas tabelas do banco de dados. Por meio de *stored procedures*, criadas para fazer a seleção de registros, foi possível notar que os dados referentes aos computadores clientes (informações de hardware) e todos os softwares instalados em cada um deles.

6ª Etapa: Finalmente, através da interface gráfica foi possível observar todos os dados coletados já exibidos de forma estruturada e organizada, prontos para o uso do administrador do sistema. Também foi possível notar que o módulo da interface web responsável pela edição de softwares não permitidos funcionou corretamente, assim como o módulo de pesquisa de computadores.

5.4. RESULTADOS OBTIDOS

Durante o período quatro horas, foram guardados um total de vinte e cinco coletas, dos quatro computadores clientes presentes na rede montada para o experimento. Entenda-se por coleta o registro de cada computador, contemplando as informações de hardware e software.

O computador cliente número 01, com endereço MAC 00:0D:F0:39:0D:4F, gerou quatro registros. O computador cliente número 02, com endereço MAC 00:1B:24:24:AE:B1, gerou oito registros. O computador cliente número 03, com endereço MAC 00:50:56:C0:00:08, gerou sete registros. O computador cliente número 04, com endereço MAC 00:03:2A:B3:0E:65, gerou seis registros.

Para os softwares instalados, foi gravado um total de 3805 registros, dando uma média aproximada de 38 softwares instalados em cada computador cliente. Para a validação dos softwares não permitidos, os seguintes programas foram definidos como não permitidos:

Id	Nome	Fabricante	Versão
1	WebDAV 7.5 For IIS 7.0	Microsoft Corporation	7.5.7054.1430
2	VMware Workstation	VMware, Inc.	6.5.1.5078
3	Vista Codec Package	Shark007	5.0.2
4	Skype™ 4.0	Skype Technologies S.A.	4.0.206
5	Nero 7 Ultra Edition	Nero AG	7.02.9753
6	Windows Live Messenger	Microsoft Corporation	14.0.8064.0206

Tabela 5. 1 – Softwares cadastrados como não permitidos.

Para todos os computadores clientes usados nos testes, ao menos um software não permitido foi encontrado, tendo como o maior deles o item 6 da tabela 5.1. Na figura 5.2 podem ser vistas, de uma forma geral, todas as informações coletadas de uma determinada máquina através do *WMI*.

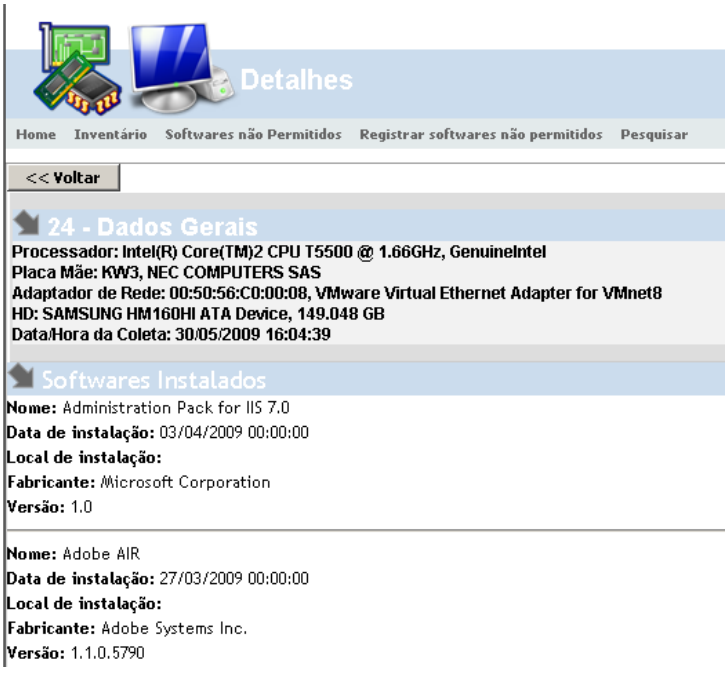


Figura 5. 2 – Tela de detalhes sobre as informações coletadas (hardware e software).

CAPÍTULO 6 – CONCLUSÃO

O projeto foi criado com alguns objetivos específicos: fazer um inventário automático de softwares instalados em uma rede; fazer a validação de todos os softwares instalados, alertando o administrador do sistema sempre que houvesse a presença de um programa não permitido; e por fim, a disponibilização de todas as informações coletadas através de uma interface gráfica, o que permite uma melhor estruturação e organização dos dados para o usuário final.

Para que os objetivos supracitados pudessem ser atingidos, foi criada uma ferramenta para o administrador de rede. O processo de desenvolvimento foi dividido em partes:

- Criação de um Windows Service responsável pela coleta das informações (hardware e software), pela gravação das informações no banco e por fim a validação dos softwares instalados nos computadores da rede;

- Criação do banco de dados onde todas as informações serão armazenadas. Isso inclui objetos como tabelas e *stored procedures* (responsáveis pelas ações de inserção, edição e seleção).

- Criação de uma interface de usuário, nesse caso uma interface web, onde todas as informações gravadas serão formatadas e organizadas para a exibição ao usuário final. Através dela também é possível fazer a edição da lista de softwares não permitidos e a pesquisa de informações já armazenadas.

A solução acima mencionada foi desenvolvida usando soluções *Microsoft*. Para a criação do serviço de Windows e da interface gráfica foi utilizado o *Microsoft Visual Studio* 2008, pois através dessa IDE pode-se desenvolver rapidamente usando o Visual C# que possui já uma série de classes preparadas para o trabalho com o WMI além da familiaridade do graduando com o software. Para a criação do banco de dados foi usado o *Microsoft SQL SERVER 2008 Express*, pois é um SGBD bem conceituado no Mercado de trabalho sem falar no fato de ser gratuito. Ao final da implementação foi realizada uma série de testes que retornaram um conjunto de resultados, através dos quais foi possível analisar todas as funcionalidades da aplicação.

Formatado: Português (Brasil)

Ao final dos testes realizados pôde-se concluir que a escolha das tecnologias e ferramentas para o projeto foi acertada e bem apropriada. O *WMI* pode ser usado de forma simples através do C# (no caso do *Visual Studio*, o *Visual C#*) e com isso a obtenção das informações necessárias se torna mais fácil. O *VS2008* também provou ser uma IDE muito eficiente para o desenvolvimento da interface de usuário. Aliado ao *WMI*, o *Visual Studio* 2008 (inclui o *VC#*) permite a criação de aplicações que ajudem o profissional responsável pela administração de redes. Através dos dados acessados pelo *WMI* e todas as funcionalidades disponibilizadas pelo aplicativo criado, é possível a criação de uma base de dados com informações bem precisas sobre tudo que é instalado em uma rede de computadores.

Ao final dos experimentos e com todas as informações coletadas, foi possível concluir que o processo de inventariado automático de software foi um sucesso tendo como referência os objetivos do projeto. Durante a experimentação, em um período de quatro horas, um total de 3805 softwares foi coletado de todos os quatro computadores utilizados e todos os dados foram armazenados no banco de dados sem nenhuma falha. Foi possível notar também que a validação dos softwares instalados funcionou corretamente. Ao final da execução do serviço do Windows, sempre que encontrado algum programa instalado que constava na lista de dos não permitidos, o alerta ao administrador era enviado.

A interface gráfica de usuário também cumpriu seu propósito de acordo com o escopo inicial definido. Na parte de apresentação das informações, os dados podem ser vistos de maneira simples e objetiva, economizando tempo e esforço do usuário final. Na área referente ao login dos usuários, o módulo provou ser funcional e eficiente, permitindo o acesso às informações apenas para usuários autorizados. Foi possível notar o correto funcionamento do módulo de inclusão de softwares não permitidos, dando liberdade aos administradores a inclusão/edição de qualquer software. Por fim, pôde-se notar que o módulo de pesquisas se mostrou funcional e muito útil ao usuário, permitindo a pesquisa de qualquer máquina através do seu endereço MAC.

O processo de criação através de etapas mostrou-se útil também na fase de testes para detecção de falhas. Dessa maneira os problema encontrados foram separados em cada uma das fases de desenvolvimento o que permitiu a correção pontual para cada módulo específico.

Para o desenvolvimento desse projeto, desde a concepção de idéia até a fase de conclusão da ferramenta, muitas disciplinas ministradas no curso de *Engenharia de Computação* foram de grande ajuda e fundamentais. Dentre elas pode-se destacar:

- **Redes I e II e Tópicos avançados de redes**: todo o conhecimento para o funcionamento das redes de computadores, bem como as responsabilidades de um administrador de redes, e isso permitiu o desenvolvimento da proposta pensando já nas possíveis tecnologias para a execução da ferramenta.

- **Banco de Dados I**: apresentação de conceitos de bancos de dados relacionais e permitiu a escolha de um SGBD apropriado para o desenvolvimento do projeto.

- **Linguagem Técnica de Programação I e II, Estrutura de Dados**: apresentou conceitos fundamentais sobre programação para o desenvolvimento da solução.

- **Arquitetura de Computadores**: forneceu conhecimentos sobre a arquitetura dos computadores, bem como sua composição e componentes. Tais informações são fundamentais para a criação dos métodos de obtenção de dados através do *WMI*.

- **Engenharia de Programas**: apresentou conceitos e técnicas para o desenvolvimento de softwares, como o modelo de desenvolvimento orientado a objetos, e que puderam ser usadas para o desenvolvimento desse projeto.

6.1. SUGESTÕES PARA TRABALHOS FUTUROS

A medida que o projeto foi sendo desenvolvido, algumas opções de melhorias e/ou evoluções aparecerem e que pelo tempo limitado não puderem ser desenvolvidas. Abaixo são listadas algumas dessas opções:

- A tecnologia WMI é exclusiva para os Sistemas Operacionais Windows, o que restringe um pouco o uso da ferramenta. A idéia principal desse trabalho poderia ser aproveitada para a criação de uma solução que atendesse a diferentes plataformas.

