P/ SCANNER

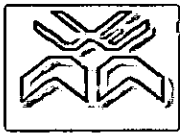
UNIVERSIDADE EDUARDO MONDLANE
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA & INFORMÁTICA

Trabalho de Licenciatura

SISTEMA DE GESTÃO DE DOCENTES DO DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA DA UEM: UMA APLICAÇÃO WEB COM ISPs_MYSQL

IT-171

BENJAMIM CHIO



UNIVERSIDADE EDUARDO MONDLANE
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

TRABALHO DE LICENCIATURA

Sistema de Gestão de Docentes do Departamento de Matemática e Informática da UEM: Uma Aplicação Web com JSPs e MySQL

Shang Lean Chao

UNIVERSIDADE EDUARDO MONDLANE
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

TRABALHO DE LICENCIATURA

Sistema de Gestão de Docentes do Departamento de Matemática e Informática da UEM: Uma Aplicação Web com JSPs e MySQL

Elaborado por:

Shang Lean Chao

Supervisor:

Dr. João Borges Dias

Maputo, Abril de 2005

DEDICATÓRIA

"Não basta introduzir o computador numa empresa para que ela se modernize. Ao contrário, a empresa moderniza-se, então precisa do computador" (Marcos Dantas, 1989).

Aos meus pais, irmãos e sobrinhos que sempre me apoiaram com muito carinho não só durante o curso mas também no dia a dia da minha vida.

À Rossana, que a sua simplicidade e dinamismo na sua carreira profissional lhe encaminhem ao sucesso.

AGRADECIMENTOS

À Deus e aos meus pais, por me terem dado a educação que hoje tenho e que hoje sirvo-me dela para atingir muitos objectivos da minha vida.

Aos meus irmãos pela força e incentivo ao longo do curso e do trabalho, especialmente ao Júlio por me ter apoiado e incentivado durante o decorrer do curso, à Sara por ter corrigido com muita cautela o Português deste relatório, à Anabela e aos meus sobrinhos que sempre me apoiaram ao longo deste trabalho.

Ao meu supervisor, Dr. João Dias pela dedicação, orientação, compreensão, críticas e ensinamentos metodológicos por ele transmitidos ao longo do curso e do trabalho.

Aos docentes do DMI que me facilitaram na recolha de dados para eleboração deste trabalho, especialmente aos Drs. Culpa, Getimane, Mulenga e Miguel.

À minha querida amiga e colega, dra. Rossana que acompanhou todas as fases do curso até o deste trabalho.

Aos meus tios queridos que tanto lhes estimo Nuro e Batule pelo imenso carinho e incentivo que sempre me deram ao longo deste trabalho.

À Shaida, Awa, Nilsa e Kadaf pela ajuda moral que sempre me deram para o término do trabalho.

Aos meus colegas e amigos, pelo carinho e força que sempre me deram, especialmente ao Hamilton pelas críticas no trabalho.

Aos Drs. Neill e Colin pelas dicas de desenvolvimento em ambiente Web.

DECLARAÇÃO DE HONRA

"Declaro por minha honra, que este trabalho é resultado da minha investigação e foi realizado somente para ser submetido como Trabalho de Licenciatura em Informática na Universidade Eduardo Mondlane."

Shang Lean Chao

(Shang Lean Chao)

Maputo, Abril de 2005

RESUMO

A necessidade de aquisição de novos espaços, a importância da divulgação de produtos e serviços por parte das organizações a fim de ampliar as redes de comunicação e a interacção com o mundo a tempo real, tem vindo a crescer a passos gigantescos, o que faz com que os desenvolvedores de sistemas pensem em modelos usando tecnologias viradas à Internet.

A famosa rede mundial de comunicações - Internet, reúne as melhores condições para ser um meio de comunicação mundial, permitindo com que milhões de computadores e de utilizadores desta rede comuniquem-se e colaborem de maneira fácil, rápida e segura através dos serviços que ela suporta.

As aplicações desenvolvidas em ambiente Web são muito atractivas para empresas e organizações, pois, por um lado são facilmente acessíveis para os utilizadores já familiarizados com a navegação e, por outro lado, os utilizadores podem aceder a fontes de informação corporativas, actualizando-se de forma instantânea e adaptando-se facilmente aos requisitos crescentes da organização.

Neste trabalho, descrevem-se algumas tecnologias que possibilitam o desenvolvimento de aplicações desta natureza com maior incidência para a escolhida. A sua implementação no Sistema de Gestão de Docentes do DMI teve como principal objectivo a automatização de alguns processos de trabalho de forma a se ter uma melhor prestação dos seus serviços.

Actualmente, a informação referente aos docentes do DMI encontra-se armazenada em inúmeros arquivos em diversos armários dificultando assim a rápida aquisição da informação e fazendo deste modo que se perca muito tempo na busca da informação referente a docentes.

Com intuito de melhorar estes processos de trabalho, surge a necessidade de se automatizar as transacções que envolvem as actividades referentes a docentes, optando-se por se desenvolver uma aplicação com interface Web, que possa disponibilizar atempadamente informação referente aos docentes do DMI, ou seja, um sistema que manipule a gestão de docentes do DMI via Internet.

Para o efeito, usou-se como linguagem de análise e desenho do modelo – UML e, na fase de desenvolvimento, a linguagem JSP que incorpora códigos JAVA na conexão com a base de dados produzida em MySQL, com vista a tornar a aplicação mais estável, robusta e que possa responder as necessidades actuais. Utilizou-se também uma tecnologia denominada STRUTS, que auxilia o desenvolvimento de sistemas Web, isto é, possui mecanismos capacitados para separar os códigos Java, JSP e HTML,

permitindo o isolamento de erros, organização do código fonte, manutenção do sistema e ainda possui bibliotecas auxiliares.

Como o Sistema Proposto usa uma interface Web, aborda-se ao longo da descrição do trabalho mecanismos de segurança que prevêem os diversos ataques físicos e lógicos que possam ocorrer durante o funcionamento dum sistema baseado em tecnologias Web.

NOTAÇÕES E GLOSSÁRIO

Notações

Notação	Descrição
FTP	File Transfer Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JDBC	Java Database Conector
JSP	Java Server Pages
OO	Object Oriented
RUP	Rational Unified Process
SMTP	Simple Mail Tranfer Protocol
SQL	Structured Query Language
SSL	Secure Socket Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
UML	Unified Modeling Language
URL	Universal Resource Language
VPN	Virtual Private Network
WWW	World Wide Web
XML	Extensible Markup Language

Glossário

Elemento	Descrição
Beans	É uma classe *Java* que segue um conjunto de convenções simples de desenho e nomeação delineado pela especificação de *JavaBeans*. Este não precisa estender uma determinada classe ou implementar uma determinada interface (Jeveaux, 2003).
Browser	É um aplicativo voltado à Internet cuja função é navegar pelas páginas da www. Os programas com esta função mais conhecidos são o *Microsoft Internet Explorer* e o *Netscape Navigator*. Eles são responsáveis por exibir as imagens, sons e textos disponíveis na *Internet* (Prado, 2003).
Container	É o ambiente em que os *servlets* são executados. Este gere as instâncias do *servlet* e provê os serviços de rede necessários para as requisições e respostas (Oliveira, 2001).
Java Beans	É uma classe *Java* padrão que segue um conjunto de convenções simples de desenho e nomeação. Este não precisa estender uma determinada classe ou implementar uma determinada interface (Jeveaux , 2003).
Plugins	Utilitário que aumenta a funcionalidade de um determinado programa. Por exemplo, para conseguir visualizar vídeos em páginas *Web* é necessário instalar o *"plug-in"* no browser (*Internet Explorer* ou *Netscape Navigator*) correspondente a essas funcionalidade (Graham, 1998).
Proxy	Um servidor deste tipo está encarregue de navegar nas páginas em nome do utilizador e guardar essas páginas no seu disco, ou seja, se o *Internet Explorer* do utilizador estiver configurado para navegar com o *proxy* do fornecedor de acesso, as páginas visitadas pelo utilizador ficam armazenadas no *proxy* (Thomas, 2001).
Tag	São comandos que informam ao navegador uma determinada acção que este deve executar. Encontram-se separados pelos caracteres "<" e " >" (Prado, 2003).
Servlet	É um programa do lado do servidor que é activado pelo servidor *Web* para atender pedidos HTTP. É executado numa máquina virtual Java no servidor *Web* e normalmente realiza alguma computação para gerar conteúdo da resposta HTTP (Morisseau-Leroy et al, 2001).

Índice

Índice de Figuras

Capítulo I

1. – INTRODUÇÃO

1.1 – Enquadramento

As Tecnologias de Informação e Comunicação (TICs) estão sendo implementadas em diversas instituições, criando condições para uma melhor produção, distribuição e uso da informação (Barreto, 1997).

A tomada de decisão nas organizações tem sido um processo complexo, devido a quantidade de informação envolvida, sua complexidade e frequência com que ela se altera (Pereira, 1998).

A Universidade Eduardo Mondlane (UEM) – a maior instituição de ensino superior do país, possui cerca de 916 docentes distribuídos por vários departamentos, pertencentes a 11 faculdades, gerindo um grande fluxo de informação em torno da actividade lectiva (42 cursos de licenciatura), de extensão e de investigação (Mazula, 2003).

Os departamentos, de modo a gerirem melhor as suas actividades de investigação, leccionamento e extensão, subdividem-se em secções correspondentes às diversas áreas/subáreas científicas existentes. Cada secção, em termos de estrutura organizacional, representa um grupo de docentes dirigido por um chefe de secção.

Compete aos departamentos e as secções o provimento em docentes para as suas licenciaturas e para as das outras faculdades, garantindo o regular funcionamento da leccionação das disciplinas. Cada curso, em termos da gestão corrente do processo de ensino-aprendizagem é dirigido por um Director do Curso.

Todos os dados possíveis de serem utilizados nos diversos processos decisórios sobre a **actividade docente** ao nível dos departamentos (ou secções), encontram-se armazenados de uma forma dispersa em relatórios, formulários e documentos, impossibilitando muitas vezes a produção rápida e fiável da informação pretendida.

No início de cada semestre, o departamento envia formulários aos docentes a fim de actualizarem os seus contratos ou então registarem-se se for o caso de candidatos a docentes. Estes, por sua vez, ao longo do período de docência de uma dada disciplina, reportam oralmente o desenvolvimento de suas actividades e no fim de cada semestre enviam relatórios e preenchem formulários. Estes processos são contínuos, morosos e despendem muita força humana de quem neste circuito participa.

O **Departamento de Matemática e Informática** (DMI) pertence à Faculdade de Ciências gerindo cerca de meia centena de docentes garantindo a leccionação de disciplinas de informática, matemática e estatística para os seus três cursos e para quase todas as licenciaturas da UEM. Para melhorar a gestão das actividades de leccionação, investigação e extensão no DMI, urge a necessidade de desenvolver um Sistema de Gestão de Docentes que disponibilize informação atempada, útil e fiável aos decisores.

1.2 – Definição do Problema

O DMI é uma unidade orgânica com uma estrutura organizacional composta por três secções – Informática, Matemática e Estatística. Cada uma destas secções de docentes possui um curso de nível de licenciatura para cada turno (diurno e pós - laboral) estando programadas acções de formação de nível de pós-graduação.

Em termos de actividade lectiva, prevê-se que a partir de 2005 os 50 docentes (45 em tempo inteiro e 5 tempo parcial) garantam, anualmente, a leccionação de cerca de 78 disciplinas internas (3 planos de estudos diurnos e 2 pós-laborais) para aproximadamente 653 estudantes. Ao nível externo (disciplinas do outros cursos da UEM), o departamento cobre cerca de 23 disciplinas para mais de 2000 estudantes. Como reflexo deste aumento da carga docente, o número de contratados a tempo parcial como assistentes e monitores é cada vez maior, tornando a gestão da actividade docente bastante complexa.

Esta gestão é baseada fundamentalmente nos dados constantes, nos formulários e relatórios que os docentes preenchem no início e fim de cada semestre/ano lectivo reportando as suas actividades nos processos individuais, nos relatórios das secções/direcções de curso e em vários elementos recolhidos de uma forma avulsa consoante as necessidades. Tem sido difícil incluir ratios de actividade e dados históricos sobre um docente ou uma disciplina no processo decisório, por exemplo, para efeitos de alocação do serviço do docente, pois o tempo de recolha e sumário dos dados necessários seria longo.

O processo de consulta da situação de cada docente num dado semestre ou ano lectivo em termos de actividade lectiva, ou seja, que disciplinas e onde leccionou (no DMI ou noutro departamento), o número de estudantes que teve, as taxas de aprovação/reprovação que obteve, é muito moroso (dados demasiado dispersos) e por ser efectuado manualmente, é susceptível a falhas e/ou erros.

A falta de partilha e centralização de todos os dados sobre os docentes origina que os mesmos sejam lançados em formulários diferentes, escritos diversas vezes (podendo originar inconsistências e redundâncias), tornando difícil a obtenção de diversas listagens e estatísticas sobretudo quando se pretende cruzar variáveis como o tipo de docente (tempo inteiro/parcial), a categoria académica, o

local (departamento), turno e semestre de leccionação, o número de estudantes e as disciplinas leccionadas. Com muito esforço se pode obter, por exemplo, o ratio docente/estudante de todas a disciplinas leccionadas fora do departamento ou por uma secção, taxas das aprovações/reprovações de uma disciplina nos últimos semestre/anos.

Para melhorar a gestão da actividade docente do DMI, propõe-se com este trabalho, a criação de uma bases dados de docentes inserida numa aplicação *Web*, de modo a facilitar a gestão dos processos relacionados com a busca/lançamento de dados pessoais, a alocação das turmas/disciplinas e outros relatórios/listagens de interesse garantindo uma integração lógica de todos os dados num único modelo conceptual, diminuindo a redundância dos lançamentos, aumentando a partilha, a acessibilidade e o tempo de resposta. Julga-se que, com o modelo proposto, se poderão colmatar muitas das insuficiências enunciadas.

1.3 – Objectivos

1.3.1 – Geral

- Desenvolver um Sistema de Gestão de Docentes (SGD) para o DMI criando-se uma base de dados e uma aplicação baseada em Tecnologias *Web*.

1.3.2 – Específicos

- Identificar e dimensionar os constrangimentos do sistema actual de gestão de docentes vigente no DMI;
- Avaliar a aplicabilidade de diversas tecnologias *Web*;
- Conceber o SGD do DMI aplicando uma metodologia de análise e modelação orientada a objectos - OO[1];
- Desenhar e propor soluções de implementação para o SGD;
- Avaliar a usabilidade da aplicação proposta.

[1]O paradigma da Orientação a objectos – OO visualiza um sistema de software como uma colecção de agentes interconectados chamados objectos (Bezerra, 2003)

1.4 – Metodologia Usada

Apresenta-se a seguir uma breve descrição da metodologia usada para alcançar cada objectivo proposto no trabalho.

Para identificar e dimensionar os problemas do sistema actual de gestão de docentes vigente no DMI foram necessários:

- Analisar o sistema actual recorrendo-se à documentação disponível no DMI;
- Entrevistar o pessoal envolvido, nomeadamente chefes de secção e docentes, com o objectivo de apurar-se as suas opiniões, sugestões e críticas em relação a implementação do modelo proposto. Foram entrevistados cinco docentes, dentre eles os (três) chefes de secção. As entrevistas de forma não estruturada, consistiram na captura de informação ao longo do desenvolvimento do sistema, na tentativa de envolver o usuário gradualmente na fase de desenvolvimento, tendo-se entrevistado por vezes mais de uma vez as mesmas entidades com o intuito de obter a mais correcta informação sobre o funcionamento das actividades referentes aos docentes.

Para avaliar a aplicabilidade de diversas tecnologias Web foi necessário:

- Elaborar uma pesquisa bibliográfica, estudando-se as vantagens e desvantagens de umas tecnologias em relação às outras (independente do problema em estudo);
- Avaliar e dominar um conjunto mais reduzido de soluções tecnológicas que estivessem de acordo, por um lado, com a pequena rede existente no departamento, os custos de implementação a suportar e a capacidade futura de manutenção da aplicação por parte do DMI (capacidade humana e material) e, por outro, com os requisitos funcionais e não funcionais do sistema a construir.

Para conceber o SGD do DMI efectuou-se:

- A análise e o desenho usando-se a linguagem de Análise Orientada à Objectos (OO) *Unified Modeling Language* (UML) seguindo-se um processo de desenvolvimento próximo do proposto pela *Rational Unified Process*;
- O desenho de uma base de dados relacional (a opção por um modelo de dados objecto - relacional ou puro de objectos foi descartada);
- Construção de um conjunto de protótipos evolutivos.

Para desenhar e propor soluções de implementação para o SGD do DMI foi necessário:

- Implementar a base de dados usando o Sistema de Gestão de Base de Dados (SGBD)-*My Structured Query Language (MySQL)*, por ser um sistema relacional de gestão de base de dados muito rápido e robusto (Welling, 2001);
- Desenhar as *interfaces Web* usando *Java Server Pages* (JSP) por ser *open source* (ou seja possui código-fonte aberto) e, porque aliado ao MySQL, permite produzir páginas *Web* dinâmicas e interativas (Bond, 2002);
- Conectar a base de dados e o *Layout* através de códigos *Java;*
- Documentar o sistema – produziu-se a documentação do sistema e um manual do utilizador para facilitar o uso e a manutenção do mesmo;
- Organizar a plataforma e os recursos necessários (*hardware* e *software*) para o sustento do modelo construído;
- Testar separadamente os módulos durante o desenvolvimento.

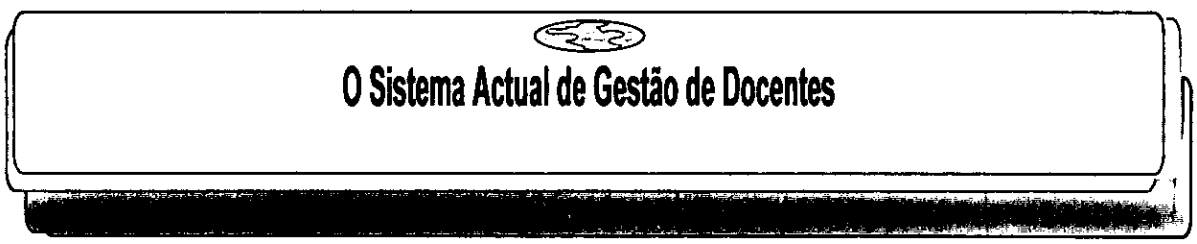
Para avaliar a aplicação proposta foi necessário:

- Elaborar-se diversos tipos de testes de forma a acompanhar o funcionamento do sistema e o desenvolvimento dos interessados, detectando falhas e/ou erros do mesmo;
- Testar-se a usabilidade da aplicação proposta por parte dos 3 chefes de secção introduzindo-se modificações no esquema de navegação e nas interfaces.

1.5 – Estrutura do Trabalho

Este trabalho está dividido em 7 capítulos apresentando-se no primeiro, o enquadramento do trabalho, os problemas identificados no sistema actual do DMI responsável pela gestão de docentes bem como os objectivos e a metodologia que se usou para atingir tais objectivos. O capítulo seguinte faz uma abordagem descritiva do funcionamento do sistema actual de gestão de docentes. O terceiro capítulo refere-se as tecnologias de suporte ao trabalho. Os dois capítulos seguintes, referem-se, a modelação do Sistema (o quarto capítulo) e os critérios de segurança previstos. As conclusões e recomendações são descritas no penúltimo capítulo, tendo se listado no último capítulo a bibliografia ao longo do trabalho.

Capítulo II



O Sistema Actual de Gestão de Docentes

2 – O SISTEMA ACTUAL DE GESTÃO DE DOCENTES

Ao longo deste capítulo serão detalhados aspectos relacionados ao funcionamento do actual sistema semi-manual de gestão de docentes que rege o DMI. Apresentar-se-ão também os requisitos e necessidades apresentadas pelos entrevistados de modo a dar uma visão do modelo proposto.

2.1 – Funcionamento do Sistema Actual

O sistema actual de gestão de docentes realiza várias actividades dentre elas o **Cadastro de Docentes** e a **Planificação das Actividades** destes em cada semestre ou ano lectivo, de acordo com o tipo de docente (*entrevistado 1*).

Cadastro de Docente:

Um indivíduo que queira ser docente deve ser pelo menos licenciado. A sua candidatura a docência passa por diversas fases, a primeira é a emissão da sua documentação aos Recursos Humanos. Esta documentação deve conter uma carta dirigida ao Reitor e o seu Certificado de Habilitações Literárias. Se o indivíduo candidato tiver um certificado nacional, ele exibe o seu certificado, caso contrário exibe um certificado de equivalência.

Os Recursos Humanos enviam a documentação ao departamento de interesse do candidato e este retorna o seu parecer ao mesmo, atendendo às necessidades do departamento e após uma entrevista feita ao candidato.

Prossegue-se então com o contracto do Docente que segundo o entrevistado 1, este poderá ser de dois tipos:

- Docente a tempo Inteiro,
- Docente a tempo parcial (tem de renovar contracto).

Planificação Anual das Actividades do Docente:

Cada docente preenche um plano anual de actividades que inclui as actividades atribuídas pelo Chefe de Secção. Este poderá leccionar um certo número de disciplinas tanto dentro do Departamento assim como fora, isto é, ter disciplinas que lecciona num outro Departamento.

O plano de actividades é um pequeno formulário que o docente tem de preencher contendo campos como (Entrevistado 1):

- Dados pessoais do docente;
- Actividades pedagógicas;
- Actividades científicas programadas;
  - Área de investigação e seu respectivo tema (pode usar um tema de Licenciatura que ele tenha supervisionado);
  - Divulgação dos resultados previstos (mês e ano);
  - Elaboração de material instrutivo;
  - Manuais;
  - Textos de apoio;
- Trabalho administrativo – O docente pode exercer actividades administrativas enquanto lecciona (no caso de ser chefe de secção, director do curso ou chefe de comissão científica);
- Trabalho de extensão programado – trabalhos prestados pelo docente fora do Departamento;
- Outras actividades programadas.

O sistema actual de gestão de docentes vigente no DMI é semi-manual, onde o docente tem de preencher um formulário que serve de cadastro para o sistema. Este formulário é reencaminhado para a Secretaria que deverá arquivá-lo e entrega-lo para análise ao chefe de secção.

No que se refere à atribuição e divisão de disciplinas por docente, este pode leccionar até três disciplinas num semestre ou ano (se a disciplina for anual – Entrevistado 1).

O docente faz uma programação semanal das horas, onde pode separá-los em aulas teóricas, práticas e aulas de consulta (ibidem).

Um docente pode realizar várias actividades que podem ser consideradas científicas (Entrevistado 2). Estas actividades devem ter uma área e um tema de investigação específico, devendo ser devidamente calendarizadas de acordo com o mês e ano previstos.

Um docente pode também elaborar material instrutivo que facilite e complemente as suas aulas. Estes materiais podem ser:

- Textos de apoio e
- Manuais.

Para além destas actividades, diversas listagens relacionadas com docentes são também produzidas pela secretaria do DMI ao longo de cada semestre lectivo, tais como: relatórios em que são alocados certos docentes a turmas e departamentos específicos, estatísticas referentes às actividades desempenhadas por um docente ao longo dum certo período. Sempre que o departamento tem necessidade de gerar relatórios para diversos fins, tem tido um trabalho muito árduo pois obriga que a entidade responsável por esta acção conte manualmente os dados dispostos em arquivos. A figura 2.1 ilustra o sistema actual de gestão de docentes do DMI:

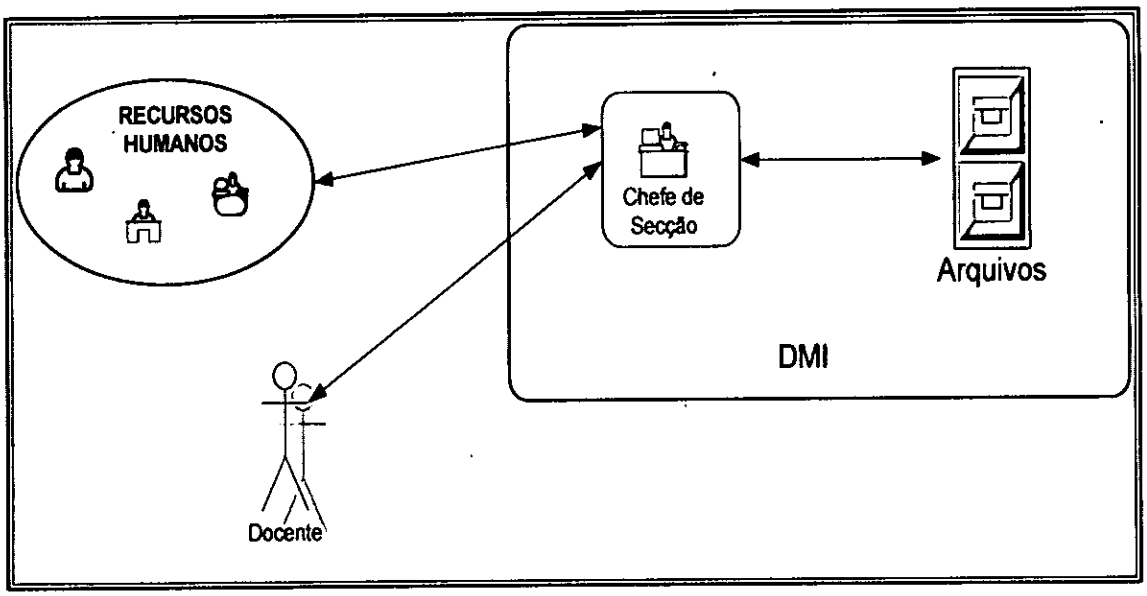


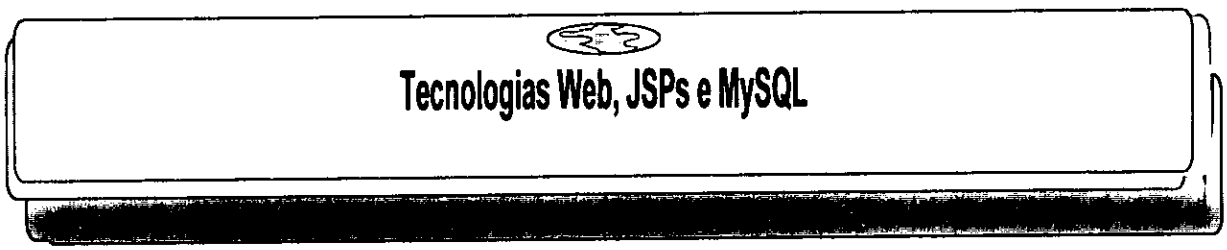
Figura 2 1 - Sistema actual de gestão de docentes do DMI

Requisitos Funcionais:

O sistema proposto deverá:

- Gerar relatórios ricos em informação, mais profundo possíveis dum determinado semestre, ano lectivo, ou conjunto de anos.
- Produzir estatísticas;
- Gerar relatórios dos artigos científicos produzidos por um docente;
- Gerar relatórios dos Manuais que este produziu e disponibilizou;
- Saber quantos alunos foram aprovados, reprovados ou prescritos numa certa disciplina leccionada por um certo docente;
- Saber dum docente:
  - Secção a que ele pertence;
  - Projectos a que ele está envolvido;
  - Percentagem de inscritos, prescritos, aprovados, reprovados, duma disciplina leccionada por um docente X.
- Gerar relatórios dos Manuais que este produziu e disponibilizou;
- Gerar relatórios de outras actividades que um docente pode realizar ou realizou;
- Gerar relatórios sobre dados dum docente;
- Gerar relatórios de possíveis justificações por parte do docente, caso este não tenha cumprido com o plano programado.

Capítulo III



3 – TECNOLOGIAS WEB, JSPS E MYSQL

Tendo como objectivo específico o estudo de várias tecnologias *Web*, este capítulo vai abordar as diversas plataformas que foram estudadas, sendo a destacar alguns conceitos ligados ao ambiente *Web*, a *Internet* e seus serviços, a tecnologia de páginas dinâmicas escolhida, o servidor e navegador *Web* bem como o Sistema de Gestão de Base de Dados usado para o armazenamento de dados.

3.1 – Internet

A *Internet* surgiu de uma pequena rede[2] experimental de computadores criada em 1969 pela *Advanced Research Projects Agency* (ARPA) do Departamento de Defesa dos EUA, para permitir a partilha de recursos computacionais, tais como bancos de dados, computadores de alto desempenho e dispositivos gráficos, entre os pesquisadores e fornecedores contratados pelo Departamento. Logo ela passou a ser usada também como meio de cooperação entre os participantes do projecto, possibilitando o uso de correio electrónico, dentre outros serviços (Web2, 2004).

Em 1980, essa rede experimental foi dividida em outras duas: a *Arpanet* para pesquisa civil com fins militares e, a *Military Net* - Milnet com fins exclusivamente militares. A interligação destas foi chamada de *Defense Advanced Research Projects Agency Internet Work*, nome que foi abreviado posteriormente para *Internet* (ibidem).

Para a comunicação dessas duas redes, foi desenvolvido um protocolo[3] denominado ***Transmission Control Protocol/Internet Protocol*** – **TCP/IP**, que usava uma tecnologia de comunicação em redes de computadores, a que se chamou de ***Internet Protocol*** – **IP** (Morisseau-Leroy et al, 2001).

A invenção de tecnologias de informação nos anos 60, como o telégrafo, telefone, rádio e computador prepararam o terreno para integração de novas capacidades e rumo a formas de comunicação mais rápidas e eficientes. A ***Internet*** é portanto, um mecanismo de disseminação da

[2] **Rede** é um conjunto de computadores interligados entre si com objectivo de intercâmbio de informação (Coelho, 1996).
[3] **Protocolo** – linguagem de comunicação entre duas ou mais redes abertas.

informação, divulgação mundial e um meio para colaboração e interacção entre indivíduos e seus computadores, independentemente de suas localizações geográficas (Leiner et al, 2004).

Os **principais serviços** que se pode utilizar através da *Internet* são:

- **Páginas de informação** – também são conhecidas por ***World Wide Web (WWW) pages***. Este, é actualmente o tipo de serviço mais popular da *Internet*. Na verdade, é o único que alguns utilizadores menos experientes conhecem ou ouviram falar, de tal modo que confundem toda a *Internet* com este serviço (Morisseau-Leroy et al, 2001);
- **Correio Electrónico** – também conhecido por ***E-mail***, é responsável pelo envio e recepção de mensagens electrónicas através de um programa apropriado (Prodigio, 2001);
- **Sala de Conversas On-Line – ou Internet Relay Chat (IRC),** é o serviço que permite conversar com outros utilizadores em tempo real em modo de texto (ibidem);
- ***File Transfer Protocol* (FTP)** – protocolo de transmissão de ficheiros. Este, faz parte dos protocolos do conjunto TCP/IP e é usado para transferir ficheiros entre terminais duma rede (CCNA, 1999);
- **ICQ** – é o programa mais popular utilizado para um serviço que ainda não tem um nome consensual. Alguns chamam a este serviço "mensagens instantâneas", outros chamam-lhe "listas de amigos", outros chamam-lhe mesmo "ICQ", embora existam outros programas com nomes diferentes que fazem a mesma coisa entre outros (Prodigio, 2001).

Para se ter acesso à *Internet*, é necessário que se tenha acesso a um computador que esteja na rede e que tenha uma conta num ***Service Provider*** (provedor de acesso à *Internet*), que é uma organização que provê acesso à *Internet* aos seus clientes (Prodigio, 2001).

Para a visualização dos aplicativos *Web* na *Internet*, estes deverão ser alocados num **servidor *Web*** a como a está descrito a baixo (Morisseau-Leroy et al 2001).

3.2 – Servidor JSP – *Tomcat*

De acordo com Santos (2004), para poder executar uma página JSP é necessário ter um servidor que receba a solicitação do usuário, processe-a e devolva uma resposta ao mesmo. O servidor mais popular existente é o *Tomcat* da *http://jakarta.Apache.org/tomcat*.

