ESCOLA POLITÉCNICA
DE PERNAMBUCO

d.s.c. UPE
Departamento de Sistemas e Computação

DESENVOLVIMENTO E IMPLANTAÇÃO DE SISTEMA DE BANCO DE CRÉDITOS E AVALIAÇÃO DE DESEMPENHO DE DOCENTES

Trabalho de Conclusão de Curso

Engenharia da computação

Ailton Francisco de Sousa Junior
Orientador: Tiago Massoni

Recife, maio de 2008

UNIVERSIDADE
DE PERNAMBUCO

ESCOLA POLITÉCNICA
DE PERNAMBUCO

d.s.c. UPE
Departamento de Sistemas e Computação

DESENVOLVIMENTO E IMPLANTAÇÃO DE SISTEMA DE BANCO DE CRÉDITOS E AVALIAÇÃO DE DESEMPENHO DE DOCENTES

Trabalho de Conclusão de Curso

Engenharia da computação

Este Projeto é apresentado como requisito parcial para obtenção do diploma de Bacharel em Engenharia da computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Ailton Francisco de Sousa Junior
Orientador: Tiago Massoni

Recife, maio de 2008

UNIVERSIDADE
DE PERNAMBUCO

Ailton Francisco de Sousa Junior

DESENVOLVIMENTO E IMPLANTAÇÃO DE SISTEMA DE BANCO DE CRÉDITOS E AVALIAÇÃO DE DESEMPENHO DE DOCENTES

Resumo

O Departamento de Sistemas e Computação (DSC) da Escola Politécnica de Pernambuco vem crescendo bastante ultimamente. Com esse crescimento, está cada vez mais difícil manter o controle de atividades dos docentes e fazer avaliações periódicas. Alguns docentes tem a necessidade de se afastar, periodicamente, do Departamento. Só que este afastamento tem que ser planejado, pois alguém terá que substituí-lo. Desta forma, o docente tem que adiantar algumas horas de trabalho para poder se afastar. O sistema proposto vem para sanar as dificuldades encontradas na hora de avaliar os professores, no momento de escolher qual docente tem prioridade para se afastar, além de facilitar a gerência dos docentes e do departamento. Para isso foram desenvolvidos dois sistemas que se integram. Um sistema de banco de créditos que ajudará na escolha de qual professor terá prioridade de se afastar do departamento. E o outro sistema para fazer as avaliações dos docentes e gerenciar o departamento. Com os sistemas integrados, iremos implantar o mesmo no servidor do DSC.

Abstract

The Department of Systems and Computer (DSC) of the Polytechnic School of Pernambuco is growing quite recently. With this growth, is increasingly difficult to maintain the control of activities for teachers and make periodic assessments. Some teachers have the need to depart, periodically, the Department. Only that this removal has to be planned, because someone will have to replace it.This way,the teacher has to advance work for a few hours of power to depart. The proposed system is to ,remedy the difficulties encountered in time to evaluate teachers at the time of choosing which teacher has priority for departing, as well as facilitate the management of teachers and the department.For that were developed two systems that integrate. A system of bank credits to assist in the choice of which teacher will have priority to depart from Department. And the other system to make the evaluations of teachers and manage the Department. With integrated systems, we will deploy the same server in the DSC.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

MVC *Model-View-Controller* (Modelo-Visão-Controle).

UML *Unified Modeling Language* (Linguagem de Modelagem Unificada).

Web Equivalente ao WWW que significa *Word Wide Web*.

RUP *Rational Unified Process*.

SGBD Sistema Gerenciador de Banco de Dados.

URL *Uniform Resource Locator* (Localizador Uniforme de Recursos).

HTML *Hiper Text Mark-up Language* (Linguagem de Marcação de Hiper Texto).

TCP *Transmission Control Protocol* (Protocolo de Controle de Transmição).

UDP *User Datagram Protocol*.

JSP *Java Server Pages*.

HTTP *Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto).

CSS *Cascading Style Sheet* (Folha de Estilo em Cascata).

J2EE *Java2 Platform Enterprise Edition*.

IDE *Integrated Development Enviroment.*

RHDS *Red Hat Developer Studio*.

Agradecimentos

Gostaria de agradecer, primeiramente, a Deus, por ser ele o guia e referência de minha vida. Agradeço a meus pais (Ailton e Marlete) e minha irmã (Vivianne), pois sempre depositaram total confiança em mim e estavam sempre ao meu lado me ajudando quando foi preciso, além disso, são responsáveis pela pessoa que sou hoje. Agradeço ainda a meu tio (Raminho) que foi quem proporcionou a minha vinda para Recife, pois moro com ele.

Agradeço também a todos que formam o DSC pelo apoio, pela vontade de ajudar sempre e pela excelente formação. Um agradecimento especial aos professores Tiago Massoni (meu orientador no trabalho de conclusão de curso) e Cristine Gusmão (minha orientadora de estágio supervisionado). Agradeço também a todos os amigos que fiz na faculdade. Sei que sem todos vocês que me apoiaram, nada disso seria possível. Obrigado a todos!

Capítulo 1

Introdução

Existe um processo de avaliação no Departamento de Sistemas e Computação (DSC) [1], porém o método de avaliação é trabalhoso, pois é feito de modo individual, através da análise de planilhas com dados de cada docente. Esse processo de avaliação é muito importante já que é ele quem pode definir qual docente terá o direito de se afastar, por um período, do departamento. Além disso, é através da avaliação que se tem o controle das atividades realizadas pelos docentes e também é averiguado se as metas de cada docente foram alcançadas. Cada atividade tem uma pontuação atribuída. Chegando ao final do período, cada docente deve atingir uma pontuação mínima. Por exemplo, ao término do período, cada docente tem que ter atingido 45 pontos, que equivalem a lecionar três disciplinas com carga horária de 60 horas (valores fictícios). No decorrer desse trabalho, iremos detalhar tais atividades.

O (DSC) da Escola Politécnica de Pernambuco (POLI) [2] vem crescendo bastante ultimamente. Com esse crescimento, está cada vez mais difícil manter o controle de atividades dos docentes e de fazer avaliações periódicas. O sistema vem para sanar essa dificuldade e facilitar o trabalho dos docentes nesses quesitos citados acima.

Em determinados momentos, alguns docentes tem a necessidade de se afastar, periodicamente, do departamento, isso pode acontecer quando o mesmo precisa se dedicar a um doutorado ou a um pós-doutorado. Esse afastamento tem que ser planejado para que o Departamento consiga manter as disciplinas de maneira organizada. É preciso que algum docente o substitua e que o docente adiante algumas horas de

[1] http://www.dsc.upe.br
[2] http://www.upe.poli.br/portal2/

trabalho pelo tempo que o mesmo vai ficar ausente. Essas horas adiantadas irão constar no banco de créditos e serão consultadas na hora da avaliação dos docentes. Por exemplo, um docente precisa se afastar e já comunicou ao coordenador do DSC, o docente vai ensinar algumas disciplinas a mais ou exercer alguma das atividades do Departamento nos períodos antes de seu afastamento. Essas horas adiantadas constarão no banco de créditos e serão consultadas na hora do afastamento do docente.

Como foi falado acima, o método atual de avaliação é muito trabalhoso. Foi daí que surgiu a idéia de fazer dois sistemas que tenham as informações integradas, que são chamados de sistemas de informação [1]. Sistemas de informação são muito úteis, pois coletam, processam e retornam dados que representam informações para o usuário. O primeiro sistema possibilitará a gerência das atividades desenvolvidas pelos docentes para a avaliação dos mesmos. O segundo sistema permitirá organizar uma fila de docentes que desejem se afastarem do departamento pelos motivos já citados. Então esses sistemas se integram para que na hora de decidir qual docente tem prioridade de se afastar, o sistema veja quem está na fila de espera, avalie o desempenho do docente e decida quem possui prioridade para se afastar. O sistema possui como usuários os docentes e o coordenador (que também é um docente, só que com alguns privilégios a mais no sistema).

Então esse projeto tem como meta fazer uma re-avaliação (com a participação do coordenador do DSC) nesse sistema que começou a ser desenvolvido no semestre anterior, visa também fazer as devidas melhorias pós-avaliação e desenvolver mais algumas iterações para que o sistema de avaliação e banco de crédito seja implantado no servidor do DSC para a utilização. Na re-avaliação serão testadas as funcionalidades existentes, serão definidas melhorias e a necessidade de novas funcionalidades.

1.1 Objetivos

Umas das maneiras mais eficientes de se trabalhar com dados hoje, é através de sistemas da informação. Como o sistema já vinha sendo desenvolvido desde o semestre passado, tem-se como objetivo fazer uma reavaliação no sistema de informação [1] que está sendo desenvolvido e consequentemente fazer os possíveis ajustes e melhorias para que o sistema possa ser implantado no DSC. Desenvolver no sistema, uma

funcionalidade que integrará o mesmo com os dados do sistema utilizado pela Escola Politécnica, que é o SIGA [3] , com o intuito de facilitar e automatizar o uso do sistema pelos docentes, já que com a integração não vai ser mais preciso preencher tantos dados, pois alguns desses dados já estão disponíveis no sistema SIGA [2].

Temos como objetivo ainda desenvolver no sistema, a integração com os dados de currículos dos docentes na plataforma Lattes [3]. Com a intenção de facilitar o uso do sistema, evitando digitar tantos dados, como publicações, orientações, dentre outras. Temos também que desenvolver e integrar os sistemas de forma que, no fim do projeto, seja possível fazer o cadastro das atividades dos docentes e, a partir disso, uma avaliação do professores para saber qual docente terá prioridade de se afastar do DSC.

E por fim, com todas as iterações funcionando e com as melhorias pós-avaliação efetivadas, o sistema será testado e implantado do DSC. Assim os professores poderão ter acesso ao sistema.

1.2 Metodologia

O desenvolvimento desse estudo foi dividido em algumas fases. De maneira que essa divisão estabeleça estratégias que auxiliem o desenvolvimento do sistema e, assim, se consiga alcançar todos os objetivos especificados. São essas fases:

Fase 1 - Estudo do sistema e da tecnologia usada

Nessa fase foi feita uma revisão da literatura de maneira que sintetize os conceitos e abordagens referentes ao tema escolhido. É feito um estudo sobre o sistema e sobre toda a tecnologia que vai ser utilizada no mesmo.

Assim, nessa fase abordamos Padrões de Projeto (para um desenvolvimento iterativo do sistema) [4], o *Framework* Struts [5], o Modelo MVC (*Model View Controller*) [6], Sistemas de Informação [1], UML [7] e Processo de Software [8].

Fase 2 – Reavaliação e Redefinição do Escopo do sistema

Depois de revisar a literatura e fazer um estudo sobre o sistema, passamos para a próxima fase que trata da reavaliação e redefinição do escopo do sistema. Nessa fase foi feita uma reavaliação no que foi desenvolvido no sistema no semestre anterior e definidas as possíveis melhorias no sistema.

[3] http://www.siga.poli.br/poli/principal.jsp

Ainda nessa fase, foi feita uma reunião com os docentes para o levantamento dos novos requisitos. A partir daí, se redefine os requisitos e define os pontos críticos.

Fase 3 – Estudo sobre a integração do Sistema com Sistemas externos

O sistema resultante desse estudo, irá se integrar com dados de dois sistemas externos. São eles: o SIGA e o Lattes. Na integração com os dados do SIGA, vamos receber um arquivo em txt, vamos ler os dados contidos no mesmo e armazenar num banco esses dados referentes aos docentes, disciplinas, carga horária, dentre outros.

Já na integração com os dados da plataforma Lattes, será feito um upload de um arquivo XML [11], esse arquivo vai ser lido e os dados, equivalentes ao docente, vão ser armazenados no banco de dados.

Fase 4 – Desenvolvimento Iterativo do Sistema

Foi escolhido um Padrão de Projeto que facilite o desenvolvimento iterativo do sistema. Para esse desenvolvimento, foi usada a linguagem de programação Orientada a Objeto Java e de alguns de seus componentes Web. Utilizamos também o *framework* Struts [5], que facilita bastante o desenvolvimento Web.

Fase 5 – Implantação e Avaliação do Sistema

Após o desenvolvimento de todas as iterações, implantaremos o sistema no servidor do DSC. É nessa fase que é feita a instalação e configuração dos softwares necessários para o sistema funcionar. Com tudo instalado e configurado, podemos colocar a aplicação no servidor.

