Resumo

Com o surgimento do programa de Mestrado em Engenharia da Computação do Departamento de Sistemas e Computação (DSC) da Escola Politécnica de Pernambuco, se torna essencial o desenvolvimento de um sistema que torne possível que as inscrições a este programa sejam feitas de forma *on-line* e que o modo de seleção dos candidatos ao programa seja feito de forma mais ágil possível. O propósito deste trabalho é o desenvolvimento de um sistema *web* que possa facilitar as inscrições ao Mestrado oferecido pelo DSC, assim como auxiliar a seleção dos candidatos, que hoje em dia é feita de forma manual com a ajuda de planilhas. As tecnologias utilizadas no desenvolvimento deste sistema visam tornam o sistema flexível e de fácil utilização, contribuindo então, para que estes objetivos sejam alcançados.

Abstract

With the sprouting of the Masters program in Computer Engineering of the Computer and System Department (DSC) of the Polytechnic School of Pernambuco, it becomes essential to develop a system that can make it possible that the applications to this program could be filled on-line and also the student selection process could be as quick and easy as possible. The purpose of this work is to develop a web based system which can make easier the applications to the Masters program of this department and also can help the student selection process which, until now, has been done manually with the aid of spreadsheets. The technologies that were used to develop this system make it very flexible and easy to use which contributes to reaching the main goals.

Sumário

Índice de Figuras v

Índice de Figuras v

Índice de Tabelas vi

Tabela de Símbolos e Sigla vii

1 Introdução 9

1.1 Motivação 10
1.2 Contribuições 11
1.3 Estrutura do Trabalho 11

2 Estado da Arte 12

2.1 Tecnologias 12
2.1.1 JavaServer Pages 12
2.1.2 Struts 15
2.1.3 Tiles 18
2.1.4 Hibernate 18
2.1.5 MySQL 20
2.2 RUP 22
2.3 Tecnologias Alternativas 26
2.3.1 PHP 27
2.3.2 ASP 27
2.4 Justificativa das tecnologias escolhidas 28

3 SGIP 29

3.1 Sistema de Informação 29
3.1.1 Requisitos do Sistema 29
3.1.2 Casos de Uso 31
3.2 Metodologia 33
3.2.1 Cronograma de Atividades 34
3.3 Diagramas 35
3.3.1 Diagrama de Casos de Uso 36
3.3.2 Diagramas de Classes 37
3.3.3 Diagramas de Seqüência 38
3.4 Arquitetura do SGIP 40
3.5 Interface do SGIP 41
3.5.1 Caso de uso: Incluir Candidato 41

4 Conclusões e Trabalhos Futuros 47

4.1 Contribuições 47
4.2 Considerações 47
4.3 Trabalhos Futuros 48

Anexo 1 Documento de Especificação de Casos de Uso 50
Anexo 2 Documento de Análise de Casos de Uso 74
Anexo 3 Documento de Arquitetura do SGIP 75

Índice de Figuras

Figura 1. Planilha de auxilio para seleção do Mestrado. 10
Figura 2. Gerando conteúdo dinâmico com elementos JSP [6]. 13
Figura 3. Elementos JSP [6]. 13
Figura 4. Exemplo de struts-config.xml. 16
Figura 5. Diagrama de seqüência indicando o fluxo realizado pelo Struts [7]. 17
Figura 6. Tabela Professor. 19
Figura 7. Arquivo Professor.hbm.xml. 19
Figura 8. Entidade Usuario. 21
Figura 9. Relacionamento entre tabelas. 21
Figura 10. Exemplo de relacionamento Um para Vários. 21
Figura 11. Exemplo de relacionamento Vários para Vários. 22
Figura 12. Arquitetura do RUP [18]. 23
Figura 13. Fases e marcos de um projeto [9]. 24
Figura 14. Fluxo de trabalho principal do RUP [19]. 25
Figura 15. Fluxo de trabalho principal detalhado do RUP [19]. 26
Figura 16. Fluxograma do processo de inscrição e seleção. 31
Figura 17. Diagrama de Casos de Uso do SGIP. 37
Figura 18. Exemplo de Diagrama de Classes. 38
Figura 19. Exemplo de Diagrama de Seqüência. 40
Figura 20. Arquitetura do SGIP. 40
Figura 21. Termo de aceitação para a inscrição *on-line*. 42
Figura 22. Formulário para cadastramento das informações básicas do candidato. 43
Figura 23. Formulário para cadastramento da formação acadêmica do candidato. 44
Figura 24. Formulário para cadastramento dos dados curriculares do candidato. 45
Figura 25. Formulário para cadastramento das preferências do candidato em relação à inscrição. 46

Índice de Tabelas

Tabela 1. Elementos de diretivas. 14
Tabela 2. Elementos de ação padrões. 14
Tabela 3. Elementos de *script*. 15
Tabela 4. Relação das principais classes do Struts e o modelo MVC 15
Tabela 5. Arquivos de configuração do Struts. 16
Tabela 6. Distribuição de esforços e programação no RUP 24
Tabela 7. Comparação de algumas das mais populares linguagens *web*. 28
Tabela 8. Cronograma de Atividades. 34

Tabela de Símbolos e Sigla

WEB - Sinônimo de WWW que por sua vez significa World Wide Web (A Grande Rede Mundial).
DSC - Departamento de Sistemas e Computação – Escola Politécnica de Pernambuco.
JSP - *JavaServer Pages* (Páginas de Servidores Java).
HTML - *Hiper Text Mark-up Language* (Linguagem de Marcação de Hiper Texto).
XML - *eXtensible Mark-up Language* (Linguagem de Marcação Extensível).
J2EE - *Java 2 Enterprise Edition* (Java Edição Empresarial).
MVC - *Model-View-Control* (Modelo-Visão-Controle).
URI - *Uniform Resource Identifiers* (Identificadores Uniformes de Recursos).
API - *Application Programming Interface* (Interface de Programação de Aplicativos).
SQL - *Structured Query Language* (Linguagem de Consulta Estruturada).
JDBC - *Java Database Connectivity* (Conectividade Java à Base de Dados).
DTD - *Document Type Definition* (Definição de Tipo de Documento).
HQL - *Hibernate Query Language* (Linguagem de Consulta Hibernate).
SGBD - Sistema de Gerenciamento de Banco de Dados.
RUP - *Rational Unified Process* (Processo Unificado da Rational).
UML - *Unified Modeling Language* (Linguagem de Modelagem Unificada).
PHP - *Hypertext Preprocessor* (Preprocessador de Hipertexto).
PHP-GTK - *Hypertext Preprocessor Graphics Toolkit* (Ferramentas Gráficas para Preprocessador de Hipertexto).
ASP - *Active Server Pages.*
OMG - *Object Management Group.*

Agradecimentos

Em primeiro lugar, a Deus que com Sua misericórdia divina fez com que tudo fosse possível. Nele temos a certeza de que podemos tudo.

A meus pais e irmão que sempre me apoiaram em absolutamente tudo, sempre com muito amor e compreensão mesmo nos momentos mais difíceis.

Ao prof. Sérgio Soares, pela paciência na orientação deste trabalho. Como também ao prof. Renato Moraes, que me ajudou muito ao tirar minhas dúvidas sobre o sistema a ser desenvolvido.

A todos os amigos e funcionários da Poli que muitas vezes, por uma atitude, tornaram esses anos mais agradáveis.

A Marcela, fiel amiga e companheira de estudos cuja amizade foi essencial durante todo o curso e será sempre importante na minha vida.

A André, a quem tenho muito que agradecer, pois sua ajuda foi essencial para a conclusão deste trabalho.

E a Máfia, grupo de excelentes amigos que fiz durante esses quase seis anos de curso, assim como todos os seus “agregados”. Espero sempre contar com todos!

Capítulo 1

Introdução

A busca pela economia de tempo, racionalização das atividades e maior eficiência na execução de tarefas são cada vez mais freqüentes por parte de indivíduos, empresas e instituições. Nesse sentido, o computador tem se apresentado como um instrumento facilitador, capaz de oferecer recursos para os mais diferentes tipos de necessidades [1]. Dentre os vários serviços oferecidos pelo computador, destacam-se os Sistemas de Informação (SIs), os quais permitem o gerenciamento e controle de vários processos. Um bom sistema de informação ajuda a empresa a atingir as suas metas [2].

Um SI pode ser então definido como todo sistema usado para prover informação, incluindo o seu processamento, e que possui vários elementos inter-relacionados que coletam, manipulam e armazenam, disseminam os dados e informações e fornecem um mecanismo de *feedback* [1]. Os *feedbacks* provenientes dos resultados normalmente estão atrelados a mais de uma pessoa, quer envolvida diretamente ou não, e podem estar relacionados com a qualidade e produtividade de uma organização [1][2].

São muitos os benefícios oferecidos pelos SIs. Dentre estes benefícios, destacam-se: garantia de maior segurança na recuperação das informações, acesso rápido às informações, aperfeiçoamento das comunicações, menor número de erros dando uma maior segurança à tomada de decisão, maior precisão nos dados obtidos com veracidade e integridade das informações, administração mais eficiente de informação e, conseqüentemente, carga de trabalho reduzida [3].

Desta forma, os SIs baseados na *web* estão ganhando mais importância a cada dia devido a uma característica fundamental: alta disponibilidade a um grande número de informações a todo momento para usuários com diferentes necessidades.

O número de usuários da *web* vem crescendo gradativamente, fator fundamental para que empresas e organizações queiram migrar seus sistemas para este ambiente. Uma razão bastante significativa foi o fato dos usuários poderem interagir com o ambiente *web*, submetendo formulários aos servidores e recebendo informações dinâmicas e processadas em tempo real [4].

Dessa forma, a *web* passou a se transformar em uma nova alternativa para o desenvolvimento de SIs que, baseados agora nesta tecnologia, e utilizando seus protocolos de comunicação, permitem que o acesso à informação seja distribuído geograficamente. As barreiras internas das redes empresariais deixam de ser impedimento e a abrangência global da *web* permite que clientes e fornecedores, empresas e consumidores, encontrem-se a mesma distância, gerando novas oportunidades de negócios [5].

Este capítulo visa descrever as principais motivações, contribuições e objetivos deste trabalho: o desenvolvimento do Sistema de Gerenciamento de Inscrições e Seleção de Pós-Graduação (SGIP) - SI baseado na *web* - para mostrar como este garantirá uma melhoria considerável na forma como são feitas as inscrições para os cursos de Pós-Graduação oferecidos pelo Departamento de Sistemas e Computação (DSC) da Universidade de Pernambuco nos dias de hoje.

1.1 Motivação

O Departamento de Sistemas e Computação (DSC) da Universidade de Pernambuco, a partir de abril de 2006, deu início ao primeiro programa de Mestrado em Engenharia da Computação do Nordeste. Este fato é uma prova do alto nível do departamento que foi criado oficialmente em setembro de 2004.

Hoje em dia, o processo de inscrição dos candidatos ao programa de Mestrado do DSC é feito de maneira burocrática e manual o que dificulta o processo e o torna custoso. O candidato deve preencher um formulário de inscrição indicando três projetos aos quais deseje se candidatar. Deve, então, levar este formulário já preenchido e uma série de outros documentos na Secretaria de Pós-Graduação do departamento ou então, enviá-los pelos Correios só para que possa estar participando do processo.

Também o processo de seleção dos candidatos feito pelo coordenador do processo vigente é através de planilhas como mostra a Figura 1.

	A	B	C	D	E	F	G	H	I	J
1										
2										
3		10	0	0	6.83	1	3	10	0	5.4645833
4		0	0	0	8.12	0.6	0		10	#VALOR!
5		7	0	0	7.22	0.8	0	10	10	6.8283333
6		10	0	0	8.28	1	0	8	10	7.5583333
7		0	0	0	7.77		0		10	#VALOR!
8		7	0	0	8.31	0.6	0	8	10	6.2470833
9		7	0	0	7.22	0.8	2	10	10	6.745
10		7	0	0	7.12	0.8	0	10	10	6.8116667
11		0	0	0	8.84		0		10	#VALOR!
12		0	0	0	????		0		10	#VALOR!
13		10	0	0	7.27	1	0	8	10	7.3479167
14		7	2	0	6.32	0.8	3	10	10	6.97
15					4.98	0.8	10			#VALOR!
16		10	0	0	7.27	1	5	10	10	7.55625
17		7	0	0	6.8	0.8	0	10	10	6.7583333

Figura 1. Planilha de auxilio para seleção do Mestrado.

Como pode ser visto na figura[1], esse processo se torna muito trabalhoso uma vez que a informação sobre os candidatos é passada uma a uma pelo coordenador para que se possa ter um *ranking* final desses candidatos.

Sendo o DSC um departamento de alto nível e prestigio na instituição, é de grande importância que ocorra um melhoramento nestes processos o que poderá acarretar em um melhor aproveitamento do tempo e diminuição dos custos do processo seletivo visando o menor esforço e burocracia possíveis.

[1] Por se tratar de uma planilha com informações confidenciais, um filtro foi aplicado à imagem fazendo com que alguns dados da figura se tornassem “ilegíveis”.

1.2 Contribuições

As principais contribuições esperadas com este trabalho são:

- Organizar as informações relacionadas ao curso de Mestrado do DSC em um sistema que poderá ser acessado via *web* a qualquer hora por professores e coordenadores.
- Diminuir os esforços por parte do candidato ao ter a oportunidade de inscrever-se de forma *on-line*.
- Diminuir o trabalho do coordenador de seleção do Mestrado fazendo com que as informações sejam passadas ao sistema de forma menos custosa do que a forma como é passado hoje, tornando esta tarefa mais agradável.
- Apresentar como os conceitos de Engenharia de *Software* vistos durante o curso de Engenharia da Computação podem ser colocados em prática através do desenvolvimento de aplicações *web*.

1.3 Estrutura do Trabalho

No capitulo 2, daremos um embasamento teórico das tecnologias utilizadas para o desenvolvimento do SGIP, explicando de forma resumida os conceitos relativos a programação Orientada a Objetos e a *web*. Falaremos sobre a criação de paginas web utilizando o *JavaServer Page* [6] e *frameworks* como o Struts [7] e o Hibernate [8], como também do *framework* de processo RUP [9] que será utilizado como metodologia de desenvolvimento neste trabalho. Por fim, algumas tecnologias opcionais serão apresentadas e uma breve comparação será feita para que se mostre o porquê da utilização das tecnologias escolhidas.

O capitulo 3 terá o enfoque nas atividades desenvolvidas no decorrer do trabalho explicando um pouco mais os requisitos associados ao sistema, detalhes de implementação e arquitetura e artefatos gerados.

No capitulo 4 são apresentadas as contribuições e conclusões deste trabalho, assim como sugestões para trabalhos futuros.

Capítulo 2

Estado da Arte

O objetivo deste capítulo é apresentar as principais tecnologias envolvidas na implementação do sistema de informação que será o produto deste trabalho, assim como a metodologia de desenvolvimento de *software* que será empregada. Ao final, faremos uma rápida comparação com outras tecnologias mais populares disponíveis no mercado e que são utilizadas para o mesmo fim.

2.1 Tecnologias

Esta seção visa passar o conhecimento necessário para que se entendam todas as tecnologias que serão empregadas no desenvolvimento do sistema.

2.1.1 JavaServer Pages

JavaServer Pages (JSP) é uma tecnologia utilizada para criação de páginas *web* de conteúdo dinâmico. Diferente de páginas HTML (*HiperText Markup Language*) puras, que possuem um conteúdo estático o qual permanece sempre o mesmo, uma página JSP pode modificar seu conteúdo dependendo de vários aspectos, tais como identificação do usuário, tipo de *browser* utilizado, informações providas e seleções feitas pelo usuário. Funcionalidades como estas, podem ser úteis para criar diversos tipos de aplicações *web*, tais como sistemas de compras *on-line*.

Uma página JSP contém elementos de marcação padrões, assim como as *tags* HTML, como qualquer página *web*. No entanto, uma página JSP contém também elementos especiais que permitem que o servidor insira conteúdo dinâmico em uma página, conteúdo este que será provido por programas escritos na linguagem Java encontrados no servidor. Os elementos de páginas JSP podem ser usados com vários propósitos, tanto para recuperação de informações de uma base de dados, como também registrar informações na mesma base [10]. Quando um usuário faz uma requisição de uma página JSP, o servidor executa os elementos JSP específicos, compõe o resultado dessa execução com o conteúdo estático da página, e envia a página dinamicamente composta para o *browser*, assim como mostrado na Figura 2.

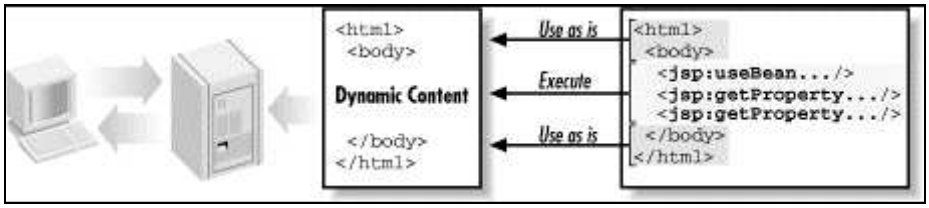


Figura 2. Gerando conteúdo dinâmico com elementos JSP [6].

JSP define um número grande de elementos padrões úteis para qualquer tipo de aplicação *web*, como acesso a componentes JavaBeans[2] e compartilhamento de informações de requisições, páginas e usuários [10].

Uma pagina JSP difere de uma pagina *web* padrão pelo uso de elementos particulares que são responsáveis por gerar o conteudo dinâmico que são diferentes em cada requisição, como já falados anteriormente e agora mostrados na Figura 3.

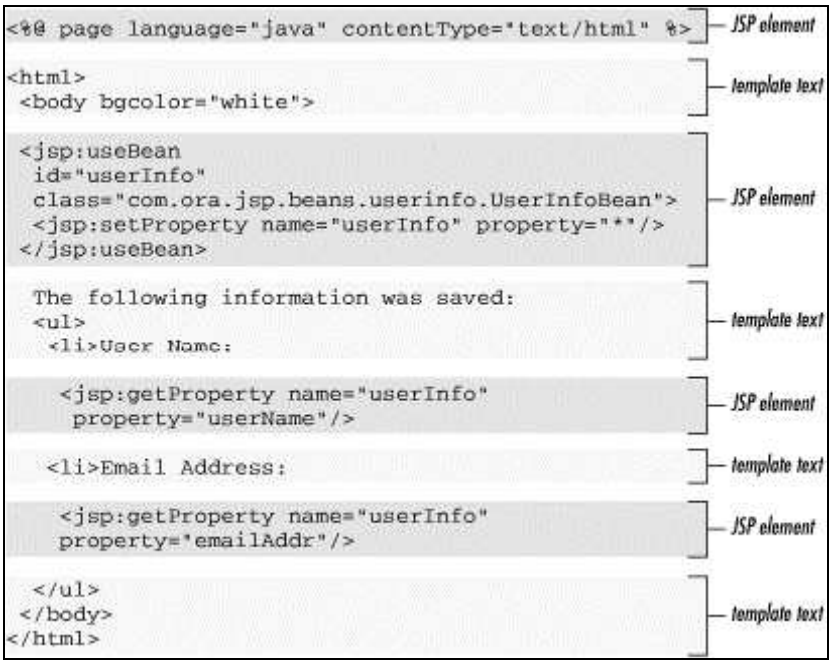


Figura 3. Elementos JSP [6].

Tudo que não é um elemento JSP na pagina, é chamado *template text*. Um *template text* pode ser qualquer tipo de texto: HTML, XML (*eXtensible Markup Language*), ou qualquer outra linguagem de marcação.

Quando uma requisição JSP é processada, o *template text* e o conteúdo JSP que foi gerado dinamicamente são automaticamente integrados, e o resultado é enviado como resposta ao *browser*.

Existem três tipos de elementos utilizados com JSP: elementos de diretivas, de ação e de *script* [6]. Os elementos de diretivas, mostrados na Tabela 1, são elementos os quais permanecem com o mesmo conteúdo entre varias requisições, como por exemplo, o tipo de linguagem de *script* utilizada e o nome de alguma página que deverá ser usada para reportar erros, se houver alguma.

---

[2] *JavaBeans* são componentes feitos em Java. Eles são projetados para ser independentes, reusáveis e modulares. Como qualquer classe Java, *JavaBeans* podem ser objetos visíveis. Eles podem ser criados e empacotados e, desta forma, distribuídos para outras pessoas poderem usá-los em outros projetos.

Tabela 1. Elementos de diretivas.

Elementos	Descricao
<%@ page ... %>	Define atributos que dependem da pagina, tais como: linguagem de *script* utilizada, página de erro, *buffering*, etc.
<%@ include ... %>	Inclue um arquivo durante o período de transição.
<%@ taglib ... %>	Declara uma biblioteca de *tags*, com ações customizadas, que serão utilizadas na página.

Elementos de ação geralmente fornecem alguma informação necessária no momento da requisição do cliente à página JSP. Um elemento de ação pode, ao ser instanciado, fazer uma busca na base de dados com informações que foram enviadas na requisição ou até mesmo gerar uma página HTML com campos preenchidos de informações recuperadas de outros sistemas, por exemplo.

A especificação do JSP inclui alguns elementos de ação padrões e ainda um *framework*[3] que auxilia o programador a extender a linguagem JSP e desenvolver seus próprios elementos com ações customizadas [6]. A Tabela 2 mostra alguns desses elementos.

Tabela 2. Elementos de ação padrões.

