WILLIAN VAGNER VICENTE CORRÊA

PROPOSTA DE UMA FERRAMENTA PARA AUDITORIA DE SISTEMAS COMPUTACIONAIS

ASSIS
2009

PROPOSTA DE UMA FERRAMENTA PARA AUDITORIA DE SISTEMAS COMPUTACIONAIS

WILLIAN VAGNER VICENTE CORRÊA

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

Orientador: Prof. Dr. Almir Rogério Camolesi

Analisador (1): _____

Analisador (2): _____

ASSIS
2009

WILLIAN VAGNER VICENTE CORRÊA

PROPOSTA DE UIMA FERRAMENTA PARA AUDITORIA DE SISTEMAS COMPUTACIONAIS

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

Orientador: Prof. Dr. Almir Rogério Camolesi

Área de Concentração: _____

_____

ASSIS
2009

DEDICATÓRIA

Aos meus familiares sem os quais não chegaria até aqui.
Aos meus amigos e professores que ajudaram e me apoiaram nos momentos mais necessários.
Em especial a minha namorada Rafaela, por ser companheira em todos os momentos e sempre me dar força nas horas mais difíceis.

Sou eternamente grato a todos.

AGRADECIMENTOS

Primeiramente a Deus e a minha família, pelo empenho em me proporcionar todas as condições necessárias para que eu pudesse estudar.

A Todos os professores da instituição e amigos, que sempre estiveram ao meu lado quando precisei.

Agradeço fraternalmente a todos

RESUMO

Um dos objetivos da auditoria na área de informática é mapear as configurações de hardware e software em uma estação de trabalho. Entretanto as informações dos mapeamentos, em grande parte das empresas, são desenvolvidas manualmente. Neste caso, o auditor tem que capturar os dados inerentes ao hardware e aos softwares instalados em cada estação. Visando agilizar a referida captura, esse trabalho tem como objetivo desenvolver uma ferramenta para capturar, automaticamente as informações de uma determinada estação de trabalho, facilitando assim a auditoria das referidas informações. A ferramenta será composta de dois softwares. O primeiro será especifico para a coleta das informações das estações de trabalho. O segundo proporcionará o acesso às informações capturadas com a primeira aplicação através de uma interface rica para internet, facilitando assim a visualização de todas as informações contidas no banco de dados (BD) e disponibilizando dados cruciais para uma auditoria remota visando o conforto e agilidade na busca das referidas informações. Os sistemas de informações das empresas, bem como o acompanhamento permanente do ambiente informatizado. Salienta-se a necessidade de oferecer informações seguras aos tomadores de decisão.

Palavra Chave: RIA. Auditoria. Interatividade.

ABSTRACT

One of the objectives of the audit in computer science is to map the hardware settings and software on a workstation. However, the information from the maps in large firms, are performed manually. In this case, the auditor has to capture data relating to hardware and software installed in each station. In order to expedite such a catch, this work aims to develop a tool to capture, automatically the information from a particular workstation, thereby facilitating the audit of such information. The tool will consist of two software programs. The first is specific to the collection of information from workstations. The second will provide access to information captured with the first application through a rich interface for the Internet, thus facilitating the visualization of all information contained in the database (DB) and providing critical data to a remote audit to ensure the comfort and agility in search of such information. The information systems of companies, as well as the continued monitoring of the electronic environment. We emphasize the need to provide reliable information to decision makers.

Key words: RIA. Audit. Interactivity.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API	Applications Programming Interface
BD	Banco de Dados
BIOS	Basic Input Output System
HD	Hard Disk
J2EE	JAVA Enterprise Edition
LGPL	Lesser General Public LIcense
MXML	Macromedia Extensible Markup Language
RIA	Rich Internet Applications
SDK	Software Developmente Kit
SGBD	Sistema Gerenciador de Banco de Dados
SWF	Small Web Format
TI	Tecnologia da Informação
XML	Extensible Markup Language
WWW	World Wide Web

SUMÁRIO

1. INTRODUÇÃO

Neste trabalho é proposta a construção de uma ferramenta que auxilie a auditoria de sistemas de informações das empresas, bem como o acompanhamento permanente do ambiente informatizado. Desta forma, salienta-se a necessidade de oferecer informações seguras aos tomadores de decisão e, evidencia-se a relevância destas informações no ambiente empresarial.

Atualmente é possível presenciar um constante crescimento dos ativos de Tecnologia da Informação (TI) no meio sócio-econômico. A busca constante por processos automatizados que viabilizem a melhoria produtiva e a agilidade com menor custo, são pontos que sustentam o fato em questão.

Para alcançar a referida agilidade houve uma massificação dos computadores pessoais no meio empresarial, sendo que a maioria das empresas não possui um controle apurado de todas as informações processadas por um determinado equipamento. Dessa forma, julga-se necessário o desenvolvimento de um software, que faça a coleta das referidas informação de forma automatizada, como por exemplo: coletar informações como Modelo de Processador, Memória, *Hard Disk*, Placa de Rede, Placa de Vídeo, Placa Mãe, Bios, Sistema Operacional, Monitores, *Softwares* Instalados, Licenças Registradas, etc.

Para realizar tal tarefa o funcionário irá iniciar a coleta dos dados, utilizando um *software* que contenha validações e verificações. O *software* será responsável por fazer a importação das informações e verificar os possíveis relacionamentos entre as tabelas do banco de dados. Já o funcionário ficará responsável somente por confirmar o processamento final das informações.

Para auxiliar o trabalho deste aplicativo também será desenvolvido um *Website* contendo todo o conteúdo coletado pelo funcionário. Com base nesses dados serão desenvolvidos cadastros que ajudarão o funcionário nas atividades do dia a dia, como por exemplo: levantamento e atualização de um inventário de *hardwares e softwares*.

Com o desenvolvimento do referido produto, espera-se que a empresa tenha dados consistentes e confiáveis de todos os equipamentos, em nosso contexto os computadores pessoais, possibilitando assim que decisões relacionadas à atualização de *software*, *upgrade*, dentre outros sejam tomadas com uma maior confiabilidade por parte dos gestores.

2. AUDITORIA DE SISTEMAS COMPUTACIONAIS

Segundo (SIMCH, 2009, p13) Auditoria é uma atividade que engloba o exame de operações, processos, sistemas e responsabilidades gerenciais de uma determinada entidade, com intuito de verificar sua conformidade com certos objetivos e políticas institucionais, orçamentos, regras, normas ou padrões.

O campo da auditoria é composto pelo objeto a ser fiscalizado, período e natureza da auditoria.

O objeto pode ser uma entidade completa (instituição pública ou privada), uma parte selecionada ou uma função dessa entidade, e o período a ser fiscalizado pode ser de um mês, um ano ou até mesmo corresponder ao período completo da gestão de determinado administrador.

O âmbito da auditoria constitui-se da amplitude e exaustão dos processos de auditoria. Definem até que ponto será aprofundado as tarefas de auditoria e seu grau de abrangência.

Segundo (MARTINELLI AUDITORES, 2008, p7) algumas das razões que levaram a disseminação do uso dos sistemas de informação:

- Única maneira de fazer determinado trabalho;

- Melhorar a eficiência;

- Aplicar controles melhores;

- Reduzir custos.

O principal benefício que a tecnologia da informação traz para as organizações e a sua capacidade de melhorar a qualidade e a disponibilidade de informações e conhecimentos importantes para a empresa, seus clientes e fornecedores. Os sistemas de informação mais modernos oferecem às empresas oportunidades sem precedentes para a melhoria dos processos internos e dos serviços prestados ao consumidor final.

O sucesso das empresas passou a depender de sua capacidade de inovar nas áreas de produtos, serviços, canais e processos. Nesse contexto, a tecnologia da

informação assume papel crítico, permitindo às empresas modificar-se rapidamente e levar essas inovações até o mercado.

Há três razões principais para preocupar-se com segurança de sistemas de TI:

2.1 DEPENDÊNCIA DOS SISTEMAS DE INFORMAÇÃO

Sistemas que ofereçam serviços adequados e no tempo certo são a chave para a maioria das organizações atuais. Sem seus computadores e sistemas de comunicação, as empresas ficariam incapazes de fornecer serviços, processar faturas, contatar clientes e fornecedores ou efetuar pagamentos. Os sistemas de informação também armazenam dados sigilosos que, se tornados públicos, causariam embaraço e em alguns casos o fracasso da organização.

2.2 VULNERABILIDADE DOS SISTEMAS DE TI

Esses sistemas exigem um ambiente estável, podendo ser danificados por desastres naturais como fogo, inundação ou terremotos, falhas no controle de temperatura ou do suprimento de energia elétrica, acidentes ou sabotagens. Os sistemas de TI são a chave para acesso a vasta quantidade de dados corporativos, tornando-se alvo atraente para *hackers* e espiões, e podem motivar administradores de sistemas a abusar de seus privilégios, vendendo informações para terceiros. A organização depende da exatidão da informação fornecida pelos seus sistemas; se essa confiança for destruída, o impacto para a entidade pode ser comparado à própria destruição do sistema. Dessa forma é importante proteger dados tanto de corrupções acidentais quanto propositais.

2.3 INVESTIMENTO EM SISTEMAS DE TI

Os sistemas de informação são caros tanto no desenvolvimento quanto na manutenção, e a administração deve proteger esse investimento como qualquer

outro recurso valioso. Bens de TI são particularmente atrativos para ladrões, por serem em alguns casos portáteis, e poderem ser facilmente vendidos.

3. PROPOSTA DE GERENCIAMENTO DE SISTEMAS COMPUTACIONAIS

Neste capítulo serão apresentadas as formas de captura das informações utilizadas em um processo de auditoria computacional. É importante salientar que em primeiro momento o trabalho foca a método manual de captura das informações. Posteriormente, os aspectos inerentes à captura automática, via ferramenta, serão apresentados.

3.1 CAPTAÇÃO DE DADOS - MÉTODO MANUAL

Atualmente o processo de captação das informações de *hardwares* e *softwares* de muitas empresas é feito de forma manual, sendo que o funcionário a efetuar tal tarefa, tem que estar atento a vários detalhes no decorrer do cadastro, para assim poder capturar as informações relativas à estação de trabalhos.

Para efetuar o processo de cadastro, o funcionário utiliza planilhas eletrônicas que não possuem validações e verificações das informações que poderiam estar implementadas em um *software*. Assim, foram definidos “passos” a serem seguidos pelo funcionário para realizar tal tarefa sem ocorrer eventuais erros. A Figura 1 apresenta o processo manual que funcionário deverá seguir para concluir o cadastro das informações.

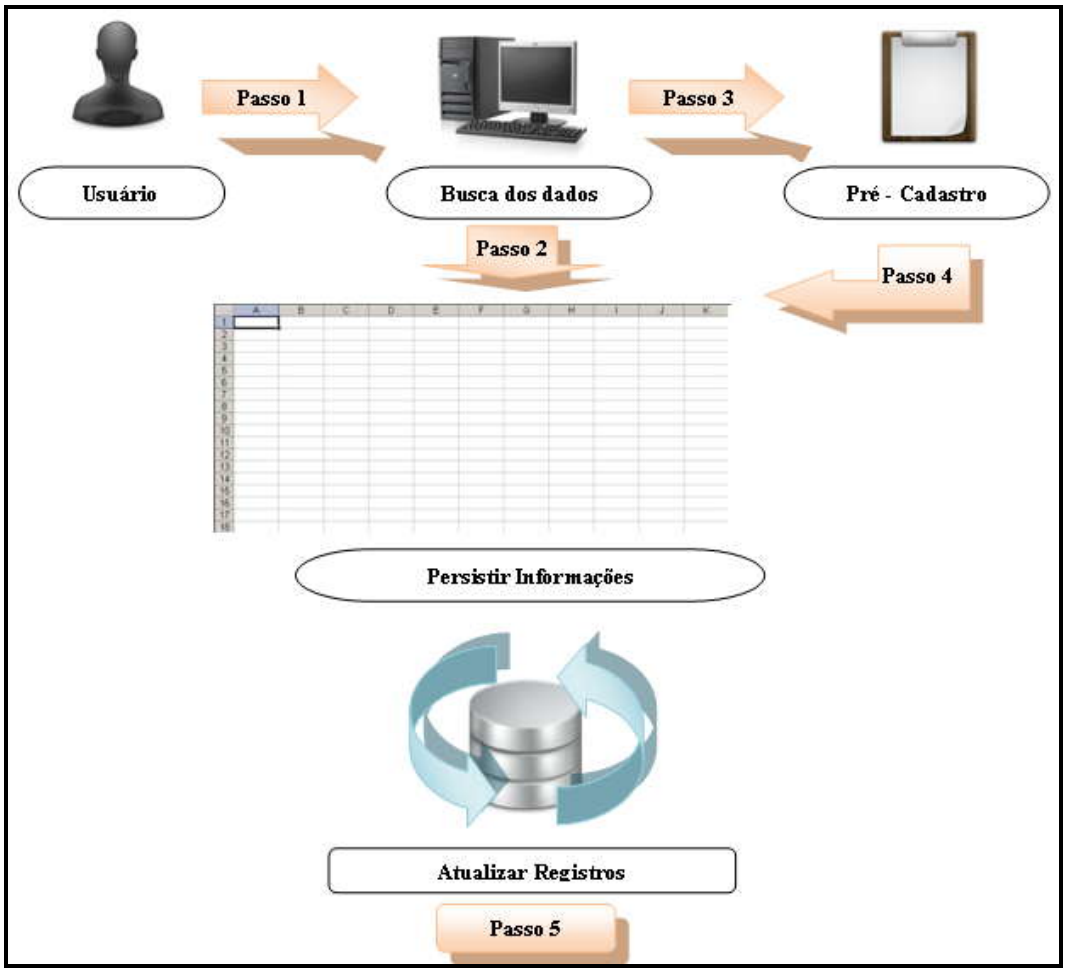

Figura 1 - Processo Manual

- Passo1

Inicialmente o funcionário se desloca até o equipamento e começa a fazer a coleta das informações da estação de trabalho de forma manual, tendo em vista que terá que ser feito vistoria criteriosa em todos os periféricos contendo números de série, marca, modelo, padrão de velocidade, tipo de software, chave de instalação, tamanho da aplicação entre outros. Torna cansativo, pois a estação de trabalho tem que ser reiniciada várias vezes para que a informação a ser coletada possa ser visualizada pelo funcionário, consumindo assim um tempo considerável até o final da coleta.