1.3 Estrutura do Trabalho

Foi montada uma estrutura do estudo realizado. Após essa fase introdutória, seguimos com alguns tópicos nos próximos capítulos, visando um melhor desenvolvimento e entendimento do trabalho.

O capítulo 2 é onde são explanados os conceitos e o embasamento teórico do trabalho desenvolvido. Nesse capítulo vamos falar sobre sistemas de informação, o método de desenvolvimento iterativo de sistemas, vai ser detalhado o modelo Cliente/Servidor. Vai tratar ainda da arquitetura MVC (Model View Controller) e do seu aprimoramento para aplicações web, que é o modelo 2 do MVC [6].

No capítulo 3 é feita uma descrição bem detalhada do sistema. Nesse capítulo ainda, vão ser mostrados os requisitos do sistema em questão e a sua importância para que a aplicação seja funcional e ajude os docentes a gerenciar o DSC.

O capítulo 4 é a parte do documento onde vai ser explicado como foi feita a implementação do sistema. Vai ser explanada a tecnologia usada, a técnica de desenvolvimento iterativo [4], o uso de programação em camadas, como é feito o acesso aos dados, o uso e as configurações do *framework* Struts [5]e as visões dos atores do sistema.

No capítulo 5 falaremos sobre a conclusão, as contribuições, considerações finais e, para terminar, os trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Nesse capítulo vamos detalhar os paradigmas utilizados no desenvolvimento do sistema e as tecnologias envolvidas no sistema. Vamos falar sobre Sistemas de Informação e suas particularidades, assim como vamos falar sobre o Desenvolvimento iterativo de software, a arquitetura Cliente/Servidor e o modelo MVC que foi utilizado para o desenvolvimento do sistema.

2.1 Sistemas de Informação

Para um melhor entendimento sobre S.I., devemos saber a diferença entre dados e informação. Dados são fatos em sua forma primária. Podemos dizer que dados podem não possuir significados, se analisados de maneira isolada. Já a informação é a junção desses dados organizadamente, de tal forma que esses dados tenham significados, representem algum valor. Podemos dar um exemplo prático do sistema envolvido nesse estudo, vamos supor que a quantidade de alunos, matriculados em uma determinada disciplina, não venha a ser importante e nem agregue valor algum para os docentes, em algum momento. Mas na hora da avaliação dos docentes, essa informação passa a ser importante, pois a quantidade de alunos pode alterar a pontuação do docente e, além disso, essa quantidade de alunos nessa disciplina se juntará aos dados relativos à quantidade de alunos em todas as disciplinas que o docente leciona. Essa junção de dados vai gerar uma informação que vai ter significado e importância para o DSC.

Sendo assim, é notório que um S.I. é uma ferramenta de trabalho fundamental para o sucesso no gerenciamento de uma empresa. Porém, existe uma série de

características necessárias para que o S.I. atenda às necessidades dos gestores. São essas características:

- Confiabilidade – O sistema tem que ser coeso e correto;
- Agilidade – Tem que está disponível no tempo certo;
- Completa – Precisa conter todas as informações importantes;
- Clareza – Apresentar o fato com clareza, sem o mascarar entre outros fatos;
- Rapidez – Executar as funcionalidades em tempo hábil;
- Relevante – A informação deve ser importante e ajudar na tomada de decisão;
- Simples – A informação não deve ser complexa, pois o intuito é de ajudar a tomar decisão.

É importante que essas características sejam obedecidas, pois o S.I. tem como objetivo ajudar na tomada de decisão dos gerentes das corporações. E a falta de algumas dessas características pode implicar numa tomada errada de decisão, e isso pode custar muito caro para a empresa.

Um sistema tem como atividades básicas: entrada, processamento e saída. A entrada envolve a captação de dados brutos. O processamento transforma a entrada bruta em uma forma mais útil e apropriada. A saída transfere a informação para as pessoas ou atividades que usarão a informação. A Figura um ilustra bem um modelo geral de um sistema de informação.

Os S.I. podem ser classificados, de acordo com o tipo de informação processada em: Sistema de Informação Operacional (SIO), Sistema de Informação Gerencial (SIG) e o Sistema de Informação Estratégico (SIE). O SIO lida com as transações de rotina das organizações. O SIG ajuda no gerenciamento da empresa, gera relatórios com os resultados obtidos. Os SIE são sistemas que dão suporte a tomada de decisões estratégicas das corporações [1].

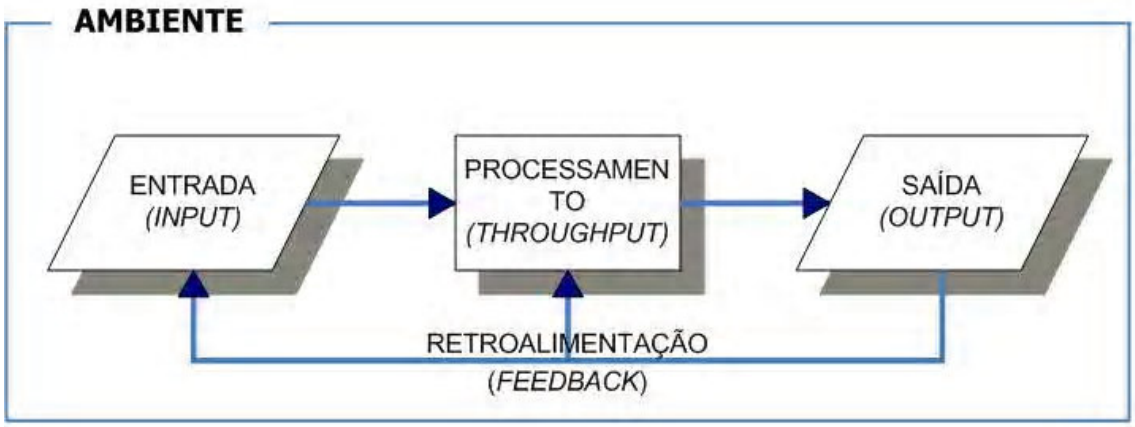


Figura 1. Modelo Geral de S.I.

2.2 Desenvolvimento Iterativo

O modelo de desenvolvimento iterativo, em detrimento do desenvolvimento em cascata, vem se destacando como o processo mais adequado para se desenvolver software. O modelo em cascata, também conhecido como clássico ou linear, é um dos modelos de desenvolvimento mais conhecido entre os desenvolvedores de sistema. No modelo em cascata, as fases de desenvolvimento seguem de maneira praticamente seqüencial, por exemplo, para a fase de análise começar, o levantamento de requisitos deve ter terminado, assim como para ter inicio a fase de projeto, a fase de análise deve ter terminado.

Nesse método de desenvolvimento iterativo, a participação do cliente é fundamental. Pois através do seu *feedback* periódico são feitos ajustes nas funcionalidades já desenvolvidas e são acrescentadas novas funcionalidades a serem implementadas. Com a participação do cliente, consegue-se um resultado melhor nas funcionalidades e evita que os desenvolvedores percam tempo implementando uma funcionalidade que vai ser pouco ou nunca utilizada.

O modelo de desenvolvimento iterativo foi criado em resposta às fraquezas do modelo em cascata, o mais tradicional. No modelo iterativo, é usada uma estratégia na qual várias partes do sistema são desenvolvidas em paralelo e só são integradas quando essas partes estiverem completas. O modelo iterativo tem algumas características

interessantes: resolução dos principais riscos antes da realização de grandes investimentos, realização da integração e do teste de forma contínua.

Como vantagens do modelo de desenvolvimento iterativo, podemos citar que os riscos mais graves são atacados primeiro e que, no caso de alguma falha, o sistema pode ser interrompido na fase inicial sem uma grande perca de tempo e investimento, outra vantagem é o fato de que uma porção do sistema pode ser entregue enquanto a outra parte do sistema ainda está em desenvolvimento, além disso, tem a vantagem de aumentar o reuso de softwares ou parte de softwares.

As iterações nas fases de iniciação/elaboração se concentram nas atividades de gerenciamento, requisitos e design. Já na fase de construção, as iterações dão ênfase no design, na implementação e no teste. E as iterações na fase de transição se focam no teste e na implantação [9]. A Figura 2 mostra o modelo de desenvolvimento iterativo.

Ao término de cada iteração, os responsáveis e interessados pelo sistema se reúnem para avaliar o sistema. Dessa avaliação, podem surgir algumas alterações nos requisitos e/ou na arquitetura do sistema, além disso, são nessas reuniões que são decididos quais passos serão dados para próxima iteração.

Figura 2. Modelo de Desenvolvimento Iterativo [9].

Falando em padrões de desenvolvimento iterativo, podemos destacar o RUP (*Rational Unified Process*). O RUP fornece técnicas a serem seguidas pelos membros da equipe de desenvolvimento com o objetivo de tornar a produtividade maior e, com isso, aumentar os lucros, além disso, evita riscos de problemas com requisitos e arquitetura. O RUP usa uma abordagem de orientação a objetos na sua concepção, é

projetado e documentado utilizando a notação UML (*Unified Modeling Language*) [4] que ilustra os processos em ação [10].

O RUP divide projeto de um software em quatro fases diferentes. É importante salientar que, em cada fase dessas, podem ocorrer mais de uma iteração. São fases do projeto de software:

1. Concepção: é a fase que foca o escopo do sistema;
2. Elaboração: foca mais na arquitetura do sistema;
3. Construção: foca no desenvolvimento do sistema;
4. Transição: foca na implantação.

2.3 O Modelo Cliente/Servidor

O modelo Cliente/Servidor possui uma arquitetura na qual o processamento das informações é dividido em módulos ou processos distintos. O processo Servidor é responsável pela manutenção da informação e os processos Cliente são responsáveis pela obtenção dos dados. Nesse modelo, os clientes enviam requisições ao servidor, pedindo o que precisam, e o servidor vai processar as requisições e enviar o resultado dos pedidos aos clientes [12]. Na Figura 3, ilustramos a comunicação cliente/servidor, e podemos notar que, nesse modelo, é possível que mais de um computador cliente se comunique com o servidor ao mesmo tempo.

Os clientes, geralmente são estações de trabalho que solicitam serviços ou informações que estão contidas no servidor. No nosso caso, o cliente pode ser qualquer computador que possua um browser de internet e que tenha acesso à comunicação com um servidor.

O servidor é, normalmente, uma máquina bastante potente que atua como um depósito de dados e fornecem serviços às redes de computadores. No servidor encontramos a aplicação e os dados que serão retornados como respostas aos clientes. Embora o termo servidor seja largamente aplicado a computadores completos, um servidor pode ser apenas um software ou uma parte de um sistema computacional. No nosso sistema, utilizaremos um único computador que atuará como um servidor. No servidor teremos que instalar um servidor web (para processar os dados da aplicação) e

um SGBD (para acessar e retornar os dados necessários às respostas das requisições). A Figura 3 mostra como funciona a comunicação no modelo Cliente/Servidor.

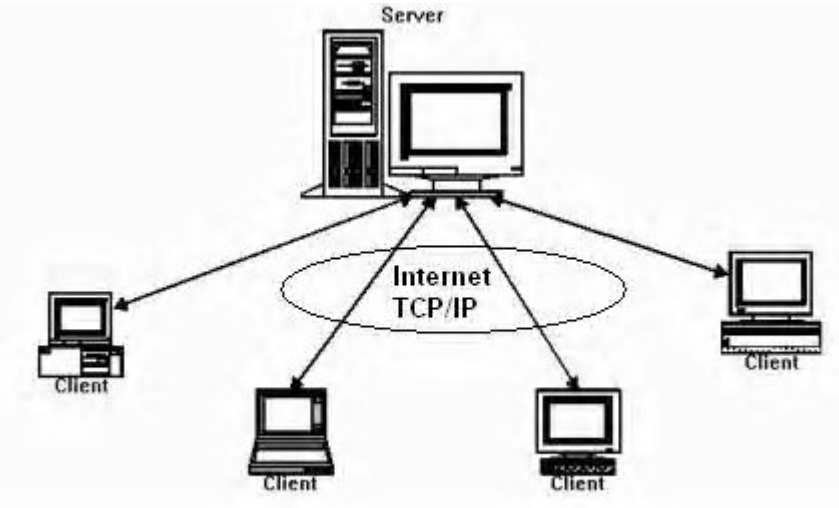


Figura 3. Comunicação no Modelo Cliente/Servidor.