- O WMI permite a busca de informações em computadores remotos. Poderia ser criada uma interface (web ou application desktop) que fizesse uma busca pela rede fazendo as coletas das informações necessárias.

- A solução criada emite um alerta ao administrador do sistema sobre softwares não permitidos sempre que o serviço de Windows é iniciado. Porém, após a execução do mesmo, a validação dos softwares só será feita após o reinicio do serviço, e não em tempo real

no caso da instalação de novos aplicativos. Pode-se criar um agente que se instalaria nas máquinas clientes dando essas informações tempo real ao administrador de rede.

- Os alertas emitidos ao administrador são sempre via correio eletrônico (e-mail). Esses alertas poderiam também ser gravados no banco de dados e dentro do módulo da interface gráfica o administrador poderia acessar todos os alertas enviados e gravados.

REFERÊNCIAS BIBLIOGRÁFICAS

ASP.NET. Microsoft ASP.NET. **ASP.NET**. 2009a. Disponível em: <http://www.asp.net/>. Acesso em: 04 mai. 2009.

CACIC. Configurador Automático e Coletor de Informações Computacionais. **CACIC**. 10 mai. 2009. Disponível em: <http://www.softwarepublico.gov.br/ver-comunidade?community_id=3585>. Acesso em: 10 mai. 2009.

CISCO. Cisco Systems, Inc. **CiscoWorks Resource Manager Essencials**. 15 jun. 2007. Disponível em: <http://www.cisco.com/en/US/products/sw/cscowork/ps2073/>. Acesso em: 09 mai. 2009.

DATE, C.J. **Introdução a Sistemas de Bancos de Dados**. tradução (4ª Ed. americana). Rio de Janeiro: Campus, 1994.

DMTF. Distributed Management Task Force. **Web-Based Enterprise Management (WBEM)**. 2009a. Disponível em: <http://www.dmtf.org>. Acesso em: 04 mai. 2009.

DMTF. Distributed Management Task Force. **Web-Based Enterprise Management (WBEM)**. 2009b. Disponível em: <http://www.dmtf.org/standards/wbem/>. Acesso em: 04 mai. 2009.

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: 09 mai. 2009.

GEORGE, Rajesh; DELANO, Lance. **Wrox's SQL Server 2005 Express Edition Starter Kit**. 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=1-11-271-272^12522_4000_100__>. Acesso em: 10 mai. 2009.

IBM. International Business Machines Corp. **IBM - IBM Tivoli Configuration Manager**. 16 jan. 2007. Disponível em: <http://www-01.ibm.com/software/tivoli/products/config-mgr/>. Acesso em: 10 mai. 2009.

KUROSE, James F; ROSS, Keith W. **Redes de Computadores e a Internet**. Addison-Wesley, 2003.

Formatado: Português (Brasil)
Formatado: Português (Brasil)
Formatado: Português (Brasil)
Formatado: Português (Brasil)
Formatado: Português (Brasil)
Formatado: Português (Brasil)
Formatado: Inglês (EUA)
Formatado: Português (Brasil)
Formatado: Português (Brasil)
Formatado: Inglês (EUA)

MICROSOFT. Microsoft Corporation. **WMI - Windows Management Instrumentation**. 2009a. Disponível em: <http://msdn.microsoft.com/en-us/library/aa394582(VS.85).aspx>. Acesso em: 04 mai. 2009.

MICROSOFT. Microsoft Corporation. **WMI – Windows Management Instrumentation (WMI - About)**. 2009b. Disponível em: <http://msdn.microsoft.com/en-us/library/aa384642(VS.85).aspx>. Acesso em: 04 mai. 2009.

MICROSOFT. Microsoft Corporation. **WMI – Windows Management Instrumentation (WMI - Architecture)**. 2009c. Disponível em: <http://msdn.microsoft.com/en-us/library/aa394553(VS.85).aspx>. Acesso em: 04 mai. 2009.

MICROSOFT. Microsoft Corporation. **WMI – Windows Management Instrumentation (WMI - Reference)**. 2009d. Disponível em: <http://msdn.microsoft.com/en-us/library/aa394572(VS.85).aspx>. Acesso em: 04 mai. 2009.

MICROSOFT. Microsoft Corporation. **WMI – Windows Management Instrumentation (WMI – Win32 Provider)**. 2009e. Disponível em: <http://msdn.microsoft.com/en-us/library/aa394084(VS.85).aspx>. Acesso em: 07 mai. 2009.

MICROSOFT. Microsoft Corporation. **Microsoft SQL Server 2008**. 2009f. Disponível em: <http://www.microsoft.com/sqlserver/2008/en/us/overview.aspx>. Acesso em: 11 mai. 2009.

MICROSOFT. Microsoft Corporation. **Microsoft® SQL Server® 2008 Express - Requirement**. 2009g. Disponível em: <http://www.microsoft.com/Downloads/details.aspx?familyid=58CE885D-508B-45C8-9FD3-118EDD8E6FFF&displaylang=pt-br#Requirements>. Acesso em: 11 mai. 2009.

MICROSOFT. Microsoft Corporation. **Serviço do Windows**. 2009h. Disponível em: <http://msdn.microsoft.com/pt-br/library/d56de412(VS.80).aspx>. Acesso em: 11 mai. 2009.

MSDN. Microsoft Developer Network. **WMI – Windows Management Instrumentation**. 2009a. Disponível em: <http://msdn.microsoft.com/pt-br/ms178709.aspx>. Acesso em: 04 mai. 2009.

SOUSA JR, Rafael Timóteo de. Universidade de Brasília. **Administração e Gerência de Rede**, Brasília, 5 abr. 2005.

WATKINS, David Mascarenhas. Centro Universitário de Brasília. **Automação de Inventário de Hardware e Registro de Uso**. 2007.

Formatado: Inglês (EUA)

Formatado: Inglês (EUA)

APÊNDICE A – CÓDIGO FONTE

Todo a programação da ferramenta foi feita usando o conceito de desenvolvimento em camadas, separando as áreas de lógica de negócio, acesso a dados e camada de visão. Abaixo segue o código com as divisões:

Coleta de Dados

Objetos → Objeto Machine

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Collections;

namespace MonografiaConsole.model
{
    public abstract class Machine
    {
        // ***************************
        // propriedades do processador
        // ***************************
        private string cpuModel;
        public string CpuModel
        {
            get { return cpuModel; }
            set { cpuModel = value; }
        }

        private string cpuManufacturer;
        public string CpuManufacturer
        {
            get { return cpuManufacturer; }
            set { cpuManufacturer = value; }
        }

        // ***************************
        // propriedades da placa mãe
        // ***************************
        private string motherBoardModel;
        public string MotherBoardModel
        {
            get { return motherBoardModel; }
            set { motherBoardModel = value; }
        }

        private string motherBoardManufacturer;
        public string MotherBoardManufacturer
        {
            get { return motherBoardManufacturer; }
            set { motherBoardManufacturer = value; }
```

```
}

// ***************************
// propriedades do HD
// ***************************
private string hdModel;
public string HdModel
{
    get { return hdModel; }
    set { hdModel = value; }
}

private double hdCapacity;
public double HdCapacity
{
    get { return hdCapacity; }
    set { hdCapacity = value; }
}

// ***********************************
// propriedades do adaptador de redes
// ***********************************
private string ntAdapterName;
public string NtAdapterName
{
    get { return ntAdapterName; }
    set { ntAdapterName = value; }
}

private string ntAdapterMAC;
public string NtAdapterMAC
{
    get { return ntAdapterMAC; }
    set { ntAdapterMAC = value; }
}

// ***********************************
// propriedades de softwares instalados
// ***********************************
private string[] softwareCaption;
public string[] SoftwareCaption
{
    get { return softwareCaption; }
    set { softwareCaption = value; }
}

private string[] softwareInstallDate;
public string[] SoftwareInstallDate
{
    get { return softwareInstallDate; }
    set { softwareInstallDate = value; }
}

private string[] softwareInstallLocation;
public string[] SoftwareInstallLocation
{
    get { return softwareInstallLocation; }
    set { softwareInstallLocation = value; }
}
```

```
        private string[] softwareVendor;
        public string[] SoftwareVendor
        {
            get { return softwareVendor; }
            set { softwareVendor = value; }
        }

        private string[] softwareVersion;
        public string[] SoftwareVersion
        {
            get { return softwareVersion; }
            set { softwareVersion = value; }
        }

        /*
         * ==__====__==__====__====__====__====__====__==__====__==
         * Constantes com as query's de pesquisa para cada item
         * ==__====__==__====__====__====__====__====__==__====__==
         */
        public const string cpuQuery = "Select * from
Win32_Processor";
        public const string motherBoardQuery = "Select * from
Win32_BaseBoard";
        public const string RAMQuery = "Select * from
Win32_PhysicalMemory";
        public const string HDQuery = "Select * from Win32_DiskDrive
WHERE InterfaceType != 'USB'";
        public const string NTQuery = "Select * from
Win32_NetworkAdapter WHERE NetConnectionStatus=2";
        public const string KeyBoardQuery = "Select * from
Win32_KeyBoard";
        public const string MouseQuery = "Select * from
Win32_PointingDevice";
        public const string MonitorQuery = "Select * from
Win32_DesktopMonitor";
        public const string CdDvdQuery = "Select * from
Win32_CDROMDrive";
        public const string SoftwareQuery = "Select * from
Win32_Product";