Alguns aspectos a serem considerados num servidor são (ibidem):

- Processamento de uma solicitação JSP:
  - Na primeira requisição a uma página JSP, a mesma é "compilada" e uma classe Java (*Servlet*) é gerada;
  - Esta classe é então usada pelo servidor JSP para processar toda solicitação do usuário àquela página e, devolver a devida resposta ao mesmo;
  - Podemos dizer então que uma página JSP passa por dois estágios: um de tradução (compilação) e outro de processamento da requisição.

- Nas requisições subsequentes à página JSP, caso a mesma não tenha sido modificada, ela não será mais compilada;
- No *Tomcat*, os ficheiros gerados da compilação da página ficam localizados no directório *work*;
- Como toda página JSP é transformada em um *Servlet Java*, os servidores JSP hoje são mais conhecidos por *Container Servlet*.

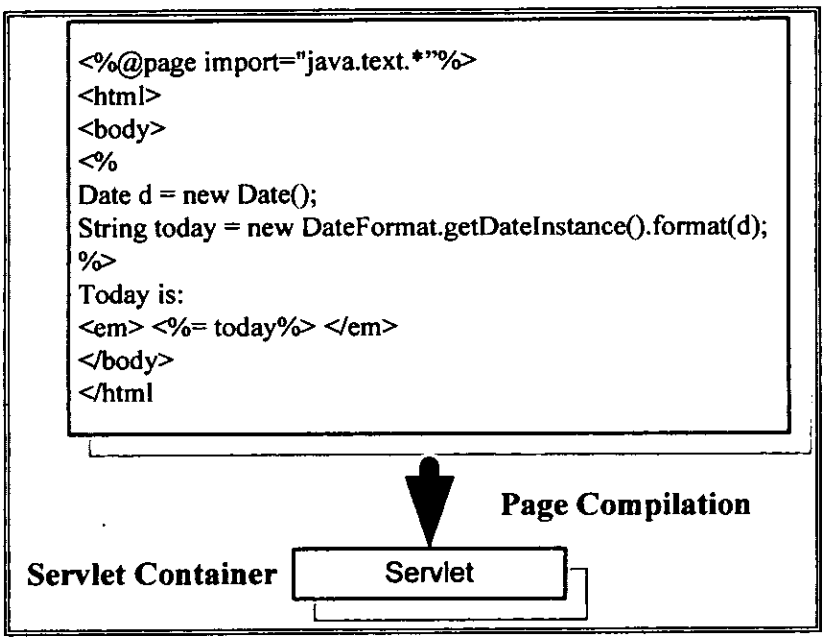


Figura 3 1 - Exemplo de execução de um JSP (Santos, 2004)

Para que seja possível a visualização das páginas alocadas num servidor é necessário que se instale um navegador da *Web*.

3.3 – Navegador *Web*

Um **navegador** (também conhecido como *browser Web* ou simplesmente *browser*) é um programa que habilita seus usuários a interagirem com documentos HTML[4] hospedados num servidor *Web* (Wikipedia, 2004).

Navegadores se comunicam com servidores *Web* usando primariamente o protocolo ***Hypertext Transfer Protocol*** – **HTTP**[5] para recuperar páginas *Web*, que são identificadas pela ***Universal Resource Locators*** – **URL** *http:* O protocolo HTTP, permite aos navegadores tanto recuperar como submeter informações para um servidor. O formato de ficheiros que uma página usa, é normalmente o HTML, sendo identificado no protocolo HTTP através de um indicador do seu tipo de conteúdo. A maioria dos navegadores suporta uma grande variedade de formatos em adição ao HTML tais como JPEG, GIF e PNG para imagens e, também podem geralmente ser estendidos para suportar outros formatos através de *plugins*[6]. Da mesma forma, muitos navegadores suportam vários outros tipos de URLs com seus protocolos correspondentes, tais como *ftp:* para FTP, *https*: para HTTPS (uma versão encriptada via SSL do HTTP) (Wikipedia, 2004).

A combinação do tipo de conteúdo da URL e do protocolo, permite que desenvolvedores de páginas *Web* embutam imagens, animações, sons e vídeo na mesma, ou tornem tais conteúdos acessíveis através das páginas (Morisseau-Leroy, 2001).

Navegadores mais primitivos suportavam somente uma versão mais simples do HTML. Porém, o desenvolvimento rápido dos navegadores levou à criação de dialectos não padronizados do HTML, causando problemas de interoperabilidade na *Web*. Navegadores mais modernos (tais como o *Internet* Explorer, Mozilla, Firefox, Opera e Safari) suportam versões padronizadas do HTML e

[4] HTML – *Hypertext Markup Language*
[5] A estrutura do protocolo HTTP será apresentada com maior destaque no capítulo que aborda conceitos de segurança – capítulo 5
[6] *Plugins* – são utilitários que aumentam a funcionalidade de um determinado programa

XML[7] e mostram páginas numa maneira uniforme através das plataformas que as suportam (Wikipedia, 2004).

3.4 – Tecnologia usada para o Desenvolvimento da Aplicação *Web* – JSP

Java Server Pages, é uma tecnologia baseada na linguagem *Java*, que simplifica o processo de desenvolvimento dinâmico de páginas *Web*. O JSP funciona como um *container* (compartimento) que incorpora elementos dinâmicos (Romão, 2004).

Como uma linguagem *script,* esta funciona do lado do servidor ou seja, as páginas JSP são ficheiros–texto com a extensão "*.jsp*" que substituem os ficheiros estáticos HTML. Estas, para além de utilizar objectos do servidor, podem incorporar e manipular objectos próprios, como *Appletos* e *Servlets*[8](ibidem).

JSP é uma tecnologia para o desenvolvimento de aplicações *Web* semelhante ao *Microsoft Active Server Pages* (ASP) porém, tem a vantagem da portabilidade de plataforma podendo ser executado em outros Sistemas Operativos além dos da Microsoft. Ela permite desenvolver páginas que produzam aplicações que permitam o acesso a base de dados, ficheiros – texto, captação de informações a partir de formulários, captação de informações sobre o visitante e sobre o servidor, uso de variáveis e ciclos entre outras coisas (Ulisses, 2004).

Para além desta tecnologia da Microsoft, a Sun produziu os *servlets* para o desenvolvimento de aplicações *Web* a partir de componentes Java que executam também no lado do servidor como os JSPs. Estas duas tecnologias fazem parte da plataforma *Java 2 Platform Enterprise Edition* - J2EE que fornece um conjunto de tecnologias para o desenvolvimento de soluções escaláveis e robustas para a *Web* (Ulisses, 2004).

Em relação aos *servlets*, os JSPs não oferecem nada que não se possa conseguir com os *servlets* puros. O JSP, entretanto oferece a vantagem de ser facilmente codificado, facilitando assim a elaboração e manutenção de aplicações *Web*. Além disso, essa tecnologia permite separar a programação lógica (parte dinâmica) da programação visual (parte estática), facilitando o

[7] XML – Extensible Markup Language

[8] ***Servlets***, assim como as páginas JSP tratam chamadas (*request*) e respostas (*response*) do HTTP. A diferença entre os JSPs e os *servlets* é que as páginas JSPs são mais fáceis de se trabalhar com o *front-end* do usuário, ou seja o HTML (Romão, 2004).

desenvolvimento de aplicações mais robustas, onde o programador e o *designer* podem trabalhar no mesmo projecto, mas de forma independente. Outra característica do JSP é produzir conteúdos dinâmicos que possam ser reutilizados (Ulisses, 2004).

3.4.1 – Ciclo de Vida JSP

JSPs são páginas HTML que incluem código Java e outros tags (instruções) especiais. A primeira vez que uma página JSP é carregada pelo *container* JSP, o código *Java* é compilado gerando um *Servlet* que é executado, este por sua vez gerando uma página HTML que é enviada ao *browser*. As chamadas subsequentes são enviadas directamente ao *Servlet* gerado na primeira requisição, não ocorrendo mais as etapas de geração e compilação do *Servlet* (Bond, 2002).

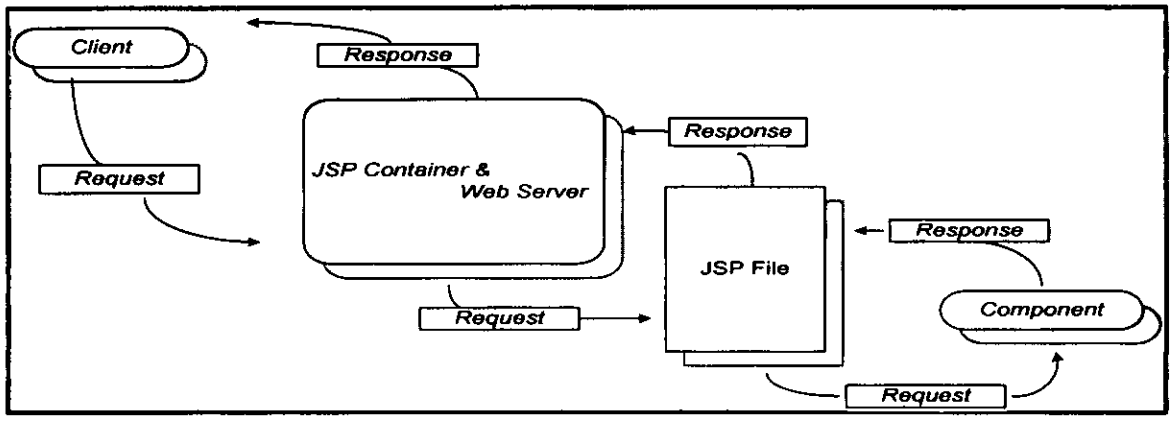


Figura 3 2 - Passagem de dados entre Cliente e Servidor (Sun, 2000)

Quando o cliente faz a solicitação de um ficheiro JSP, é enviado um *object request* (pedido) para a *JSP Container*. A *JSP engine* envia a solicitação de qualquer componente (podendo ser um *JavaBeans component*, *servlet* ou *Enterprise Bean*) especificado no ficheiro (Ulisses, 2004).

O componente controla a requisição possibilitando a recuperação de ficheiros em base de dados ou outro dado armazenado, em seguida passa o *object response* (resposta) de volta para a *JSP engine* (ibidem).

A *JSP engine* e o servidor *Web* enviam a página JSP revista de volta para o cliente, onde o usuário pode visualizar os resultados através dum *browser*. O protocolo de comunicação usado entre o cliente e o servidor pode ser HTTP ou outro(ibidem).

3.4.2 – Conexão com a Base de Dados

Para que a comunicação das interfaces dinâmicas (produzidas em JSP) com a base de dados seja possível, as firmas que produzem estas tecnologias disponibilizam também a conexão da base de dados para tecnologias compatíveis designadas **conectores.** O conector desenvolvido para a ligação das interfaces JSPs e a base de dados tem a denominação: *Java Data Base Conector* – **JDBC** (Brodbeck, 2004).

JDBC é um programa que permite alcançar a origem dos dados usando para tal os códigos Java. A comunicação de *Java* puro com a base de dados é uma tarefa muito difícil devido a enorme quantidade de bases de dados do mercado com linguagens proprietárias. Por este motivo, os fabricantes da *JavaSoft* resolveram criar uma *interface* entre a linguagem *Java* e uma outra linguagem que todas as bases de dados suportem como é o caso da *Structured Query Languge* - SQL (Jeveaux, 2003).

Abaixo ilustra-se a arquitectura do conector JDBC:

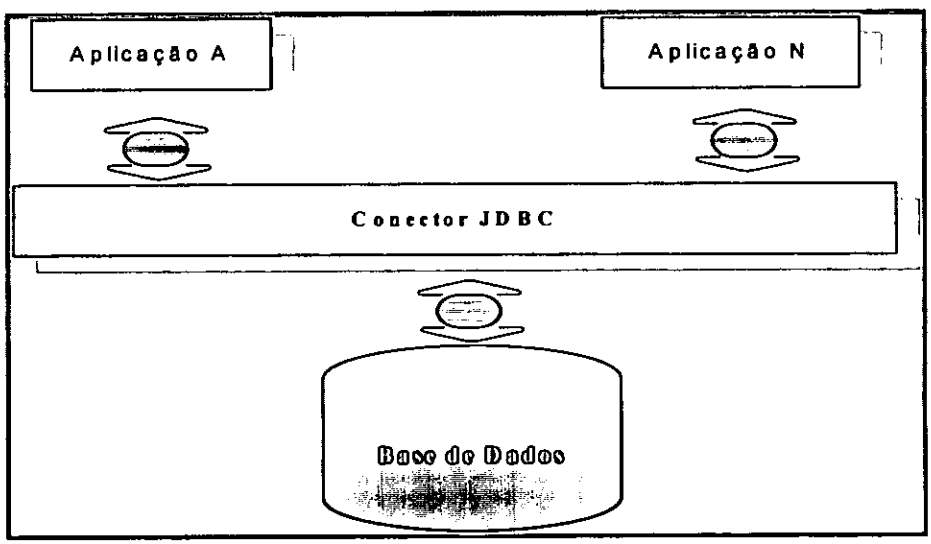


Figura 3 3 - Arquitectura do conector JDBC (Reese, 2001)

Em seguida está apresentada a sintaxe de JSP que segundo Santos (2004) é constituída por

- *Tags de JSP*,
- Objectos Implícitos e
- Acções.

Tags em JSP

Geralmente para construir uma aplicação usando JSP escreve-se o texto HTML e inclui-se códigos JSP entre ***tags*** JSP. Estas normalmente começam com "<%" e terminam com "%>" (Ulisses, 2004).

As *tags* fornecidas pelo JSP estão subdivididas em cinco categorias nomeadamente (ibidem):

1) Directivas;

2) Declarações ;

3) Expressões ;

4) *Scriptlets* e

5) Comentários.

1) Directivas

Directivas são mensagens para o *container* de JSP. Elas não enviam nada para a página mas são importantes para definir atributos JSP e dependências com o *container* de JSP. A forma geral das directivas é a seguinte (Oliveira, 2001):

<%@ Directiva atributo ="valor"%>

As principais directivas que se podem encontrar são:

i. Directiva *page*;

ii. Directiva *include*;

i. Directiva *page* – são atributos usados na tradução e na posterior execução da página (Santos, 2004).

Directiva Page	Descrição
Language	Linguagem de programação a ser utilizada. <%@ page language = "java"%>
Extends	Informa uma *super classe* Java para a classe gerada a partir da página JSP. <%@ page extends = "br.ucb.MyHttpJspBase"%>
.Import	Importa para a página actual o pacote ou a classe especificada. <%@ page import = "java.util.Date"%>
.Session	Informa se a página actual faz ou não parte da sessão actual. <%@ page session = "true"%>
Buffer	Especifica o tamanho do buffer armazenará os dados enviados na resposta. padrão: 8 Kb. <%@ page buffer = "none"%>
.autoFlush	Determina se os dados devem ser enviados automaticamente ao cliente. <%@ page autoFlash = "true"%>
.isThreadSafe	Informa ao container que a página JSP pode lidar com solicitações múltiplas simultaneamente. <%@ page isThreadSafe = "true"%>
Info	Empregado para adicionar informações às páginas. <%@ page info = "Registro de Docentes"%>
.isErrorPage	Indica que a página em questão deve ser apresentada em caso de erro em outra página. <%@ page isErrorPage = "true"%>
.errorPage	URL para a página de erro (definida com isErrorPage) a ser exibida caso ocorra algum erro na página em questão. <%@ page errorPage = "/errors/error.jsp"%>
.contentType	Indica qual o tipo MIME[9] do conteúdo a ser enviado na resposta. <%@ page contentType = "text/html"%>
.pageEncoding	Define o conjunto de caracteres de codificação para a página JSP. <%@ page pageEncoding = "ISSO-8859-I"%>

Figura 3 4 - Directivas Page (Santos, 2004)

[9] MIME – Multipurpose Internet Mail Extensions, fornece o formato para e-mail permitindo mostrar todo o tipo de caracteres no cabeçalho da página (Web7, 2002).

Esta directiva permite a definição dos seguintes atributos:

ii. Diretiva *include*

Esta é usada para incluir outros ficheiros contendo texto ou códigos JSP na página actual (Santos, 2004).

```
<%@ include file = "copyright.html" %>
```

2) Declarações (entre <%! e %>): são usadas para definir variáveis e métodos específicos para uma página JSP. Os métodos e variáveis declaradas podem então ser referenciadas por outros elementos de criação de *scriptlets* na mesma página. A cada declaração deve ser finalizada ou separada por "ponto-e-vírgula" e, pode assumir a seguinte sintaxe (Ulisses, 2004):

```
<%! declaração%>
```

3) Expressões (entre <%= e%>): podem conter alguma expressão válida da linguagem de *script* usada nessa página (o padrão é que a linguagem seja *Java*) mas sem "ponto-e-vírgula". A sintaxe para este elemento de criação de *scripts* é a seguinte (ibidem):

```
<%= expressão%>
```

Para construir uma expressão em JSP pode-se colocar entre as *tags* qualquer expressão definida na especificação da linguagem *Java*. Ao contrário dos *scriptlets* (que veremos a seguir), uma expressão não aceita "ponto e vírgula" e define somente uma expressão da linguagem (ibidem).

4) *Scriptlets* (entre <% e%>): permitem escrever trechos de código da linguagem usada na página. Veja o exemplo abaixo (ibidem):

```
<%
   String option;
   int numberOption = Integer.parseInt(request.getParameter("option"));
   if (numberOption == 1) { option = "CadastrarDocente"; }
   else if (numberOption == 2) { option = "ActualizarDocente"; }
   else { option = "NãoExisteDocente"; }
%>
```

5) Comentários: existem dois tipos principais de comentários que podem ser usados em páginas JSPs (Brandi, 2004):

Comentário de Conteúdo: esses comentários são transmitidos de volta para o navegador como parte da resposta de JSP e, são visíveis na visualização do código da página. O comentário de conteúdo tem a seguinte sintaxe (ibidem):

```
<!– comentário –>\
```

Tais comentários não produzem qualquer *output* visível mas, podem ser visualizados pelo usuário final através do item "*view source*" do navegador.

Comentários JSP: não são enviados para o cliente e são visíveis apenas nos ficheiros fonte JSP originais. O corpo do comentário é ignorado pelo *container* JSP. Os comentários JSP podem assumir duas sintaxes (ibidem):

```
<% – comentário – %>
```

e

```
<% /* comentário */ %>
```

O segundo comentário é introduzido dentro da página através de um *scriptlet*, usando a sintaxe de comentário nativa da linguagem de criação de *scripts*, no caso *Java*.

Objectos Implícitos

São objectos que segundo Santos (2004) estão sempre disponíveis numa página JSP para serem usados em qualquer momento.

Objecto	Classe ou Interface	Descrição
Page	Javax.servlet.jsp.HttpJspPage	Instância de *Servlet* da página
Config	Javax.servlet.ServletConfig	Dados de configuração do *Servlet*
Request	Javax.servlet.http.HttpServletReque-st	Dados de solicitação, incluindo parâmetros
Response	Javax.servlet.http.HttpServletRespo-nse	Dados da resposta
Out	Javax.servlet.jsp.JspWriter	Fluxo de saída para o conteúdo da página
Session	Javax.servlet.http.HttpSession	Dados de sessão específicos do usuário
Applicat-ion	Javax.servlet.ServletContext	Dados compartilhados por todas as páginas da aplicação
pageCon-text	Javax.servlet.jsp.PageContext	Dados de contexto para execução da página
Excepti-on	Javax.lang.Throwable	Erros não capturados ou excepções

Figura 3 5 - Objectos Implícitos (Santos, 2004)

Acções de JSP

Estas geralmente afectam a saída gerada e, usam, criam ou modificam objectos Java (Web5, 2004). Existem tipos distintos de acções[10], sendo ilustradas abaixo algumas consideradas as mais importantes:

[10] Os conceitos referentes a acções seguem a literatura Web5 (2004) sendo as sintaxes das consideradas mais importantes descritas por um outro autor.

i) <jsp:include>: inclui um documento estático ou dinâmico na página actual. Possui dois parâmetros: *page* e *flush,* sendo o primeiro obrigatório (ibidem). A sua sintaxe[11], segundo Jeveaux (2003) é ilustrada abaixo:

```
<jsp:include page="URL relativa| <%= expressão %>"
  flush = "true" />
ou
<jsp:include page="URL relativa| <%= expressão %>"
 flush = "true" >
<jsp:param name="nome do parâmetro"
  value="nome do parâmetro | <%= expressão %>" />
</jsp:include>
```

ii) <jsp:forward>: redirecciona a solicitação para uma outra página HTML ou JSP. A sua sintaxe está expressa abaixo:

```
<jsp:forward page="(URL relativa | <%= expressão %>)"
/>
ou
<jsp:forward page="(URL relativa | <%= expressão %>)"
>
<jsp:param name="nome do parâmetro" value="(valor do
parâmetro| <%= expressão %>)"/>
</ jsp:forward >
```

iii) <jsp:useBean>: localiza ou instancia um JavaBean com um determinado nome e escopo. A sua sintaxe é:

```
<jsp:useBean  id="nome da instancia"
scope="page|request|session|application"
 class="package.class"|
   type="package.class"|
   class="package.class" type="package.class"|
   beanName="package.class | <%= expressão %>"
   type="package.class" /> |
> outros elementos (tags de corpo)
</jsp:useBean>
```

[11] As sintaxes descritas nas acções são da autoria de Jeaveaux (2003)

iv) <jsp:getProperty>: obtém o valor de uma determinada propriedade do JavaBean. A sua sintaxe é:

```
<jsp:getProperty name= "nome do objecto (bean)"
Property="nome da propriedade"/>
```

v) <jsp:setProperty>: altera o valor de uma determinada propriedade do JavaBean.

```
<jsp:setProperty
 name="nome de instância do bean"
 property="*" |
  property= "nome da propriedade" [param="nome do
parâmetro"] |
  property= "nome da propriedade" value="string | <%=
expressão %>"
 />
```

vi) <jsp:plugin>: usado por exemplo quando se deseja executar um *applet*

vii) <jsp:param>: usado para prover informações adicionais para a página que será incluída ou para a qual será redireccionada. Possui dois atributos: *name* e *value*.

viii) <jsp:params>: contém um conjunto de nós filhos formados pela acção *<jsp:param>*. Representa os parâmetros a serem passados para o *plugin*.

xi) <jsp:callback>: funciona como uma estrutura condicional. Pode ser usado, por exemplo, para fornecer uma mensagem caso o *plugin* não possa ser inicializado.

3.4.3 – Vantagens do Uso de JSP

Santos (2004) aponta algumas vantagens do uso das páginas desenvolvidas em JSP que estão listadas abaixo:

- Criação rápida de aplicações *Web;*

- Separação entre a interface (camada de apresentação) e a lógica de aplicação (camada de negócio);
- Independência da plataforma;
- Código aberto;
- Conteúdo dinâmico pode ser embutido em vários formatos: HTML/DHTML, WML, XML, etc.

3.4.4 – Sistema de Gestão de Base de Dados– MySQL

O programa **MySQL** é um servidor robusto de base de dados SQL muito rápido, multi-tarefa e multi-usuário. O Servidor MySQL pode ser usado em sistemas de produção com alta carga e missão crítica bem como, ser embutido em programa de uso de várias transacções ao mesmo tempo (MySQL, 2004).

MySQL é a mais popular base de dados *open source* do mercado. Fornece recursos simplificados e apropriados para as suas aplicações, tendo um custo extremamente reduzido.

Possui ferramentas para a maioria das exigências necessárias para uma aplicação de base de dados corporativa, fornecendo uma arquitectura extremamente rápida e de fácil utilização (Welling, 2001).

Algumas Vantagens do MySQL (Katalogo, 2004):

Confiabilidade e Desempenho:
MySQL fornece recursos eficientes para todos os seus *softwares* de base de dados, permitindo ser testado antes que seja adquirido esse produto.

Facilidade de Utilização e Distribuição:
A arquitectura do MySQL fornece recursos customizados de fácil e rápida utilização, sua arquitectura única de multi-processamento permite uma maior flexibilidade para gestão de base de dados, possibilitando uma maior velocidade, estabilidade, e facilidade de distribuição.

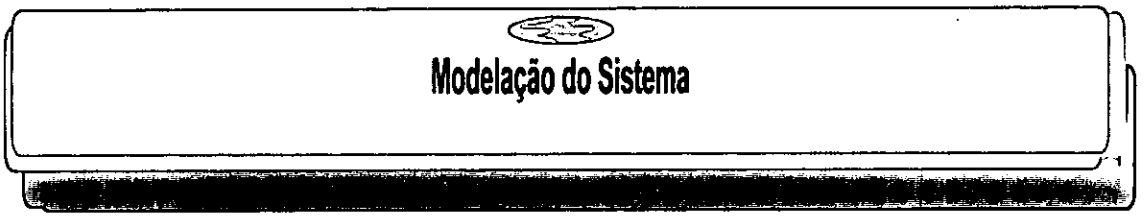
Recursos multi-plataforma:

Fornece um completo acesso ao código fonte, assegurando uma maior liberdade para as diversas plataformas existentes.

Suporte para uma série de plataformas:

MySQL está disponível para mais de 20 plataformas diferentes, incluindo as plataformas Linux, MacOS X, UNIX e Microsoft Windows.

Capítulo IV



4 – MODELAÇÃO DO SISTEMA

Para a descrição dos processos do Sistema de Gestão de Docentes do DMI durante a fase de modelação, seguiram-se notações padronizadas da linguagem de modelação UML usando o processo *Rational Unified Process* - RUP. O presente capítulo ilustra os processos deste sistema em forma de diagramas, dando alguns conceitos ligados a metodologia usada no contexto do modelo proposto.

4.1 – Fases de Desenvolvimento de um Sistema em UML

De acordo com McFadden et al (1999), existem cinco fases no desenvolvimento de sistemas de *software*: .

- **Análise de requisitos** – onde é feita a captura das necessidades dos usuários do sistema, através de processos denominados *use cases,* abaixo citados. Destas necessidades básicas, são extraídos os requisitos funcionais e não funcionais sendo ilustrados no diagrama de *use case;*
- **Análise** – esta fase está relacionada com as primeiras abstracções (classes e objectos) e mecanismos que estarão presentes no domínio do problema. As classes são modeladas e ligadas a outras classes através de relacionamentos e, são descritas no diagrama de classes (descrito ao longo deste capítulo). A análise modela o problema principal (classes, objectos) e cria um modelo ideal do sistema sem levar em conta os requisitos técnicos do sistema;
- **Desenho** – esta fase reflecte soluções técnicas resultantes da fase de análise. Nesta, adicionais classes são introduzidas no modelo de forma a providenciar uma infra-estrutura técnica, como por exemplo: a *interface* para o usuário, a gestão da base de dados dentre outras, tornando possível alterar tanto o domínio do problema como a infra-estrutura das classes. Esta fase expande e adapta os modelos de análise para um ambiente técnico com soluções técnicas especificamente detalhadas.
- **Programação** – nesta fase as classes obtidas no desenho são convertidas para o código da linguagem de programação OO. A conversão do desenho para código pode ser um processo fácil ou moroso dependendo da linguagem escolhida. Esta é uma fase separada e distinta onde os modelos criados são convertidos em código.

- **Testes** – um sistema pode ser submetido a diversos testes em diversos estágios de desenvolvimento de modo a verificar se o sistema a ser desenvolvido responde na íntegra as necessidades dos usuários. Estes testes podem ser subdividido nos seguintes grupos:
  - ***Testes de Unidade*** – realizados geralmente pelos programadores em que são testadas classes individuais ou grupos de classes;
  - ***Testes de Integração*** – são efectuados usando as classes e componentes integrados para se confirmar se as classes cooperam uma com as outras como foi especificado no modelo;
  - ***Testes de Aceitação*** - observam o sistema como uma caixa preta e verificam se o sistema está a funcionar como foi especificado nos primeiros diagramas de *use cases*.

4.2 – Visões de UML

Bezerra (2003) define **UML** como uma linguagem visual para modelar sistemas OO ou seja, uma linguagem constituída de elementos gráficos (visuais) usados na modelação que permitem representar os conceitos do paradigma da OO. Através dos elementos gráficos definidos nesta linguagem pode-se construir diagramas que representam diversas perspectivas de um sistema.

Cada um destes diagramas fornece uma perspectiva parcial sobre o sistema sendo modelado, consistente com as demais perspectivas (ibidem). Porém, nem sempre é necessário ilustrá-los todos. Esta depende do nível de complexidade, necessidade e dimensão do sistema a ser construído (Barros, 1998).

Segundo Neto (2001), os diagramas de UML subdividem-se nas seguintes visões:

Estrutural:

- **Diagrama de Classes** – dá uma visão estática do sistema. Exibe um conjunto de classes, interfaces e seus relacionamentos. As classes especificam a estrutura e o comportamento dos objectos (que são instâncias de classes) (ibidem);
- **Diagrama de Objectos** – exibe um cenário estático. Representam instâncias estáticas de elementos dos diagramas de classes (ibidem).

Comportamental:

- **Diagrama de *Use Case*** – organiza e modela o comportamento do sistema. Mostra a funcionalidade que o sistema disponibiliza e, quais os usuários que se comunicarão com o sistema em alguns casos para usar esta funcionalidade (McFadden, 1999);
- **Diagrama de Interacção** – ilustra a implementação dinâmica do sistema. Estes subdividem-se em dois tipos (Neto, 2001):
  - **Diagrama de Sequência** – é aquele que enfatiza o ordenamento das mensagens;
  - **Diagrama de Colaboração** - este enfatiza a organização estrutural dos objectos que trocam mensagens.
- **Diagrama de Estados** – é utilizado para modelar o comportamento dos objectos (Nunes, 2001);
- **Diagrama de Actividades** – é um tipo especial do diagrama de estados em que são representados os estados duma actividade em vez dos estados de um objecto (Bezerra, 2003).

Arquitectural:

- **Diagrama de Componentes** – ilustra a implementação estática do sistema mostrando um conjunto de componentes e, como se relacionam em termos de *software* (Neto, 2001);
- **Diagrama de Instalação** – modela aspectos físicos (relacionados com *hardware*) ou seja, o ambiente em que o sistema será executado (ibidem).

Abaixo estão representados a maioria dos diagramas citados, com excepção de alguns que possuem a mesma visão que outros e não diferem muito nas suas funcionalidades, como o Diagrama de Objectos. Os diagramas que representam a parte arquitectural não serão descritos por especificarem a implementação do sistema. Os estudos para a instalação deste sistema, não estão previstos nesta fase.

4.3 – *Rational Unified Process*

A UML oferece semântica e notação em padrões para descrever a estrutura e comportamento de objectos, tendo surgido como o meio de opção de *design* para desenvolver aplicações de objecto distribuídos em larga escala. Ampliada pelo *Rational Unified Process - RUP*, por um extenso conjunto de directrizes de desenvolvimento de *software* e pela ferramenta de modelagem *Rational*

Rose, a UML facilita deste modo o desenvolvimento de aplicações de qualidade baseados em objectos, que correspondem tanto aos prazos quando às exigências (Terry, 2001).

De acordo com Franklin (2004), a *Rational* é conhecida pelo seu investimento em OO. A empresa foi a criadora da UML, assim como de várias ferramentas que a suportam.

RUP é constituída por processos unificados completos criados pela *Rational* para viabilizar que grandes projectos de *software* sejam bem sucedidos. Este é um produto composto de material de referência na forma de páginas HTML, descrevendo toda a metodologia (ibidem).

Princípios Básicos

Um grande problema nos projectos actuais é o grande dinamismo e complexidade dos negócios nos dias de hoje. Cada vez mais os sistemas são complexos e, precisam estar prontos em menos tempo. Mais do que isso, as necessidades mudam ao longo do tempo e a especificação de um sistema provavelmente será alterada durante seu desenvolvimento. Além disso, temos tecnologias novas (software e hardware) surgindo a cada dia. Algumas funcionam bem, outras não. Visando atacar estes problemas, RUP adopta as seguintes premissas básicas (ibidem):

- Uso de interacções para evitar o impacto de mudanças no projecto;
- Gestão de mudanças e,
- Abordagens dos pontos de maior risco o mais cedo possível.

RUP suporta diversas práticas de desenvolvimento de *software* tais como (Reis, 2004):

- Processo de desenvolvimento de *software* interactivo;
- Propõe a gestão integrada de requisitos desde a sua identificação até à implementação;
- Propõe o desenvolvimento de *software* baseado em arquitecturas de *software* e em componentes;
- Defende a modelação visual;
- Controle de qualidade é permanente.

4.4 – Diagrama de Use Case

Os **diagramas de *Use Cases*** constituem a técnica em UML que serve para representar o levantamento de requisitos[12] de um sistema e, para assegurar que tanto o utilizador final como o perito numa determinada área possua um entendimento comum dos requisitos (Nunes, 2001).

O seu objectivo é mostrar o que um sistema deve efectuar e não como irá funcionar. Desde sempre que o correcto levantamento de requisitos de desenvolvimento de sistemas de informação, tenta garantir que este será útil para o utilizador final, estando de acordo com as suas necessidades (ibidem).

Estes diagramas utilizam as seguintes abstracções de modelação:

- **Actores** – representam utilizadores que interagem com o sistema (ibidem);
- ***Use Cases*** – é a especificação de uma sequência de interacções entre um sistema e os agentes externos que utilizam este sistema (Bezerra, 2003);
- **Relações** (*Uses*, *Extends* e Generalização) – representam a informação de quais os actores estão associados a que casos de uso (ibidem);

Os actores identificados no modelo proposto são:

- Docente;
- Chefe de Secção;
- Director do Curso.

Um actor pode assumir vários papéis ao mesmo tempo. Por exemplo, o Docente pode também ser Chefe da Secção ou ainda o director de um dos cursos do DMI. Em geral, um actor pode invocar vários *use cases* e um *use case* pode ser invocado por vários actores.

Segundo Nunes (2001), os *use cases* podem ser definidos numa perspectiva de Negócio ou de Sistema. Na primeira perspectiva, tenta-se identificar a forma em termos de processo de negócio, se responde a um cliente ou evento. Na perspectiva do Sistema, procura-se identificar as interacções com a aplicação a desenvolver (*software*).

[12] **Requisito** num sistema é uma funcionalidade ou característica considerada relevante na óptica do utilizador. Normalmente representa o comportamento esperado do sistema, que na prática consiste num serviço que deve ser disponibilizado a um utilizador Nunes (2001).

A principal razão para esta distinção (por vezes pouco nítida) pretende-se com o facto de nem todos os *use cases* (processos) de negócio virem a ser suportados pelo sistema informático. Além disso, ao focalizar apenas o sistema, podemos estar a esquecer que uma alteração no processo de negócio poderá ser a solução mais correcta (eficiente e eficaz) (Nunes, 2001).

Após a identificação dos actores deve-se identificar para cada actor os *use cases* em que este interage com o sistema, identificados na tabela seguinte:

Actores e Use Cases

Actor	Use Cases
Docente	• Introduzir Actividades Científicas • Introduzir Material Instrutivo • Actualizar Material Instrutivo • Actualiza Actividade Científica
Director do Curso	• Consultar Dados Docente
Chefe da Secção	• Introduzir Docente • Alterar Dados Docente • Consultar Dados Docente • Remover Dados Docente • Introduzir Actividades Científicas • Remover Actividades Científicas • Listar Actividades Científicas • Listar Materiais Instrutivos

Figura 4 1 - Representação da Relação actor-use case

Cada um dos *use cases* identificados deve ser detalhado ou descrito em termos de **cenários**[13] **de utilização** que foram descritos em anexo (Anexo A).

A figura abaixo ilustra a comunicação entre os *use cases* e actores através do Diagrama de *Use Cases* do Sistema de Gestão de Docentes do DMI.

[13] **Cenários** - são os possíveis caminhos seguidos dentro do *use case*, de forma a fornecer ao actor uma resposta (Nunes, 2001).

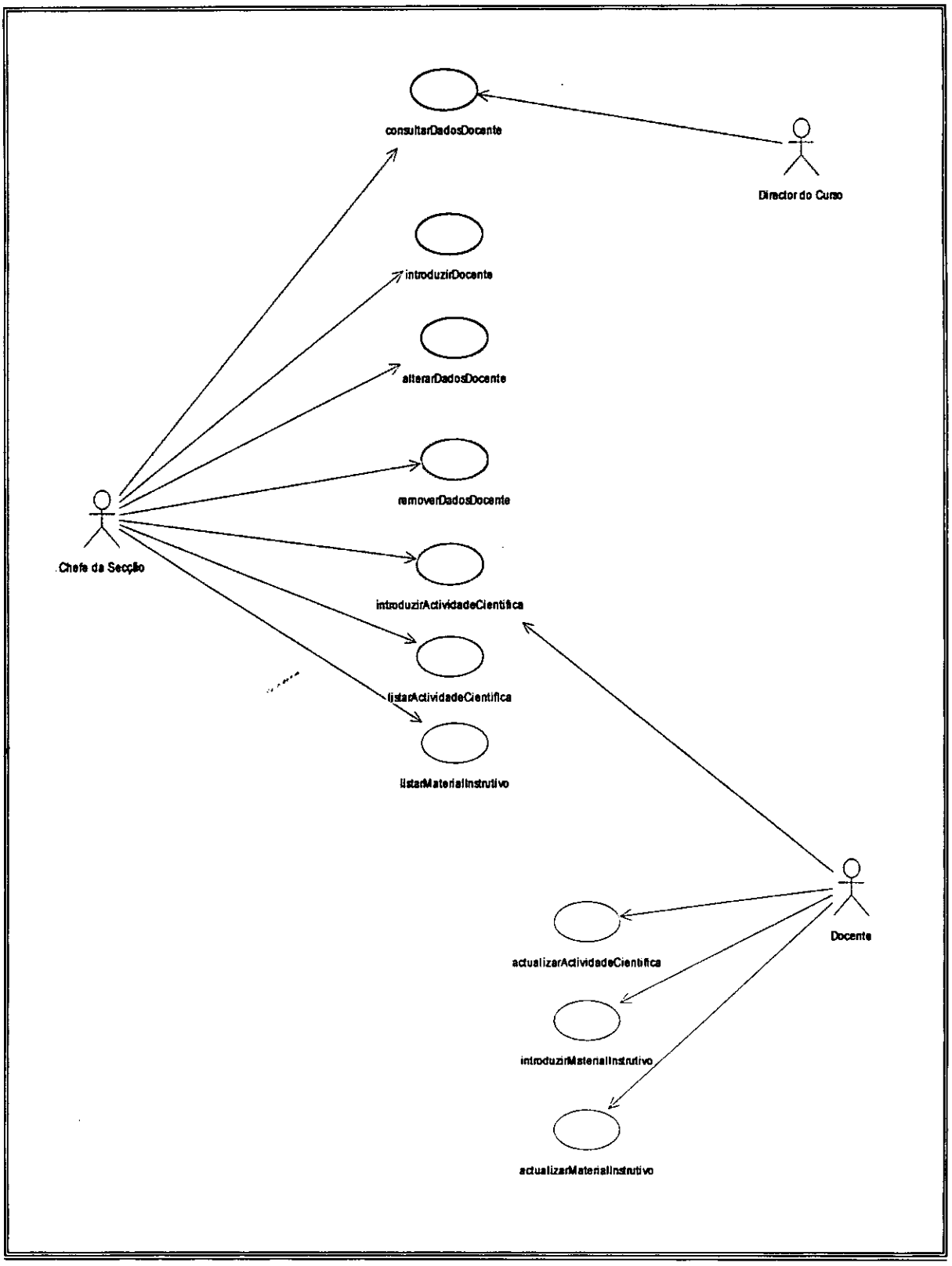


Figura 4 2 - Diagrama de Use Case do Sistema de Gestão de Docentes do DMI

4.5 – Diagramas de Interacção

Os **Diagramas de Interacção**[14] são utilizados na UML para modelar os aspectos dinâmicos do sistema em termos dos objectos e suas relações, com base nas mensagens trocadas entre objectos (Nunes, 2001).

Abaixo exemplifica-se o diagrama de colaboração e sequência para o caso do cadastro de um docente, respectivamente nas figuras 4.3 e 4.4. Em anexo (Anexo B), apresenta-se o uso destes dois diagramas para outros processos do sistema, como é o caso da remoção de um docente ou introdução de material instrutivo ou ainda de uma actividade científica ou outros.

Diagrama de Colaboração de Cadastro de Docente

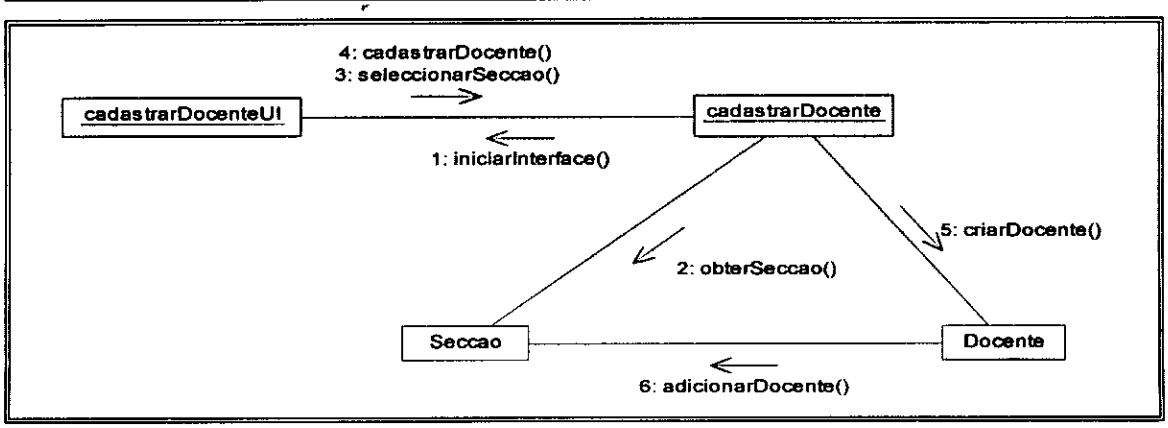


Figura 4 3 - Diagrama de Colaboração de Cadastro de Docente

A partir deste surgem as classes Docente e Secção como se pode verificar abaixo:

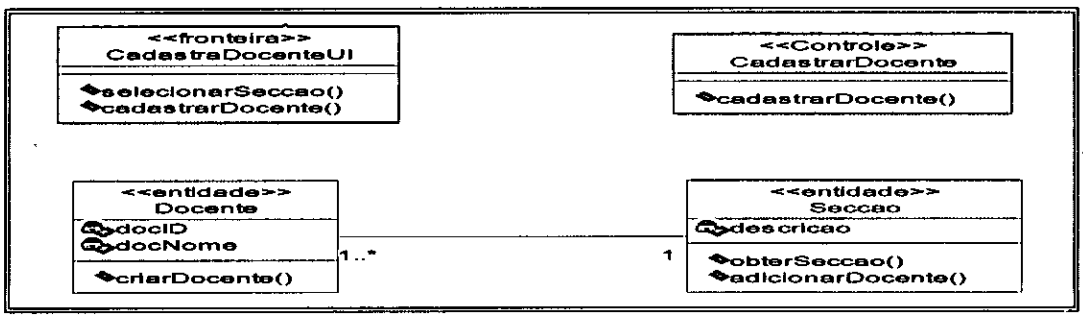


Figura 4 4 - Surgimento da Classe Docente e Secção

[14] Interacção – segundo Booch et al (1999) é o comportamento que consiste na troca de um conjunto de mensagens entre objectos, dentro de um contexto, para atingir um objectivo.

Diagrama de Sequência do Cadastro de Docente

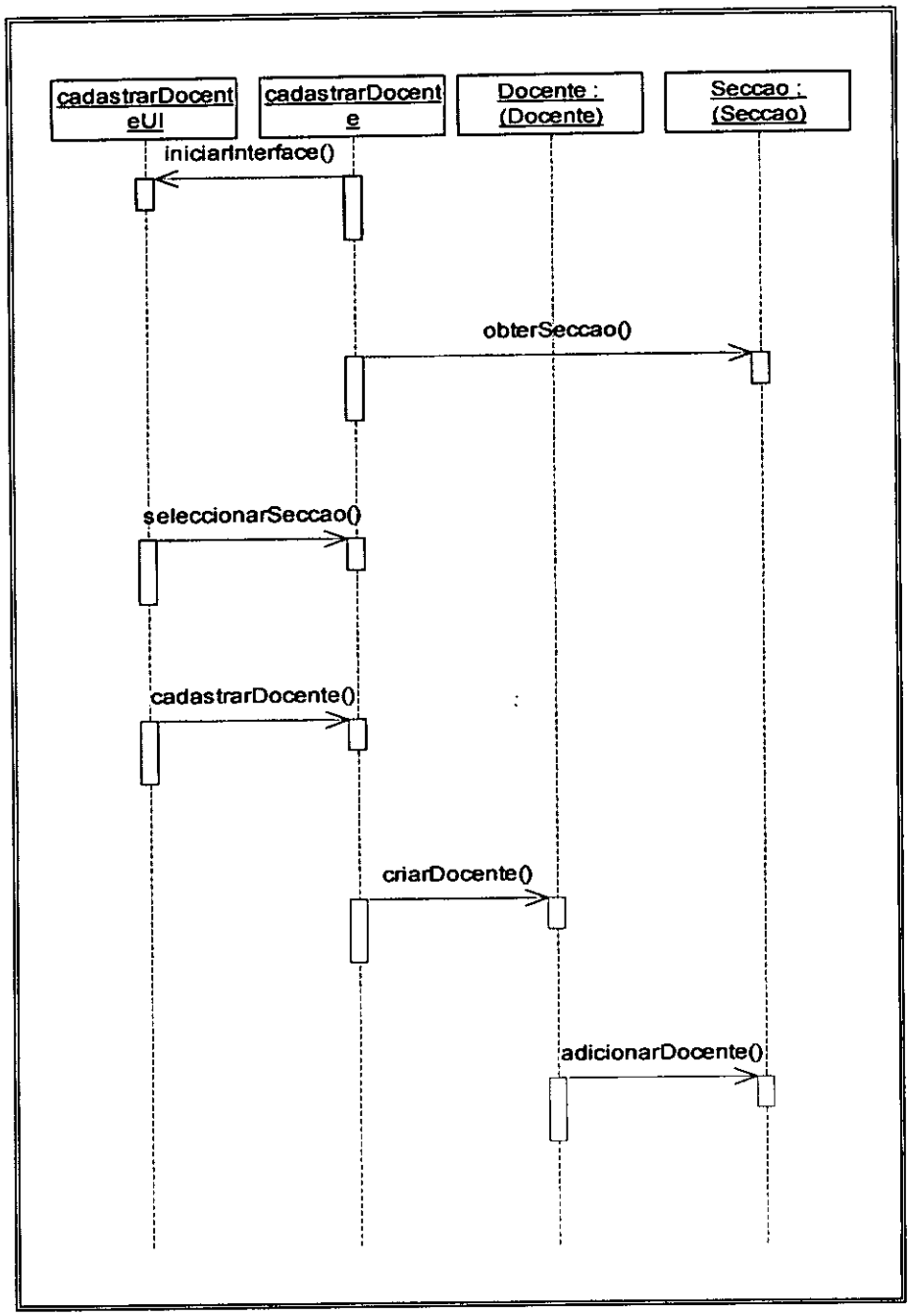


Figura 4 5 - Diagrama de Sequências do Cadastro de Docente

4.6 – Diagrama de Classes

Um **Diagrama de Classes** demonstra a estrutura estática de um sistema onde estas representam as "coisas" que são geridas pela aplicação modelada (Pablo, 2000).

Uma classe num diagrama pode ser directamente implementada utilizando-se uma linguagem de programação OO, que tenha suporte directo para construção de classes. Para criar um diagrama de classes, as classes têm que estar identificadas, descritas e relacionadas entre si (ibidem).

As associações entre as classes e o diagrama de classes na fase de análise estão representados em anexo (Anexo C) sendo ilustrado a seguir o Digrama de Classes da fase de Desenho.

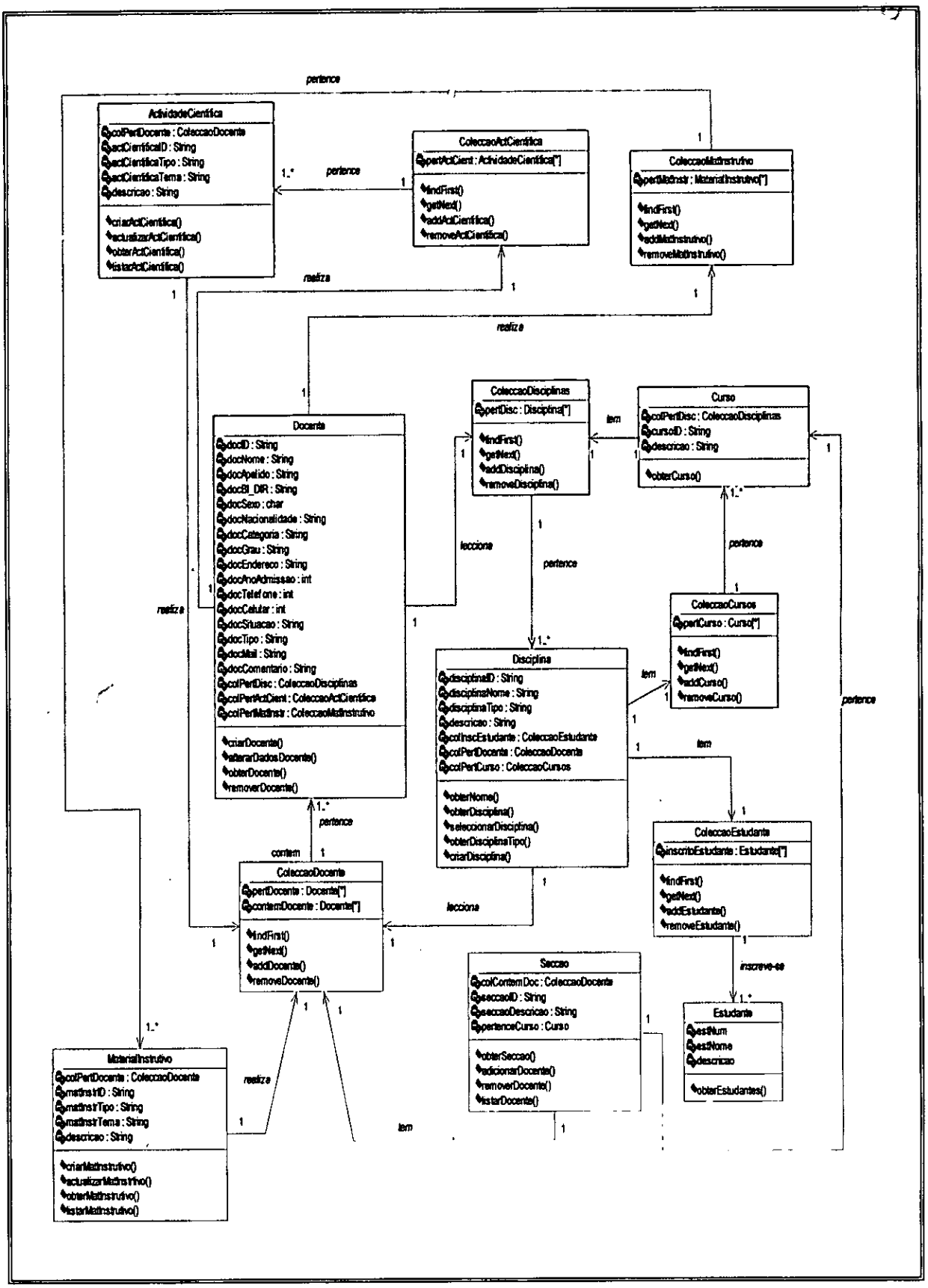


Figura 4 6 - Diagrama de Classes da Fase de Desenho

4.7 – Diagrama de Actividades

Um **Diagrama de Actividades** pode ser considerado como caso especial de diagrama de estados, em que quase todos os estados são acções de estados e quase todas as transições evoluem com o término dessa acção. Permite representar transições internas em margens das transições externas (Vilas, 2001).

Um diagrama de actividades permite expressar a ordem em que as coisas se realizam, por isso é adequado para modelação de organizações e de programas concorrentes[15] (ibidem).

De acordo com Nunes (2001) o Diagrama de Actividades é particularmente útil quando se pretende detalhar um *use case* associado a um processo de negócio[16]

Um diagrama de actividades é constituído pelos seguintes elementos de modelação:

- **Linhas verticais de responsabilidade** – descrevem quais os objectos responsáveis por cada uma das actividades (Nunes, 2001);
- **Actividades de início e de fim** – a inicial poderá ser puramente virtual, usada para identificar o início do diagrama, ou corresponder a uma actividade operacional do sistema e é descrita por um círculo preto. A final é usada para identificar o término de um fluxo, e é representada por um círculo preto delimitado por uma linha de circunferência (ibidem);
- **Actividade intermédia;**
- **Transição de actividade e símbolos de comportamento condicional** – a transição de actividades permite descrever a sequência pela qual as actividades se realizam, e é representada por uma seta. E o comportamento condicional é descrito pelos caminhos alternativos que podem existir num fluxo de actividades. Para a sua representação utilizam-se "guardas"[17] e diamantes de decisão em forma de diamantes e descrevem uma convergência ou divergência no fluxo de controlo (ibidem).

[15] Programas concorrentes – Segundo Maia (1999), são programas escritos param sistemas concorrentes, onde programas concorrentes são aqueles que exibem concorrência. Sobre concorrência entende-se como sendo aqueles aspectos fundamentais de sistemas com agentes múltiplos e simultâneos de computação, os quais interagem uns com os outros

[16] Processo de Negócio – Nunes (2001) define como sendo o conjunto integrado de actividades de uma organização que procura satisfazer um determinado objectivo e no qual participam um ou mais actores.

[17] Guardas – Segundo Nunes (2001) são expressões booleanas limitadas por parênteses rectos [ ], que têm de ser verificadas para se realizar a transição para uma nova actividade.

A figura abaixo ilustra o processo de cadastro de um docente através de um diagrama de actividades:

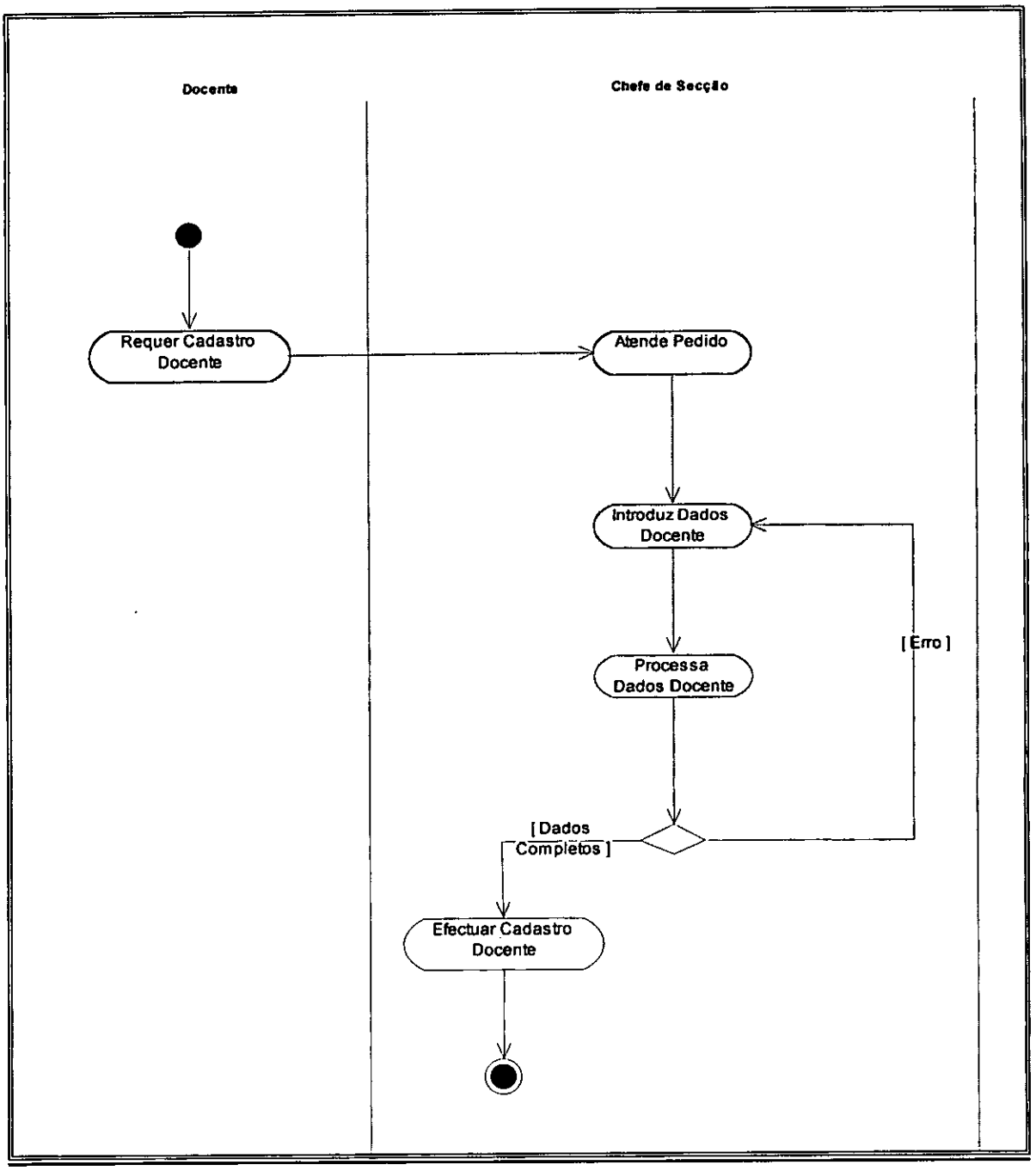


Figura 4 7 - Diagrama de Actividades de Cadastro de Docente

4.8 – Desenho de Base de Dados

De uma forma genérica, qualquer conjunto de dados é uma **Base de Dados** (BD), ou seja, uma agenda com os endereços de pessoas conhecidas, um livro, os dados guardados nos computadores e a *World Wide Web* (Brazdil et al, 2004).

Segundo Bennet et al, (2002), existem 3 tipos de abordagens de base de dados, a relacional, a orientada a objectos e a híbrida objecto – relacional.

- **Base de dados relacional** - é uma colecção de **dados** estruturados em tabelas devidamente relacionadas através de registos que possuem campos com informação comum (web1, 2003);
- **Base de dados orientada a objectos** – aquela capaz de armazenar objectos com toda sua estrutura complexa. Não é necessário transformar as classes do modelo do sistema a fim de armazena-las em objectos da base de dados OO. Mantém similaridade para sistemas OO, que permitem fazer o armazenamento directo dos objectos na base de dados (Bennet et al, 2002).
- **Híbrida objecto-relacional** – é uma combinação da base de dados relacional com a OO (ibidem).

Uma maneira alternativa a estas abordagens é o **mapeamento de classes para tabelas** que segue um conjunto de regras listadas em anexo (ANEXO E) para mapear as classes e multiplicidades em tabelas de uma base de dados relacional (ibidem).

- Classes com uma simples estrutura de dados são transformadas em tabelas;
- Identificadores de objectos tornam-se chaves primárias - um único identificador é gerado para todos os objectos e pode ser usado como chave primária na tabela relacional a que ela pertence;
- Classes que contém uma instância de outra classe como atributo - uma tabela separada deve ser criada para a classe embutida. Objectos da classe embutida devem ser alocados a um único identificador. O identificador deve substituir o objecto embutido na tabela da classe que contém a colecção, como chave estrangeira;
- Classes que contém colecções - deve ser alocado um identificador de objecto à classe contida na colecção. Esta classe será representada por uma tabela. Deve ser criada uma

tabela que contém duas colunas, a primeira com o identificador dos objectos que contém a colecção, a segunda com o identificador dos objectos contidos na colecção;

- Associações de um-para-muitos podem ser tratadas como colecções;
- Associações de muitos-para-muitos tornam-se tabelas separadas. Cria-se uma tabela com duas colunas. Cada linha contém um par de identificadores de atributos, um de cada objecto participante na associação;
- Associações de um-para-um são implementadas como chaves estrangeiras. Cada classe ganha um atributo extra para carregar o identificador do objecto associado.

Para identificação das tabelas usam-se chaves, que são os atributos responsáveis pela identificação destas. As chaves podem ser primárias ou secundárias.

Uma **Chave Primária** é definida como um conjunto "k" de atributos verificando: unicidade (os valores de chave Primária são únicos e não nulos) e minimalidade (nenhum atributo componente de k pode ser retirado sem perda da unicidade) (Rodrigues, 2004).

A **Secundária** é usada em base de dados relacional para criar relações entre tabelas. Porque os objectos não têm chaves então, são alocados identificadores para eles. Quando se usa uma base de dados relacional, as colecções de classes que existem só para aceder a um conjunto de objectos da mesma classe não precisa fazer parte dos mesmos dados que são guardados nas tabelas (Bennet et al, 2002).

O modelo proposto usa SGBD relacional com mapeamento de tabelas. A Base de Dados resultante deste mapeamento é descrita na figura a seguir:

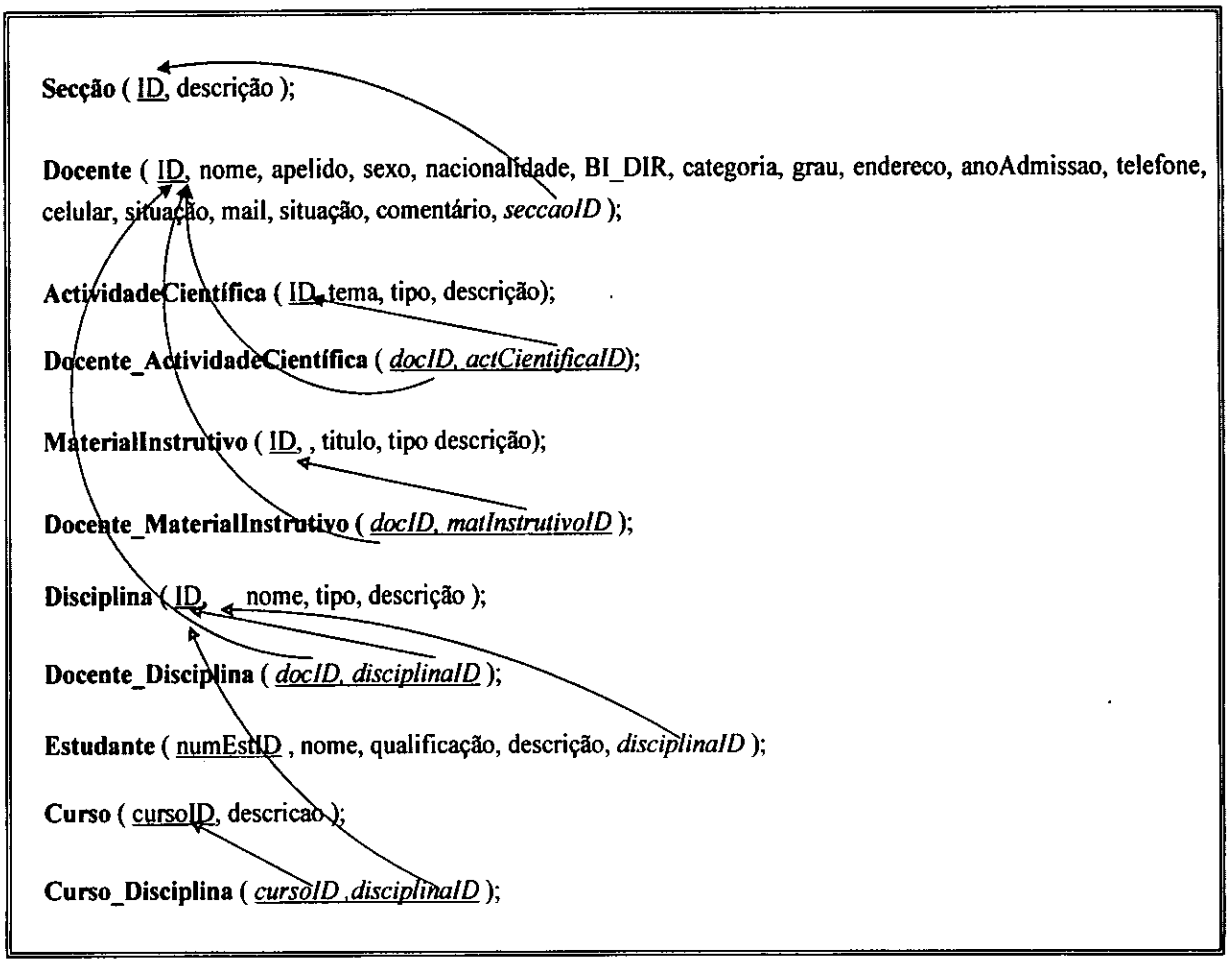


Figura 4 8 - Modelo Relacional de Base de Dados com Mapeamento de Tabelas

4.9 – Arquitectura de Implementação Usada

De acordo com Morisseau-Leroy (2003), a arquitectura dum modelo de programação JSP depende das necessidades da aplicação a ser desenvolvida. Porém, como os requisitos da maioria das aplicações evolui com o tempo, aconselha-se o uso duma arquitectura que permita o aprimoramento modular e flexível da funcionalidade no futuro, embora ela possa parecer demasiadamente complexa para as suas necessidades.

O Sistema de Gestão de Docentes do DMI usou a tecnologia JSP para produzir conteúdo dinâmico das páginas *Web*. Este processo é possível usando código Java embutido nas páginas JSP, os chamados *Scriptlets* (tendo se detalhado o seu estudo no capítulo 3).

Os principais inconvenientes desta técnica são os seguintes (Web6, 2004):

- *Scriptlets* misturam lógica com apresentação,
- *Scriptlets* quebram a separação de papeis desenvolvedor/*Web designer*,
- Páginas JSP ficam mais difíceis de ler e manter com a presença de *scriptlets*

Para contornar estas situações indesejadas, a aplicação desenvolvida usa uma extensão da tecnologia JSP que habilita a criação de novos *tags*, ou seja usa ***Struts***[18]. Estes contêm muitos *tags* juntos apresentados em livrarias com vista a reduzir o número de *Scriptlets* no JSP.

Abaixo está ilustrada a arquitectura de *Struts* mostrando como este faz parte da arquitectura J2EE:

[18] ***Struts*** = Esteio ou Escora, ou seja é um suporte para construir aplicações Web (Web6, 2004).

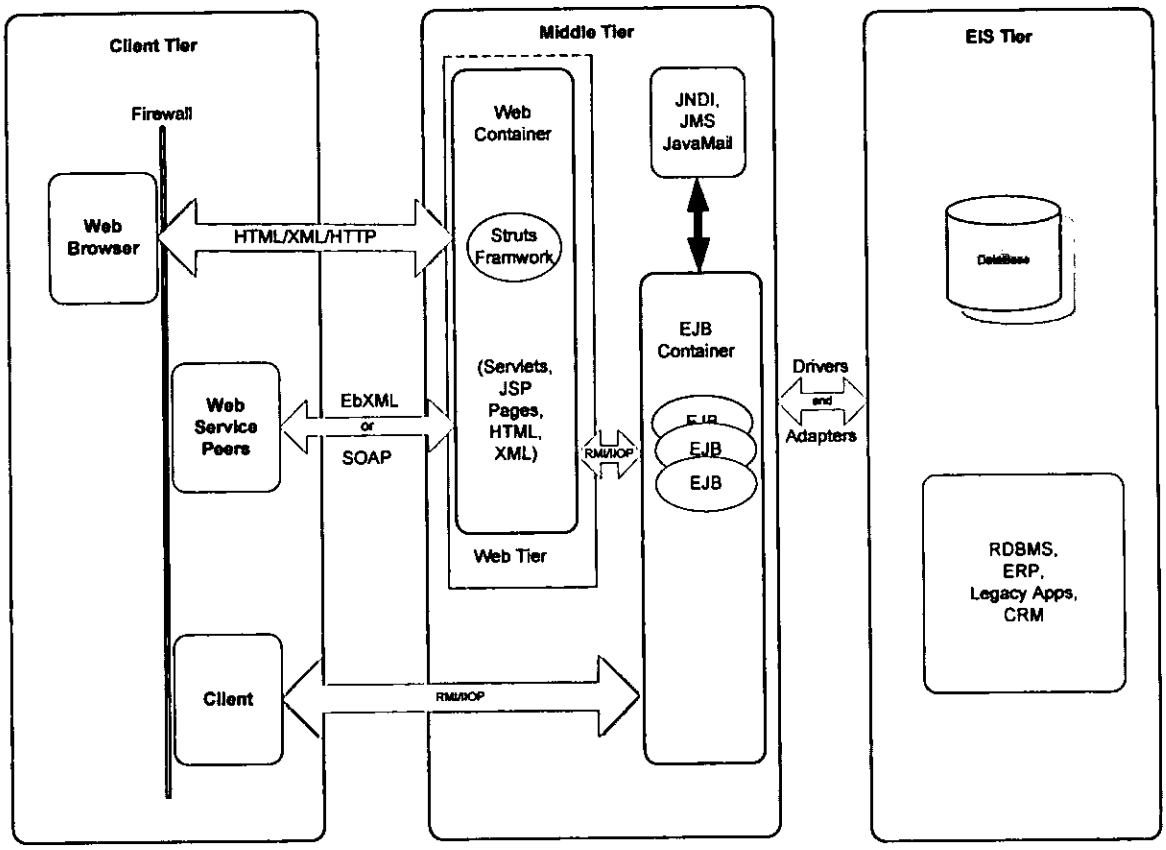


Figura 4 9 - Estrutura de Struts dentro da Arquitectura J2EE (Web6, 2004)

Os *Struts* devem ser enquadrados numa arquitectura específica na fase de implementação. No caso de JSP, a arquitectura pode apresentar uma:

- **Estrutura Simples** – esta arquitectura consiste num JSP que gere uma conexão de base de dados para um determinado serviço. É uma técnica fácil de programar, porém é adequada apenas para aplicações pequenas tendo a desvantagem de o código poder ficar difícil manejar dificultando assim a manutenção (Morisseau-Leroy, 2001);
- **Estrutura baseada em Componentes** – usa componentes *beans* (ou outros componentes dependendo das combinações a serem efectuadas) para encapsular a lógica dinâmica. Essa separação permite decompor organizadamente o código *Java* do código HTML e torna a manutenção mais fácil. Entretanto, essa estratégia tem o problema de congestionamento da JSP para a lógica de pesquisa e tratamento de excepções (ibidem);
- **Estrutura *Model View Controller* – MVC** – esta arquitectura foi desenhada para cenários mais complexos que exijam por exemplo a autenticação dos usuários. Nesses casos, é aconselhável dividir a lógica de aplicação do "*front end*"(*interface*) num JSP ou *servlet* separado que não faça apresentação. Esse programa intercepta pedidos do cliente e

normalmente inicializa componentes *Java Beans* que manipulam a apresentação de dados computados. Os componentes *Java Beans* representam o "**modelo**" e o JSP ou *servlet* do *front end* actua como o "**controlador**" para o processamento de pedidos (Morisseau-Leroy, 2001).

O modelo proposto segue a arquitectura MVC, por esta se apresentar como sendo a mais modular dos três modelos, e usa *Struts* para reduzir o volume de código Java encapsulado nas páginas JSPs e, facilitar algumas manipulações como por exemplo, a validação dos dados introduzidos através dos formulários do sistema. Estes formulários, bem como a descrição do funcionamento do modelo proposto são descritos por meio de um manual do utilizador (Anexo G).

Abaixo ilustra-se a arquitectura MVC:

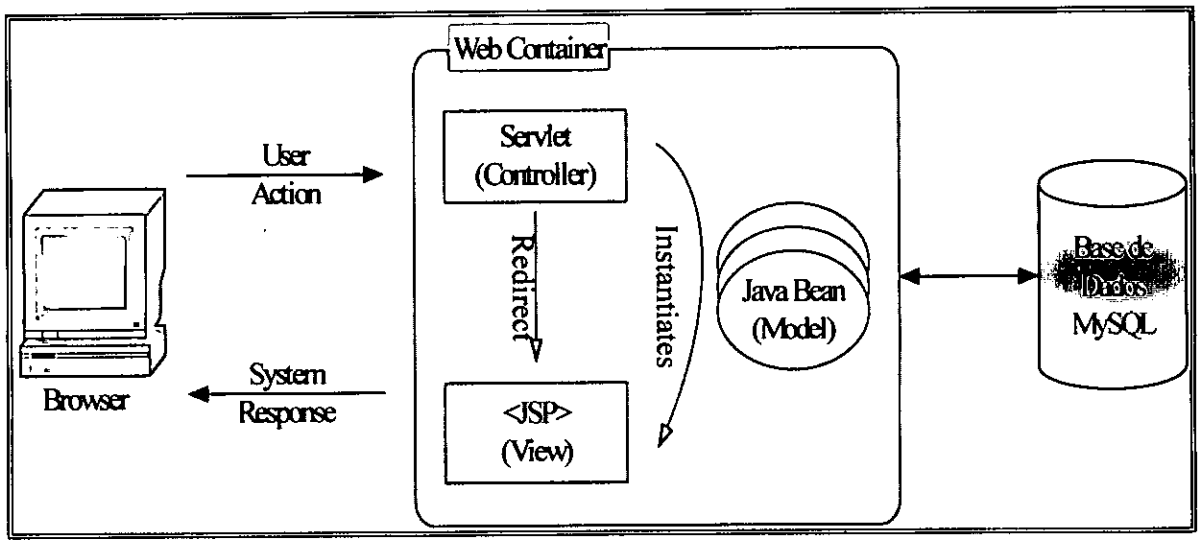


Figura 4 10 - Arquitectura de Implementação MVC

O sistema desenvolvido usa a tecnologia *Web* fazendo com que o usuário possa ter uma página *Web* (estática ou dinâmica) como interface para introdução de dados, que serão validados e posteriormente armazenados numa base de dados produzida em MySQL, de forma a facilitar e melhorar as transacções e o acesso à informação a ser disponibilizada.

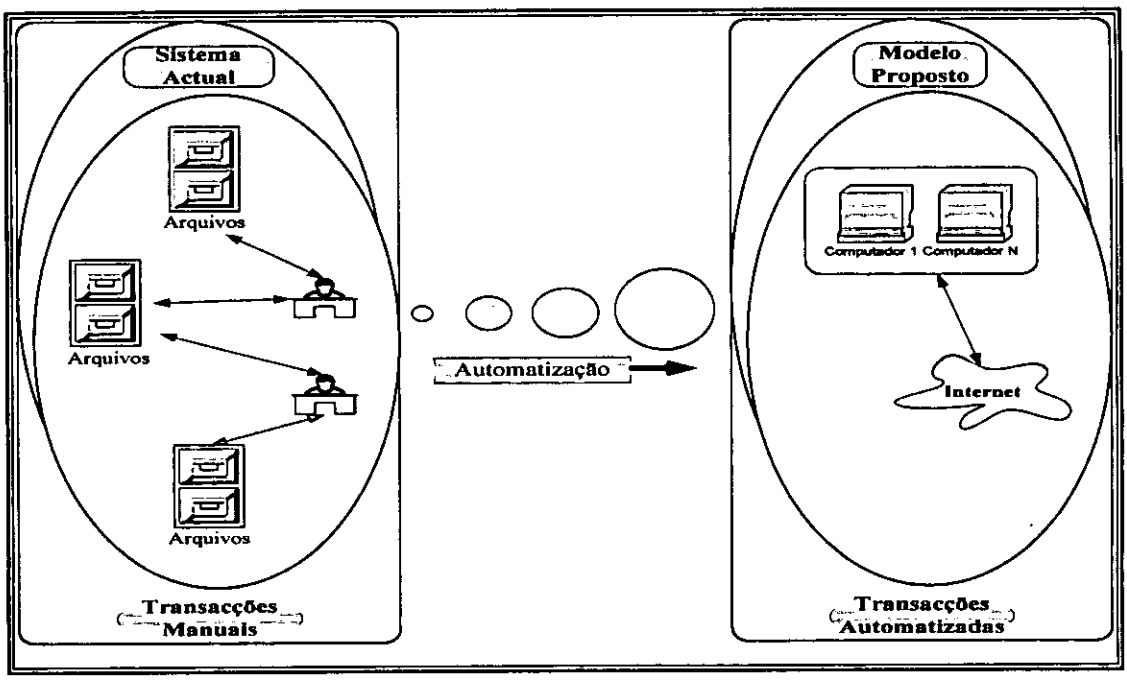


Figura 4 11 - Processo de Automatização do Sistema de Gestão de Docentes do DMI

A transição do sistema actual para o modelo proposto foi concebida a partir da implementação das TICs, de forma a transformar as transacções realizadas ou relacionadas com docente de forma manual em automatizada, não contudo tendo alterada a estrutura organizacional e funcional das entidades que fazem parte do sistema.

O modelo proposto como ilustrado na figura abaixo, é constituído por dois servidores sendo um local a ser colocado no DMI e, um servidor *Web* cujo tráfego é controlado por um firewall conectado à *Internet*. Estes dois servidores estão ligados a base de dados produzida em MySQL.

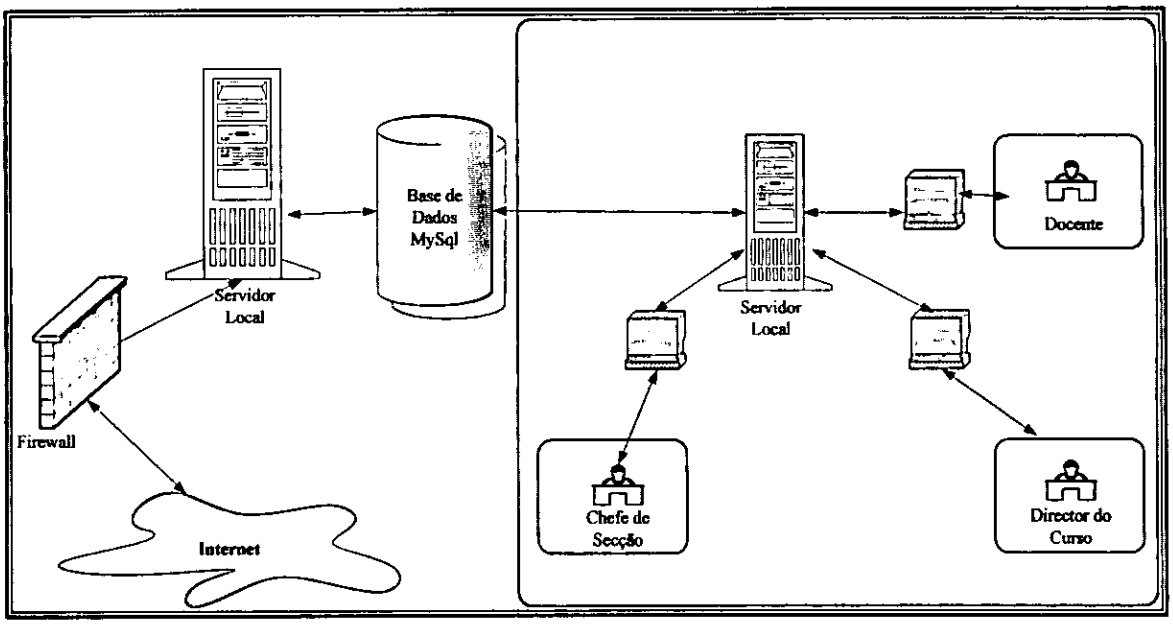


Figura 4 12 - Modelo Proposto

Em termos de equipamento de suporte para o desenvolvimento do Sistema Proposto foram usados:

- Servidor *Web* que suporta a aplicação;
- Base de dados para armazenar os dados;
- *Browser*, para testar as páginas;
- Ferramentas para edição de páginas estáticas e dinâmicas.

Para o suporte das páginas e sua divulgação na rede mundial (*Internet*) é necessário o uso de um **Servidor *Web***[19]. O servidor *Web* usado para suportar a aplicação desenvolvida foi o *Apache Tomcat 5* (este foi descrito no capítulo que aborda o ambiente *Web*). Porém qualquer outro servidor *Web* compatível com a tecnologia usada podia ter sido usado, tais como:

- JRun;
- JBOSS;
- *JavaServer*;

Para a **base de dados**, como vimos anteriormente, foi usada o SGBD MySQL, visto esta ser rápida em aplicações *Web* e não exigir grande suporte em termos de *hardware*

[19] *"Um servidor Web é um programa que recebe pedidos que chegam através de uma rede, executa alguma lógica baseada nos parâmetros do pedido e retorna os resultados para o aplicativo cliente" (Morisseau-Leroy et al, 2001:30).*

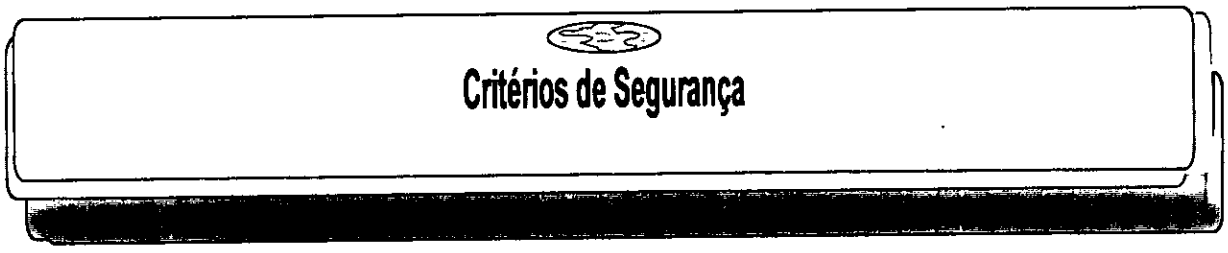
Para visualizar as páginas JSP, qualquer *browser* poderia ter sido usado. A aplicação desenvolvida usou o *Internet Explorer* e *Firefox* para os testes. Poderiam ter sido usados outros tais como:

- *Netscape;*
- *Opera e*
- *Mozilla.*

Para editar e construir as páginas dinâmicas existem diversas ferramentas disponíveis, dentre as quais se destacam as que foram usadas para a construção do modelo:

- *Textpad;*
- *Notepad.*
- *Macro Media Studio 2004;*
- *Microsoft Frontpage 2002;*
- *Eclipse 3.0;*
- *Adobe Photo Shop 7.0;*

Capítulo V



5 – CRITÉRIOS DE SEGURANÇA

Nos últimos anos, os sistemas de rede têm crescido consideravelmente em tamanho, complexidade e susceptibilidade a ataques. Ao mesmo tempo, o conhecimento, as ferramentas e as técnicas disponíveis aos agressores têm crescido com a mesma rapidez ou até mais rápido (Symantec, 2004).

Infelizmente, as técnicas de defesa não têm crescido tão rápido. As tecnologias actuais vêm atingindo suas limitações, sendo necessárias soluções inovadoras para lidar com o nível das ameaças actuais e futuras (ibidem).

Visto que o modelo proposto funciona em ambiente *Web,* e mais de um usuário tem acesso ao mesmo, ele não está isento de vulnerabilidades a malfeitores. Assim, torna-se necessário a implementação de medidas de Segurança da Informação[20], tal como o uso de chaves de acesso ao sistema, garantindo o acesso restrito da informação.

Ao longo da descrição que se segue, faz-se uma abordagem sobre alguns mecanismos de segurança física que poderão ser implementados futuramente, com vista a dar suporte ao Sistema de Gestão de Docentes do DMI e, abordam-se também diversos aspectos ligados a segurança lógica de dados. No que refere a parte lógica de segurança, o modelo proposto implementou alguns dos mecanismos de segurança tais como o protocolo SSL e, prevê a configuração de *firewall* na rede de transmissão dos dados do sistema.

[20] **Segurança de Informação** é um mecanismo que providencia meios para reduzir as vulnerabilidades existentes num Sistema de Informação. Tais mecanismos envolvem tecnologia, processos e pessoas (Misaghi, 2004).

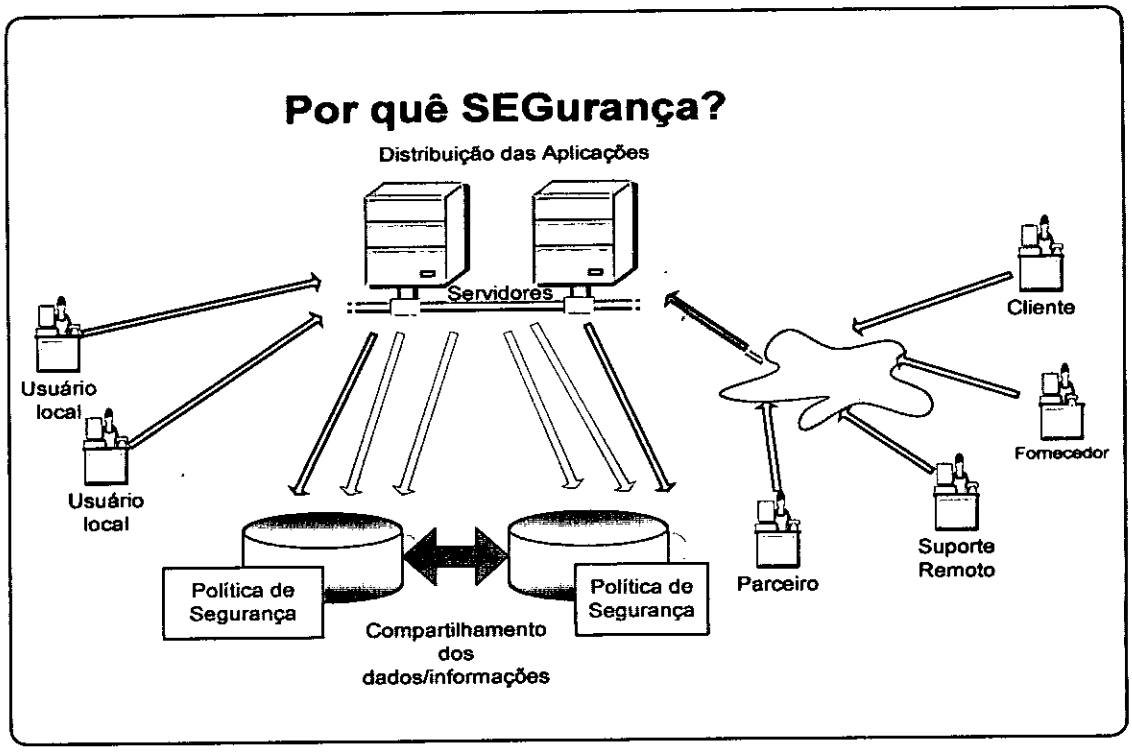


Figura 5 1 - Porquê Segurança (Ávila, 2001)

A segurança é subdividida em dois tipos: segurança física e lógica.

5.1 – Segurança Física

A **Segurança Física** é o processo que providencia mecanismos para restringir o acesso (físico) às áreas críticas de uma organização (Misaghi, 2004).

Quanto a este tipo de segurança, devem ser considerados os seguintes aspectos (Velloso, 2002):

- Os servidores devem possuir redundância completa;
- Deve existir um sistema de alimentação eléctrica sem interrupção (***no-break***) para os servidores e equipamentos activos;
- Devem existir duas salas – cofre (uma na organização e outra fora) para guardar cópias de segurança;
- O acesso a áreas de acesso às máquinas de manobra dever ser restrito, bem como o acesso aos concentradores principais, secundários e servidores.

5.2 – Segurança Lógica

A **Segurança Lógica** visa fornecer mecanismos para garantir confidencialidade, integridade e autenticidade dos dados (Misaghi, 2004).

Para garantir os mecanismos acima mencionados, são descritos alguns objectivos referentes à segurança de dados tais como (Figueirêdo, 2002):

- **Confidencialidade ou Privacidade** – proteger as informações contra acesso de qualquer pessoa não autorizada pelo gestor da informação. Este objectivo envolve medidas como controle de acesso e criptografia;
- **Integridade dos Dados** – evitar que os dados sejam apagados, ou alterados sem a permissão do gestor da informação;
- **Disponibilidade** – garantir o funcionamento do serviço de informática, sob demanda, sempre que necessário aos usuários autorizados. As medidas relacionadas a esse objectivo podem ser duplicação de equipamentos/sistemas e *backup* (cópia de segurança). Um bom exemplo de ataque contra disponibilidade é a sobrecarga provocada por usuários ao enviar enormes quantidades de solicitação de conexão com o intuito de provocar *crash* (congestionamento) nos sistemas;
- **Consistência** – certificar-se de que o sistema actua de acordo com a expectativa dos usuários;
- **Isolamento ou uso legítimo** – controlar o acesso ao sistema. Garantir que somente usuários autorizados possuam acesso ao sistema;
- **Auditoria** – proteger os sistemas contra erros e actos cometidos por usuários autorizados. Para identificar autores e acções, são utilizadas trilhas de auditorias e *logs* (mecanismo usado para relatar cada evento que ocorre durante a execução do sistema), que registam o que foi executado no sistema, por quem e quando.
- **Confiabilidade** – garantir que, mesmo em condições adversas, o sistema actuará conforme esperado.

Para que estes objectivos sejam alcançados, é necessário que se criem mecanismos que garantam a integridade das redes privadas e se consiga uma transmissão segura dos dados pela *Internet* usando para tal algumas ferramentas disponíveis no mercado como, por exemplo, *firewalls* nas redes e o protocolo SSL em combinação com o protocolo responsável pela transmissão de dados – o HTTP.

No modelo proposto, foram implementados alguns mecanismos de segurança de modo a garantir a transmissão segura dos dados na rede, como é o caso do protocolo HTTP em conjunto com SSL e mecanismo de criptografia para acesso ao sistema.

5.3 – Firewall

Firewall é um dispositivo formado por um ou mais equipamentos incluindo *software* e/ou *hardware*, projectado para proteger uma rede contra intrusos que possam estar em qualquer lugar da *Internet* (Lemos, 2004).

5.3.1 – Funcionamento de um Firewall

Os sistemas de *firewall* disponíveis comercialmente se diferenciam de duas maneiras: seu conjunto de características e sua arquitectura básica.

Segundo as suas arquitecturas, os *firewalls* estão disponíveis em três tipos:

- **Filtragem de Pacotes** (de dados) – cada pacote de IP é analisado isoladamente, sem nenhum tipo de correlação com outros pacotes dependendo de regras de filtragem embutidas em seu sistema operativo. Este projecto é considerado o mais rápido e flexível dos dois. (Serafim, 2004);
- **O *Firewall* a Nível de Aplicação** – esse tipo não permite que nenhum pacote passe directamente entre a rede externa e a rede corporativa. Ao invés disso, todos os pacotes são enviados a um servidor *proxy*[21]. O *proxy* determina quando se deve ou não estabelecer a conexão. Este sistema é mais lento que o da filtragem de pacotes e mais inflexível (*proxy* tem que conhecer a aplicação e, se uma nova aplicação for necessária na rede deve obter com o fornecedor, um novo código que a contemple) (Figueirêdo, 2002);
- **O *Firewall* Híbrido**: nesses sistemas estão inclusos elementos dos dois tipos anteriores de *firewall* de modo a maximizar a segurança e a performance (Figueirêdo, 2002).

[21] ***Proxy*** - máquina específica que abriga os programas dos serviços de filtragem e intermediação. É, também, o servidor Internet (Velloso, 2002).

5.3.2 – Características Básicas do Firewall

Os sistemas de *firewall* disponíveis hoje em dia vêm com quase todas as possíveis combinações de características. Algumas são essenciais, outras necessárias apenas em circunstâncias especiais. Figueirêdo (2002) aponta as seguintes características de *firewalls* como as básicas:

- **Controle de Acesso Básico**: essa é a razão de ser do *firewall* – controlar o acesso à rede, mas alguns destes oferecem controles de acesso mais poderosos, fáceis de administrar e difíceis de burlar;
- **Serviços Suportados**: isso se refere aos protocolos (no nível de aplicação) que o *firewall* reconhece e passa;
- **Autenticação de Usuários**: os *firewalls* a nível de aplicação normalmente utilizam esse conceito interrompendo aplicações e exigindo aos usuários que se autentiquem antes de continuarem a conexão no destino requisitado. Alguns serviços permitem isso: Telnet, FTP, e HTTP;
- **Auditoria e Alarmes**: os administradores de *firewall* devem ser notificados quando alguém tenta invadir a rede. Eles precisam saber que tipo de ataque está sendo direccionado contra eles, para que possam aumentar a segurança contra o ataque e o atacante. Eles também necessitam saber, quando possível, exactamente quem está atacando. Se essa informação for disponibilizada em tempo real, o administrador do *firewall* tem mais hipóteses de identificar o potencial invasor.

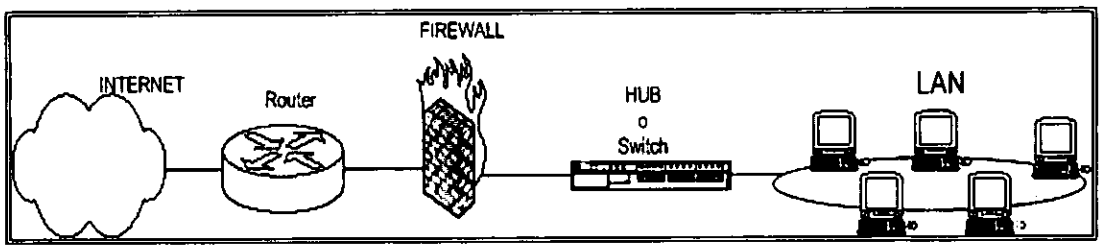


Figura 5 2 - Firewall em Redes (Morea, 1997)

No que refere a transmissão de dados que trafegam pela *Internet*, esta é efectuada através do protocolo **HTTP**. Para que haja garantia de privacidade durante a comunicação entre os diferentes intervenientes e, para que as suas mensagens não sejam modificadas ou corrompidas, adiciona-se no meio de transmissão, uma tecnologia responsável pela transmissão segura dos dados denominada ***Secure Socket Layer*** – **SSL** (Figueirêdo, 2002).

5.4 – Secure Socket Layer

SSL foi uma solução apresentada pela Netscape que se tornou um padrão, sendo incorporado nos mais populares browsers e servidores *Web* (Araújo, 1998).

SSL permite que o cliente se conecte a página *Web*, de forma transparente criando um canal de comunicação seguro entre a página *Web* e o cliente. Uma vez que esta conexão é feita, poderá ocorrer a troca de informação sem que se possa interceptar os dados, ou seja, de uma maneira segura (ibidem).

Esta segurança é garantida pela encriptação, um processo pelo qual os usuários que interceptarem a mensagem no caminho, ficam impedidos de aceder o conteúdo da mensagem, já que não conseguirão entender o que está sendo transmitido (ibidem).

Este protocolo é constituído por um conjunto de três protocolos situados dois deles, a nível de aplicação e, o terceiro entre o protocolo de aplicação e o *Transfer Control Protocol* (TCP), como ilustra a figura:

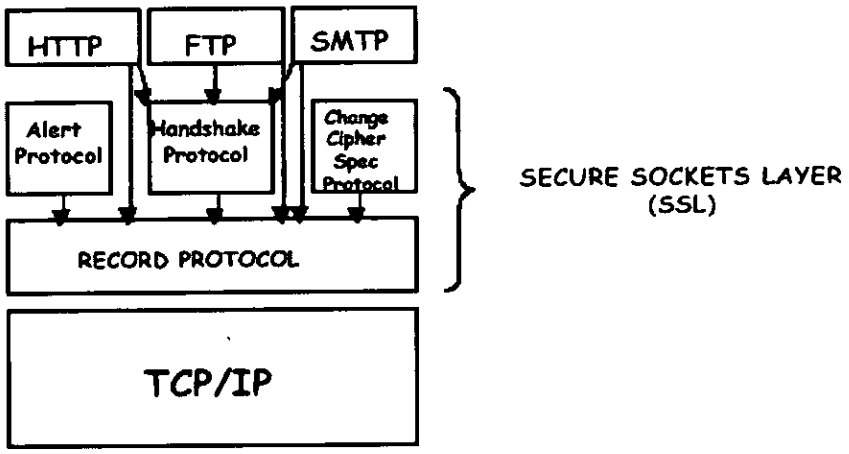


Figura 5 3 - Níveis de Segurança SSL (web4, 2004)

Legenda

TCP/IP – é o protocolo de comunicação básico da *Internet*, que serve para interligar redes cujos componentes usam sistemas operativos distintos.

Record Protocol – especifica o encapsulamento de todas as transmissões e recepções de dados. Faz parte das negociações entre o cliente e o servidor, no qual o emissor pode identificar qual o algoritmo de cifragem suportado.

Alert Protocol – é usado para enviar alertas a outras camadas.

Handshake Protocol – é usado para negociar os parâmetros de segurança na conexão SSL.

Change Cipher Spec Protocol – é usado para gerir os algoritmos de cifragem usados.

HTTP - é o protocolo de transmissão de dados usado para navegar na *Web* ou seja transferir ficheiros através de *browsers*.

FTP - através deste protocolo é possível efectuar a transferência de ficheiros pela *Internet* de uma maneira fiável. Actualmente existem programas que utilizam exclusivamente este protocolo, para que se possam efectuar transferências de ficheiros de uma forma prática.

SMTP (*Simple Mail Transfer Protocol*) – protocolo que fornece serviços de serviço electrónico.

SSL oferece suporte de segurança criptográfica para a informação que circula entre os diferentes intervenientes através de algoritmos simétricos, *message digest* (*hashing*) e métodos de autenticação e gestão de chaves (assimétricos) (Maia et tal, 1998)

Métodos de Criptografia

Criptografia é um conjunto de técnicas que permitem tornar incompreensível uma mensagem originalmente escrita com clareza, de forma a permitir que apenas o destinatário a decifre e compreenda (Web3, 2004).

O uso da criptografia tem como objectivo garantir que uma mensagem ou informação só será lida e compreendida pelo destinatário autorizado para isso.

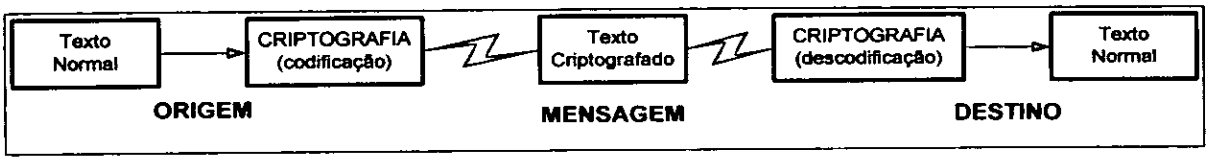


Figura 5 4 - Esquema de Criptografia (Velloso, 2002)

Tipos de Criptografia

Existem dois tipos básicos de criptografia, Criptografia Simétrica e Criptografia Assimétrica, conforme veremos a seguir (Web3, 2004):

Criptografia Simétrica – também conhecida por criptografia de chave secreta. Este, funciona transformando um texto numa mensagem cifrada, através da definição de uma chave secreta, que será utilizada posteriormente para decriptografar a mensagem, tornando novamente num texto simples (ibidem).

Os algoritmos simétricos usam chaves simétricas tais como: o DES, DESX, IDEA, RC2, RC4, *Blowfish*, para criptografar e decriptografar dados (Velloso, 2002).

Este algoritmo tem a **vantagem** de rapidez na criptografia e decriptografia das informações, porém tem a **desvantagem** de ser transmitida ou comunicada para o receptor, tornando-a mais vulnerável a roubo (Web3, 2004).

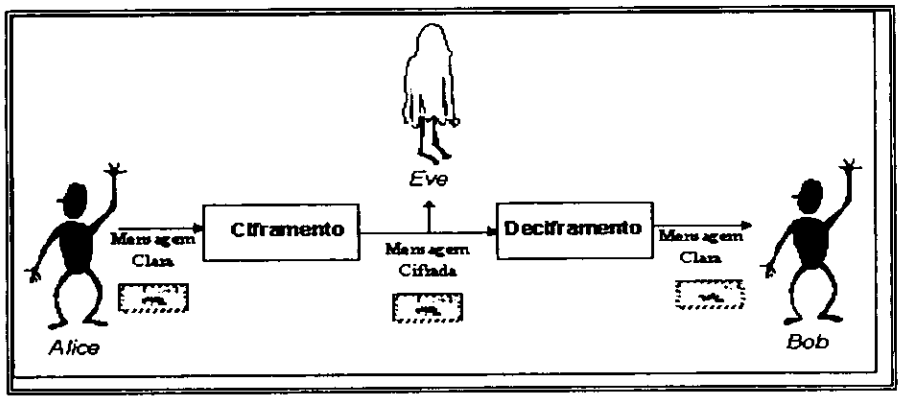


Figura 5 5 - Algoritmo Simétrico (Stallings, 1998)

Criptografia Assimétrica – conhecida como a de chave pública, utiliza duas chaves, uma para cifrar o texto ou mensagem, e outra para decifrar. Pode ser empregue para assinatura digital e autenticação. É possível combinar a criptografia simétrica com a assimétrica, somando a segurança com a rapidez (Web3, 2004).

Tem a **Vantagem** de ser mais segura que a criptografia simétrica, por não precisar comunicar o receptor, a chave necessária para decriptografar, a mensagem e, pode ser usada em assinatura digital (ibidem).

A **Desvantagem** deste algoritmo é que costuma ser mais lento do que a criptografia simétrica (Stallings, 1998).

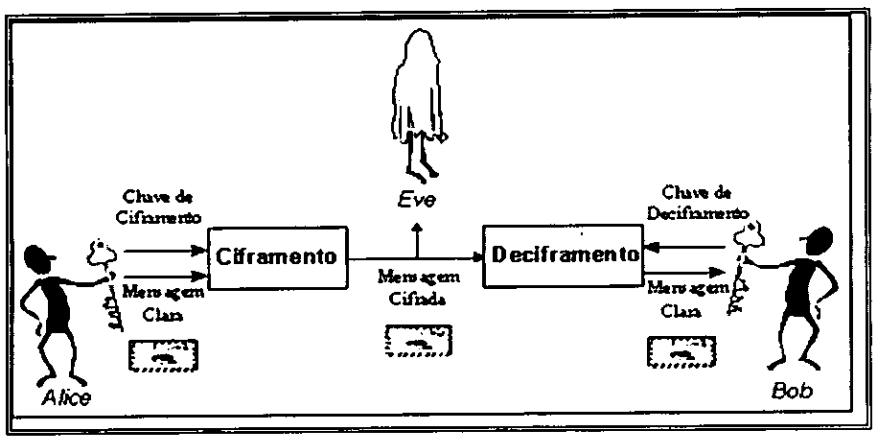
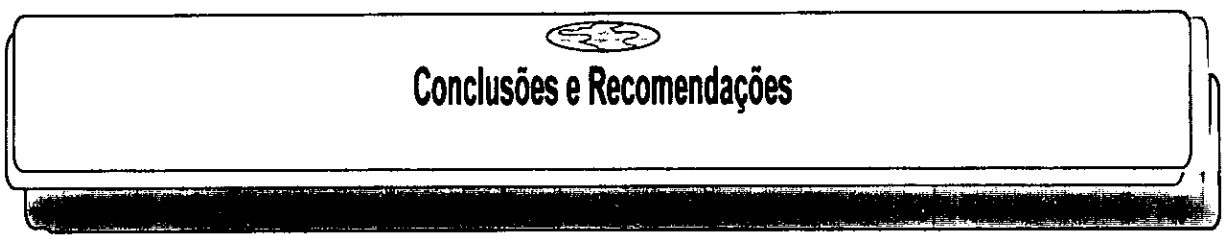


Figura 5 6 - Algoritmo Assimétrico (Stallings, 1998)

Capítulo VI



Conclusões e Recomendações

6 – CONCLUSÕES E RECOMENDAÇÕES

Este trabalho foi escrito para facilitar e disponibilizar as informações referentes a docentes através de processos automatizados desenvolvidos em ambiente *Web*. O sistema actual de docentes que rege o DMI possui transacções manuais exigindo bastante atenção e paciência por parte dos seus intervenientes. Isto faz com que este não esteja isento de constrangimentos ou mesmo disponibilização de informações incorrectas, devido ao atraso de algumas transacções e o tempo que se leva a processar a informação.

A proposta de implementação do modelo que funciona em ambiente *Web*, tem em vista solucionar estes constrangimentos pelo uso do sistema, como por exemplo:

- Falta de informação disponível em tempo real,
- Perda de tempo,
- E outros recursos adicionais.

O Sistema desenvolvido responde também a algumas necessidades que o DMI enfrenta tais como:

- Armazenamento de informação referente ao historial dum docente,
- Acesso a alguns relatórios como por exemplo:
  - Lista de disciplinas que um docente lecciona,
  - Número de estudantes que um docente possui por turma,
  - Lista de docentes por secção,
  - Planificação de actividades de docente e,
  - Por ser em ambiente *Web* seja possível realizar todas estas transacções em qualquer ponto do mundo.

Neste contexto concluiu-se que:

- A necessidade de automatização dos processos de trabalho e acesso à informação em tempo real faz com que os desenvolvedores de sistemas adoptem tecnologias que disponibilizem interface *Web*. Para tal, foram desenvolvidas páginas lógicas e dinâmicas que permitam com que os usuários possam aceder remotamente as funcionalidades do sistema;

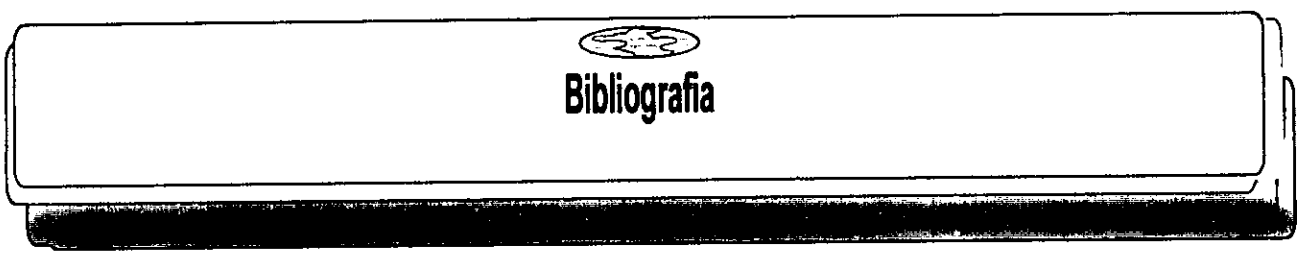
- A tecnologia usada para implementação deste modelo depende de certos factores tais como: dimensão da organização, número de usuários, natureza da informação que irá tratar, vantagens e desvantagens do seu uso, que não implique custos não suportáveis da aplicação desenvolvida e que seja fundamentalmente de fácil manutenção e segura. Assim sendo, foi usada a tecnologia JSP para o desenvolvimento das páginas *Web* para o modelo proposto;
- JSP é uma tecnologia para desenvolvimento de páginas *Web*, que se destaca das demais, por apresentar uma solução baseada em códigos Java embutidos em HTML e que corre sobre qualquer Sistema Operativo, desde que possua um servidor que a suporte. Para o efeito, usou-se o Tomcat, que se mostrou o servidor mais adequado para suportá-lo pelas suas funcionalidades e por ter a possibilidades de ser obtido gratuitamente na *Internet*;
- O SGBD usado, MySQL, é o mais adequado para este Sistema visto ele:
  - Fornecer recursos simplificados e apropriados para as suas aplicações, tendo um custo extremamente reduzido;
  - Fornecer um completo acesso ao código fonte, assegurando uma maior liberdade para as diversas plataformas existentes;
  - Ser suportado por uma série de plataformas como o Linux, MacOS X, UNIX e Microsoft Windows;
  - E em relação a outros como por exemplo ORACLE, poder ser adquirido gratuitamente na *Internet*.
- As aplicações produzidas em ambiente *Web*, devem possuir mecanismos de segurança, visto que a informação está disposta numa rede mundial. No modelo proposto, investigou-se a segurança a dois níveis, nomeadamente a física e a lógica, propondo-se a instalação de uma sala para o servidor, um firewall que controle o tráfego na *Internet* e o protocolo SSL para a transmissão segura dos dados.
- Após diversos testes de usabilidade com alguns dos usuários do sistema conclui-se também que, a implementação deste modelo, trouxe soluções automatizadas para transacções que dizem respeito à Gestão de Docentes do DMI.

Recomendações:

De modo a resolver os actuais problemas de gestão de docentes no DMI, com recurso aos resultados do presente trabalho recomenda-se:

- A instalação e uso do Sistema proposto pelo DMI para a realização das transacções relacionadas com docentes;
- O estudo para a implementação do modelo proposto em outros departamentos da UEM;
- O treinamento do pessoal envolvido, através do manual do utilizador que está em anexo (Anexo G);
- A instalação de uma sala para armazenar o servidor local onde correrá a aplicação. Esta sala deverá conter alguns mecanismos de segurança abordados no trabalho.

Capítulo VII



7 – BIBLIOGRAFIA

7.1 – Bibliografia Referenciada

Araújo, G. (1998), Transacções Seguras via *Web*, http://www.rnp.br, consultado em 17/07/04.

Ávila, E. (2001), Segurança Lógica de Dados, http://www.prodase.com.br, consultado em 23/09/04.

Barreto, A. (1997). http://www.alternex.com.br, consultado em: 22:05:04 as 23:40 hrs

Barros, P. (2000), Linguagem de Modelagem Unificada, http://www.eng.uerj.br consultado: 08/08/04;

Bennet, M. et al (2002), Object-Oriented System and Design using UML, Mc Graw-Hill, Second Edition, 2002

Bezerra, E. (2003). Princípios de Análise e Projecto de Sistemas com UML, Rio de Janeiro, Editora Campus

Graham, I. S. (1998), A Glossary of Internet and Web Terminology, http://www.utoronto.ca, consultado em 12/01/05

Bond, M. et al (2002). Teach Yourself J2EE in 21 Days, United States of America, Sams Publishing, 581-628pp.

Brandi, V.J. (2004), Objetos Implícitos e Ações das JSP, http://www.unimep.br, consultado em 19/09/04

Brazdil et al, (2004), http://www.liacc.up.pt, consultada a 19/09/04

Brodbeck, H. (2004), Desenvolvimento, Segurança e Qualidade na Internet, http://www.inf.ufrgs.br, UFRGS - Banco do Brasil, consultado em 20/06/04

CCNA, (1999), Cisco Networking Academy Curriculum, http://cisco.netacad.net, EUA, consultado em 22/09/04

Coelho, P. (1996), Internet em Windows 95 & 3.1, Lisboa, 2ª Edição, FCA – Editora de Informática

Figueirêdo, L. S. (2002), Segurança da Tecnologia da Informação, http://www.modulo.com.br, consultada em 26/09/04

Franklin, (2004), Rational Unified Process, http://www.portaljava.com.br, consultado em 13/08/04

Jeveaux, P. C. (2003), Java Server Pages, http://portaljava.com, consultado em 13/07/04

Katalogo, (2004) MySQL, http://www.katalogo.com.br, consultado em 23/09/04

Leiner et al, (2004), História da Internet, http://www.aisa.com.br, consultado 23/09/04

Lemos, E., (2004) Firewalls, http://www.securityexperts.com.br/, consultado em 19/09/04

Maia, M. A. (1999), Especificação Formal da Interação de Componentes de Sistemas Computacionais, http://www.dcc.ufmg.br, consultado em 17/09/04

Maia et al, (1998), Criptografia e Certificação Digital, http://www.training.com.br, consultado em 20/11/04

McFadden, F. et al. (1999), Modern Database Management, Fifth Edition, Eddison Wesley

Misaghi, M. (2004), Conceitos Básicos de Segurança, http://www.vision.ime.usp.br, consultado em 25/09/04

Morea, L. (1997), http://www.monografias.com, consultada em 29/09/04

Morisseau-Leroy, N. et al (2001), Oracle 8i – Programação de Componentes Java com EJB, CORBA e JSP, Brazil, Editora Campus Limitada

MySQL, (2004), MySQL Manual, http://dev.mysql.com , consultado em 22/09/04

Neto, A.C. (2001), Análise e Projeto de Sistemas I, http://www.dcce.ufs.br, consultado em 21/08/04

Nunes, M. E O'Neill, H (2001). Fundamental de UML, Lisboa, Editora de Informática FCA

Oliveira, A. P. (2001), Apostila Servlet/JSP, Universidade Federal de Viçosa

Pablo, B. (2000), Linguagem de Modelagem Unificada, http://www.eng.uerj.br , consultado: 27/08/04

Pereira, J. L. (1998). Tecnologia de Base de Dados, Lisboa, Editora de Informática

Prado, I. B. (2003), Dicionário de Informática, http://planeta.terra.com.br/informatica/dicinfo, consultado 10/10/04

Prodigio, (2001), http://bvi.clix.pt , consultada em 24/09/04

Reese, G. (2001), Database Programming with JDBC and Java , Second Edition, O'Reill

Reis, O (2004), Myrup: uma Adaptação do RUP para Projetos de Pequeno e Médio Porte, http://.clei2004.spc.org.pe, consultada em 24/09/04

Rodrigues, M (2004), Modelo Relacional parte 1, http://www.dimap.ufrn.br, consultado em 09/09/04

Romão, B. (2004). Portal da Programação, http://www.portaldaprogamacao.com, consultado em 13/09/04

Santos, G. A. (2004), Java Server Pages, http://www.ucb.br, consultado em 20/09/04

Serafim, V. (2004), Implantação de Firewalls: Teoria e Prática, http://www.apostilando.com , consultado em 27/09/04

Stallings, W. (1998), Cryptography and Network Security ,2ª edição, Prentice Hall, EUA

Sun, M. (2000), JavaServer Pages(TM) Tutorial, http://www.sun.com, consultada em 22/09/04

Terry, Q. (2001), Modelagem Visual com Rational Rose 2000 e UML, Brazil, Editora – Ciência Moderna,

Thomas, S. (2001), HTTP Essentials, Protocols for Secure, Scaleable Web Sites, New York, Wiley Computer Publishing.

Ulisses, T. (2004), Java Server Pages e Servlets, http://www.jspbrasil.com.br, consultado em 14/09/04

Velloso, F. C, (2002), Informática: Conceitos Básicos, 6ª Edição, Lisboa: Editora Campus

Vilas, A. F.(2001), Diagramas de Actividades, http://www.gris.det.uvigo.es , consultado em 17/09/94

Web1, (2003), http://docentes.lis.ulusiada.pt , consultado em 12/09/04

Web2, (1997), Tudo Sobre a Internet, http://gerencia@torque.com.br , consultada em 23/09/04

Web3, (2004), SCUA, Segurança de Informação, http://www.scua.com.br , consultado em 20/09/04

Web4, (2004), NIVEL DE SEGURIDAD SSL (Secure Sockets Layer), http://www.pegasi.com, consultado em 23/09/04.

Web5, (2004), Unidade3 – Java Server Pages, http://www.unama.br, consultado em 17/09/04

Web6 (2004), Struts e a Arquitetura MVC: Servlets e JSP juntos , http://jacques.dsc.ufcg.edu.br , consultado em 18/09/04

Web7, (2002), Multipurpose Internet Mail Extensions MIME, http://www.mhonar.org , consultada em 20/09/04

Welling, L. e Thompson, L. (2001),.Php and MySQL Web Development, Sams Publishing

Wikipedia, (2004), Wikipedia, a Enclicopédia Livre, http://www.webster-dictionary.org , consultado em 13/09/04

7.2 – Bibliografia Consultada

Amaral, W. (1999), Guia Para Apresentação de Teses, Dissertações, Trabalhos de Graduação, 2ª Edição, Livraria Universitária, UEM

Afonso, C. J. (2003), InterSystemsCaché:Um Banco de Dados de Terceira Geração, http://www.linhadecodigo.com.br, consultado em 12/09/04

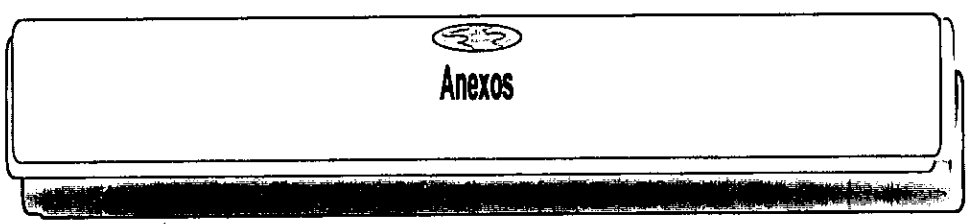
Guerrero, D., (2003), Servicios de Certificación, http://www.rediris.es,consulta em 14/11/04

Kienetz et al, (2004), Um hacker na Espanha, http://www.noticiaslinux.com.br, consultado em 26/11/04

Nesi, G. M. (2004), XML - Extensible Mark-up Language, http://www.gta.ufrj.br, consultado em 08/12/2004

Silberschatz, A. et al, (1999). Sistema de Banco de Dados, 3ª Edição, São Paulo: Makron Books

UEM, (2001), Estatísticas de 5 de Setembro – Universidade Eduardo Mondlane, Maputo



Anexos

ANEXOS

Anexo A – Cenários de Use Cases e Actores

Estes cenários são os possíveis caminhos seguidos dentro dos *use cases* de forma a fornecer ao actor uma resposta. Esta descrição pode assumir a forma de texto livre ou estruturada seguindo um conjunto de passos enumerados, ficando esta decisão ao critério do analista (Nunes, 2001). A lista de requisitos inclui a coluna que mostra que "caso de uso" realiza a funcionalidade de cada requisito representada a seguir duma forma estruturada seguindo abordagem ilustrada por McFadden (1999).

Nº	**Requisitos**	**Actor(s)**	**Caso de Uso**	**Descrição**
1	Introduzir um novo Docente no sistema.	Chefe de Secção	Introduzir Docente	Quando um novo Docente entra para o Departamento, seus dados têm de ser introduzidos no sistema, para tal: 1. O actor entra para o sistema e selecciona no Menu a opção de Introduzir Docente; 2. O sistema abre uma tela de Cadastro de Docente; 3. O actor introduz dados para cadastro de Docente e selecciona em SUBMETER; 4. O sistema pergunta se deseja mesmo salvar estes dados; 5. O actor confirma cadastro de Docente; 6. O Sistema salva estes dados e dá confirmação;
2	Para alterar dados de um	Chefe da Secção	Alterar Dados do Docente	Após mudanças no estado de um Docente (podendo ser os anos de experiência, categoria, tipo de docente, secção a que ele está afecto), deve-se actualizar o estado deste no sistema para tal:

	Docente, bem como as sua funções no sistema.			1. O actor escolhe opção para alterar dados de Docente; 2. Sistema apresenta uma janela com a lista de docentes existentes no Sistema; 3. O actor escolhe um docente e clica no botão SUBMETER; 4. Sistema abre uma janela com dados deste Docente para alterar; 5. O actor altera os campos desejados e selecciona a opção SUBMETER; 6. Sistema salva estes dados e dá confirmação.
3		Chefe da Secção/Director do Curso	Consultar Dados Docente	Cada vez que o Chefe da Secção ou o Director do curso desejar consultar dados de um Docente ele pode fazê-lo, para tal deverá. 1. O actor selecciona no menu a opção 'Consultar Dados Docente' e clica num botão SUBMETER; 2. Sistema abrirá uma janela com lista de docentes; 3. O actor selecciona o docente que pretende consultar seus dados e clica num botão SUBMETER; 4. Sistema abrirá uma janela com os dados do docente;
4	Permite alterar Situaçã o do docente no sistema, poderá ser	Chefe da Secção	Remover Dados Do Docente	Cada vez que o chefe da Secção deseja remover um docente, ele poderá fazê-lo, para tal deverá . 1. O actor selecciona no menu a opção Remover Docente e clica no botão SUBMETER; 2. O sistema abrirá uma janela com a lista de docentes existentes no sistema; 3. O actor selecciona um docente e clica num botão SUBMETER; 4. O sistema altera a situação do docente e dá

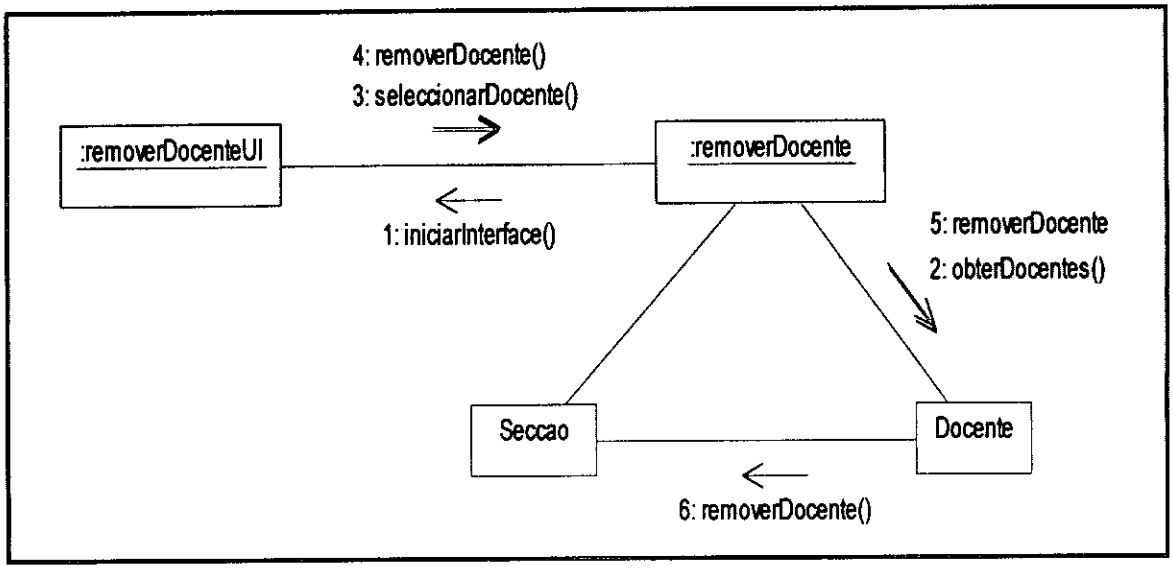
	"Activo" ou "Inactivo"			confirmação;
5	Permite Introduzir no sistema actividade científica produzida por um docente.	Docente e/ou Chefe da Secção	Introduzir Actividade Científica	Cada vez que um Docente realiza uma actividade cientifica, ele pode desejar introduzir no sistema, para tal ele, ou o Chefe da Secção podem fazê-lo seguindo os seguintes passos: 1. O actor selecciona no menu a opção Introduzir Actividade Científica; 2. O sistema abre a janela para introdução de actividade Instrutiva; 3. O actor introduz os dados da actividade científica e clica num botão *Submeter*; 4. Sistema salva estes dados e dá confirmação.
5	Permite Introduzir no sistema material instrutivo produzido por um docente	Docente e/ou Chefe da Secção	Introduzir Material Instrutivo	Cada vez que um Docente produz um material instrutivo, ele pode desejar introduzir no sistema, para tal ele, ou o Chefe da Secção podem fazê-lo seguindo os seguintes passos: 5. O actor selecciona no menu a opção Introduzir Actividade Instrutiva; 6. O sistema abre a janela para introdução de actividade Instrutiva; 7. O actor introduz os dados da actividade Instrutiva e clica num botão *Submeter*; 8. Sistema salva estes dados e dá confirmação.

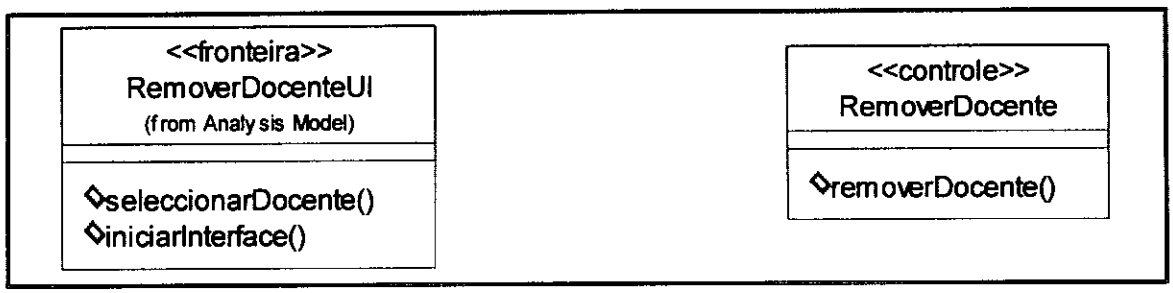
6	Permite alterar ou adicionar o conteúdo do material Instrutivo	Docente e/ou Chefe da Secção	Actualizar Material Instrutivo	Cada vez que um docente pretende disponibilizar um material instrutivo ele pode introduzi-lo no sistema, para tal ele terá de seguir os seguintes passos: 1. O actor selecciona no menu a opção Actualizar Material Instrutivo; 2. O sistema abre uma janela com uma lista de Docentes; 3. O actor selecciona o docente sobre o qual pretende operar e clica no botão *Submeter*. 4. O sistema abre uma janela com a lista de material instrutivo publicado por este docente; 5. O actor selecciona o material instrutivo pretendido e clica no botão SUBMETER. 6. O sistema abre uma janela de actualização de material instrutivo; 7. O actor actualiza o material instrutivo e clica no botão SUBMETER. 8. O sistema actualiza e dá confirmação.
7	Permite listar as actividades científicas existentes	Chefe da Secção	Listar Actividades Científicas	Cada vez que o Chefe de Secção desejar visualizar as actividades cientificas ele poderá fazê-lo de várias maneiras para tal deverá seguir os seguintes passos: 1. O actor selecciona no menu a opção relatórios; 2. Sistema abrirá janela com uma das opções Actividades Cientificas; 3. O actor selecciona no a opção listar Actividades Científicas e clica num botão SUBMETER; 4. Sistema abre uma janela com lista das Actividades Científicas;

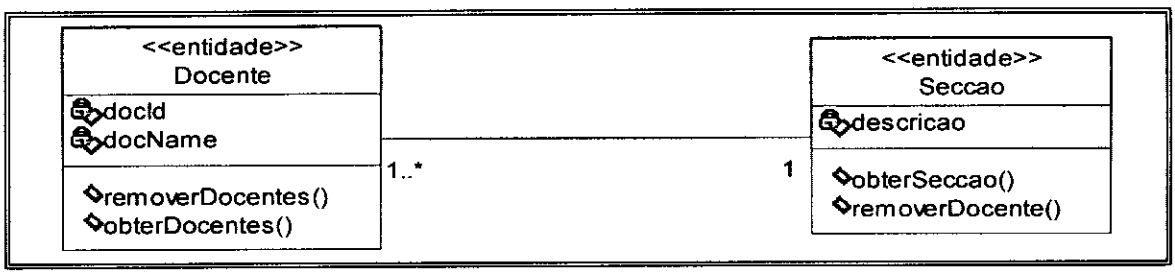
8	Permite listar as actividades instrutivas existentes	Chefe da Secção	Listar Actividades Instrutivas	Cada vez que o Chefe de Secção desejar visualizar as actividades instrutivas ele poderá fazê-lo de várias maneiras para tal deverá seguir os seguintes passos: 5. O actor selecciona no menu a opção relatórios; 6. Sistema abrirá janela com uma das opções Actividades Instrutivas; 7. O actor selecciona no a opção listar Actividades Instrutivas e clica num botão SUBMETER; 8. Sistema abre uma janela com lista das Actividades Instrutivas;

Anexo B – Uso de Diagramas de Interacção no Sistema

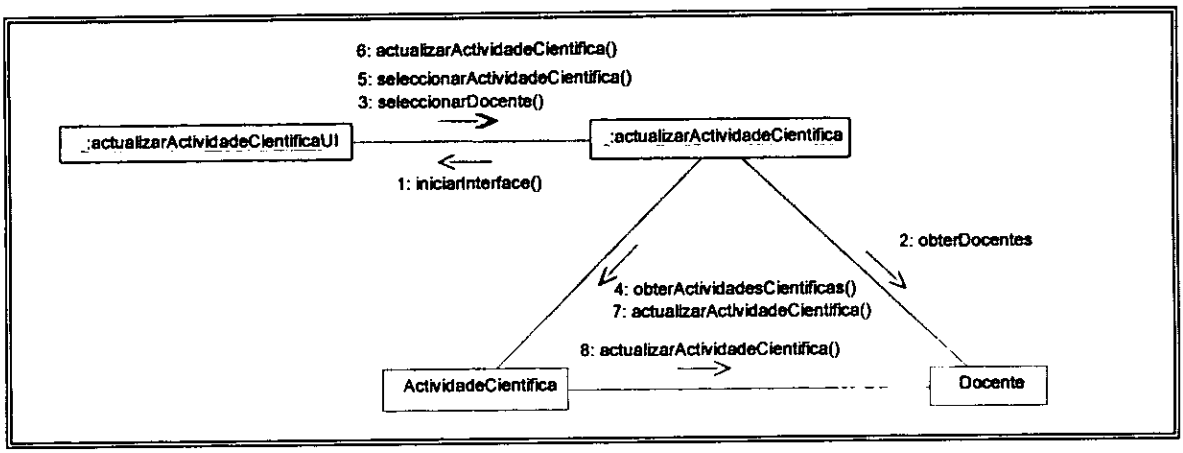
Remover Docente

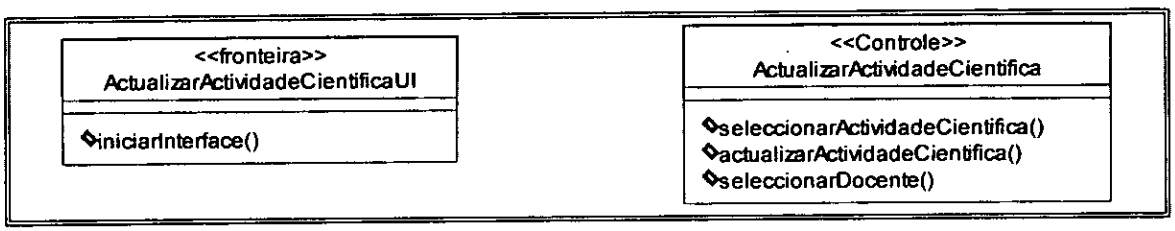


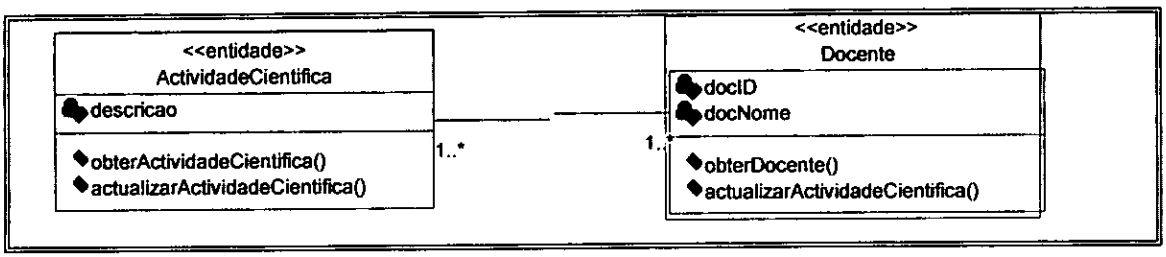




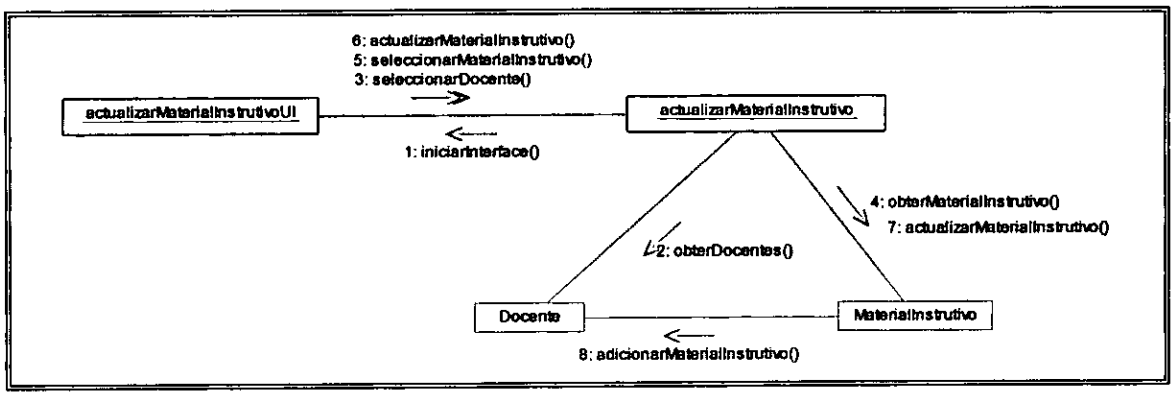
Actualizar Actividade Cientifica

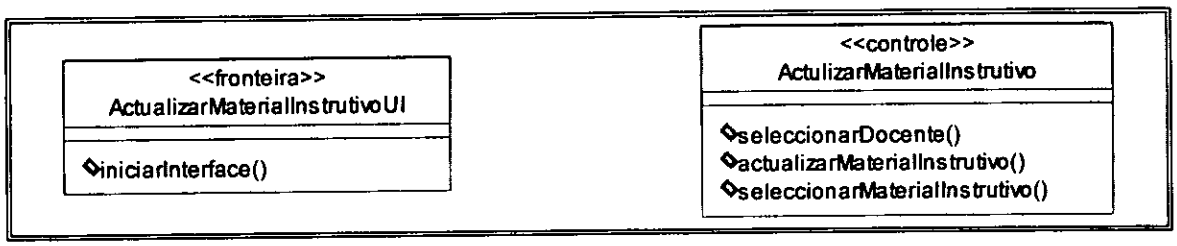


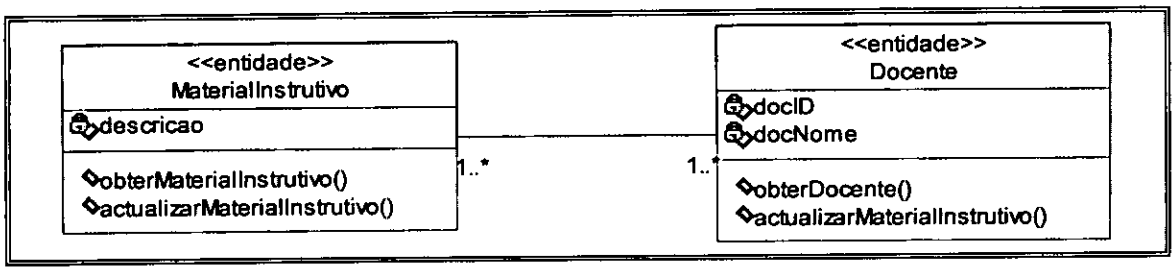




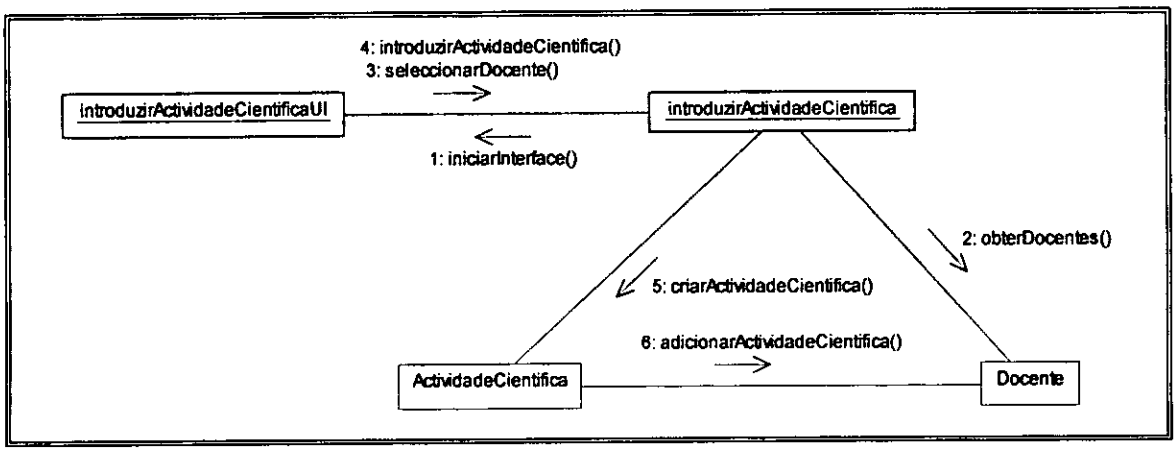
Actualizar Material Instrutivo

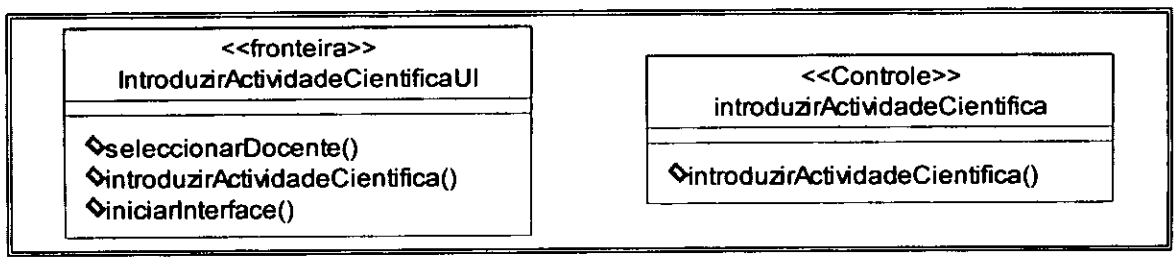


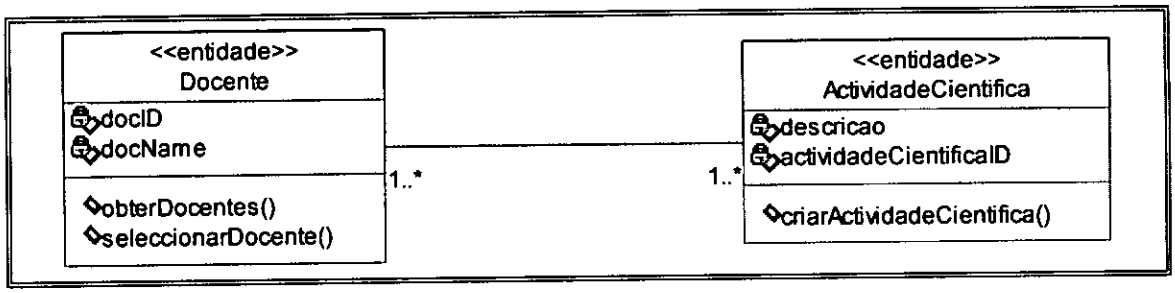




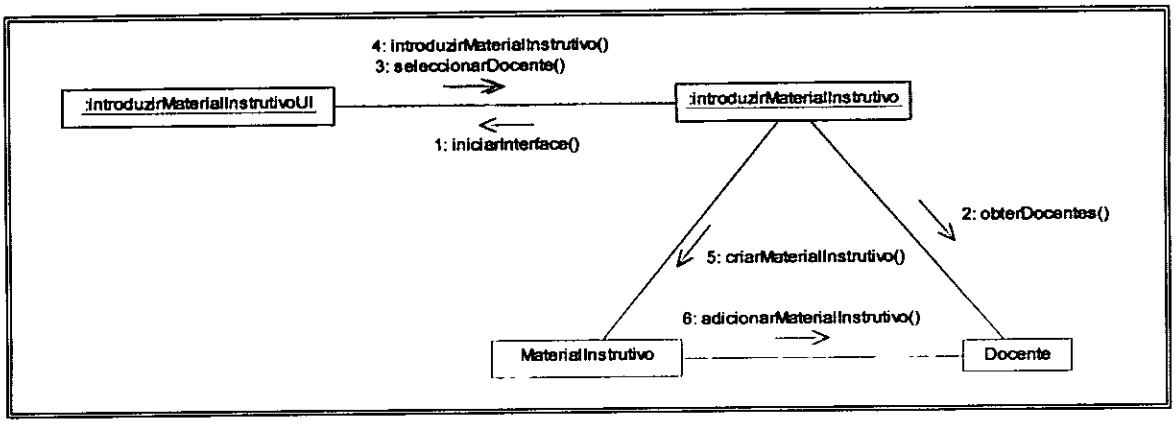
Introduzir Actividade Cientifica

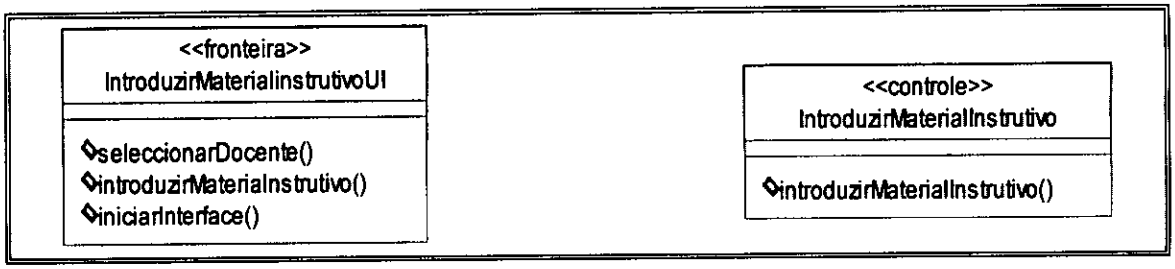


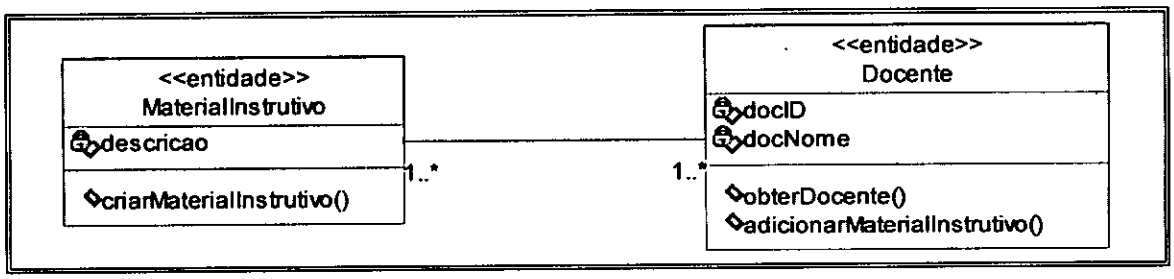




Introduzir Material Instrutivo







Anexo C – Tipo de Associações entre Classes

As **associações** entre as classes permitem especificar que objectos de uma dada classe se relacionam com objectos de outra classe (Bennet et al, 2002).

As relações possibilitam as conexões necessárias para que a passagem de mensagens ocorram. Quando se decide como implementar uma associação é importante analisar a mensagem que está vinculada pela ligação. Bennet et al (2002), divide as associações na fase de desenho em três grupos principais:

Associação do tipo Um-Para-Um

Associação de *um-para-um* : é aquela em que um objecto de uma classe pode enviar uma mensagem para outro de outra classe mas não vice-versa (ibidem).

Para o sistema em estudo, este tipo de associação poderá ser notada na classe do tipo *Secção* que precisa enviar uma mensagem para objectos do tipo *Curso,* mas não o inverso. Esta particular associação poderá ser implementada pondo um atributo para guardar o identificador (ou referência) do objecto da classe *Curso* na classe *Secção.* Assim, como o objecto *Secção* tem um objecto identificador de *Curso* então, este poderá enviar mensagem para o objecto *Curso.* O objecto *Curso* não pode enviar mensagem para objecto *Secção* porque ele não contém um objecto identificador do objecto *Secção.*

A associação *tem,* é um exemplo de associações de uma direcção. A cabeça da linha mostra a direcção para onde é possível navegar. Assim, antes de se desenhar uma associação, é importante decidir em que direcção ou direcções as mensagens devem ser enviadas (ibidem).

Se um objecto de uma **classe A** não precisa enviar mensagem para nenhum objecto nem receber mensagem de outro objecto, então a razão da sua existência deve ser questionada (ibidem).
A figura abaixo (Figura C.1) mostra uma associação deste tipo do modelo proposto.

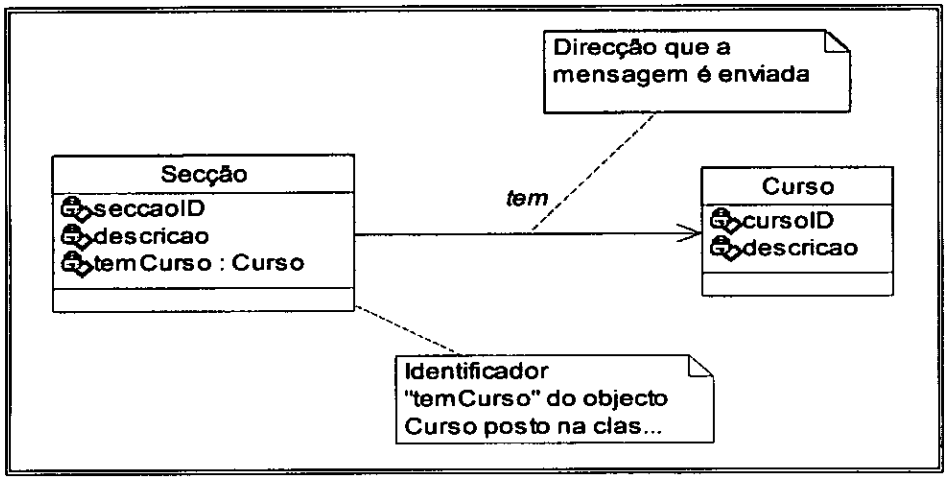


Figura C 1 - Uma direcção, associação de um-para-um

Associação do tipo Um-Para-Muitos

Associação um-para-muitos : é aquela em que objectos de uma classe podem enviar uma ou mais mensagens para objectos de outra classe, mas não vice-versa (Bennet et al, 2002).

Na figura C.2 a seguir, objectos da classe *Secção* precisam enviar mensagens para objectos da classe *Docente,* mas não o inverso. Se a associação entre as classes fosse de um-para-um, a associação poderia ser implementada bastando por um atributo para guardar o objecto identificador da classe *Docente* na classe *Secção.* Mas entretanto, a associação é de facto do tipo um-para-muitos e muitos identificadores do objecto *Docente* precisam estar ligados a um único objecto da *Secção.*

Uma maneira de implementar o agrupamento de identificadores de objectos *Docente* que é fácil de se reusar, é agrupando-os num objecto separado, numa colecção de objectos que tem mais operações para gerir identificadores de objectos e isso comporta-se como se fosse um índice de *Docentes* para o objecto *Secção.* Isto é ilustrado na fragmentação de diagrama de classes. Haverão várias instâncias da classe *ColecçãoDocente* bem como existem para cada objecto da classe Secção (tem sua própria colecção de objectos identificadores de *Docente).* Note que a classe *Colecção de Docentes* contém operações que são especialmente concernadas para gestão da colecção.

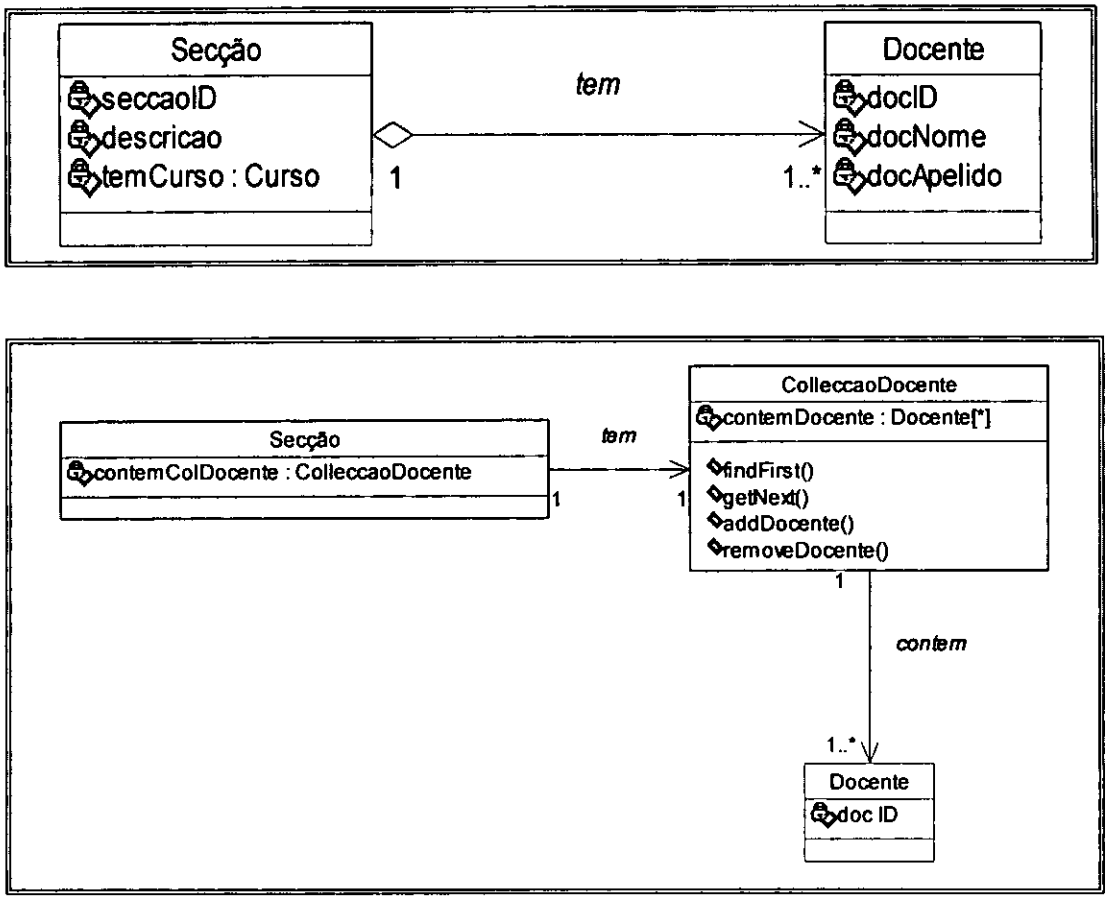


Figura C 2 - Associação de um-para-muitos usando colecção de classe

Associação do tipo Muitos-para-Muitos

No fragmento de diagrama de classe ilustrado na figura C.3, objectos da classe *Docente* precisam enviar mensagens para objectos da classe *ActividadeCientífica* e vice-versa.

Assumindo que esta é uma associação em duas direcções, um objecto *Docente* precisará ter uma colecção de objectos identificadores de *ActividadesCientificas*, e cada objecto *ActividadeCientifica* terá de ter uma colecção de identificadores de objecto *Docente*.

A implementação deste caso é ilustrada na figura C.3.1 em que ambos *Docente* e *ActividadeCientifica* contém atributos para carregar objectos de suas respectivas *Colecções* de classes.

Colecções de classes podem ser desenhadas para providenciar suporte adicional para navegação de objectos.

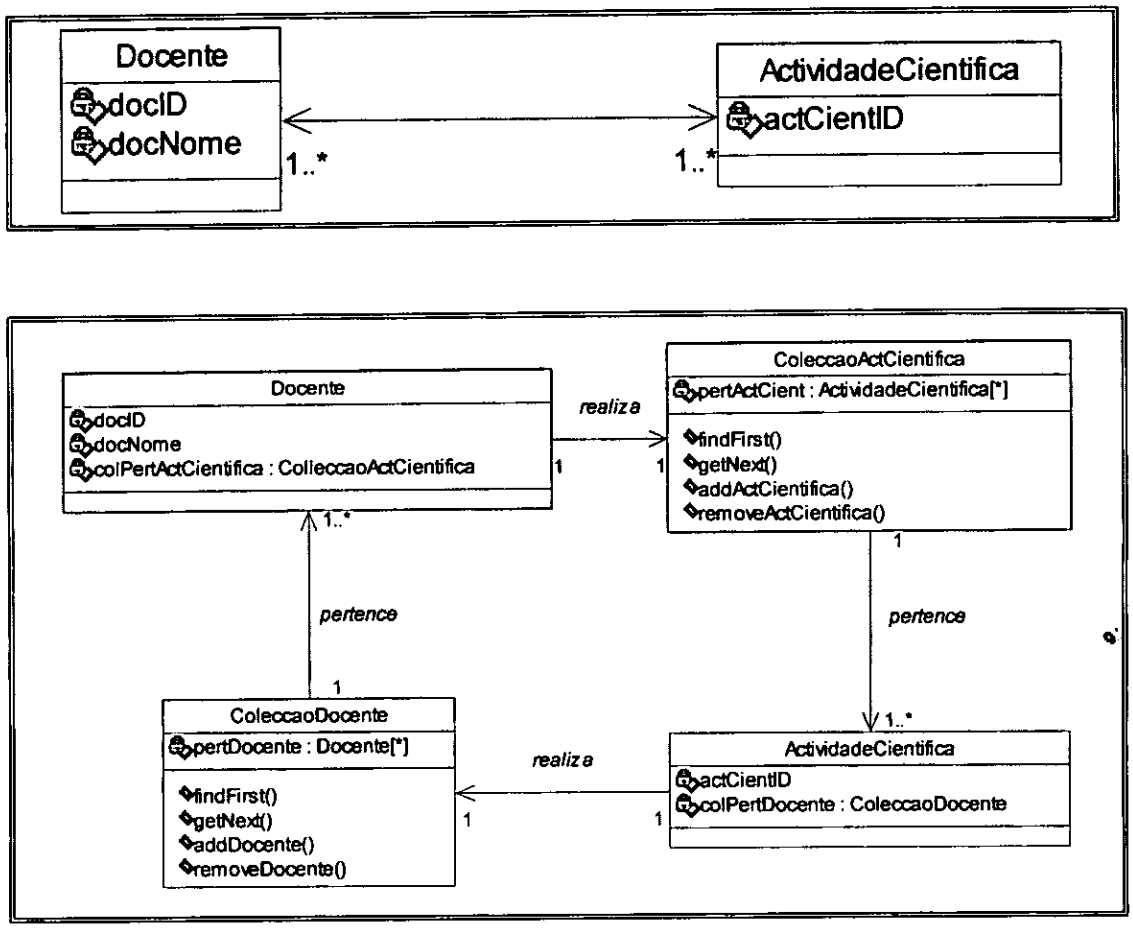


Figura C 3 - Duas direcções Associação de muitos para muitos

Estas refinações foram implementadas em todo o modelo de classes do sistema em estudo. Abaixo, ilustra-se resumidamente em forma de diagramas de classes da fase de análise (figura C.4)

Diagrama de Classes da fase de Análise

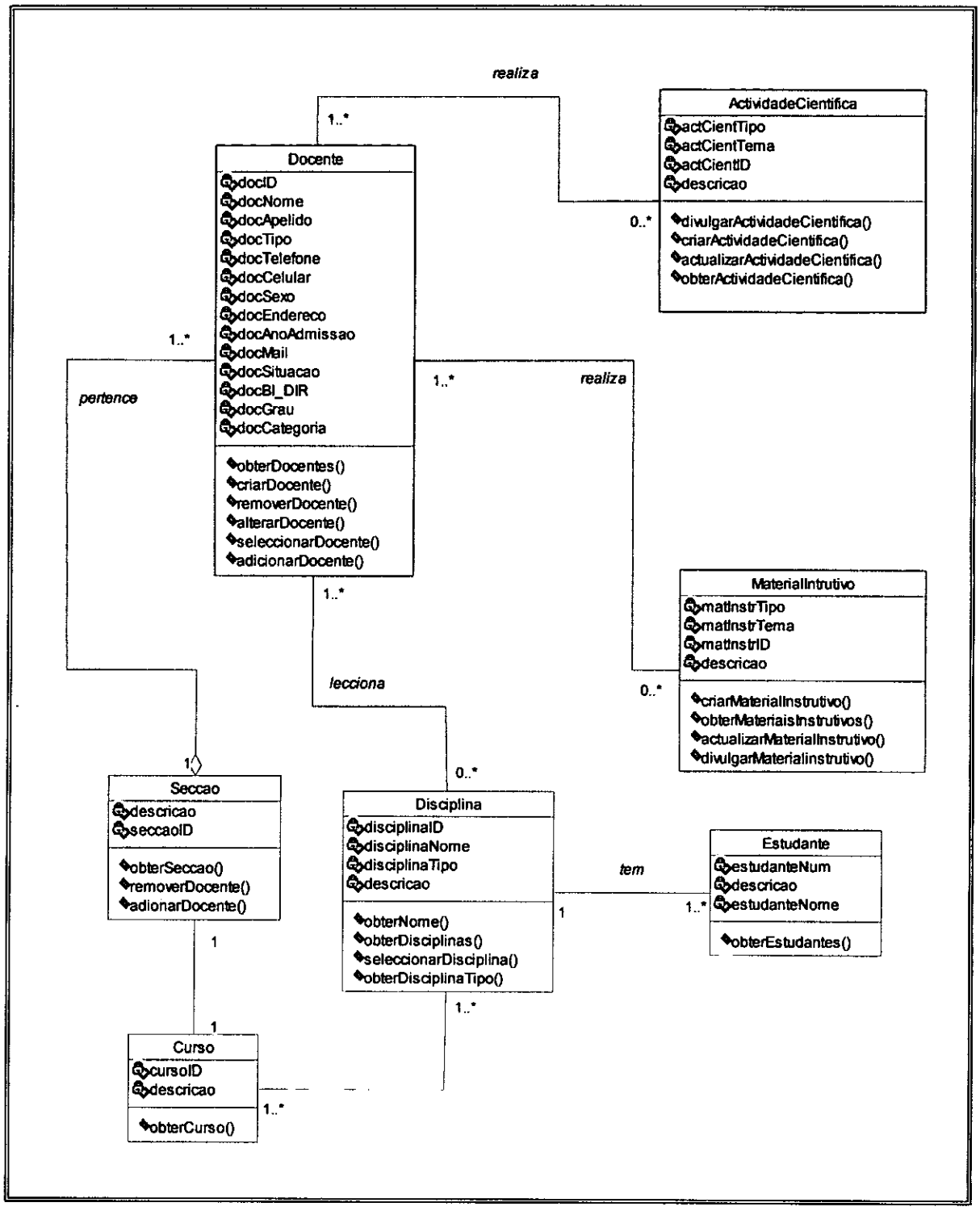


Figura C 4 - Diagrama de Classes da fase de Análise

Anexo D – Requisitos de um Sistema

Segundo Booch (1999), um **requisito** num sistema é uma funcionalidade ou característica considerada relevante na óptica do utilizador. Normalmente representa o comportamento esperado do sistema, que na prática consiste num serviço que deve ser disponibilizado a um utilizador.

Nunes (2001) identifica três categorias de requisitos:

- **Requisitos funcionais** – descrevem o que um sistema faz ou é esperado que faça. Estes são os requisitos que inicialmente são levantados, abrangendo a descrição de processamentos a efectuar pelo sistema, entradas (*inputs*) e saídas (*outputs*) de informação em papel ou no ecrã que derivam da interacção com pessoas e outros sistemas.
- **Requisitos não funcionais** – relacionados com as características qualitativas do sistema, descrevendo a qualidade com que o sistema deverá fornecer os requisitos funcionais. Abrange medidas de desempenho como por exemplo, tempos de resposta, volume de dados ou considerações de segurança.
- **Requisitos de facilidade de utilização** – garantem que existirá uma boa ligação entre o sistema desenvolvido, utilizadores do sistema e também as tarefas que desempenham utilizando o sistema.

Anexo E – Regras para Desenho de Base de Dados

Segundo Bennet et al, (2002), o mapeamento de classes para tabelas segue-se um conjunto de regras que possibilitam a passagem das classes e suas multiplicidades em tabelas de uma base de dados relacional. Tais regras estão listadas abaixo:

- Classes com uma simples estrutura de dados, são transformadas em tabelas;
- Identificadores de objectos tornam-se chaves primárias. Um único identificador é gerado para todos os objectos e pode ser usado como chave primária na tabela relacional a que ela pertence;
- Classes que contém uma instância de outra classe como atributo. Uma tabela separada deve ser criada para a classe embutida. Objectos da classe embutida devem ser alocados um único identificador. O identificador deve substituir o objecto embutido na tabela da classe que contém a colecção, como chave estrangeira;
- Classes que contém colecções. Deve ser alocado um identificador de objecto à classe contida na colecção. Esta classe será representada por uma tabela. Deve ser criada uma tabela que contém duas colunas, a primeira com o identificador dos objectos que contém a colecção, a segunda com o identificador dos objectos contidos na colecção;
- Associações de um-para-muitos podem ser tratadas como colecções;
- Associações de muitos-para-muitos tornam-se tabelas separadas. Cria-se uma tabela com duas colunas. Cada linha contém um par de identificadores de atributos, um de cada objecto participante na associação;
- Associações de um-para-um são implementadas como chaves estrangeiras. Cada classe ganha um atributo extra para carregar o identificador do objecto associado.

Para identificação das tabelas usam-se chaves, que são os atributos responsáveis pela identificação destas. As chaves podem ser primárias ou secundárias.

Uma **Chave Primária** é definida como um conjunto "k" de atributos verificando: unicidade (os valores de chave Primária são únicos e não nulos) e minimalidade (nenhum atributo componente de k pode ser retirado sem perda da unicidade) (Rodrigues, 2004).

A **Secundária** é usada em base de dados relacional para criar relações entre tabelas. Porque os objectos não têm chaves, então são alocados identificadores para eles. Quando se usa uma base de dados relacional, as colecções de classes que existem só para aceder a um conjunto de objectos da mesma classe não precisa fazer parte dos mesmos dados que são guardados nas tabelas (Bennet et al, 2002).

Anexo F – Guião de Entrevistas

No âmbito da automatização do Sistema de Gestão de Docentes do DMI, foram levadas a cabo a cinco entrevistados. Dentre eles, docentes e chefes das três secções do DMI com vista a apurar-se as suas opiniões, sugestões e críticas em relação a implementação do modelo proposto. As entrevistas consistiram na captura de informação sob forma de entrevistas abertas tendo-se entrevistado por vezes mais de uma vez aos envolvidos com o intuito de obter a mais correcta informação sobre o funcionamento das actividades referentes aos docentes.

Para tal, estão listadas abaixo, diversas questões colocadas ao longo das entrevistas:

1. Pode falar no geral do ciclo de vida de um Docente?

_____

_____

_____.

2. Quais são os requisitos necessários para ser Docente?

_____

_____.

3. Quando um Docente é contratado a primeira vez quais são as fases pelas quais ele passa até que ele leccione?

_____

_____

_____.

4. Como é feito o plano anual de actividades de um Docente, e quantas Disciplinas um docente pode leccionar durante um período de aulas (semestre)?

_____

_____

_____.

5. Um docente que é supervisor de um candidato a licenciatura, este pode usar o trabalho deste como uma actividade científica?

6. A elaboração de materiais auxiliares como manuais e textos de apoio, que servem de apoio aos estudantes, afectam de alguma forma no perfil do Docente?

7. O que se pretende com trabalho de extensão programado? (formulário anual de actividades)?

8. Que outras actividades um Docente pode praticar?

9. O que gostaria que este Sistema de Gestão de Docentes do DMI em por desenvolver disponibilizasse em termos de relatórios?

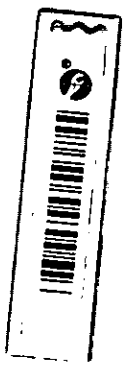
10. Mais alguma coisa para reforçar?

Manual do Utilizador:

Sistema de Gestão de Docentes do DMI Via Web

Anexo G – Manual do Utilizador

Nesta secção do documento, descreve-se o funcionamento do sistema através do manual do utilizador com vista a facilitar a navegação na aplicação e facilitar a compreensão dos utilizadores do sistema.



G.1 – Introdução

O Sistema de Gestão de Docentes do Departamento de Matemática e Informática - DMI foi concebido para facilitar os diversos processos decisórios que refere a **actividade docente**, bem como o seu cadastro no sistema e a geração de relatórios afins. Estão previstos como utilizadores deste sistema, o docente, os chefes de secção e o director do curso (visualiza relatórios).

Com a ajuda do sistema produzido, os Chefes de Secção, que são os utilizadores com maior número de opções de funcionalidades (permissões) no sistema, poderão, introduzir informação referente a docente (cadastrar, actualizar e remover docente), actividades relativas a docentes, visualizar relatórios contendo as estatísticas referentes a docentes e diversas outras opções. Outros utilizadores mas, com menos permissões no sistema podem introduzir e actualizar as actividades realizadas por docentes, é o caso do Docente e do Director de Curso que poderão visualizar relatórios contendo informação sobre docentes.

O sistema é composto por três módulos, o primeiro onde introduz-se, altera-se e remove-se os dados pessoais do docente, o segundo onde introduz-se e actualiza-se as actividades relativas a docentes e o último onde se gera e se visualiza relatórios referentes a docente - actividades realizadas.

Os dois primeiros módulos foram implementados com sucesso e, o último tendo sido implementado só uma parte (relatório de docentes existentes). Atendendo que a aplicação desenvolvida tem uma estrutura modular, devido a tecnologia de desenvolvimento usada, (o Java Server Pages - JSP), esta poderá ser facilmente continuada por outro(s) desenvolvedor(s) de sistemas de informação.

Este manual possui informação útil para todos os utilizadores do sistema, de modo a facilitar o ensino-aprendizagem do funcionamento do sistema.

G.2 – Estrutura do Sistema

O Sistema de Gestão de Docentes do DMI foi concebido para ser usado pelos seguintes utilizadores:

- Administrador do Sistema;
- Chefes de Secção;
- Docentes;
- Director do Curso.

O administrador do sistema é o responsável pela manutenção da aplicação e portanto possui acesso a todas as páginas desta. Este usuário não participa activamente nas actividades do sistema, estando sempre pronto a intervir quando surgem imprevistos. Os restantes usuários, como é o caso dos chefes de secção, dos docentes e director do Curso, possuem privilégios definidos segundo as suas actividades dentro do sistema.

Para que os usuários acedam a página inicial do sistema devem digitar na barra de endereço do browser o endereço seguinte:

https://localhost/GD/jsps/indice.html

A figura abaixo (figura G 1) ilustra a Página Inicial do Sistema.

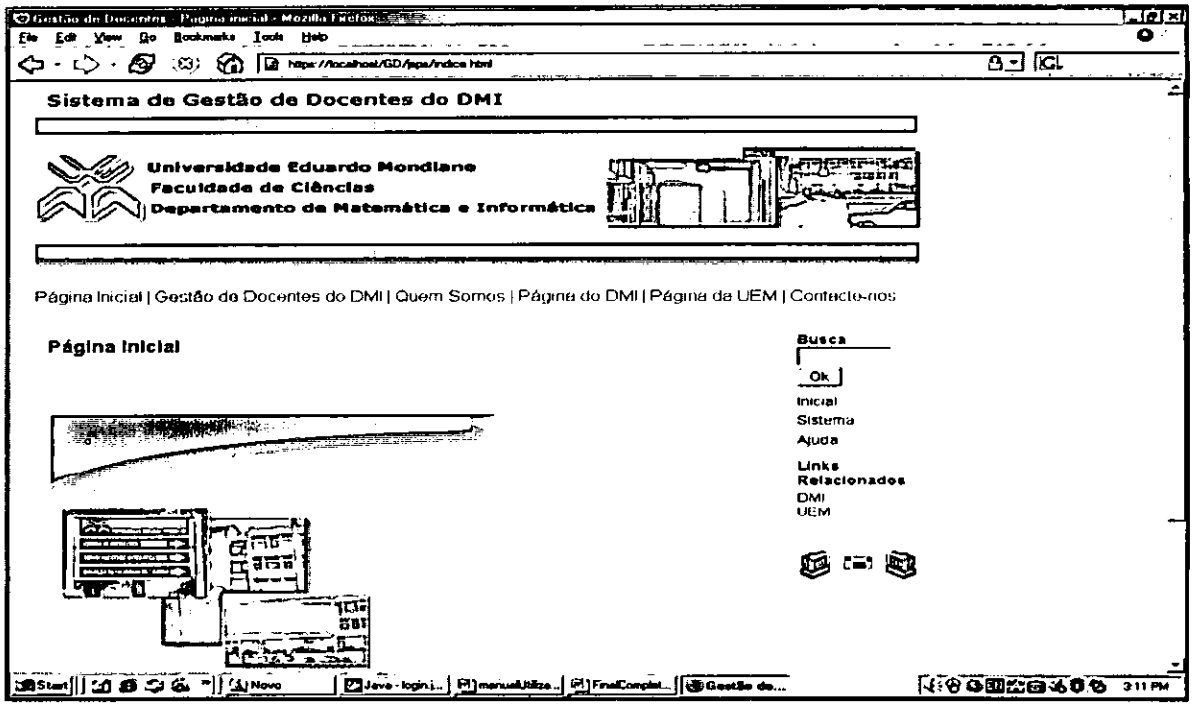


Figura G 1 - Página Inicial do Sistema de Gestão de Docentes do DMI

Para aceder a secção referente à Gestão de Docentes deverá clicar na opção *Gestão de Docentes do DMI* e, aparecerá a janela ilustrada na figura G.2 com uma breve descrição do Sistema.

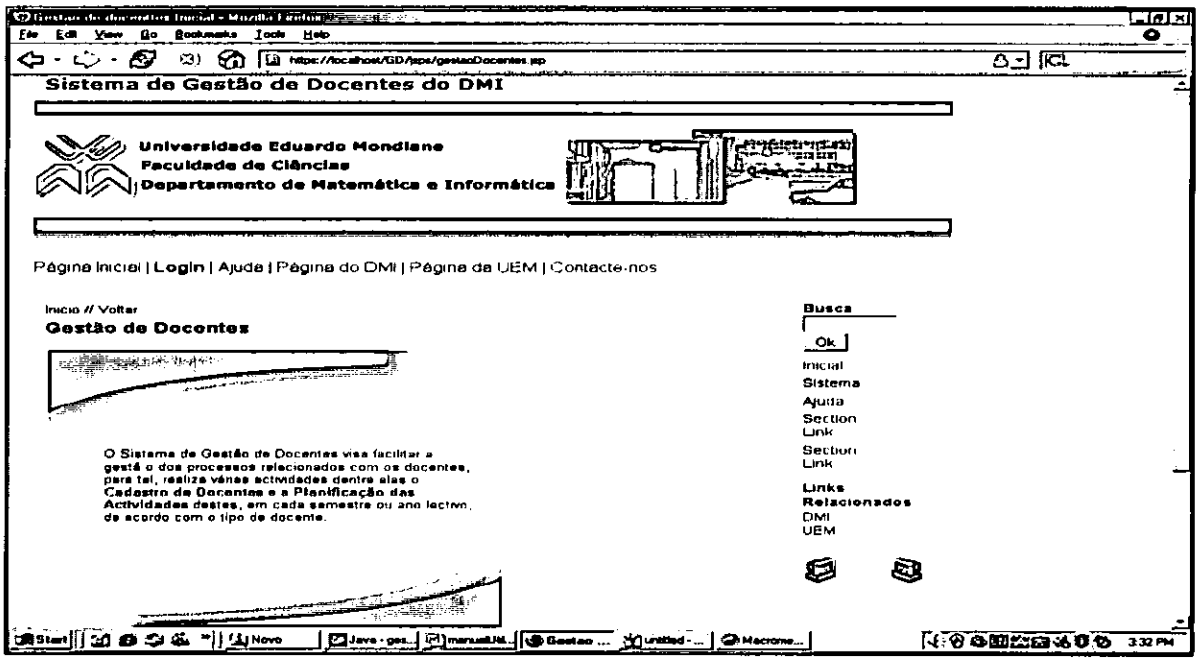


Figura G 2 - Página de Entrada na Secção de Gestão de Docentes do DMI

Para aceder as diversas funcionalidades do Sistema só um usuário autenticado deverá fazê-lo, para tal deverá clicar na opção *Login,* de onde aparecerá uma janela onde ele deverá digitar os seus dados de acesso "Usuário" e "Senha", como mostra a figura G 3. Após digitar os seus dados de autenticação, ele deverá clicar no botão *Entrar* para aceder as opções a que tenha acesso.

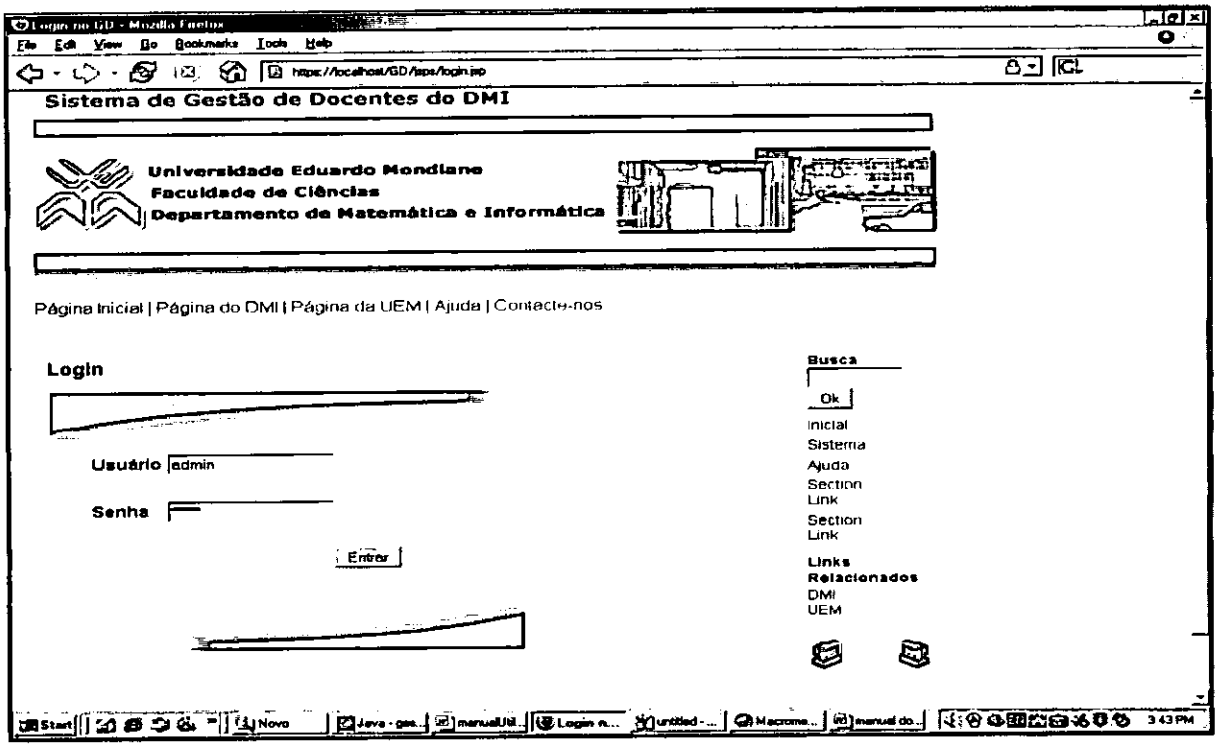


Figura G 3 - Página de Autenticação de Usuários

Como já se disse anteriormente, o administrador é o usuário que faz a manutenção do sistema portanto, possui todas as funcionalidades que o sistema dispõe porém, ele não participa nas actividades do sistema.

Por esta razão, não se vai descrever as suas opções considerando que estas serão descritas implicitamente através dos outros usuários.

G 3 – Permissão por Usuário

O sistema foi desenhado e implementado tendo em conta as diferentes actividades disponíveis a cada tipo de usuário.

A seguir estão descritas as actividades referentes aos Chefes de Secção por este ser o actor com maior número de permissões no Sistema:

Chefe de Secção

Cada Chefe de Secção é responsável pela manutenção dos docentes (introdução, edição e remoção dos dados dos docentes) e planificação das suas actividades. Eles poderão também gerar relatórios relacionados com docentes. A figura a baixo mostra as opções relativas ao chefe de secção.

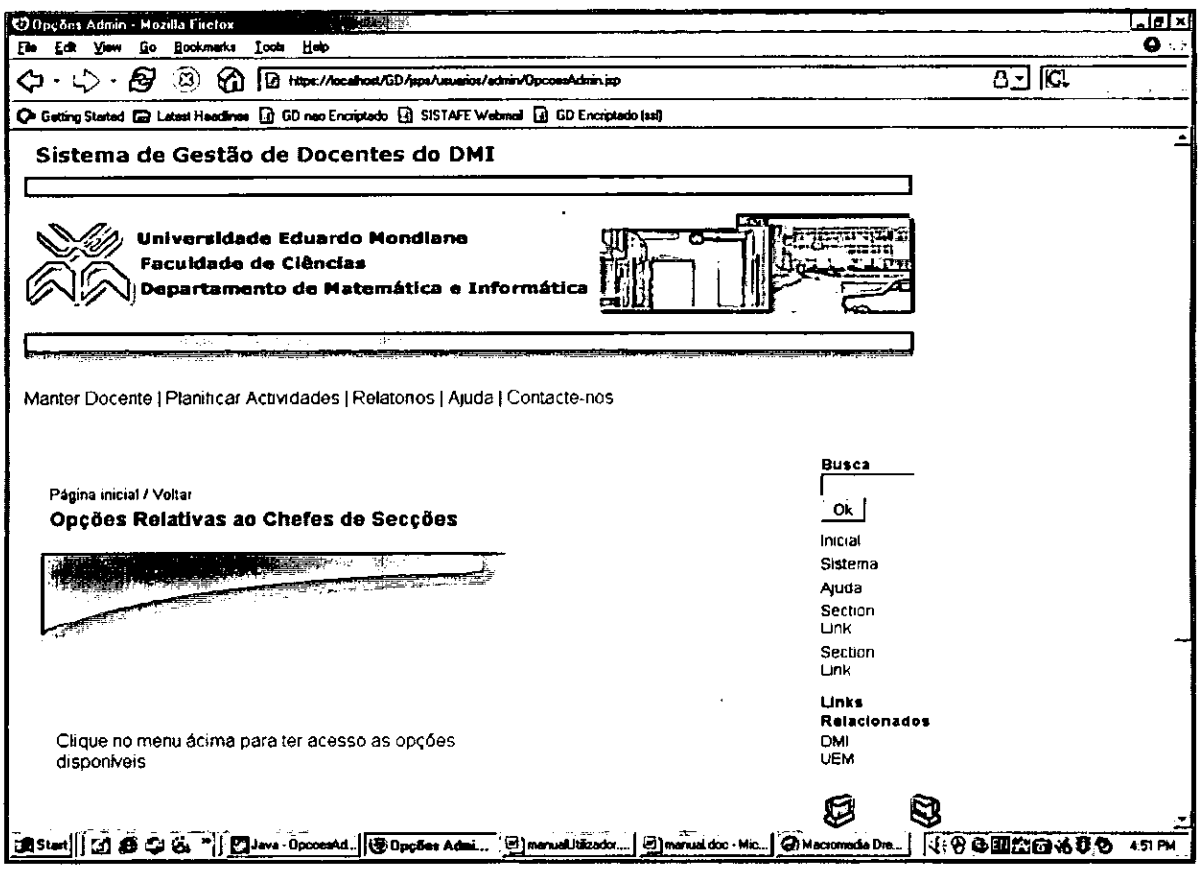


Figura G 4 - Opções Relativas aos Chefes das Secções

Este se desejar introduzir, editar ou remover dados referentes a um docente poderá fazê-lo clicando na opção *Manter Docente* e, aparecerá uma janela – figura G 5, contendo as opções, *Cadastrar* – para introduzir dados do docente, *Editar* – para alterar os dados de um docente no sistema e *Remover* – que remove os dados dos docentes.

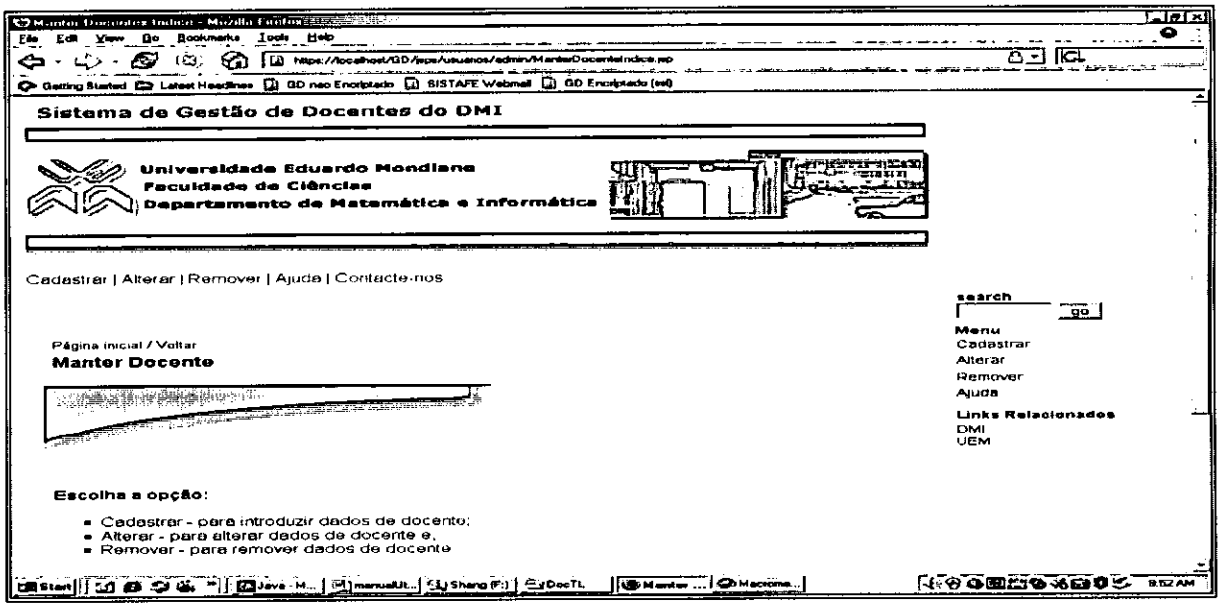


Figura G 5 - Opções Para Manter Docente

Se clicar na opção *Cadastrar,* abrirá uma janela contendo um formulário para introdução de dados de docente como mostra a figura a seguir:

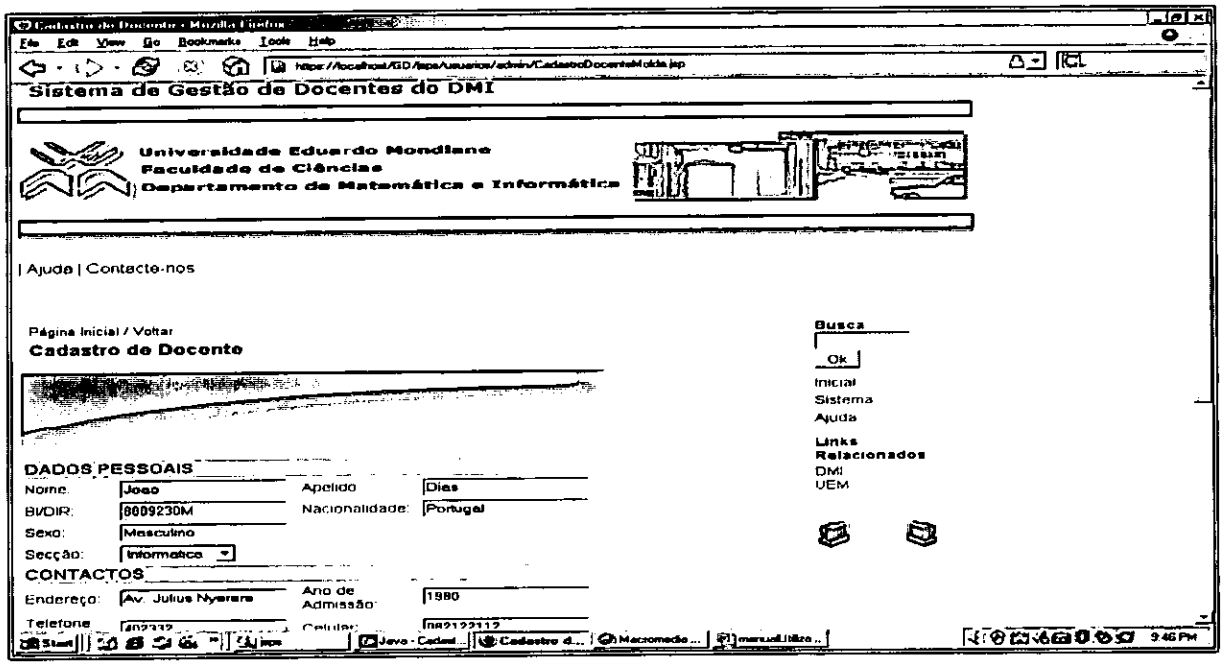


Figura G 6 - Cadastro de Docente

O usuário deverá introduzir os dados do docente e clicar no botão *Submete*" e, o sistema envia uma mensagem de sucesso caso os dados tenham êxito na sua submissão para base de dados.

Para além de se cadastrar docentes, o sistema permite a alteração dos seus dados e também a sua remoção. Devido a similaridade destas funcionalidades não serão ilustradas para o caso de Manutenção de Docentes tendo-se ilustrado para outros casos como por exemplo da alteração e remoção na planificação das actividades dos docentes.

Assim sendo, se o usuário desejar planificar as actividades dos docentes deverá clicar no botão *Planificar Actividades* da figura G 4 e, abrirá uma janela como mostra a figura G 7.

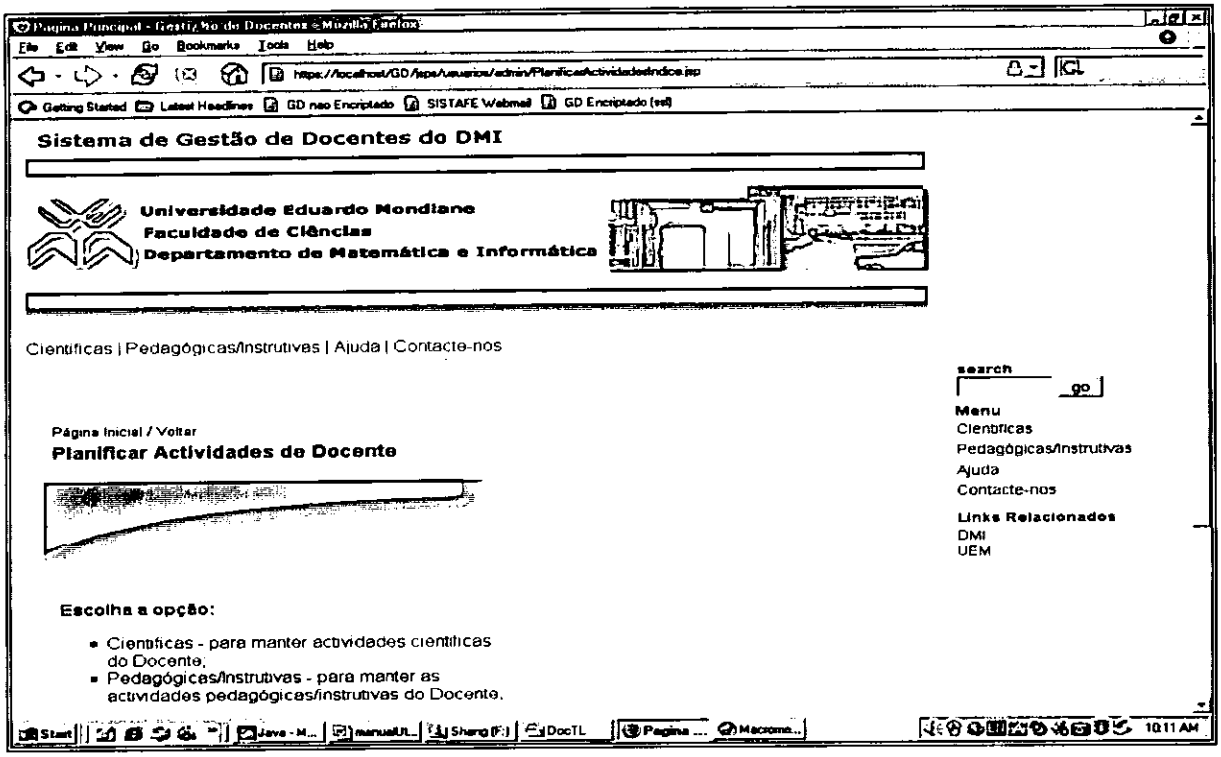


Figura G 7 - Opções de Planificação de Actividades dos Docentes

Para a manutenção das actividades científicas dos docentes, o usuário deverá clicar na opção *Científicas* e terá acesso a página como mostra a figura G 8.

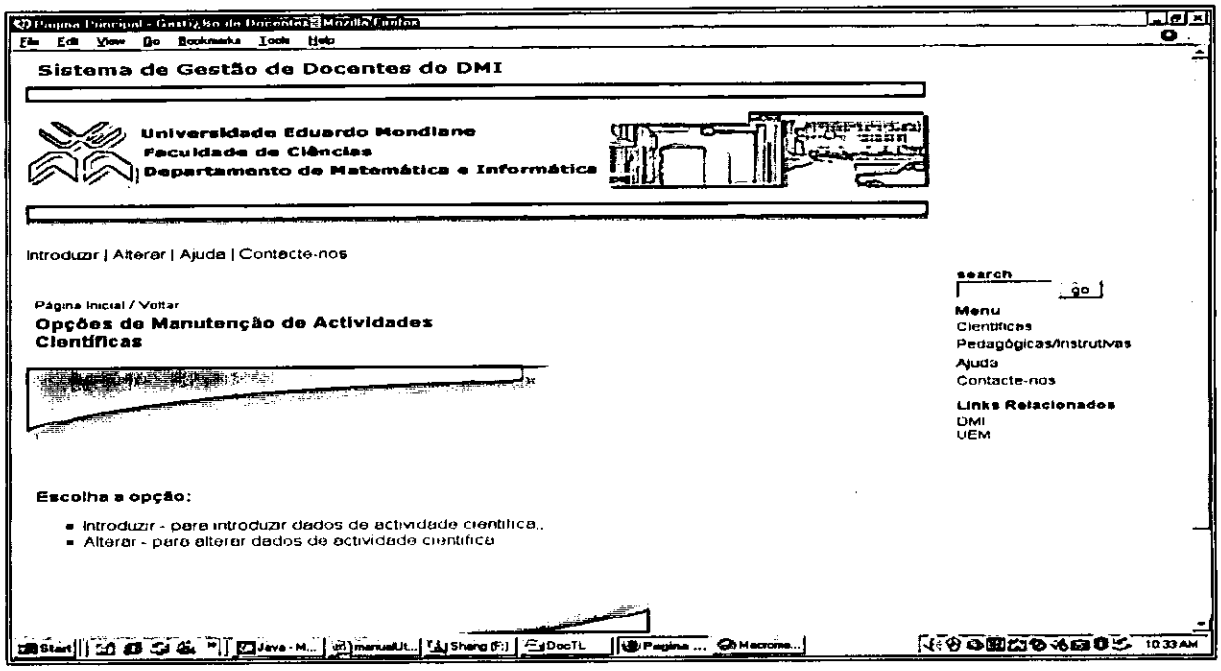


Figura G 8 - Manutenção de Actividades Científicas dos Docentes

Para introduzir dados de actividade científica deverá clicar na opção *Introduzir* e o sistema abrirá uma janela com um formulário de introdução de actividade como mostra a figura a seguir:

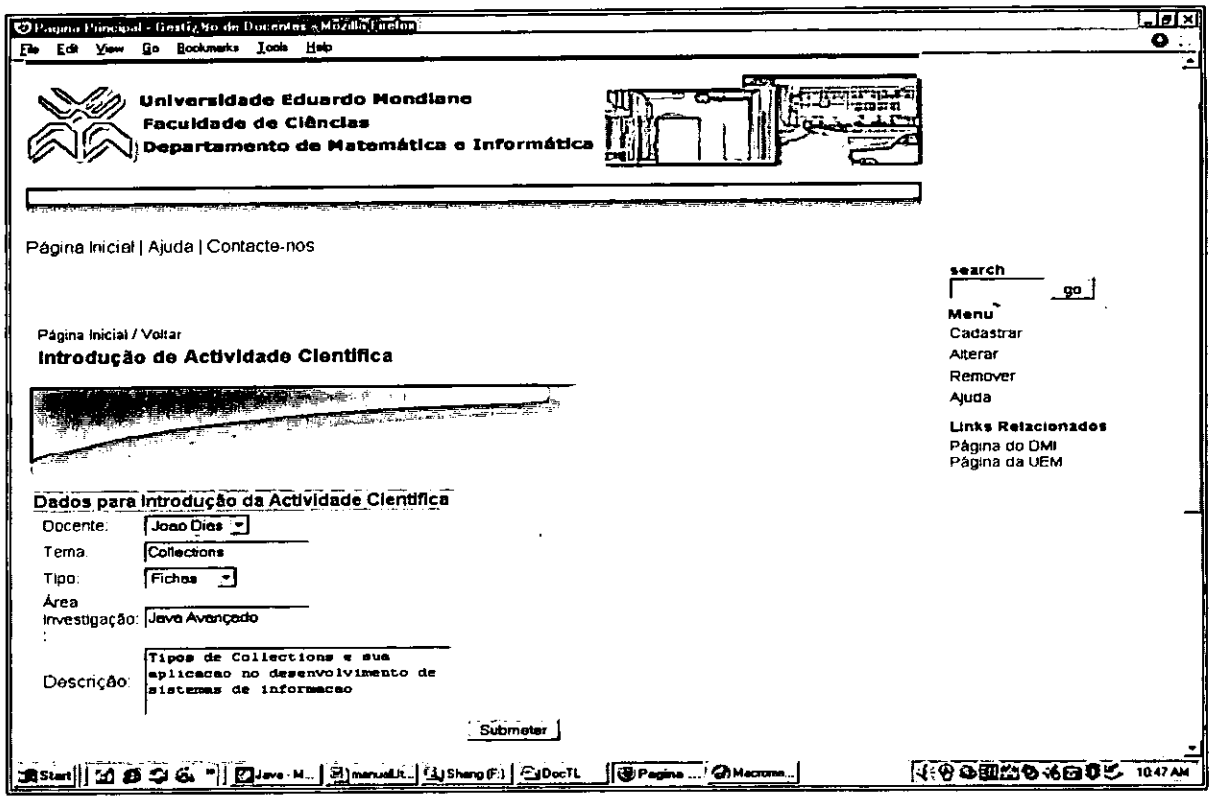


Figura G 9 - Introdução de Actividade Cientifica

O usuário deverá preencher os campos do formulário e após terminada a introdução dos dados deverá clicar no botão *Submeter* e o sistema dará uma mensagem de êxito caso os dados tenham sido introduzidos com sucesso à Base de Dados.