O acesso dos clientes ao servidor é feito através do endereço do servidor (URL). O cliente geralmente se comunica com o servidor, enviando formulários HTML com requisições. Essa comunicação entre o cliente e o servidor, geralmente é feita usando o protocolo de comunicação TCP, mas também pode ser usado o protocolo UDP.

2.4 Modelo 2 do MVC

O Modelo MVC (*Model View Control*) é um padrão de arquitetura de software que teve seu surgimento na década de 70. Um software é formado de vários componentes, como os dados, as classes de controle, as telas de apresentação, dentre outros. Todos esses componentes precisam se comunicar para o funcionamento do software, só que essa comunicação está ficando cada vez mais complexa, assim como os softwares.

Com o aumento da complexidade das aplicações desenvolvidas, tornou-se fundamental separar os dados (*Model*) do layout (*View*). Com isso, conseguimos fazer alterações no layout de maneira tal que essas alterações não afetem a manipulação dos dados. Assim como é possível fazer alterações nos dados sem alterar o layout. O MVC resolve o problema da complexidade de comunicação entre os componentes através da separação das tarefas de acesso aos dados e lógica de negócio, lógica de apresentação e de interação com o usuário, introduzindo um componente entre os dois: o *Controller*.

No MVC dividimos os componentes em 3 grupos:

- *Model* (Modelo) – Representa os dados da aplicação, assim como as regras que regem o acesso e as atualizações dos mesmos.
- *View* (Visão) – Essa camada fica responsável pela interface com o usuário. Realiza o acesso aos dados através da camada de modelo e retorna os dados como eles devem ser apresentados ao usuário.
- *Controller* (Controle) – Processa e responde aos eventos, além de poder invocar alterações no *Model*. Em outras palavras, o *Controller* é quem tem a responsabilidade de controlar a aplicação.

A Figura 4 mostra como é feita a comunicação entre os componentes do modelo MVC. Note que a *View* passa a sua ação (*Action*) para o Controller, e este, por sua vez, mapeia o *Action* no arquivo de configuração e encaminha a requisição ao *Model*. A *View* fica observando o *Model* para ver se tem alguma alteração de estado e com isso retornar ao usuário a alteração.

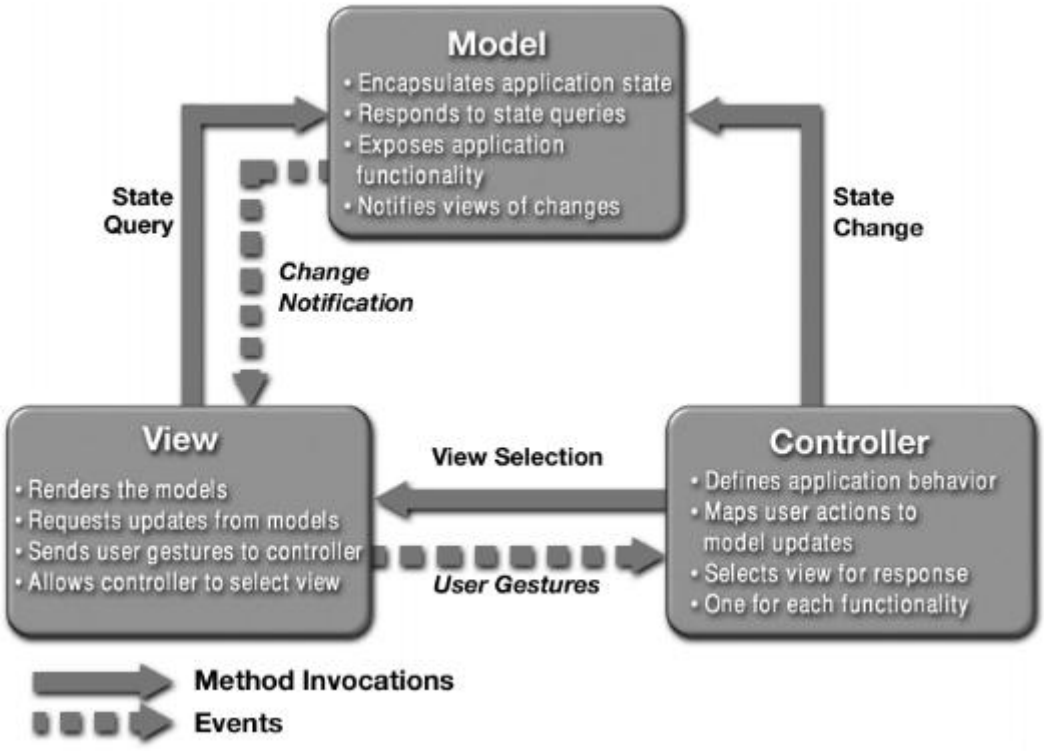


Figura 4. Comunicação no MVC [13].

O MVC acelerou bastante o desenvolvimento dos softwares, mas ainda tinha um problema. O modelo separa em camadas, mas ainda podem existir arquivos Java que

tenham participação em duas camadas (*View* e *Controller*, por exemplo), e com isso, não é possível dividir cada parte para equipes separadas desenvolverem.

Foi nesse contexto que surgiu o Modelo 2 do MVC ou MVC 2. O MVC 2 é um avanço do MVC, só que é específico para aplicações Web. Esse modelo consegue fazer com que cada camada trabalhe isoladamente, sendo assim os *designers* podem ficar responsáveis pela *View*, assim como outras duas equipes de desenvolvedores ficam responsáveis pelas camadas de *Controller* e *Model*.

Na arquitetura do MVC 2 é introduzido um *servlet* como controlador (Controller) entre o browser e as páginas JSPs, ou mesmo um *servlet* que serve como índice para as próximas páginas. *Servlet* é uma tecnologia que insere novos recursos a um servidor, por isso são consideradas extensões de servidores, pois disponibiliza aos programadores uma interface para o servidor web. Aplicações baseadas no *servlet* geram conteúdo dinâmico (normalmente HTML) e interagem com os usuários, utilizando o modelo *request/response* (requisição/resposta). Os *servlets* normalmente utilizam o protocolo HTTP, apesar de não serem restritos a ele.

No MVC 2 os JSPs fazem parte apenas da camada de visão (*View*). Na camada de modelo, temos as classes Java que possuem as regras de negócio do sistema. Já na camada de controle, temos classes Java de controle do sistema. Já a camada de modelo é a que menos sofre alteração. A única alteração sofrida é na hora da resposta da requisição, agora ela se comunica com o *Controller* e não mais diretamente com a *View*.

Capítulo 3

Requisitos do Sistema

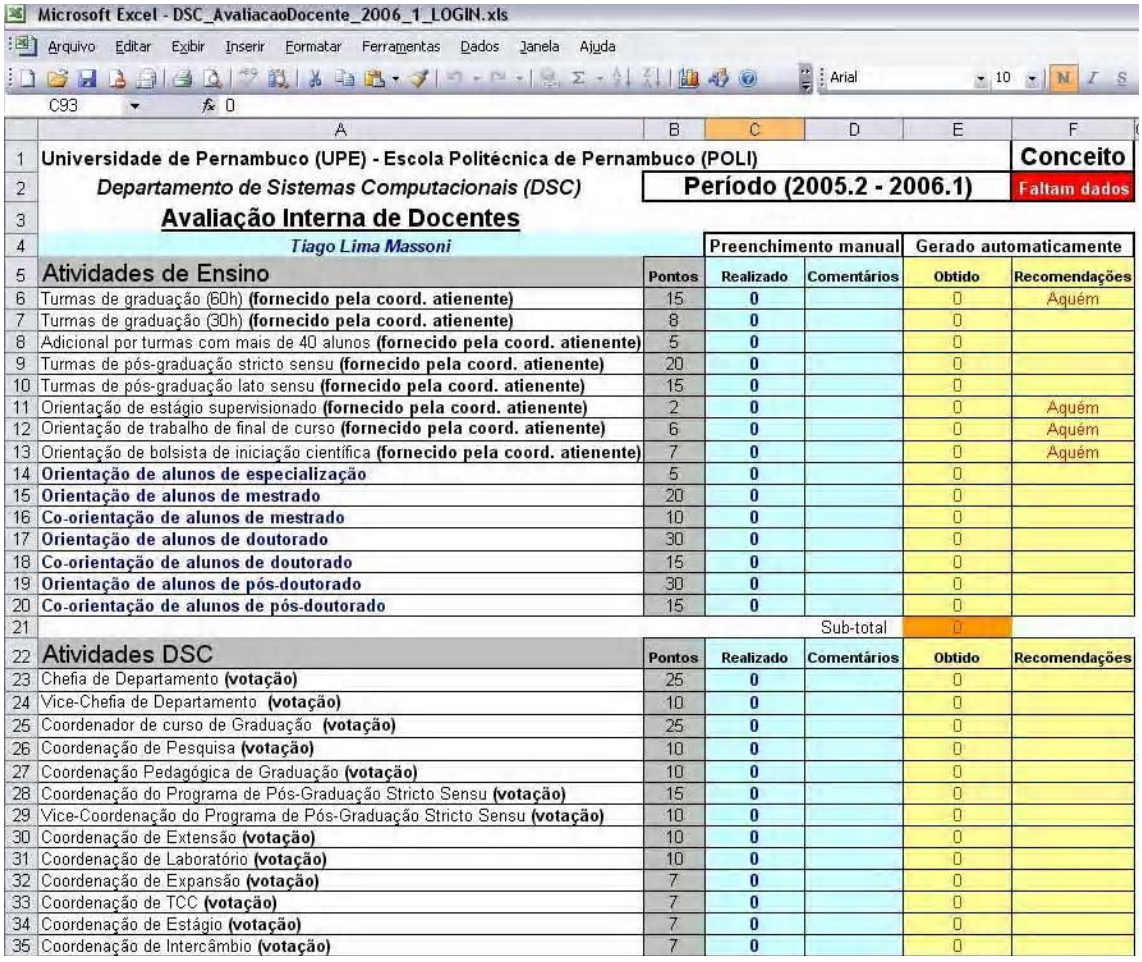
Este trabalho procurou desenvolver um sistema de informação que auxilie no gerenciamento do DSC - UPE. Além de auxiliar no gerenciamento, traz facilidades no manuseio do mesmo, automatiza certas tarefas e contribui com a crescente modernização e desenvolvimento do DSC.

3.1 Método de Avaliação Atual

Por ser um departamento novo, o DSC ainda busca alcançar uma melhor estrutura para os que fazem parte do mesmo. Como já foi falado no Capítulo 1 desse trabalho, o departamento está crescendo rapidamente. O número de docentes e discentes vem aumentando cada vez mais. Com a abertura de novos cursos de pós-graduação, o mestrado e a busca pelo doutorado, esse número vai aumentar ainda mais. Esse fato é muito positivo para o departamento, mas temos que levar em conta que, com isso, fica mais difícil gerir o DSC. Muitas das atividades do coordenador do departamento, são feitas, hoje em dia, manualmente. Algumas dessas atividades são cansativas para ser feitas manualmente.

O coordenador faz avaliações para medir o desempenho dos docentes, além de controlar as atividades dos mesmos. Na avaliação, cada docente tem que alcançar, pelo menos, a pontuação mínima para cada semestre. Essa pontuação é feita de acordo com as atividades desenvolvidas pelo docente. Cada atividade tem um valor associado e no final do semestre são somados os valores das atividades desenvolvidas por cada professor e essa soma é a pontuação do docente. São 95 atividades ao todo, atualmente.

Essas atividades se dividem em áreas de atuação, podem ser atividades de ensino, atividades do departamento, publicações de artigos e revistas, orientações de mestrados, dentre outras. Como já foi descrito no parágrafo anterior, essa avaliação é uma das atividades que são feitas manualmente através de uma planilha que contem todas as atividades. Essa planilha é entregue a cada docente para que o mesmo preencha e entregue ao coordenador do departamento. Há de se convir que esse método é bastante trabalhoso, já que o coordenador, sequer, tem uma visão conjunta dos docentes, visto que ele tem que acompanhar uma a uma cada planilha. A planilha de avaliação pode ser visualizada na Figura 5. Na planilha abaixo, mostramos as atividades do Departamento, a quantidades de pontos de cada atividade e alguns campos de verificação e comentários para o coordenador realizar.