        //
************************************************************
        // Métodos que preenchem as propriedades do objeto
        //
************************************************************
        public abstract void getProcessor();
        public abstract void getMotherBoard();
        public abstract void getHD();
        public abstract void getNetworkAdapter();
        public abstract void getSoftwareList();
    }
}
```

Objetos → Objeto MachineLocal

```
using System;
using System.Collections.Generic;
```

```
using System.Web;
using System.Management;

namespace MonografiaConsole.model
{
    public class MachineLocal : Machine
    {
        /*
         * Modificando o construtor padrão da classe
         */
        public MachineLocal()
        {
            this.getProcessor();
            this.getMotherBoard();
            this.getHD();
            this.getNetworkAdapter();
            this.getSoftwareList();
        }


        //
************************************************************
        // Métodos que preenchem as propriedades do objeto
        //
************************************************************
        public override void getProcessor()
        {
            try
            {
                ManagementObjectSearcher searcher = new
ManagementObjectSearcher(Machine.cpuQuery);
                foreach (ManagementObject result in searcher.Get())
                {
                    this.CpuModel =
result.GetPropertyValue("Name").ToString();
                    this.CpuManufacturer =
result.GetPropertyValue("Manufacturer").ToString();
                }
            }
            catch (Exception erro)
            {
                throw new Exception(erro.Message);
            }
        }

        public override void getMotherBoard()
        {
            try
            {
                ManagementObjectSearcher searcher = new
ManagementObjectSearcher(Machine.motherBoardQuery);
                foreach (ManagementObject result in searcher.Get())
                {
                    this.MotherBoardModel =
result.GetPropertyValue("Product").ToString();
                    this.MotherBoardManufacturer =
result.GetPropertyValue("Manufacturer").ToString();
                }
            }
            catch (Exception erro)
```

```
            {
                throw new Exception(erro.Message);
            }
        }

        public override void getHD()
        {
            try
            {
                ManagementObjectSearcher searcher = new
ManagementObjectSearcher(Machine.HDQuery);
                foreach (ManagementObject result in searcher.Get())
                {
                    this.HdModel =
result.GetPropertyValue("Model").ToString();
                    this.HdCapacity =
Convert.ToDouble(result.GetPropertyValue("Size"));
                }
            }
            catch (Exception erro)
            {
                throw new Exception(erro.Message);
            }
        }

        public override void getNetworkAdapter()
        {
            try
            {
                ManagementObjectSearcher searcher = new
ManagementObjectSearcher(Machine.NTQuery);
                foreach (ManagementObject result in searcher.Get())
                {
                    this.NtAdapterName =
result.GetPropertyValue("Name").ToString();
                    this.NtAdapterMAC =
result.GetPropertyValue("MACAddress").ToString();
                }
            }
            catch (Exception erro)
            {
                throw new Exception(erro.Message);
            }
        }

        public override void getSoftwareList()
        {
            try
            {
                int icont = 0, itot = 0;
                ManagementObjectSearcher searcher = new
ManagementObjectSearcher(Machine.SoftwareQuery);
                foreach (ManagementObject result in searcher.Get())
                {
                    itot++;
                }

                if (itot > 0)
                {
                    this.SoftwareCaption = new string[itot];
```

```
                    this.SoftwareInstallDate = new string[itot];
                    this.SoftwareInstallLocation = new string[itot];
                    this.SoftwareVendor = new string[itot];
                    this.SoftwareVersion = new string[itot];
                }
                else
                {
                    throw new Exception("Não houve resultado para a
pesquisa.");
                }

                foreach (ManagementObject result in searcher.Get())
                {
                    this.SoftwareCaption[icont] = String.Empty;
                    this.SoftwareInstallDate[icont] = String.Empty;
                    this.SoftwareInstallLocation[icont] =
String.Empty;
                    this.SoftwareVendor[icont] = String.Empty;
                    this.SoftwareVersion[icont] = String.Empty;

                    if (!(result.GetPropertyValue("Caption") == null))
                    {
                        if
(!(String.IsNullOrEmpty(result.GetPropertyValue("Caption").ToString())
))
                            this.SoftwareCaption[icont] =
result.GetPropertyValue("Caption").ToString();
                    }

                    if (!(result.GetPropertyValue("InstallDate") ==
null))
                    {
                        if
(!(String.IsNullOrEmpty(result.GetPropertyValue("InstallDate").ToStrin
g())))
                            this.SoftwareInstallDate[icont] =
result.GetPropertyValue("InstallDate").ToString();
                    }

                    if (!(result.GetPropertyValue("InstallLocation")
== null))
                    {
                        if
(!(String.IsNullOrEmpty(result.GetPropertyValue("InstallLocation").ToS
tring())))
                            this.SoftwareInstallLocation[icont] =
result.GetPropertyValue("InstallLocation").ToString();
                    }

                    if (!(result.GetPropertyValue("Vendor") == null))
                    {
                        if
(!(String.IsNullOrEmpty(result.GetPropertyValue("Vendor").ToString()))
)
                            this.SoftwareVendor[icont] =
result.GetPropertyValue("Vendor").ToString();
                    }

                    if (!(result.GetPropertyValue("Version") == null))
                    {
```

```
                        if
(!(String.IsNullOrEmpty(result.GetPropertyValue("Version").ToString())
))
                            this.SoftwareVersion[icont] =
result.GetPropertyValue("Version").ToString();
                    }

                    icont++;
                }
            }
            catch (System.Management.ManagementException ex1)
            {
                throw new ManagementException(ex1.Message);
            }
            catch (Exception ex2)
            {
                throw new Exception(ex2.Message);
            }
        }
    }
}
```

Camada de Acesso a dados

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using System.Collections;

namespace MonografiaConsole.dao
{
    class DatabaseActions
    {
        private string connectionString;

        public DatabaseActions()
        {
            this.connectionString =
System.Configuration.ConfigurationSettings.AppSettings["ConnectionStri
ngSql"].ToString();
        }

        //CONSTANTES TIPOS DE DADOS
        public const int TIPO_INT = 0;
        public const int TIPO_FLOAT = 1;
        public const int TIPO_VARCHAR = 2;
        public const int TIPO_DATE = 3;
        public const int TIPO_XML = 4;
        public const int TIPO_TEXTO = 5;
        public const int TIPO_NVARCHAR = 6;
        public const int TIPO_MONEY = 7;

        //CONSTANTES TIPOS DE RETORNO DE DADOS
        public const int RETORNA_INTEIRO = 0;
        public const int RETORNA_DATASET = 1;
```

```
        public const int RETORNA_DATATABLE = 2;
        public const int RETORNA_CHAVE = 3;

        public int executarSP(ColecaoItemDB colecao)
        {
            try
            {
                return
Convert.ToInt32(executarComando(spParameters(colecao),
                                                        colecao.NomeSP,

RETORNA_INTEIRO));
            }
            catch (Exception e)
            {
                throw new Exception("Erro em executarSP: " +
e.StackTrace);
            }
        }

        public DataTable getDataTable(ColecaoItemDB colecao)
        {
            try
            {
                return
(DataTable)executarComando(spParameters(colecao),
                                                  colecao.NomeSP,
                                                  RETORNA_DATATABLE);
            }
            catch (Exception e)
            {
                //throw new Exception("Erro em getDataTable: " +
e.StackTrace);
                throw new Exception("Erro em getDataTable: " +
e.Message.ToString());
            }
        }

        public DataSet getDataSet(ColecaoItemDB colecao)
        {
            try
            {
                return (DataSet)executarComando(spParameters(colecao),
                                                colecao.NomeSP,
                                                RETORNA_DATASET);
            }
            catch (Exception e)
            {
                throw new Exception("Erro em getDataSet: " +
e.StackTrace);
            }
        }

        public int getChavePrimaria(ColecaoItemDB colecao)
        {
            try
            {
                return
Convert.ToInt32(executarComando(spParameters(colecao),
                                                colecao.NomeSP,
```

```
                                                RETORNA_CHAVE));
            }
            catch (Exception e)
            {
                throw new Exception("Erro em getChavePrimaria: " +
e.StackTrace);
            }
        }

        private List<SqlParameter> spParameters(ColecaoItemDB colecao)
        {
            ArrayList chaves = colecao.getChaves();
            Hashtable lista = colecao.getColecaoDeItensDB();
            List<SqlParameter> spParameters = new
List<SqlParameter>();

            for (int i = 0; i < chaves.Count; i++)
            {
                ItemDB item = (ItemDB)lista[chaves[i]];
                SqlParameter sqlp = new SqlParameter(item.NomeCampo,
item.getTipoCampo());
                sqlp.Value = item.ValorCampo;

                spParameters.Add(sqlp);
            }

            return spParameters;
        }

        private object executarComando(List<SqlParameter> lista,
string nomeSP, int tipoRetorno)
        {
            SqlConnection connection = new
SqlConnection(connectionString);
            SqlCommand cmd = new SqlCommand();

            cmd.CommandType = CommandType.StoredProcedure;
            cmd.CommandText = nomeSP;

            foreach (SqlParameter param in lista)
            {
                cmd.Parameters.Add(param);
            }

            cmd.Connection = connection;
            connection.Open();

            SqlTransaction transaction =
connection.BeginTransaction();
            cmd.Transaction = transaction;

            try
            {
                object retorno;

                if (tipoRetorno == RETORNA_DATATABLE)
                {
                    DataTable table = new DataTable();
                    SqlDataAdapter adapter = new SqlDataAdapter();
```

```
                    adapter.SelectCommand = cmd;
                    adapter.Fill(table);

                    retorno = table;
                }
                else if (tipoRetorno == RETORNA_DATASET)
                {
                    DataSet set = new DataSet();
                    SqlDataAdapter adapter = new SqlDataAdapter();

                    adapter.SelectCommand = cmd;
                    adapter.Fill(set);
                    retorno = set;
                }
                else if (tipoRetorno == RETORNA_CHAVE)
                {
                    retorno = cmd.ExecuteScalar();
                    transaction.Commit();
                }
                else
                {
                    retorno = cmd.ExecuteNonQuery();
                    transaction.Commit();
                }

                return retorno;
            }
            catch (SqlException e)
            {
                transaction.Rollback();
                throw new Exception("Erro em executarComando:\n\r
StackTrace: " + e.StackTrace + "\n\r Message: " + e.Message);
            }
            finally
            {
                connection.Close();
                connection.Dispose();
            }
        }

    }
}

using System;
using System.Collections.Generic;
using System.Text;
using System.Collections;
using System.Data;
using System.Data.Sql;


namespace MonografiaConsole.dao
{
    public class ColecaoItemDB
    {
        private Hashtable colecaoDeItens = new Hashtable();
        private string nomeSP;
        private ArrayList chaves = new ArrayList();
        private ArrayList listaDeChaves = new ArrayList();
```

```
        public string NomeSP
        {
            get { return nomeSP; }
            set { nomeSP = value; }
        }

        public void adicionarItemDB(ItemDB item)
        {
            chaves.Add(item.NomeCampo);
            colecaoDeItens.Add(item.NomeCampo, item);
        }

        public void adicionarItemDB(string nomeDoCampo, object
valorDoCampo, int constanteTipoCampo)
        {
            ItemDB item = new ItemDB();

            item.NomeCampo = nomeDoCampo;
            item.ValorCampo = valorDoCampo;
            item.setTipoCampo(constanteTipoCampo);

            chaves.Add(nomeDoCampo);
            colecaoDeItens.Add(item.NomeCampo, item);
        }

        public ArrayList getChaves()
        {
            return chaves;
        }

        public Hashtable getColecaoDeItensDB()
        {
            return colecaoDeItens;
        }

    }
}

using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.Sql;

namespace MonografiaConsole.dao
{
    public class ItemDB
    {
        private string nomeCampo;
        private Object valorCampo;
        private int tipoCampo;

        public string NomeCampo
        {
            get { return nomeCampo; }
            set { nomeCampo = value; }
        }

        public Object ValorCampo
        {
```

```
            get { return valorCampo; }
            set { valorCampo = value; }
        }

        public void setTipoCampo(int constante)
        {
            this.tipoCampo = constante;
        }

        public SqlDbType getTipoCampo()
        {
            return getTipoDBDataType(tipoCampo);
        }

        private SqlDbType getTipoDBDataType(int constante)
        {
            switch (constante)
            {
                case DatabaseActions.TIPO_INT:
                    return SqlDbType.Int;
                case DatabaseActions.TIPO_FLOAT:
                    return SqlDbType.Float;
                case DatabaseActions.TIPO_DATE:
                    return SqlDbType.DateTime;
                case DatabaseActions.TIPO_XML:
                    return SqlDbType.Xml;
                case DatabaseActions.TIPO_VARCHAR:
                    return SqlDbType.VarChar;
                case DatabaseActions.TIPO_NVARCHAR:
                    return SqlDbType.NVarChar;
                case DatabaseActions.TIPO_TEXTO:
                    return SqlDbType.Text;
                case DatabaseActions.TIPO_MONEY:
                    return SqlDbType.Money;
                default:
                    return SqlDbType.VarChar;
            }
        }
    }
}
```

