



# **FERRAMENTA PARA GESTÃO DA DISCIPLINA DE PROJETO DE FINAL DE CURSO**

**Trabalho de Conclusão de Curso**

**Engenharia da Computação**

**Caio César Bernardes Lacerda Lima**  
**Orientador: Prof. Sérgio Campello**



**Universidade de Pernambuco  
Escola Politécnica de Pernambuco  
Graduação em Engenharia de Computação**

**Caio César Bernardes Lacerda**

**FERRAMENTA PARA GESTÃO DA  
DISCIPLINA DE PROJETO DE FINAL  
DE CURSO**

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

**Recife, Dezembro de 2012.**

### MONOGRAFIA DE FINAL DE CURSO

#### Avaliação Final (para o presidente da banca)\*

No dia 14 de 12 de 2012, às 9:00 horas, reuniu-se para deliberar a defesa da monografia de conclusão de curso do discente CAIO CESAR BERNARDES LACERDA LIMA, orientado pelo professor Sérgio Campello Oliveira, sob título FERRAMENTA PARA GESTÃO DA DISCIPLINA DE PROJETO DE FINAL DE CURSO, a banca composta pelos professores:

**José Paulo G. de Oliveira**

**Sérgio Campello Oliveira**

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

☒ Aprovada

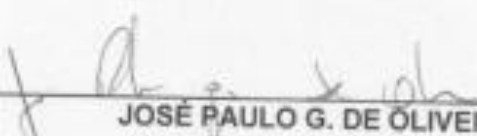
☐ Aprovada com Restrições\*

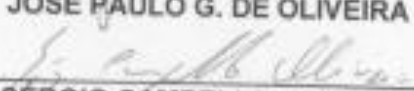
☐ Reprovada

e foi-lhe atribuída nota: 9,00 ( NOVE )

\*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O discente terá 7 dias para entrega da versão final da monografia a contar da data deste documento.

  
\_\_\_\_\_  
**JOSÉ PAULO G. DE OLIVEIRA**

  
\_\_\_\_\_  
**SÉRGIO CAMPELLO OLIVEIRA**

\* Este documento deverá ser encadernado juntamente com a monografia em versão final.

*Dedico a minha família.*

# Agradecimentos

Agradeço as pessoas que sempre estiveram comigo em todas as etapas deste trabalho, minha família, Lana (minha mãe), e minha noiva Sâmia, que sempre acreditaram em mim, “apesar dos pesares”.

Agradeço muito ao meu orientador Sergio Campello, que não titubeou quando pedi para ser meu orientador e espero levar sua amizade comigo para sempre.

# Resumo

A disciplina de Projeto de Final de Curso (PFC), é uma disciplina diferenciada para todo curso superior, por ser a disciplina que envolve alunos, professores, orientadores, professores examinadores e um coordenador, logo há um problema inerente de gestão da disciplina. Existem várias ferramentas para auxílio a educação, seja ela ensino a distancia ou gerenciamento de cursos, porém falta ao curso de Engenharia de computação (ECOMP) – da Escola Politécnica de Pernambuco (POLI) da Universidade de Pernambuco (UPE) uma ferramenta que auxilie o professor da disciplina PFC nos seus processos e tarefas. Este trabalho desenvolveu uma ferramenta voltada para esta gestão da disciplina PFC. São três usuários da ferramenta: professor da disciplina, orientador e aluno. O professor da disciplina se comporta como o administrador do sistema. O professor orientador pode ver todos os orientandos, podendo aprovar ou reprovar seus textos (projeto e monografia), além de ter uma área onde pode avaliar alunos, de cuja banca examinadora ele faz parte. O aluno pode enviar seu projeto e sua monografia. Todas essas tarefas foram retiradas das normas da disciplina do ECOMP. Esta ferramenta buscou minimizar o tempo gasto com questões gerenciais da disciplina, deixando alunos e professores focados no conteúdo de seus trabalhos.

# Abstract

The discipline of Project Ending of Course (PFC), is a distinct discipline for any college, being the discipline with the student will take what he learned in all the graduation. There are several tools to aid education, as if distance learning or course management, but lack the ECOMP - POLI University of Pernambuco a tool to assist the subject teacher PFC in their processes and tasks. This work developed a tool focused on this management. Three users of the tool: the discipline teacher, mentor and student. The subject teacher behaves as the system administrator. The supervising teacher can see every student advisees, and may approve or disapprove their texts (Project and monography), besides having an area where students can assess in which part of the committee. Students can send their projects and monographies. All these tasks were taken from the norms of discipline ECOMP. This tool aimed to improve and optimize the time spent on managerial issues of discipline, leaving students and teachers focused on the content of their work.