Microsoft Excel - DSC_AvaliacaoDdocente_2006_1_LOGIN.xls

Arquivo Editar Exibir Inserir Formatar Ferramentas Dados Janela Ajuda

C93 0

	A	B	C	D	E	F
1	**Universidade de Pernambuco (UPE) - Escola Politécnica de Pernambuco (POLI)**					**Conceito**
2	***Departamento de Sistemas Computacionais (DSC)***		**Período (2005.2 - 2006.1)**			**Faltam dados**
3	**Avaliação Interna de Docentes**					
4	***Tiago Lima Massoni***		**Preenchimento manual**		**Gerado automaticamente**	
5	**Atividades de Ensino**	**Pontos**	**Realizado**	**Comentários**	**Obtido**	**Recomendações**
6	Turmas de graduação (60h) **(fornecido pela coord. atienente)**	15	0		0	Aquém
7	Turmas de graduação (30h) **(fornecido pela coord. atienente)**	8	0		0	
8	Adicional por turmas com mais de 40 alunos **(fornecido pela coord. atienente)**	5	0		0	
9	Turmas de pós-graduação stricto sensu **(fornecido pela coord. atienente)**	20	0		0	
10	Turmas de pós-graduação lato sensu **(fornecido pela coord. atienente)**	15	0		0	
11	Orientação de estágio supervisionado **(fornecido pela coord. atienente)**	2	0		0	Aquém
12	Orientação de trabalho de final de curso **(fornecido pela coord. atienente)**	6	0		0	Aquém
13	Orientação de bolsista de iniciação científica **(fornecido pela coord. atienente)**	7	0		0	Aquém
14	**Orientação de alunos de especialização**	5	0		0	
15	**Orientação de alunos de mestrado**	20	0		0	
16	**Co-orientação de alunos de mestrado**	10	0		0	
17	**Orientação de alunos de doutorado**	30	0		0	
18	**Co-orientação de alunos de doutorado**	15	0		0	
19	**Orientação de alunos de pós-doutorado**	30	0		0	
20	**Co-orientação de alunos de pós-doutorado**	15	0		0	
21				Sub-total	0	
22	**Atividades DSC**	**Pontos**	**Realizado**	**Comentários**	**Obtido**	**Recomendações**
23	Chefia de Departamento **(votação)**	25	0		0	
24	Vice-Chefia de Departamento **(votação)**	10	0		0	
25	Coordenador de curso de Graduação **(votação)**	25	0		0	
26	Coordenação de Pesquisa **(votação)**	10	0		0	
27	Coordenação Pedagógica de Graduação **(votação)**	10	0		0	
28	Coordenação do Programa de Pós-Graduação Stricto Sensu **(votação)**	15	0		0	
29	Vice-Coordenação do Programa de Pós-Graduação Stricto Sensu **(votação)**	10	0		0	
30	Coordenação de Extensão **(votação)**	10	0		0	
31	Coordenação de Laboratório **(votação)**	10	0		0	
32	Coordenação de Expansão **(votação)**	7	0		0	
33	Coordenação de TCC **(votação)**	7	0		0	
34	Coordenação de Estágio **(votação)**	7	0		0	
35	Coordenação de Intercâmbio **(votação)**	7	0		0	

Figura 5. Planilha de Avaliação dos Docentes.

Com o crescimento do departamento e a abertura de novos cursos, é preciso que, cada vez mais, os docentes estejam capacitados e especializados. Porém, para isso pode ser necessário o afastamento do docente para concluir um mestrado ou doutorado, por exemplo. Isso é um problema no DSC, já que não existe, atualmente, nenhuma forma de controlar essas saídas de maneira segura, pois é necessário que um docente assuma a vaga deixada pelo docente que vai se afastar. Existe a possibilidade de o docente lecionar algumas disciplinas a mais no período anterior ao seu afastamento, com isso ele fica com créditos no departamento. Mas, atualmente, não tem como controlar esse processo de adiantamento de disciplinas.

3.2 Objetivos

O resultado desse trabalho será o desenvolvimento, a integração e a implantação de dois sistemas de informação. O primeiro sistema possibilitará a gerência das atividades desenvolvidas pelos docentes para a avaliação dos mesmos. O segundo sistema permitirá organizar uma fila de docentes que desejem se afastar do departamento pelos motivos já citados. Então esses sistemas se integram para que na hora de decidir quem tem prioridade de se afastar, o sistema veja quais docentes estão na fila de espera, faça uma avaliação do desempenho de cada docente que estiver na fila de espera e decida quem vai se afastar através do resultado da avaliação.

O sistema automatizará muitas atividades dos usuários do mesmo. Para evitar que o docente digite manualmente todas as atividades desenvolvidas por ele no departamento, foi feito uma tela no sistema que vai possibilitar fazer o *upload* de um arquivo (.txt) que virá do SIGA. Esse arquivo conterá todas as atividades de ensino dos docentes do departamento e, com isso, ao fazer o *upload*, os dados do arquivo serão interpretados e salvos no banco de dados sem a necessidade do docente digitar todas as atividades no sistema. Essa funcionalidade foi feita com o objetivo de automatizar o processo de inclusão de atividades manualmente, que também é possível de ser feito. O sistema também permite o *upload* do currículo Lattes no formato XML. Com isso, o arquivo é interpretado e alguns dados, relacionados a atividades desenvolvidas no departamento, são salvos no banco de dados repetindo o mesmo processo explicado no *upload* do arquivo txt. Com isso, conseguimos um alto grau de automação no sistema.

O resultado final do trabalho vai ser um sistema que possibilitará a avaliação dos professores em um processo mais rápido, automatizado e moderno e, além disso, possibilitará o controle do sistema de banco de créditos. Resumindo, o sistema vai facilitar a gerência do departamento e o controle no afastamento dos docentes.

3.3 Requisitos

O sistema resultante desse trabalho ajudará na gerência do departamento e no controle das saídas e é baseado na junção de informações entre cinco itens principais que constituem o sistema.

3.3.1 Docentes

Os docentes são os atores principais do sistema. Sem eles, a maioria das funcionalidades do sistema não fazem sentido algum. Pois quem pedirá afastamento do DSC por um período é o docente, além disso, é o docente quem vai ser avaliado é quem desenvolve atividades. O docente deve, semestralmente, cadastrar as atividades por ele exercidas no sistema. Os docentes precisam fazer, no sistema, o *upload* do seu currículo na plataforma Lattes, no formato XML, essa tarefa é realizada uma vez a cada semestre.

3.3.2 Coordenador do Curso

O coordenador também é um docente, só que possui alguns privilégios a mais no sistema e algumas responsabilidade a mais no departamento. É o coordenador o responsável por cadastrar os docentes, as atividades, tipos de atividades. É ele ainda quem controla a fila de espera dos docentes que desejam se afastar por um período além de lembrar aos docentes da necessidade de cadastrar as atividades a cada semestre e fazer as avaliações dos docentes. O coordenador precisa fazer, no sistema, o *upload* do arquivo txt com os dados do SIGA uma vez por semestre para integrar os dados do sistema com os do SIGA.

3.3.3 Atividades

Existe no sistema a funcionalidade que cadastra as atividades desenvolvidas pelos docentes. Esse cadastro é feito pelo coordenador do departamento. Como já foi

citado no capítulo anterior, cada atividade tem um valor associado. Esse valor pode ser alterado, pelo coordenador, a cada semestre. Atualmente, existem noventa e cinco atividades cadastradas, divididas em oito tipos de atividades.

3.3.4 Créditos

Os créditos são fundamentais para o controle de afastamento dos docentes. Pois os créditos funcionam como a quantidade de horas trabalhadas pelo docente. Para se afastar, o docente tem que adiantar algumas horas de trabalho, ou seja, lecionar algumas disciplinas a mais que a quantidade de turma normalmente. Essas disciplinas a mais que forem lecionadas, contarão como créditos para um possível afastamento seu.

3.3.5 Fila de Espera

A solicitação de afastamento deve ser agendada com antecedência junto ao coordenador do departamento. Após a solicitação, o coordenador colocará o nome do docente na fila de espera onde se encontram todos os docentes que querem se afastar. O critério de desempate na fila de espera é a avaliação dos docentes. Se no semestre tiver mais de um docente querendo se afastar, cabe ao coordenador decidir, baseado na avaliação dos docentes, qual docente tem a prioridade de se afastar no período.

3.4 Casos de Uso

Vamos falar agora sobre os casos de uso do sistema que será resultado desse trabalho. Os requisitos funcionais do sistema foram levantados a partir de uma reunião com o coordenador do departamento Renato Moraes. Nessa reunião, foi feita uma reavaliação do sistema desenvolvido no semestre anterior, foram decididas as melhorias e acrescentados novos requisitos. Baseado nos requisitos foi criado um diagrama de casos de uso.

Um ator de um sistema é qualquer coisa ou pessoa que interage com o sistema. No caso do nosso sistema, temos dois atores: os Professores e o Coordenador (que também é um professor). O coordenador tem direito de usar todos os casos de uso, enquanto os professores tem acesso limitado aos casos de uso. A Figura 7 mostra os atores do sistema e os casos de uso a que cada ator tem direito.

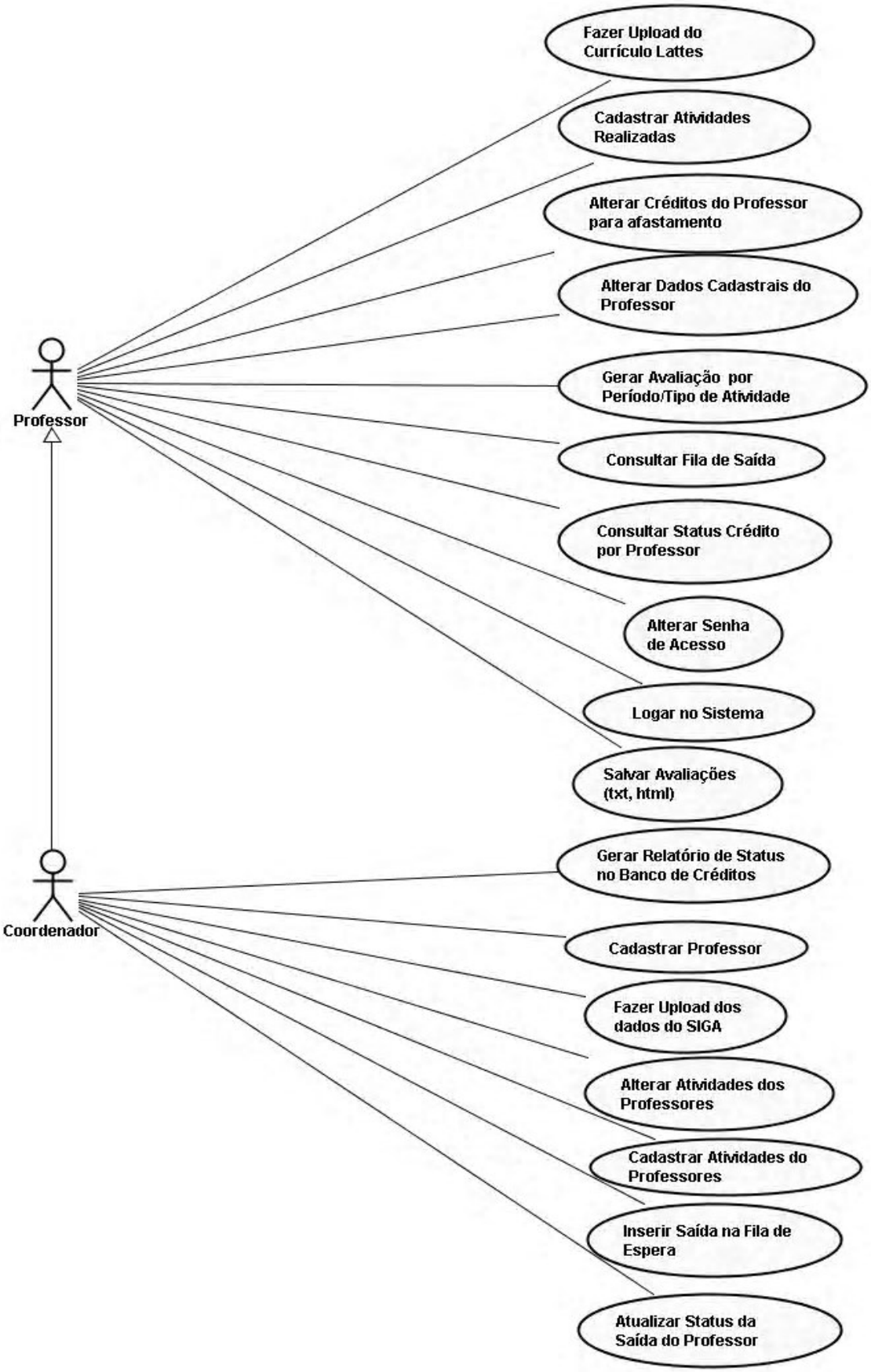


Figura 6. Diagrama de Casos de Uso.

Como visto na Figura 6, temos 17 casos de uso no sistema. Temos casos de uso restritos apenas ao coordenador do Departamento. Os casos de uso são as funcionalidades que o sistema terá. Temos como exemplo o caso de uso Cadastrar Atividades Realizadas, que é a funcionalidade onde o docente cadastra as atividades desenvolvidas por ele. Escolhemos dois casos de uso para detalhar nesse trabalho como exemplo. Esses casos de uso foram escolhidos por se tratarem de casos de uso mais críticos do sistema.

No detalhamento do caso de uso, temos seções que ajudam o entendimento do mesmo. Na seção Pré-condições, escrevemos o que é preciso para executar o caso de uso. Por exemplo, para fazer o *upload* é preciso está autenticado e possuir o arquivo com os dados do SIGA. Na seção Pós-condições, detalhamos o que deve acontecer após a execução do caso de uso. Na seção Fluxo Básico, escrevemos o passo a passo para executar o caso de uso.

Caso de Uso: Fazer Upload dos Dados do SIGA

Descrição: Caso de uso responsável por fazer o upload, interpretar o arquivo e salvar os dados no banco.

Ator Primário: Coordenador.

Pré-condições: Coordenador está identificado e autenticado. O coordenador ter o arquivo com os dados do SIGA e desejar fazer a atualização semestral das atividades dos docentes.

Pós-condições: As atividades são relacionadas aos docentes correspondentes e devidamente salvas no banco de dados. O docente passa a ter atividades desenvolvidas sem a necessidade de cadastrá-las manualmente.

Fluxo Básico (Cenário de Sucesso):

1. O coordenador inicia o processo de Upload.
2. O coordenador tem acesso à tela de Upload para que ele escolha o arquivo com os dados.
3. Depois de ter escolhido o arquivo, o coordenador solicita a atualização dos dados.
4. A tela de confirmação da execução da operação é exibida ao coordenador.

Caso de Uso: Salvar Avaliações

Descrição: Caso de uso responsável por salvar o resultado da avaliação dos professores nos formatos txt ou html.

Ator Primário: Professor, Coordenador.

Pré-condições: O ator está identificado e autenticado. O ator ter realizado uma avaliação dos professores.

Pós-condições: Ser gerado um arquivo do tipo escolhido pelo ator.

Fluxo Básico (Cenário de Sucesso):

1. O ator inicia o processo para salvar o resultado da avaliação.
2. O ator escolhe o tipo de arquivo que deseja gerar (txt ou html).
3. Após escolher o tipo de arquivo, o ator solicita que o arquivo seja gerado.

Capítulo 4

Implementação

Este é o capítulo onde iremos detalhar como foi projetado o sistema e como foi feita a implementação do mesmo. O resultado desse trabalho será um sistema web e como tal, resolvemos usar o *framework* Struts, o modelo MVC 2.

Optamos por usar a linguagem Java [4] e as suas particularidades para o desenvolvimento voltado a web. Essa escolha foi feita baseada no fato de que a linguagem Java é portável, sendo assim funciona independente do sistema operacional do usuário. Foi usada também a linguagem Javascript para auxiliar em validações de campos, por exemplo. Além disso, usamos a linguagem CSS para formatarmos as nossas páginas, definirmos a fonte, cores e tamanho das letras, por exemplo. O desenvolvimento foi feito baseado em iterações, como já foi descrito em capítulos anteriores.

4.1 Struts e o MVC

O Struts é um *framework* para web que tem se tornado padrão entre os desenvolvedores J2EE. O Struts proporciona uma implementação do modelo 2 (uma variação do paradigma de projeto MVC) para construções de aplicações web. O Struts foi desenvolvido sobre a responsabilidade de uma equipe de aproximadamente 30 desenvolvedores, mas foi Craig R. MacClanahan o primeiro arquiteto e desenvolvedor do *framework* [16] . Após o desenvolvimento, o Struts foi doado para o *Apache Software Fundation*, onde continua sendo desenvolvido e aperfeiçoado.

[4] http://java.sun.com/

O Struts fornece um servlet controlador para a aplicação. O controlador tem como tarefas receber solicitações do cliente, decidir, de acordo com a regra de negócio, qual funcionalidade vai ser executada e, em seguida, delegar responsabilidades para produzir a próxima fase da interface do usuário. Um dos principais componentes desse controlador é a classe ActionServlet. Este servlet é configurado através da definição de um conjunto de ActionMappings. Um ActionMapping é responsável pelo mapeamento entre o caminho lógico e as URIs reais. O ActionServlet e ActionMapping fazem parte das classes básicas do framework Struts, além deles ainda participam dessas classes básicas o ActionForward, o ActionForm e as Classes Action (que interagem com o modelo para realizar alguma alteração no estado ou consulta e avisa o ActionServlet sobre a próxima exibição) [14].

Observe na Figura 7 o fluxo normal de uma aplicação utilizando o Struts, e também quais tecnologias estão envolvidas em cada uma das etapas. As etapas estão descritas abaixo de acordo com numeração da Figura 7:

1. É feita a solicitação HTTP. Esta solicitação normalmente é definida como requisicao.do, que é um nome lógico para a requisição do usuário.
2. A solicitação é mapeada no arquivo struts-config.xml, que é onde estão todas as definições do controlador do *framework*. O arquivo é lido por um ActionServlet (que faz o papel do controlador da aplicação) na inicialização da aplicação criando um banco de objetos com o arquivo de configuração. No arquivo de configuração são definidos os Actions (requisições dos usuários) para cada solicitação.
3. O ActionServlet define o Action correspondente para a solicitação.
4. É no caso de a requisição HTTP ser feita através de um formulário html. O ActionServlet coloca a entrada em um JavaBean. Esses JavaBeans são definidos como FormBeans no Struts.
5. O Action pode acessar o FormBean, efetuar qualquer operação e armazenar o resultado em um ResultBean.
6. O Action interage com a camada de negócio e, dessa interação, a base de dados poderá ser atualizada.

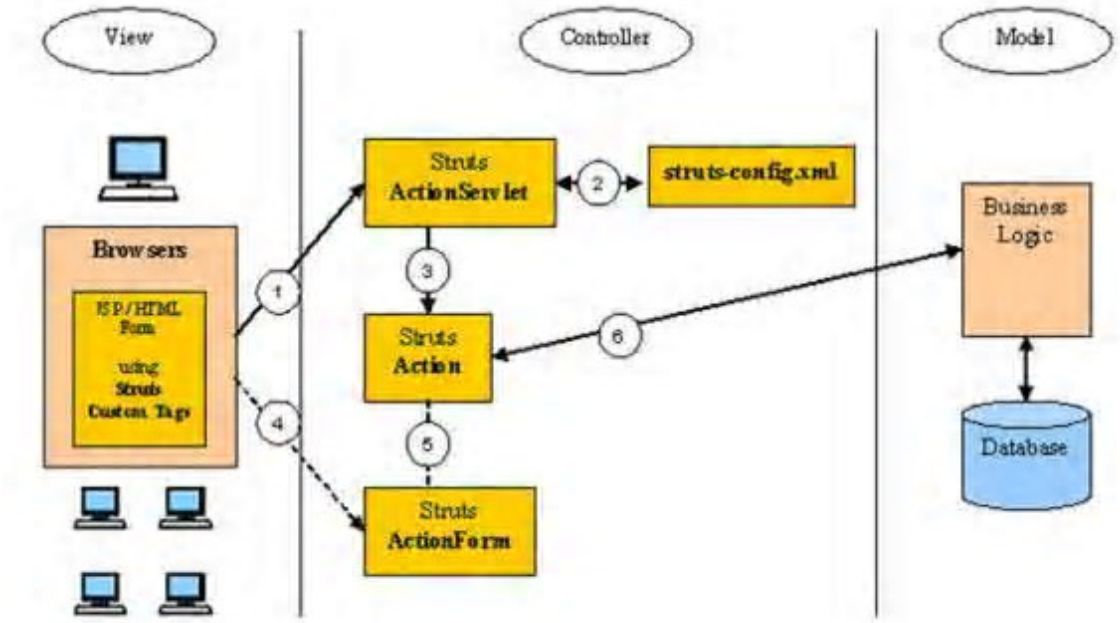


Figura 7. Fluxo de uma aplicação com Struts [16].

O Struts possui alguns arquivos de configuração, dentre eles está o *struts-config.xml* que é de grande importância. Esse arquivo possui detalhes que o ActionServlet precisa para lidar com as solicitações feitas para a aplicação. Além disso, é nesse arquivo que declaramos nossas classes Action e ActionForms. O struts-config.xml possui armazenado as configurações padrão dos objetos do controlador. O arquivo struts-config.xml mantém uma estrutura, onde temos a tag <*form-beans*> onde são declarados os ActionForms usados no sistema e a tag <*action-mappings*> onde são declaradas as Classes Action, dentre outras tags.

4.2 Arquitetura em Camadas

Desde o surgimento da arquitetura cliente/servidor, o desenvolvimento de software em camadas passou a ser adotado como base de arquitetura de sistemas. Usamos a programação em camadas no sistema resultante desse trabalho.

Esse tipo de paradigma de programação é bastante interessante para quem faz uso da programação orientada a objetos, já utiliza a idéia de abstração. A programação em camadas oferece diversos benefícios, entre eles está o fato de ser possível entender uma única camada, de maneira coerente, como um todo, sem a necessidade de ter um

conhecimento mais amplo sobre as outras camadas. As camadas podem ser substituídas por implementações alternativas dos mesmos serviços, o que é excelente para a reusabilidade e extensibilidade. Além disso, a dependência entre as camadas diminui bastante com o uso desse paradigma de programação.

Sendo assim, cada camada tem como função provê serviços para a camada superior, de maneira que a camada superior não precise entender nem ter conhecimento detalhado sobre esses serviços fornecidos. Dessa maneira, a implementação das camadas deve ser feita de forma que possibilite que a camada superior não saiba dos detalhes de implementação da camada inferior.

As camadas de negócios do sistema em questão estão divididas em Fachada, Controlador, Cadastro e Repositório (mais detalhes na Figura 8). A classe fachada tem a função de inicializar os controladores. O sistema é dividido e organizado em pacotes, onde cada pacote possui um controlador. Como o próprio nome já diz, é o controlador quem controla as ações a serem tomadas e chama os métodos correspondentes. Os cadastros dos dados no sistema possuem um repositório de dados. Um repositório de dados, nada mais é que um lugar onde se armazenam dados. O repositório representa uma interface onde se encontra a assinatura dos métodos que acessam a base de dados. Para o nosso sistema, foram criadas as classes RepositórioBDR (que implementa a interface repositório e acessa os dados). Na Figura 8 vemos uma representação da camada de negócios e das suas subcamadas. As setas indicam que cada classe da camada superior, possui uma instância das classes da camada imediatamente abaixo.

4.3 Manipulação dos Dados

Utilizamos um banco de dados para guardar as informações do sistema. Optamos por usar o MySQL [5], que é um Sistema de Gerenciamento de Banco de Dados (SGBD). O SGBD é um software utilizado para retirar da aplicação cliente a responsabilidade de gerenciar o acesso, manipulação e organização dos dados. Esse SGBD utiliza a linguagem SQL (*Structured Query Language*). SQL é uma linguagem de pesquisa declarativa para banco de dados relacional, é a linguagem mais utilizada nos SGBD. Muitas das características originais do SQL foram inspiradas na álgebra relacional.