Camada de Lógica de Negócio

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using MonografiaConsole.dao;
using MonografiaConsole.model;

namespace MonografiaConsole.controller
{
    public class InsertMachine
    {
        /*
         * Método de inserção de novas máquinas
         */
```

```
        public static int newMachine(MachineLocal machine)
        {
            int retorno = 0;
            try
            {
                ColecaoItemDB colecao = new ColecaoItemDB();

                colecao.adicionarItemDB("@ProcessorModel",
machine.CpuModel.ToString(), DatabaseActions.TIPO_VARCHAR);
                colecao.adicionarItemDB("@ProcessorManufacturer",
machine.CpuManufacturer.ToString(), DatabaseActions.TIPO_VARCHAR);
                colecao.adicionarItemDB("@MotherBoardModel",
machine.MotherBoardModel.ToString(), DatabaseActions.TIPO_VARCHAR);
                colecao.adicionarItemDB("@MotherBoardManufacturer",
machine.MotherBoardManufacturer.ToString(),
DatabaseActions.TIPO_VARCHAR);
                colecao.adicionarItemDB("@HdModel",
machine.HdModel.ToString(), DatabaseActions.TIPO_VARCHAR);
                colecao.adicionarItemDB("@HdCapacity",
machine.HdCapacity.ToString(), DatabaseActions.TIPO_FLOAT);
                colecao.adicionarItemDB("@NtAdapterName",
machine.NtAdapterName.ToString(), DatabaseActions.TIPO_VARCHAR);
                colecao.adicionarItemDB("@NtAdapterMAC",
machine.NtAdapterMAC.ToString(), DatabaseActions.TIPO_VARCHAR);

                colecao.NomeSP = "INSERT_MACHINE";

                DatabaseActions dtactions = new DatabaseActions();

                retorno = dtactions.getChavePrimaria(colecao);
            }
            catch (Exception erro)
            {
                throw new Exception(erro.Message);
            }

            return retorno;
        }

        public static void InsertMachineSoftwares(MachineLocal
machine, int ID)
        {
            try
            {
                ItemDB id = new ItemDB();
                id.NomeCampo = "@id";
                id.setTipoCampo(DatabaseActions.TIPO_INT);
                id.ValorCampo = ID;

                ItemDB NtAdapterMAC = new ItemDB();

NtAdapterMAC.setTipoCampo(DatabaseActions.TIPO_VARCHAR);
                NtAdapterMAC.NomeCampo = "@NtAdapterMAC";
                NtAdapterMAC.ValorCampo = machine.NtAdapterMAC;

                ItemDB SoftwareCaption = new ItemDB();

SoftwareCaption.setTipoCampo(DatabaseActions.TIPO_VARCHAR);
                SoftwareCaption.NomeCampo = "@SoftwareCaption";
```

```
                ItemDB SoftwareInstallDate = new ItemDB();

SoftwareInstallDate.setTipoCampo(DatabaseActions.TIPO_VARCHAR);
                SoftwareInstallDate.NomeCampo =
"@SoftwareInstallDate";

                ItemDB SoftwareInstallLocation = new ItemDB();

SoftwareInstallLocation.setTipoCampo(DatabaseActions.TIPO_VARCHAR);
                SoftwareInstallLocation.NomeCampo =
"@SoftwareInstallLocation";

                ItemDB SoftwareVendor = new ItemDB();

SoftwareVendor.setTipoCampo(DatabaseActions.TIPO_VARCHAR);
                SoftwareVendor.NomeCampo = "@SoftwareVendor";

                ItemDB SoftwareVersion = new ItemDB();

SoftwareVersion.setTipoCampo(DatabaseActions.TIPO_VARCHAR);
                SoftwareVersion.NomeCampo = "@SoftwareVersion";

                ColecaoItemDB colecao = new ColecaoItemDB();

                DatabaseActions dtactions = new DatabaseActions();


                for (int icontador = 0; icontador <
machine.SoftwareCaption.Length; icontador++)
                {
                    colecao = null;
                    colecao = new ColecaoItemDB();

                    colecao.NomeSP = "INSERT_SOFTWARE_LIST";

                    SoftwareCaption.ValorCampo =
machine.SoftwareCaption[icontador].ToString();
                    SoftwareInstallDate.ValorCampo =
machine.SoftwareInstallDate[icontador].ToString();
                    SoftwareInstallLocation.ValorCampo =
machine.SoftwareInstallLocation[icontador].ToString();
                    SoftwareVendor.ValorCampo =
machine.SoftwareVendor[icontador].ToString();
                    SoftwareVersion.ValorCampo =
machine.SoftwareVersion[icontador].ToString();

                    colecao.adicionarItemDB(id);
                    colecao.adicionarItemDB(NtAdapterMAC);
                    colecao.adicionarItemDB(SoftwareCaption);
                    colecao.adicionarItemDB(SoftwareInstallDate);
                    colecao.adicionarItemDB(SoftwareInstallLocation);
                    colecao.adicionarItemDB(SoftwareVendor);
                    colecao.adicionarItemDB(SoftwareVersion);

                    dtactions.executarSP(colecao);
                }
            }
            catch (Exception erro)
            {
                throw new Exception(erro.Message);
```

```
            }
        }
    }
}


using System;
using System.Net.Mail;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using MonografiaConsole.model;
using MonografiaConsole.dao;
using System.Net;
using System.Collections;
using System.Diagnostics;
using System.Configuration;

namespace MonografiaConsole.controller
{
    public class VerifySoftwareList
    {
        //valida se existe algum software não permitido instalado na
máquina cliente
        public ArrayList validaSoftwares(MachineLocal machine,
ArrayList blackList)
        {
            ArrayList retorno = new ArrayList();

            if (blackList.Count > 0)
            {
                for (int i = 0; i < blackList.Count; i++)
                {
                    for (int iContSoftwares = 0; iContSoftwares <
machine.SoftwareCaption.Length; iContSoftwares++)
                    {
                        if
                        (

(machine.SoftwareCaption[iContSoftwares].ToString().ToLower() ==
blackList[i].ToString().Split(';')[0].ToString().ToLower()) &&

(machine.SoftwareVendor[iContSoftwares].ToString().ToLower() ==
blackList[i].ToString().Split(';')[1].ToString().ToLower()) &&