- Passo 2

Após a coleta das informações da estação de trabalho, caso a mesma esteja ligada em rede, o funcionário pode acessar um diretório pré-definido no servidor e centralizar todas as informações da referida estação em uma planilha eletrônica e dessa forma cadastrar os dados de maneira manual (um por um). Lembrando que o acesso à escrita na planilha torna-se exclusivo a uma pessoa, impossibilitando os demais funcionários de continuar cadastrando ao mesmo tempo, tornando a mesma somente leitura.

No processo de cadastro o funcionário terá que ficar atento, pois as planilhas eletrônicas não contêm validações e verificações dos dados em questão facilitando a margem de erros de registros inconsistentes.

- Passo 3

Caso a referida estação de trabalho não esteja conectada a um ponto de rede, a captura das informações deverá ser realizada utilizando uma Prancheta contendo uma folha de papel, seguindo todos os critérios adotados no “passo 1”.

- Passo 4

Com as informações em mãos o funcionário começa a cadastrar os dados na planilha eletrônica seguindo os requisitos do passo 2.

- Passo 5

Quando houver a necessidade de atualizar os registros de algumas informações da planilha eletrônica, o funcionário terá que localizar a linha e coluna pertencente a aquela estação de trabalho, fazer as modificações necessárias, sempre com atenção para não modificar dados não relacionados à atualização, pois a mesma não possui validações e verificações de erros.

3.2 CAPTAÇÃO DE DADOS - MÉTODO AUTOMÁTICO

Com o desenvolvimento de uma aplicação para a captação das informações de maneira automática em relação ao método manual, a produtividade e a portabilidade das informações dos mesmos serão superiores no quesito “Tempo”. Pois comparado ao tempo total gasto na captura das informações das estações de trabalho no

método manual, o método automático terá uma redução no tempo de coleta em torno de 97% assim sendo a utilização da aplicação torna-se indispensável, pois irá reduzir a mão de obra consideravelmente.

Visando que as informações coletadas serão validadas e verificadas conforme o necessário, o funcionário terá um controle e um padrão para todas as informações contando com o conforto e tranqüilidade no processo do mesmo. A Figura 2 apresenta a forma que o funcionário deverá seguir para concluir o cadastro das informações usando os aplicativos.

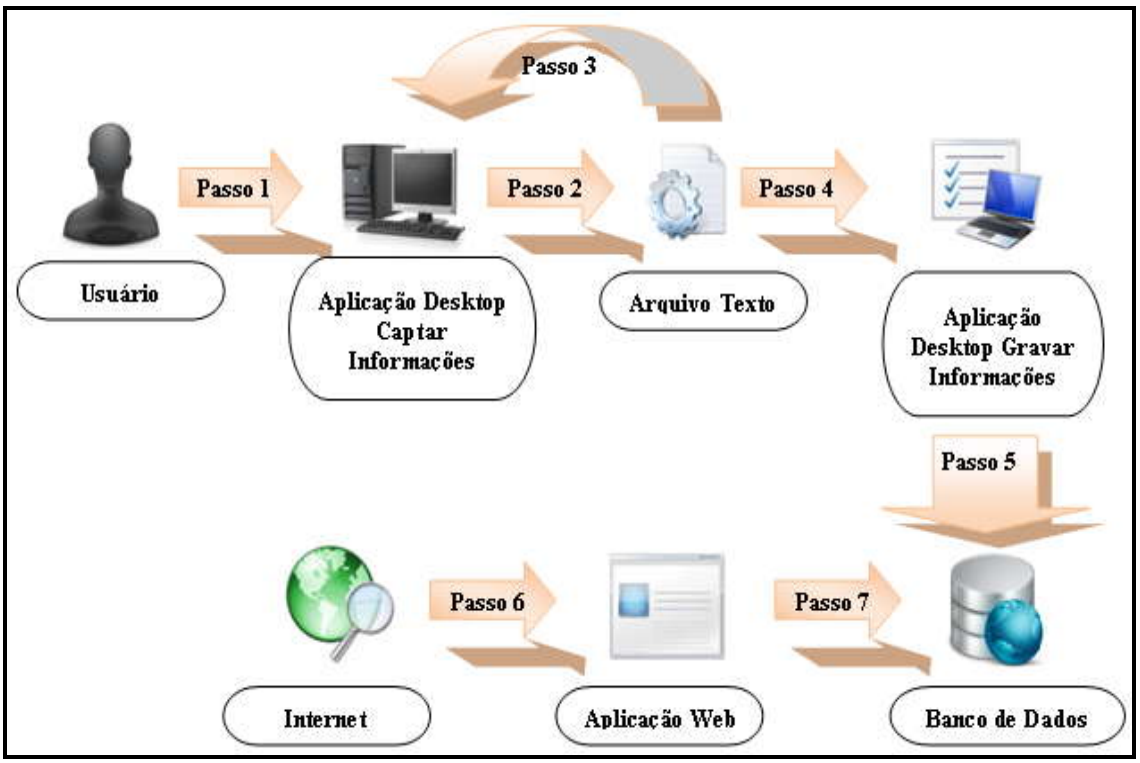

Figura 2 - Processo Automatizado

- Passo 1

Tendo em vista o montante de dados a serem transferidos pela rede, optou-se por programar uma aplicação *desktop* onde o fluxo de processamento ficará local e somente utilizará a rede para fazer a persistência das informações no bando de dados.

Inicialmente o funcionário se desloca até a estação de trabalho para capturar as informações ou acessa remotamente a mesma para efetuar o procedimento de captura.

- Passo 2

O funcionário executará uma ferramenta que realizará a coleta de todas as informações contida na estação de trabalho, logo após, uma pré-visualização de todas as informações coletadas serão apresentadas. Em seguida o funcionário irá salvar o conteúdo capturado em um arquivo no formato texto para ser processado posteriormente.

- Passo 3

Com o arquivo texto gerado, o operador poderá fazer o mesmo procedimento “Passo1 e Passo 2” em outras estações de trabalho caso necessite.

- Passo 4

Funcionário executará o aplicativo que irá fazer a leitura do conteúdo do arquivo texto salvo na etapa anterior, buscando automaticamente todas as informações referentes aos campos de cadastros definidos na aplicação.

Durante a leitura dos dados o aplicativo irá fazer a validação das informações, verificando se as mesmas já existem cadastradas no banco de dados. Caso existam, o aplicativo completa automaticamente os campos do cadastro com as referências dos dados existente no arquivo texto gerado no “Passo 2”.

- Passo 5

Caso o aplicativo encontre alguma informação de determinado campo que ainda não foi preenchido automaticamente, aparecerá uma sugestão de cadastro para o funcionário, e, se o mesmo optar por cadastrar, a informação será usada de forma automática. O funcionário não precisará digitar dados, somente ficará responsável por validar o resultado final da aplicação.

Após o funcionário concluir o pré-cadastro, o aplicativo irá validar todos os campos definidos como obrigatórios, se por algum motivo, algum desses não esteja devidamente preenchido ou validado, aparecerá uma janela contendo todas as

ocorrências registradas. Assim sendo o *software* não irá habilitar a opção para persistir as informações de todas as tabelas no banco de dados.

- Passo 6

Será disponibilizado na internet um *WebSite* contendo todas as informações persistidas com a aplicação *desktop.* Tal site tem por finalidade manutenção e a manipulação dos dados conforme o necessário.

Todos os recursos que terá na aplicação *desktop* estarão disponíveis na aplicação *web*, facilitando a atualização e visualização dos mesmos.

- Passo 7

A Aplicação *web* será produzida com uma interface rica para internet, que possibilitará a visualização da aplicação como se fosse uma ferramenta *desktop,* Tornando–se auto-intuitiva com suas janelas bem decoradas e robustez no desempenho e na manipulação dos dados.

- Passo 8.

Após o pré-cadastro das informações das tabelas os mesmos serão validados e verificados conforme a necessidade, antes de serem persistidos no banco de dados.

3.3 FACILIDADES QUE A FERRAMENTA DISPONIBILIZARÁ PARA REALIZAR UMA AUDITORIA

Com a execução das aplicações o processo de auditoria irá disponibilizar de recursos que ajudarão no seu decorrer. Pois as ferramentas disponibilizaram de relatórios referentes a qualquer alteração feita no banco de dados e a qualquer tabela que necessite. Portanto, as informações serão facilmente rastreadas conforme a necessidade do auditor. A Figura 3 mostra como serão mapeado esses relacionamentos.

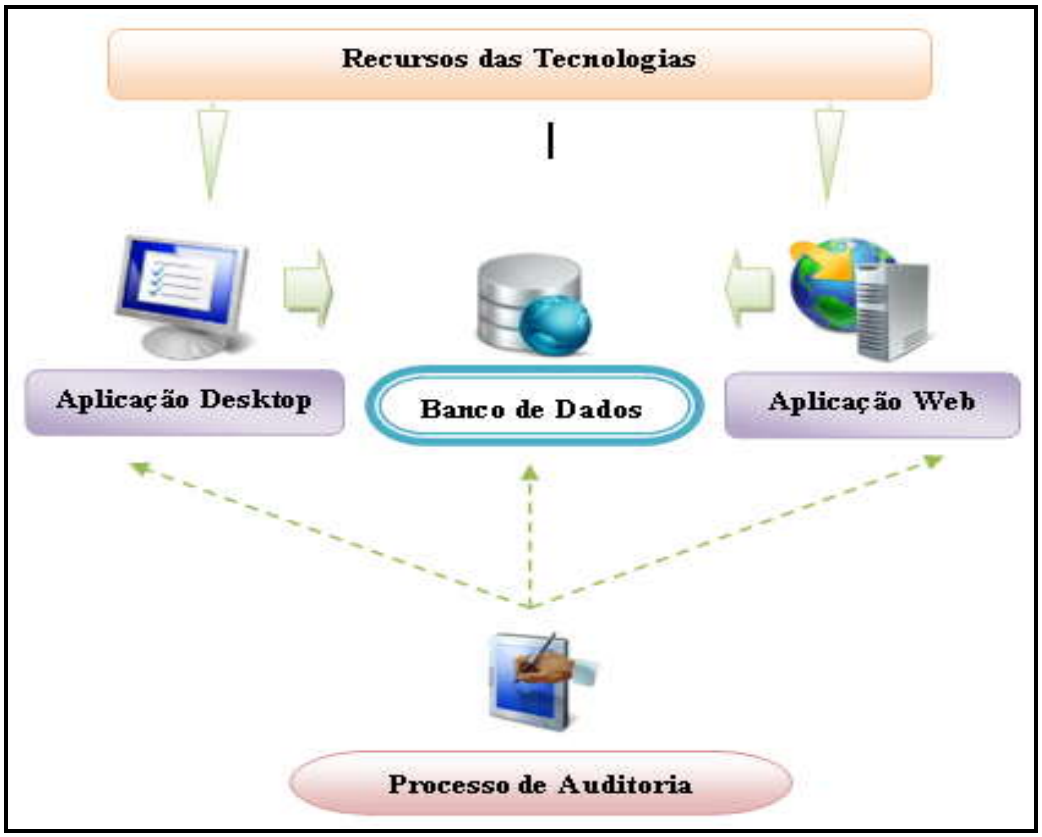

Figura 3 - Auditoria

Caso haja a necessidade do auditor efetuar uma análise das informações, o mesmo pode ser feito de maneira local ou remota, pois as referidas informações serão disponibilizadas no mesmo banco de dados, tanto na aplicação *desktop* quanto na aplicação *web*, não havendo assim problemas com inconsistências.

4. MODELAGEM

Para alcançar os objetivos propostos, este capítulo discute a modelagem da ferramenta que irá disponibilizar recursos para a auditoria.

A ferramenta disponibilizará dois aplicativos O primeiro para captar automaticamente as informações referentes aos dados das estações de trabalho e a outra para disponibilizar e manipular as informações conforme o necessário via *web* usando uma tecnologia (RIA). As aplicações devem proporcionar ao seu manipulador, um total conforto entre a navegação e a manipulação entre os dados.

Para a concepção dessa ferramenta, em primeiro instante foi feita uma modelagem do sistema, na busca de atender e descrever as necessidades requeridas. Foi elaborada uma análise de requisitos, em seguida a análise do diagrama de casos de uso. Após estas etapas foi possível fazer o levantamento das entidades para descrever o diagrama de classe, e, por fim, as interações serão denotadas por meio do diagrama de seqüência.