# Sumário

<b>Capítulo 1 Introdução</b>	<b>1</b>
1.1 Comportamento da ferramenta	2
1.2 Objetivos	2
1.3 Resultados	3
1.4 Estruturação do trabalho	3
<b>Capítulo 2 Metodologias de Desenvolvimento de Software</b>	<b>4</b>
2.1 Visão Geral das etapas de um desenvolvimento de software	4
2.1.1 Levantamento de requisitos	4
2.1.2 Análise de requisitos	5
2.1.3 Projeto	5
2.1.4 Desenvolvimento	6
2.1.5 Testes	6
2.2 Metodologia de Desenvolvimento Ágil	7
2.2.1 Scrum	8
2.2.2 Extreme Programming (XP)	9
2.3 Ferramentas análogas	11
2.3.1 SIG@	11
2.3.2 Moodle	11
<b>Capítulo 3 Normas e requisitos de TCC</b>	<b>13</b>
3.1 Tarefas do professor da disciplina PFC	13



3.2	Tarefas do professor orientador	14
3.3	Tarefas do professor avaliador	14
3.4	Atividades	14
3.5	Atribuição da nota	15
3.6	Banca de avaliação	15
<b>Capítulo 4 Ferramenta de auxílio à gerência da disciplina de Projeto de Final de Curso</b>		<b>17</b>
4.1	Levantamento de Requisitos	18
4.2	Análise de requisitos e Projeto do software	18
4.2.1	Protótipos de tela	18
4.2.2	Diagramas de Casos de Uso UML e suas especificações	21
4.2.3	Diagramas de Classes UML	22
4.2.4	Modelo ER e modelo lógico	26
4.3	Desenvolvimento	28
4.3.1	Linguagem de Programação	28
4.3.2	Estruturação	28
4.4	Funcionamento do sistema	29
4.4.1	Tela de Login	30
4.4.2	Tela do professor orientador/avaliador	30
4.4.3	Tela do Aluno	32
4.4.4	Tela do Professor da disciplina	34
<b>Capítulo 5 Resultados e Testes</b>		<b>36</b>

5.1	Modelagem do Sistema	36
5.1.1	Diagramas de Casos de Uso	36
5.1.2	Especificações de Casos de Uso	38
<b>Capítulo 6 Conclusão e Trabalhos Futuros</b>		<b>46</b>
6.1	Conclusão	46
6.2	Trabalhos Futuros	46
<b>Bibliografia</b>		<b>48</b>
<b>Apêndice A Diagrama de Classe completo</b>		<b>50</b>
<b>Apêndice B Exemplos de código</b>		<b>51</b>

# Índice de Figuras

<b>Figura 1.</b>	Protótipo Professor Disiciplina.....	19
<b>Figura 2.</b>	Protótipo Aluno.....	20
<b>Figura 3.</b>	Protótipo Professor Orientador/Avaliador .....	21
<b>Figura 4.</b>	Exemplo Caso de Uso.....	21
<b>Figura 5.</b>	Diagrama Classe data .....	23
<b>Figura 6.</b>	Diagrama classe service .....	24
<b>Figura 7.</b>	Diagrama classe bussiness.....	25
<b>Figura 8.</b>	Diagrama classe gui.....	25
<b>Figura 9.</b>	Modelo Entidade-Relacionamento.....	27
<b>Figura 10.</b>	Modelo Lógico .....	27
<b>Figura 11.</b>	Tela Login.....	30
<b>Figura 12.</b>	Dados Gerais Professor Orientador .....	31
<b>Figura 13.</b>	Tela Orientações Professor Orientador .....	32
<b>Figura 14.</b>	Tela Avaliação Professor Orientador.....	32
<b>Figura 15.</b>	Tela Dados Pessoais Aluno .....	33
<b>Figura 16.</b>	Tela Projeto e Monografia Aluno .....	34
<b>Figura 17.</b>	Tela Inserir Usuário Professor da disciplina .....	34
<b>Figura 18.</b>	Tela Planilha TCC .....	35
<b>Figura 19.</b>	Caso de Uso Professor Orientador.....	37

<b>Figura 20.</b>	Caso de Uso Aluno.....	37
<b>Figura 21.</b>	Caso de Uso Professor Disciplina .....	38
<b>Figura 22.</b>	Diagrama de Classe completo.....	50

# Índice de Tabelas

<b>Tabela 1.</b> Pontuação do TCC .....	13
---	----

# Tabela de Símbolos e Siglas

ECOMP – Engenharia da Computação

ER – Entidade-Relacionamento

Moodle - *Modular Object-Oriented Dynamic Learning Environment*

PFC – Projeto de final de Curso

POLI – Escola Politécnica de Pernambuco

SIG@ - Sistema de Informações e Gestão Acadêmica

TCC –Trabalho de Conclusão de Curso

UC – Caso de Uso

UML – Unified Modeling Language

UPE – Universidade de Pernambuco

# Capítulo 1

## Introdução

A tecnologia aplicada à educação conceitualmente é a utilização de recursos e materiais com o objetivo de melhorar e facilitar o trabalho do professor de uma disciplina, dentro ou fora da sala de aula. A gestão de cursos e disciplinas nem sempre se dá de forma organizada, com isso abriu-se caminho para ferramentas que possam auxiliar e organizar esta gestão, comumente chamadas de Sistema de Gestão de Aprendizagem [1].