(machine.SoftwareVersion[iContSoftwares].ToString().ToLower() ==
blackList[i].ToString().Split(';')[2].ToString().ToLower())
                        )
                        {
                            retorno.Add("<strong>Nome:</strong> " +
machine.SoftwareCaption[iContSoftwares].ToString() + ";
<strong>Fabricante:</strong> " +
machine.SoftwareVendor[iContSoftwares].ToString() + ";
<strong>Versão:</strong> " +
machine.SoftwareVersion[iContSoftwares].ToString());
                        }
                    }
                }
            }
```

```
            return retorno;
        }

        //gera a lista com os softwares não permitidos
        public static ArrayList BlackList()
        {
            ArrayList retorno = new ArrayList();

            ColecaoItemDB colecao = new ColecaoItemDB();
            colecao.NomeSP = "SELECT_SOFTWARE_BLACKLIST";

            DatabaseActions databaseActions = new DatabaseActions();
            DataTable dtResultado =
databaseActions.getDataTable(colecao);

            for (int icont = 0; icont < dtResultado.Rows.Count;
icont++)
            {
                //retorno.Add(dtResultado.Rows[icont][0].ToString() +
";" + dtResultado.Rows[icont][1].ToString() + ";" +
dtResultado.Rows[icont][2].ToString());
                retorno.Add(dtResultado.Rows[icont][1].ToString() +
";" + dtResultado.Rows[icont][2].ToString() + ";" +
dtResultado.Rows[icont][3].ToString());
            }

            return retorno;
        }

        // ==============================
        // rotina de envio de e-mail
        // ==============================
        public static string sendMail(string from, string To, string
msg, string subject, string smptServer, int portSmtpServer)
        {
            string retorno = "";

            try
            {
                MailMessage Message = new MailMessage();
                Message.From = new MailAddress(from);

                Message.To.Add(new MailAddress(To));

                Message.Subject = subject;
                Message.IsBodyHtml = true;
                Message.Body = msg;

                NetworkCredential SMTPUserInfo = new
NetworkCredential(ConfigurationSettings.AppSettings["user"].ToString()
, ConfigurationSettings.AppSettings["pass"].ToString());

                SmtpClient client = new SmtpClient(smptServer,
portSmtpServer);
                client.UseDefaultCredentials = false;
                client.Credentials = SMTPUserInfo;
                client.Send(Message);

                retorno = "Mensagem enviada com sucesso!";
```

```
            }
            catch (SmtpException erro)
            {
                retorno += "\n";
                retorno += "Message: " + erro.Message + "\n";
                retorno += "InnerException: " +
erro.InnerException.ToString() + "\n";
                retorno += "Source: " + erro.Source.ToString() + "\n";
                retorno += "StackTrace: " + erro.StackTrace.ToString()
+ "\n";
                retorno += "StatusCode: " + erro.StatusCode.ToString()
+ "\n";
            }

            return retorno;
        }
    }
}
```

Windows Service

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.ServiceProcess;
using System.Text;
using MonografiaConsole.model;
using MonografiaConsole.controller;
using WindowsServiceMonografia.controller;
using System.Collections;

namespace WindowsServiceMonografia
{
    public partial class ServiceMonografia : ServiceBase
    {
        public ServiceMonografia()
        {
            InitializeComponent();
        }

        protected override void OnStart(string[] args)
        {
            try
            {
                MachineLocal machine = new MachineLocal();
                InsertMachine.InsertMachineSoftwares(machine,
InsertMachine.newMachine(machine));

                Console.WriteLine("Máquina inserida com sucesso!");

                VerifySoftwareList verify = new VerifySoftwareList();

                ArrayList BlackList = VerifySoftwareList.BlackList();
                ArrayList valida = verify.validaSoftwares(machine,
BlackList);
```

```
                if (valida.Count > 0)
                {
                    string msg = "<span style=\"font-family:Tahoma;
font-size:13pt;\"><strong>Automação de Inventário de
Software</strong><span><br>";
                    msg += "<span style=\"font-family:Tahoma; font-
size:12pt;\">A computador abaixo está com softwares instalados que não
são permitidos.</span><br><br>";
                    msg += "<span style=\"font-family:Tahoma; font-
size:11pt;\"><strong>Informações básicas:</strong></span><br>";
                    msg += "<span style=\"font-family:Tahoma; font-
size:9pt;\">";
                    msg += "<strong>&raquo; Processador:</strong>
" + machine.CpuModel + ", " + machine.CpuManufacturer + "<br>";
                    msg += "<strong>&raquo; Placa Mãe:</strong> "
+ machine.MotherBoardModel + ", " + machine.MotherBoardManufacturer +
"<br>";
                    msg += "<strong>&raquo; Adaptador de
rede:</strong> " + machine.NtAdapterMAC + ", " + machine.NtAdapterName
+ "<br>";
                    msg += "<strong>&raquo; Disco
Rígido:</strong> " + machine.HdModel + ", " +
Convert.ToString(machine.HdCapacity / (1024 * 1024));
                    msg += "</span>";
                    msg += "<br><br>";

                    msg += "<span style=\"font-family:Tahoma; font-
size:11pt;\"><strong>Lista de softwares não
permitidos:</strong></span><br>";

                    msg += "<span style=\"font-family:Tahoma; font-
size:9pt;\">";
                    for (int iValida = 0; iValida < valida.Count;
iValida++)
                    {
                        msg += "<strong>&raquo; </strong>" +
valida[iValida].ToString() + "<br>";
                    }
                    msg += "</span>";

                    Console.WriteLine
                    (
                        VerifySoftwareList.sendMail
                        (

ConfigurationSettings.AppSettings["smtpFrom"].ToString(),

ConfigurationSettings.AppSettings["smtpTo"].ToString(),
                            msg,

ConfigurationSettings.AppSettings["MailMsgSubtitle"].ToString(),

ConfigurationSettings.AppSettings["smtpServer"].ToString(),

Convert.ToInt32(ConfigurationSettings.AppSettings["smtpPort"].ToString
())
                        )
                    );
                }
```

```
            }
            catch (Exception erro)
            {
                Console.WriteLine("Erro: " + erro.Message);
            }
        }

        protected override void OnStop()
        {
        }
    }
}
```

Interface Gráfica de Usuário

Login.aspx (responsável pelo login do usuário)

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="login.aspx.cs" Inherits="MonografiaWEB.view.login.login"
Title="Automação de Inventário de Software" %>

<%@ Register Assembly="AjaxControlToolkit"
Namespace="AjaxControlToolkit" TagPrefix="cc1" %>

<%@ Register Assembly="System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35"
    Namespace="System.Web.UI" TagPrefix="asp" %>

<!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></title>
    <style type="text/css">
        #divLogin
        {
            position:absolute;
            width:250px;
            height:auto;
            left:50%;
            top:50%;
            margin-left:-125px;
            margin-top:-170px;
        }

        #divBox
        {
            position: absolute;
            width:650px;
            height: 170px;
            left:50%;
            top:50%;
            margin-left:-325px;
            margin-top:-100px;
```

```
            border:solid 2px #CBDCED;
        }

        #divBoxInside
        {
            position: relative;
            width: 400px;
            height: auto;
            margin-left:-250px;
            margin-top:-60px;
            left:50%;
            top:50%;
        }

        #divTit
        {
          background-color: #CBDCED;
          width:100%;
          height:50px;
          font-family: Arial;
          font-size:15pt;
          font-weight:bold;
          color:White;
        }

        #divCabecalho
        {
            position: relative;
            left:20%;
            font-family: Arial;
            font-size:9px;
            font-weight:bold;
        }
    </style>
</head>
<body style="margin-left:0px; margin-top:0px;">

    <div>
        <div>
            <table style="border:solid 0px black; width:100%;">
                <tr style="vertical-align: bottom;">
                    <td align="left">
                        <img src="../resources/img/logo_uniceub.gif"
alt="Projeto Final - Automação de inventário de software." />  
                        <font style="font-family:Arial; font-size:9px;
font-weight:bold;">AUTOMAÇÃO DE INVENTÁRIO DE SOFTWARE</font>
                    </td>
                </tr>
            </table>
        </div>
        <hr style="border-style:dotted; border-width:1px; border-
color:Black;" />
    </div>

    <form id="form1" runat="server">

    <div>
        <div style="position:relative; left:20px; top:20px;">
            <img src="../resources/img/ico_logo_3.png" alt="Automação
de inventário de software." width="150" />
```

```
        </div>
    </div>

    <div id="divTit">
        <asp:Label ID="LabelTit" runat="server" Text="Login"
style="position:relative; left:160px; top:10px;"></asp:Label>
    </div>

    <div id="divBox">

        <div id="divBoxInside">

            <div>
                <asp:Login ID="LoginMonografia" runat="server"
                    DestinationPageUrl="~/view/restrict/Home.aspx"
                    FailureAction="Refresh"
                    Font-Names="Arial"
                    Font-Size="9">
                    <LayoutTemplate>
                        <table border="0" cellpadding="1"
cellspacing="0"
                            style="border-collapse:collapse;">
                            <tr>
                                <td>
                                    <table border="0" cellpadding="0">
                                        <tr>
                                            <td align="center"
colspan="2"

style="color:White;font-weight:bold;">
                                                </td>
                                        </tr>
                                        <tr>
                                            <td align="right">
                                                <asp:Label
ID="UserNameLabel" runat="server"
AssociatedControlID="UserName"><strong>Nome de
usuário</strong></asp:Label>   
                                            </td>
                                            <td>
                                                <asp:TextBox
ID="UserName" runat="server" Width="200px"></asp:TextBox>

<asp:RequiredFieldValidator ID="UserNameRequired" runat="server"

ControlToValidate="UserName" ErrorMessage="O campo 'Nome do usuário' é
obrigatório."
                                                    ToolTip="O campo
'Nome do usuário' é obrigatório."
ValidationGroup="LoginMonografia">*</asp:RequiredFieldValidator>
                                            </td>
                                        </tr>
                                        <tr>
                                            <td align="right">
                                                <asp:Label
ID="PasswordLabel" runat="server"
AssociatedControlID="Password"><strong>Senha</strong></asp:Label> 
;  
                                            </td>
                                            <td>
```