4.1 ANÁLISE DE REQUISITOS

Na concepção deste aplicativo foram levantados requisitos necessários para a ferramenta atender os objetivos. A seguir serão listados os requisitos dos aplicativos propostos.

4.1.1 Aplicativo *Desktop*

Inicialmente a aplicação deve dispor de *login* e senha para iniciá-la.

Deve permitir que o funcionário faça a captação dos dados das estações de trabalho de forma automática. Sendo que as informações serão conflitadas com o banco de dados para ver se as mesmas já não existem.

As informações referentes à empresa devem ser relacionadas com o do banco de dados existente no local *(Oracle*) através de uma (*view*).

A aplicação também deve permitir que o funcionário faça a captação das informações separadamente: Exemplo “Monitor, Computador”.

4.1.2 Aplicativo *Web*

Inicialmente a aplicação deve dispor de login e senha para iniciá-la. Deverá conter todas as informações captadas pela aplicação *desktop.* O aplicativo deve disponibilizar informações sobre todos os dados referentes à estação de trabalho.

O operador deve ter como interagir com essas informações de maneiras diferenciadas, com comandos de pesquisa e auxílio à navegação das informações.

Deve ser representadas informações de forma que facilitem a visualização e manipulação das mesmas. Para não tornar a aplicação cansativa para o funcionário, a mesma terá que conter um papel de parede da forma que a cada atualização de sessão o mesmo seja trocado. A aplicação deve dispor também de relatórios cujos mesmos serão utilizados para auditoria, contendo informações de acesso, alteração, exclusão, data/hora e funcionário que efetuou tal modificação.

Todos os relatórios deveram ter opções para exportar para um arquivo texto.

4.2 CASOS DE USO

O Diagrama de caso de uso é utilizado para demonstrar as funcionalidades de um sistema após a análise de requisitos. São possíveis por meio deste tipo de diagrama, modelar as funções e serviços que o sistema deve prever, entendendo assim, o uso que o sistema terá e para que tipo de aplicações ele seja empregado (ALISTAIR COCKBURN, 2004, p36). A Figura 4 mostra uma visão geral do caso de uso adotado.

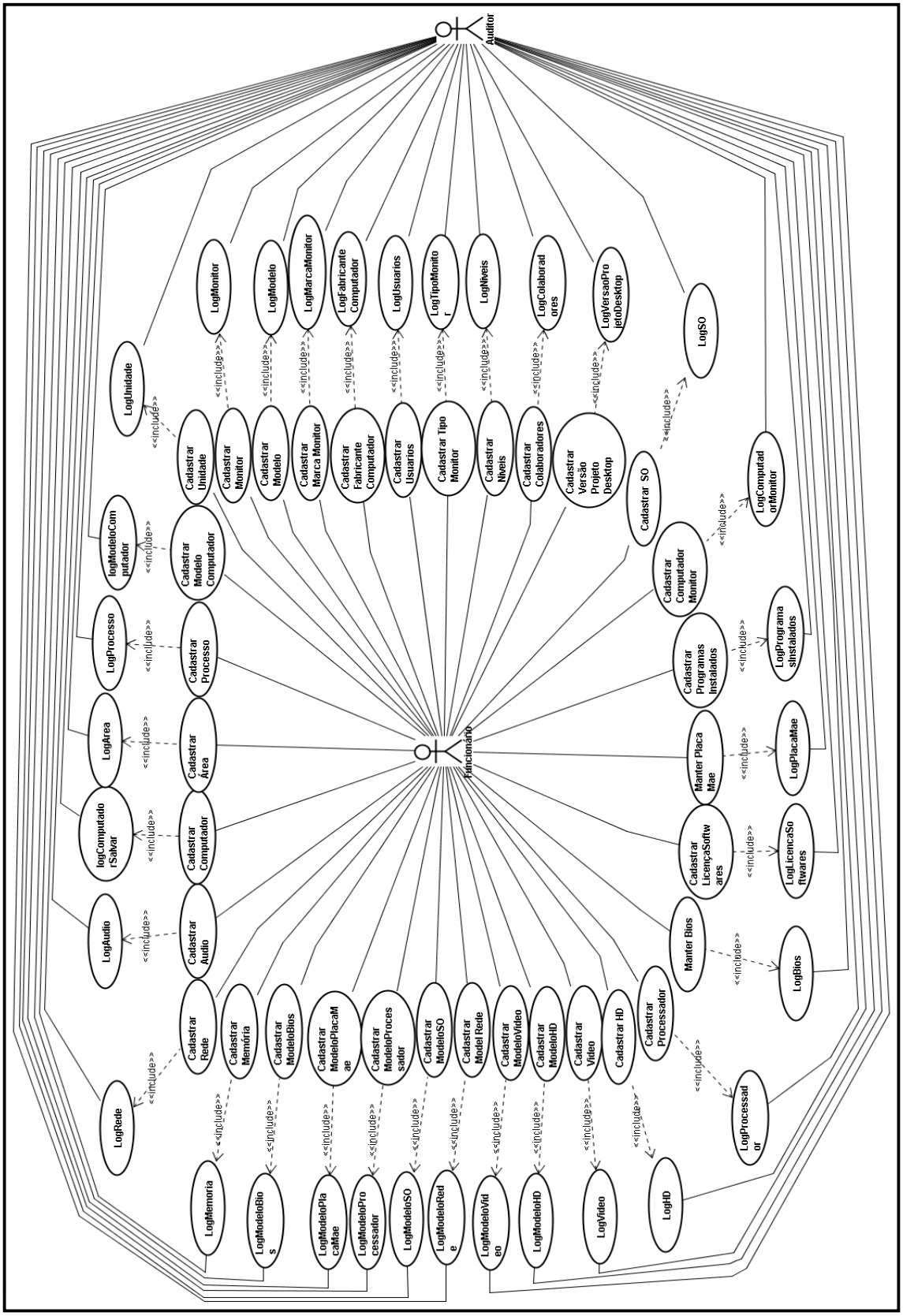

Figura 4 - Diagrama de Caso de Uso – Visão Geral

A Figura 5 mostra o caso de uso que trata das interações do funcionário perante o sistema, os mesmos podem ser acionados por meio de uma ação do funcionário, ManterBios, CadastrarModeloBios, CadastrarModeloPlacaMae, CadastrarPlacaMae, CadastrarProcessador, CadastrarModeloProcessador, CadastrarAudio, CadastrarSO, CadastrarModeloSO, CadastrarRede, CadastrarModeloRede, CadastrarVideo, CadastrarModeloVideo, CadastrarMemoria, CadastrarHD, CadastrarModeloHD, CadastrarArea, CadastrarUnidade, CadastrarProcesso, CadastrarUsuarios, CadastrarNivel, ComputadorModelo, CadastrarFabricanteComputador, CadastrarLicencaSoftware, CadastrarProgramasInstalados, CadastrarMonitor, CadastrarModelo, CadastrarMarca, CadastrarTipoModelo, CadastrarComputadorMonitor, CadastrarVersaoProjetoDesktop CadastrarComputador.,

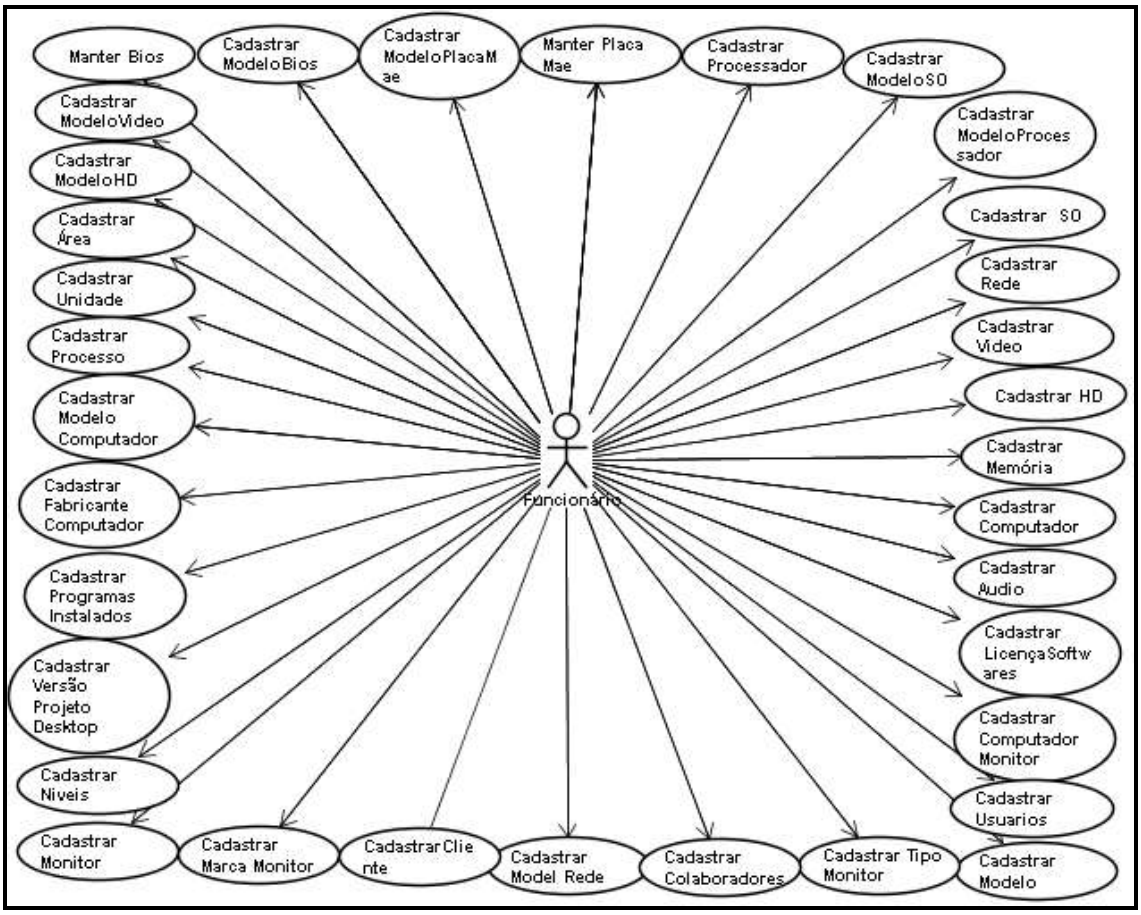

Figura 5 – Diagrama de Caso de Uso – Funcionário

Figura 6 mostra o caso de uso onde o auditor analisa os casos referentes aos *logs* feitos pelas interações do funcionário.

O Auditor terá acesso aos seguintes casos de uso: LogBios, LogModeloBios, LogModeloPlacaMae, LogPlacaMae, LogProcessador, LogModeloProcessador, LogAudio, LogSO, LogModeloSO, LogRede, LogModeloRede, LogVideo, LogModeloVideo, LogMemoria, LogHD, LogModeloHD, LogArea, LogUnidade, LogProcesso, LogUsuarios, LogNivel, LogModelo, LogFabricanteComputador, LogLicencaSoftware, LogProgramasInstalados, LogMonitor, LogModelo, LogMarca, LogTipoModelo, LogComputadorMonitor, LogVersaoProjetoDesktop, LogComputador.

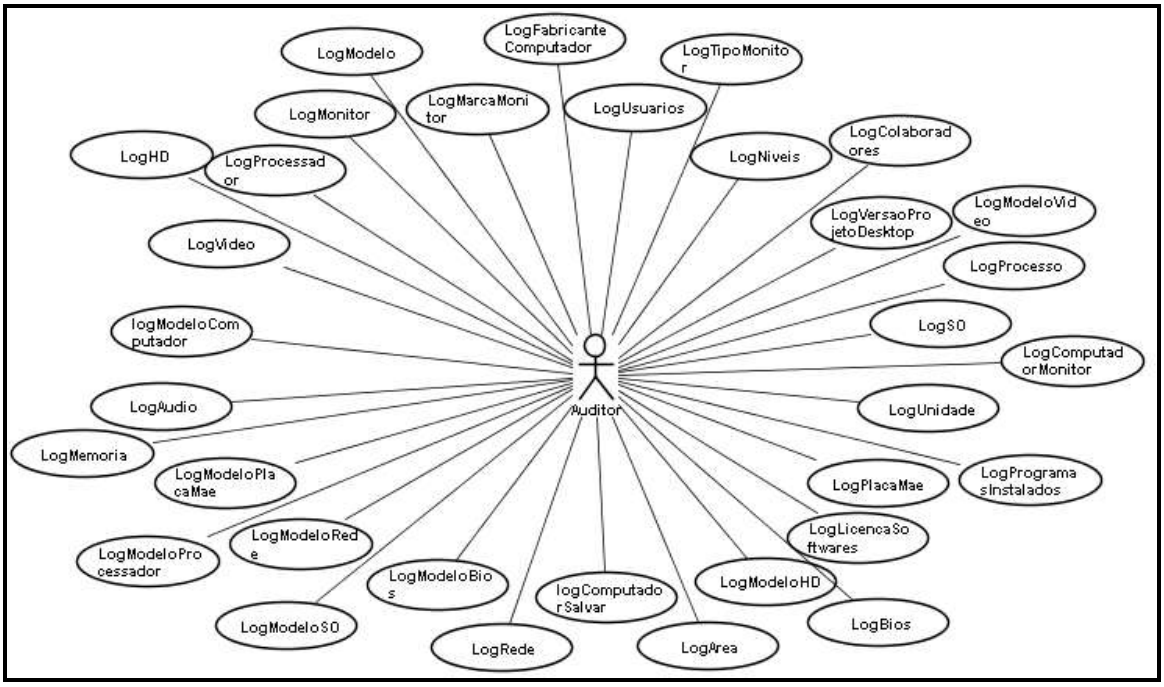

Figura 6 – Diagrama de Caso de Uso - Auditor

4.3 DIAGRAMA DE CLASSES

Após a análise dos requisitos necessários para a o desenvolvimento da aplicação proposta neste trabalho e a modelagem dos casos de uso desta aplicação é possível descrever o formato das classes e objetos que se fazem necessários para o desenvolvimento dos aplicativos.