Um Sistema de Gestão de Aprendizagem disponibiliza um conjunto de funcionalidades projetadas para armazenar, distribuir e gerenciar conteúdos relativos à aprendizagem [1].

Dentre estes sistemas, dois se destacam e são discutidos e demonstrados neste trabalho: O SIG@ [2] e o Moodle [3].

O SIGA é um portal digital criado para o uso de estudantes e professores universitários, onde os mesmos poderão conferir informações sobre o curso, horários de aula e provas, editais, concursos, seminários, informativos entre outros.

O Moodle é um software livre de apoio à aprendizagem. Ele permite a criação de cursos online, páginas para disciplinas contendo materiais disponibilizados pelo professor, grupos de trabalho e de estudo. Ainda é possível fazer avaliações e testes através da ferramenta.

Este trabalho desenvolveu uma ferramenta voltada para a gestão dos processos da disciplina de Trabalho de Conclusão de Curso (TCC) do curso de Engenharia da Computação da Universidade de Pernambuco (UPE) de acordo com o documento de normas da disciplina [4].

## 1.1 Comportamento da ferramenta

A ferramenta possibilita o acesso de três usuários distintos: o professor da disciplina, alunos e professores em geral.

O professor da disciplina se comporta como o administrador do sistema, ou seja, é o usuário que detém o poder para executar todos os processos da ferramenta, sendo exclusivo a ele cadastrar alunos e professores, inicializar senhas de acesso, organizar os participantes das bancas das apresentações e gerar as declarações necessárias para cada trabalho apresentado.

Os alunos podem realizar upload nos arquivos de concordância de orientação, projeto e monografia. Ao enviar os arquivos, o professor orientador receberá um aviso por email podendo validar ou invalidar aquele documento enviado.

Os professores em geral têm a visão de todos seus alunos orientandos e de suas bancas avaliadoras já agendadas. Dos alunos de cuja banca será presidente, poderá inserir as notas no sistema após a apresentação do trabalho. Dos alunos orientados poderá validar ou invalidar os documentos enviados.

## 1.2 Objetivos

Este trabalho teve como objetivo principal desenvolver uma ferramenta que dê suporte a todos os processos que envolvem a gestão da disciplina de Trabalho de Conclusão de Curso (TCC) do curso de engenharia de computação do ECOMP/UPE.

O objetivo geral desdobrou-se nos seguintes objetivos específicos:

- Estudo dos conceitos dos processos que envolvem a disciplina;
- Estudo e desenvolvimento do projeto da ferramenta;
- Desenvolvimento dos algoritmos da ferramenta;



- Desenvolvimento dos exemplos de uso da ferramenta e testes.

## 1.3 Resultados

Este trabalho teve como resultado uma ferramenta de gestão de TCC, onde será possível a cada indivíduo responsável por uma parte do processo, seja ele o professor da cadeira, o aluno, professor orientador ou presidente da banca, poder interagir com o sistema a fim de completar um procedimento específico. Além de possibilitar ao professor da disciplina gerar as declarações necessárias: declaração de orientação, de co-orientação e de participação em banca, de forma automatizada para cada trabalho apresentado.

O desenvolvimento da ferramenta se deu de maneira mais genérica possível para possibilitar seu uso futuro em outros cursos da POLI e da UPE.

## 1.4 Estruturação do trabalho

O presente trabalho é estruturado da seguinte maneira: O Capítulo 2 apresenta uma visão geral sobre os assuntos teóricos necessários para o entendimento completo do trabalho. Uma revisão dos conceitos sobre metodologias de desenvolvimento é explicada e duas ferramentas de gestão para o ensino, o SIG@ e o Moodle. O capítulo 3 está voltado para uma revisão bibliográfica, onde serão apresentadas as normas da disciplina TCC do curso de Engenharia da Computação da UPE. O capítulo 4 trata do desenvolvimento da ferramenta, onde será mostrado toda a metodologia e a estratégia de ação utilizadas, mostrando protótipos de tela, fluxogramas, trechos de código e funcionalidades do sistema. O capítulo 5 mostra os resultados obtidos na criação do software, os diagramas, especificações, exemplos de uso e testes. Por fim, o capítulo 6 apresenta as conclusões e trabalhos futuros.

# Capítulo 2

## Metodologias de Desenvolvimento de Software

### 2.1 Visão Geral das etapas de um desenvolvimento de software

Existem várias abordagens diferentes para o desenvolvimento de software, alguns desenvolvedores tomam uma forma mais estruturada, ou seja, o software é criado a partir de uma solução de negócio também em desenvolvimento, enquanto outros podem ter uma abordagem mais incremental, ou seja, onde o software evolui a cada passagem de tempo [5].