```
                                                    <asp:TextBox
ID="Password" runat="server" TextMode="Password"
Width="200px"></asp:TextBox>

<asp:RequiredFieldValidator ID="PasswordRequired" runat="server"

ControlToValidate="Password" ErrorMessage="O campo 'Senha' é
obrigatório."
                                                        ToolTip="O campo
'Senha' é obrigatório.."
ValidationGroup="LoginMonografia">*</asp:RequiredFieldValidator>
                                                    </td>
                                                </tr>
                                                <tr>
                                                    <td></td>
                                                    <td align="left"
style="color:Red;">
                                                        <asp:Literal
ID="FailureText" runat="server" EnableViewState="False"></asp:Literal>
                                                    </td>
                                                </tr>
                                                <tr>
                                                    <td></td>
                                                    <td align="left">
                                                        <asp:Button
ID="LoginButton" runat="server" CommandName="Login" Text="Log In"

ValidationGroup="LoginMonografia" Width="55" />
                                                    </td>
                                                </tr>
                                            </table>
                                        </td>
                                    </tr>
                                </table>
                            </LayoutTemplate>
                            <TitleTextStyle BackColor="#6B696B" Font-
Bold="True" ForeColor="#FFFFFF" />
                        </asp:Login>
                    </div>

                </div>

            </div>

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

Home.aspx (possui os links para as funcionalidades)

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeBehind="MasterPage1.master.cs"
Inherits="MonografiaWEB.view.templates.MasterPage1" %>
<%@ Register Assembly="System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35"
Namespace="System.Web.UI" TagPrefix="asp" %>
```

```
<!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></title>

    <link href="../resources/css/css.css" rel="stylesheet"
type="text/css" />
    <link href="../resources/css/SpryCollapsiblePanel.css"
rel="stylesheet" type="text/css" />
    <script src="../resources/javascript/SpryCollapsiblePanel.js"
type="text/javascript"></script>

    <style>
        #divTit
        {
          background-color: #CBDCED;
          width:100%;
          height:50px;
          font-family: Arial;
          font-size:15pt;
          font-weight:bold;
          color:White;
        }
    </style>

</head>
<body style="margin-left:0px; margin-top:0px;">
    <div>
        <div>
            <table style="border:solid 0px black; width:100%;">
                <tr style="vertical-align: bottom;">
                    <td align="left">
                        <img src="../resources/img/logo_uniceub.gif"
alt="Projeto Final - Automação de inventário de software." />  
                        <font style="font-family:Arial; font-size:9px;
font-weight:bold;">AUTOMAÇÃO DE INVENTÁRIO DE SOFTWARE</font>
                    </td>
                    <td align="right">
                        <asp:LoginView ID="LoginViewMonografia"
runat="server">
                            <LoggedInTemplate>
                                Olá <strong><asp:LoginName
ID="MonoLoginName" runat="server" /></strong>. Seja Bem-Vindo! 
                                <asp:HyperLink ID="HyperLinkLogout"
runat="server"
NavigateUrl="~/view/login/logout.aspx">[Logout]</asp:HyperLink>
                                  
                            </LoggedInTemplate>
                        </asp:LoginView>
                    </td>
                </tr>
            </table>
        </div>
        <hr style="border-style:dotted; border-width:1px; border-
color:Black;" />
    </div>

    <form id="form1" runat="server">
```

```
        <div>
            <div style="position:relative; left:20px; top:20px; z-
index:2">
                <img src="../resources/img/ico_logo_3.png"
alt="Automação de inventário de software." width="150" />
            </div>
        </div>

        <br />

        <div id="divTit" style="position: absolute; top:110px; z-
index:1;">
            <asp:ContentPlaceHolder ID="head" runat="server">
            </asp:ContentPlaceHolder>
        </div>

        <div style="position:absolute; top:160px; z-index:3;">
            <table style="width:100%; border:solid 0 black;
background-color:#E3EAEB;">
                <tr>
                    <td align="left">
                        <asp:Menu ID="MenuMonografia" runat="server"
BackColor="#E3EAEB"
                            DynamicHorizontalOffset="2" Font-
Names="Tahoma" Font-Size="7pt"
                            ForeColor="#666666"
Orientation="Horizontal" StaticSubMenuIndent="10px"
                            Font-Bold="True">
                            <StaticSelectedStyle BackColor="#1C5E55"
/>
                            <StaticMenuItemStyle
HorizontalPadding="5px" VerticalPadding="2px" />
                            <DynamicHoverStyle BackColor="#666666"
ForeColor="White" />
                            <DynamicMenuStyle BackColor="#E3EAEB" />
                            <DynamicSelectedStyle BackColor="#1C5E55"
/>
                            <DynamicMenuItemStyle
HorizontalPadding="5px" VerticalPadding="2px" />
                            <StaticHoverStyle BackColor="#666666"
ForeColor="White" />
                            <Items>
                                <asp:MenuItem
NavigateUrl="~/view/restrict/Home.aspx" Text="Home">
                                </asp:MenuItem>
                                <asp:MenuItem
NavigateUrl="~/view/restrict/Inventario.aspx" Text="Inventário"
                                    Value="Inventário"></asp:MenuItem>
                                <asp:MenuItem
NavigateUrl="~/view/restrict/blacklist.aspx" Text="Softwares não
Permitidos"
                                    Value="Softwares não
Permitidos"></asp:MenuItem>
                                <asp:MenuItem
NavigateUrl="~/view/restrict/RegisterSoftwares.aspx"
                                    Text="Registrar softwares não
permitidos" Value="Registrar softwares não permitidos">
                                </asp:MenuItem>
```

```
                                <asp:MenuItem
NavigateUrl="~/view/restrict/Pesquisar.aspx"
                                    Text="Pesquisar"
Value="Pesquisar">
                                </asp:MenuItem>
                            </Items>
                        </asp:Menu>
                    </td>
                </tr>
            </table>
        </div>

        <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
runat="server">

        </asp:ContentPlaceHolder>

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

Inventario.aspx (página responsável pela apresentação do inventário)

```
<%@ Page Title="" Language="C#"
MasterPageFile="~/view/templates/MasterPage1.Master"
AutoEventWireup="true" CodeBehind="Inventario.aspx.cs"
Inherits="MonografiaWEB.view.templates.Inventario" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
    <asp:Label ID="LabelHead" runat="server" Text="Inventário"
style="position:absolute; left:170px; top:10px;"></asp:Label>
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">
    <div style="position:relative; top:30px;">
        <asp:GridView ID="GridViewInventario" runat="server"
Width="100%"
            AllowPaging="True" PageSize="30" AllowSorting="True"
AutoGenerateColumns="False"
            CellPadding="4" DataSourceID="SqlDataSourceInventario"
ForeColor="#333333"
            GridLines="None" ShowFooter="True" DataKeyNames="ID">
            <RowStyle BackColor="#EFF3FB" />
            <Columns>

                <asp:HyperLinkField DataNavigateUrlFields="id"

DataNavigateUrlFormatString="detalhes.aspx?machine={0}" Text="Ver
Detalhes" ItemStyle-Font-Size="8pt" />

                <asp:BoundField DataField="Processor"
HeaderText="Processador" ReadOnly="True"
                    SortExpression="Processor">
                <HeaderStyle HorizontalAlign="Left" />
                </asp:BoundField>
                <asp:BoundField DataField="MotherBoard"
HeaderText="Placa mãe" ReadOnly="True"
                    SortExpression="MotherBoard">
```

```
                <HeaderStyle HorizontalAlign="Left" />
                </asp:BoundField>
                <asp:BoundField DataField="NtAdapter"
HeaderText="Adaptador de rede"
                    ReadOnly="True" SortExpression="NtAdapter">
                <HeaderStyle HorizontalAlign="Left" />
                </asp:BoundField>
                <asp:BoundField DataField="HD" HeaderText="HD"
ReadOnly="True"
                    SortExpression="HD" DataFormatString="{0}">
                <HeaderStyle HorizontalAlign="Left" />
                </asp:BoundField>
                <asp:BoundField DataField="Data_Coleta"
HeaderText="Data da coleta"
                    SortExpression="Data_Coleta">
                <HeaderStyle HorizontalAlign="Left" />
                </asp:BoundField>
                <asp:CommandField />
            </Columns>
            <FooterStyle BackColor="" Font-Bold="True"
ForeColor="White" />
            <PagerStyle BackColor="#2461BF" ForeColor="White"
HorizontalAlign="Center" />
            <SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True"
ForeColor="#333333" />
            <HeaderStyle BackColor="#507CD1" Font-Bold="True"
ForeColor="White" />
            <EditRowStyle BackColor="#2461BF" />
            <AlternatingRowStyle BackColor="White" />
        </asp:GridView>
    </div>
    <asp:SqlDataSource ID="SqlDataSourceInventario" runat="server"
        ConnectionString="<%$
ConnectionStrings:monografiaConnectionString %>"
        SelectCommand="SELECT_MACHINES"
SelectCommandType="StoredProcedure">
    </asp:SqlDataSource>
</asp:Content>
```