Elementos	Descricao
<jsp:useBean>	Faz com que uma classe Java fique disponível na página.
<jsp:getProperty>	Adiciona à resposta o valor de alguma propriedade da classe Java.
<jsp:setProperty>	Modifica um determinado valor de alguma propriedade da classe Java.
<jsp:include>	Inclue a resposta de alguma página JSP durante o período de processamento da requisição.
<jsp:forward>	Encaminha o processamento de uma requisição para alguma outra página JSP.
<jsp:param>	Adiciona algum parâmetro em uma requisição repassada a alguma outra página JSP através de um <jsp:include> ou <jsp:forward>.

Elementos de *script*, como mostrados na Tabela 3, permitem que pequenos trechos de código sejam adicionados à página JSP, como, por exemplo, uma declaração *if* que pode gerar diferentes códigos HTML dependendo que certas condições sejam alcançadas. Como os elementos de ação, os elementos de *script* também são executados quando a página JSP é requisitada.

[3] Um *framework* pode ser visto como uma aplicação reutilizável e semi-completa que pode ser especializada para produzir *softwares* personalizados.

Tabela 3. Elementos de *script*.

Elementos	Descricao
<% ... %>	*Scriptlet*, usado para incluir códigos do *script* utilizado na página.
<%= ... %>	Esse elemento introduz expressões Java quando o resultado dessa expressão deve ser adicionado à resposta.
<%! ... %>	Usado para declarar variáveis e métodos na página JSP que implementa uma determinada classe Java.

Os elementos JSP, como os elementos de ação e os de *script*, são utilizados para trabalhar diretamente com o objeto Java.

2.1.2 Struts

O Struts é um *framework* para desenvolvimento de sistemas para *web* que se tornou padrão para os desenvolvedores da comunidade J2EE (*Java 2 Enterprise Edition*[4]). O Struts provê uma implementação do modelo MVC (*Model-View-Controller*[5]) [11] para construções de aplicações *web* e pode ser visto como um *Servlet*[6] de controle que pode ser usado para gerenciar o fluxo entre páginas JSP e outras camadas de apresentação [7].

O Struts implementa o padrão MVC pelo uso de *ActionForwards* e *ActionMappings* para manter o fluxo de controle fora da camada de apresentação. Os JSPs irão apontar para destinos lógicos, enquanto os componentes da camada de controle irão prover as "verdadeiras" URIs (*Uniform Resource Identifier*) em tempo de execução.

A Tabela 4 mostra as principais classes do Struts e suas respectivas funções relacionadas ao padrão MVC.

Tabela 4. Relação das principais classes do Struts e o modelo MVC.

Classe	Descricao
ActionForward	Uma ação do usuário ou seleção de determinado componente visual.
ActionForm	Onde os dados para a mudança de estado são encontrados.
ActionMapping	Representa a informação que o Controlador sabe sobre o mapeamento de uma determinada requisção para uma

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

[5] Este padrão de desenvolvimento permite que engenheiros Java e os desenvovedores HTML trabalhem cada um em sua parte da aplicação. Desta forma, a aparência da aplicação pode ser alterada (HTML) sem que a funcionalidade ou fluxo de navegação precise acompanhar as alterações.

[6] *Servlets* são considerados extensões de servidores. Essa tecnologia disponibiliza ao programador da linguagem Java uma interface para o servidor web, através de uma API (*Application Programming Interface*). As aplicações baseadas no *Servlet* geram conteúdo dinâmico (normalmente HTML) e interagem com os clientes, utilizando o modelo requisição/resposta.

	nova instância.
ActionServlet	A parte do Controlador que recebe as ações do usuário, mudanças de estado e restrições do componente visual
Classes Action	A parte do Controlador que interage com a camada de Modelo para executar mudanças de estado, pedidos e avisa ao ActionServlet que componente visual deverá ser selecionado.

Em adição a estas classes, o Struts usa alguns documentos de configuração para unir a camada de controle à camada de visão. A Tabela 5 mostra os arquivos de configuração necessários assim como seu propósito.

Tabela 5. Arquivos de configuração do Struts.

Classe	Descricao
ApplicationResources.properties	Uma ação do usuário ou seleção de determinado componente visual.
struts-config.xml	Onde os dados para a mudança de estado são encontrados.

A Figura 4 mostra um exemplo do arquivo *struts-config.xml*. É nesse arquivo onde declaramos as classes *Actions* e *ActionsForms* e fazemos mapeamentos entre as camadas de controle e a de visão. Este arquivo armazena a configuração padrão para os objetos do controlador, que inclui as ações do usuário, as alterações de estado e as consultas de estados suportadas por esse modelo. A seção *<form-beans></form-beans>* é onde declaramos os nossos formulários (*ActionForm*) que usaremos na camada de visão, através da *tag <form-bean/>*. Utilizamos as tags *<action-mappings><actionmappings/>* para declararmos as ações (Classes *Action*) e seus caminhos, através da *tag <action />*. Como componente da *tag <action/>*, podemos encontrar a *tag <forward/>* que indica os possíveis encaminhamentos (*ActionForward*) dessa ação.

```
<struts-config>
   <form-beans>
      <form-bean name="nameForm" type="example.NameForm"/>
   </form-beans>
   <action-mappings>
      <action path="/Name" type="example.NameAction" name="NameForm" >
         <forward name="success" path="/quote.jsp"/>
         <forward name="failure" path="/index.jsp"/>
      </action>
   </action-mappings>

   <message-resources parameter="example.ApplicationResources"/>
</struts-config>
```

Figura 4. Exemplo de struts-config.xml.

Para facilitar o entendimento do Struts, observe na Figura 5 o processo de requisição-resposta apresentado em um diagrama de seqüência.

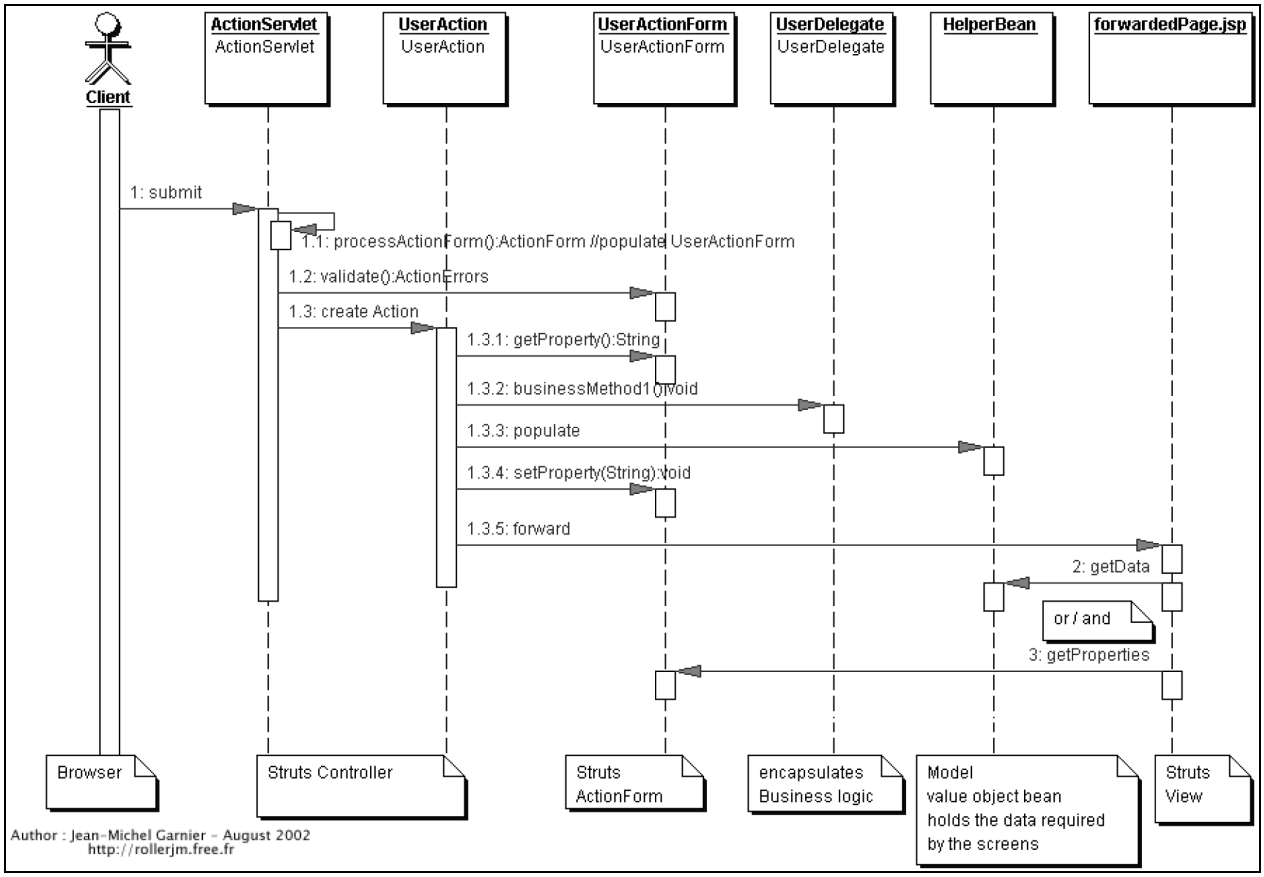


Figura 5. Diagrama de seqüência indicando o fluxo realizado pelo Struts [7].

As etapas serão descritas a seguir, de acordo com a numeração indicada na Figura 5:

- O cliente solicita o caminho (1).
- Se o mapeamento especificar um componente do formulário, o *ActionServlet* verá se já existe um, ou criará um (1.1).
- Se o mapeamento possuir a propriedade de validação definida para *true*; chamará essa validação no componente do formulário (1.2).
- Se o mapeamento especificar um tipo *Action*, uma classe Java será reutilizada, se alguma instância da classe já existir, ou será instanciada. (1.3).
- O *Action* recupera as propriedades do formulário (1.3.1), chama os métodos de negócio responsáveis pela validação dessas propriedades (1.3.2) e preenche os componentes do formulário (1.3.4).
- O *Action* retorna um *ActionForward* para o *ActionServlet* (1.3.5).
- Se o *ActionForward* for para outro *Action*, todo o processo será refeito; do contrário, será enviado para uma página de exibição ou algum outro recurso (2,3).

2.1.3 Tiles

Normalmente durante o desenvolvimento de uma aplicação *web*, o grupo responsável pela *interface* com o usuário cria todo o *design* da aplicação. Baseando-se neste *design*, este grupo cria páginas HTML que representam as funcionalidades do sistema e a forma de navegação. Com uma implementação baseada em *Servlet* e JSP, onde as páginas HTML são convertidas em *Servlets* e JSP, os *designers* da *interface* com o usuário identificam componentes HTML e JSP comuns, como o cabeçalho (*Header*), o corpo (*Body*), o rodapé da página (*Footer*) e o menu. Todos estes componentes passam, então, a serem repetidos em todas as páginas que são escritas e, à medida que este número de páginas cresce, estes códigos vão sendo replicados.

Muitas vezes o número excessivo de páginas incrementa a complexidade de desenvolvimento, de administração e manutenção da aplicação de maneira drástica, pois quanto mais código repetido, mais difícil será a tarefa de desenvolver e dar manutenção na aplicação. Uma simples modificação pode desencadear uma série de modificações em cascata com conseqüências imprevisíveis e quase sempre custosas. Logo, uma ótima maneira de se adquirir reusabilidade, é eliminando código repetido [7].

É nesse contexto que o Tiles *Framework* se apresenta como uma solução efetiva e eficiente para a organização dos componentes de apresentação da aplicação.

Através de elementos de marcação da biblioteca do Tiles, podemos inserir conteúdo estático em uma página JSP, conteúdo este que será repetido em todas as páginas do sistema a ser desenvolvido. O trecho de código abaixo mostra estes elementos de marcação:

```
<head><body>
     <tiles:insert attribute="head"/>
     <tiles:insert attribute="body"/>
     <tiles:insert attribute="footer"/>
</body></head>
```

Através da *tag* <*tiles:insert* /> podemos manipular os elementos da propriedade *attribute* e inserir elementos (outras páginas HTML) na página JSP em questão. Assim, quando alguma mudança tiver de ser feita nessas páginas que estão sendo inseridas, apenas modificaremos em um só arquivo ao invés de vários.

2.1.4 Hibernate

O Hibernate é um *framework* de mapeamento objeto/relacional para Java. Ela transforma os dados tabulares de um banco de dados em um conjunto de objetos definido pelo desenvolvedor. Usando o Hibernate, o desenvolvedor deixa de escrever códigos de acesso a banco de dados e de SQL (*Structured Query Language*), aumentando a velocidade de codificação e diminuindo os custos do desenvolvimento [8].

Melhor do que usar SQL e JDBC (*Java Database Connectivity*), podemos utilizar domínios de objetos Java e de forma simples criar arquivos de mapeamento baseados em XML [8]. Estes arquivos de mapeamento do Hibernate indicam quais campos (atributos de um objeto) serão mapeados para suas respectivas colunas (na tabela). Para esclarecer como ocorre o mapeamento de uma classe Java em uma tabela, tomemos como base o trecho de código da classe Professor.java abaixo:

```
public class Professor {

      private Long id;
      private String nome;
```

```
    private String email;

    ...

}
```

A tabela que representará essa entidade no banco é mostrada na Figura 6 e o mapeamento desta entidade pode ser visto no arquivo XML mostrado pela Figura 7.

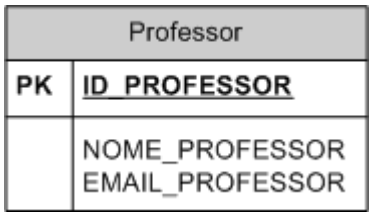


Figura 6. Tabela Professor.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>

<class name="Professor" table="PROFESSOR">

            <id column="ID_PROFESSOR" name="id" type="java.lang.Long">
                <generator class="sequence"/>
            </id>

            <property column="NOME_PROFESSOR" length="50" name="nome" not-null="true"
                type="java.lang.String" />

            <property column="EMAIL_PROFESSOR" length="50" name="nome" not-null="true"
                type="java.lang.String" />

</class>

</hibernate-mapping>
```

Figura 7. Arquivo Professor.hbm.xml.

Como podemos ver na Figura 7, o arquivo de mapeamento é um arquivo XML que define as propriedades e os relacionamentos de uma classe para o Hibernate. Este arquivo pode conter classes, classes componentes e consultas em SQL [12]. No exemplo, temos apenas uma classe Professor.java sendo mapeada no arquivo. O arquivo XML começa normalmente com as definições da DTD (*Document Type Definition*) e da *tag* raiz, o <*hibernate-mapping*>.

Na *tag* <*class*> nós definimos a classe que está sendo mapeada e para qual tabela ela vai ser mapeada. Seguindo em frente no exemplo, temos a *tag* <*id*> que é o identificador (chave) dessa classe no banco de dados. Nesta *tag* definimos a propriedade que guarda o identificador do objeto no atributo *name*, que no nosso caso é "id", e a coluna no banco de dados que irá identificar este atributo. As próximas *tags* do arquivo são as *tags* <*property*> que indicam propriedades simples dos nossos objetos, como Strings, tipos primitivos, objetos Date, Integer e outros. Nesta *tag* os atributos mais importantes são *name*, que define o nome da propriedade,

column utilizado quando a propriedade não tiver o mesmo nome da coluna na tabela (neste caso ele não precisa ser referenciado) e *type* para definir o tipo do objeto que a propriedade guarda.

Além desses arquivos de mapeamento, o Hibernate também possui uma poderosa linguagem de pesquisa chamada *Hibernate Query Language* (HQL). Esta linguagem permite possamos escrever códigos em SQL, mas também usar semânticas orientadas a objeto [8]. Assim sendo, o objetivo do Hibernate é diminuir a complexidade entre os programas Java, baseado no modelo orientado a objeto, que precisam trabalhar com um banco de dados do modelo relacional (presente na maioria dos SGDBs - Sistemas Gerenciadores de Banco de Dados). Em especial, no desenvolvimento de consultas e atualizações dos dados [12].

Sua principal característica é a transformação das classes Java para tabelas de dados (e dos tipos de dados Java para os da SQL).

A *interface Session* do Hibernate é similar a uma conexão com o banco de dados; ela tem de ser aberta e fechada nos tempos apropriados para prevenir erros e *leaks* de memória [8]. O código abaixo mostra um exemplo de um típico uso do Hibernate:

```
 1 ...
 2 SessionFactory sf = new Configuration().configure().buildSessionFactory();
 3 Session session = sf.OpenSession();
 4 Transaction tx = session.beginTransaction();
 5 Professor professor = new Professor();
 6 professor.setId(123456);
 7 professor.setNome("Sergio");
 8 professor.setEmail("sergio@dsc.upe.br");
 9 session.save(professor);
10 tx.commit();
11 session.close();
12 ...
```

De acordo com o trecho de código acima, o primeiro passo a ser feito é abrir uma sessão com o banco de dados (linhas 2 e 3) e então, através dessa sessão, acontecerá o mapeamento do objeto como foi mostrado anteriormente, alimentando o banco com os dados passados através do objeto.

O Hibernate gera as chamadas SQL e "libera" o desenvolvedor do trabalho manual da conversão dos dados resultante, mantendo o programa portável para quaisquer bancos de dados SQL, porém isto pode causar um pequeno aumento no tempo de execução [8].

2.1.5 MySQL

Um banco de dados é uma coleção integrada de dados. Um sistema de banco de dados envolve os próprios dados, o *hardware* em que os dados residem, o *software* que controla o armazenamento e a recuperação deles (SGBD) e os próprios usuários [10]. Dentre as várias vantagens de se usar um sistema de banco de dados, estão:

- A redundância pode ser reduzida – Em sistemas sem banco de dados, cada aplicativo distinto mantém seus próprios arquivos, frequentemente com redundância considerável e uma variedade de formatos físicos. Em sistemas de banco de dados, a redundância é reduzida integrando arquivos separados.
- Os dados podem ser compartilhados – Aplicativos separados podem acessar a mesma base de dados.
- Restrições de segurança podem ser aplicadas – Os sistemas de banco de dados devem ser projetados com controles de acesso sofisticado.

Toda a informação de um banco de dados relacional é armazenada em Tabelas, que na linguagem do modelo relacional, também são chamadas de Entidades [13]. A Figura 8 nos mostra um exemplo de Entidade, Usuario, onde todas as informações necessárias sobre os usuários do sistema serão armazenadas. Essas diversas informações são chamadas atributos da Tabela Usuario.

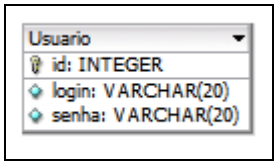


Figura 8. Entidade Usuario.

Embora as informações estejam separadas em cada uma das Tabelas, na prática devem existir relacionamentos entre as tabelas. Como por exemplo, na Figura 9 vemos o relacionamento entre a Entidade Usuario e a Entidade Professor.

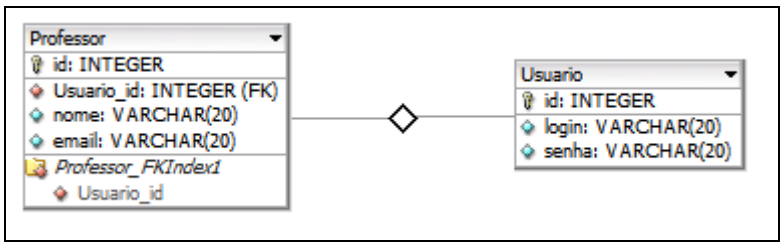


Figura 9. Relacionamento entre tabelas.

Os relacionamentos entre tabelas podem ser de três tipos:

1. Relacionamento do tipo Um para Um: É quando um relacionamento ocorre de uma única linha de uma tabela para uma única linha de outra tabela. É mais comum ocorrer quando dois campos são únicos em suas respectivas tabelas. A Figura 9 mostra um relacionamento Um para Um, onde um professor e cadastrado somente uma vez e será relacionado a um usuário que também e cadastrado somente uma vez no sistema.
2. Relacionamento do tipo Um para Vários: É o tipo de relacionamento mais comum entre duas tabelas. Uma das tabelas (o lado *um* do relacionamento) possui um campo que é único e a outra tabela (o lado *vários* do relacionamento) se relaciona através de um campo cujos valores relacionados podem se repetir várias vezes. A Figura 10 mostra um relacionamento Um para Vários, onde um professor pode esta ligado a um ou mais projetos de pesquisa.

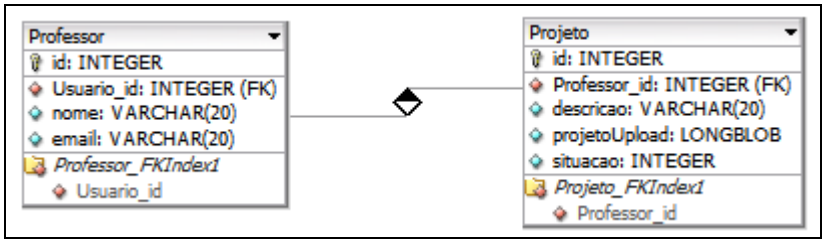


Figura 10. Exemplo de relacionamento Um para Vários.

3. Relacionamento do tipo Vários para Vários: Este tipo de relacionamento acontece em uma situação onde em ambos os lados do relacionamento os valores poderiam se repetir. Vamos considerar o caso entre Projetos e Candidatos. Podemos ter vários Projetos nos quais aparece um determinado Candidato, além disso, vários Candidatos podem aparecer no mesmo Projeto. Esta é uma situação em que temos um relacionamento do tipo Vários para Vários. Na prática, não é possível implementar um relacionamento deste tipo, devido a uma série de problemas que seriam introduzidos no modelo do banco de dados. Para evitar este tipo de problema é bastante comum transformarmos um relacionamento do tipo Vários para Vários em dois relacionamentos do tipo Um para Vários. Isso é feito através da criação de uma nova tabela, a qual fica com o lado *Vários* dos relacionamentos. Podemos ver um exemplo concreto no caso da Entidade CandidatoProjeto na Figura 11.

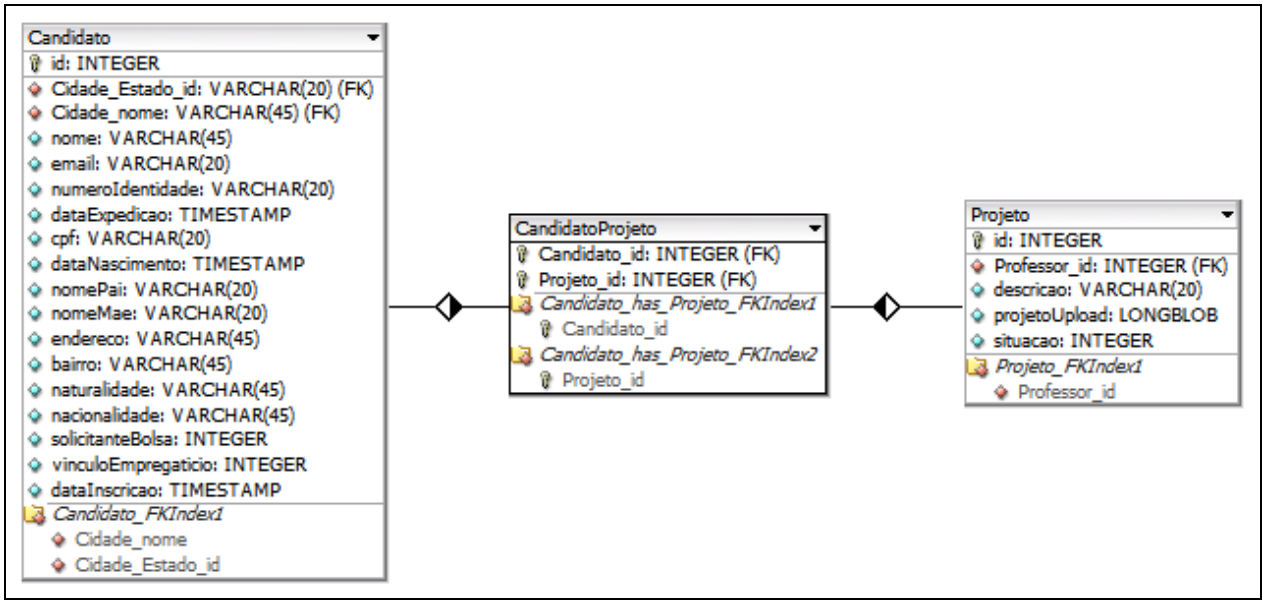


Figura 11. Exemplo de relacionamento Vários para Vários.

O MySQL [14] é um SGBD relacional, que utiliza a linguagem SQL como interface. É atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações pelo mundo [14]. SQL se diferencia de outras linguagens de consulta a banco de dados no sentido em que uma consulta SQL especifica a forma do resultado e não o caminho para chegar a ele. Ela, portanto, é uma linguagem declarativa em oposição a outras linguagens procedurais [13].

Servidores de banco de dados MySQL são rápidos e fáceis de usar. O MySQL possui código aberto[7] e pode funcionar em um grande número de sistemas operacionais.

2.2 RUP

Um grande problema na execução dos projetos da atualidade é o grande dinamismo e complexidade dos negócios que está cada vez maior. Cada vez mais os sistemas se tornam mais complexos e precisam estar prontos em menos tempo. Mais do que isso, as necessidades mudam

[7] O termo código aberto, ou *open source*, foi cunhado pela OSI (*Open Source Initiative*) e se refere ao mesmo *software* também chamado de *software livre*.

ao longo do tempo e a especificação de um sistema provavelmente será alterada durante seu desenvolvimento. Além disso, temos tecnologias novas (tanto de *software* como *hardware*) surgindo a cada dia. É nesse contexto que precisamos de um processo que seja seguido e que evite os riscos presentes em todo projeto de *software*.

O RUP, abreviação de *Rational Unified Process* (ou Processo Unificado da *Rational*), é um processo proprietário de engenharia de *software* criado pela *Rational Software Corporation*[8], adquirida pela IBM tornando-se uma *brand* na área de *software*, fornecendo técnicas a serem seguidas pelos membros da equipe de desenvolvimento de *software* com o objetivo de aumentar a sua produtividade e evitar riscos. Ele oferece uma abordagem baseada em disciplinas para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento [9]. Sua meta é garantir a produção de *software* de alta qualidade que atenda às necessidades dos usuários dentro de um cronograma e de um orçamento previsíveis [9].

O RUP usa a abordagem da orientação a objetos em sua concepção e é projetado e documentado utilizando a notação UML (*Unified Modeling Language*) [16] para ilustrar os processos em ação, utilizando técnicas e práticas aprovadas comercialmente [17].

A Figura 12 mostra a arquitetura geral do RUP. Na figura, o eixo horizontal representa o tempo e mostra os aspectos do ciclo de vida do processo à medida que se desenvolve. O eixo vertical representa as disciplinas, que agrupam as atividades de maneira lógica, por natureza. Este gráfico mostra como a ênfase varia através do tempo. Por exemplo, nas iterações iniciais, dedicamos mais tempo aos requisitos. Já nas iterações posteriores, se gasta mais tempo com o desenvolvimento da aplicação [17].

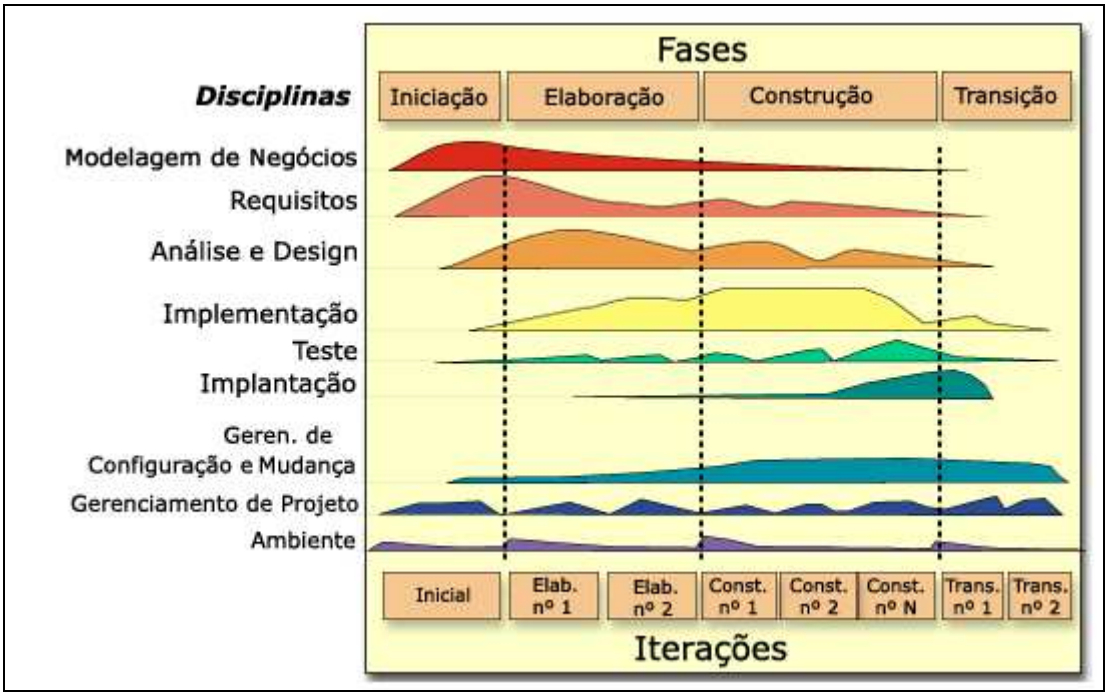


Figura 12. Arquitetura do RUP [18].

A partir de uma perspectiva de gerenciamento, o ciclo de vida de *software* do RUP é dividido em quatro fases seqüenciais, cada uma concluída por um marco principal, ou seja, cada fase é basicamente um intervalo de tempo entre dois marcos principais. Em cada final de fase é executada uma avaliação para determinar se os objetivos da fase foram alcançados. Uma

[8] *http://www-306.ibm.com/software/rational/*

avaliação satisfatória permite que o projeto passe para a próxima fase. A Figura 13 mostra as fases do RUP e seus marcos.

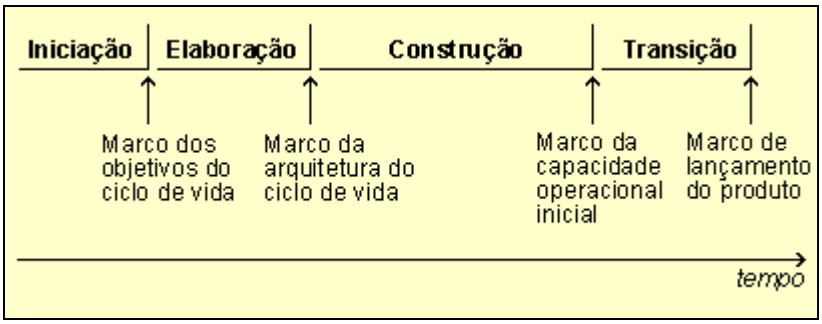


Figura 13. Fases e marcos de um projeto [9].

Cada fase pode ser entendida como:

- Iniciação: entendimento da necessidade e visão do projeto;
- Elaboração: especificação e abordagem dos pontos de maior risco;
- Construção: desenvolvimento principal do sistema;
- Transição: ajustes, implantação e transferência de propriedade do sistema.

Estas fases não são idênticas em termos de programação e esforço, embora isso varie muito de acordo com o projeto. A Tabela 6 mostra o quanto se deve prever de distribuição de esforço e programação em um ciclo de desenvolvimento inicial típico para um projeto de médio porte [9].

Tabela 6. Distribuição de esforços e programação no RUP.

	Iniciação	Elaboração	Construção	Transição
Esforço	~5%	20%	65%	10%
Programação	10%	30%	50%	10%

O RUP ainda conta com uma série de artefatos que podem ser entendidos como informações produzidas, modificadas ou utilizadas em um processo. Os artefatos são os produtos de um projeto e são produzidos durante toda a vida útil do projeto. Artefatos são utilizados como entradas de atividades e são produzidos como saída. Podem ter várias formas [19]:

- Um modelo: como um modelo de caso de uso, um modelo de projeto;
- Um elemento de um modelo: como uma classe, um caso de uso, um subsistema;
- Um documento: como um caso de negócio, glossário, visão;
- Código fonte;
- Executáveis.

A enumeração de atividades, papéis e artefatos, porém, não constituem um processo. É necessário saber a seqüência do desenvolvimento das atividades para que possam ser produzidos artefatos de valor para o projeto.

Um fluxo de trabalho[9] é uma seqüência de atividades que são executadas para a produção de um resultado valioso para o projeto. Fluxos de trabalho podem ser representados por

[9] O termo fluxo de trabalho vem do termo inglês *workflow*. Pode ser traduzido também como fluxo de atividade.

diagramas de seqüência, diagramas de colaboração e diagramas de atividades na linguagem UML. O RUP utiliza três tipos de fluxos de trabalho [15]:

- Fluxos de trabalho principais, associados com cada disciplina (Figura 14);
- Fluxos de trabalho de detalhe, para detalhar cada fluxo de trabalho principal (Figura 15);
- Planos de iteração, que mostram como a iteração deverá ser executada.

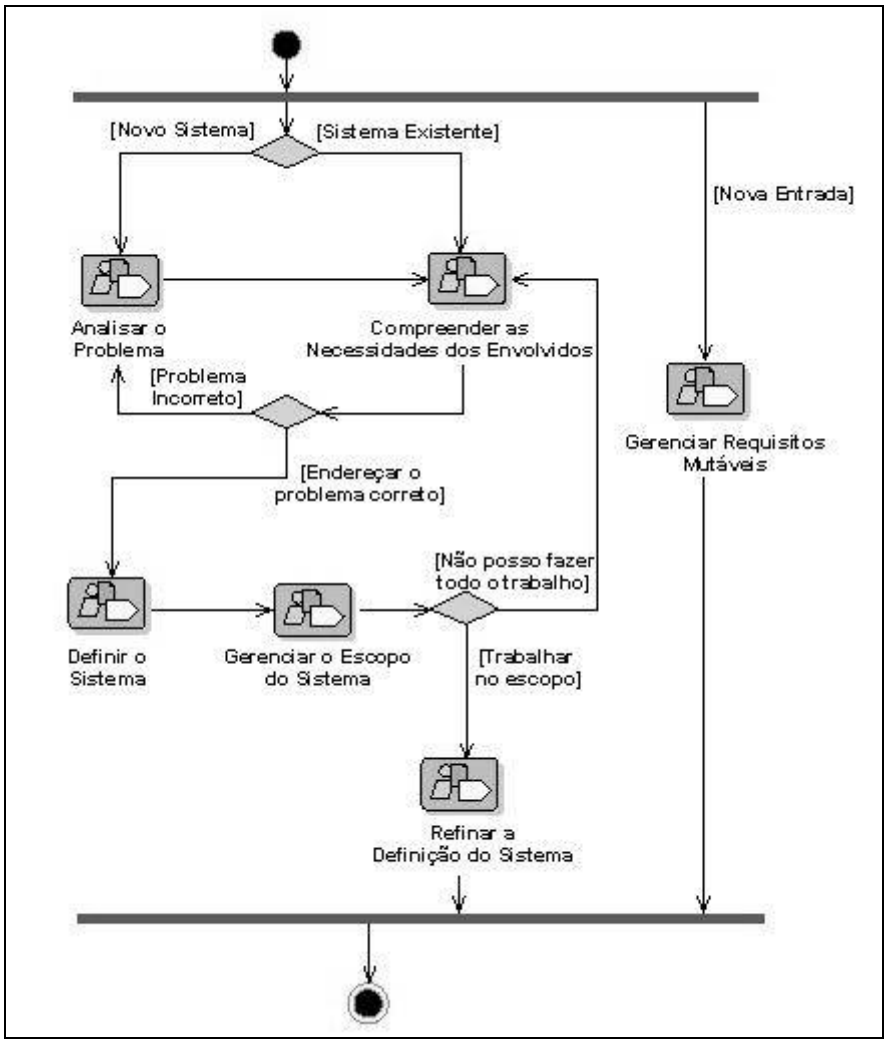


Figura 14. Fluxo de trabalho principal do RUP [19].

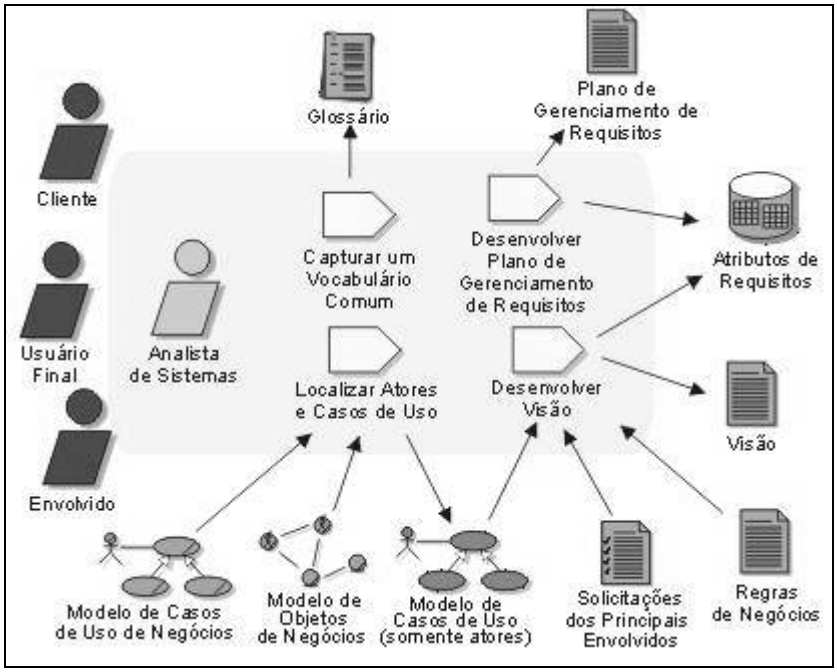


Figura 15. Fluxo de trabalho principal detalhado do RUP [19].

Com a utilização de uma metodologia de desenvolvimento de software como o RUP, é possível obter:

- Qualidade de *software*;
- Produtividade no desenvolvimento, operação e manutenção de *software*;
- Controle sobre desenvolvimento dentro de custos, prazos e níveis de qualidade desejados;
- Estimativa de prazos e custos com maior precisão.

Com base nestes recursos a adoção do RUP pode ser feita de mais de uma maneira. Um extremo seria usar o RUP à risca, ou seja, aplicar todos os métodos e processos exatamente como são propostos, porém, deve-se ter a consciência de que o RUP é um processo de *software* muito complexo. A vantagem desta abordagem é que nada deve ser alterado, pois o RUP é bem completo e detalhado. Uma desvantagem é que isto implicaria em treinamentos de pessoal, projetos piloto, etc. Propostas de projetos de adoção do RUP são descritos no próprio produto [15]. O extremo oposto seria adotar outro modelo de processo mais simples e utilizar o material do RUP como, por exemplo, os modelos de documentos, como fonte de referência complementar para assuntos não abordados em outro modelo.

2.3 Tecnologias Alternativas

Várias tecnologias podem ser usadas para o desenvolvimento de aplicações *web*. Este sessão visa mostrar duas das tecnologias mais populares de desenvolvimento *web*. Falaremos de suas principais características e compararemos com o JSP, tecnologia que será usada para a implementação do produto deste trabalho.

2.3.1 PHP

PHP (*Hipertext Preprocesor*) [20] é uma linguagem de programação do lado do servidor, gratuita e independente de plataforma. É rápida e possui uma grande livraria de funções e muita documentação.

Uma linguagem do lado do servidor é aquela que se executa no servidor *web*, antes da página ser enviada ao cliente. As páginas executam no servidor e podem realizar acessos a banco de dados, conexões em rede, e outras tarefas para criar a página final que será vista pelo cliente. O cliente somente recebe uma página com o código HTML resultante da execução da PHP. Como a página resultante contém unicamente código HTML, é compatível com todos os navegadores.

A linguagem PHP é uma linguagem de programação de domínio específico, ou seja, seu escopo se estende a um campo de atuação que é o desenvolvimento *web*, embora tenha variante como o PHP-GTK (PHP *Graphical Toolkit*) para o desenvolvimento de aplicações *desktop* multi-plataforma [20]. Algumas características pricipais do PHP são:

- Velocidade de execução e desenvolvimento;
- Estruturado e orientado a objetos[10];
- Portabilidade;
- Sintaxe similar a liguagem *C*.

PHP é uma linguagem bastante simples e altamente indicada para o desenvolvimento de aplicações também simples na *web*. Quando comparada a JSP, as linguagens possuem a mesma simplicidade de programação, porém a vantagem do JSP não é exatamente do JSP em si, mas da plataforma Java como um todo associado a esta, que fornece serviços e tecnologias avançados que PHP ainda não suporta [21].

Java possui *frameworks* e códigos prontos para praticamente tudo que precisamos trabalhar. Ferramentas como Hibernate e Struts descritas anteriormente, são realmente um ponto muito forte no desenvolvimento com Java aplicado a *web*, ajudando a separar de forma sucinta todas as partes de um sistema quando utilizamos o padrão MVC.

Assim, o desenvolvimento de sistemas *web* utilizando Java e não PHP é indicado para sistemas que necessitam de robustez e segurança.

2.3.2 ASP

ASP (*Active Server Pages*) [22] é a tecnologia desenvolvida pela Microsoft[11] para a criação de páginas dinâmicas no servidor. ASP se escreve na página *web*, utilizando a linguagem *Visual Basic Script* ou *Jscript*.

Com ASP pode-se realizar vários tipos de aplicações distintas. Esta linguagem permite-nos acesso ao banco de dados, ao sistema de arquivos do servidor e, em geral, a todos os recursos que existam no próprio servidor [22]. Atualmente, já foi apresentada a segunda versão de ASP, o ASP.NET, que compreende algumas melhoras em relação às possibilidades da linguagem e rapidez com que funciona.

Assim como o PHP, o ASP sofre com o acoplamento da camada de apresentação e negócio; o que a princípio torna fácil o seu aprendizado acaba por tornar-se um empecilho em projetos de maior porte, dificultando as tarefas de desenvolvedores e *webdesigners*. As características de

[10] Em junho de 2004 foi lançada a versão 5 do PHP, introduzindo um novo modelo de orientação a objeto.

[11] *http://www.microsoft.com*

orientação a objetos são, como no PHP, “tímidas” perto do JSP [21]. Por ser uma tecnologia proprietária, o ASP não atrai desenvolvedores adeptos do mundo *Open Source*. Quanto ao seu desempenho e segurança, repete-se aqui a já histórica e inflamada discussão "Unix" *vs* "Windows" [23].

2.4 Justificativa das tecnologias escolhidas

O JSP apresenta-se como uma solução profissional, e pode ser considerado o futuro das aplicações *web*: desacopla com facilidade a camada de apresentação da camada de negócio, permite a compilação em tempo real das aplicações (uma vez executado o *script* pela primeira vez, uma versão compilada se apresenta às requisições seguintes) e integra-se com perfeição ao “mundo” Java [23], dando suporte aos princípios de Orientação a Objetos que prezam pela qualidade de software, bem como pela facilidade de reuso no futuro.

A utilização do JSP só não é maior devido ao conjunto de conhecimentos exigidos para que se tire proveito de seus melhores atributos: o entendimento do paradigma de orientação a objetos e a grande extensão de sua linguagem Java. Vendo por este lado, o JSP é uma aposta mais consistente e com visão de longo prazo, além de não encontrar concorrentes quando se pensa em aplicações críticas em ambientes corporativos [21].

A Tabela 7 mostra um pequeno comparativo das linguagens aqui citadas para desenvolvimento de aplicações *web*.

Tabela 7. Comparação de algumas das mais populares linguagens *web*.

Aspecto	JSP	PHP	ASP
Curva de Aprendizado	Aguda	Suave	Suave
Código Fonte	Aberto	Aberto	Proprietário
Porte típico dos projetos envolvidos	Médios e Grande	Pequenos e Médios	Pequenos e Médios
Ambiente de Execução	Java	Unix	Microsoft

Concluímos aqui que não há como saber se uma solução é ideal para todas as necessidades de aplicações *web*. Na hora de decidir qual tecnologia escolher, devemos levar em consideração muito mais o ambiente que dará suporte às aplicações e o perfil dos desenvolvedores que se dispõem a implementar a aplicação, do que qualquer tentativa de formular uma "regra universal" para esta escolha.

Provavelmente uma das três alternativas discutidas aqui deve preencher os requisitos de uma aplicação *web*; todas as três são soluções largamente disponíveis, suportadas e com grande número de adeptos, o que minimiza os riscos de um erro de escolha. Nesta hora, uma análise do que mais é utilizado no mercado para o tipo de aplicação que será desenvolvida pode até influenciar, mas é o conjunto dos fatores descritos acima que acabará ditando a escolha por uma ou outra [23].

Capítulo 3

SGIP

Este capítulo tem por objetivo descrever todas as atividades envolvidas no desenvolvimento do sistema SGIP – Sistema de Gerenciamento de Inscrições e Seleção de Pós-Graduação. Começaremos com uma breve introdução ao conceito de Sistema de Informação e sua relação com este trabalho, complementando o que já foi falado anteriormente no capitulo 1, e os requisitos do sistema serão apresentados. Nas sessões seguintes, falaremos da Metodologia de desenvolvimento empregada neste trabalho e dos diagramas UML e documentos confeccionados. Por fim, a *interface* do sistema será apresentada.

3.1 Sistema de Informação

Sistema de Informação é a expressão utilizada para descrever um sistema automatizado, ou mesmo manual, que abrange pessoas, máquinas, e/ou métodos organizados para coletar, processar, transmitir e disseminar dados que representam informação para o usuário [24].

A principal vantagem proporcionada pelos Sistemas de Informação é a capacidade de processar um grande número de dados simultaneamente, tornando disponíveis estas informações a todo o momento. Porém, de pouco adianta esse potencial se os sistemas (rotinas, processos e métodos) não estiverem muito bem coordenados e analisados. Informatizar sistemas ruins traz novos problemas e nenhuma solução, além de “nublar” as possíveis causas dessas falhas [25].

Em um Sistema, várias partes trabalham juntas visando um objetivo em comum. Em um Sistema de Informação não é diferente, porém o objetivo é um fluxo mais confiável e menos burocrático das informações. É com essa finalidade que o SGIP foi desenvolvido: para prover aos professores do DSC informações de boa qualidade sobre o processo seletivo ao curso de Mestrado oferecido por este departamento a cada semestre, o que é essencial para uma boa tomada de decisão.

3.1.1 Requisitos do Sistema

Requisitos são capacidades e condições as quais o sistema deve possuir. Um desafio primário do trabalho com requisitos é achar, comunicar e registrar o que é realmente necessário no sistema, numa forma que fale claramente ao cliente, assim como ao time de desenvolvimento [26].

Os requisitos de um sistema expressam as características e restrições do produto de *software* do ponto de vista de satisfação das necessidades do usuário. Em geral, independem da tecnologia empregada na construção da solução, porém a arquitetura do sistema pode ser fortemente influenciada pelos seus requisitos como, por exemplo, um requisito de alta performance poderá influenciar na escolha dos componentes de *hardware* e suas configurações [27].

A especificação dos requisitos é uma parte importante no desenvolvimento de sistemas, pois sem ela um produto não pode ser construído ou, pelo menos, não pode ser construído corretamente. Segundo [26], um estudo revelou que cerca de 37% dos fatores relacionados a problemas nos requisitos fazem com que estes sejam a maior causa de problemas no *software*. O desafio seria então, utilizar um processo que uniria mudanças de requisitos e o conhecimento dessas mudanças como encaminhadores centrais no levantamento de requisitos.

O levantamento de requisitos do SGIP foi feito a partir de entrevistas com o professor Renato Mariz de Moraes que foi o coordenador responsável pela seleção da turma de Mestrado do primeiro semestre de 2008. A partir destas entrevistas foi identificado que o SGIP terá de ser desenvolvido para facilitar o processo de inscrição ao curso de Mestrado do DSC por parte dos candidatos e para diminuir os esforços empregados no processo de seleção destes candidatos por parte do coordenador do processo seletivo vigente, o que vem sendo feito de maneira muito custosa.

Desta forma, dois principais usuários/atores do sistema foram identificados:

- Professores / corpo administrativo do DSC - São os servidores do Departamento de Sistemas e Computação da Escola Politécnica de Pernambuco.
- Candidato - Toda e qualquer pessoa que desejar interagir com o sistema de inscrições *on-line*.

Os requisitos por parte do Candidato resumem-se ao cadastro de Informações Pessoais, Informações Acadêmicas, Dados Curriculares e Informações relacionadas ao interesse do candidato pelos projetos escolhidos no ato da inscrição. Cabe ao coordenador do processo, o cadastro de usuários (professores) que, por meio do sistema, irão adicionar projetos para que o candidato tenha acesso às informações mais atualizadas da corrente seleção. Também será função do coordenador, classificar os candidatos informando ao sistema os respectivos pesos dos dados passados pelos candidatos na hora da inscrição.

Em resumo, os principais requisitos e atividades que deverão ser implementadas no sistema podem ser visualizadas no fluxograma[12] do processo representado pela Figura 16.

Como pode-se perceber, após a inscrição do candidato, a inscrição deverá ser confirmada pelo coordenador através da documentação que deve ser entregue pelo candidato para comprovar os dados que foram passados no ato da inscrição. Após esta confirmação, o candidato já estará participando de fato do processo, cabendo ao coordenador fornecer ao sistema os pesos que serão associados às suas informações curriculares, acadêmicas e das respectivas cartas de recomendação apresentadas pelo candidato.

Ainda de acordo com o fluxograma, podemos perceber que, após a avaliação curricular e classificação das cartas de recomendação de todos os alunos de inscrição confirmada no sistema, o coordenador deve finalizar a seleção, podendo assim, gerar o primeiro relatório que indicará uma lista de alunos e suas respectivas notas obtidas na seleção. Paralelamente, os professores poderão entrevistar os candidatos cujos projetos escolhidos indicam cada professor que será responsável pelas entrevistas. Após a informação das notas obtidas nas entrevistas por todos os alunos, o relatório final poderá ser visualizado.

[12] Um fluxograma é uma representação simbólica, que descreve a sequência das fases de um processo.

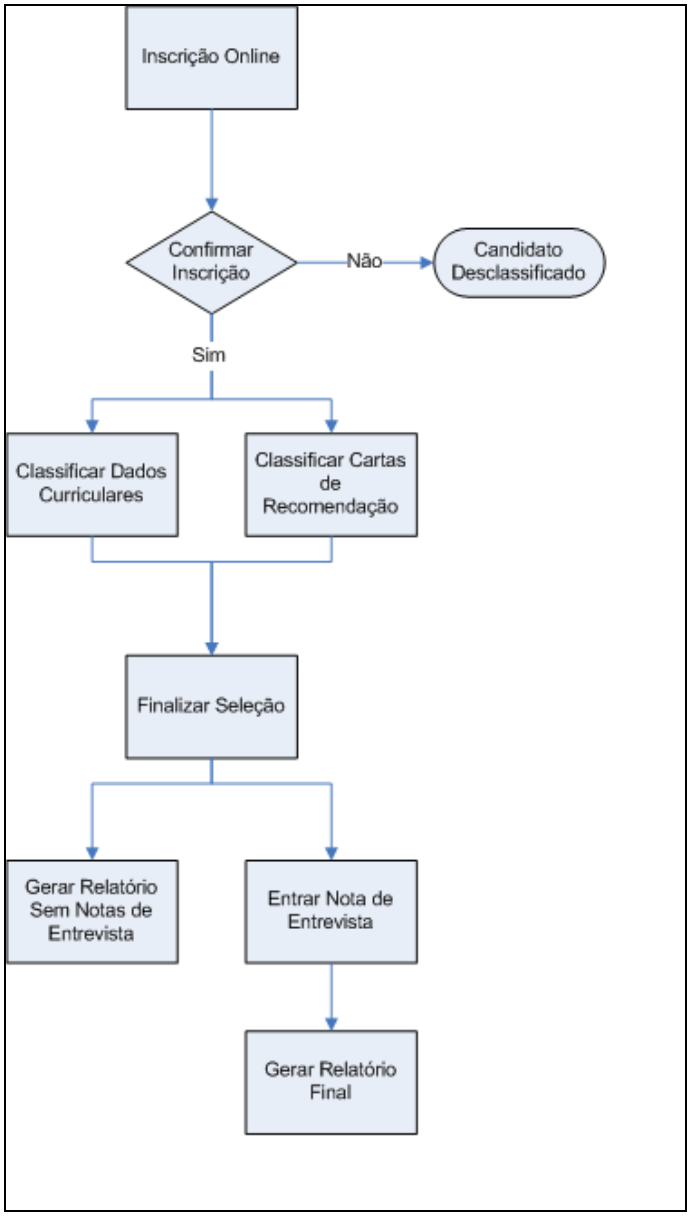


Figura 16. Fluxograma do processo de inscrição e seleção.

Na seção seguinte, serão descritos resumidamente os principais casos de uso (Requisitos Funcionais) do sistema SGIP.

3.1.2 Casos de Uso

O processo de identificação de requisitos tem uma função fundamental no correto desenvolvimento de sistemas, porém pode se tornar um processo extenso e trabalhoso [28]. Na década de 80, vários autores sugeriram diversas técnicas para que o processo de levantamento e validação de requisitos se tornasse mais rápido e eficiente.

Uma das técnicas mais populares é a utilização de Casos de Uso para descrever claramente todos os requisitos de um dado sistema. Essa técnica foi proposta por Ivar Jacobson em sua metodologia de sistemas orientados a objetos [27].

Cada caso de uso descreve um cenário de possível interação no sistema. Essa interação pode ser feita por alguém ou algo que tente utilizar o sistema ou até mesmo outro sistema. Os

casos de uso devem ser o mais claro possível para que todos (clientes e projetistas) possam entendê-los de igual modo, evitando-se utilizar termos técnicos que possam possivelmente dificultar esta compreensão da funcionalidade descrita [28].

Cada caso de uso deve descrever somente uma funcionalidade ou objetivo do sistema, representando uma unidade discreta da interação entre um usuário (humano ou máquina) e o sistema propriamente dito [29].

Nesta seção, falaremos um pouco sobre os casos de uso do SGIP. Para uma visão mais detalhada destes casos de uso, recomenda-se uma leitura mais aprofundada através do documento de Especificação de Casos de Uso no Anexo 1 deste trabalho e do Diagrama de Casos de Uso que será mostrado na seção 3.4.

1. Incluir Candidato
    a. Descrição: Possibilita o cadastro de candidatos no banco de dados para que participem do processo seletivo.
    b. Ator: Candidato.
    c. Pré-condições: nenhuma.
    d. Pós- condições: Candidato cadastrado no processo seletivo com inscrição não confirmada.
2. Incluir Usuário
    a. Descrição: Possibilita o cadastro de novos usuários no sistema.
    b. Ator: Usuário do sistema com nível de administrador.
    c. Pré-condições: Não há.
    d. Pós- condições: Novo usuário cadastrado no sistema.
3. Manter Usuário
    a. Descrição: Possibilita a manutenção (atualização ou exclusão) de usuários já cadastrados no sistema.
    b. Ator: Usuário do sistema com nível de administrador.
    c. Pré-condições: Para alteração ou exclusão de usuário, o mesmo deverá ser cadastrado previamente no sistema.
    d. Pós- condições: Usuário atualizado ou excluído do sistema.
4. Incluir Professor
    a. Descrição: Possibilita o cadastro de novos professores no sistema.
    b. Ator: Usuário do sistema com nível de administrador.
    c. Pré-condições: nenhuma.
    d. Pós- condições: Professor cadastrado no sistema.
5. Manter Professor
    a. Descrição: Possibilita a manutenção (atualização ou exclusão) de professores já cadastrados no sistema.
    b. Ator: Usuário do sistema com nível de administrador ou professor.
    c. Pré-condições: Para alteração ou exclusão de professor, o mesmo deverá ser cadastrado previamente no sistema.
    d. Pós- condições: Professor atualizado ou excluído do sistema.
6. Incluir Projeto
    a. Descrição: Possibilita o cadastro de novos projetos no sistema.
    b. Ator: Usuário do sistema com nível de professor.
    c. Pré-condições: nenhuma.
    d. Pós- condições: Novo projeto cadastrado no sistema.
7. Manter Projeto

   a. Descrição: Possibilita a manutenção (atualização ou exclusão) de projetos já cadastrados no sistema.
   b. Ator: Usuário do sistema com nível de professor.
   c. Pré-condições: Para alteração ou exclusão de projeto, o mesmo deverá ser cadastrado previamente no sistema.
   d. Pós- condições: Projeto atualizado ou excluído do sistema.
8. Confirmar Inscrição de Candidato
   a. Descrição: Possibilita que a inscrição de um candidato seja confirmada.
   b. Ator: Usuário do sistema com nível de administrador.
   c. Pré-condições: Candidato deve ter sido previamente cadastrado no sistema.
   d. Pós- condições: Candidato com inscrição confirmada.
9. Entrar Avaliação de Carta de Recomendação
   a. Descrição: Possibilita que uma carta de recomendação de um candidato seja avaliada e cadastrada no sistema.
   b. Ator: Usuário do sistema com nível de administrador.
   c. Pré-condições: Candidato deve ter tido sua inscrição confirmada.
   d. Pós- condições: Não há.
10. Entrar Avaliação Curricular
   a. Descrição: Possibilita que o administrador possa avaliar e associar pesos aos dados curriculares dos candidatos inscritos na seleção.
   b. Ator: Usuário do sistema com nível de administrador.
   c. Pré-condições: Candidato precisa já ter sua inscrição confirmada.
   d. Pós- condições: Não há.
11. Trocar Datas da Seleção
   a. Descrição: Possibilita que as datas da seleção sejam modificadas para que as inscrições cessem ou recomecem.
   b. Ator: Usuário do sistema com nível de administrador.
   c. Pré-condições: Não há.
   d. Pós- condições: Não há.
12. Gerar Relatório Final
   a. Descrição: Possibilita a geração do relatório final com as notas dos candidatos inscritos no processo seletivo.
   b. Ator: Usuário do sistema com nível de administrador.
   c. Pré-condições: Todos os candidatos devem ter sido avaliados e suas inscrições devem ter sido confirmadas.
   d. Pós- condições: Não há.
13. Entrar Nota de Entrevista
   a. Descrição: Possibilita que um professor cadastre a nota da entrevista do candidato.
   b. Ator: Usuário do sistema com nível de professor.
   c. Pré-condições: O professor deve possuir candidatos alocados a ele.
   d. Pós- condições: Não há.

3.2 Metodologia

O RUP oferece um *framework* de processos centralizado na arquitetura e é baseado em boas práticas de desenvolvimento. Ele foi escolhido neste projeto por ser maduro e amplamente difundido e utilizado, e por atender todas as necessidades de projeto desde a fase de planejamento

até a fase de implantação. Devido à necessidade de um processo preditivo, onde os custos e prazos são definidos no início do projeto, foi decidido utilizar uma instância leve [30] (processo simplificado em relação ao número de papéis, atividades e artefatos) do RUP como processo de desenvolvimento de *software* do SGIP.

O RUP estabelece uma maneira de desenvolvimento de *software* que é iterativa e guiada por casos de uso. Dessa maneira, requisitos e riscos podem ser identificados, um projeto pode ser realizado e uma implementação pode ser construída para esse projeto, sendo assim, validada e testada. Esse processo se repete com outras partes do sistema até que o sistema inteiro seja terminado. Isso é chamado de modo iterativo. Uma iteração resulta em um incremento, ou seja, é a repetição de um conjunto de atividades que gera um acréscimo de funcionalidade ao sistema.

No desenvolvimento do SGIP, duas iterações serão necessárias para a implementação dos principais módulos do sistema. São elas:

- Iteração 1: Módulo de Cadastros
  - Neste módulo, todos os casos de uso descritos na seção 3.1.2 serão desenvolvidos. Este módulo visa à implementação das principais regras de negócio[13] relativas ao gerenciamento e seleção de candidatos ao curso de Mestrado.
- Iteração 2: Módulo de Relatórios
  - O principal objetivo deste módulo é a geração de relatórios que serão utilizados como forma de documentar todo o processo de seleção.

A cada iteração, as ferramentas e *frameworks* utilizados serão os mesmos. Todo o projeto será desenvolvido utilizando JSP, tendo o auxílio do *framework* de apresentação Struts, do *framework* de mapeamento objeto/relacional Hibernate e do banco de dados relacional MySQL. Todos já foram descritos no capítulo 2.

3.2.1 Cronograma de Atividades

O projeto foi desenvolvido seguindo o cronograma apresentado na Tabela 8.

Tabela 8. Cronograma de Atividades.

Atividades	Semanas									
	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										

As atividades desenvolvidas podem ser vistas mais detalhadamente na sequência abaixo, de acordo com a numeração mostrada na tabela acima:

1. Entrevistas com os professores do DSC que participaram da comissão de seleção do mestrado em 2008 para levantamento de requisitos.

[13] Regras de negócio são todas as regras existentes num sistema de informação, que ditam seu comportamento, suas restrições e suas validações.

- Entrevistas foram necessárias para que se tivesse um conhecimento das necessidades encontradas pelos professores no processo de seleção e para que, através destas entrevistas, os requisitos podessem ser levantados fazendo com que todas as funcionalidades necessárias fossem implementadas no sistema.

2. Analise dos requisitos levantados e riscos.
    - A análise dos requisitos foi feita para que se podesse perceber o grau de dificuldade que poderia ser encontrado, assim como, para gerar dúvidas e possíveis novos requisitos podessem ser levantados.
3. Escrita dos artefatos e diagramas UML.
    - Em paralelo a análise de requisitos, os documentos de Especificação de Casos de Uso foram escritos para que os mesmos podessem ser revisados visto que contam como documentação necessária na entrega final deste projeto.
4. Implementação do sistema e banco de dados.
    - Nesta fase, foi dada total atenção e dedicação ao desenvolvimento do sistema e geração dos *scripts* utilizados no banco de dados.
5. Implantação da primeira versão do sistema para uso e testes.
    - Antes da entrega final do projeto, pretendendia-se testá-lo junto aos docentes do departamento para que as alterações e validações necessárias fossem feitas, seja no protótipo, seja na regra de negócio. O sistema ficou disponível *on-line* o que realmente tornou possível a validação de algumas telas do sistema.
6. Escrita da Monografia.
    - A todo o momento do projeto, a escrita do texto monográfico foi feita com o propósito de descrever as tecnologias e processo que foram utilizados no decorrer do desenvolvimento do SGIP.

As semanas contadas no cronograma equivalem às semanas compreendidas entre os dias 7 de março de 2008 ao dia 20 de maio de 2008.

3.3 Diagramas

O RUP utiliza a UML para especificar, modelar e documentar artefatos. A UML é um padrão definido pelo OMG (*Object Management Group*) [31] e tem se tornado o padrão empresarial para a modelagem de projetos orientados a objetos.

Um diagrama UML é uma apresentação gráfica de um conjunto de elementos (classes, *interfaces*, componentes, etc.) que são usados para visualizar o sistema sob diferentes perspectivas. A UML define um número de diagramas que permite dirigir o foco para aspectos diferentes do sistema de maneira independente. Se bem usados, os diagramas facilitam a compreensão do sistema que está sendo desenvolvido [32].

Os diagramas utilizados pela UML são compostos de nove tipos: diagramas de casos de uso, de classes, de objetos, de estados, de seqüência, de colaboração, de atividades, de componentes e o de execução.

Tendo todo sistema características estáticas (estrutural) e dinâmicas (comportamental), a UML suporta modelos estáticos e dinâmicos, assim como também funcionais [31]. A modelagem estática é suportada pelo diagrama de classes e o de objetos, que consiste nas classes e seus relacionamentos. Os modelos dinâmicos são suportados pelos diagramas de estado, seqüência, colaboração e atividade. A modelagem funcional do sistema é suportada pelos diagramas de

componente e execução. Neste trabalho, abordaremos diagramas de casos de uso e de classes (modelagem estática) e diagramas de seqüência (modelagem dinâmica) [14].

3.3.1 Diagrama de Casos de Uso

O objetivo do diagrama de caso de uso é representar e definir os requisitos funcionais de um sistema [26]. Ele é descrito em termos de atores externos e casos de uso. Os atores representam uma entidade externa ao sistema, como um usuário ou algum outro sistema que deseje interagir com o sistema modelado [28].

Um ator pode ser conectado a um ou mais casos de uso através de associações e, tanto atores como casos de uso, podem possuir relacionamentos de generalização que definem um comportamento de herança.

Os casos de usos podem se relacionar de três formas:

- Através da operação *include*: Quando um caso de uso A inclui (*include*) outro caso de uso B, implica que ao executar o caso de uso A executa-se também o caso de uso B, ou seja, durante a execução de A ele "pula", em algum momento, para executar B.
- Através da operação *extends*: Quando um caso de uso A tem um relacionamento do tipo *extends* com outro caso de uso B, implica que ao executar o caso de uso A não necessariamente o caso de uso B será executado, ou seja, indica que o caso de uso A poderá ser acrescentado para descrever o comportamento de B.
- Através de uma *generalização* ou *especialização*: São relações do tipo "é um". Isto significa dizer que o caso de uso A *é um* caso de uso B (B é uma generalização de A, ou A é uma especialização de B). Neste caso, será de melhor entendimento dizer que é um relacionamento entre um caso de uso genérico para um mais específico, que herda todas as características de seu "pai".

Atores também podem herdar as funcionalidades de outros atores. Podemos ver este tipo de comportamento, como outros descritos, na Figura 17, na qual mostra o diagrama completo de casos de uso do sistema SGIP. Vejamos que um ator do tipo Usuário - Administrador é, também, visto como um Usuário - Comum possuindo todas as funcionalidades deste, no caso, a de Efetuar Login que também será herdada pelo Usuário – Professor.

[14] Com exceção do diagrama de casos de uso, nesta seção serão apenas exibidas "partes" dos diagramas que foram confeccionados para que possa haver um melhor entendimento das notações presentes em cada tipo de diagrama descrito. Os diagramas completos podem ser vistos nos Anexos deste trabalho.

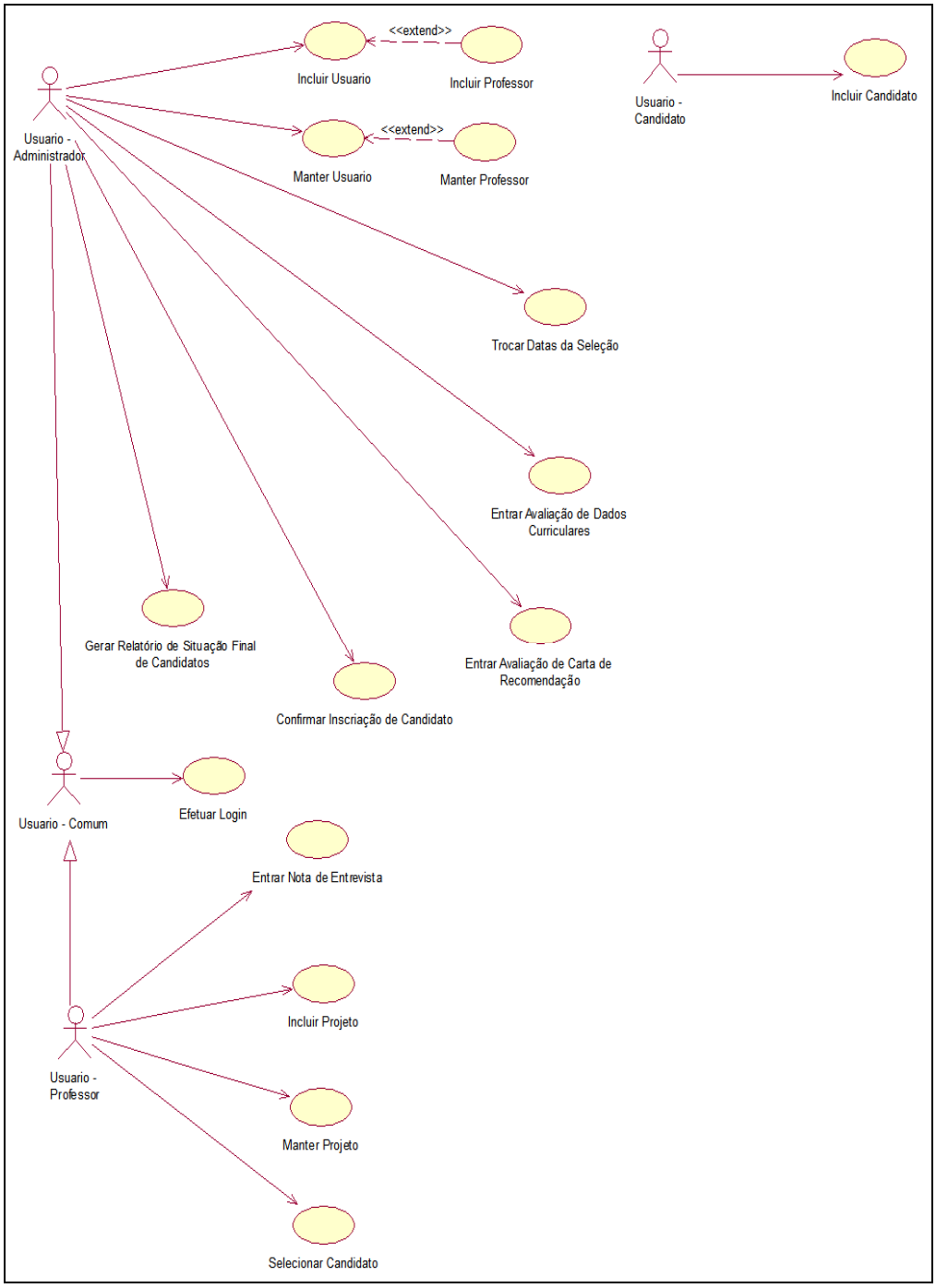


Figura 17. Diagrama de Casos de Uso do SGIP.

3.3.2 Diagramas de Classes

Um diagrama de classes é uma representação da estrutura e relações das classes que servem de modelo para os objetos [31]. É uma modelagem muito útil para o sistema, pois define todas as classes que o sistema necessita e serve como base para a construção dos diagramas de comunicação, de sequência e de estados. É um dos diagramas mais importantes da documentação, pois nele podemos encontrar as informações sobre métodos, atributos e como as classes serão

integradas. Um diagrama de classes bem modelado é fundamental para auxiliar o desenvolvedor a construir o sistema [26].

No modelo de classes trabalhamos com um único elemento que é representado por um retângulo dividido em três partes: a primeira divisão é utilizada para o nome da classe, a segunda divisão é utilizada para as informações de atributos e a última divisão é utilizada para identificar os métodos. As classes definem os tipos de objetos que existem dentro do sistema. Elas podem conter atributos que são geralmente membros de dados primitivos de objetos e operações definidoras de métodos que podem ser aplicados sobre os objetos de cada classe [31].

Na UML, os atributos são mostrados com pelo menos seu nome, e podem também mostrar seu tipo, valor inicial e outras propriedades. Eles também podem ser exibidos com sua visibilidade: "+" indica se um atributo é público, "#" indica se um atributo é protegido e "-" indica se um atributo é privado.

As operações (métodos) também são exibidas com pelo menos seu nome e podem mostrar seus parâmetros e valores de retorno e o tipo desses valores. As operações podem, como os atributos, mostrar sua visibilidade (as mesmas notações são utilizadas, neste caso).

Outro conceito igualmente importante é o de Associações. Uma associação é uma relação que permite especificar que objetos de uma dada classe se relacionam com objetos de outra classe (Ex.: um Candidato pode possuir várias graduações). Cada associação possui duas extremidades que são diretamente ligadas a objetos e cada uma dessas extremidades da associação tem uma multiplicidade, que nada mais é do que uma indicação de quantos objetos pode participar de um dado relacionamento [26].

A Figura 18 mostra um exemplo de diagrama de classes, onde estas notações descritas anteriormente podem ser observadas.

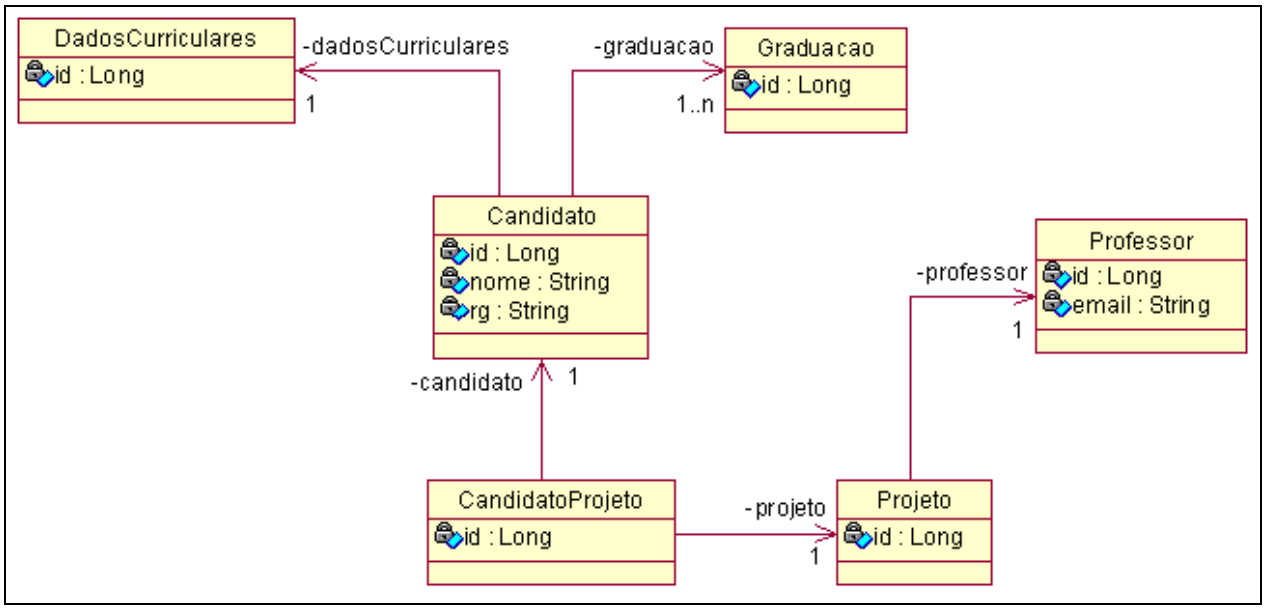


Figura 18. Exemplo de Diagrama de Classes.

O diagrama de classes completo do SGIP encontra-se disponível para leitura no documento de Análise de Casos de Uso no Anexo 2.

3.3.3 Diagramas de Seqüência

Um diagrama de seqüência ilustra os eventos de saída e de entrada relacionados ao sistema e tem o objetivo de mostrar como as mensagens entre os objetos são trocadas no decorrer do

tempo para a realização de uma operação, identificando o comportamento do sistema em relação aos casos de uso. Ele demonstra este comportamento através de uma "caixa preta" onde evidenciamos o que o sistema faz e não como ele faz [26].

O diagrama de seqüência é uma figura que mostra, para um cenário particular de um caso de uso, os eventos que os atores externos geram, a ordem destes eventos e os eventos ocasionados por estes no interior do sistema. Em um diagrama de seqüência, os seguintes elementos podem ser encontrados [26]:

- Linhas verticais (representando o tempo de vida de um objeto - *lifeline*): estas linhas verticais são preenchidas por barras verticais que indicam exatamente quando um objeto passou a existir. Quando um objeto não mais existe, coloca-se um "X" na parte inferior da barra.
- Linhas horizontais ou diagonais (representando mensagens trocadas entre objetos): estas linhas são acompanhadas de um "rótulo" que contém o nome da mensagem e, opcionalmente, os parâmetros da mesma. Também podem existir mensagens enviadas para o mesmo objeto, representando uma iteração.
- Condição: uma condição é representada por uma mensagem cujo rótulo é envolvido por colchetes.
- Mensagens de retorno (representadas por linhas horizontais tracejadas): este tipo de mensagem não é freqüentemente representada nos diagramas. Muitas vezes sua utilização leva a um grande número de setas no diagrama, atrapalhando o entendimento do mesmo. Este tipo de mensagem só deve ser mostrado quando for fundamental para a clareza do diagrama.

A Figura 19 mostra um exemplo de um diagrama de seqüência do caso de uso Incluir Projeto. Neste diagrama, o ator Professor solicita ao sistema que seja cadastrado um projeto. Para isso, ele deve informar o nome do projeto, selecionar um arquivo para *upload* e acionar o botão de confirmação presente na tela. Após estas ações, as setas horizontais continuam mostrando o fluxo que acontece até o cadastramento do objeto no banco de dados, o que acontece quando o objeto é passado à classe *RepositorioProjeto.java*.

Como é mostrado no diagrama, a requisição é feita através da classe *IncluirProjetoAction.java* na camada de apresentação que envia esta mesma requisição para a classe *Fachada.java* que ira tramitar a ocorrência através do *ControladorProjeto.java*, onde as regras de negócio foram implementadas, para a classe *RepositorioProjeto.java* que abrirá uma sessão no banco através do Hibernate solicitando a inserção do objeto Projeto na sua tabela respectiva já mapeada anteriormente[15].

Este diagrama pode mostrar resumidamente como estão organizadas as tramitações de requisições no SGIP até que as mesmas cheguem ao banco MySQL e possam ser persistidas.

Os diagramas de seqüência de alguns casos de uso documentados neste sistema poderão ser encontrados no Anexo 1.

[15] O mapeamento de todas as classes do sistema que serão persistidas no banco de dados é feito da mesma forma como foi mostradoem exemplo no capítulo 2 na subseção 2.1.3 sobre o Hibernate.

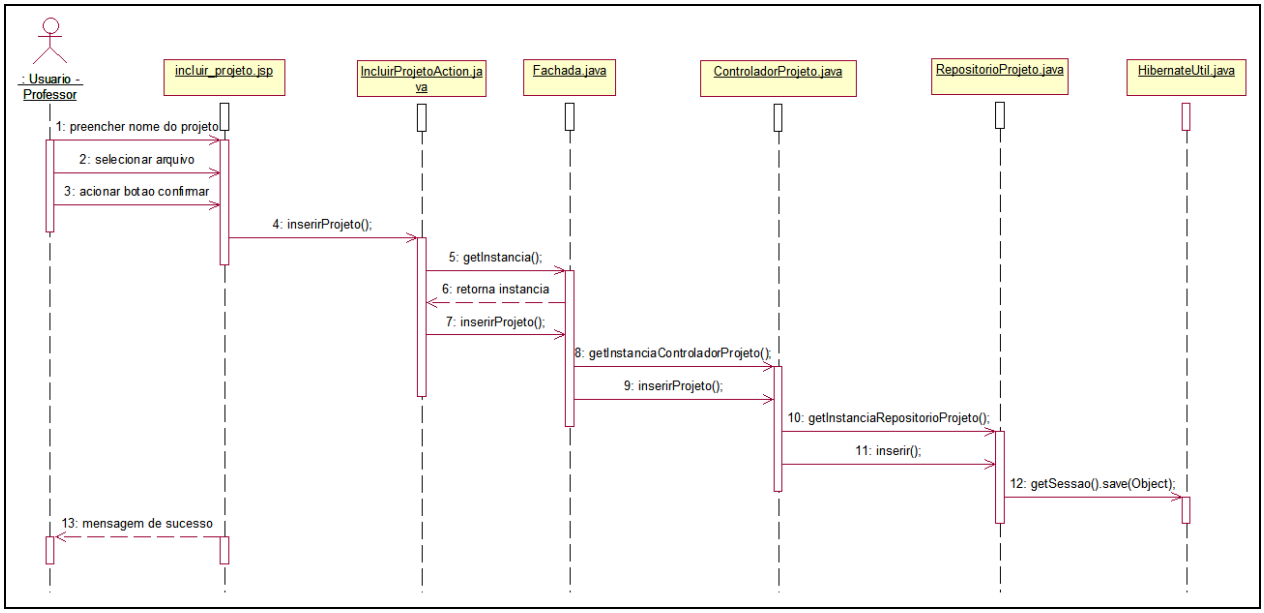


Figura 19. Exemplo de Diagrama de Seqüência.

3.4 Arquitetura do SGIP

O padrão de arquitetura utilizado neste projeto foi o MVC. Este padrão possibilita que separemos toda a lógica da camada de negócio das camadas de modelo e de apresentação.

Na arquitetura MVC, a camada de modelo representa a informação (dados) da aplicação e as regras de negócio utilizadas para manipular os dados. A camada de apresentação corresponde aos elementos da *interface* com o usuário como texto fornecidos e todos os elementos que serão manipulados pelo usuário do sistema. Por fim, a camada de controle gerencia os detalhes que envolvem a comunicação das classes do sistema e as ações do usuário.

A Figura 20 mostra, resumidamente, como está organizada a arquitetura do SGIP.

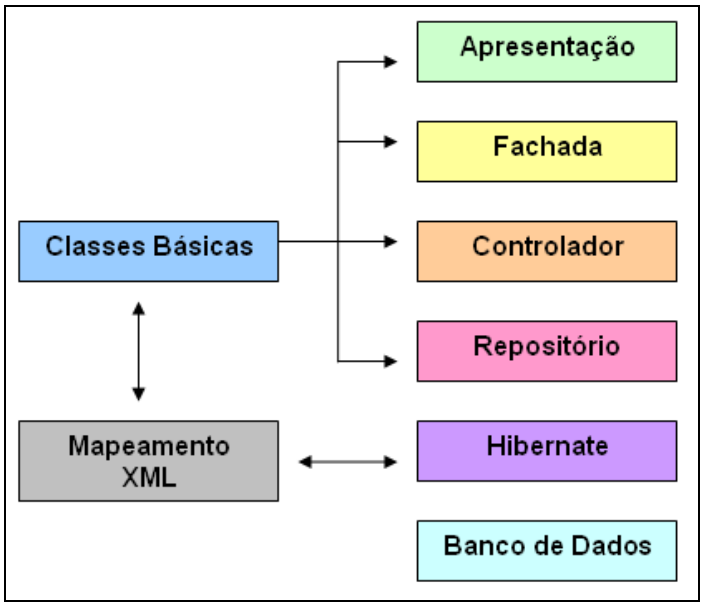


Figura 20. Arquitetura do SGIP.

Como podemos observar, cada classe básica do sistema terá seu arquivo de mapeamento do Hibernate, assim como as classes que se comunicarão com a camada de apresentação (classes *Action* e *ActionForm*) que implementam as funcionalidades obtidas através do uso do *framework* Struts já explicado em capítulo anterior.

Ainda através da figura, podemos visualizar que existirá uma classe Fachada a qual servirá de "ponte" entre as classes de apresentação e as classes de negócio representadas pelas classes Controlador. É importante perceber que cada classe básica terá sua respectiva classe Controlador que implementará todas as regras de negócio relativas à esta classe básica.

Por fim, uma classe Repositório foi criada para cada classe básica que deverá abrir uma sessão com o Hibernate e se comunicará diretamente com o banco de dados proporcionando a persistência dos objetos. Nesta classe Repositório também devem ser implementados todos os métodos que se comunicam com o Hibernate como códigos HQL[16].

3.5 Interface do SGIP

A elaboração da interface gráfica é de extrema importância no projeto de um sistema, pois ela é responsável por uma parte fundamental: a parte visível para o usuário, através da qual ele se comunica para realizar as tarefas desejadas [33]. Ela também é responsável em fazer com que o usuário consiga realizar suas tarefas de maneira fácil, ágil e satisfatória e, quando bem projetada, pode tornar-se uma fonte facilitadora ou, caso contrário, pode transformar-se em um ponto decisivo na rejeição de um sistema, na limitação de uma ferramenta ou na execução de tarefas.

Nesta seção serão exibidas e comentadas apenas as telas do SGIP às quais o caso de uso Incluir Candidato[17] faz referência. Uma breve explicação sobre o fluxo de ações será dada sobre cada *interface* mostrada.

3.5.1 Caso de uso: Incluir Candidato

O candidato interessado em inscrever-se *on-line* para o Mestrado do DSC, deve acessar o menu em "Mestrado Acadêmico" e em seguida escolher a opção "Inscrições On-line!", o que resultará no aparecimento da tela mostrada na Figura 21, onde o candidato irá clicar no botão "Prosseguir" após concordar que todas as informações que ele dispuser neste formulário deverá ser comprovada com documentação entregue até o final da seleção vigente para a Secretária da Pós-Graduação no departamento.

---

[16] Todas as regras de como as classes devem ser desenvolvidas, assim como nomenclaturas padrões utilizadas, podem ser vistas mais afundo através do documento de Arquitetura no Anexo 3 desta monografia.

[17] O fluxo de ações feitas pelo usuário neste caso de uso, como em todos os outros do sistema, podem ser encontrados detalhadamente no documento de Especificação de Casos de Uso no Anexo 2.

Figura 21. Termo de aceitação para a inscrição *on-line*.

Após prosseguir para a próxima tela, mostrada na Figura 22, o candidato deve informar todos os campos pedidos no formulário para cadastro das informações básicas. Neste formulário, todas as informações são obrigatórias.

O usuário poderá escolher a opção de “Avançar” fazendo com que, internamente, este formulário seja validado e, só assim, o candidato será encaminhado à próxima página mostrada na Figura 23. Este formulário contém tudo que é necessário para que o candidato informe toda sua formação acadêmica, tanto em nível de graduação quanto em nível de pós-graduação, caso exista alguma.

Desta vez, o candidato poderá escolher entre a opção de “Voltar”, caso deseje modificar as informações deixadas na página anterior, ou “Avançar” caso deseje ir para a próxima página.

Caso escolha a opção “Avançar”, o candidato será levado à página mostrada como exemplo na Figura 24, onde este deverá cadastrar os seus dados curriculares. Esses dados envolvem iniciações científicas concluídas, experiências profissionais na área, produções científicas publicadas, cursos de curta duração concluídos, nível da língua inglesa, nota obtida no PosComp, etc. É importante observar que nem tudo pode ser visto nesta na Figura 24 que serve só para exemplificar este protótipo do sistema, pois se trata de um formulário muito grande, mas que pode ser visto na íntegra através do acesso ao sistema.

No quarto e último formulário a ser apresentado ao candidato, mostrado na Figura 25, ele deve informar até três projetos aos quais gostaria de se candidatar e informar, ainda, se solicitará bolsa durante o Mestrado e se manterá algum vínculo empregatício durante o mesmo. Ambas as informações são de extrema importância ao processo seletivo. Ao clicar no botão “Concluir” toda a informação será cadastrada no banco de dados, porém só será processada (levada em

consideração no processo de seleção) caso haja uma confirmação de todos os dados submetidos nos formulários, o que também deverá ser feito através do sistema[18].

Também é importante deixar claro que um processo de validação de informações é feito em todas as etapas da inscrição. O usuário será auxiliado através de mensagens impressas na própria tela sobre os erros cometidos e, em alguns casos, exemplos serão mostrados de como alguns campos específicos podem ser preenchidos.

Figura 22. Formulário para cadastramento das informações básicas do candidato.

[18] UC_008 Confirmar Inscrição de Candidato.

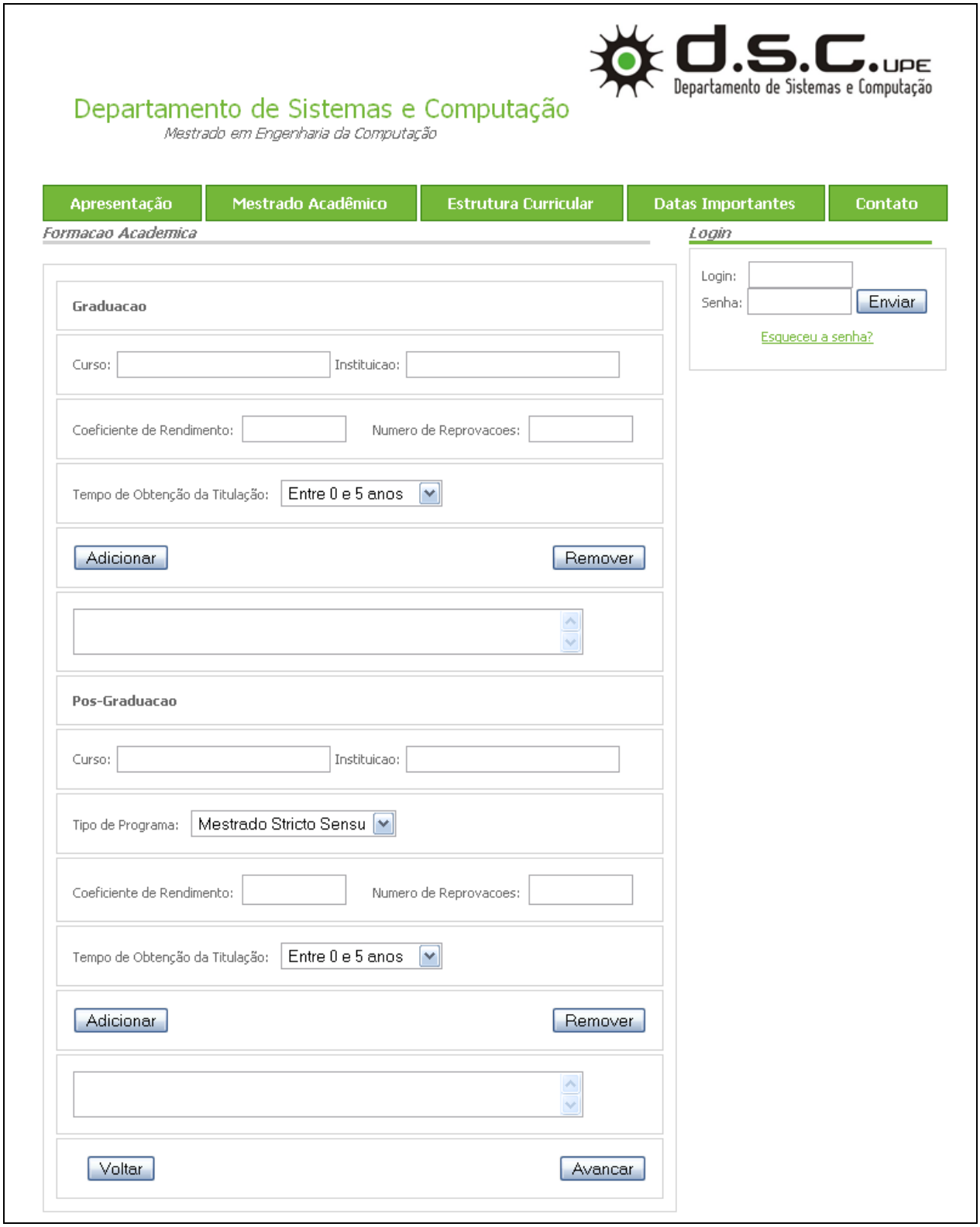


Figura 23. Formulário para cadastramento da formação acadêmica do candidato.

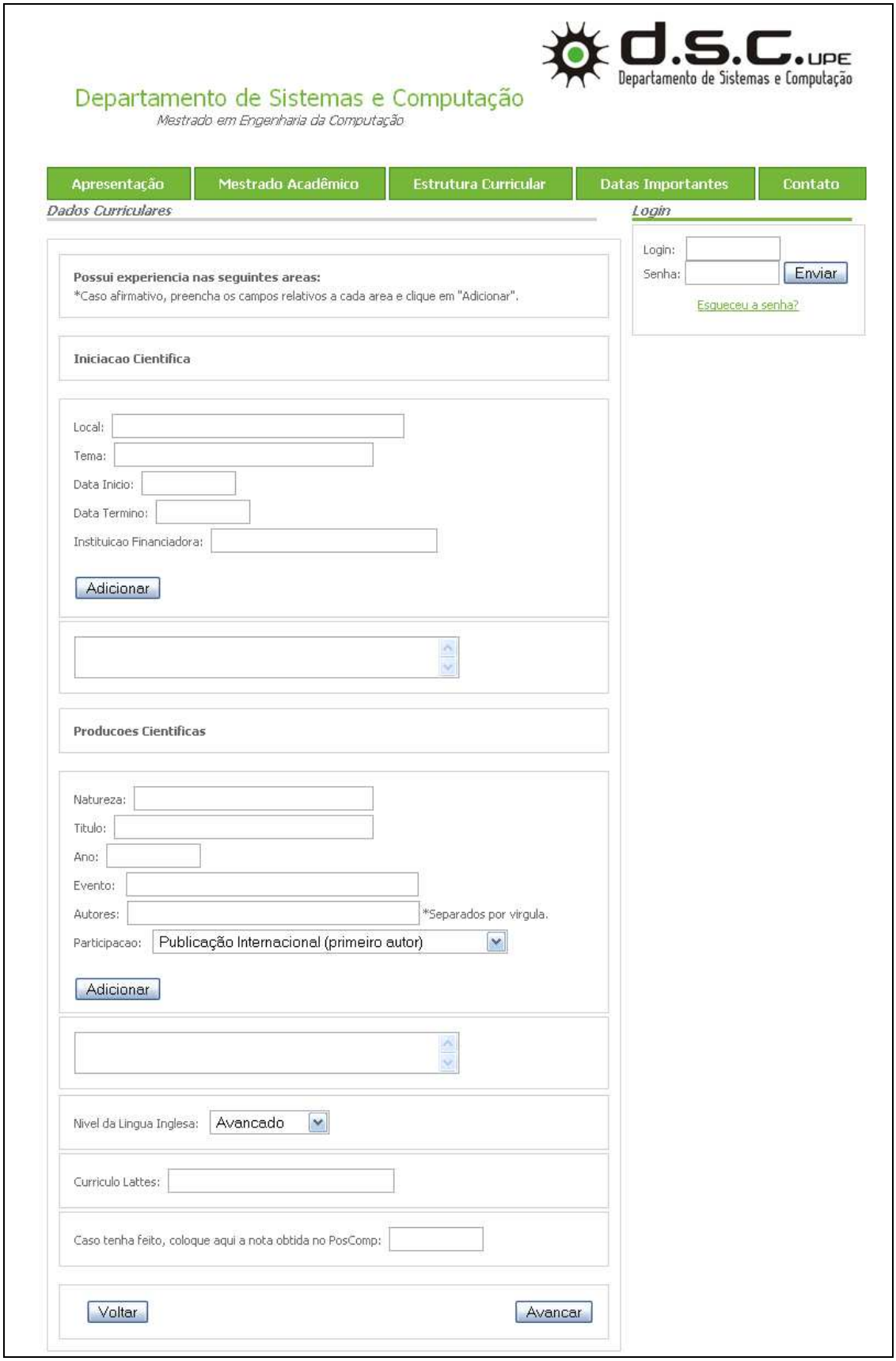


Figura 24. Formulário para cadastramento dos dados curriculares do candidato.

Figura 25. Formulário para cadastramento das preferências do candidato em relação à inscrição.

Capítulo 4

Conclusões e Trabalhos Futuros

O objetivo deste trabalho foi criar um sistema que facilitasse o processo pelo qual, hoje, a inscrição ao Mestrado oferecido pelo DSC é feito, assim como o processo de seleção dos candidatos. Este capítulo visa listar as contribuições e considerações finais deste trabalho e indicar alguns trabalhos futuros que poderão dar continuidade a este o que poderá ser de grande utilidade à coordenação de Pós-Graduação do departamento.

4.1 Contribuições

O desenvolvimento do SGIP, como trabalho de conclusão de curso, possibilitara que o DSC tenha mais controle sob as inscrições dos candidatos ao Mestrado e organização nas atividades que envolvem a seleção destes candidatos. Também é muito importante para mostrar na prática como os conceitos de Engenharia de *Software* vistos durante o curso podem ser utilizados para o desenvolvimento de sistemas *web*, como também para aprendizado de tecnologias e padrões de projeto que não são estudados a fundo, mas que são de extrema importância no mercado de trabalho hoje.

O sistema desenvolvido neste projeto aparece como um facilitador do processo de inscrição, seleção e conclusão do fluxo de atividades que envolvem a seleção de novos alunos para o Mestrado oferecido pelo departamento. Além de facilitar parte do trabalho do coordenador da seleção, o sistema também pretende facilitar a vida do candidato, colocando a sua disposição um mecanismo de inscrição *on-line*.

4.2 Considerações

É importante perceber que o sistema desenvolvido neste trabalho possui um escopo ainda reduzido. Nem toda burocracia do processo de inscrição poderá ser eliminada, uma vez que os alunos que utilizarão o sistema para inscrever-se no Mestrado terão que comprovar as informações submetidas nos formulários, através de documentos que deverão ser levados à Secretaria de Pós-Graduação do departamento nas datas previstas no edital de inscrição. Estes documentos deverão ser sujeitos à análise e comparados com as informações cadastradas no banco de dados e, somente assim, o candidato terá sua inscrição confirmada.

Na parte de seleção de candidatos, o trabalho do coordenador será reduzido em comparação ao processo utilizado hoje, porém ele ainda deverá informar vários dados ao sistema que devem participar do processo de seleção, tal como a avaliação das cartas de recomendação, visto que pode ser algo ainda bastante subjetivo.

4.3 Trabalhos Futuros

O SGIP é um sistema voltado para a área de Pós-Graduação do DSC. Sendo uma área relativamente nova no departamento, seriam interessantes que outras atividades da fase "pós-seleção" de alunos ao Mestrado fossem implementadas e acopladas a este sistema visto que, por ser baseado na *web*, torna-o bastante acessível.

Sistemas que envolvam controle de atividades de alunos, ementa de disciplinas, horários, ou qualquer outra particularidade da área do Mestrado Acadêmico da instituição, podem ser desenvolvidos e atrelados a este sistema de forma bastante simples e não custosa. A própria arquitetura orientada a objetos na qual foi baseado este trabalho visa facilitar este reuso e o desenvolvimento de novos módulos, assim como a geração de novas versões.

Ainda fará parte do escopo deste trabalho toda a implantação do sistema, mudança de requisitos, manutenção e reimplementação de funcionalidades de acordo com o aparecimento de necessidades de mudanças que possam vir a ocorrer nas futuras seleções. É importante ressaltar, também, que o código deste sistema e toda a documentação desenvolvida no decorrer de seu desenvolvimento serão entregues ao departamento.

Bibliografia

[1] STAIR, R. M. Princípios de Sistemas de Informação. 2ª ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora S.A., 1998, 451p.

[2] FERREIRA, P. A. Desenvolvimento de Sistema de Informação Web para o controle interno de protocolos da Escola Politécnica de Pernambuco. Recife: DSC-UPE, 2006. Monografia de Trabalho de Conclusão de Curso.

[3] O' BRIEN, A. J. , Sistema de Informação e as Decisões Gerenciais da era da Internet. ed. Saraiva 2003.

[4] FIELDS, D.K.; KOLB, M.A. – Desenvolvendo na Web com *JavaServer Pages* – Editora Ciência Moderna, 2000.

[5] SCHMITT, R. A. Sistemas de Informação Baseados na WEB. Universidade Federal do Rio de Janeiro, 2005.

[6] BERGSTEN, H. *JavaServer Pages*, 3ª ed. O'Reilly 2003.

[7] BAUER, C.; KING, G. *Struts in Action*. 1ª ed. Manning 2003.

[8] BAUER, C.; KING, G. *Hibernate in Action*. 1ª ed. Manning 2005.

[9] KROLL, P. e KRUCHTEN P. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. ed. Addison Wesley 2003.

[10] DEITEL, H. M. e DEITEL P.J. Java Como Programar. 3ª ed. Porto Alegre: BookMan, 2001.

[11] FOWLER, M. e RICE, D. *Patterns of Enterprise Application Architecture*. 1ª ed. Addison Wesley 2003.

[12] IVERSON, W. *Hibernate: A J2EE Developer's Guide*. 1ª ed. Addison Wesley 2004.

[13] DUBOIS, P. *MySQL Cookbook: Solutions and Examples for MySQL Database Developers*. 1ª ed. O'Reilly 2003.

[14] *MySQL: Case Studies*. Disponível em <http://www.mysql.com/why-mysql/case-studies/>. Acesso em 3 de abril de 2008.

[15] BERGSTROM S. e RABERG L., H. *Adopting the Rational Unified Process: Success with the RUP*. 1ª ed. Addison Wesley 2004.

[16] FOWLER, M. *UML distilled: A brief guide to the standard object modeling language*. 1ª ed. Addison Wesley 2004.

[17] JUNIOR, S. L. Um estudo comparativo entre RUP e XP. Monografia (Especialização em Engenharia de Software). - Escola Politécnica da Universidade de São Paulo 2004.

[18] *Rational Unified Process*. Disponível em <http://www.wthreex.com/rup/>. Acesso em 3 de abril de 2008.

[19] Processo Unificado e RUP. Disponível em <http://paginas.terra.com.br/negocios/processos2002/processo_unificado_e_rup.htm>. Acesso em 3 de abril de 2008.

[20] SICA, C. Programação Segura Utilizando PHP: Fale a Linguagem da Internet. 1.ed. Rio de Janeiro - RJ: Ciência Moderna, 2007.

[21] FIELDS, D. K. e KOLB, M. A. Desenvolvendo na Web com *JavaServer Pages*. RJ: Editora Ciência Moderna Ltda, 2000.

[22] ONION, F. *Essential ASP .NET with examples in C#*. 1ª ed. Addison Wesley 2003.

[23] QUATI, M. R. O desafio da escolha: soluções para aplicações WEB. Cepromat - Centro de Processamento de Dados do Estado de Mato Grosso 2003.

[24] LINDSAY, J. *Information Systems – Fundamentals and Issues. Kingston University, School of Information Systems*, 2000.

[25] A importância dos Sistemas de Informação. Disponível em <http://www.bonde.com.br/colunistas/colunistasd.php?id_artigo=1646>. Acesso em 8 de abril de 2008.

[26] LARMAN, C. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*. 2ª ed. Prentice Hall 2002.

[27] JACOBSON, I. *Object-oriented Software Engineering: A Use Case Driven Approach*. 1ª ed. Addison Wesley 1992.

[28] KULAK, D. e GUINEY, E. *Use Cases: Requirements in Context*. 2ª ed. Addison Wesley 2003.

[29] WIEGERS, K. E. *Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle*. 2ª ed. Redmond: Microsoft Press.

[30] MARQUES, H. M., RAMOS T. R. e SILVA G. L. I. Adaptação de um Processo de Desenvolvimento para Fábricas de Software Distribuídas. Centro de Informática, Universidade Federal de Pernambuco - *8th International Database Engineering and Applications Symposium* 2004.

[31] JÉZÉQUEL, Jean-Mark, HUSSMAN, H. e COOK, S. *UML 2002 - The Unified Modeling Language: Model Engineering, Concepts and Tools*. 1ª ed. Springer 2002.

[32] YANG, H. *Software Evolution with UML and XML*. 1ª ed. Idea Group Inc (IGI) 2005.

[33] ALVES, L. R. G., NOVA, C. C. Educação e tecnologia: trilhando caminhos. Salvador: Editora da UNEB, 2003, v.1. p.263.

Anexo 1

Documento de Especificação dos Casos de Uso

Este documento tem como propósito demonstrar a especificação de todos os casos de uso do SGIP.

Detalhamento de Requisitos

Especificação de Casos de Uso

SGIP

Histórico

Data	Autor	Versão	Descrição
22/04/2008	Izaura Dias	V1.0	Versão Inicial

Convenções utilizadas

Controle de alteração no documento:

- Textos marcados em [informar a cor definida para o projeto] indicam alterações realizadas para o projeto[nome do projeto]/[nome da iteração];

Identificação de atributos:

- Os campos marcados com # indicam campos *read-only*;
- Os campos marcados com * indicam campos obrigatórios;

UC_001 Incluir Candidato

Descrição: Este caso de uso tem por finalidade permitir o registro de um novo candidato no processo seletivo do Mestrado.

Ator: Qualquer usuário que deseje interagir com o sistema.

Pré-condição:
- Não há.

Pós-condição:
- Candidato registrado com inscrição no *status* 'Não Confirmada'.

Fluxo de eventos principal:

- O usuário selecionará a função de cadastro através do menu "Mestrado Acadêmico / Inscrições Online".
- O sistema exibirá uma página com as informações a serem preenchidas (SB Informações Básicas).
- O usuário poderá utilizar uma das seguintes opções:
    - Confirmar a entrada das informações básicas através do botão "Avançar" (SB Formação Acadêmica).

Sub-Fluxos (SB):

SB Informações Básicas
- O sistema solicitará as seguintes informações:
    - Nome*
    - Email *
    - Telefone para contato *
    - Identidade *
    - Órgão de Expedição *
    - Data de Expedição *
    - CPF *
    - Data de Nascimento *
    - Naturalidade *
    - Nacionalidade *
    - Nome do Pai *
    - Nome da Mãe *
    - Endereço *
    - Bairro *
    - Estado *
    - Cidade *
    - CEP *

- O usuário selecionará o botão “Avançar” para concluir o registro das informações básicas (SB Formação Acadêmica).
- O sistema realizará as validações:
  - *FS1 – Campos obrigatórios não preenchidos;*
  - *FS2 - CPF valido;*
  - *FS3 – Data valida;*

SB Formação Acadêmica

- O sistema solicitará as seguintes informações sobre as graduações e/ou pós-graduações concluídas do candidato:
  - Curso
  - Instituição
  - Coeficiente de Rendimento
  - Numero de Reprovações
  - Tempo de obtenção da titulação
  - Tipo de programa

- O candidato poderá cadastrar uma graduação clicando na opção “Adicionar”.
- O sistema realizará as validações:
  - *FS1 – Campos obrigatórios não preenchidos;*
- O candidato poderá remover uma graduação, selecionando-a e clicando na opção “Remover”.
- O usuário poderá utilizar uma das seguintes opções:
  - Confirmar a entrada das informações através do botão “Avançar” (SB Dados Curriculares).
  - Clicar no botão “Voltar” para voltar a tela anterior (SB Informações Básicas).

SB Dados Curriculares

- O sistema solicitará as seguintes informações sobre as iniciações cientificas concluídas do candidato:
  - Local
  - Tema
  - Data de Inicio
  - Data de Termino
  - Instituição Financiadora
- O candidato poderá cadastrar uma iniciação cientifica clicando na opção “Adicionar”.
- O sistema realizará as validações:
  - *FS1 – Campos obrigatórios não preenchidos;*
- O candidato poderá remover uma iniciação cientifica, selecionando-a e clicando na opção “Remover”.
- O sistema solicitará as seguintes informações sobre as experiências como docente concluídas do candidato:
  - Local
  - Tema
  - Data de Inicio
  - Data de Termino
- O candidato poderá cadastrar sua experiência como docente clicando na opção “Adicionar”.
- O sistema realizará as validações:
  - *FS1 – Campos obrigatórios não preenchidos;*
- O candidato poderá remover uma experiência como docente, selecionando-a e clicando na opção “Remover”.

- O sistema solicitará as seguintes informações sobre as experiências profissionais do candidato:
  - Local
  - Cargo
  - Data de Inicio
  - Data de Termino
- O candidato poderá cadastrar uma experiência profissional clicando na opção “Adicionar”.
- O sistema realizará as validações:
  - *FS1 – Campos obrigatórios não preenchidos;*
- O candidato poderá remover uma experiência profissional, selecionando-a e clicando na opção “Remover”.
- O sistema solicitará as seguintes informações sobre as produções cientificas do candidato:
  - Natureza
  - Titulo
  - Ano
  - Evento de publicação
  - Autores separados por vírgula
  - Participação do candidato

- O candidato poderá cadastrar uma produção cientifica clicando na opção “Adicionar”.
- O sistema realizará as validações:
  - *FS1 – Campos obrigatórios não preenchidos;*
- O candidato poderá remover uma produção cientifica, selecionando-a e clicando na opção “Remover”.
- O sistema solicitará as seguintes informações sobre as produções tecnológicas do candidato:
  - Natureza
  - Titulo
  - Ano
  - Tipo do registro
- O candidato poderá cadastrar uma produção tecnológica clicando na opção “Adicionar”.
- O sistema realizará as validações:
  - *FS1 – Campos obrigatórios não preenchidos;*
- O candidato poderá remover uma produção tecnológica, selecionando-a e clicando na opção “Remover”.
- O sistema solicitará as seguintes informações sobre os cursos de curta duração concluídos do candidato:
  - Instituição
  - Tema
  - Carga horária

- O candidato poderá cadastrar um curso de curta duração clicando na opção “Adicionar”.
- O sistema realizará as validações:
  - *FS1 – Campos obrigatórios não preenchidos;*
- O candidato poderá remover um curso de curta duração, selecionando-o e clicando na opção “Remover”.

- O usuário poderá utilizar uma das seguintes opções:
  - Confirmar a entrada das informações através do botão “Avançar” (SB Sobre Inscrição).
  - Clicar no botão “Voltar” para voltar a tela anterior (SB Formação Acadêmica).

SB Sobre Inscrição

- O sistema exibira uma lista com todos os projetos cadastrados no sistema na situação "Ativo".
- O sistema solicitará as seguintes informações sobre as preferências do candidato:
  - Três projetos na ordem de preferência do candidato.
  - Se o candidato solicitara bolsa durante o Mestrado.
  - Se o candidato manterá vinculo empregatício durante o Mestrado.

- O candidato poderá cadastrar uma opção de projeto, selecionando um projeto e clicando na opção "Adicionar".
- *FS4 – Quantidade máxima de projetos informado;*
- O candidato poderá remover uma opção de projeto, selecionando-o e clicando na opção "Remover".
- O usuário poderá utilizar uma das seguintes opções:
  - Confirmar a entrada das informações através do botão "Avançar" (SB Sucesso).
  - Clicar no botão "Voltar" para voltar a tela anterior (SB Dados Curriculares).

SB Sucesso

- O sistema exibirá em uma nova tela com uma mensagem de sucesso para o candidato.

Fluxos Secundários (FS):

FS1 - Campo obrigatório não preenchido

- Caso o usuário não preencha algum dos campos obrigatórios, o sistema exibirá uma mensagem única informando que alguns campos obrigatórios não foram preenchidos
  .

FS2 – CPF valido

- Condição: O usuário informou um número de CPF invalido.
- Mensagem: "Número de CPF invalido."

FS3 – Data valida

- Condição: O usuário informou uma data invalida ao sistema.
- Mensagem: "Data invalida!".

FS4 – Quantidade máxima de projetos informada

- Condição: O usuário deseja adicionar mais um projeto na lista dos projetos selecionados onde constam já 3 projetos.
- Mensagem: "Você só poderá optar por 3 (três) projetos apenas!".

Requisitos Não Funcionais (RNF):

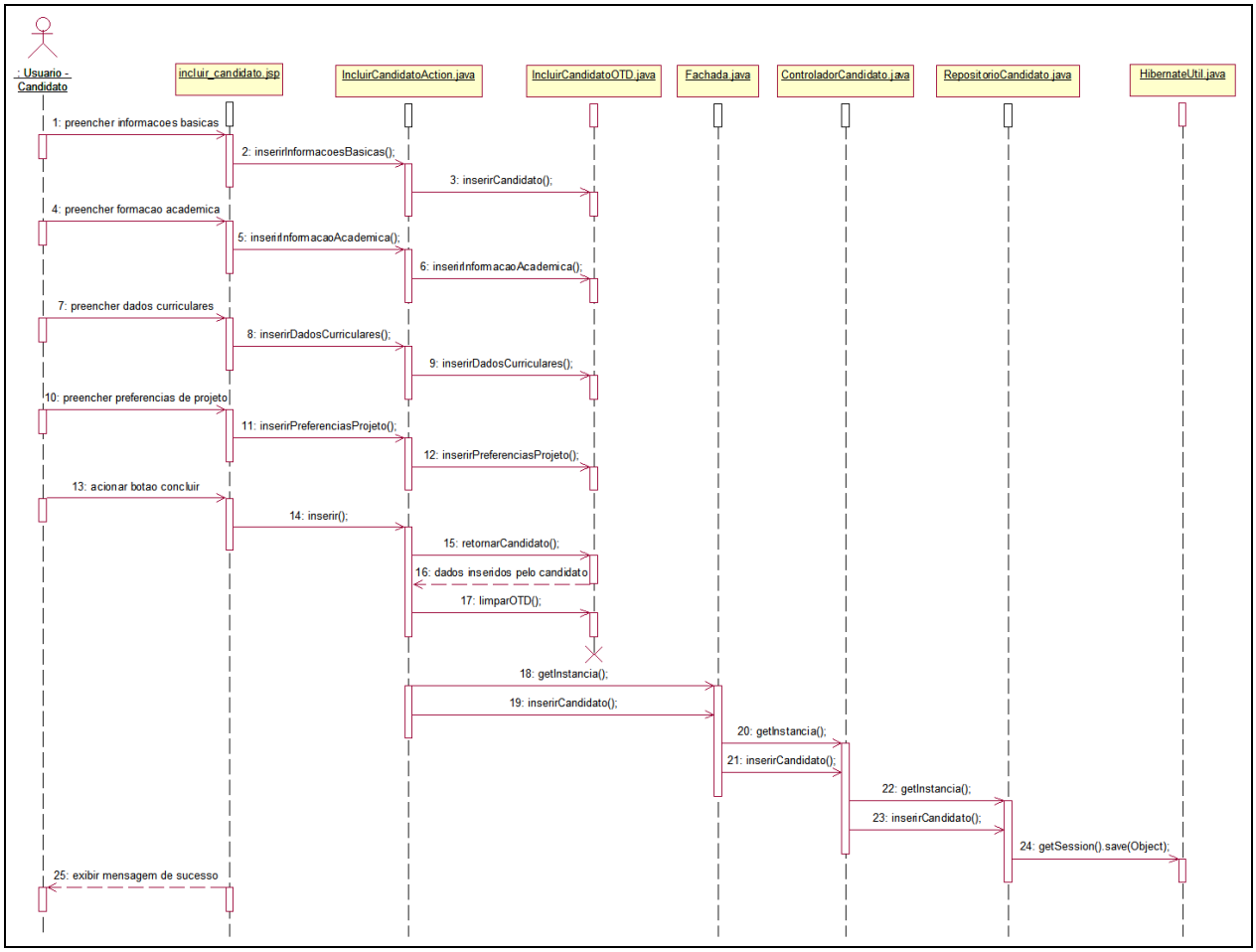
- *Não há*

Anexos e/ou Apêndices:

- **Fórmulas:** *Não há*

- **Lógica de Implementação:** *Não há*

Diagrama de seqüência:



UC_002 Incluir Usuário

Descrição: Este caso de uso tem por finalidade permitir o registro de um novo usuário no sistema.

Ator: Usuário com nível de administrador.

Pré-condição:
- Não há.

Pós-condição:
- Novo usuário cadastrado no sistema.

Fluxo de eventos principal:

- O usuário devidamente logado como administrador selecionará a função de cadastro de usuários através do menu “Cadastrar Usuarios”.
- O sistema exibirá uma página com as informações a serem preenchidas (SB Dados de Entrada).
- O usuário poderá utilizar uma das seguintes opções:
    - Confirmar a entrada das informações através do botão “Cadastrar” (SB Cadastrar).

Sub-Fluxos (SB):

SB Dados de Entrada

- O sistema solicitará as seguintes informações:
  - Login*
  - Senha *
  - Nivel do Usuario *

- O usuário selecionará o botão “Cadastrar” para concluir o registro das informações do novo usuario (SB Cadastrar).
- O sistema realizará as validações:
  - *FS1 – Campos obrigatórios não preenchidos;*

SB Cadastrar

- O sistema incluirá no banco de dados mais um usuário do sistema com o nível selecionado anteriormente.

Fluxos Secundários (FS):

FS1 - Campo obrigatório não preenchido

- Caso o usuário não preencha algum dos campos obrigatórios, o sistema exibirá uma mensagem única informando que alguns campos obrigatórios não foram preenchidos

Requisitos Não Funcionais (RNF):

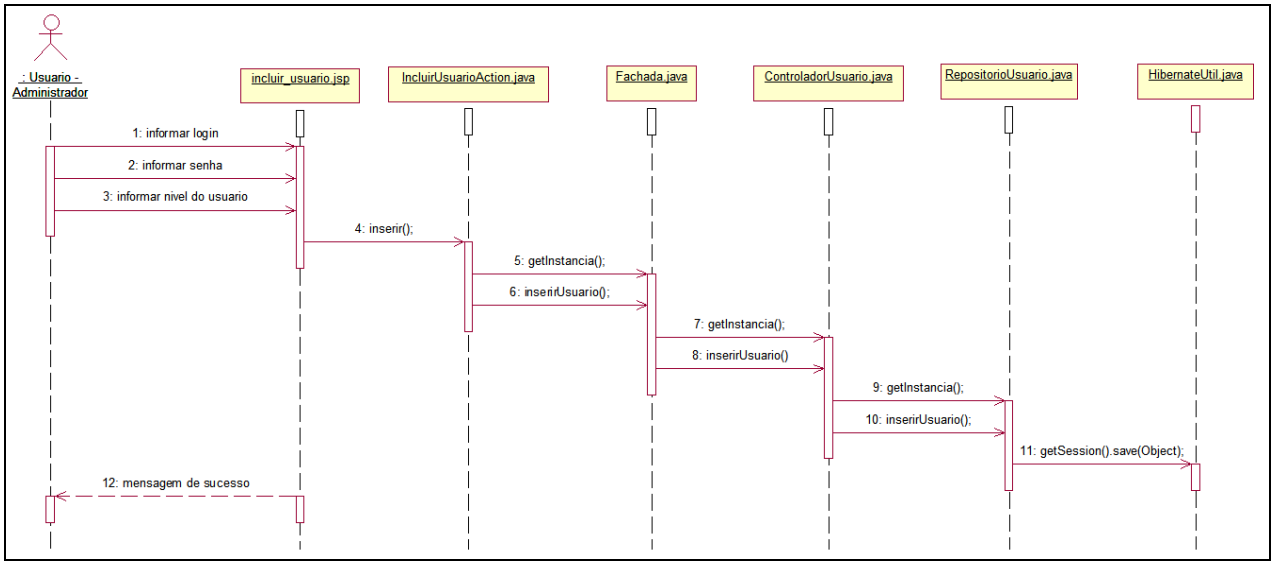
- *Não há*

Anexos e/ou Apêndices:

- **Fórmulas:** *Não há*

- **Lógica de Implementação:** *Não há*

Diagrama de seqüência:



UC_003 Manter Usuário

Descrição: Este caso de uso tem por finalidade permitir a manutenção de um usuário do sistema.

Ator: Usuário com nível de administrador.

Pré-condição:
- Não há.

Pós-condição:
- Não há.

Fluxo de eventos principal:

- O usuário devidamente logado como administrador selecionará a função de manutenção de usuários através do menu “Pesquisar Usuários”.
- O sistema exibirá uma página com as informações a serem preenchidas (SB Dados de Entrada).
- O usuário poderá utilizar uma das seguintes opções:
    - Confirmar a entrada das informações através do botão “Pesquisar” (SB Pesquisar).

Sub-Fluxos (SB):

SB Dados de Entrada
- O sistema solicitará as seguintes informações de filtro de pesquisa:
    - Login*
    - Nivel do Usuario *

- O usuário selecionará o botão “Pesquisar” (SB Pesquisar).

SB Pesquisar
- O sistema exibira uma lista com nome de login seguindo as exigências dos filtros da tela anterior.
- O usuário poderá utilizar uma das seguintes opções:
- Selecionar um registro e clicar no botão “Atualizar” (SB Atualizar).
- Selecionar um registro e clicar no botão “Excluir” (SB Excluir).
- *FS1 – Nenhum registro selecionado;*

SB Atualizar
***UC_ 002** Incluir Usuario. SB Dados de Entrada*

SB Excluir
- O sistema excluira o usuário do sistema.

Fluxos Secundários (FS):

FS1 – Nenhum registro selecionado
- Caso o usuário clique em algum botão e não tenha selecionado um registro, o sistema exibira uma mensagem dizendo que o usuário deve selecionar um registro.

Requisitos Não Funcionais (RNF):

- *Não há*

Anexos e/ou Apêndices:

- **Fórmulas:** *Não há*
- **Lógica de Implementação:** *Não há*

UC_004 Incluir Professor

Descrição: Este caso de uso tem por finalidade permitir o registro de um novo professor no sistema.

Ator: Usuário com nível de administrador.

Pré-condição:

- Não há.

Pós-condição:

- Novo professor cadastrado no sistema.

Fluxo de eventos principal:

- O usuário devidamente logado como administrador selecionará a função de cadastro de professores através do menu “Cadastrar Professor”.
- O sistema exibirá uma página com as informações a serem preenchidas (SB Dados de Entrada).
- O usuário poderá utilizar uma das seguintes opções:
  - Confirmar a entrada das informações através do botão “Cadastrar” (SB Cadastrar).

Sub-Fluxos (SB):

SB Dados de Entrada

- O sistema solicitará as seguintes informações:
  - Nome*
  - Email *
  - Senha *

- O usuário selecionará o botão “Cadastrar” para concluir o registro das informações do novo professor (SB Cadastrar).
- O sistema realizará as validações:
  - *FS1 – Campos obrigatórios não preenchidos;*

SB Cadastrar

- O sistema incluirá no banco de dados mais um professor do sistema com o login igual ao email informado.

Fluxos Secundários (FS):

FS1 - Campo obrigatório não preenchido

- Caso o usuário não preencha algum dos campos obrigatórios, o sistema exibirá uma mensagem única informando que alguns campos obrigatórios não foram preenchidos

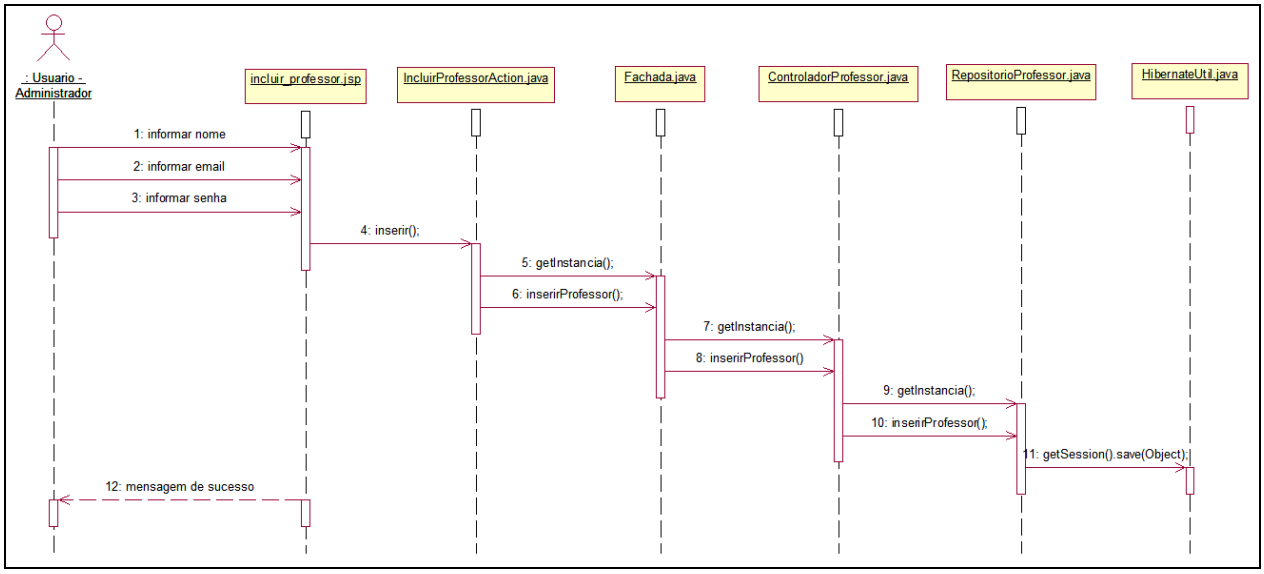
Requisitos Não Funcionais (RNF):

- *Não há*

Anexos e/ou Apêndices:

- **Fórmulas:** *Não há*
- **Lógica de Implementação:** *Não há*

Diagrama de seqüência:



UC_005 Manter Professor

Descrição: Este caso de uso tem por finalidade permitir a manutenção de um professor do sistema.

Ator: Usuário com nível de administrador.

Pré-condição:

- Não há.

Pós-condição:

- Não há.

Fluxo de eventos principal:

- O usuário devidamente logado como administrador selecionará a função de manutenção de professores através do menu “Pesquisar Professor”.

- O sistema exibirá uma página com as informações a serem preenchidas (SB Dados de Entrada).
- O usuário poderá utilizar uma das seguintes opções:
    - Confirmar a entrada das informações através do botão “Pesquisar” (SB Pesquisar).

Sub-Fluxos (SB):

SB Dados de Entrada
- O sistema solicitará as seguintes informações de filtro de pesquisa:
    - Nome*
- O usuário selecionará o botão “Pesquisar” (SB Pesquisar).

SB Pesquisar
- O sistema exibira uma lista com nome dos professores seguindo as exigências do filtro da tela anterior.
- O usuário poderá utilizar uma das seguintes opções:
- Selecionar um registro e clicar no botão “Atualizar” (SB Atualizar).
- Selecionar um registro e clicar no botão “Excluir” (SB Excluir).