A maioria das metodologias apresenta, de forma geral, uma combinação das seguintes fases de desenvolvimento de software: Levantamento de requisitos; Análise de Requisitos; Projeto; Desenvolvimento; Testes; Implantação; Manutenção e correção de bugs [5].

Das fases apresentadas, apenas uma não foi executada neste trabalho, a de manutenção e correção de *bug*, uma vez que seu desenvolvimento terminou na implantação do sistema.

#### 2.1.1 Levantamento de requisitos

Esta etapa tem como objetivo uma compreensão do problema proposto, onde o cliente e os desenvolvedores buscam encontrar as necessidades que os usuários do software deverão ter. Estas necessidades são chamadas requisitos [5].

Em outras palavras, este processo basicamente consiste em identificar e detalhar o que deve ser feito do ponto de vista de negócios e recursos em um

determinado sistema. Pode-se entender requisito como “uma coisa que o sistema deve fazer”.

Um requisito pode ser uma função ou tarefa que o software deva empenhar, ou seja, estes requisitos buscam especificar resultados particulares do sistema. Este requisito é conhecido como funcional [5].

Um requisito também pode especificar uma característica geral, que não é, necessariamente, uma função do software, como restrição de custo, confiabilidade, segurança. Este requisito é conhecido como não-funcional [5].

### **2.1.2 Análise de requisitos**

Após o levantamento dos requisitos, os desenvolvedores devem estudar detalhadamente os dados encontrados, para então criar modelos que representem o software especificado. Com isso, consegue-se definir o que o sistema deve fazer, mesmo antes de definir como será feito [5].

O objetivo da análise de requisitos é a produção de um documento, comumente chamado de Especificação do software, que registra os resultados encontrados e os modelos criados.

De acordo com [6], “uma Análise de Requisitos bem sucedida deve, normalmente, representar corretamente as necessidades do cliente e dos usuários, satisfazendo, porém, às três partes envolvidas (cliente, usuário e desenvolvedor)”. O que muitas vezes é encontrado nos projetos de software é que nem sempre o cliente entende quais são suas reais necessidades. Da mesma maneira, usuários tem certa dificuldade também de se expressar quanto ao software que está em desenvolvimento. Por isso é necessário neste momento do desenvolvimento o esclarecimento do que é necessário e esperado [6].

### **2.1.3 Projeto**

A etapa de Projeto consiste na aplicação de técnicas, buscando um detalhamento do sistema que seja suficiente para a sua realização. O desenvolvedor responsável pelo projeto é chamado projetista [6].

A tarefa do projetista nada mais é do que produzir um modelo que represente o software a ser desenvolvido. Nesta etapa, os requisitos definidos na etapa anterior devem servir de referência para a obtenção da representação do software [6].

Esta fase é conduzida, de acordo com [6], por dois principais estágios:

- **Projeto preliminar**, onde, a partir dos requisitos, a arquitetura do sistema é estabelecida;
- **Projeto detalhado**, onde a estruturação do que será o software é aperfeiçoada e é definida qual linguagem de programação deve ser utilizada.

No fim desta fase, um documento de projeto de software deve ser criado. Este documento deve conter uma descrição completa do software, mostrando os requisitos e suas representações (modelos), linguagem de programação e a arquitetura do sistema.

#### **2.1.4 Desenvolvimento**

Com o documento do projeto em mãos, os desenvolvedores começam a codificação do que foi proposto, a fim de gerar um executável para o software.

Nesta etapa, objetos, classes e estruturas de dados são definidos e criados para conseguir alcançar o que foi estipulado pelos requisitos funcionais e não-funcionais. É interessante, também, o uso de bibliotecas preexistentes e a utilização de algumas ferramentas complementares buscando a agilidade no desenvolvimento.

#### **2.1.5 Testes**

Inicialmente, apenas o desenvolvimento de software utilizando metodologias, técnicas e ferramentas não oferece a garantia da qualidade do produto final. Por esta razão, é necessária esta etapa de Testes, uma vez que ela é fundamental na

obtenção de um alto nível de qualidade do software a ser produzido e por ser a última etapa de revisão da especificação, do projeto e da codificação [6].

O objetivo principal desta fase é o de encontrar erro, e pode ser descrito através das três regras [7]:

- Executar o programa com a intenção de descobrir erros;
- Um bom caso de teste é aquele que há a maior probabilidade de revelar um erro não descoberto;
- Para um teste ser bem sucedido, ele deve revelar um erro não descoberto.

O teste utilizado neste trabalho foi o teste de Caixa-Preta. Este teste refere-se a todo o teste que busca verificar o funcionamento do software, através de suas interfaces, o que permite checar a operacionalidade de suas funções. É importante salientar que este tipo de teste não importa como está organizado internamente o software e sim suas funcionalidades[6].