Detalhes.aspx (página que possui os detalhes do inventário)

```
<%@ Page Title="" Language="C#"
MasterPageFile="~/view/templates/MasterPage1.Master"
AutoEventWireup="true" CodeBehind="Detalhes.aspx.cs"
Inherits="MonografiaWEB.view.templates.Detalhes" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
    <asp:Label ID="LabelHead" runat="server" Text="Detalhes"
style="position:relative; left:170px; top:10px;"></asp:Label>
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">

    <div id="CollapsiblePanel1" class="CollapsiblePanel"
style="position:relative; top:30px;">
        <div>
```

```
            <input id="ButtonVoltar" type="button" value="<< Voltar"
runat="server" style="font-family:Tahoma; font-size:11px; font-
weight:bold;" onclick="javascript:history.back();" />
        </div>
        <div style="position: relative; left:2px;"
class="CollapsiblePanelTab" tabindex="0">
            <asp:Literal ID="LiteralDetalhes"
runat="server"></asp:Literal>
        </div>
        <div class="CollapsiblePanelContent">
            <asp:Literal ID="LiteralDetalhes2"
runat="server"></asp:Literal>
        </div>
    </div>

    <script type="text/javascript">
        var CollapsiblePanel1 = new
Spry.Widget.CollapsiblePanel("CollapsiblePanel1");
    </script>

</asp:Content>

using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Monografia.DAO;
using System.Data;

namespace MonografiaWEB.view.templates
{
    public partial class Detalhes : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!(Page.IsPostBack))
            {
                if
(!(String.IsNullOrEmpty((string)Request.QueryString["machine"])))

getDetails((string)Request.QueryString["machine"]);
            }
        }

        // =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
        // =+=+=+=+ método que gera a saída dos detalhes =+=+=+=+=+
        // =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
        protected void getDetails(string ID)
        {
            LiteralDetalhes.Text = "";
            LiteralDetalhes2.Text = "";

            ColecaoItemDB colecao = new ColecaoItemDB();
            colecao.adicionarItemDB("@ID", ID,
DatabaseActions.TIPO_INT);
            colecao.NomeSP = "SELECT_MACHINES_SOFTWARES";

            DatabaseActions dataBaseActions = new DatabaseActions();
```

```
            DataTable tabela = dataBaseActions.getDataTable(colecao);

            if (tabela.Rows.Count > 0)
            {
                for (int iContador = 0; iContador < tabela.Rows.Count;
iContador++)
                {
                    if (iContador == 0)
                    {
                        LiteralDetalhes.Text += "<br />";
                        LiteralDetalhes.Text += "<div
style=\"background-color: #CBDCED; font-family:Arial; font-size:18px;
color:#fff;\"><img src=\"../resources/img/arrow_details.png\"
/> " + tabela.Rows[iContador]["ID"] + " - Dados Gerais</div>";
                        LiteralDetalhes.Text += "<div
style=\"background-color: #F2F2F2; font-family:Arial; font-
size:12px;\">";
                        LiteralDetalhes.Text +=
"<strong>Processador:</strong> " + tabela.Rows[iContador]["Processor"]
+ "<br />";
                        LiteralDetalhes.Text += "<strong>Placa
Mãe:</strong> " + tabela.Rows[iContador]["MotherBoard"] + "<br />";
                        LiteralDetalhes.Text += "<strong>Adaptador de
Rede:</strong> " + tabela.Rows[iContador]["NtAdapter"] + "<br />";
                        LiteralDetalhes.Text += "<strong>HD:</strong>
" + tabela.Rows[iContador]["HD"] + "<br />";
                        LiteralDetalhes.Text += "<strong>Data/Hora da
Coleta:</strong> " + tabela.Rows[iContador]["Data_Coleta"] + "<br />";
                        LiteralDetalhes.Text += "</div>";
                        LiteralDetalhes.Text += "<br />";
                        LiteralDetalhes2.Text += "<div
style=\"background-color: #CBDCED; font-size:18px; color:#fff;\"><img
src=\"../resources/img/arrow_details.png\"> Softwares
Instalados</div>";
                    }

                    LiteralDetalhes2.Text += "<strong>Nome:</strong> "
+ tabela.Rows[iContador]["Caption"] + "<br />";
                    LiteralDetalhes2.Text += "<strong>Data de
instalação:</strong> " + tabela.Rows[iContador]["InstallDate"] + "<br
/>";
                    LiteralDetalhes2.Text += "<strong>Local de
instalação:</strong> " + tabela.Rows[iContador]["InstallLocation"] +
"<br />";
                    LiteralDetalhes2.Text +=
"<strong>Fabricante:</strong> " + tabela.Rows[iContador]["Vendor"] +
"<br />";
                    LiteralDetalhes2.Text += "<strong>Versão:</strong>
" + tabela.Rows[iContador]["Version"];
                    LiteralDetalhes2.Text += "<hr />";
                }
            }

        }
    }
}
```

Blacklist.aspx (página que lista os softwares não permitidos)

```
<%@ Page Title="" Language="C#"
MasterPageFile="~/view/templates/MasterPage1.Master"
AutoEventWireup="true" CodeBehind="blacklist.aspx.cs"
Inherits="MonografiaWEB.view.templates.blacklist" %>
<%@ Register Assembly="System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35"
Namespace="System.Web.UI" TagPrefix="asp" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
    <asp:Label ID="LabelHead" runat="server" Text="Softwares Não
Permitidos" style="position:relative; left:170px;
top:10px;"></asp:Label>
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">

    <div style="position:relative; top:30px;">
        <asp:Panel ID="PanelMsg" runat="server" BorderStyle="Double"
BorderWidth="2" BorderColor="Red" BackColor="#ffffcc" Visible="false">
            <asp:Label ID="LabelMsg" runat="server"
Text=""></asp:Label>
        </asp:Panel>

        <br />

        <asp:GridView ID="GridViewBlackList" runat="server"
            AllowPaging="True"
            AllowSorting="True"
            AutoGenerateColumns="False"
            CellPadding="4"
            DataKeyNames="id"
            DataSourceID="SqlDataSourceBlackList"
            ForeColor="#333333"
            GridLines="None"
            PageSize="15"
            Width="100%">

            <RowStyle BackColor="#EFF3FB" />
            <Columns>

                <asp:CommandField
EditImageUrl="~/view/resources/img/btn_edit.png"
                    ShowEditButton="True" CancelText="Cancelar"
                    EditText="Editar" InsertText="Inserir"
InsertVisible="False" NewText="Novo"
                    UpdateText="Atualizar">
                <HeaderStyle HorizontalAlign="Left"
VerticalAlign="Middle" />
                </asp:CommandField>

                <asp:BoundField DataField="id" HeaderText="id"
ReadOnly="True"
                    SortExpression="id" InsertVisible="False" >
                <HeaderStyle HorizontalAlign="Left" />
                </asp:BoundField>

                <asp:BoundField DataField="SoftwareCaption"
HeaderText="SoftwareCaption" ReadOnly="False"
                    SortExpression="SoftwareCaption" >
```

```
                <HeaderStyle HorizontalAlign="Left" />
                </asp:BoundField>

                <asp:BoundField DataField="SoftwareVendor"
HeaderText="SoftwareVendor"
                    ReadOnly="False" SortExpression="SoftwareVendor" >
                <HeaderStyle HorizontalAlign="Left" />
                </asp:BoundField>

                <asp:BoundField DataField="SoftwareVersion"
HeaderText="SoftwareVersion"
                    ReadOnly="False" SortExpression="SoftwareVersion"
>
                <HeaderStyle HorizontalAlign="Left" />
                </asp:BoundField>

            </Columns>
            <FooterStyle BackColor="#507CD1" Font-Bold="True"
ForeColor="White" />
            <PagerStyle BackColor="#2461BF" ForeColor="White"
HorizontalAlign="Center" />
            <SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True"
ForeColor="#333333" />
            <HeaderStyle BackColor="#507CD1" Font-Bold="True"
ForeColor="White" />
            <EditRowStyle BackColor="#f2f2f2" />
            <AlternatingRowStyle BackColor="White" />
        </asp:GridView>

        <hr />
        <a href="RegisterSoftwares.aspx">cadastro de software</a>

    <asp:SqlDataSource ID="SqlDataSourceBlackList" runat="server"
        ConnectionString="<%$
ConnectionStrings:monografiaConnectionString %>"
        SelectCommand="SELECT_SOFTWARE_BLACKLIST"
SelectCommandType="StoredProcedure"
        UpdateCommand="UPDATE_SOFTWARE_BLACKLIST"
UpdateCommandType="StoredProcedure">
        <UpdateParameters>
            <asp:Parameter Name="id" Type="Int32" />
            <asp:Parameter Name="SoftwareCaption" Type="String" />
            <asp:Parameter Name="SoftwareVendor" Type="String" />
            <asp:Parameter Name="SoftwareVersion" Type="String" />
        </UpdateParameters>
    </asp:SqlDataSource>
</div>
</asp:Content>
```