- *FS1 – Nenhum registro selecionado;*

SB Atualizar
***UC_ 004** Incluir Professor. SB Dados de Entrada*

SB Excluir
- O sistema excluira o professor do sistema.

Fluxos Secundários (FS):

FS1 – Nenhum registro selecionado
- Caso o usuário clique em algum botão e não tenha selecionado um registro, o sistema exibira uma mensagem dizendo que o usuário deve selecionar um registro.

Requisitos Não Funcionais (RNF):
- *Não há*

Anexos e/ou Apêndices:
- **Fórmulas:** *Não há*
- **Lógica de Implementação:** *Não há*

UC_006 Incluir Projeto

Descrição: Este caso de uso tem por finalidade permitir o registro de um novo projeto no sistema.

Ator: Usuário com nível de professor.

Pré-condição:

- Não há.

Pós-condição:

- Novo projeto cadastrado no sistema.

Fluxo de eventos principal:

- O usuário devidamente logado como professor selecionará a função de cadastro de projetos através do menu “Cadastrar Projeto”.
- O sistema exibirá uma página com as informações a serem preenchidas (SB Dados de Entrada).
- O usuário poderá utilizar uma das seguintes opções:
    - Confirmar a entrada das informações através do botão “Cadastrar” (SB Cadastrar).

Sub-Fluxos (SB):

SB Dados de Entrada

- O sistema solicitará as seguintes informações:
    - Descrição*
    - Arquivo*
- O usuário selecionará o botão “Cadastrar” para concluir o registro das informações do novo projeto (SB Cadastrar).
- O sistema realizará as validações:
    - *FS1 – Campos obrigatórios não preenchidos;*

SB Cadastrar

- O sistema incluirá no banco de dados mais um projeto do sistema com a situação em “Ativo”.

Fluxos Secundários (FS):

FS1 - Campo obrigatório não preenchido

- Caso o usuário não preencha algum dos campos obrigatórios, o sistema exibirá uma mensagem única informando que alguns campos obrigatórios não foram preenchidos

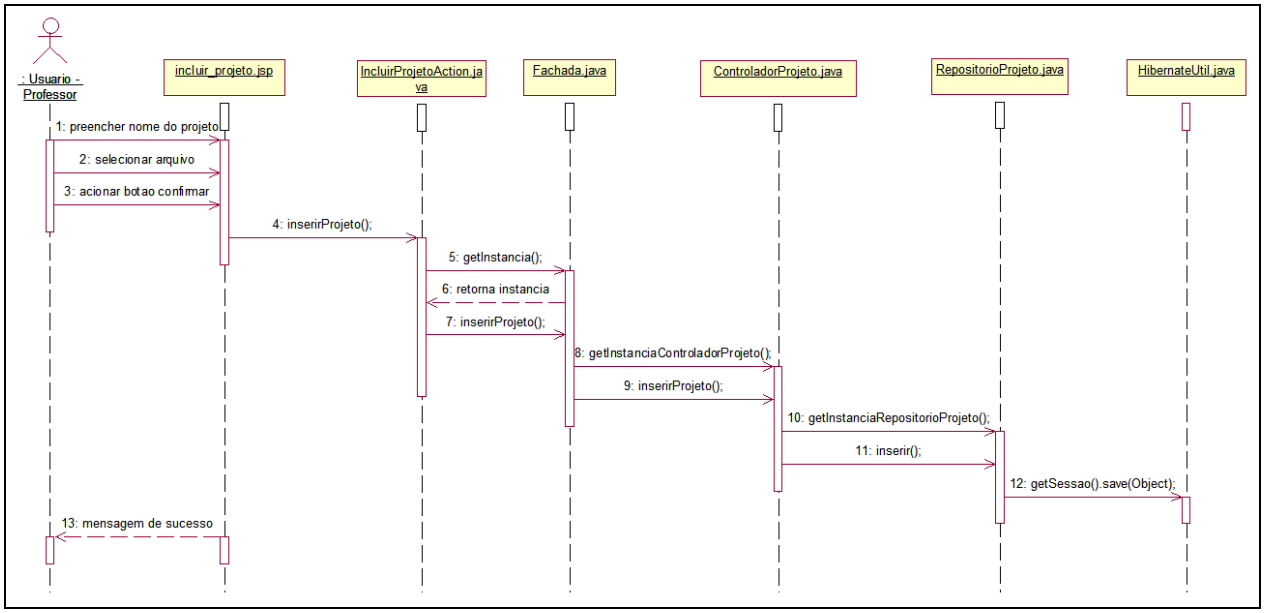
Requisitos Não Funcionais (RNF):

- *Não há*

Anexos e/ou Apêndices:

- **Fórmulas:** *Não há*
- **Lógica de Implementação:** *Não há*

Diagrama de seqüência:



UC_007 Manter Projeto

Descrição: Este caso de uso tem por finalidade permitir a manutenção de um projeto no sistema.

Ator: Usuário com nível de professor.

Pré-condição:
- Não há.

Pós-condição:
- Não há.

Fluxo de eventos principal:

- O usuário devidamente logado como professor selecionará a função de manutenção de projetos através do menu “Atualizar Projeto”.
- O sistema exibirá uma lista com os projetos cadastrados do professor logado.
- O usuário poderá utilizar uma das seguintes opções:
  - Selecionar um registro e selecionar o botão “Atualizar” (SB Atualizar).
  - *FS1 – Nenhum registro selecionado;*

Sub-Fluxos (SB):

SB Atualizar
- O sistema solicitará as seguintes informações:
  - Descrição*
  - Arquivo*
  - Situação*
- O usuário informara os campos e clicara no botão “Atualizar”.
- *FS2 – Campos obrigatórios não preenchidos;*

Fluxos Secundários (FS):

FS1 – Nenhum registro selecionado
- Caso o usuário clique em algum botão e não tenha selecionado um registro, o sistema exibira uma mensagem dizendo que o usuário deve selecionar um registro.

FS2 - Campo obrigatório não preenchido
- Caso o usuário não preencha algum dos campos obrigatórios, o sistema exibirá uma mensagem única informando que alguns campos obrigatórios não foram preenchidos.

Requisitos Não Funcionais (RNF):
- *Não há*

Anexos e/ou Apêndices:
- **Fórmulas:** *Não há*
- **Lógica de Implementação:** *Não há*

UC_008 Confirmar Inscrição de Candidato

Descrição: Este caso de uso tem por finalidade permitir a confirmação de um candidato no processo seletivo.

Ator: Usuário com nível de administrador.

Pré-condição:
- Candidato cadastrado no sistema.

Pós-condição:
- Candidato com inscrição confirmada.

Fluxo de eventos principal:

- O usuário devidamente logado como administrador selecionará a função de confirmar inscrição de candidatos através do menu “Confirmar Inscrição de Candidato”.
- O sistema exibirá uma página com uma lista de candidatos inscritos.
- O usuário poderá utilizar uma das seguintes opções:
  - Selecionar um registro e clicar no botão “Confirmar Inscrição” (SB Confirmar Inscrição).
  - *FS1 – Nenhum registro selecionado;*

Sub-Fluxos (SB):

SB Confirmar Inscrição
- O sistema exibira as seguintes informações do candidato:
  - Nome

- Email
- Data de Inscrição

- O usuário selecionará o botão “Confirmar” para concluir a confirmação da inscrição do candidato.
- O sistema mudara o registro do Candidato no banco para a situação “Confirmado”.

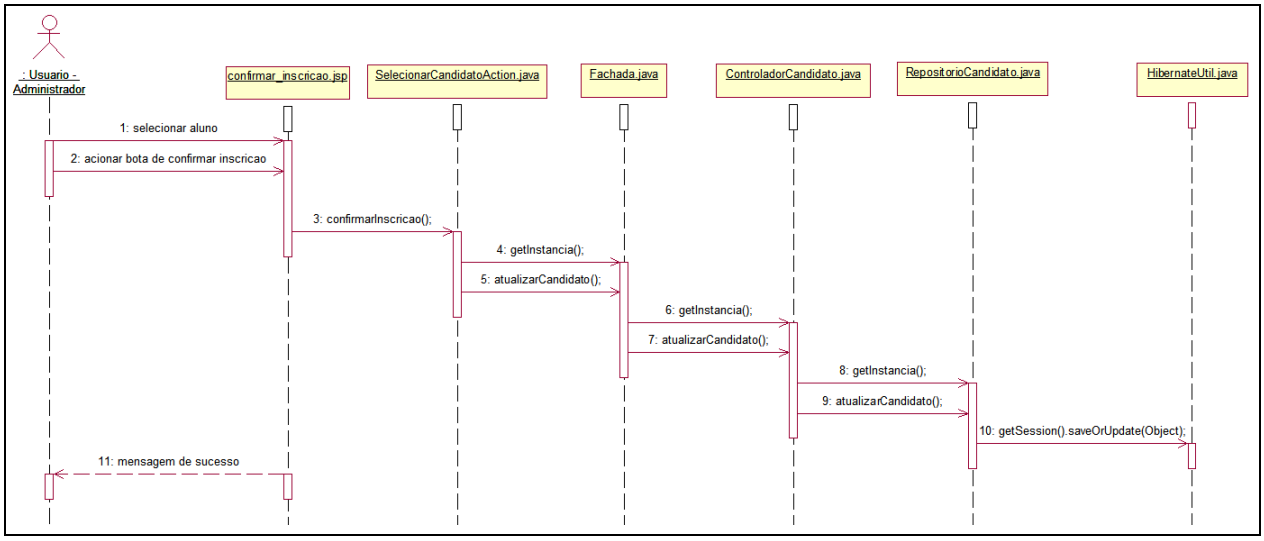
Fluxos Secundários (FS):

FS1 – Nenhum registro selecionado
- Caso o usuário clique em algum botão e não tenha selecionado um registro, o sistema exibira uma mensagem dizendo que o usuário deve selecionar um registro.

Requisitos Não Funcionais (RNF):
- *Não há*

Anexos e/ou Apêndices:
- **Fórmulas:** *Não há*
- **Lógica de Implementação:** *Não há*

Diagrama de seqüência:



UC_009 Entrar Avaliação de Carta de Recomendação

Descrição: Este caso de uso tem por finalidade permitir o registro de uma nova carta de recomendação de um candidato no sistema.

Ator: Usuário com nível de administrador.

Pré-condição:
- Candidato com inscrição confirmada no sistema.

Pós-condição:

- Carta de recomendação do candidato registrada no sistema.

Fluxo de eventos principal:

- O usuário selecionará a função de cadastro de cartas de recomendação através do menu “Entrar Avaliação de Cartas”.
- O sistema exibirá uma página com uma lista de candidatos com a inscrição já confirmada anteriormente.
- O usuário poderá utilizar uma das seguintes opções:
    - Selecionar um candidato e clicar no botão “Classificar Carta” (SB Avaliar Carta).
    - *FS1 – Nenhum registro selecionado;*

Sub-Fluxos (SB):

SB Avaliar Carta

- O sistema solicitará as seguintes informações:
    - Sobre quem recomenda*
    - Instituição onde atua *
    - Relação com o candidato *
    - Professor em disciplina *
    - Capacidade intelectual *
    - Motivação *
    - Trabalho Individual *
    - Facilidade de expressão na escrita *
    - Facilidade de expressão oral *
    - Assiduidade *
    - Iniciativa *
    - Relacionamento *
    - Avaliação geral *
    - Informações adicionais
    - Grau de recomendação *

- O usuário selecionará o botão “Concluir” para concluir o registro das informações sobre a carta de recomendação.
- O sistema realizará as validações:
    - *FS2 – Campos obrigatórios não preenchidos;*

Fluxos Secundários (FS):

FS1 – Nenhum registro selecionado

- Caso o usuário clique em algum botão e não tenha selecionado um registro, o sistema exibira uma mensagem dizendo que o usuário deve selecionar um registro.

FS2 - Campo obrigatório não preenchido

- Caso o usuário não preencha algum dos campos obrigatórios, o sistema exibirá uma mensagem única informando que alguns campos obrigatórios não foram preenchidos
.

Requisitos Não Funcionais (RNF):

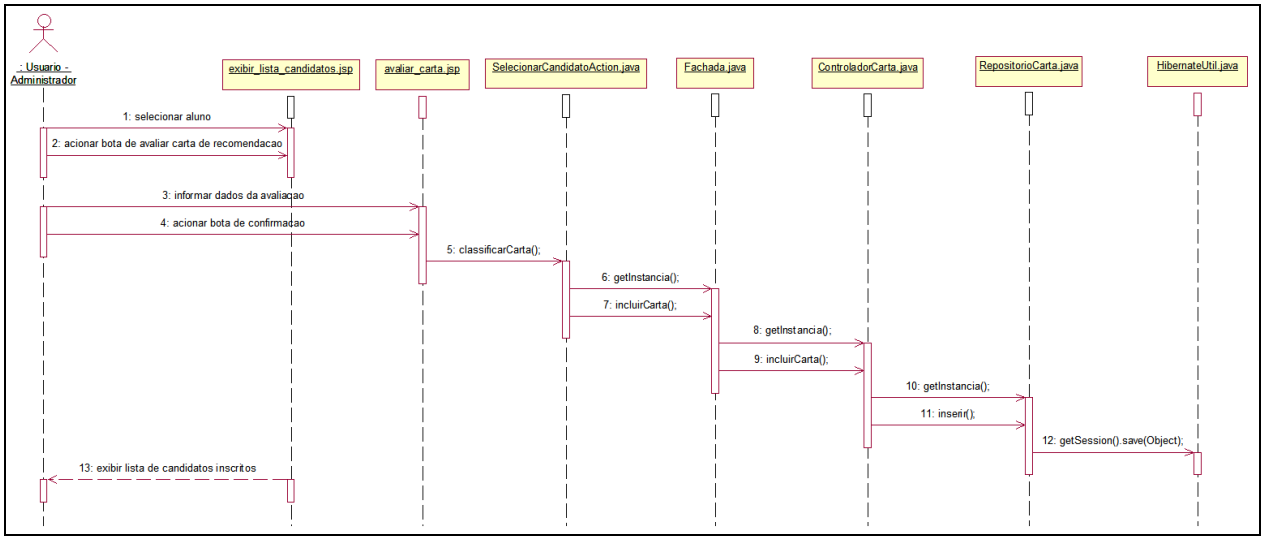
- *Não há*

Anexos e/ou Apêndices:

- **Fórmulas:** *Não há*

- **Lógica de Implementação:** *Não há*

Diagrama de seqüência:



UC_010 Entrar Avaliação Curricular

Descrição: Este caso de uso tem por finalidade permitir o registro da avaliação curricular de um candidato no sistema.

Ator: Usuário com nível de administrador.

Pré-condição:
- Candidato com inscrição confirmada no sistema.

Pós-condição:
- Dados curriculares do candidato em situação “avaliado” no sistema.

Fluxo de eventos principal:

- O usuário selecionará a função de avaliação de dados curriculares através do menu “Entrar Avaliação Curricular de Candidato”.
- O sistema exibirá uma página com uma lista de candidatos com a inscrição já confirmada anteriormente.
- O usuário poderá utilizar uma das seguintes opções:
  - Selecionar um candidato e clicar no botão “Entrar Avaliação Curricular” (SB Avaliação Curricular).
  - *FS1 – Nenhum registro selecionado;*

Sub-Fluxos (SB):

SB Avaliação Curricular
- O sistema solicitará as seguintes informações:

- Tipo da instituição de ensino*
- Tipo da titulação *
- Tipo de formação *
- Peso das experiências curriculares, caso haja alguma cadastrada no sistema *

- O usuário selecionará o botão “Concluir” para concluir o registro das informações sobre os dados curriculares do candidato.
- O sistema realizará as validações:
  - *FS2 – Campos obrigatórios não preenchidos;*

Fluxos Secundários (FS):

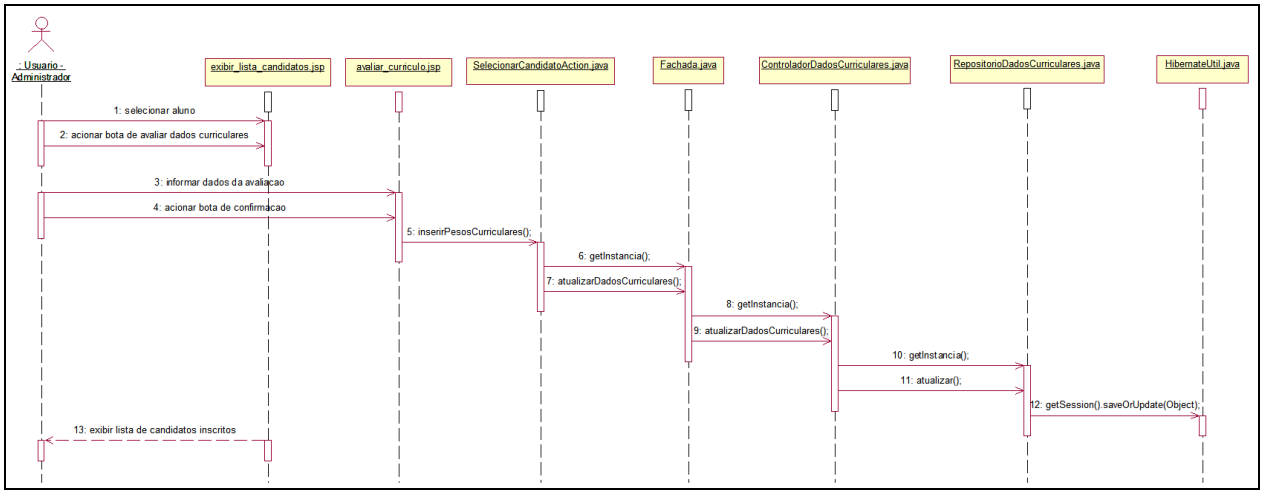
FS1 – Nenhum registro selecionado
- Caso o usuário clique em algum botão e não tenha selecionado um registro, o sistema exibira uma mensagem dizendo que o usuário deve selecionar um registro.

FS2 - Campo obrigatório não preenchido
- Caso o usuário não preencha algum dos campos obrigatórios, o sistema exibirá uma mensagem única informando que alguns campos obrigatórios não foram preenchidos
.

Requisitos Não Funcionais (RNF):
- *Não há*

Anexos e/ou Apêndices:
- **Fórmulas:** *Não há*
- **Lógica de Implementação:** *Não há*

Diagrama de seqüência:



UC_011 Trocar Datas da Seleção

Descrição: Este caso de uso tem por finalidade permitir a mudança das datas do processo seletivo, caso este venha a encerrar-se ou iniciar.

Ator: Usuário com nível de administrador.

Pré-condição:
- Não há.

Pós-condição:
- Datas da seleção modificadas.

Fluxo de eventos principal:

- O usuário devidamente logado como administrador selecionará a função de mudança de datas da seleção de Mestrado candidatos através do menu “Atualizar Datas da Seleção”.
- O sistema exibirá uma página com os campos das novas datas para serem preenchidos (SB Atualizar Datas).

Sub-Fluxos (SB):

SB Atualizar Datas
- O sistema solicitará as seguintes informações:
  - Data de Início*
  - Data de Término*

  FS1 – Campos obrigatórios não preenchidos;
  FS2 – Data inválida;

- O usuário selecionará o botão “Confirmar” para concluir a confirmação da mudança de datas da seleção.
- O sistema mudará o registro do Processo no banco e as novas datas serão consideradas no processo de inscrição.

Fluxos Secundários (FS):

FS1 - Campo obrigatório não preenchido
- Caso o usuário não preencha algum dos campos obrigatórios, o sistema exibirá uma mensagem única informando que alguns campos obrigatórios não foram preenchidos

FS2 – Data inválida
- Condição: O usuário informou uma data invalida ao sistema.
- Mensagem: “Data invalida!”.

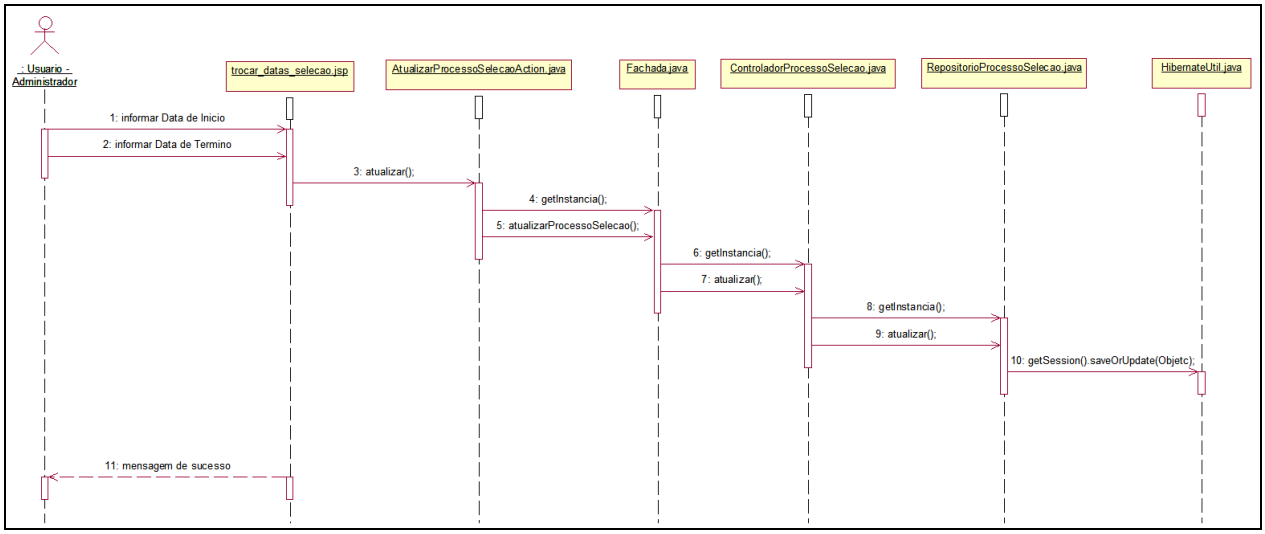
Requisitos Não Funcionais (RNF):
- *Não há*

Anexos e/ou Apêndices:

- **Fórmulas:** *Não há*

- **Lógica de Implementação:** *Não há*

Diagrama de seqüência:



UC_012 Gerar Relatório Final

Descrição: Este caso de uso tem por finalidade permitir a geração do relatório final com os nomes e notas dos candidatos classificados no processo de seleção.

Ator: Usuário com nível de administrador.

Pré-condição:

- Não há.

Pós-condição:

- Não há.

Fluxo de eventos principal:

- O usuário devidamente logado como administrador selecionará a função de geração de relatório final através do menu “Gerar Relatório Final”.
- O sistema exibirá uma página com os campos do relatório preenchidos (SB Gerar Relatório).
- *FS1 – Candidatos pendentes de avaliação;*

Sub-Fluxos (SB):

SB Gerar Relatório

- O sistema exibirá as seguintes informações:
  - Nome do candidato

- Nota da Fase 1 – Peso CR
- Nota da Fase 1 Normalizada – Peso CR
- Nota da Fase 2
- Nota total final
- Nome do professor do projeto escolhido pelo candidato

Fluxos Secundários (FS):

FS1 – Candidatos pendentes de avaliação

- Caso o usuário requisite a geração do relatório final e algum candidato inscrito na seleção ainda estiver com alguma avaliação ou inscrição pendente de confirmação, o sistema devera informar através de mensagem que não será possível a visualização do relatório, pois ainda existem candidatos que não foram avaliados.

Requisitos Não Funcionais (RNF):

- *Não há*

Anexos e/ou Apêndices:

- **Fórmulas:** *Não há*
- **Lógica de Implementação:** *Não há*

UC_013 Entrar Nota de Entrevista

Descrição: Este caso de uso tem por finalidade o cadastro da nota obtida por um candidato na fase de entrevista.

Ator: Usuário com nível de professor.

Pré-condição:

- Candidato cadastrado no sistema e classificado para a fase de entrevista.

Pós-condição:

- Nota obtida pelo candidato na entrevista cadastrada no sistema.

Fluxo de eventos principal:

- O usuário devidamente logado como professor selecionará a função de entrar nota de entrevista de candidatos através do menu “Entrar Nota de Entrevista”.
- O sistema exibirá uma página com uma lista de candidatos inscritos e alocados ao professor logado.
- O usuário poderá utilizar uma das seguintes opções:
  - Selecionar um registro e clicar no botão “Entrar Nota de Entrevista” (SB Cadastrar Nota de Entrevista).
  - *FS1 – Nenhum registro selecionado;*
  - *FS2 – Nenhum candidato encontrado;*

Sub-Fluxos (SB):

SB Cadastrar Nota de Entrevista

- O sistema solicitará a seguinte informação:
  - Nota da entrevista

- O usuário selecionará o botão “Confirmar” para concluir a confirmação da nota obtida pelo candidato na entrevista.
- O sistema registrará a nota da entrevista do candidato no banco de dados.

Fluxos Secundários (FS):

FS1 – Nenhum registro selecionado
- Caso o usuário clique em algum botão e não tenha selecionado um registro, o sistema exibira uma mensagem dizendo que o usuário deve selecionar um registro.

FS2 – Nenhum candidato encontrado
- Caso o professor não possua nenhum candidato alocado a ele nesta fase, o sistema exibirá uma mensagem dizendo que não foi possível localizar nenhum candidato.

Requisitos Não Funcionais (RNF):
- *Não há*

Anexos e/ou Apêndices:
- **Fórmulas:** *Não há*

- **Lógica de Implementação:** *Não há*

Anexo 2

Documento de Análise de Casos de Uso

Este documento tem como propósito analisar todos os casos de uso do SGIP, assim como mostrar o diagrama de classes do sistema.

Anexo 3

Documento de Arquitetura do SGIP

Este documento tem como propósito demonstrar e especificar todo o padrão envolvido na organização da arquitetura do SGIP.

Anexo 3

Documento de Arquitetura do SGIP

Este documento tem como propósito demonstrar e especificar como está organizada a arquitetura do SGIP.