## **2.2 Metodologia de Desenvolvimento Ágil**

As metodologias ágeis buscam minimizar o risco de um desenvolvimento do software utilizando desenvolvimento em curtos períodos, chamados de iteração. Cada iteração pode ser vista como um pequeno projeto de software, e busca incluir uma funcionalidade ou conjunto de funcionalidades ao programa em desenvolvimento. Passa por todos os processos descritos na seção anterior, mas pode conter algumas modificações. Um desenvolvimento de software ágil busca implantar uma nova versão do software ao fim de cada iteração, e então prioridades são revistas para então passar a próxima iteração.

A idéia é garantir a comunicação da equipe de desenvolvimento em si e com o cliente seja em tempo real. Por isso, a produção de documentação, se comparado a outros métodos tradicionais de desenvolvimento de software, pode ser considerada “pobre” [8].

De acordo com [8], os princípios do desenvolvimento de software são:

- Satisfação do cliente através de entregas contínuas e funcionais do programa;
- Mudanças após a definição do escopo podem ocorrer e são até bem vistas;
- Comunicação constante entre a parte de negócios e os desenvolvedores;
- Simplicidade;
- Prezar pela excelência técnica do design;
- Indivíduos e interações ao invés de processos e ferramentas;
- Mais software do que documentação;
- Colaboração do cliente intensa;
- Fazer a parte para chegar ao todo.

Existem duas metodologias ágeis mais utilizadas pelos desenvolvedores e mais abrangentes na literatura, são elas Scrum e *Extreme Programming* (XP).

### 2.2.1 Scrum

Tem como principal objetivo fornecer um processo conveniente para projeto e desenvolvimento orientado a objeto. Apresenta uma abordagem empírica na qual busca inserir idéias de controle de processos industriais para o desenvolvimento de softwares, reintroduzindo as idéias de flexibilidade, adaptabilidade e produtividade. O foco desta metodologia é encontrar uma forma de trabalho dos membros da equipe para produzir o software de forma flexível e em um ambiente em constante mudança [9].

A idéia principal do Scrum é que o desenvolvimento de softwares é imprevisível e complexo, por apresentar muitas variáveis, como requisitos, recursos

e tecnologia, que podem mudar durante o processo. Com isso, é necessário uma flexibilidade para acompanhar as mudanças. O resultado do processo deve ser um software que seja realmente útil para o cliente [9].

Esta metodologia é, baseada basicamente em: pequenas equipes de desenvolvimento, requisitos instáveis ou desconhecidos no começo do projeto e iterações curtas para prover a visibilidade necessária ao desenvolvimento [10].

O ciclo de vida Scum é baseado em três fases principais:

- **Pré-planejamento:** Os requisitos são descritos, priorizados e têm seu tempo de execução estimado.
- **Desenvolvimento:** Pega-se o documento da etapa anterior e as tarefas são executadas por prioridade.
- **Pós-planejamento:** Onde um balanço é feito sobre o que foi produzido, os eventuais atrasos e um novo ciclo é iniciado.

### 2.2.2 Extreme Programming (XP)

XP é uma metodologia ágil para pequenas e médias equipes que desenvolvem softwares com requisitos vagos e que estão em constante modificação [9].

Existem, nesta metodologia, três principais características que a distingue das demais: *Feedback* constante, ser incremental e a comunicação é encorajada.

A maioria das regras da XP causa polêmica à primeira vista e muitas não fazem sentido se aplicadas isoladamente. Ela enfatiza o desenvolvimento rápido visando garantir a satisfação do cliente cumprindo suas estimativas de prazo [9].

São doze as práticas nas quais o XP se baseia [9]:

- **Planejamento:** Consiste em decidir o que é necessário ser feito num dado momento e o que pode ser adiado.

- **Entregas frequentes:** Visa a construção de versões simples do software e conforme os requisitos surgem, este software é atualizado.
- **Metáfora:** Descrição do software sem a utilização de termos técnicos, com intuito de guiar o desenvolvimento.
- **Projeto simples:** O programa desenvolvido deve ser o mais simples possível e satisfazer os requisitos atuais, sem a preocupação com futuros requisitos.
- **Testes:** O software é desenvolvido criando primeiramente os testes.
- **Refatoração:** Focaliza o aperfeiçoamento do software e presente em todas as etapas do desenvolvimento.
- **Programação em pares:** A implementação do código é feito em dupla, onde um implementa o código de fato e seu companheiro observa continuamente o trabalho do primeiro, buscando erros semânticos e sintáticos no código.
- **Propriedade coletiva:** O código do projeto pertence a todos os membros da equipe de desenvolvimento.
- **Integração contínua:** Interagir com o sistema e construí-lo várias vezes ao dia, mantendo os programadores em sintonia com o sistema.
- **40 horas semanais:** A XP assume que não se deve fazer horas extras continuamente, caso seja necessário, existe um sério problema no projeto que deve ser resolvido com um melhor planejamento.
- **Cliente sempre presente:** O cliente deve estar disponível em todas as etapas de desenvolvimento para sanar todas as dúvidas da equipe de desenvolvimento.
- **Código padrão:** Padronização na arquitetura do código seguida por todos os programadores.