[5] http://www.mysqlbrasil.com.br

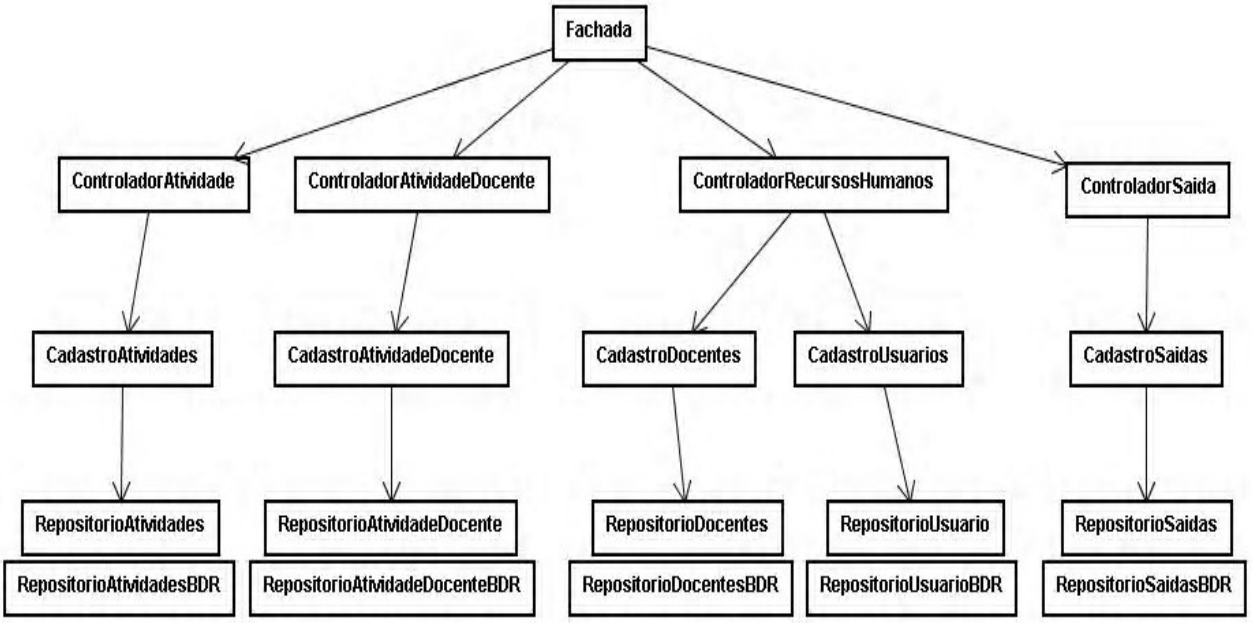


Figura 8. Camadas de Negócio do Sistema.

Pela definição, podemos dizer que bancos de dados relacionais são conjuntos de registros (dados) dispostos em uma estrutura regular que possibilita a reorganização dos mesmos para uma possível produção de informação. O termo banco de dados deve ser aplicado apenas aos dados, enquanto que o termo SGBD deve ser aplicado ao software com a capacidade de manipular bancos de dados de forma geral.

Utilizamos uma API (Application Programming Interface) da linguagem de programação Java para fazermos o acesso ao banco de dados. Essa API é chamada de JDBC (Java Database Connectivity) [6]. API é um conjunto de rotinas e padrões estabelecidos para utilização de suas funcionalidades por programas aplicativos, isto é, programas que não querem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.

O JDBC é a API padrão da indústria para a conexão com banco de dados independente de conectividade entre a linguagem Java e uma grande quantidade de banco de dados. JDBC é um conjunto de classes e interfaces escritas em Java que faz o envio de instruções SQL para qualquer banco de dados relacional. Para cada banco de dados há um *Driver* JDBC correspondente. Um *Driver* nada mais é que o conjunto de classes nativas que implementam as interfaces do JDBC. A Figura 9 mostra o modelo de entidade-relacionamento do banco de dados utilizado no nosso sistema.

[6] http://java.sun.com/javase/technologies/database/

Para estabelecer uma conexão com o banco de dados, é preciso seguir uma série de passos. São esses passos: carregar o *Driver*, estabelecer a conexão com o banco, executar o comando SQL, retornar os resultados (caso seja preciso) e fechar a conexão com o banco. Esses passos devem ser seguidos a cada vez que for se conectar ao banco.

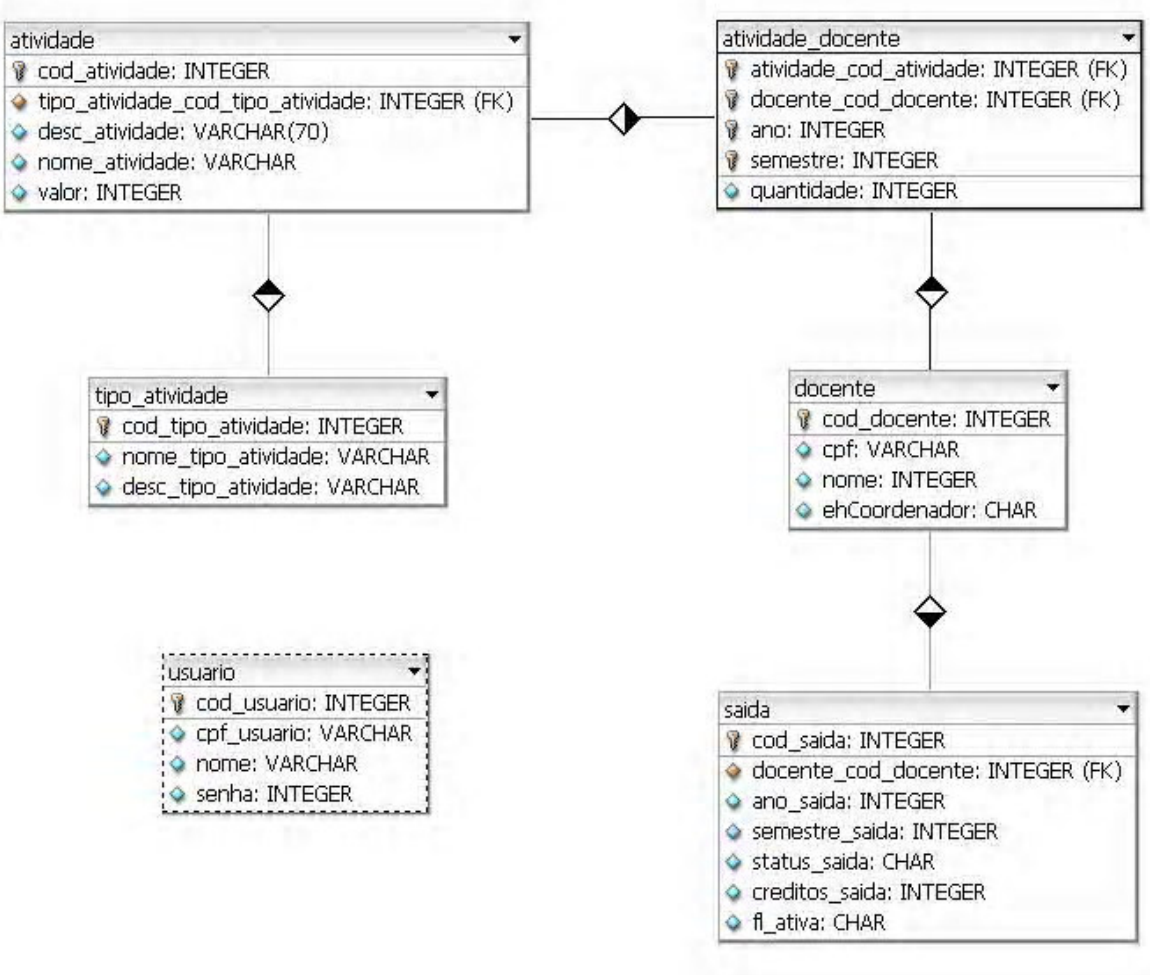


Figura 9. Estrutura do modelo de banco de dados do sistema.

É possível ver na Figura 9 a maneira como as tabelas se relacionam. Para facilitar o entendimento do modelo, vamos explicar como se dão algumas relações entre as principais tabelas. A tabela atividade possui um tipo de atividade, essa relação entre as tabelas acontece pelo código do tipo de atividade (campo único). A tabela atividade_docente se relaciona com a tabela atividade e com a tabela docente diretamente, com a primeira tabela a relação é pelo código da atividade (campo único), já com a tabela docente a relação é pelo código do docente (campo único).

4.4 Desenvolvimento das Iterações

Ao início desse trabalho, ficou definido que teríamos duas iterações no projeto para desenvolver. Após uma avaliação dos requisitos funcionais, dividimos os requisitos entre as iterações e os prazos para entrega de cada iteração.

No decorrer do desenvolvimento do sistema, tivemos a necessidade de mudar, um pouco, os planos. Por questões burocráticas, desistimos de sincronizar o nosso sistema com os dados da CAPES [7]. A sincronização com os dados da CAPES, ajudariam a automatizar ainda mais o sistema, pois evitaria que os docentes precisassem cadastrar algumas atividades desenvolvidas por eles, por exemplo, a publicação de um artigo em uma conferência ou revista. Então ficou definido que na primeira iteração seriam desenvolvidos três casos de uso e na segunda iteração mais três. Na Tabela 1, detalhamos os casos de uso e a iteração correspondente a cada caso de uso.

Tabela 1. Iterações e seus Casos de Uso.

Iteração	**Caso de Uso**
1ª Iteração	Fazer *Upload* do arquivo XML (Lattes)
	Fazer *Upload* do arquivo txt (SIGA)
	Salvar o resultado da Avaliação (txt)
2ª Iteração	Gerar indicador de posição do docente na fila de espera
	Melhorar Interface
	Implantar o sistema

4.5 Softwares Usados

Pensando em facilitar o desenvolvimento dos sistemas, usamos algumas ferramentas que agilizaram bastante a implementação do mesmo. Para o desenvolvimento das classes Java e telas JSPs, utilizamos a IDE (Integrated Development Enviroment) Red Hat Developer Studio (RHDS) [8]. Essa IDE é o resultado da integração do Exadel Studio (antigo plugin do Eclipse usado para o desenvolvimento de aplicativos web) e do

[7] http://qualis.capes.gov.br/webqualis/
[8] http://www.redhat.com/developers/rhds/

Eclipse [15] à linha de produtos da Red Hat. Podemos ver na Figura 11 um screenshot da IDE utilizada.

O RHDS possui integrado também o JBoss [9], que é um servidor de aplicação baseado na plataforma J2EE e que foi utilizado nesse trabalho. Um servidor de aplicação é um software que disponibiliza um ambiente para a instalação e execução de certas aplicações, no nosso caso aplicações web utilizando Java. Como é baseada em Java, JBoss pode ser usado em qualquer Sistema Operacional que suporte Java. Além disso, o RHDS dá suporte ao desenvolvimento usando Struts, que foi o caso do nosso sistema.

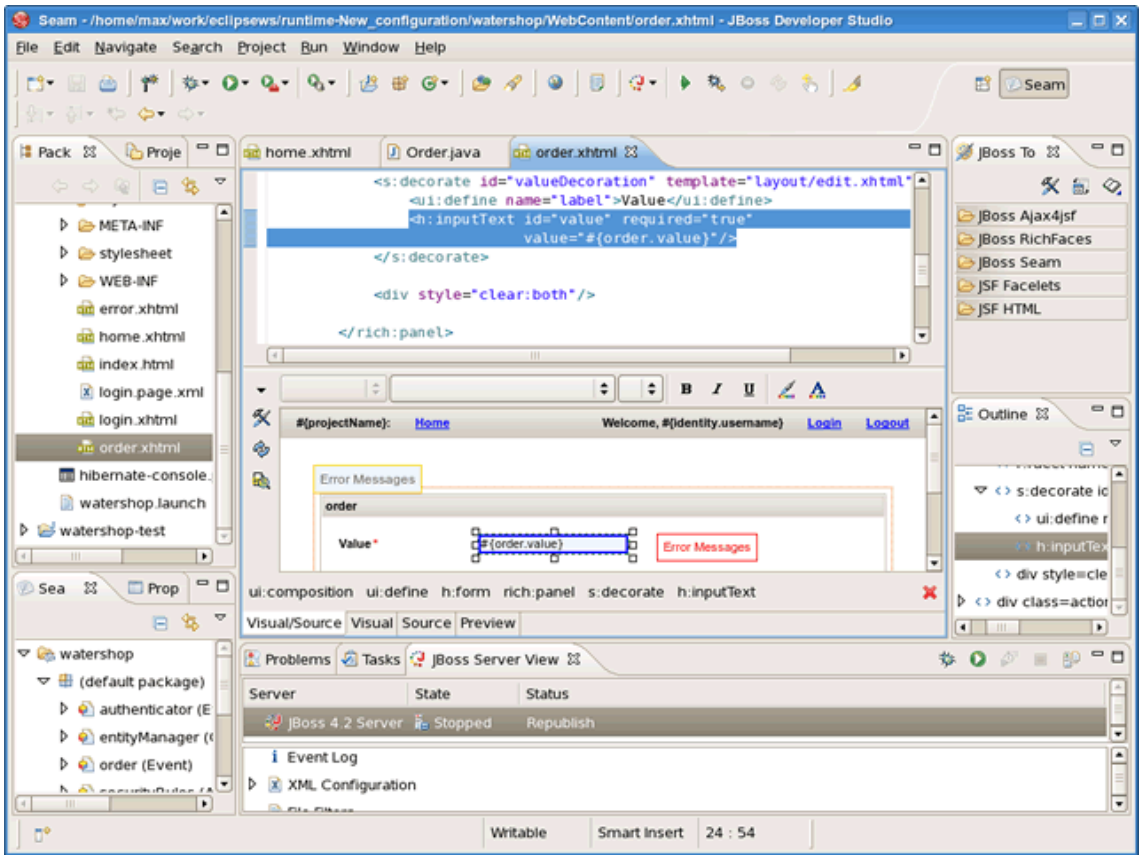


Figura 10. Screenshot da IDE RHDS.

Como já foi falado nesse trabalho, para controlar o banco de dados usamos o SGBD MySQL. O MySQL é atualmente um dos bancos de dados (de código aberto) mais populares, com mais de 10 milhões de instalações pelo mundo. Usamos um software denominado HeidiSQL [10] para interagir com o MySQL. O HeidiSQL é um software simples, com uma interface agradável e que consome pouquíssimos recursos

[9] http://labs.jboss.com/
[10] http://www.heidisql.com/

do computador. O HeidiSQL é muito útil para o gerenciamento do banco de dados, pois possui uma interface semelhante aos aplicativos do Windows e permite que o usuário navegue, facilmente, pelas tabelas. A Figura 12 mostra um screenshot do HeidiSQL.

Para fazermos os diagramas de seqüência, de classes e de casos de uso utilizamos o software Jude [11]. O Jude possui uma versão gratuita e uma versão paga (que é a profissional), utilizamos a versão gratuita.

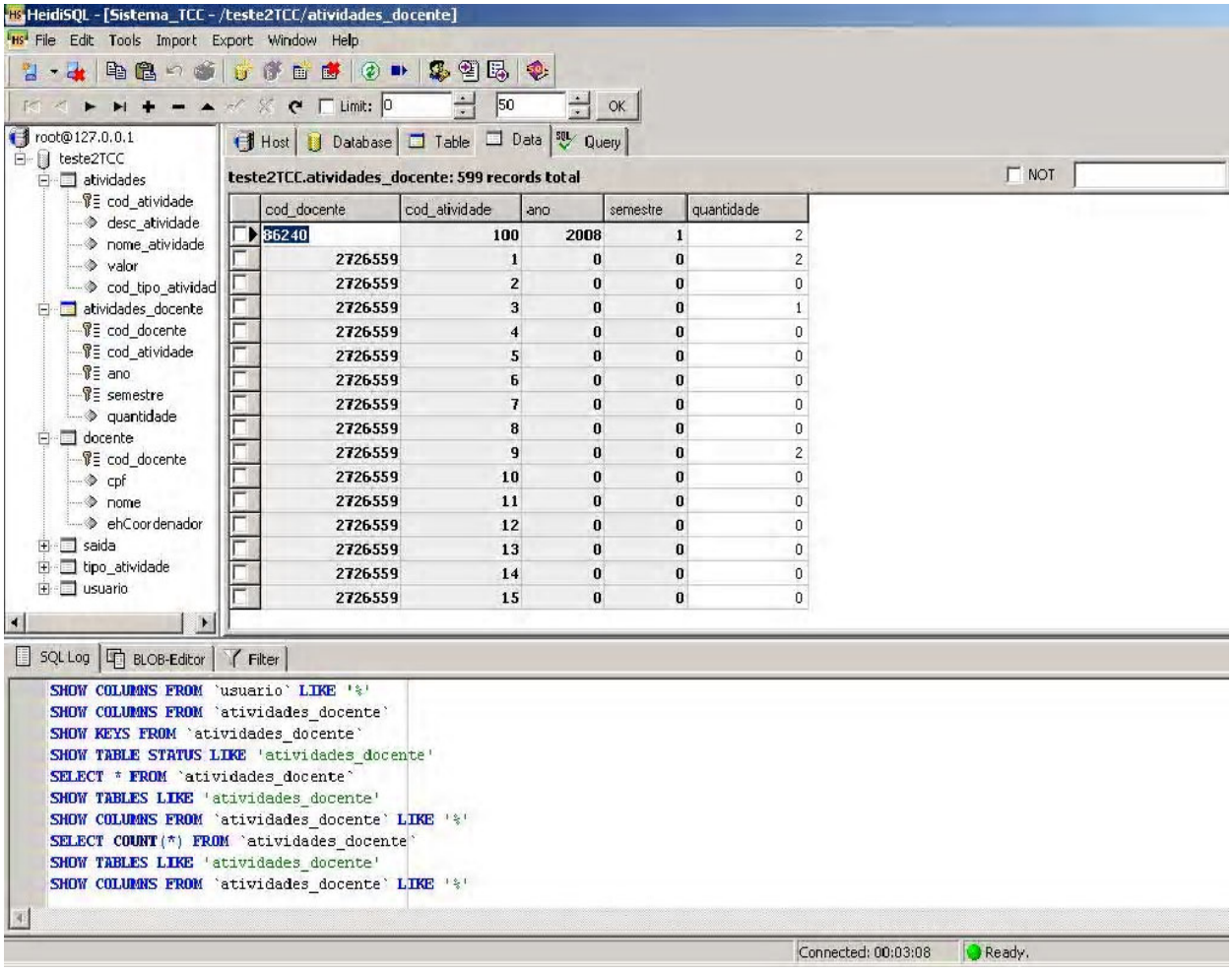


Figura 11. Screenshot do HeidiSQL.

4.6 Dificuldades Encontradas

Na fase de desenvolvimento encontramos algumas dificuldades, algumas já esperadas, outras que apareceram e não estavam no planejamento. Passada a fase de reuniões, onde

[11] http://jude.change-vision.com/jude-web/index.html

não tivemos tantas dificuldades, para definição dos requisitos, partimos para configurar o ambiente de desenvolvimento. Pela falta de experiência, tivemos que arriscar em algumas decisões a serem tomadas. Em algumas acertamos, em outras tivemos que refazer o trabalho optando por outras formas. Isso foi considerado normal, já que, pela primeira vez, desempenhamos o papel de desenvolvedor e gerente de projetos.

Enfrentamos um pouco de burocracia para conseguirmos sincronizar o nosso sistema com os dados do SIGA e do Lattes. Entramos várias vezes em contato com a equipe responsável pelo Lattes e não obtivemos respostas. Após tentativas diversas de mandar e-mails sem respostas, surgiu a idéia de cada docente entrar na página do Lattes e salvar seu currículo no formato XML. Com essa decisão conseguimos alcançar os nossos objetivos. Porém já tínhamos perdido um bom tempo tentando obter contato com os responsáveis pelo Lattes.

Para ter acesso aos dados do SIGA, também encontramos burocracia. Antes de falar com o pessoal responsável pelo SIGA, tivemos que fazer um documento relatando a necessidade de ter acesso aos dados e explicando os motivos. Esse documento foi entregue ao Diretor da Escola Politécnica de Pernambuco (POLI), ele concordou e assinou o documento. Só assim conseguimos falar com os responsáveis pelo SIGA e marcamos uma reunião onde explicamos nossas necessidades e solicitamos os dados que nos eram precisos.

A tentativa de obter contato com a CAPES foi frustrada. Devido a questões burocráticas e que não estavam ao nosso alcance, tivemos que desistir de integrar os dados da CAPES com os dados de nosso sistema.

Outra dificuldade foi em relação a algumas tecnologias usadas no desenvolvimento dos sistemas. Umas delas era o Struts, que já tínhamos usado, mas fazia um bom tempo e não se tinha um conhecimento necessário no momento.

4.7 Implantação do Sistema

Para implantar o sistema em alguma máquina servidora será preciso ter um *web container* "rodando" na máquina, a máquina virtual Java, um banco de dados MySQL e o arquivo da extensão ".war" da nossa aplicação. Um *web container* é o módulo do servidor de aplicação que corresponde ao servidor web tradicional. A extensão ".war" é

o arquivo compactado de nossa aplicação, é um arquivo com formato especial (*web arquive*).

Após fazer o download de tudo que for preciso para funcionar nossa aplicação, vamos agora instalá-los. Organizamos o passo a passo da instalação em uma seqüência de passos para facilitar o entendimento:

- Instale primeiro o JRE na sua máquina, depois instale o Tomcat (nosso servidor de aplicação), e o MySQL. O JRE é um Ambiente de Tempo de Execução Java, e é utilizado para executar as aplicações da plataforma Java. É composto por bibliotecas (APIs) e pela Máquina virtual Java (JVM).
- Estando com os aplicativos devidamente instalados e tendo em mãos o arquivo compactado da nossa aplicação (avaliação.war), coloque esse arquivo dentro da classe webapps do Tomcat. Essa pasta fica no diretório padrão de instalação do Tomcat. Por exemplo, C:\Program Files\ Tomcat 5.5\webapps.
- Depois selecione o nosso banco de dados e coloque na pasta "data" do diretório de instalação do MySQL (por exemplo C:\mysql\data). Ou então, de posse do arquivo com o script SQL dos dados, abra o software HeidiSQL e execute esse arquivo, após isso o banco será criado com todas as tabelas e dados já existentes no nosso banco de dados.
- O próximo passo agora é iniciar o Tomcat. É só executar o programa tomcat.exe na pasta "bin" do diretório de instalação do Tomcat, espere o Tomcat inicializar por completo e, após isso, siga para o próximo passo.
- Para acessar a nossa aplicação, vá ao endereço http://IP_DA_MAQUINA:8080/avaliacao. Onde IP_DA_MAQUINA é o endereço IP da máquina onde você hospedou o sistema. Para acessar da mesma máquina basta digitar http://localhost:8080/avaliacao.

Capítulo 5

Conclusão e Trabalhos Futuros

Este trabalho procurou desenvolver dois sistemas de informação que se integram para facilitar o gerenciamento do DSC. O primeiro sistema possibilitará o controle das atividades desenvolvidas pelos docentes para a avaliação dos mesmos. O segundo sistema permitirá a organização de uma fila de espera para os docentes que desejem se afastar do departamento por algum motivo, por exemplo, um doutorado, especialização.

5.1 Contribuições

O resultado desde trabalho serão dois sistemas integrados que permitirão, ao coordenador, uma maior organização e o controle do departamento através de uma ferramenta simples. Algumas atividades que eram executadas manualmente e de maneira cansativa, agora são realizadas no sistema de maneira mais rápida e eficaz.

O sistema ajudará na tomada de decisões em prol do DSC. Através do sistema serão realizadas avaliações dos docentes, o resultado dessas avaliações poderá ser impresso ou salvo para um acompanhamento posterior. Além disso, o sistema permitirá que algum docente possa se afastar do departamento, de maneira organizada e controlada, sem prejuízo algum para o DSC. Para o desenvolvimento do sistema, utilizamos o modelo MVC e algumas metodologias de processo, como o RUP.

5.2 Considerações Finais

Os sistemas resultantes desse trabalho auxiliarão bastante no controle de gerenciamento do DSC, e com isso, contribuirão para um crescimento e sucesso ainda maior que a

situação atual. O sistema auxiliará no controle das atividades desenvolvidas pelos docentes, permitindo assim que o coordenador faça avaliações periódicas dos docentes para acompanhar o desempenho dos mesmos. O sistema permitirá ainda que se tenha um controle dos docentes que desejem se afastar do departamento para terminar um doutorado, por exemplo.

O sistema automatizou algumas funcionalidades que, antes, eram feitas manualmente perdendo tempo e executando tarefas cansativas. Fazem parte dessa automação de funcionalidades, o *Upload* dos arquivos XML e txt que pegam os dados contidos nos mesmo e salvam no banco de dados, evitando assim que o docente tenha que digitar todas as atividades que desempenha no departamento.

5.3 Trabalhos Futuros

Como foi citado no capítulo anterior, por problemas burocráticos, tivemos que mudar um pouco os planos com relação aos requisitos levantados inicialmente. Por não conseguirmos obter sucesso no contato com a equipe responsável pela CAPES, tivemos que desistir de integrar nosso sistema com a base de dados da CAPES.

Pensando nisso, pensamos que uma forma interessante de dar continuidade a esse sistema é conseguir fazer essa integração do nosso sistema com os dados do CAPES. A sincronização com os dados do CAPES tornaria o sistema praticamente todo automatizado, já que os docentes não precisariam cadastrar as atividades que estivesse nos dados da CAPES, isso seria excelente para os docentes e o coordenador do DSC, pois não precisariam mais perder tempo cadastrando todas as suas atividades. Dentre as atividades que se encontram no CAPES, estão artigos publicados, aceitos em conferências e revistas.

Bibliografia

[1] Sistemas de Informação, Wikipedia, disponível na Url: http://pt.wikipedia.org/wiki/Sistemas_de_Informa%C3%A7%C3%A3o. Acesso em: 17/03/2008

[2] POLI DIGITAL, Sistema sig@ poli, disponível na Url: http://www.siga.poli.br/poli/principal.jsp . Acesso em : 13/03/2008

[3] Plataforma Lattes, disponível na Url: http://lattes.cnpq.br/index.htm. Acesso em: 13/03/2008.

[4] LARMAN, Craig. Applying UML and patterns: an introduction to object-oriented analysis and design and interative development. 3 ed. Massachussetts: Prentice Hall, 2004. 702p.

[5] Framework Struts, disponível na Url: http://struts.apache.org/. Acesso em: 13/03/2008.

[6] Modelo MVC, disponível na Url: http://www.tecnoclasta.com/2008/02/14/aula-14-o-modelo-mvc/. Acesso em: 13/03/2008.

[7] UML, disponível na Url: http://pt.wikipedia.org/wiki/UML. Acesso em: 17/03/2008.

[8] Processos de Software, Wikipedia, disponível na Url: http://pt.wikipedia.org/wiki/Engenharia_de_software#Processo_de_Software. Acesso em: 18/03/2008.

[9] Desenvolva Iterativamente, disponível na Url: http://www.wthreex.com/rup/manuals/intro/im_bp1.htm. Acesso em: 18/04/2008

[10] IBM Rational Unified Process (RUP), disponível na Url: http://pt.wikipedia.org/wiki/RUP. Acesso em: 04/05/2008.

[11] XML, Wikipedia, disponível na Url: http://pt.wikipedia.org/wiki/XML. Acesso em: 18/03/2008.

[12] TENENBAUN, Andrew S. Redes de Computadores. 3ª ed. Rio de Janeiro: Campus, 1997,923p.

[13] MVC – detailed. Disponível na Url: http://java.sun.com/blueprints/patterns/MVC-detailed.html. Acesso em: 08/05/2008.

[14] HUSTED, Ted; DUMOULIN Cedric; FRANCISCUS, George; WINTERFELDT David. Struts em ação. . Rio de Janeiro: Editora Ciência Moderna, 2004. 604p.

[15] Eclipse, an open development platform, disponível na Url: http://www.eclipse.org/. Acesso em: 07/05/2008.

[16] Struts na Prática de ponta a ponta, disponível na Url: http://www.linhadecodigo.com.br/Artigo.aspx?id=1045&pag=1. Acesso em 07/05/2008.

Anexo 1

Telas do Sistema

O propósito desse anexo é mostrar algumas telas mais importantes do sistema. A Figura 13 mostra a tela de *Upload* dos Dados do SIGA.

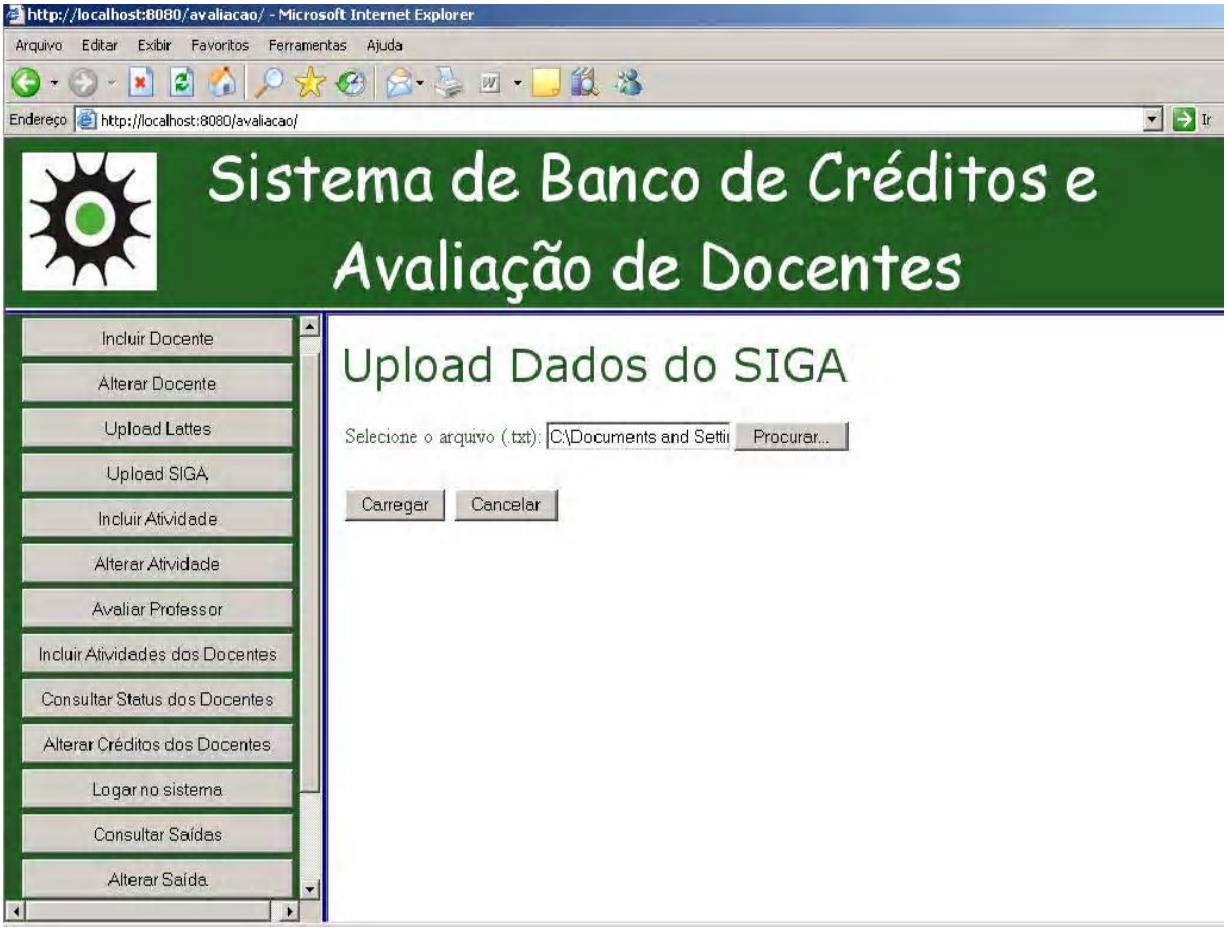


Figura 12. Screenshot da Tela de *Upload* do Lattes.

A Figura 14 mostra o resultado da Avaliação dos Professores, no detalhe está a tela onde temos a opção de salvarmos o resultado da avaliação nos formatos txt ou html. Primeiro escolhemos o diretório onde salvaremos o arquivo, depois escrevemos o nome e o tipo do arquivo a ser salvo.

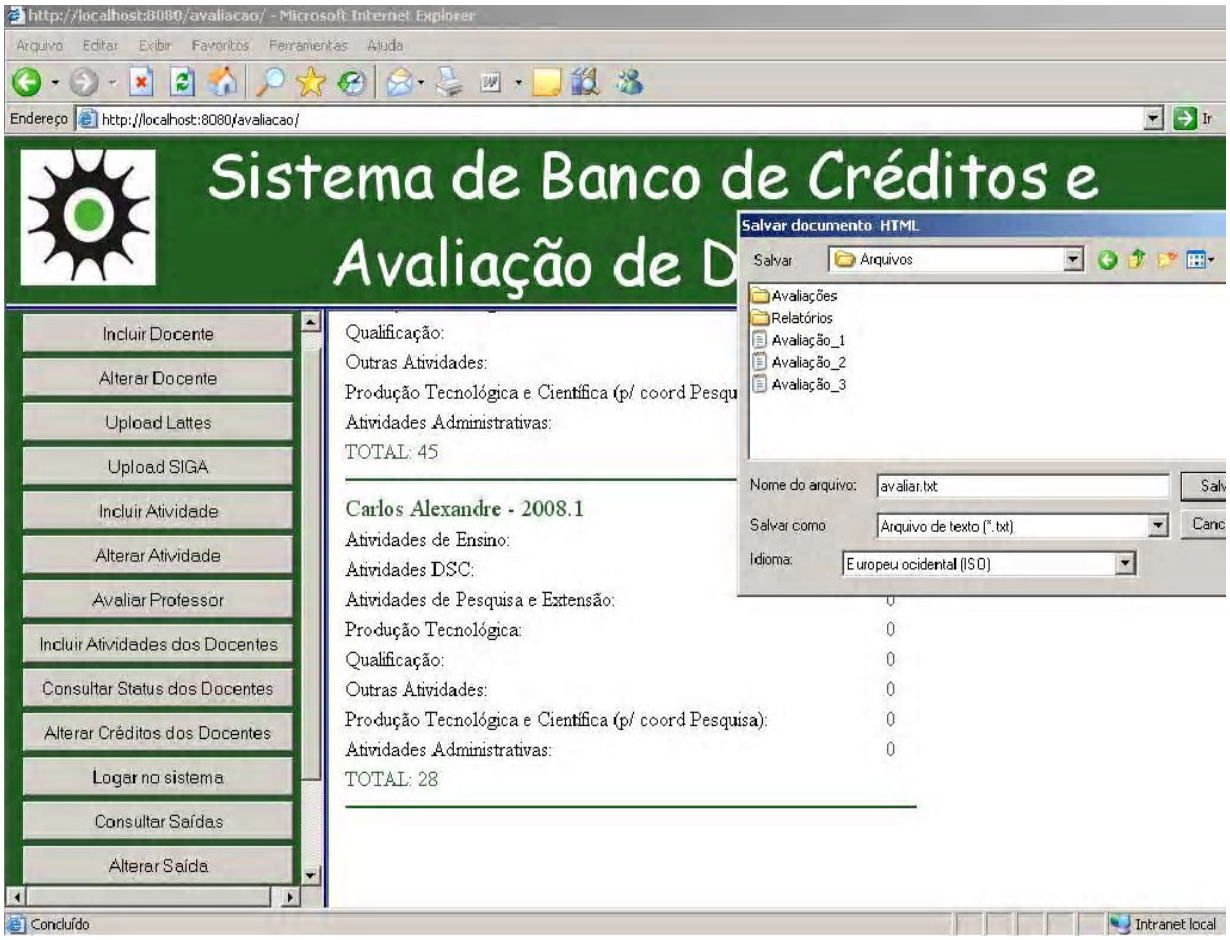


Figura 13. Screenshot da Tela de Avaliação com a possibilidade de salvar o resultado.