Em uma aplicação orientada a objetos, tais itens são pontos principais para a construção deste tipo de aplicação, sendo que um sistema orientado a objetos é proposto por classes e objetos que interagem entre si para a execução dos serviços necessários ao funcionamento do aplicativo (STADZIZZ, 2002, p.15). Desta forma é apresentada nos seguintes diagramas, a figura 7 mostra uma visão geral do diagrama de classe adotado.

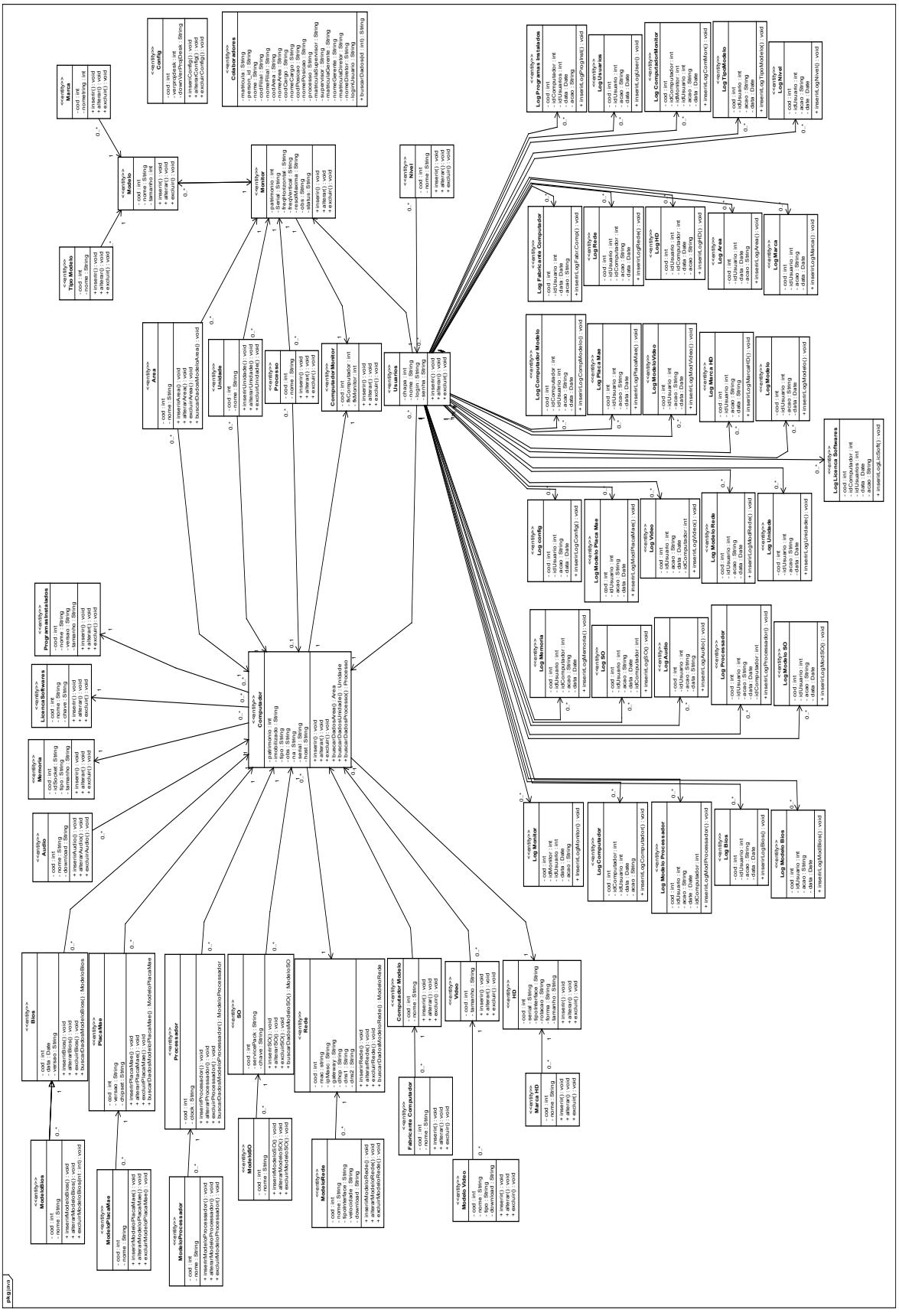

Figura 7 - Diagrama de Classe

A Figura 8 mostra somente o diagrama de classe referente ao Computador

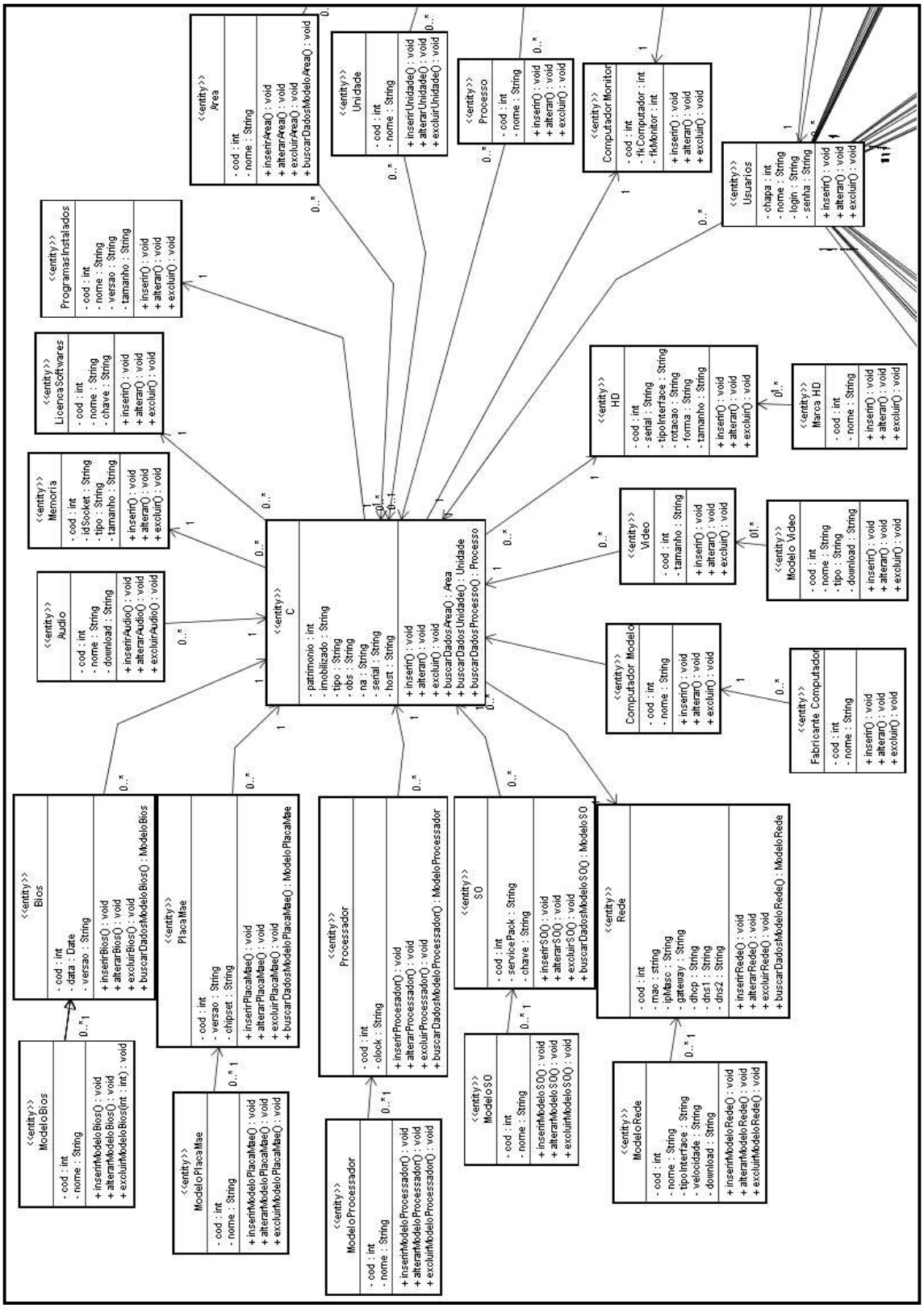

Figura 8 - Diagrama de Classe – Computador

Figura 9 mostra somente o diagrama de classe referente ao Monitor.

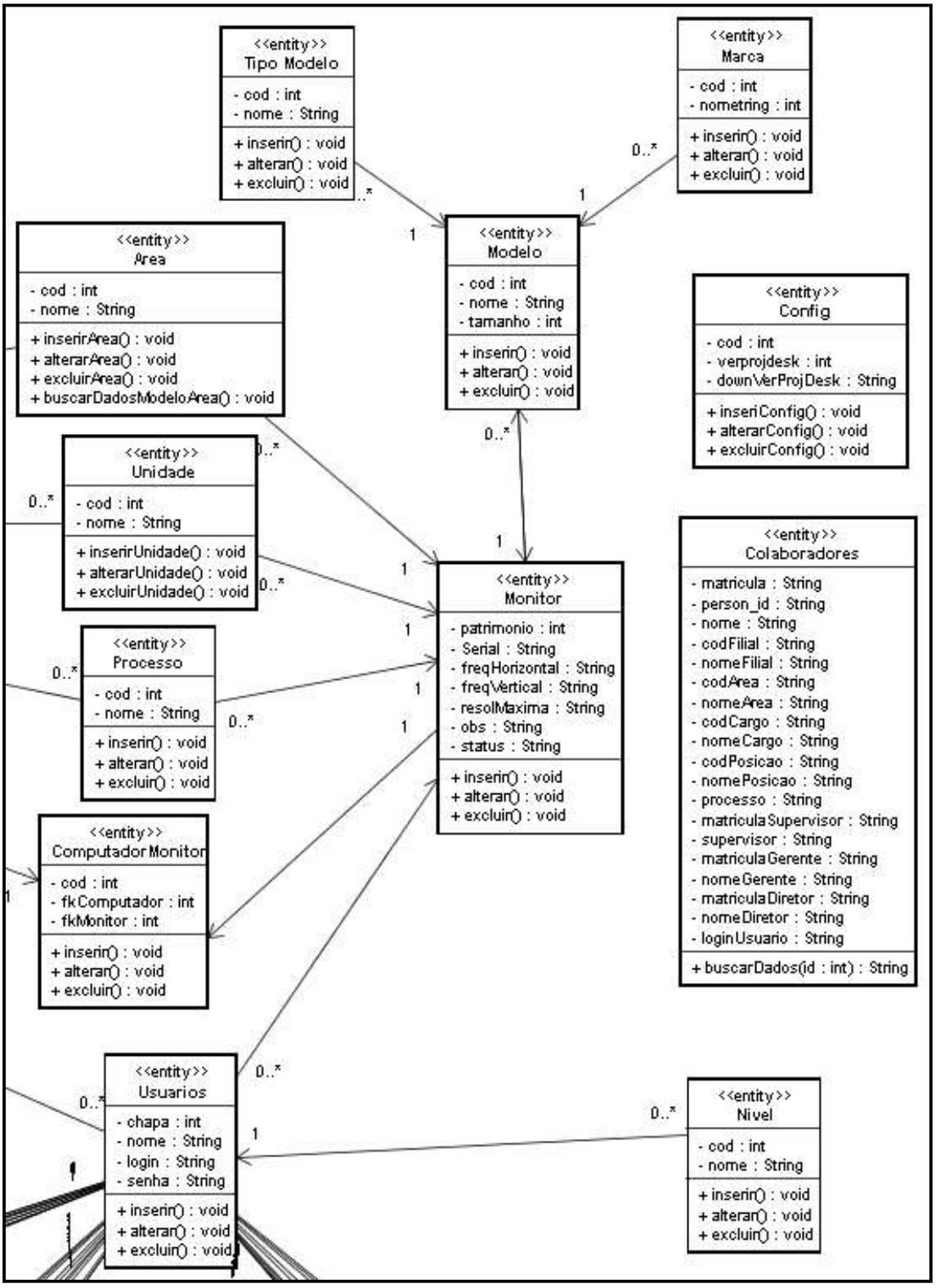

Figura 9 - Diagrama de Classe - Monitor

A Figura 10 mostra somente o diagrama de classe referente ao "*Logs*" das ações feita no sistema.

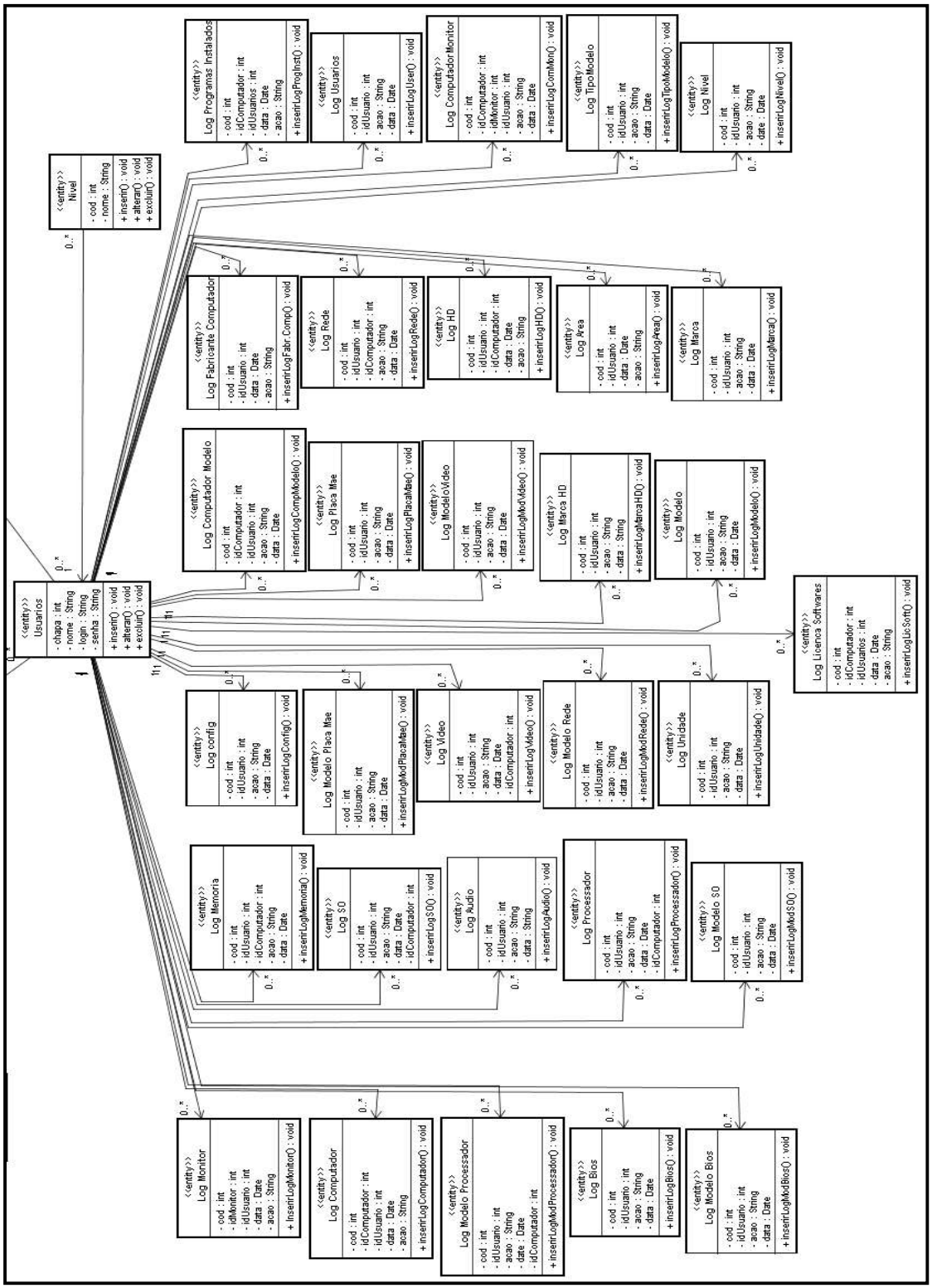

Figura 10 - Diagrama de Classe - Logs

4.4 DIAGRAMA DE SEQUÊNCIA

Um diagrama de seqüência é formulado por um conjunto de objetos, onde são demonstradas as comunicações necessárias entre os objetos, para que possa ser executado aos processos do sistema. Por meio de uma linha de tempo estes diagramas descrevem a seqüência de comunicações entre estes objetos (STADZIIZ, 2002, p. 29).

A partir desta definição foram elaborados os seguintes diagramas de seqüência para os casos anteriormente definidos. Caso de uso para cadastrar os dados referentes à tabela de Sistemas Operacionais (SO) é ilustrado na Figura 11.

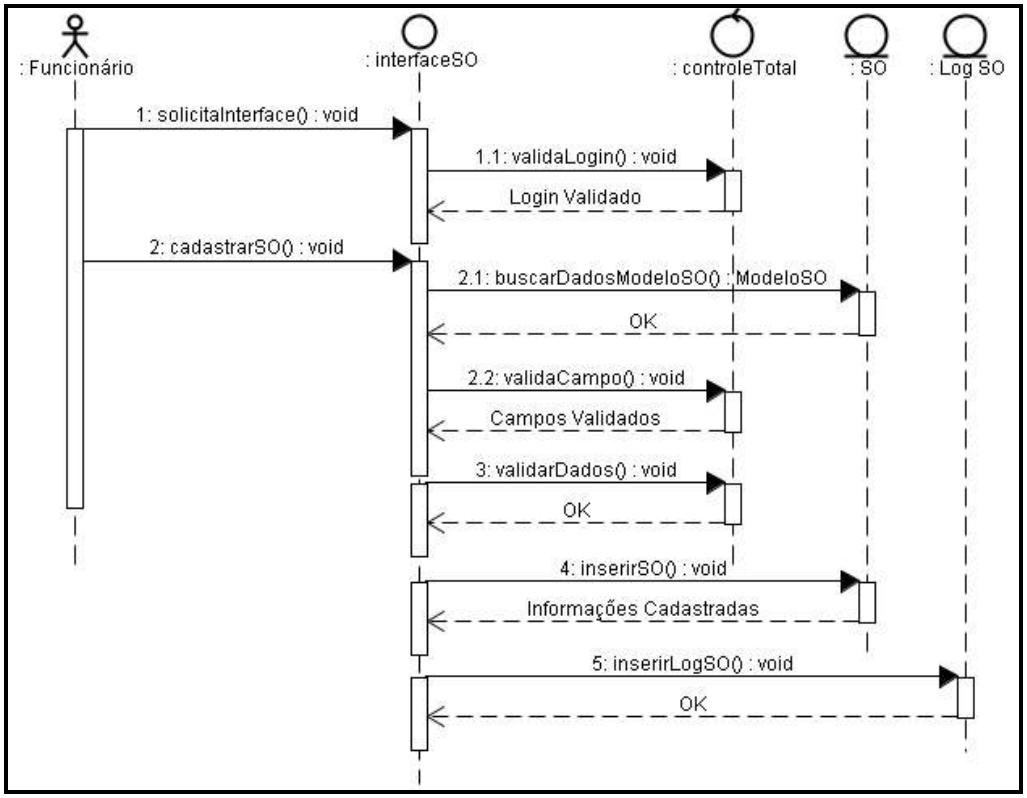

Figura 11 - Diagrama de Seqüência – SO - Cadastro

Após a etapa de cadastro será demonstrado o caso de uso para editar os dados referentes à tabela de Sistemas Operacionais (SO), como segue a Figura 12.

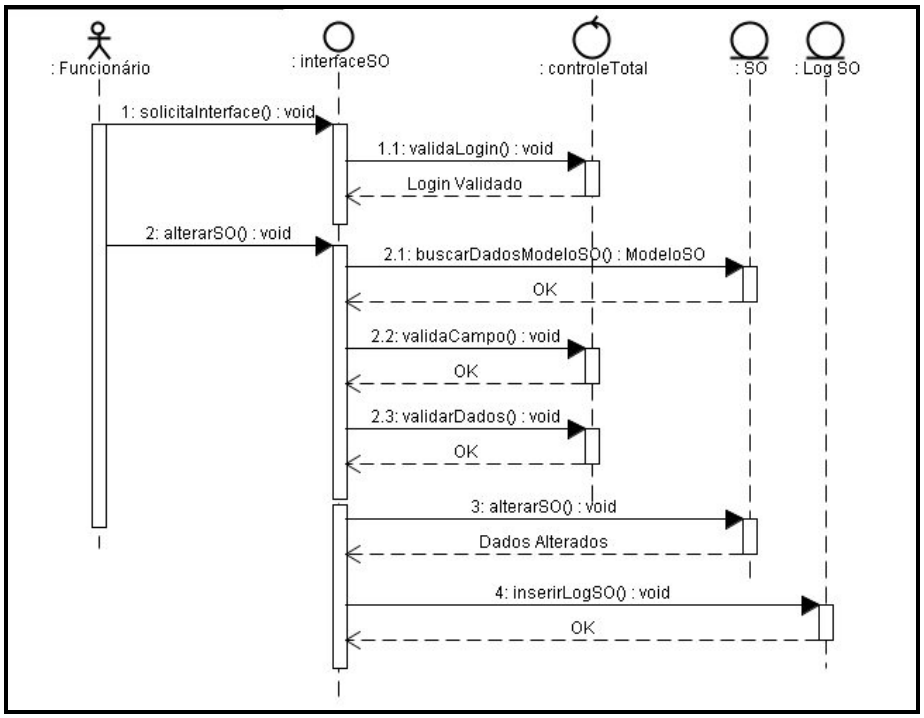

Figura 12 - Diagrama de Seqüência – SO - Editar

Em seqüência será demonstrado o Caso de Uso para excluir os dados referentes à tabela de Sistemas Operacionais (SO), como demonstra a Figura 13.

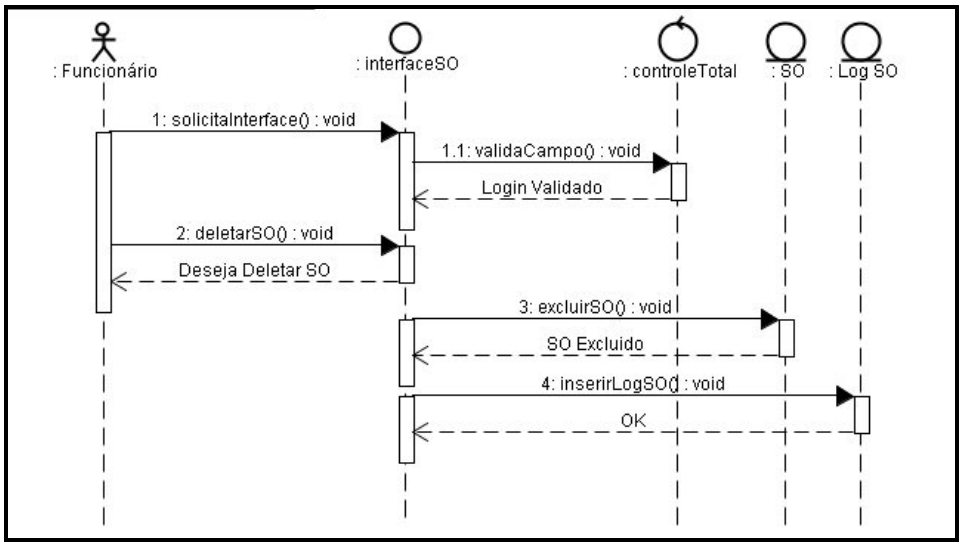

Figura 13 - Diagrama de Seqüência – SO - Excluir

Seguindo esta linha de raciocínio são montados os demais diagramas de seqüência para a conclusão da ferramenta.

5. TECNOLOGIAS

Inicialmente será desenvolvida a aplicação que realizará a Coleta Automática das Informações das estações de trabalho, sendo implementado em Java, usando o banco de dados *PostgreSQL*, com a comunicação entre as duas tecnologias, a ser realizada com o *framework Hibernate.*

Posteriormente será produzido um *website* que gerenciará toda a coleta realizada pela primeira aplicação, utilizando de aplicações ricas para *Internet* (RIA) no contexto atual a tecnologia *Adobe Flex.* Para realizar a comunicação entre a tecnologia *Adobe Flex* e a tecnologia Java o mesmo será feito através do *framework BlazeDS*.

5.1 TECNOLOGIA JAVA

A linguagem Java originou-se da Linguagem C por ser uma linguagem totalmente orientada a objetos, além de possuir portabilidade e ser independente do *hardware.*

Com a explosão da *Internet* em 1993, os criadores da linguagem Java viram um grande potencial em utilizar Java para criar páginas *Web* com o chamado conteúdo dinâmico. Em 1995 a *Sun* lançou o Java formalmente, despertando interesse na comunidade comercial, almejando melhorar o desempenho dos servidores *World Wide Web* (WWW), disponibilizando aplicativos e dispositivos para o usuário final. Sistemas Java se dividem em partes: um ambiente, a linguagem, a interface de programas aplicativos (*Applications Programming Interface* - API) Java e várias bibliotecas de classes (DEITEL, 2001).

5.2 HIBERNATE

O *Hibernate* é um *framework* utilizado no mapeamento objeto-relacional escrito na linguagem Java, mas também é disponível em *Net* como o nome *NHibernate.*

Este programa facilita o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação, mediante o uso de arquivos XML (*Extensible Markup Language* – Linguagem de Marcação Extensível) para estabelecer esta relação.

Hibernate é um software livre de código aberto distribuído com a licença LGPL (BAUER, 2005).

5.3 BANCO DE DADOS - *POSTGRESQL*

O *PostgreSQL* é um SGBD (Sistema Gerenciador de Banco de Dados) Objeto-Relacional de código aberto, com mais de 15 anos de desenvolvimento. É extremamente robusto e confiável, além de ser extremamente flexível e rico em recursos. Ele é considerado objeto-relacional por produzir, além das características de um SGBD relacional, algumas características de orientação a objetos, como herança e tipos personalizados.

5.4 TECNOLOGIA *ADOBE FLEX*

O *Adobe Flex* (antes chamado de *Macromedia Flex* e depois rebatizado como *Adobe Flex* pela *Adobe*) é o nome de uma tecnologia lançada em Março de 2004 pela *Macromedia*, que suporta o desenvolvimento de aplicações ricas para a Internet, baseadas na plataforma do *Macromedia Flash.* A versão inicial possuía um SDK (*Software Development Kit)* e uma integração com o J2EE (JAVA *Enterprise Edition*) também conhecido como *Flex Data Services*.

O *Flex* SDK é um compilador em modo caracter. É possível desenvolver aplicações *Flex* gratuitamente utilizando este compilador. O programador utiliza algum editor de sua preferência (Eclipse, Bloco de Notas, *Dreamweaver*, etc), salva o arquivo MXML e através do SDK compila este arquivo para binário swf (*Small Web Format*).

Flex Data Services oferece um conjunto de avançados recursos de gerenciamento de dados no lado do servidor que permitem aos desenvolvedores fornecer rapidamente aplicativos *Flex* que fazem intensos uso de dados.

5.5 TECNOLOGIA *BLAZEDS*

BlazeDS é um *framework* baseado em Java e *Web Remoting* mensagens, tecnologia essa que permite conectar-se a distribuição de dados *Back-End* e carregar os dados em tempo real para o ambiente *Adobe Flex e Adobe AIR rich Internet applications* (RIA).

5.6 SOFTWARE EVEREST

Everest é um sistema de diagnóstico de *hardware e softwares* que tem como objetivo a busca e captura das informações e é mantido pela empresa Lavalys.

6. IMPLEMENTAÇÃO DA APLICAÇÃO

Este capítulo tem como objetivo demonstrar parte da configuração do ambiente de desenvolvimento e também exemplificar o uso das aplicações (*Java* e *Adobe Flex*) usando o *framework BlazeDS*. Em um primeiro momento é dada a atenção ao desenvolvimento da aplicação *desktop* juntamente com toda a sua configuração. Seguindo-se com a incorporação do ambiente de desenvolvimento *Adobe Flex* utilizando o *framework BlazeDS.*

6.1 DESENVOLVIMENTO DA APLICAÇÃO *DESKTOP*

Visando agilidade da captura das informações dos *hardwares e softwares* instalados nos computadores, optou-se por estar utilizando um *software* gratuito (*Everest Portable*) para fazer a coleta das informações.

6.1.1 Everest Portable

Software utilizado com o intuito de agilizar o processo na captura das informações. Sendo que para funcionar, não precisa de qualquer instalação no computador, com um duplo clique do mouse em cima do aplicativo o mesmo já se executa e começa a fazer a tarefa desejada no nosso caso à captura das informações. Após a execução do aplicativo o mesmo salva um arquivo no formato texto para fazer a integração com a aplicação *desktop* como mostra a Figura 14.

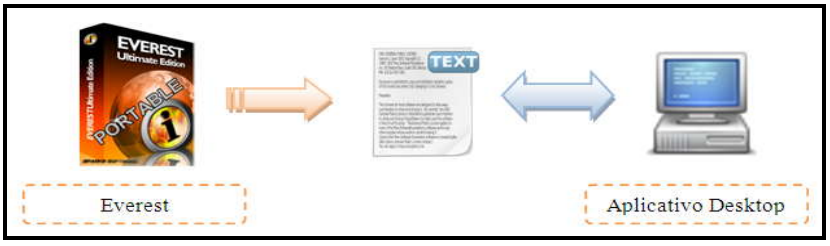

Figura 14 - Integração entre as Aplicações

6.1.2 Importação do arquivo texto

Após o arquivo texto ser gerado, a aplicação *desktop* começa a fazer todo o processo de leitura, verificando a integridade das informações contidas no mesmo, sendo que para cada "dado" adquirido com a leitura, é feita uma verificação para ver se o mesmo já não está cadastrado no banco de dados. Estando, ele é selecionado nos campos do formulário automaticamente, caso contrário, pode ser cadastrado utilizando as informações contidas nos campos de "dicas" Figura 15.

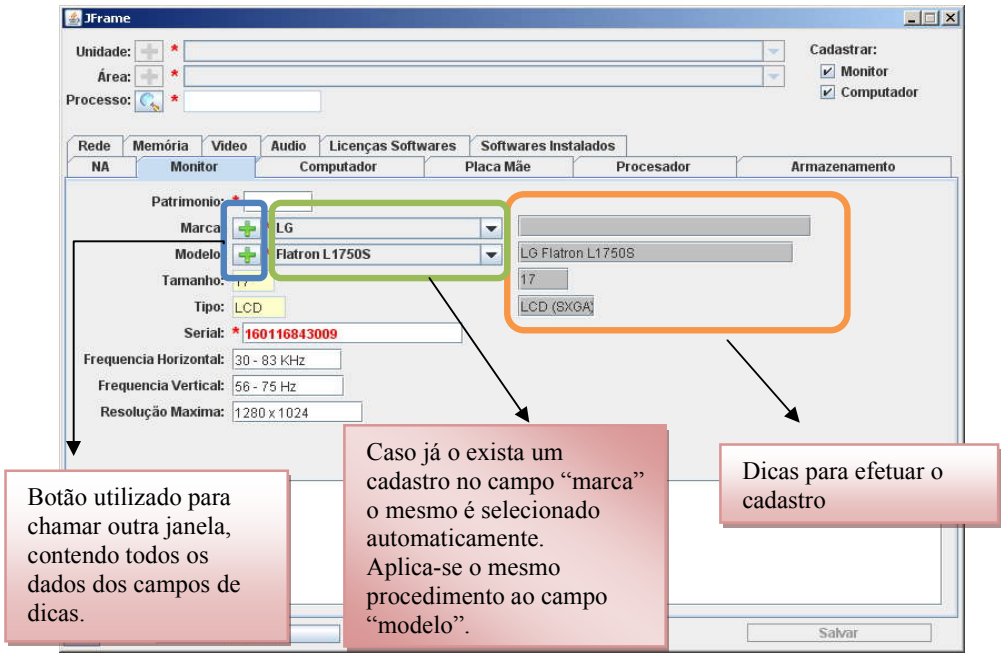

Figura 15 – Importação das informações

Supondo-se que o cadastro do campo "modelo" não tenha sido localizado automaticamente, então terá que ser feito o seu cadastro. Clicando no botão  irá aparecer à janela responsável pela manutenção das informações referente ao cadastro Figura 16.

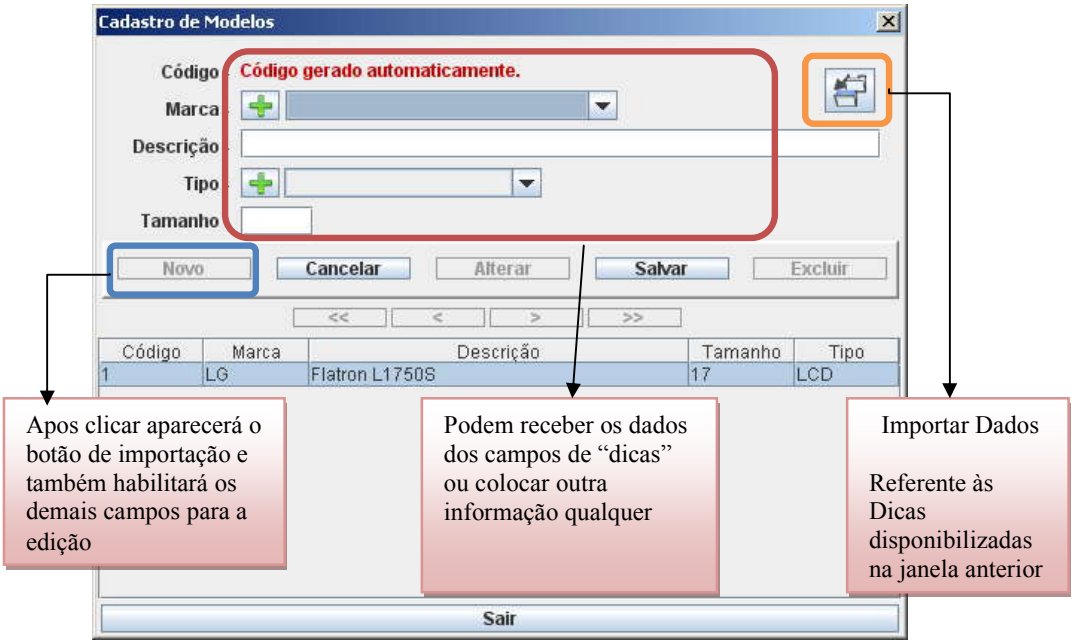

Figura 16 – Processo automatizado do cadastro

Feito o cadastro, todos os arquivos textos que contiverem tais informações serão automaticamente selecionados quando estiver sendo efetuada a leitura.

6.1.3 Validação de Dados

Após realizar a importação das informações do arquivo texto, alguns campos definidos como obrigatórios podem não ser preenchidos automaticamente. Para fazer a validação desses campos, existe um botão "VALIDAR", que tem como objetivo varrer todas as inconsistências e gerar um *logs* dos dados, discriminando todas as ocorrências encontradas como mostra a Figura 17.

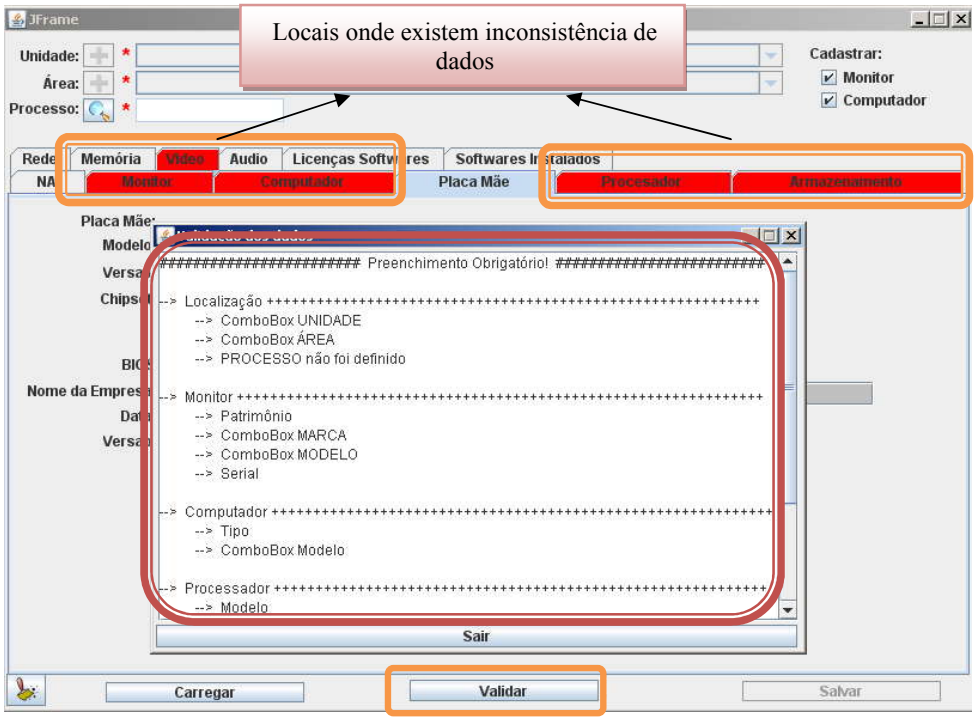

Figura 17 - Validação

Após efetuar todas as validações, o processo de gravação é concluído e, por fim, gerado um *log* de todos os eventos realizados durante a persistência das informações. Como segue a Figura 18.

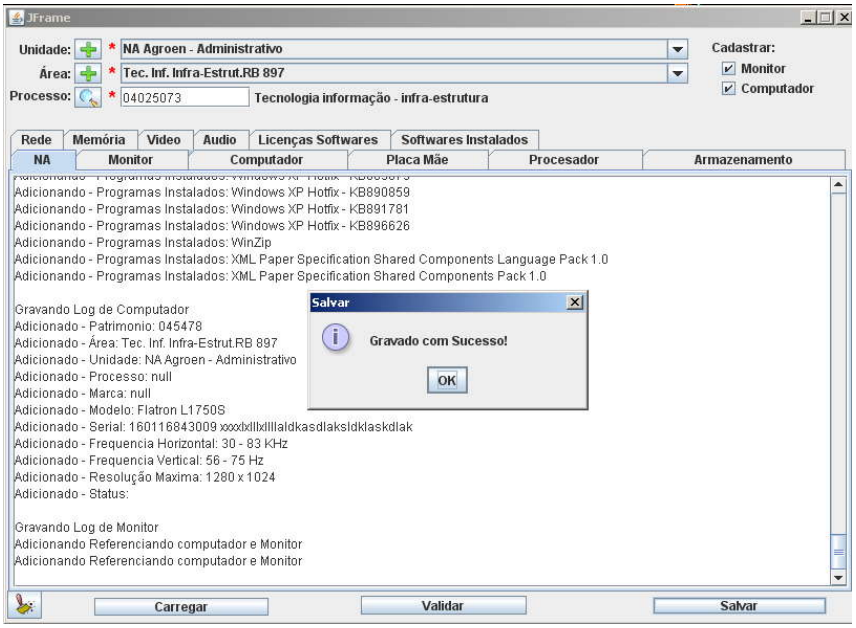

Figura 18 - Conclusão do Cadastro

6.2 CONFIGURAÇÃO E IMPLEMENTAÇÃO DO AMBIENTE DE DESENVOLVIMENTO *ADOBE FLEX*

Este tópico demonstra como realizar a integração entre as tecnologias *adobe flex* e a tecnologia *Java,* utilizando o *framework BlazeDS* para fazer a integração.

6.2.1 Configurações das Bibliotecas

Para realizar a integração entre as tecnologias as seguintes bibliotecas foram utilizadas conforme mostrada a Figura 19.

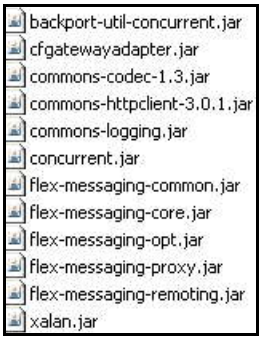

Figura 19 - Bibliotecas

6.2.2 Mapeamento das Classes de Persistência DAO

Para fazer a integração entre as tecnologias é obrigatório a definição do seguinte *servlet* dentro do arquivo de configuração *web.xml* como demonstra a Figura 20.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <!-- MessageBroker Servlet -->
    <servlet>
        <servlet-name>MessageBrokerServlet</servlet-name>
        <display-name>MessageBrokerServlet</display-name>
        <servlet-class> flex.messaging.MessageBrokerServlet </servlet-class>
        <init-param>
            <param-name>services.configuration.file</param-name>
            <param-value>/WEB-INF/flex/services-config.xml</param-value>
        </init-param>
        <init-param>
            <param-name>flex.write.path</param-name>
            <param-value>/WEB-INF/flex</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>MessageBrokerServlet</servlet-name>
        <url-pattern>/messagebroker/*</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

Figura 20 - Arquivo WEB.xml

Após o processo de configuração do *servlet*, será feito o mapeamento das classes de persistência utilizada na aplicação *desktop*, para isso exemplificarei a configuração da classe de persistência de “áreas” utilizada para a manipulação do banco de dados conforme Figura 21.

```
package banco;

import java.util.Date;

public class AreaPST {

    public List<Area> listar(){

    public List<Area> listar(String nomeColunaBanco, String ordenacao){

    public boolean excluir(Area a, Usuarios user){

    public boolean excluir2( Area area ) {

    public boolean atualizar(Area a, Usuarios user){

    public boolean atualizar2(Area area) {

    public Area buscarCod(int codigo) {

    public int seqCod(){
}
```

Figura 21 - Classe de persistência referente a Áreas

Será feito a configuração desses mapeamentos no arquivo *“remonting-config.xml”* que se localiza no diretório demonstrado conforme a Figura 22.

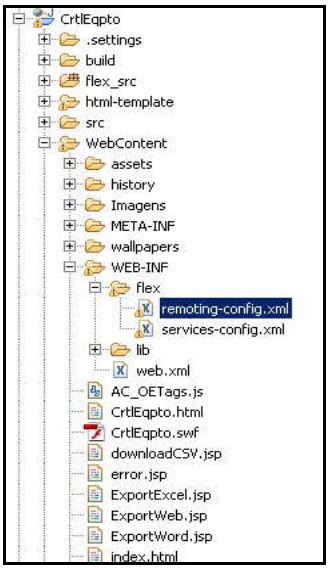

Figura 22 – Local de configuração das classes de persistência

Dentro do arquivo *"remoting-config.xml",* será necessário criar as *tags ("destination", "properties" e "source")* para definir os mapeamentos entre os objetos.

Sendo que a *"id" da tag "destination"*, será utilizado nas classes xmlx das aplicações *adobe flex* como caminho para chegar à classe de persistência desejada. E a *tag "properties*" define o caminho propriamente dito até chegar à classe, como demonstra a Figura 23.

```
<?xml version="1.0" encoding="UTF-8"?>
<service id="remoting-service" class="flex.messaging.services.RemotingService">
    <adapters>
        <adapter-definition id="java-object"
            class="flex.messaging.services.remoting.adapters.JavaAdapter"
            default="true" />
    </adapters>
    <default-channels>
        <channel ref="my-amf" />
    </default-channels>

    <destination id="ObjAreaPST">
    <properties>
        <source>banco.AreaPST</source>
    </properties>
    </destination>

    <destination id="ObjAudioPST">
        <properties>
            <source>banco.AudioPST</source>
        </properties>
    </destination>
</service>
```

Figura 23 - Mapeamento das Classes

Seguindo o raciocínio será demonstrado o design de manipulação dos dados referente a tabela de Áreas como mostra a Figura 24, *design* esse desenvolvido já utilizando a tecnologia *adobe flex.*

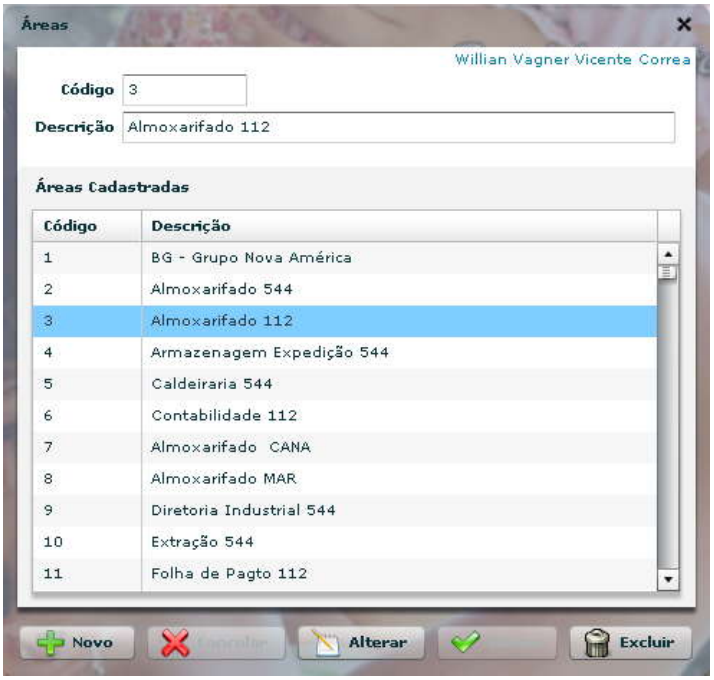

Figura 24 - Cadastro de Áreas

Importante atenção é dada nos cadastro nos quais a manipulação dos dados é feita através de *"RemoteObject"*, ou seja objeto remoto, sendo que a configuração do *''destination*" do mesmo aponta para o arquivo de configuração *"remoting-config.xml".*

Após esse mapeamento é possivel efetuar a chamada dos métodos da classe de persistência para ser processados. Como segue a Figura 25

```
<mx:RemoteObject id="areaService" showBusyCursor="true" fault="{onFault(event)}" destination="ObjAreaPST">
    <mx:method name="atualizar"    result="{onResultAtualizar(event)}"    fault="{onFault(event)}"/>
    <mx:method name="excluir"      result="{onResultExcluir(event)}"      fault="{onFault(event)}"/>
    <mx:method name="listar"       result="{onResultListar(event)}"       fault="{onFault(event)}"/>
    <mx:method name="seqCod"       result="{onResultSeqCod(event)}"       fault="{onFault(event)}"/>
</mx:RemoteObject>
```

Figura 25 - Chamada de método do ambiente *java.*

6.2.3 Configuração das Classes no *Adobe Flex*

A figura 26 mostra a classe utilizada para manipular os objetos na tecnologia *Java*.

```
1  package objetos;
2
3  import javax.persistence.Column;
10
11 @Entity
12 @Table(name="Eqpto_Una_Area")
13 public class Area {
14
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     @Column(name="COD")                    private int cod;
18     @Column(name= "NOME",length=50)        private String nome;
19
20     public Area(){
23
24     public Area(int cod) {
28
29     public Area(int cod, String nome) {
34
35     public int getCod() {
38
39     public void setCod(int cod) {
42
43     public String getNome() {
46
47     public void setNome(String nome) {
50 }
```

Figura 26 - Classe de Área utilizada pela tecnologia Java

Importante atenção é dada na criação das classes desenvolvidas em *action script* que manipulem os dados referentes às classes feitas com a tecnologia *Java,* o caminho para a classe deve ser definido da seguinte maneira como mostra a figura 27.

```
1  package objetos{
2      [RemoteClass(alias="objetos.Area")]
3
4      [Bindable]
5      public class Area{
6          public var cod:Number;
7          public var nome:String;
8      }
9  }
```

Figura 27 - Classe de Área utilizada pelo *Adobe Flex*

6.2.4 Manutenção dos Dados

Após a coleta das informações utilizando à aplicação *desktop*, é possível visualizar com aplicação *web* disponibilizar todo o conteúdo coletado e o ajustar conforme o *design* definido.

Visando a facilidade de manutenção dos mesmos, foi desenvolvido um *layout* simples e funcional e bem intuitivo para a compreensão das informações, todos os recursos de cadastros existentes na aplicação *desktop* também estão disponíveis na aplicação *web* como demonstra a figura 28.

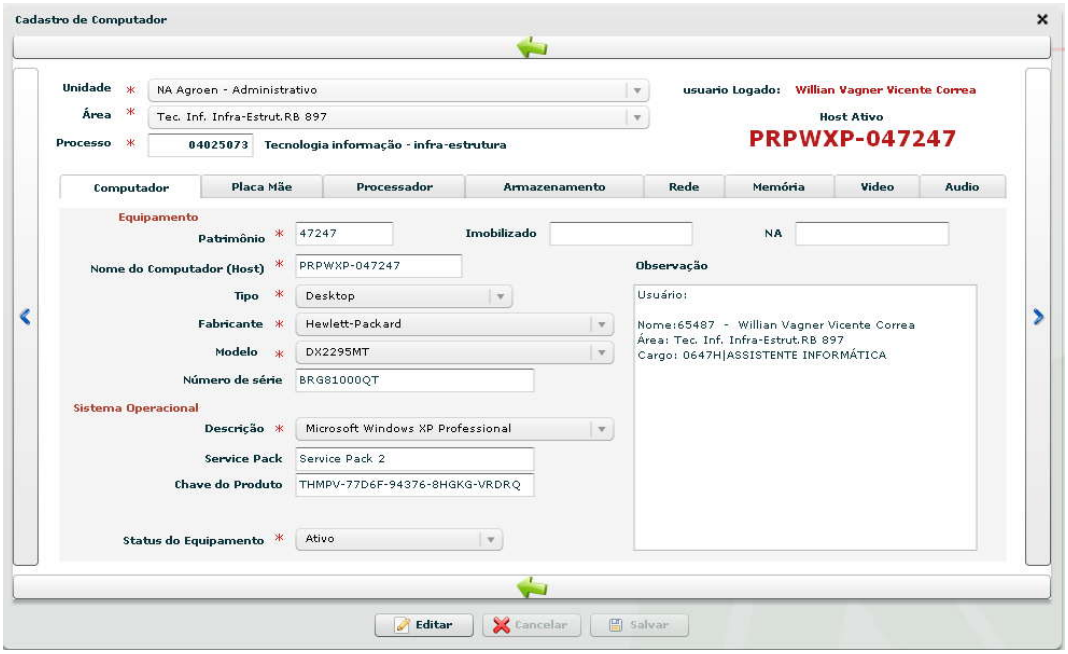

Figura 28 - Cadastro de Computadores

Foi desenvolvido também teclas de atalho que ajudaram o usuário no seu dia a dia como, por exemplo, fechar as janelas da aplicação utilizando a tecla ESC do teclado.

Todos os formulários possuem validações e verificações de campos obrigatórios.

Após o clique sobre os botões de busca é feita uma chamada dos métodos referentes aos objetos DAO. E para o preenchimento desses dados na *interface* foi utilizada a *metadata tag [Bindable]* na plataforma *flex,* que consiste em enviar uma instrução ao compilador que gera o código necessário para sua propriedade ser a fonte de dados para outras propriedades (ADOBE, 2007), como segue a figura 29.

```
[Bindable]  private var computador:Computador = new Computador();
[Bindable]  private var user:Usuarios;
```

Figura 29 - Declaração de Variáveis [Bindable]

Um atributo referenciado com esta instrução pode tornar-se uma fonte de dados a outros atributos e no momento que ele for atualizado, automaticamente, os atributos que o tenham utilizado como fonte de dados, recebem a atualização.

6.2.5 Filtros

Visando a agilidade na geração de relatórios, foram produzidos filtros que irão facilitar a visualização e integridade dos dados.

Essas janelas são totalmente redimensionáveis, podendo ocultar partes que não serão utilizadas naquele momento e assim ganhando uma área de visualização maior.

Os filtros foram definidos da seguinte forma: no primeiro campo do tipo *combobox* fica a coluna referente aos dados do cadastro selecionado; em seguida vem o tipo de filtro que pode variar de (Começa com, Contém, Termina com, É igual a, Não contém); após isso o filtro será feito automaticamente no campo desejado como apresenta a figura 30.

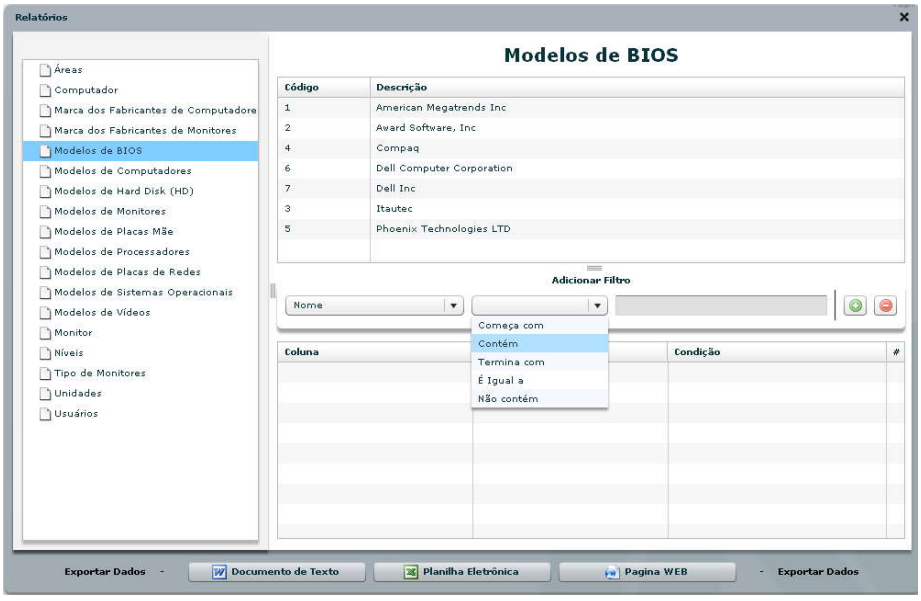

Figura 30 - Tela de Filtros

Após o registro do filtro o mesmo pode ser removido ou então caso necessário adicionar um outro como a figura 31.

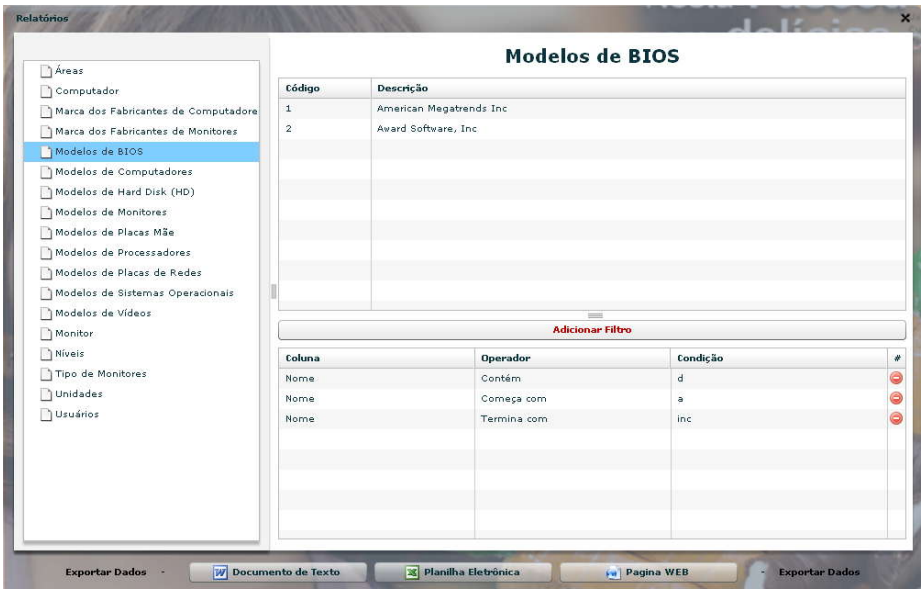

Figura 31 - Busca Personalizada

Após a conclusão dos filtros o conteúdo pode ser exportado para tipos de arquivos compatíveis com as aplicações mais utilizadas como, por exemplo, (Documento de Texto, Planilha Eletrônica ou Pagina Web), facilitando assim a manipulação dos dados conforme o necessário.

6.2.6 Logs do sistema

Visando a consistência dos dados da aplicação, foram desenvolvidos *logs* de acesso a todas as telas de cadastros do sistema. Quando qualquer alteração for realizada como salvar, atualizar ou excluir será registrada em uma tabela específica no banco de dados às ações realizadas, data da alteração, tipo de alteração e o usuário que a efetuou assim sendo a empresa tem um total controle sobre as ações de seus usuários como demonstra as figuras 32 e 33.

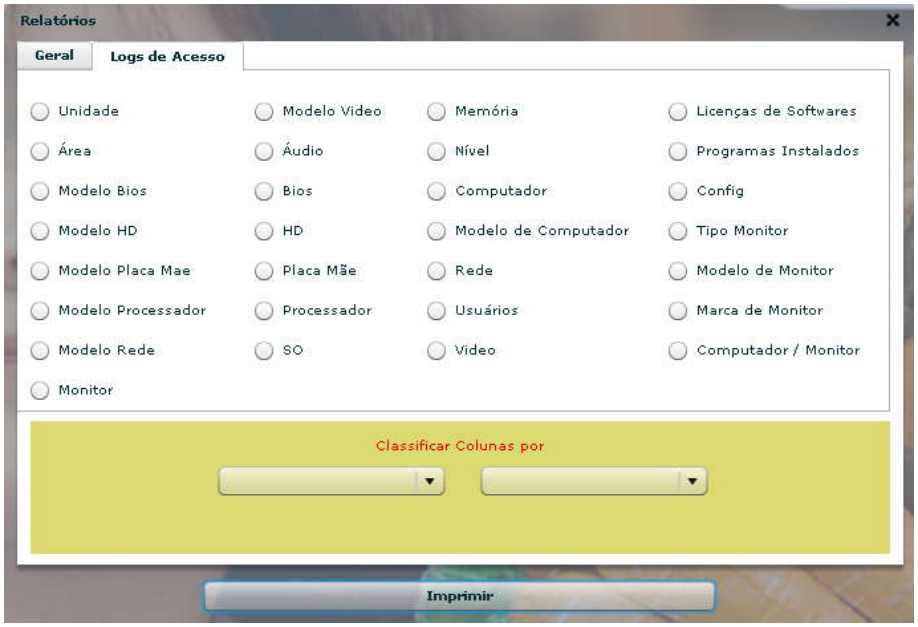

Figura 32 - Gerenciamento de Logs

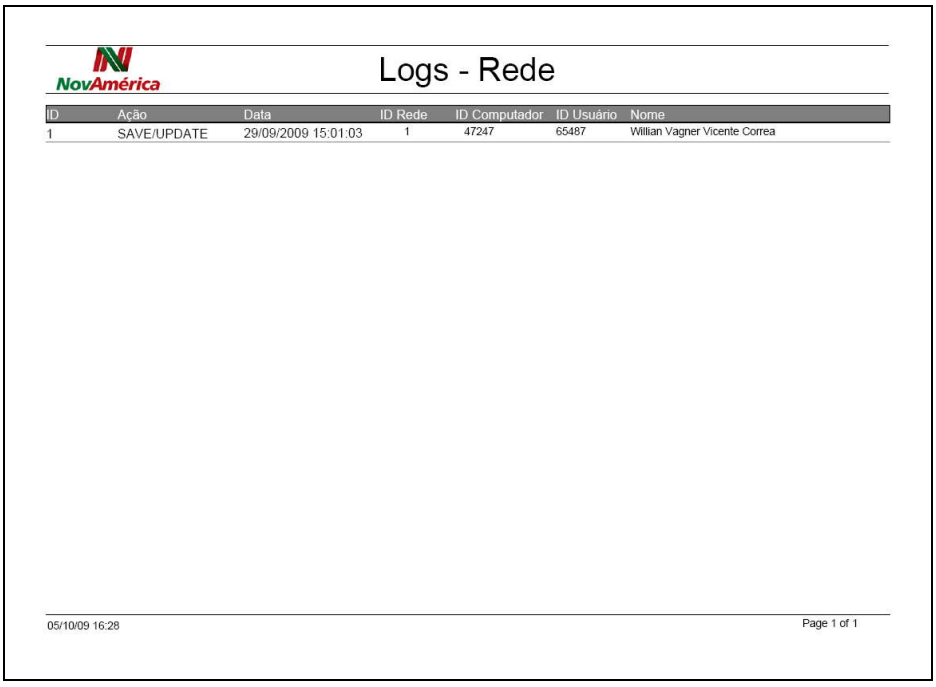

Figura 33 - Relatório de *Logs* referente ao cadastro de Redes

6.2.7 Papel de parede aleatório

Contanto com a beleza e a facilidade da manipulação dos componentes da linguagem *ActionScritp* foi desenvolvido para o usuário um sistema aleatório de troca de papéis de parede. Visando a publicidade e propaganda de novos produtos disponibilizados pela empresa como demonstra a figura 34

Figura 34 – Tela Principal da Aplicação

Esse processo é composto por mais de 25 papéis de paredes definidos pela empresa, sendo que a troca do mesmo é feita randomicamente quando o usuário estabelecer uma nova sessão no navegador de internet ou também ocorrerá à troca quando à página for atualizada.

6.2.8 Aplicação em modo de tela cheia

Para facilitar ainda mais a utilização da aplicação foi produzido um recurso que disponibiliza a visualização de toda a aplicação em modo de “tela cheia”, como demonstra a figura 35.

Figura 35 - Ambiente de Trabalho

Depois de pressionado o botão para definir a aplicação em modo de “tela cheia” todos os componentes visível da tela serão automaticamente redimensionados conforme as dimensões da janela atual. Após sua utilização o usuário pode voltar a modo de “tela normal” clicando no mesmo botão ou simplesmente pressionando a tela ESC do teclado.

CONCLUSÃO

Realmente a integração entre as tecnologias *Java e Adobe Flex* mostrou ser inovadora e dispor de muitos recursos interativos úteis ao desenvolvimento, estes recursos atendem a finalidade de exibição de informações muito bem e com bastante diversidade.

Em relação à integração das classes entre as tecnologias ouve um ponto negativo, pois é necessário duplicar as entidades, pois, para haver a transferência de objetos entre as tecnologias abordadas, é necessário que em ambos os lados tenham a classe implementada, o que gera uma quantidade de linhas codificadas maior.

Com o desenvolvimento das aplicações (*Desktop, Web)* tendo em vista as informações e as vantagens que estão sendo gerados para a empresa os mesmos são consistentes e confiáveis, tendo em vista a maneira de como estava sendo feita anteriormente.

Fazer as aplicações desenvolvidas ter um desempenho alto no quesito velocidade de persistência, tendo em vista a manipulação de muitas tabelas do banco de dados ao mesmo tempo e a possibilidade de fazer isso através da *internet,* houve uma avaliação criteriosa na análise de modelagem, para assim, alcançar o recurso desejado.

Com todos os tópicos abordados, a auditoria dos ativos de tecnologia e informação da empresa terá recursos para uma melhor avaliação dos *hardwares e softwares* utilizados.

Como sugestão para trabalhos futuros, fica o estudo da possibilidade de gerenciar com a aplicação *web* todos os equipamentos enviados para manutenção e assim poder fazer uma melhor avaliação referente ao grau de depreciação do imobilizado da empresa.

REFERÊNCIAS BIBLIOGRÁFICAS

DEITEL, H. M., DEITEL, P. J. **Java Como Programar,** trad. Edson Furmankiewicz, 3ª Edição, Editora Bookman, 2001.

ALISTAIR COCKBURN **Livro Escrevendo Casos de Uso Eficazes**, 1ª Edição, Editora Bookman

BlazeDS. Documentação disponível em: <http://livedocs.adobe.com/blazeds/1/blazeds_devguide>. Acessado em: 06 Abr. 2009.

ADOBE Flex. Visão geral do Flex: 30 mar.2007 Disponível em: <http://www.adobe.com/products/flex/overview/>. Acessado em: 06 abr. 2009.

PostgreSQL Documentação disponível em: <http://www.postgresql.org.br >. Acessado em 06 abr. 2009.

Hibernate BAUER, Christian; KING Gravin. **Hibernate in Action.** Grennwich: manning Publication Co., 2005.

SIMCH, Maicom Rafael Victor **Auditoria dos Sistemas de Informação Aliada à Gestão Empresarial.** Disponível **em:<**http://w3.ufsm.br/revistacontabeis/anterior/artigos/vIVn02/t005.pdf>. Acessado em 20 de junho de 2009.

Stadzisz, Paulo Cézar. **Projeto de Software usando a UML.** 2002. Centro Federal de Educação Tecnológica do Paraná.

MARTINELLI AUDITORES, **Noções de Auditoria de Sistemas Disponível em:** http://www.google.com.br/url?sa=t&source=web&ct=res&cd=4&ved=0CA4QFjAD&url=http%3A%2F%2Fwww.reitoria.rei.unicamp.br%2Fauditoria%2Fdocumentos%2Fmod2_ap_dia9.pdf&rct=j&q=defini%C3%A7%C3%A3o+auditoria+de+sistemas&ei=OGLwSpyGJoiWtgfIwbD6Bw&usg=AFQjCNFuCABFSXSHuyMh7YW9kI3L2SANUQ.

W3C (World wide Web Consortium). **Padrões web.** Disponivel em: <http://www.w3.org/ >. Acessado em 17 abr. 2009.

O'REILLY, Tim. **Web 2.0 Compact Definition: Trying Again.** 10 Dez. 2006. Disponivel em < http://radar.orelly.com/archives/2006/12/web-20-compact-definition-tryi.html > acessado em: 24 mar.2009.