## 2.3 Ferramentas análogas

Existe na literatura diversas ferramentas para a educação, seja de gestão de disciplinas e notas ou de ensino a distância. Dentre estas, existem duas ferramentas que se destacam e são amplamente utilizadas pela academia. São elas o SIG@ e o Moodle.

### 2.3.1 SIG@

O SIG@ (Sistema de Informações e Gestão Acadêmica) é um sistema fechado de acesso remoto criado pela Universidade Federal de Pernambuco e tem como objetivo a gestão de disciplinas, notas, matrículas e procedimentos diversos de uma universidade, controlando seus procedimentos acadêmicos e administrativos [2].

Toda a comunidade acadêmica da universidade que utiliza o SIG@ é usuária dela, discentes de graduação e de pós-graduação, docentes e servidores.

Os discentes podem fazer matrícula em disciplinas, podem checar quantidade de faltas, notas das disciplinas matriculadas, pode modificar seus dados pessoais, além de poder checar dados de pagamento de mensalidades. Podem, também, verificar os horários das aulas e fazer requerimentos de documentos, como declaração de vínculo, histórico escolar, entre outros.

Os docentes por sua vez, podem inserir notas e faltas dos alunos de sua disciplina, também podem modificar seus dados pessoais, e também fazer requerimentos de documentos.

### 2.3.2 Moodle

O Moodle é um software livre de apoio à aprendizagem, executado em um ambiente virtual, acessado através da internet ou até de uma rede local. Seu nome vem do inglês *Modular Object-Oriented Dynamic Learning Environment*. Utilizado num contexto de Ensino a Distância, permite a criação de cursos, páginas de disciplinas, grupos de estudo, chats, avaliações e fóruns de discussões [3].

Por ser um software livre, é possível a adaptação de seus recursos e funcionalidades para diversas instituições de ensino. Pode adaptar para um curso totalmente virtual, pode ser um complemento a cursos presenciais. Existem também, outros setores que não da educação que utilizam o Moodle, empresas privadas o utilizam para treinamentos e até para interação dos seus colaboradores [3].

## Capítulo 3

# Normas e requisitos de TCC

De acordo com [4], “o Projeto de Final de Curso (PFC) constitui requisito parcial e obrigatório para conclusão dos cursos de Engenharia da Escola Politécnica de Pernambuco, Universidade de Pernambuco. O PFC constitui etapa fundamental no processo de formação e avaliação dos alunos concluintes dos cursos, onde o aluno deverá pôr em prática o conhecimento adquirido em sua formação. O PFC possibilita ainda a investigação e eliminação de deficiências pedagógicas, em função das dificuldades encontradas pelos alunos no projeto e encaminhamento de seus respectivos trabalhos.”

O desenvolvimento de um PFC envolve a elaboração e execução de um trabalho individual pelo aluno concluinte. Para esse trabalho dá-se o nome de Trabalho de Conclusão de Curso (TCC). Esse trabalho inclui três documentos resultantes: Concordância de Orientação, Projeto e Monografia. Esse trabalho desenvolve-se ao longo de um semestre letivo [4].

### 3.1 Tarefas do professor da disciplina PFC

O professor da disciplina possui sete atribuições, das quais destacam-se para o desenvolvimento da ferramenta descrita neste trabalho [4]:

- Organizar as defesas das monografias, buscando os recursos necessários para as mesmas;
- Definir as bancas de avaliação das defesas, podendo ter algum professor recomendado pelo orientador do trabalho;
- Calcular e atribuir a nota de cada aluno.

## 3.2 Tarefas do professor orientador

O professor orientador deve orientar o aluno na elaboração de uma monografia contando que seu conteúdo esteja inserido na área de conhecimento do curso, além de ser compatível com as qualificações do orientador[4].

Diferente do professor da disciplina, o qual deve preocupar-se com os aspectos de gestão da disciplina, o orientador é responsável pelo conteúdo apresentado, sendo sua responsabilidade e do aluno garantir que esse atende às exigências de qualidade impostas [4].

Existe a possibilidade de o aluno contar com um co-orientador. Este terá responsabilidade apenas de auxiliar o professor orientador quanto aos rumos do conteúdo apresentado, porém quem garante se as exigências da disciplina foram atendidas é o professor orientador.

## 3.3 Tarefas do professor avaliador

O professor avaliador de um TCC deve participar da apresentação da monografia como presidente da banca avaliadora, e deve dar uma nota, de 0 a 10 a apresentação e ao texto escrito na monografia [4]. É importante salientar que esta nota não é a final, ainda está atrelada a entrega da concordância de orientação e do projeto no prazo.

## 3.4 Atividades

As seguintes atividades deverão ser cumpridas, no prazo dado pelo professor da disciplina, entre outras:

- Entrega da concordância de orientação;
- Entrega do projeto pelo aluno, validada pelo orientador;
- Entrega do texto da monografia pelo aluno, também validada pelo orientador;

- Submissão dos textos para os membros da banca;
- Apresentação da monografia, pelos alunos;
- Entrega da versão final do texto, por parte dos alunos;
- Apresentação;
- Elaboração e entrega dos certificados para os membros da banca.

### 3.5 Atribuição da nota

São três as atividades que valem para a nota, são elas: entrega da concordância de orientação (etapa 1), entrega do projeto (etapa 2) e apresentação da monografia (etapa 3). Cada atividade atribui uma pontuação que, somada, dará a nota final do aluno, como mostra a tabela 1.

**Tabela 1.** Pontuação do TCC

Etapa	Pontuação Máxima
1	1,0 ponto
2	1,0 ponto
3	8,0 pontos

### 3.6 Banca de avaliação

Em cada banca examinadora, deverá conter, pelo menos dois docentes. Um destes é o professor orientador do aluno, enquanto o outro será considerado o Presidente da Banca, o qual será responsável pela validação da versão final do texto e pela atribuição da nota da apresentação. O professor da disciplina deve estipular prazos para a entrega do texto a fim que haja tempo hábil para a leitura e avaliação dos mesmos pelo presidente da banca.

Podem compor a banca professores externos ao curso e à universidade e professores do curso.

O co-orientador pode substituir, caso necessário, o professor orientador na banca examinadora. Pode existir ainda um terceiro membro para a banca, sugerido pelo professor orientador e aprovado pelo professor da disciplina.

## **Capítulo 4**

# **Ferramenta de auxílio à gerência da disciplina de Projeto de Final de Curso**

A ferramenta desenvolvida utiliza uma técnica genérica de desenvolvimento ágil iterativa e incremental. Este paradigma foi utilizado por se basear na orientação a objetos e por ser possível utilizar a notação UML para ilustrar seus processos além de disciplinar o desenvolvimento através das etapas pré-estabelecidas: levantamento de requisitos, análise de requisitos, projeto de software, desenvolvimento e testes.

O desenvolvimento se deu de forma iterativa, pois a cada nova funcionalidade ou grupo de funcionalidades a serem desenvolvidos as etapas de análise de requisitos, projeto, desenvolvimento e teste foram executadas.

Foi de forma incremental pois ao fim de cada iteração, este novo desenvolvimento era integrado aos demais já finalizados, e novos testes eram feitos.

Apesar de a documentação de projetos em metodologias ágeis ser mais simples que outros métodos de desenvolvimento, sua dinâmica e possível customização tornou viável a construção de uma documentação mais robusta a partir dos diagramas UML e suas especificações.

Esta documentação foi apresentada com a modelagem do sistema desenvolvido. Os artefatos produzidos foram: protótipos de tela; diagramas de Casos de Uso UML; especificações de Casos de Uso; diagrama de Classe UML; modelo Entidade-Relacionamento (ER) e modelo lógico.

## 4.1 Levantamento de Requisitos

O levantamento de requisitos se deu junto ao professor da disciplina de Trabalho de Conclusão de Curso (TCC) do Ecomp-poli, além do documento de normas do TCC descrito em [4].

Esta etapa foi responsável por encontrar os agentes (atores) do sistema, quais as funções que deveriam ser possíveis destes realizarem (Requisitos funcionais) e como o sistema deveria ser implementado (requisitos não-funcionais).

## 4.2 Análise de requisitos e Projeto do software

Nesta etapa do desenvolvimento os modelos, diagramas e especificações foram criados, são eles:

- **Modelos:** modelo ER e modelo lógico do banco de dados.
- **Diagramas:** Diagramas de Casos de Uso UML e diagramas de classe UML.
- **Especificações:** Protótipos de tela e especificações de Casos de Uso.

### 4.2.1 Protótipos de tela

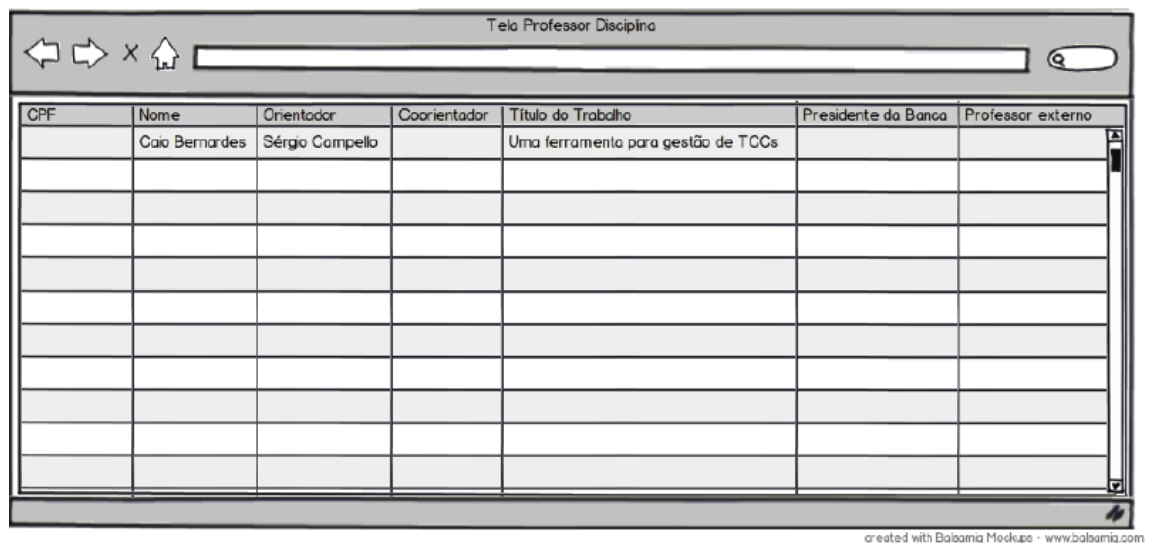
Para o design dos protótipos foi utilizada a prototipação exploratória. Este tipo de prototipação foi escolhido por conseguir fornecer as telas de maneira mais ágil sem se preocupar com código escrito ou com detalhes acerca das funcionalidades. Além disso, foi utilizada por ser possível procurar novas funcionalidades antes não vistas através do levantamento de requisitos, sendo possível integrar, após uma análise do mesmo, ao projeto do software.