Se desejar alterar a actividade científica introduzida o usuário poderá fazê-lo clicando no botão *Alterar* da figura G 8 e, o sistema abrirá uma janela pedindo o nome do tema da actividade científica por alterar. O usuário deverá escolher a actividade desejada e clicar o botão *Próximo* como mostra a figura a seguir:

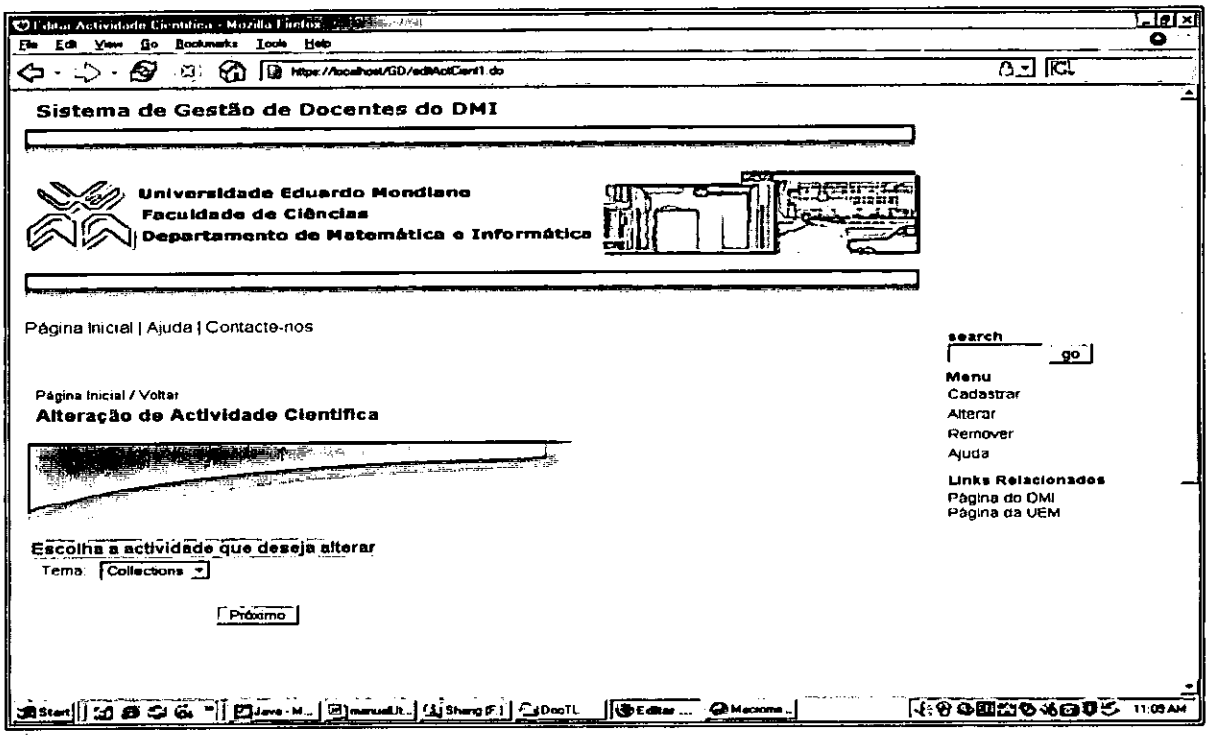


Figura G 10 - Escolha de Tema de Actividade Científica para Alterar

Após o usuário ter clicado o botão *Próximo*, o sistema abrirá uma página com todos os detalhes da actividade seleccionada onde, o usuário poderá alterar os campos que desejar e clicar no botão *Submeter* como mostra a figura G 11:

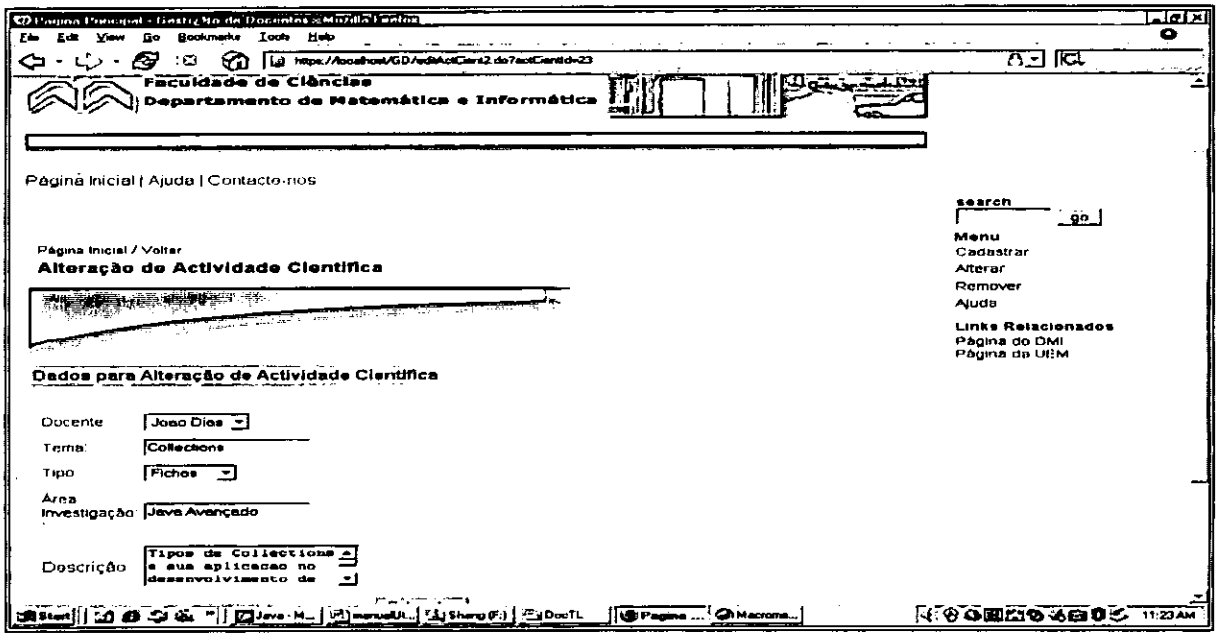


Figura G 11 - Alteração de Actividade Científica

Depois de clicar no botão *Submeter,* o sistema mostra uma mensagem de sucesso caso os novos dados tenham sido enviados com sucesso à Base de Dados.

Para fazer a manutenção das actividades instrutivas, o usuário poderá fazê-lo bastando clicar na opção *Pedagógicas/Instrutivas* da figura G 7 e aparecerá uma janela como mostra a figura G 12.

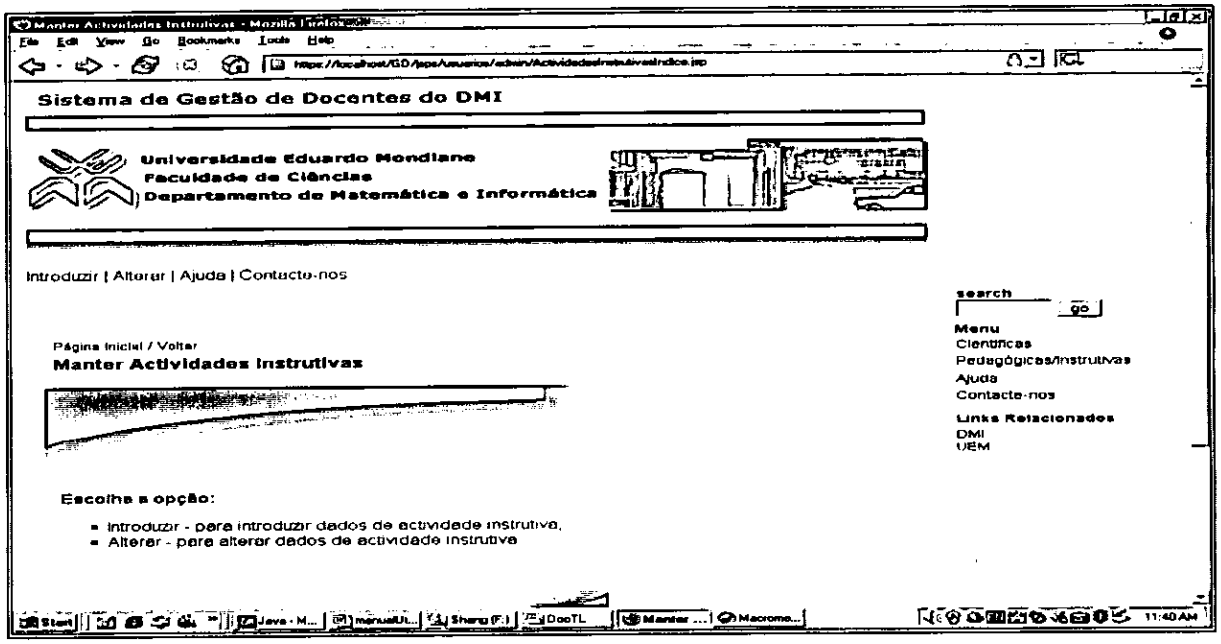


Figura G 12 - Manter Actividades Instrutivas

Se desejar introduzir uma nova actividade instrutiva o usuários deverá clicar na opção *Introduzir* e, o sistema abrirá uma janela com um formulário de dados para a actividade instrutiva como mostra a figura seguinte:

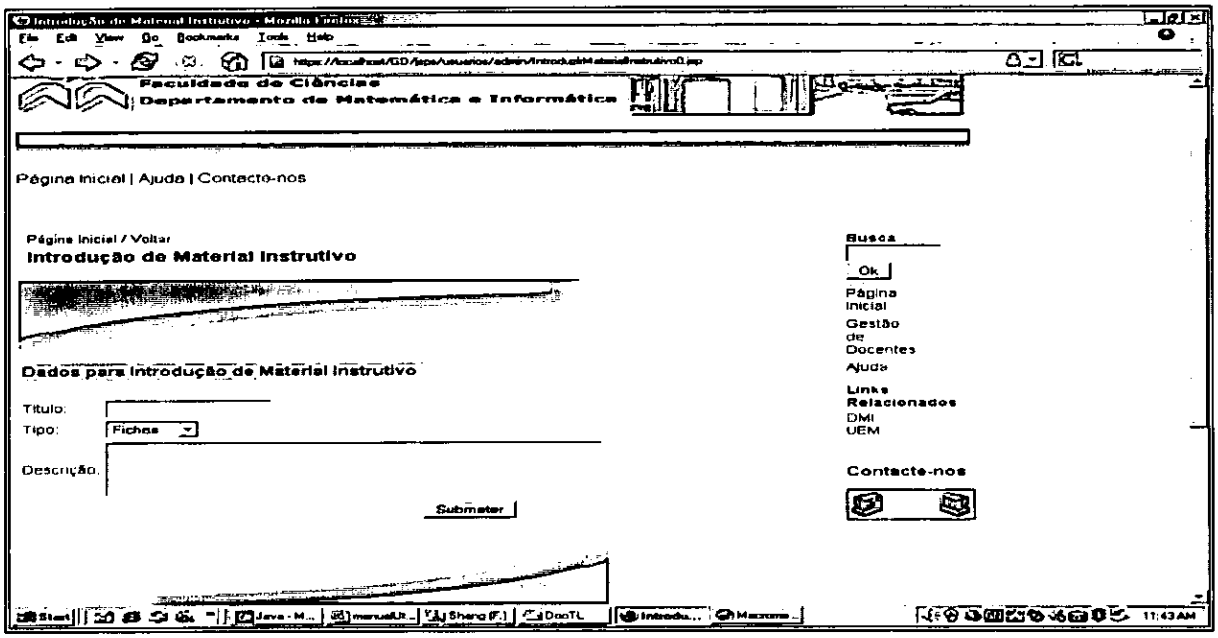


Figura G 13 - Introdução de Actividade Instrutiva

O usuário introduz os dados e clica no botão *Submeter* e o sistema abrirá uma janela onde o deverá escolher o nome ou nomes de Docentes que elaboraram esta actividade como mostra a figura:

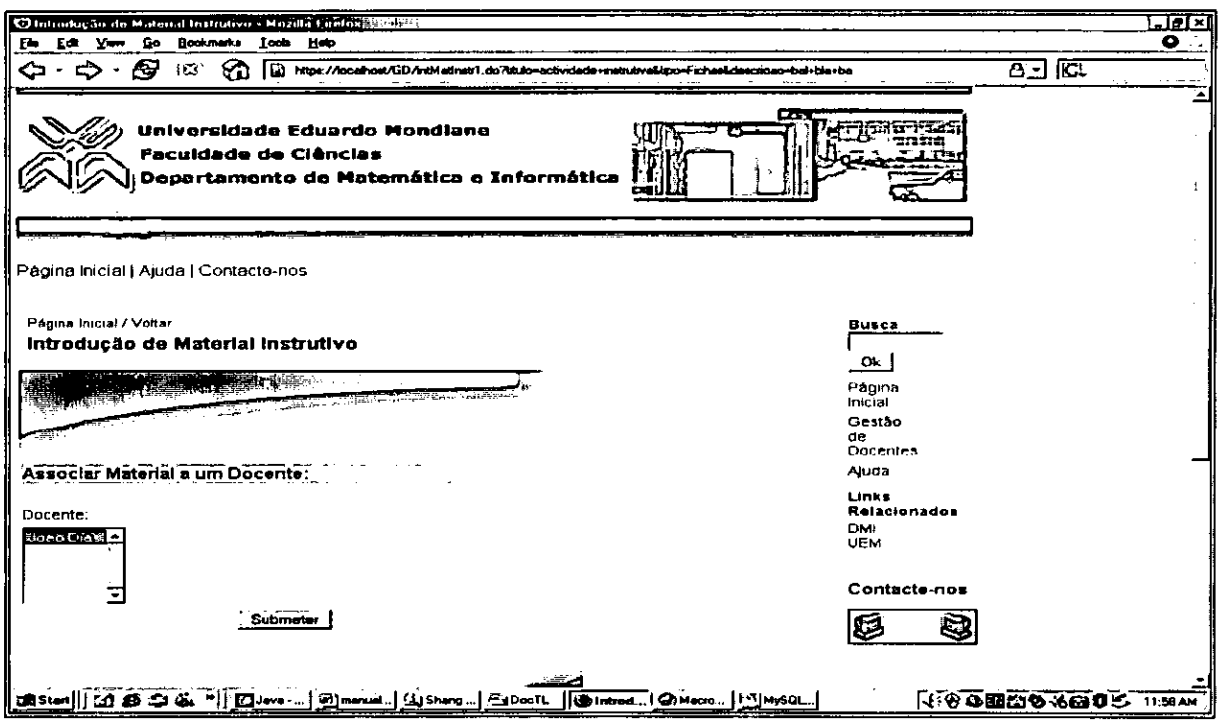


Figura G 14 - Associar Actividade a Docente

A seguir o usuário deverá clicar no botão *Submeter* e, o sistema dará uma mensagem de êxito caso os dados tenha sido enviados com sucesso à Base de Dados.

Tal como as actividades cientificas, o sistema permite também alterar os dados das actividades instrutivas introduzidas existentes na Base de Dados para tal, o usuário deverá clicar na opção *Alterar* na página da figura G12 e, o sistema abrirá uma janela (figura G15) onde deverá escolher a actividade instrutiva por alterar e clicar no botão *Próximo*:

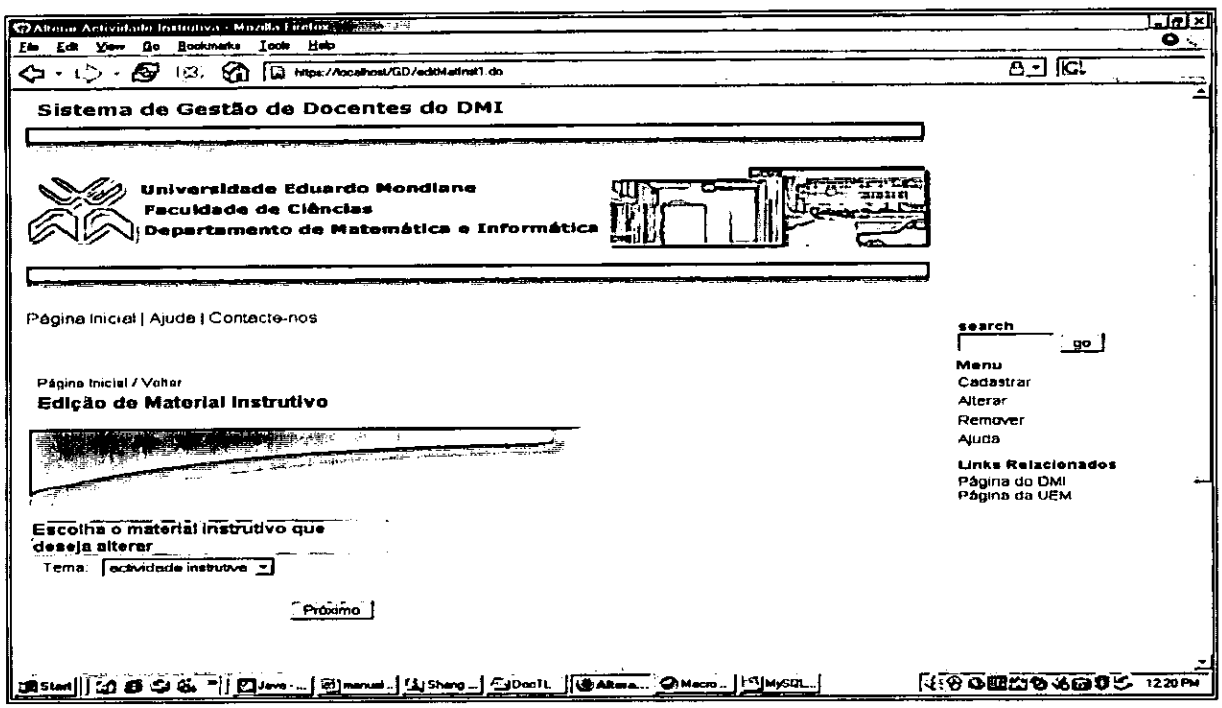


Figura G 15 - Escolha da Actividade Instrutiva por Alteração

Depois de clicar no botão *Próximo* o sistema abrirá uma janela (figura G16) com um formulário contendo todos os detalhes da actividade escolhida e, o usuário deverá alterar os campos desejados e clicar no botão *Submeter*.

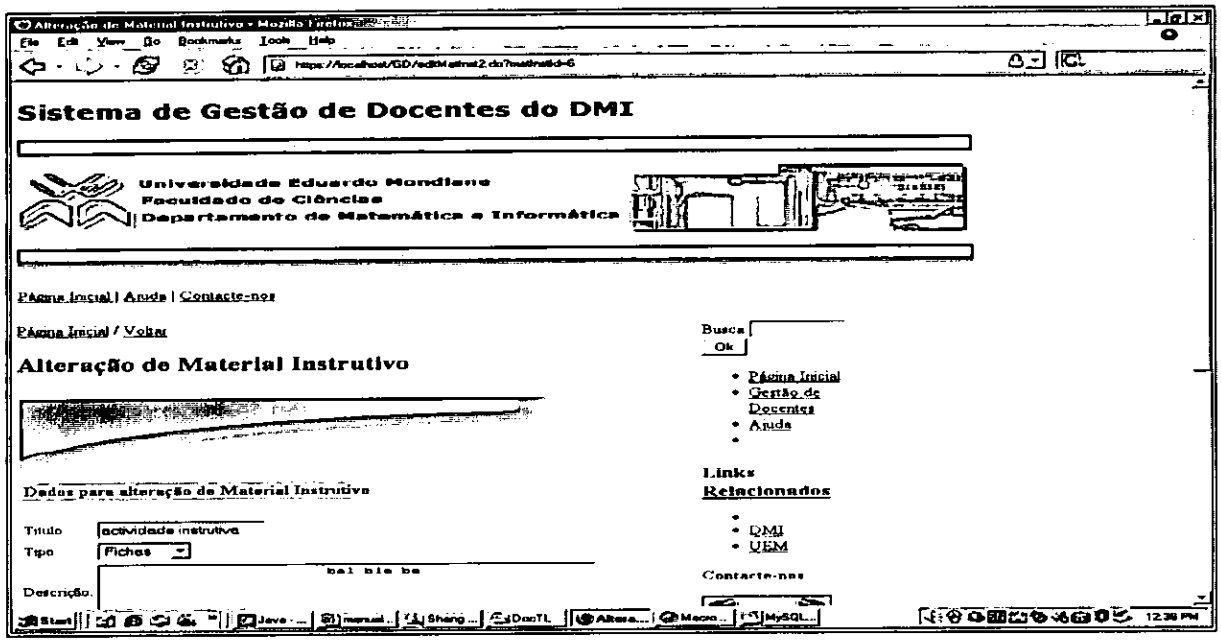


Figura G 16 - Alteração da Actividade Instrutiva

Após alterar os campos desejados, o usuário deverá clicar no botão *Próximo* e o sistema abrirá uma janela onde deverá escolher um docente para associar a esta actividade e, clicar no botão *Submeter* conforme mostra a figura seguinte:

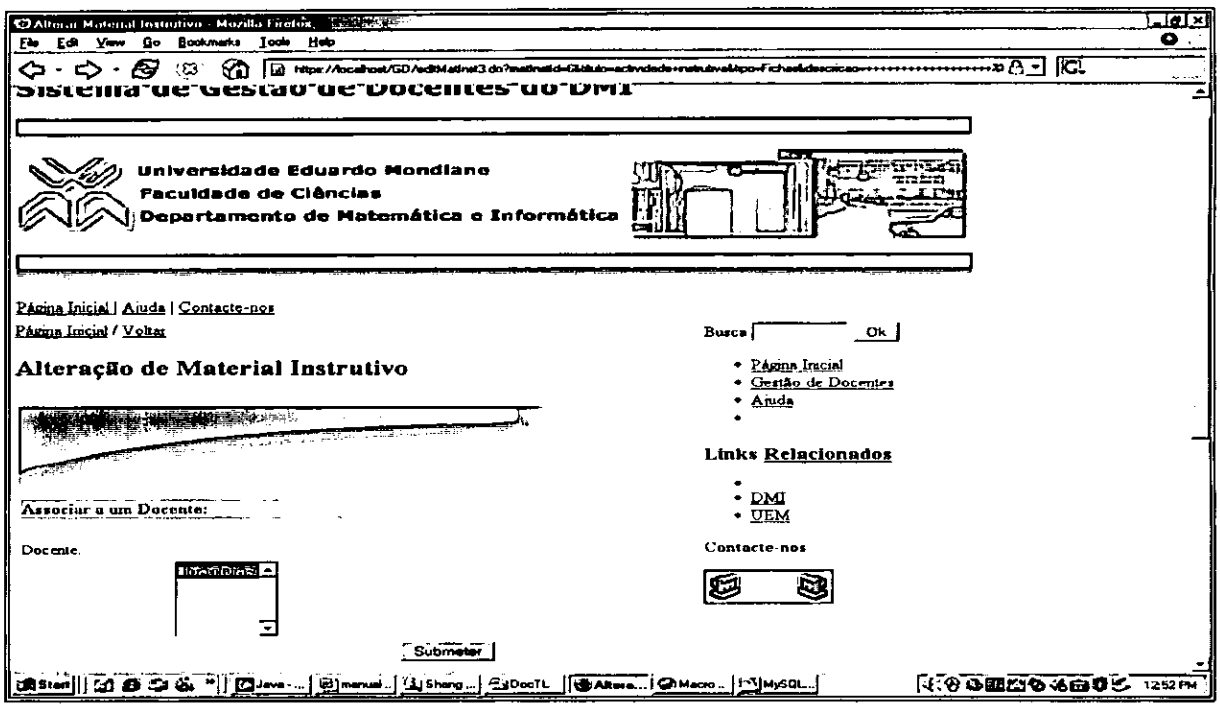


Figura G 17 - Associar Docente a Actividade Instrutiva Alterada

Após clicar o botão *Submeter*, o sistema envia uma mensagem de êxito se os novos dados tiverem sido enviados com sucesso à Base de Dados.

A última funcionalidade que este sistema oferece é, a de gerar relatórios sobre os docentes e as suas actividades, por estes serem diversificados apresenta-se aqui um dos relatórios que o sistema disponibiliza.

Para gerar relatórios, o usuário deverá clicar na opção *Relatórios* na página da figura G 4 e o sistema apresentará um menu de opções de relatórios disponíveis que neste caso mostra de Docentes (figura G 18).

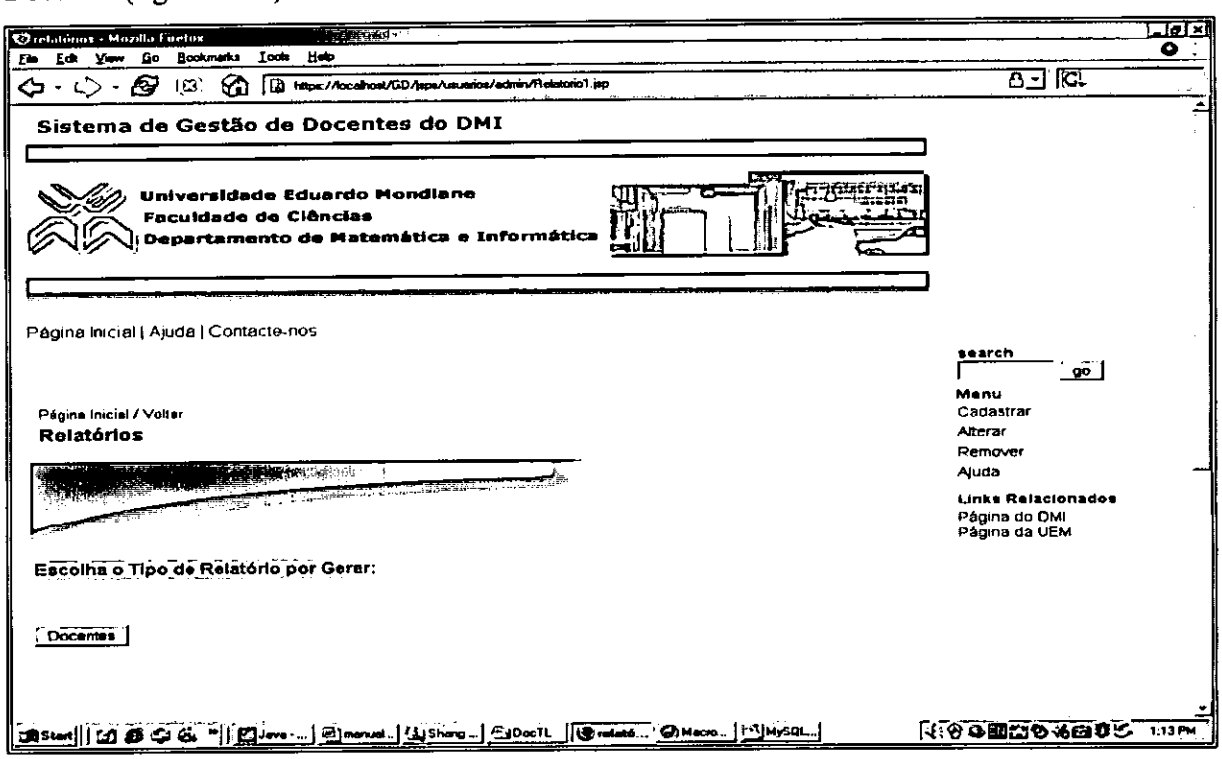


Figura G 18 - Opções de Relatórios

Para gerar o relatório de docentes o usuário deverá clicar no botão *Docentes* e, o sistema gerará e abrirá uma janela contendo este relatório como mostra a figura G 19.

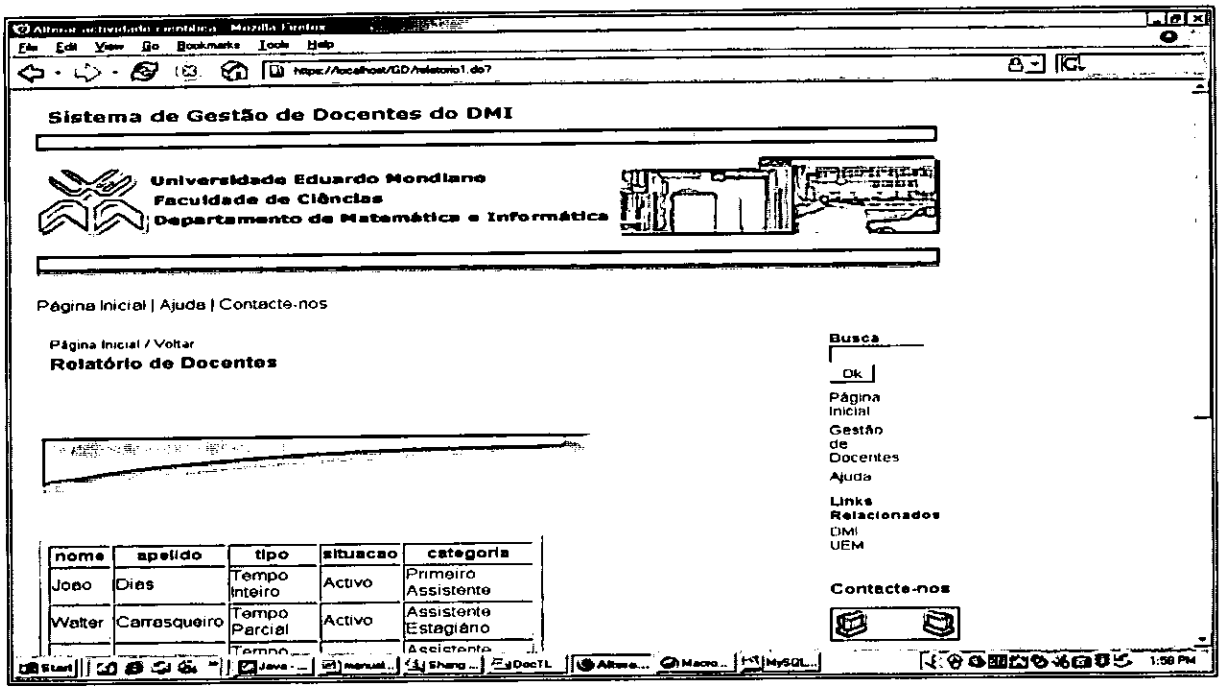


Figura G 19 - Relatório de Docentes