RegisterSoftwares.aspx (página que permite a inclusão de softwares não permitidos)

```
<%@ Page Title="" Language="C#"
MasterPageFile="~/view/templates/MasterPage1.Master"
AutoEventWireup="true" CodeBehind="RegisterSoftwares.aspx.cs"
Inherits="MonografiaWEB.view.templates.RegisterSoftwares" %>
<%@ Register Assembly="AjaxControlToolkit"
Namespace="AjaxControlToolkit" TagPrefix="cc1" %>
```

```
<%@ Register Assembly="System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35"
Namespace="System.Web.UI" TagPrefix="asp" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
    <asp:Label ID="LabelHead" runat="server" Text="Registro de
softwares não permitidos" style="position:relative; left:170px;
top:10px;"></asp:Label>
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">
    <div style="position:relative; top:30px;">
        <div style="position:relative; left:50px; top:20px;
border:solid 2px #CBDCED; width:450px; height:180px;">
            <table style="border:solid 0px black; width:auto; font-
family:Arial; font-size:11px; font-weight:bold; position:relative;
left:20px; top: 20px">
                <tr>
                    <td>Nome</td>
                    <td>
                        <asp:TextBox ID="TextBoxNome"
runat="server"></asp:TextBox>
                        <asp:RequiredFieldValidator
ID="RequiredFieldValidatorTextBoxNome"       runat="server"
ErrorMessage="O campo <strong><u>'Nome'</u></strong> é obrigatório!"
ControlToValidate="TextBoxNome"
ValidationGroup="RegisterSoftware">O campo
<strong><u>'Nome'</u></strong> é
obrigatório!</asp:RequiredFieldValidator>
                    </td>
                </tr>
                <tr>
                    <td>Versão</td>
                    <td>
                        <asp:TextBox ID="TextBoxVersao"
runat="server"></asp:TextBox>
                        <asp:RequiredFieldValidator
ID="RequiredFieldValidatorTextBoxVersao"     runat="server"
ErrorMessage="O campo <strong><u>'Versão'</u></strong> é obrigatório!"
ControlToValidate="TextBoxVersao"
ValidationGroup="RegisterSoftware">O campo
<strong><u>'Versão'</u></strong> é
obrigatório!</asp:RequiredFieldValidator>
                    </td>
                </tr>
                <tr>
                    <td>Fabricante</td>
                    <td>
                        <asp:TextBox ID="TextBoxFabricante"
runat="server"></asp:TextBox>
                        <asp:RequiredFieldValidator
ID="RequiredFieldValidatorTextBoxFabricante" runat="server"
ErrorMessage="O campo <strong><u>'Fabricante'</u></strong> é
obrigatório!" ControlToValidate="TextBoxFabricante"
ValidationGroup="RegisterSoftware">O campo
<strong><u>'Fabricante'</u></strong> é
obrigatório!</asp:RequiredFieldValidator>
                    </td>
                </tr>
```

```
                <tr>
                    <td></td>
                    <td align="left">
                        <asp:Button ID="BtnInsertMachine"
runat="server" Text="Inserir" onclick="BtnInsertMachine_Click"
ValidationGroup="RegisterSoftware" />
                    </td>
                </tr>
                <tr>
                    <td></td>
                    <td align="left">
                        <asp:Label ID="LblMsg" runat="server"
Text=""></asp:Label>
                    </td>
                </tr>
            </table>
        </div>
    </div>
</asp:Content>
```

Pesquisar.aspx (página responsável pela busca de máquinas do inventário)

```
<%@ Page Title="" Language="C#"
MasterPageFile="~/view/templates/MasterPage1.Master"
AutoEventWireup="true" CodeBehind="Pesquisar.aspx.cs"
Inherits="MonografiaWEB.view.templates.Pesquisar" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
    <asp:Label ID="LabelHead" runat="server" Text="Pesquisar"
style="position:relative; left:170px; top:10px;"></asp:Label>
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">

    <div style="position:relative; top:30px;">
        <br />

        <div style="position:relative; vertical-align:middle;
left:100px;">
            <asp:Label ID="Labelpesquisar" runat="server" Text="MAC
ADDRESS: "
                Font-Names="Tahoma" Font-Size="11px" Font-
Bold="True"></asp:Label>
            <asp:TextBox ID="TextBoxPesquisar" runat="server" Font-
Names="Tahoma"
                Font-Size="11px" Width="300px"></asp:TextBox> 
            <asp:Button ID="ButtonPesquisar" runat="server"
Text="Pesquisar"
                onclick="ButtonPesquisar_Click" Font-Bold="True" Font-
Names="Tahoma"
                Font-Size="11px" />
             
            <asp:Label ID="LabelMsg" runat="server" Text="" Font-
Names="Tahoma" Font-Size="11px" Font-Bold="true"
ForeColor="Red"></asp:Label>
        </div>

        <hr />
```

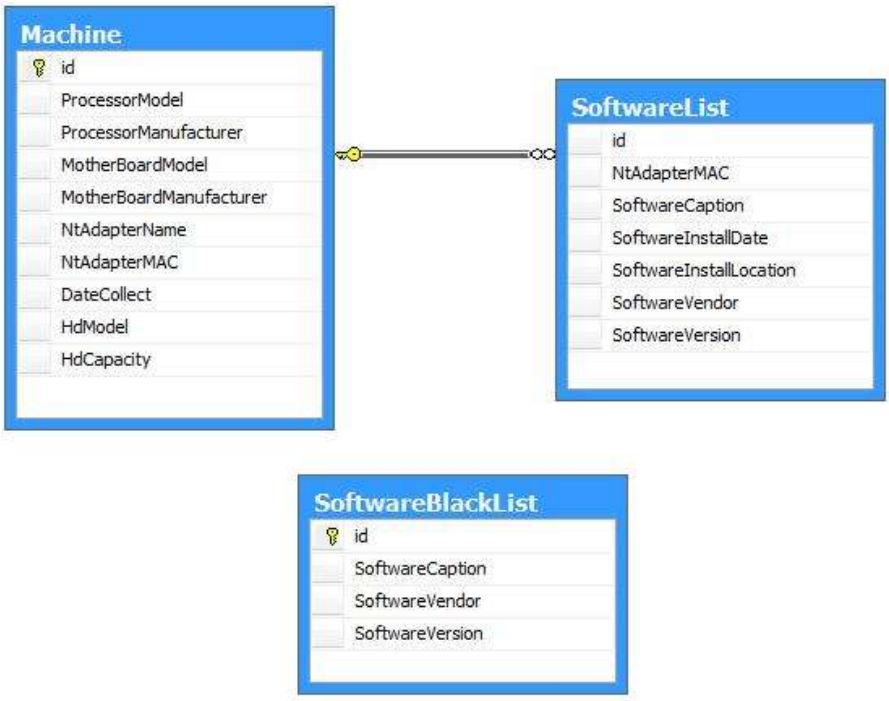
```
        <br />

        <div style="position:relative;">
            <asp:GridView ID="GridViewPesquisa" runat="server"
Width="100%"
                AllowPaging="True" AllowSorting="True"
AutoGenerateColumns="False"
                CellPadding="4" ForeColor="#333333"
                GridLines="None" ShowFooter="True" DataKeyNames="ID"
PageSize="30">
                <RowStyle BackColor="#EFF3FB" />
                <Columns>

                    <asp:HyperLinkField DataNavigateUrlFields="id"

DataNavigateUrlFormatString="detalhes.aspx?machine={0}" Text="Ver
detalhes" />

                    <asp:BoundField DataField="Processor"
HeaderText="Processador" ReadOnly="True"
                        SortExpression="Processor">
                    <HeaderStyle HorizontalAlign="Left" />
                    </asp:BoundField>
                    <asp:BoundField DataField="MotherBoard"
HeaderText="Placa mãe" ReadOnly="True"
                        SortExpression="MotherBoard">
                    <HeaderStyle HorizontalAlign="Left" />
                    </asp:BoundField>
                    <asp:BoundField DataField="NtAdapter"
HeaderText="Adaptador de rede"
                        ReadOnly="True" SortExpression="NtAdapter">
                    <HeaderStyle HorizontalAlign="Left" />
                    </asp:BoundField>
                    <asp:BoundField DataField="HD" HeaderText="HD"
ReadOnly="True"
                        SortExpression="HD" DataFormatString="{0}">
                    <HeaderStyle HorizontalAlign="Left" />
                    </asp:BoundField>
                    <asp:BoundField DataField="Data_Coleta"
HeaderText="Data da coleta"
                        SortExpression="Data_Coleta">
                    <HeaderStyle HorizontalAlign="Left" />
                    </asp:BoundField>
                    <asp:CommandField />
                </Columns>
                <PagerStyle ForeColor="Black" HorizontalAlign="Center"
/>
                <SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True"
ForeColor="#333333" />
                <HeaderStyle BackColor="#507CD1" Font-Bold="True"
ForeColor="White" />
                <EditRowStyle BackColor="#2461BF" />
                <AlternatingRowStyle BackColor="White" />
            </asp:GridView>
        </div>
    </div>

</asp:Content>

using System;
```

```
using System.Data;
using System.Data.SqlClient;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using MonografiaWEB.business;

namespace MonografiaWEB.view.templates
{
    public partial class Pesquisar : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void ButtonPesquisar_Click(object sender, EventArgs
e)
        {
            pesquisar(TextBoxPesquisar.Text, GridViewPesquisa);
        }

        /*
*****************************************************************
         * Desc.: Método do botão pesquisa que aciona a busca pelo
endereço MAC

*****************************************************************
*/
        private void pesquisar(string MAC, GridView GridViewPesquisa)
        {
            LabelMsg.Text = "";

            if (TextBoxPesquisar.Text.Equals(""))
            {
                LabelMsg.Text = "Não foram encontrados resultados para
a pesquisa";
                GridViewPesquisa.DataSource = null;
                GridViewPesquisa.DataBind();
            }
            else
            {
                /*Pesquisa*/
                DataTable DataTableSource =
SearchMachine.searchMachine(MAC);

                if (DataTableSource.Rows.Count < 1)
                {
                    LabelMsg.Text = "Não foram encontrados resultados
para a pesquisa '" + MAC + "'.";
                    GridViewPesquisa.DataSource = null;
                    GridViewPesquisa.DataBind();
                }
                else
                {
                    GridViewPesquisa.DataSource = DataTableSource;
                    GridViewPesquisa.DataBind();
                }
```

```
            }
        }
    }}
```

APÊNDICE B – MODELO DE ENTIDADE E RELACIONAMENTO

Para armazenar as informações foi usado o SQL Server 2008, um banco de dados relacional. Abaixo segue o relacionamento das tabelas usadas no desenvolvimento do projeto.



APÊNDICE C – DIAGRAMA DE CLASSES

Todo o projeto foi feito seguindo o modelo de orientação a objetos. Abaixo seguem as classes criadas no desenvolvimento da solução: