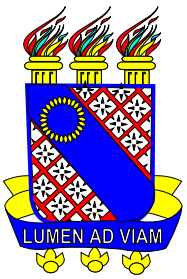

UNIVERSIDADE ESTADUAL DO CEARÁ

WESLEY JEFFERSON OLIVEIRA ALCOFORADO

TCC-MANAGER - UM GERENCIADOR DE TRABALHOS DE CONCLUSÃO DE CURSO PARA O CURSO DE CIÊNCIAS DE COMPUTAÇÃO DA UECE

FORTALEZA, CEARÁ

2011

WESLEY JEFFERSON OLIVEIRA ALCOFORADO

TCC-MANAGER - UM GERENCIADOR DE TRABALHOS DE CONCLUSÃO DE CURSO PARA O CURSO DE CIÊNCIAS DE COMPUTAÇÃO DA UECE

Monografia submetida à coordenação do curso de Ciências da Computação da Universidade Estadual do Ceará, no ano de 2011, como requisito parcial para obtenção do grau de Bacharel em Ciências da Computação.

Orientador: Mariela Cortés

FORTALEZA, CEARÁ

2011

A000z Alcoforado, Wesley Jefferson Oliveira.

TCC-Manager - um gerenciador de trabalhos de conclusão de curso para o curso de Ciências de Computação da UECE / Wesley Jefferson Oliveira Alcoforado. – Fortaleza, 2011.

45p.;il.

Orientador: Prof[a]. Dr[a]. Mariela Cortés

Monografia (Graduação em Ciências da Computação) - Universidade Estadual do Ceará, Centro de Ciências e Tecnologia.

1. Gerenciador 2. Processo 3. Projeto final I. Universidade Estadual do Ceará, Centro de Ciências e Tecnologia.

CDD:000.0

WESLEY JEFFERSON OLIVEIRA ALCOFORADO

TCC-MANAGER - UM GERENCIADOR DE TRABALHOS DE CONCLUSÃO DE CURSO PARA O CURSO DE CIÊNCIAS DE COMPUTAÇÃO DA UECE

Monografia submetida à coordenação do curso de Ciências da Computação da Universidade Estadual do Ceará, no ano de 2011, como requisito parcial para obtenção do grau de Bacharel em Ciências da Computação.

Aprovada em: __/__/____

BANCA EXAMINADORA

Profª. Drª. Mariela Cortés
Universidade Estadual do Ceará - UECE
Orientador

Prof. Dr. Numero Um
Universidade Estadual do Ceará - UECE
Co-orientador

Prof. Me. Numero Dois
Universidade Estadual do Ceará - UECE
Co-orientador

Prof. Dr. Numero Tres
Universidade Estadual do Ceará - UECE

AGRADECIMENTOS

À minha esposa Raquel por ter me incentivado a escrever minha monografia, me apoiando nos momentos em que fraquejei.

Aos amigos de graduação mais próximos pelo apoio mútuo durante toda a jornada na universidade.

Aos professores por terem compartilhado seus conhecimentos conosco.

"L'ordinateur obéit à vos ordres, pas à vos intentions."
Anônimo

SUMÁRIO

1 Introdução 13
1.1 Motivação 13
1.2 Objetivo 13
1.3 Metodologia 13
1.4 Organização do trabalho 14

2 Regulamento de Projeto Final 15
2.1 Entidades envolvidas e suas atribuições 15
2.2 Desenvolvimento do projeto 16
2.2.1 Proposta do projeto 16
2.2.2 Defesa do projeto 17
2.2.3 Resumo do fluxo do desenvolvimento do projeto final 17

3 Processo de desenvolvimento da aplicação 19
3.1 Escolha das tecnologias 19
3.1.1 Linguagem de programação 19
3.1.2 Framework 20
3.1.2.1 MVC 20
3.1.2.2 ORM 21
3.1.2.3 *Convention over configuration* 21
3.1.3 SGBD 22
3.1.4 IDE 23
3.1.5 Outros requisitos de plataforma 23
3.2 Casos de uso e especificação de requisitos do sistema 24
3.2.1 Diagrama de classes 24
3.2.1.1 Autenticação de usuários 27

3.2.1.2 Formulários 27

4 Utilização do sistema 29

5 Conclusões e trabalhos futuros 30

5.1 Trabalhos futuros 30

Referências Bibliográficas 31

6 Apêndice 32

6.1 Casos de uso 33

6.1.1 Manter Professores 33

6.1.2 Manter Estudantes 35

6.1.3 Manter Projetos 37

6.1.4 Manter Propostas 39

6.1.5 Manter Defesas 41

7 Anexos 43

7.1 Formulário de proposta de projeto final 44

7.2 Formulário de solicitação de defesa e banca 45

LISTA DE FIGURAS

Figura 2.1 Fluxograma do desenvolvimento do projeto final 18

Figura 3.1 Arquitetura MVC 21

Figura 3.2 Definição de um mapeamento no Hibernate 22

Figura 3.3 Tabela Users no banco de dados 22

Figura 3.4 Descrição de uma tabela no Symfony 24

Figura 3.5 Diagrama de classes base 25

Figura 3.6 Diagrama de classes extendidas 25

Figura 3.7 Diagrama de classes da camada de controle 26

Figura 3.8 Diagrama de classes dos formulários 27

LISTA DE TABELAS

LISTA DE SIGLAS

UECE	Universidade Estadual do Ceará
PHP	PHP: Hypertext Preprocessor
TCC	Trabalho de Conclusão de Curso
IES	Instituição de Ensino Superior
MVC	Model-View-Controller
HTML	HyperText Markup Language
ORM	Object-relational mapping
SQL	Structured Query Language
XML	eXtensible Markup Language
SGBD	Sistema de Gerenciamento de Banco de Dados
DARPA	Advanced Research Project Agency
ARO	Army Research Office
IDE	Integrated Development Environment
HTTP	Hypertext Transfer Protocol
PDO	PHP Data Objects
SMTP	Simple Mail Transfer Protocol
YAML	YAML Ain't Markup Language
URL	Uniform Resource Locator
CRUD	Create, Retrieve, Update e Delete

RESUMO

Este trabalho tem como objetivo apresentar uma ferramenta que informatiza o processo de avaliação de trabalhos de conclusão da graduação em Ciências da Computação da UECE. Essa ferramenta foi projetada para minimizar a quantidade de documentos impressos e para proporcionar uma visão geral do andamento das atividades. Além disso, ela é capaz de notificar as pessoas envolvidas automaticamente, alertando sobre o cumprimento dos prazos.

Palavras-Chave: Gerenciador, Processo, Projeto final

ABSTRACT

This work aims to present a manager tool for the process of evaluating the course completion assignments of the undergraduating students in Computer Science at Ceara's State University. This tool was projected to minimize the quantity of printed documents and to provide an overview on the progress of activities. Besides, it is capable of notifying involved people automatically, making them aware of deadlines.

Keywords: Manager, Process, Course completion assignment

1 INTRODUÇÃO

Através da informatização de processos, é possível otimizar tarefas que demandam tempo, um bem escasso, e também diminuir a quantidade de documentos impressos nas nossas mesas, papéis que muitas vezes se perdem no meio de outros documentos.

1.1 Motivação

No último semestre do curso de Ciências da Computação existe a disciplina de Projeto Final, a qual é regida por um regulamento e possui todo um processo burocrático que pode ser informatizado, de forma a organizar melhor as tarefas dos professores e alunos envolvidos.

A construção de um sistema que gerencie o processo de submissão de projetos ajudaria não apenas na organização dos documentos e datas, mas também possibilitaria aos orientadores e à Comissão de Projeto Final ter um controle dos alunos que, por um motivo ou outro, não sabem por onde começar ou estão parados no meio do caminho, além de manter todos informados sobre o andamento dos seus projetos.

1.2 Objetivo

Com a construção de um sistema de gerenciamento de submissão de projetos, espera-se uma diminuição no esforço despendido na organização e um maior controle sobre os documentos, datas, alunos e professores envolvidos.

1.3 Metodologia

Para o desenvolvimento deste trabalho foram inicialmente construídos casos de usos simplificados, onde as funcionalidades foram expostas com mais clareza, mas sem muito aprofundamento para garantir que o projeto fosse concluído em tempo hábil.

A partir dos casos de uso, o banco de dados foi modelado, assim como as classes de acesso a dados.

Após o término da modelagem, foi iniciada a fase de codificação dos casos de uso,

seguidos pelo desenvolvimento do design da aplicação.

Por último foram realizados testes de aceitação e correções de bugs encontrados.

Para a construção dessa aplicação foi utilizada a linguagem de script PHP, na sua versão 5.3.3 em conjunto com o framework Symfony 1.4.2. Como IDE, foi utilizado o NetBeans 6.9, um framework para desenvolvimento e testes com PHP. O banco de dados escolhido foi o PostgreSQL versão 8.4.

Estas ferramentas foram escolhidas devido a serem gratuitas e possuirem versões estáveis para Linux, que é a plataforma onde se pretende implantar a versão final da aplicação. Mais detalhes sobre as tecnologias são expostas na Seção 3.1.

1.4 Organização do trabalho

Além deste capítulo introdutório, o presente trabalho consiste em mais 4 capítulos. No Capítulo 2 é feita uma apresentação do processo de entrega de TCCs na UECE, de acordo com o regulamento da instituição. No Capítulo 3 é descrito como foi o processo de desenvolvimento da aplicação, assim como a organização do banco de dados e os requisitos do sistema. O Capítulo 4 demonstra como utilizar o sistema. O Capítulo 5 apresenta as conclusões e uma breve discussão sobre trabalhos futuros.

2 REGULAMENTO DE PROJETO FINAL

O processo de submissão e avaliação de projetos finais no curso de Ciências da Computação da UECE segue um regulamento que normatiza o tipo do conteúdo da monografia, o procedimento para o desenvolvimento e para a aprovação do projeto, a constituição da Comissão de Projeto Final e as atribuições desta, do orientador e do aluno.

2.1 Entidades envolvidas e suas atribuições

O desenvolvimento do projeto final deve ser desempenhado individualmente pelo aluno, sob a orientação de um docente, o orientador. O orientador deve ser um docente lotado no curso de Ciências da Computação da UECE, seja ele professor efetivo, substituto ou visitante. O aluno pode ainda contar com a colaboração de co-orientadores, podendo estes serem docentes da UECE ou de outras IES ou ainda profissionais com graduação plena em Ciências da Computação ou cursos afins e com no mínimo 3 (três) anos de experiência em orientação de alunos ou coordenação de projetos.

A Comissão de Projeto Final, ou simplesmente Comissão, é o órgão responsável pelo acompanhamento do processo de desenvolvimento do projeto final. Ela é composta por 3 (três) docentes efetivos pertencentes ao curso de Bacharelado em Ciência da Computação da UECE, havendo ainda 2 (dois) membros suplentes, sendo todos esses (permanentes e suplentes) escolhidos através de eleição no Colegiado do curso de Ciência da Computação e nomeados pelo Coordenador da graduação. O membro da Comissão fica impedido de emitir parecer sobre o trabalho de seus orientandos, que, neste caso, deverão ser avaliados por um membro suplente.

Compete ao aluno:

a. elaborar projeto de Proposta de Projeto Final;

b. conduzir e executar o Projeto Final;

c. cumprir os prazos estabelecidos no cronograma pré-estabelecido;

d. redigir e defender o Projeto Final;

e. entregar cópia corrigida do Projeto Final à secretaria;

f. tomar ciência dos prazos estabelecidos pela Comissão de Projeto Final e cumpri-los.

Compete ao orientador e co-orientador:

a. orientar o aluno na organização de seu plano de estudo, pesquisa e assistí-lo na preparação da monografia;

b. viabilizar a realização do Projeto Final;

c. encaminhar a Proposta de Projeto Final e a Solicitação de Defesa à Comissão;

d. propor à Comissão a composição da Banca Examinadora;

e. encaminhar a Ata de Defesa, devidamente preenchida e assinada, ao Coordenador do curso.

Compete à Comissão:

a. aprovar a proposta e plano de trabalho de Projeto Final;

b. aprovar as indicações dos orientadores de Projeto Final que não sejam docentes do curso;

c. aprovar os membros das bancas avaliadoras do Projeto Final;

d. autorizar a defesa de monografia de Projeto Final;

2.2 Desenvolvimento do projeto

O projeto pode ser iniciado antes do aluno se matricular na disciplina de Projeto Final, porém o processo de desenvolvimento do trabalho deve ser realizado no último ano do curso. O desenvolvimento do projeto final é constituido das seguintes partes:

a. Apresentação da proposta de projeto final à comissão de projeto final;

b. Solicitação de defesa do projeto final e indicação de comissão examinadora à comissão de projeto final;

c. Defesa do projeto final em seção pública diante da comissão examinadora;

d. Entrega do texto final da monografia.

2.2.1 Proposta do projeto

O aluno deve apresentar uma proposta de projeto (ver Anexo 7.1) a partir do início do período letivo em que se matriculou na disciplina de Projeto Final. A data máxima de apresentação da proposta é de 100 (cem) dias antes da data da colação oficial ou especial.

A apresentação da proposta deve ter a anuência do orientador e é avaliada pela comissão. Após a aprovação da proposta pela comissão, fica autorizado o início do desenvolvimento do trabalho. A proposta deve conter os seguintes tópicos:

a. Motivação e Objetivo;

b. Fundamentação teórica;

c. Metodologia;

d. Bibliografia;

e. Cronograma.

2.2.2 Defesa do projeto

O aluno, com a anuência do orientador, deve encaminhar uma solicitação de defesa à comissão (ver Anexo 7.2), no mínimo 60 (sessenta) dias após a aprovação da proposta e no máximo até 30 (trinta) dias antes da colação oficial ou especial. Junto da solicitação deverão seguir a data de sugestão da defesa, a indicação dos membros da comissão examinadora e 1 (uma) cópia da monografia.

Sendo o parecer da comissão favorável, o aluno tem um prazo de até 10 (dez) dias antes da colação oficial ou especial para realizar a defesa, e 7 (sete) dias antes desta para entregar à comissão examinadora exemplares da monografia.

Sendo o aluno aprovado na defesa, ele deverá entregar à secretaria do curso 3 (três) exemplares impressos da monografia e 1 (uma) cópia em meio eletrônico. Somente após isso é que será autorizada a emissão e entrega do diploma ao aluno.

2.2.3 Resumo do fluxo do desenvolvimento do projeto final

A Figura 2.1 apresenta um fluxo relativo ao processo de desenvolvimento do projeto final. Inicialmente, o aluno submete uma proposta de acordo com o modelo contido no Anexo 7.1, que deve ter a anuência do seu orientador. Caso o orientador não aprove a proposta, convém que o aluno converse com seu orientador para corrigir os eventuais problemas para que dessa forma possa submeter a proposta novamente. Após obter o aval do orientador, a proposta pode ser encaminhada à comissão. Surgindo algum problema na aprovação da proposta por parte da comissão, o aluno deve corrigir os problemas indicados e submetê-la novamente. Caso a proposta seja aprovada, o aluno pode começar a desenvolver o trabalho. Finalizado o desenvolvimento, o aluno pode solicitar a defesa do seu projeto enviando 1 (uma) cópia da monografia e preenchendo o formulário do Anexo 7.2. O pedido deve novamente contar com a anuência do orientador, para depois seguir para a comissão. Sendo a defesa aprovada pela comissão, o aluno está apto para defender seu projeto ante a comissão examinadora.

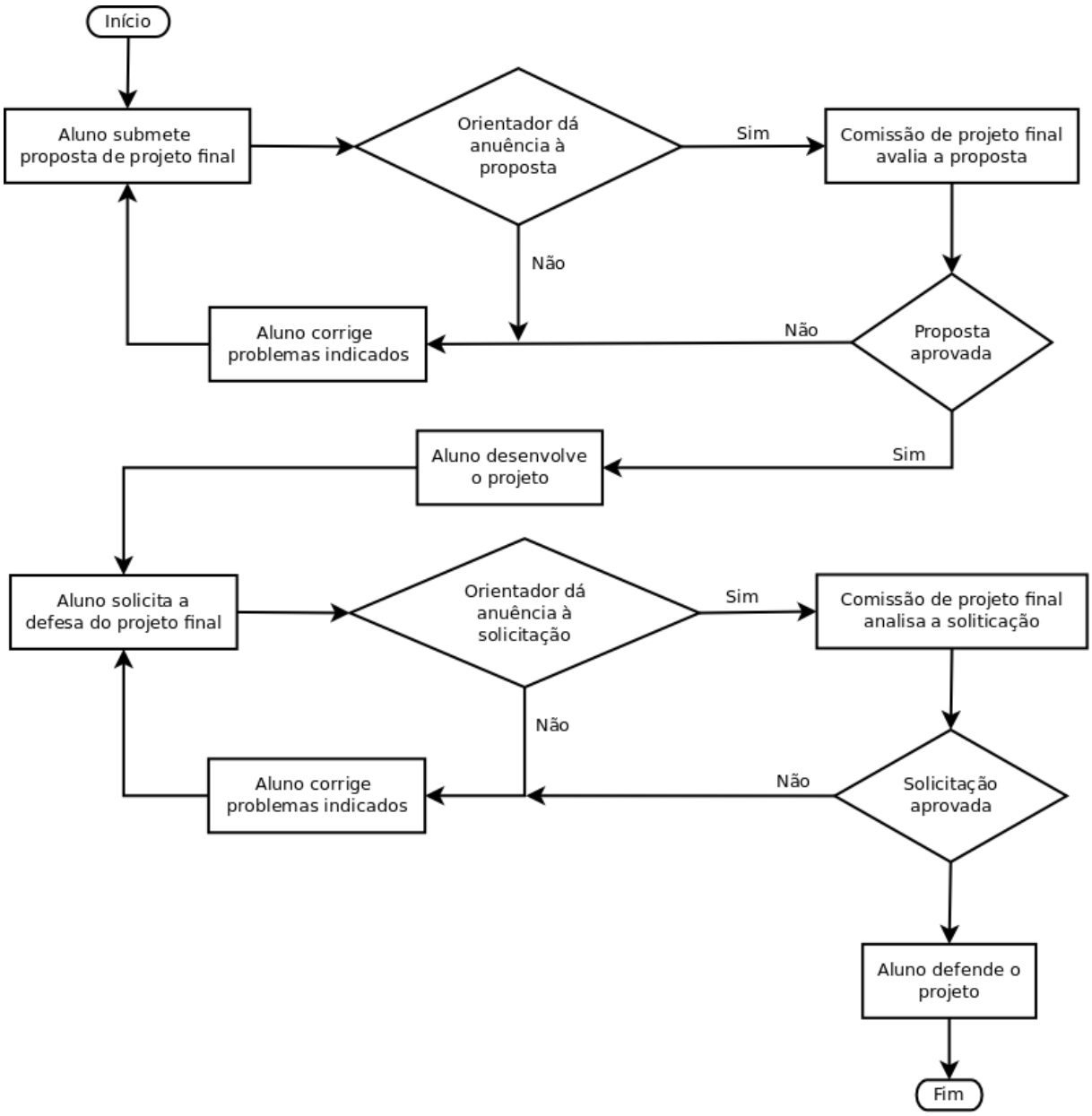

Figura 2.1: Fluxograma do desenvolvimento do projeto final

3 PROCESSO DE DESENVOLVIMENTO DA APLICAÇÃO

No capítulo anterior foi apresentado o processo de desenvolvimento e avaliação dos TCCs no curso de Ciências da Computação da UECE, como este se encontra atualmente.

A partir da visão geral fornecida pelo fluxograma da Figura 2.1 foi possível definir o comportamento básico da aplicação. Contudo, a aplicação não se detém a apenas seguir esse fluxo básico. Ela também possui funcionalidades de alerta no sistema, para que os envolvidos no processo sejam sempre notificados a cada vez que um estudante, orientador ou comissão executem alguma operação em cima de um projeto.

Neste capítulo discutiremos sobre como o sistema foi modelado, assim como sobre as estratégias delineadas para a notificação de eventos.

3.1 Escolha das tecnologias

A seguir são apresentadas as tecnologias escolhidas como parte da plataforma de desenvolvimento do sistema.

3.1.1 Linguagem de programação

O PHP é uma linguagem de programação interpretada, multiparadigma, de código aberto, e especialmente voltada para o desenvolvimento de aplicações para a Web. Possui uma sintaxe que lembra C, Java e Perl, e se distingue das demais por sua facilidade de aprendizado. Começou a ser desenvolvida em 1994 por Rasmus Lerdorf, mas o paradigma de orientação a objetos só foi introduzido a partir da versão 3 (três), amadurecendo na versão 5 (cinco) (PHP.NET, 2011; WIKIPEDIA, 2011a).

O PHP se encontra disponível na grande maioria dos servidores Web e, devido a sua facilidade de aprendizado, possui uma vasta comunidade de desenvolvedores. Ele é usado em alguns gigantes da Web, como Facebook, Wikipédia e Wordpress (INFOQ, 2009; WIKIPEDIA, 2011b; WORDPRESS, 2011).

Entre outros fatores, esta linguagem foi escolhida devido a sua difusão, sendo então mais provável encontrar outros desenvolvedores que possam dar continuidade ao projeto deste

trabalho, adicionando novas funcionalidades ou corrigindo eventuais problemas.

3.1.2 Framework

De acordo com (MINETTO, 2007), um framework de desenvolvimento é uma base de onde se pode desenvolver algo maior ou mais específico. É uma coleção de códigos-fonte, classes, funções, técnicas e metodologias que facilitam o desenvolvimento de novos softwares.

O uso de um framework é essencial para desenvolver uma aplicação rapidamente sem deixar de seguir boas práticas de programação. Além disso, um programador que conheça um framework não tem dificuldades para entender o código de outras pessoas, pois o framework obriga todos a seguirem as mesmas convenções. Dessa forma o programador que não conhece a aplicação pode se manter apenas no entendimento da lógica de negócio, sem se perder na arquitetura da aplicação.

Para a aplicação desenvolvida neste trabalho, o framework escolhido foi o Symfony. O Symfony é um framework que possui alta aceitação na comunidade PHP, boa documentação, é patrocinado pela empresa francesa Sensio Labs, que garante suporte técnico a longo prazo, e tem várias outras qualidades que o coloca entre os melhores frameworks PHP, como:

a. suporte a PHP 5;

b. suporte a MVC;

c. validação de formulários;

d. extensa documentação;

e. suporte a plugins;

f. geração de código;

g. suporte a ORM e múltiplos bancos de dados;

h. convention over configuration;

i. suporte a testes unitários.

3.1.2.1 MVC

O MVC é um acrônimo para Model-View-Controller, um padrão de projeto que tem como intuito separar a lógica de negócio, a interface e os modelos de acesso a dados. Essa separação de conceitos tem como propósito evitar que o código fique difícil de manter e auxilia significamente no aumento do reuso de código. A Figura 3.1 apresenta de maneira geral a estrutura desse tipo de arquitetura.

O modelo de acesso a dados é representado pela camada Model (ou camada de modelo). O modelo é um objeto que representa alguma informação sobre o domínio. É um objeto

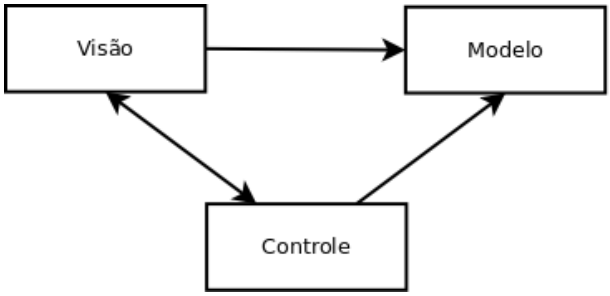

Figura 3.1: Arquitetura MVC

não-visual que contém todos os dados e comportamentos outros que não os utilizados pela interface (FOWLER, 2006).

A camada View (ou camada de visão) representa a interface da aplicação. No trabalho em questão, ela é a parte em HTML. Esta camada não deve possuir nenhuma lógica de negócio, detendo-se apenas à captura e exibição de dados.

A camada Controller (ou camada de controle) é a responsável por conectar as outras duas camadas. O controlador recebe a entrada do usuário (capturado pela visão), manipula o modelo e faz com que a visão seja atualizada apropriadamente (FOWLER, 2006).

3.1.2.2 ORM

Atualmente a maioria das aplicações é desenvolvida utilizando o paradigma de programação orientado a objetos apoiada por um banco de dados relacional. Essas aplicações precisam carregar dados de um banco de dados, criar objetos para representar esses dados em memória, executar operações em cima destes objetos e depois salvar de volta as alterações no banco.

Ferramentas de mapeamento objeto-relacional (ou ORM) são frameworks que recuperam e persistem objetos. Seu objetivo é dar suporte à complexa atividade de gerenciar conexões entre objetos e um banco de dados relacional. A persistência fica transparente ao desenvolvedor, já que ele não precisa se preocupar com os detalhes de implementação. A ponte entre objetos e seus relacionamentos é realizada pela ferramenta ORM segundo a especificação de mapeamento dos dados (CABIBBO; CAROSI, 2005).

3.1.2.3 *Convention over configuration*

Frameworks de propósito geral normalmente necessitam de um ou mais arquivos de configuração para serem utilizados adequadamente. Um arquivo de configuração mapeia uma classe e um recurso (uma tabela no banco de dados) ou um evento (uma requisição web). À medida em que a complexidade das aplicações cresce, os arquivos de configuração também crescem, tornando-se difíceis de manter (CHEN, 2006). Para evitar este mal desnecessário, muitos frameworks atualmente procuram seguir o modelo de desenvolvimento de software de Convenção sobre Configuração (*Convention over Configuration*). A idéia é basicamente

fazer com que o desenvolvedor só precise definir aquilo que não segue uma convenção pré-estabelecida.

```
<hibernate-mapping>
<class name="User" table="users">
  <id name="ID" column="id" type="string">
      <generator class="assigned"></generator>
  </id>
  <property name="password"  column="password"  type="string" />
</class>
</hibernate-mapping>
```

Figura 3.2: Definição de um mapeamento no Hibernate

```
CREATE TABLE users (
  id VARCHAR(20) NOT NULL,
  password VARCHAR(20),
  PRIMARY KEY(id)
);
```

Figura 3.3: Tabela Users no banco de dados

A Figura 3.2 apresenta um arquivo de mapeamento para Hibernate, um framework de mapeamento objeto-relacional para Java. O código da Figura 3.2 mapeia a classe User com a tabela Users no banco de dados. A tabela Users é descrita na Figura 3.3 usando SQL. Os campos da classe User também são mapeados para as colunas da tabela Users.

O ato de modificar arquivos de configuração, normalmente em XML, é tedioso e propenso a erros. A maioria dos problemas de configuração só vai ser detectado em tempo de execução, disparando exceções na aplicação, que tendem a diminuir o ritmo do desenvolvimento e consequentemente a produtividade. Mais importante ainda, uma grande parte do mapeamento poderia ser inferido facilmente pela estrutura da tabela sem a necessidade de configuração alguma. Por exemplo, pode-se estabelecer uma convenção de que:

1. Nomes de tabelas devem ser o nome da classe no plural.
2. As colunas na tabela devem ter nomes idênticos aos campos que a classe mapeia.

Estas duas convenções são naturais, e, de fato, já são seguidas pela maioria dos desenvolvedores. O padrão de convenção sobre configuração reduz a quantidade de configuração ao estabelecer um conjunto de convenções de nomenclatura que todos os desenvolvedores devem seguir (CHEN, 2006).

3.1.3 SGBD

O PostgreSQL é um SGBD livre, de código aberto, bastante robusto e confiável. Derivou-se do projeto POSTGRES da universidade de Berkley, que iniciou-se em 1986 e foi patrocinado por instituições militares americanas como a DARPA (Agência de Projetos de Pesquisa Avançada para Defesa) e ARO (Departamento de Pesquisa Militar). A linguagem de consultas SQL foi inserida quando o nome do projeta era Postgres95, tendo sido rebatizado para o nome atual em 1996 para enfatizar a relação do SGBD com o SQL (POSTGRESQL, 2003).

Apesar do PostgreSQL ter sido escolhido para o desenvolvimento deste trabalho, outros SGBDs podem ser utilizados em seu lugar, devido ao framework Symfony possuir um

ORM que abstrai a comunicação da aplicação com o banco de dados. Basta configurar a conexão do banco de dados no Symfony e avisá-lo qual SGBD estará em uso que o seu ORM se encarregará de fazer a comunicação correta com o banco de dados.

3.1.4 IDE

O NetBeans é um ambiente integrado de desenvolvimento (IDE) gratuito, também de codigo aberto e atualmente patrocinado pela Oracle. Originalmente suportava apenas a linguagem Java, mas atualmente consegue trabalhar com diversas linguagem de programação, entre elas o PHP, e além disso possui plugins que facilitam a utilização de alguns frameworks, inclusive o Symfony.

Foi escolhido para este trabalho por fornecer um bom suporte ao PHP e ao Symfony, e por possuir fácil integração com a ferramenta de depuração Xdebug, específica para PHP.

3.1.5 Outros requisitos de plataforma

Para que a aplicação funcione corretamente, é necessário que o servidor onde ela será atenda a alguns pré-requisitos. São eles:

PHP Deve ser utilizada a versão 5.2.4 ou mais recente (exceto a versão 5.2.9).

Servidor Recomenda-se a utilização do servidor Apache versão 2 ou superior, com a extensão mod_rewrite instalada. O projeto não foi testado com outros servidores HTTP.

SGBD Recomenda-se PostgreSQL 8.4 como SGBD por ter sido utilizado durante o desenvolvimento da aplicação, mas de acordo com a documentação do Doctrine, o ORM do Symfony, qualquer banco de dados suportado pelo PHP através dos drivers PDO pode ser utilizado, já que ele utiliza PDO para se comunicar com o banco de dados. A aplicação foi seguramente testada com MySQL Community Edition 5.5.9 e SQLite 3.7.5, podendo estes também serem utilizados sem prejuízo algum ao funcionamento do sistema. Para qualquer que seja o banco escolhido, o driver PDO deve estar instalado e configurado no PHP.

E-mail É necessária uma conta de email em um servidor que aceite conexões externas via SMTP para o envio das mensagens eletrônicas. Alternativamente pode-se utilizar o Sendmail, caso este esteja configurado no servidor, ou deixar a cargo da função mail do PHP. Recomenda-se configurar um servidor SMTP, especialmente pela facilidade de configuração deste se comparado ao Sendmail. Não é recomendado utilizar a função mail do PHP, pois os emails enviados tendem a ser identificados como spam em muitos servidores de email.

3.2 Casos de uso e especificação de requisitos do sistema

Para aproveitar a funcionalidade de comunicação com diferentes tipos de SGBDs, o ORM do Symfony nos permite definir toda a estrutura do banco de dados textualmente, no formato YAML, que possui maior legibilidade que o SQL. A Figura 3.4 apresenta um exemplo de como as tabelas são descritas no Symfony. Após a descrição de todas as tabelas, executamos um comando no Symfony que gera o SQL necessário para a criação das tabelas de acordo com o SGBD escolhido. Ele também possui um comando que gera todas as classes e formulário necessários para que possamos trabalhar com essas tabelas que acabamos de definir.

```yaml
Estudante:
  columns:
    id:
      primary: true
      type: integer
    telefone:
      type: string(14)
  relations:
    Usuario:
      local: id
      type: one
      foreignType: one
      cascade: [delete]
```

Figura 3.4: Descrição de uma tabela no Symfony

Essa geração automática de código é uma ferramenta extremamente poderosa que o framework nos fornece para evitar perder tempo com tarefas triviais. Mas e se já tivermos começado a codificar nas classes que ele gerou, e lembrarmos que precisamos de mais um campo na tabela? Será que vamos ter todo o código perdido, já que ele vai sobrescrever as classes geradas automaticamente? Isso não acontece, pois o Symfony gera duas classes para cada tabela que nós definirmos. Cada classe herda de uma classe base abstrata, que é sobrescrita toda vez que pedimos ao Symfony que gere as classes de acordo com a especificação. Nós temos que trabalhar em cima dessas classes que herdam as classes bases abstratas, pois essas últimas é que o Symfony sempre vai sobrescrever.

Pra ficar mais claro, damos uma olhada no diagrama da Figura 3.5. Ela apresenta todas as classes base que o Symfony gerou automaticamente para a aplicação deste trabalho. Para cada classe BaseEntidade, há uma classe Entidade, como exposto na Figura 3.6, que é criada uma única vez, caso ela ainda não exista. É nesta última que colocamos os métodos da nossa lógica de negócios. Tendo isso em mente, podemos prosseguir na explicação dos diagramas.

3.2.1 Diagrama de classes

A Figura 3.5 apresenta o diagrama de classes do TCC-Manager. Todas as classes herdam direta ou indiretamente da classe sfDoctrineRecord, pertencente ao framework ORM do Symfony. Essa associação não foi explicitada no diagrama para que ele ficasse mais claro. A classe sfDoctrineRecord é uma classe abstrata que possui todos os métodos de persistência dos objetos.

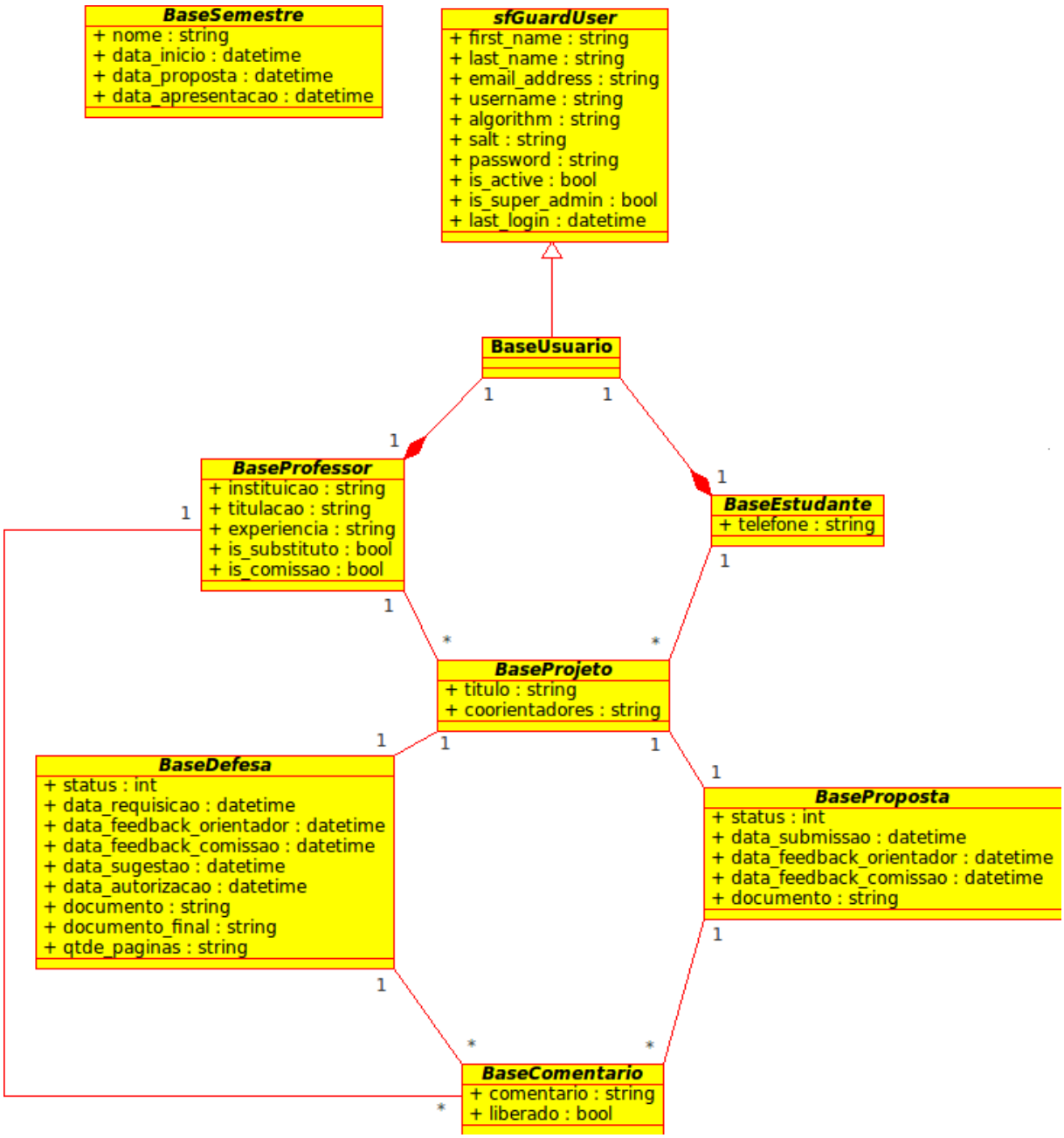

Figura 3.5: Diagrama de classes base

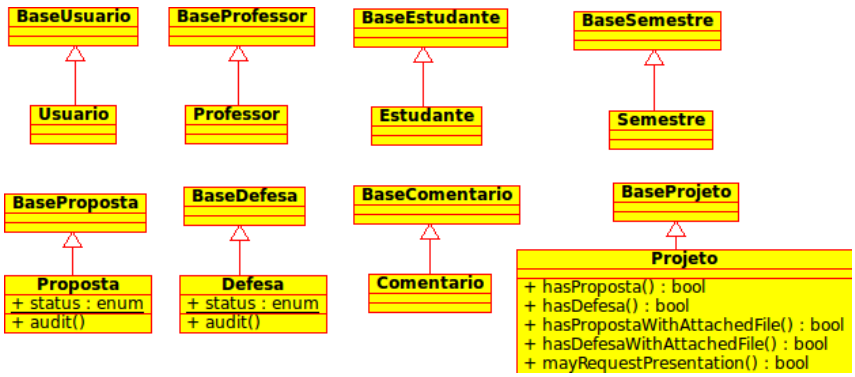

Figura 3.6: Diagrama de classes extendidas

A Figura 3.6 apresenta as classes das entidades de trabalho, isto é, aquelas que herdam das classes base. Nelas é que deve ser inserida qualquer lógica associada à entidade, pois o framework não escreve nessas classes, a não ser para criá-las pela primeira vez. As classes Proposta e Defesa possuem o método audit, que é invocado quando da avaliação, por parte da comissão, da proposta ou da solicitação de defesa. Cada vez que um professor integrante da comissão dá seu parecer, este método salva as informações do integrante, seu parecer e um possível comentário feito por ele sobre a proposta/defesa. Além disso, ele calcula o parecer final dependendo da decisão da maioria da comissão. Essas duas entidades também guardam uma enumeração que indica seu status, que podem ser os seguintes:

- NAO_ANALISADO: Proposta/Defesa não analisada
- APROVADO: Proposta/Defesa aprovada pelo orientador
- REPROVADO: Proposta/Defesa reprovada pelo orientador
- LIBERADO: Proposta/Defesa aprovada pela comissão
- NAO_LIBERADO: Proposta/Defesa reprovada pela comissão

A classe Projeto possui alguns métodos de verificação de situação. Os nomes dos métodos são bem autodescritivos e indicam se o projeto possui uma proposta/defesa, se o estudante anexou o documento da proposta ou a cópia da monografia na solicitação de defesa, e se o projeto já pode ser defendido.

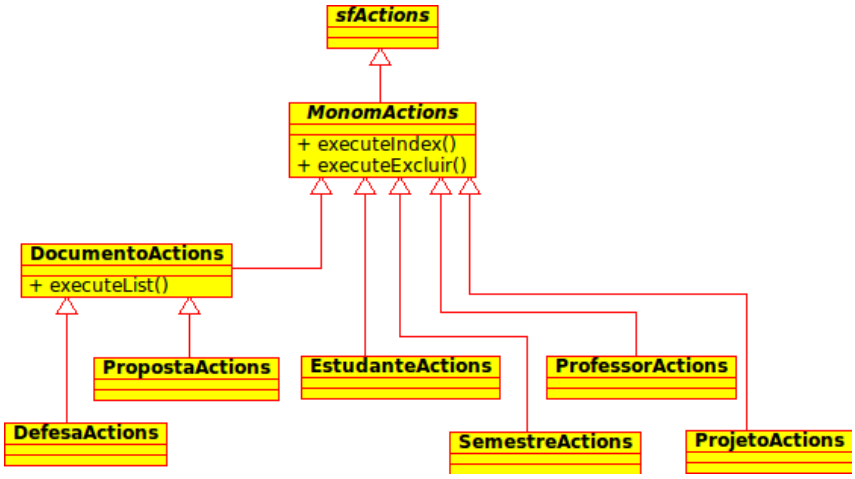

Figura 3.7: Diagrama de classes da camada de controle

A Figura 3.7 apresenta o diagrama de classes para a camada de controle. A classe sfActions pertence ao Symfony e cuida da comunicação entre o navegador e a aplicação. Internamente é ela que instancia as classes necessárias para a execução do framework; decodifica a URL da requisição, de forma a determinar qual ação e qual módulo estão sendo requisitados, e se não existirem, exibe uma mensagem de página não encontrada; recebe os parâmetros de entrada; chama a ação do módulo requisitado e renderiza a saída.

Na classe de controle monomActions encontram-se métodos de CRUD padronizados, e que são usados pela maior parte dos módulos da aplicação. A classe de controle documentoActions contém métodos de controle de upload de arquivos, que são usados nos módulos de proposta e defesa. Dessa forma, foi possível fazer um grande reuso de código.

3.2.1.1 Autenticação de usuários

Na Figura 3.5 podemos observar a classe sfGuardUser, da qual a classe Usuario extende. Ela vem de um plugin para o Symfony de controle de usuários, que oferece recursos de segurança e autorização em cima dos recursos de segurança oferecidos pelo framework. Aproveitamos esse plugin para fazer um bom reuso de código e contar com toda a segurança que o plugin já disponibiliza (SYMFONY, 2010).

As classes Estudante e Professor possuem um relacionamento de composição com a classe Usuario. Inicialmente foi pensado em fazer com que essas classes herdassem da classe Usuario, mas visto que existe o usuário administrador que não é nem estudante, nem professor, a modelagem não pôde ser feita dessa forma. De qualquer forma, só podem existir estudantes e professores se houver um usuário associado a eles.

Na classe Usuario encontram-se os dados mais básicos de cada usuário, como nome, endereço de email e senha. Há ainda outras informações que dizem respeito a como a senha é criptografada no banco de dados. O campo algorithm contém o algoritmo utilizado para criptografar a senha, que por padrão é o SHA1. Outros algoritmos podem ser utilizados, como o MD5, mas optou-se por deixar o algoritmo padrão. O campo salt é uma string que é concatenada à senha antes da criptografia, para dificultar a a descoberta do valor original.

3.2.1.2 Formulários

Além de gerar as entidades, o Symfony também possui a funcionalidade de gerar formulários que possuem validação própria. Dessa forma, só é necessário ajustar detalhes de aparência ou validações mais complexas, como datas que devem preceder umas às outras ou valores que devem estar dentro de um conjunto fechado.

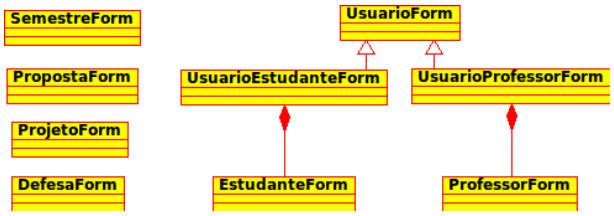

Figura 3.8: Diagrama de classes dos formulários

A Figura 3.8 apresenta os formulários que foram gerados a partir da descrição do esquema do banco de dados. Todas as classes desse diagrama herdam das suas correspondentes classes base, assim como acontece com as classes da Figura 3.6. Elas não são apresentadas

aqui por efeito de clareza no diagrama. As classes base, por sua vez, herdam da classe BaseFormDoctrine, do framework, que possui os métodos básicos de validação e persistência dos formulários.

O formulário do Anexo 7.1 é representado no sistema pelos formulários de estudante, professor, projeto e proposta. O formulário do Anexo 7.2 é representado no sistema pelos formulários de estudante, professor, projeto e defesa.

4 UTILIZAÇÃO DO SISTEMA

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi apresentada uma solução para um problema comum do dia-a-dia da coordenação do curso de Ciências da Computação da UECE - ter que lidar com muita papelada referente aos TCCs dos alunos.

A ferramenta se propõe a gerenciar o processo de submissão de projetos finais, seguindo o regulamento estabelecido pela universidade, e também a manter todos atentos ao cumprimento dos prazos.

Pessoalmente, espero que a aplicação desenvolvida neste trabalho seja útil para o curso de Ciências da Computação da UECE, público alvo desta monografia, e que eu tenha conseguido dar minha humilde contribuição para o nosso curso, de onde tirei a maior parte do meu conhecimento em computação.

5.1 Trabalhos futuros

A partir da contribuição apresentada pelo presente trabalho, alguns trabalhos futuros podem ser indicados visando estender sua funcionalidade como:

- Construir um sistema de visualização e pesquisa das monografias defendidas, com exportação de citações para diferentes formatos, como o BIBTEX.
- Estender o funcionamento do sistema para também atender defesas de mestrado e doutorado.
- Generalizar a ferramenta para atender quaisquer tipo de defesa de qualquer curso universitário.

REFERÊNCIAS BIBLIOGRÁFICAS

CABIBBO, L.; CAROSI, A. Managing inheritance hierarchies in object/relational mapping tools. In: PASTOR, O.; CUNHA, J. Falcão e (Ed.). *Advanced Information Systems Engineering*. Springer Berlin / Heidelberg, 2005, (Lecture Notes in Computer Science, v. 3520). p. 93–124. Disponível em: <http://dx.doi.org/10.1007/11431855_11>.

CHEN, N. *Convention over Configuration*. 2006. Acesso em 24 de fevereiro de 2011. Disponível em: <http://softwareengineering.vazexqi.com/files/pattern.html>.

FOWLER, M. *Padrões de Arquitetura de Aplicações Corporativas*. São Paulo: Bookman, 2006.

INFOQ. *Facebook: Science and the Social Graph*. 2009. Acesso em 24 de fevereiro de 2011. Disponível em: <http://www.infoq.com/presentations/Facebook-Software-Stack>.

MINETTO, E. L. *Frameworks para Desenvolvimento em PHP*. Chapecó - SC: Novatec, 2007.

PHP.NET. *Manual do PHP*. 2011. Acesso em 24 de fevereiro de 2011. Disponível em: <http://www.php.net/manual/pt_BR>.

POSTGRESQL. *Introdução e Histórico*. 2003. Acesso em 25 de fevereiro de 2011. Disponível em: <http://wiki.postgresql.org/wiki/Introdu%C3%A7%C3%A3o_e_Hist%C3%B3rico>.

SYMFONY. *Plugins: sfGuardPlugin*. 2010. Acesso em 27 de fevereiro de 2011. Disponível em: <http://www.symfony-project.org/plugins/sfGuardPlugin>.

WIKIPEDIA. *PHP*. 2011. Acesso em 24 de fevereiro de 2011. Disponível em: <http://en.wikipedia.org/wiki/PHP>.

WIKIPEDIA. *Wikipedia:FAQ/Technical - What software is used to run Wikipedia?* 2011. Acesso em 24 de fevereiro de 2011. Disponível em: <http://en.wikipedia.org/wiki/Wikipedia:FAQ/Technical#What_software_is_used_to_run_Wikipedia.3F>.

WORDPRESS. *About Wordpress*. 2011. Acesso em 24 de fevereiro de 2011. Disponível em: <http://wordpress.org/about/>.

6 APÊNDICE

6.1 Casos de uso

6.1.1 Manter Professores

Atores	Administrador
Pré-condições	O administrador deve estar logado no sistema.
Fluxo básico	1. O caso de uso se inicia quando o administrador seleciona manter professores no menu do sistema. 2. Uma vez que o administrador seleciona uma das opções disponíveis (incluir, alterar, excluir, listar): a) Se o administrador selecionar a opção incluir, o caso de uso segue para o sub-fluxo 2 - Incluir Professor. b) Se o administrador selecionar a opção alterar, o caso de uso segue para o sub-fluxo 3 - Alterar Professor. c) Se o administrador selecionar a opção excluir, o caso de uso segue para o sub-fluxo 4 - Excluir Professor. d) Se o administrador selecionar a opção listar, o caso de uso segue para o sub-fluxo 5 - Listar Professores. 3. O caso de uso se encerra.
Incluir Professor	1. Este sub-fluxo se inicia quando o administrador seleciona incluir um novo professor. 2. O sistema exibe os seguintes campos (os campos com asterisco são obrigatórios): * Nome de usuário * Email * Senha * Confirmação de senha Nome Sobrenome * Instituição * Titulação Experiência Substituto - Campo de escolha única fechada (valores: sim, não) Comissão - Campo de escolha única fechada (valores: sim, não) Ativo - Campo de escolha única fechada (valores: sim, não) Superusuário - Campo de escolha única fechada (valores: sim, não) 3. O administrador preenche os campos e seleciona a opção salvar. 4. O sistema valida se os campos obrigatórios foram preenchidos. 5. O sistema inclui o professor no banco de dados. 6. O caso de uso se encerra.

Alterar Professor	Pré-condições: O administrador deve ter selecionado um professor para a alteração. 1. Este sub-fluxo se inicia quando o administrador seleciona alterar professor. 2. O sistema exibe os campos preenchidos. 3. O administrador altera os dados e solicita salvar os dados. 4. O sistema valida se os campos obrigatórios foram preenchidos. 5. O sistema salva as alterações no banco de dados. 6. O caso de uso se encerra.
Excluir Professor	Pré-condições: O administrador deve ter seleciona umdo professor para a exclusão. 1. Este sub-fluxo se inicia quando o administrador seleciona excluir professor. 2. O sistema solicita que o administrador confirme a exclusão. 3. O adminstrador confirma a mensagem. 4. O sistema exclui o professor do banco de dados. 5. O caso de uso se encerra.
Listar Professores	1. Este sub-fluxo se inicia quando o administrador seleciona listar professores. 2. O sistema exibe a listagem dos professores, contendo os seguintes campos: a) Nome b) Sobrenome c) Nome de usuário d) Email 3. O caso de uso se encerra.
Fluxos alternativos	Dados obrigatórios não preenchidos 1. Este sub-fluxo se inicia no passo 4 dos sub-fluxos Incluir Professor e Alterar Professor, quando o usuario não informou todos os campos obrigatórios. 2. O sistema exibe ao lado do campo uma mensagem de que o campo deve ser preenchido e aguarda até que o administrador o preencha. 3. O subfluxo segue para o passo 2 do sub-fluxo do qual ele se originou.

6.1.2 Manter Estudantes

Atores	Administrador
Pré-condições	O administrador deve estar logado no sistema.
Fluxo básico	1. O caso de uso se inicia quando o administrador seleciona manter estudantes no menu do sistema. 2. Uma vez que o administrador seleciona uma das opções disponíveis (incluir, alterar, excluir, listar): a) Se o administrador selecionar a opção incluir, o caso de uso segue para o sub-fluxo 2 - Incluir Estudante. b) Se o administrador selecionar a opção alterar, o caso de uso segue para o sub-fluxo 3 - Alterar Estudante. c) Se o administrador selecionar a opção excluir, o caso de uso segue para o sub-fluxo 4 - Excluir Estudante. d) Se o administrador selecionar a opção listar, o caso de uso segue para o sub-fluxo 5 - Listar Estudantes. 3. O caso de uso se encerra.
Incluir Estudante	1. Este sub-fluxo se inicia quando o administrador seleciona incluir um novo estudante. 2. O sistema exibe os seguintes campos (os campos com asterisco são obrigatórios): * Matrícula * Email * Senha * Confirmação de senha Nome Sobrenome Telefone Ativo - Campo de escolha única fechada (valores: sim, não) 3. O administrador preenche os campos e seleciona a opção salvar. 4. O sistema valida se os campos obrigatórios foram preenchidos. 5. O sistema inclui o estudante no banco de dados. 6. O caso de uso se encerra.

Alterar Estudante	Pré-condições: O administrador deve ter selecionado um estudante para a alteração. 1. Este sub-fluxo se inicia quando o administrador seleciona alterar estudante. 2. O sistema exibe os campos preenchidos. 3. O administrador altera os dados e solicita salvar os dados. 4. O sistema valida se os campos obrigatórios foram preenchidos. 5. O sistema salva as alterações no banco de dados. 6. O caso de uso se encerra.
Excluir Estudante	Pré-condições: O administrador deve ter seleciona umdo estudante para a exclusão. 1. Este sub-fluxo se inicia quando o administrador seleciona excluir estudante. 2. O sistema solicita que o administrador confirme a exclusão. 3. O adminstrador confirma a mensagem. 4. O sistema exclui o estudante do banco de dados. 5. O caso de uso se encerra.
Listar Estudantes	1. Este sub-fluxo se inicia quando o administrador seleciona listar estudantes. 2. O sistema exibe a listagem dos estudantes, contendo os seguintes campos: a) Nome b) Sobrenome c) Matrícula d) Telefone e) Email 3. O caso de uso se encerra.
Fluxos alternativos	Dados obrigatórios não preenchidos 1. Este sub-fluxo se inicia no passo 4 dos sub-fluxos Incluir Estudante e Alterar Estudante, quando o usuario não informou todos os campos obrigatórios. 2. O sistema exibe ao lado do campo uma mensagem de que o campo deve ser preenchido e aguarda até que o administrador o preencha. 3. O subfluxo segue para o passo 2 do sub-fluxo do qual ele se originou.

6.1.3 Manter Projetos

Atores	Estudante
Pré-condições	O estudante deve estar logado no sistema.
Fluxo básico	1. O caso de uso se inicia quando o estudante seleciona manter projetos no menu do sistema. 2. Uma vez que o estudante seleciona uma das opções disponíveis (incluir, alterar, excluir, listar, visualizar comentários): a) Se o estudante selecionar a opção incluir, o caso de uso segue para o sub-fluxo 2 - Incluir Projeto. b) Se o estudante selecionar a opção alterar, o caso de uso segue para o sub-fluxo 3 - Alterar Projeto. c) Se o estudante selecionar a opção excluir, o caso de uso segue para o sub-fluxo 4 - Excluir Projeto. d) Se o estudante selecionar a opção listar, o caso de uso segue para o sub-fluxo 5 - Listar Projetos. e) Se o estudante selecionar a opção visualizar comentários, o caso de uso segue para o sub-fluxo 6 - Visualizar Comentarios. 3. O caso de uso se encerra.
Incluir Projeto	1. Este sub-fluxo se inicia quando o estudante seleciona incluir um novo projeto. 2. O sistema exibe os seguintes campos (os campos com asterisco são obrigatórios): * Titulo * Orientador * Coorientadores 3. O estudante preenche os campos e seleciona a opção salvar. 4. O sistema valida se os campos obrigatórios foram preenchidos. 5. O sistema inclui o projeto no banco de dados. 6. O caso de uso se encerra.
Alterar Projeto	Pré-condições: O estudante deve ter selecionado um projeto para a alteração. 1. Este sub-fluxo se inicia quando o estudante seleciona alterar projeto. 2. O sistema exibe os campos preenchidos. 3. O estudante altera os dados e solicita salvar os dados. 4. O sistema valida se os campos obrigatórios foram preenchidos. 5. O sistema salva as alterações no banco de dados. 6. O caso de uso se encerra.

Excluir Projeto	Pré-condições: O estudante deve ter selecionado um projeto para a exclusão. 1. Este sub-fluxo se inicia quando o estudante seleciona excluir projeto. 2. O sistema solicita que o projeto confirme a exclusão. 3. O estudante confirma a mensagem. 4. O sistema exclui o projeto do banco de dados. 5. O caso de uso se encerra.
Listar Projetos	1. Este sub-fluxo se inicia quando o estudante seleciona listar projetos. 2. O sistema exibe a listagem dos projetos, contendo os seguintes campos: a) Orientador b) Título c) Proposta d) Defesa e) Status - Status mais atual da defesa, ou se esta não tiver sido iniciada, o status mais atual da proposta. 3. O caso de uso se encerra.
Visualizar Comentários	1. Este sub-fluxo se inicia quando o estudante solicita visualizar os comentários. 2. Uma vez que o estudante seleciona uma das opções disponíveis (comentários do orientador, da comissão): a) Se o estudante selecionar visualizar os comentários do orientador, o sistema exibe uma listagem com os comentários do orientador. b) Se o estudante selecionar visualizar os comentários da comissão, o sistema exibe uma listagem com os comentários da comissão. 3. O caso de uso se encerra.
Fluxos alternativos	Dados obrigatórios não preenchidos 1. Este sub-fluxo se inicia no passo 4 dos sub-fluxos Incluir Estudante e Alterar Estudante, quando o usuario não informou todos os campos obrigatórios. 2. O sistema exibe ao lado do campo uma mensagem de que o campo deve ser preenchido e aguarda até que o administrador o preencha. 3. O subfluxo segue para o passo 2 do sub-fluxo do qual ele se originou.

6.1.4 Manter Propostas

Atores	Estudante, Orientador, Comissão
Pré-condições	O estudante deve estar logado no sistema e ter selecionado um projeto. O orientador deve estar logado no sistema. A comissão deve estar logada no sistema.
Fluxo básico	1. O caso de uso se inicia quando o estudante seleciona a opção Anexar Proposta para o projeto selecionado. 2. O sistema exibe o campo Documento 3. O estudante escolhe um arquivo PDF e seleciona a opção Salvar. 4. O sistema valida o formato e tamanho do arquivo. 5. O sistema anexa o documento ao projeto, exibe uma mensagem de sucesso e redireciona o estudante à listagem de projetos. 6. O sistema envia um email ao orientador, informando do envio de uma nova proposta por um de seus orientandos. 7. O orientador seleciona a opção Visualiza Proposta, do projeto em questão. 8. O sistema exibe a proposta e solicita ao orientador para que ele seleciona uma das opções disponíveis (Aprovar/desaprovar) 9. O orientador seleciona a opção aprovar. 10. O sistema envia um email à comissão, informando do envio de uma nova proposta e atualiza o status da proposta para ”Aprovada pelo orientador”. 11. A comissão seleciona a opção Visualizar Proposta, do projeto em questão. 12. O sistema exibe a proposta e um campo de comentários e solicita à comissão para que ela selecione uma das opções disponíveis (Aprovar/desaprovar) 13. A comissão comenta (opcionalmente) na proposta e a aprova. 14. O sistema envia um email ao orientador e ao estudante, informando de que a proposta foi aprovada e atualiza o status da proposta para ”Aprovada pela comissão”. 15. O caso de uso se encerra.

Fluxos alternativos	Formato e/ou tamanho inválidos 1. O subfluxo se inicia no passo 4 do fluxo básico, quando o sistema detecta que o formato e/ou o tamanho do arquivo são inválidos. 2. O sistema informa ao usuário dos dados inválidos e solicita-o que os corrija. 3. O subfluxo segue para o passo 2 do fluxo básico. Desaprovação da proposta 1. O subfluxo se inicia no passo 9 do fluxo básico, quando o usuário for o orientador, ou no passo 13 do fluxo básico, quando o usuário for da comissão. O usuário selecionou a reprovar. 2. O sistema envia um email ao estudante (e ao orientador, caso a proposta tenha sido reprovada pela comissão), informando que sua proposta foi reprovada. 3. O sistema atualiza o status da proposta para ”Reprovada pelo orientador”ou ”Reprovada pela comissão”, dependendo de qual usuário tenha reprovado a proposta. 4. O caso de uso se encerra.

6.1.5 Manter Defesas

Atores	Estudante, Orientador, Comissão
Pré-condições	O estudante deve estar logado no sistema e ter selecionado um projeto. O orientador deve estar logado no sistema. A comissão deve estar logada no sistema.
Fluxo básico	1. O caso de uso se inicia quando o estudante seleciona a opção Solicitar Defesa para o projeto selecionado. 2. O sistema exibe os seguintes campos: a) Quantidade de páginas da monografia b) Data sugerida para realização da defesa c) Copião (arquivo PDF) 3. O estudante preenche os campos e seleciona a opção Salvar. 4. O sistema valida os campos. 5. O sistema anexa o copião ao projeto, exibe uma mensagem de sucesso e redireciona o estudante à listagem de projetos. 6. O sistema envia um email ao orientador, informando do envio da solicitação de defesa por um de seus orientandos. 7. O orientador seleciona a opção Solicitação de Defesa, do projeto em questão. 8. O sistema exibe o copião e solicita ao orientador para que ele selecione uma das opções disponíveis (Aprovar/desaprovar) 9. O orientador seleciona a opção aprovar. 10. O sistema envia um email à comissão, informando do envio de uma nova solicitação de defesa e atualiza o status do projeto para ”Defesa aprovada pelo orientador”. 11. A comissão seleciona a opção Solicitação de Defesa, do projeto em questão. 12. O sistema exibe os seguintes campos: a) Copião (arquivo para download) b) Comentários c) Data autorizada para realização da defesa 13. O sistema solicita à comissão para que ela selecione uma das opções disponíveis (Aprovar/Desaprovar) 14. A comissão comenta (opcionalmente) na solicitação, preenche a data de realização da defesa e a aprova. 15. O sistema envia um email ao orientador e ao estudante, informando de que a solicitação foi aprovada e atualiza o status do projeto para ”Defesa aprovada pela comissão”. 16. O caso de uso se encerra.

Fluxos alternativos	Formato e/ou tamanho inválidos 1. O subfluxo se inicia no passo 4 do fluxo básico, quando o sistema detecta que o formato e/ou o tamanho do arquivo são inválidos. 2. O sistema informa ao usuário dos dados inválidos e solicita-o que os corrija. 3. O subfluxo segue para o passo 2 do fluxo básico. Desaprovação da proposta 1. O subfluxo se inicia no passo 9 do fluxo básico, quando o usuário for o orientador, ou no passo 13 do fluxo básico, quando o usuário for da comissão. O usuário selecionou a reprovar. 2. O sistema envia um email ao estudante (e ao orientador, caso a proposta tenha sido reprovada pela comissão), informando que sua proposta foi reprovada. 3. O sistema atualiza o status da proposta para ”Reprovada pelo orientador”ou ”Reprovada pela comissão”, dependendo de qual usuário tenha reprovado a proposta. 4. O caso de uso se encerra.

7 ANEXOS

7.1 Formulário de proposta de projeto final

UNIVERSIDADE ESTADUAL DO CEARÁ
CENTRO DE CIÊNCIAS E TECNOLOGIA
CURSO BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

PROPOSTA DE PROJETO FINAL

Nome : _____

Matrícula: _____

Tel: (____) _____ E-mail: _____

Orientador: _____

Titulo do Trabalho: _____

_____

_____ _____
Local e Data Assinatura do Aluno

_____
Assinatura do Orientador

Parecer da Comissão de Projeto Final

☐ Proposta Aprovada ☐ Proposta Reprovada

Comentários:_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____ _____
Local e Data Nome:

_____
Nome:

_____
Nome:

IMPORTANTE: Anexar Proposta de Projeto Final conforme Modelo

Formulário Proposta de Projeto Final Versão 1.0 – julho/2007

7.2 Formulário de solicitação de defesa e banca

UNIVERSIDADE ESTADUAL DO CEARÁ
CENTRO DE CIÊNCIAS E TECNOLOGIA
CURSO BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

SOLICITAÇÃO DE DEFESA E INDICAÇÃO DA COMISSÃO EXAMINADORA DE PROJETO FINAL

Nome : _____

Matrícula: _____

Tel: (____) _____ E-mail: _____

Orientador: _____

Titulo do Trabalho: _____

_____

_____

Quantidade de páginas da monografia: _____

Indicação da Comissão Examinadora:

1)Nome: _____

Instituição: _____

Titulação: _____ Experiência: _____ anos

2)Nome: _____

Instituição: _____

Titulação: _____ Experiência: _____ anos

Data sugerida para realização da defesa: ____ / _____ / _____ ____ : _____ horas

_____ _____

Local e Data Assinatura do Aluno

_____

Assinatura do Orientador

IMPORTANTE: Anexar copião do texto de monografia.

Formulário Solicitação de Defesa e Indicação de Comissão Examnadora de projeto Final Versão 1.0 – julho/2007