Para a criação dos protótipos de tela, foi utilizada a ferramenta Balsamiq Mockups[14] versão online. Sua utilização ocorreu pelo prévio conhecimento da ferramenta pelos envolvidos no desenvolvimento, por ser um



software que não necessita instalação e por ser possível criar elementos próprios a partir de um desenho simples.

A figura 1 mostra o protótipo da tela do professor da disciplina. Serve para mostrar como será a visão do professor, através de uma tabela, onde será possível para ele modificar dados do aluno.



**Figura 1.** Protótipo Professor Disiciplina

A figura 2 mostra o protótipo de tela do aluno. É possível observar um campo para a modificação dos dados pessoais do aluno, como nome, email, nome do orientador e do co-orientador e outro campo onde aparece o título do projeto e da monografia e é possível dar *upload* nos arquivos textos dos mesmos.

Protótipo de tela do aluno. A interface é dividida em duas colunas principais. A coluna da esquerda, intitulada 'Dados Pessoais', contém campos para 'Nome' (preenchido com 'Caio César Bernardes'), 'Email' (preenchido com 'ccbl@ecomp.poli.br'), 'Orientador' (preenchido com 'Sergio Campelo') e 'CoOrientador' (campo vazio). A coluna da direita, intitulada 'Projeto', contém um campo 'Título' (preenchido com 'Título Provisório'), um botão 'Carregar arquivo' e uma mensagem de status 'arquivo "ProjetoFinal.pdf" carregado.'. Abaixo disso, há uma seção 'Monografia' com um campo 'Título' (preenchido com 'Título Oficial'), outro botão 'Carregar arquivo' e uma mensagem de status 'arquivo "Monografia.pdf" carregado.'. No rodapé da interface, há um botão 'Confirmar Modificações'.

Figura 2. Protótipo Aluno

A figura 3 mostra o protótipo da tela do professor orientador/examinador. Esta tela dá a idéia da visão que o professor terá, duas listas em forma de tabela, onde uma mostra os alunos os quais deve avaliar e a outra mostra seus orientandos. Na lista dos orientandos, ele pode aprovar ou reprovar os documentos de texto enviados pelos alunos, e na lista de avaliação, ele pode fazer o *download* da monografia do aluno e atribuir a sua nota.

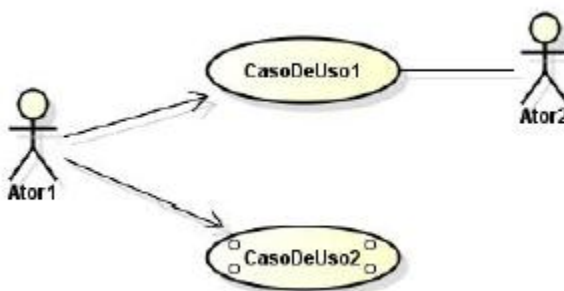
Protótipo de tela do professor. A interface é dividida em duas seções principais. A primeira seção, intitulada 'Lista Orientandos', contém uma tabela com as seguintes colunas: 'CPF', 'Nome', 'Projeto' e 'Monografia'. A primeira linha da tabela mostra 'xxxx' no CPF, 'Caio' no Nome, e opções de 'Aprovar' e 'Reprovar' nas duas últimas colunas. Abaixo da tabela, há um botão 'Salvar Modificações dos orientandos'. A segunda seção, intitulada 'Avaliação', contém uma tabela com as seguintes colunas: 'CPF', 'Nome', 'Monografia' e 'Nota da defesa'. A primeira linha da tabela mostra 'xxxx' no CPF, 'Caio' no Nome, 'Monografia.pdf' no Monografia, e '10,0' no campo de nota. Abaixo da tabela, há um botão 'Salvar Modificações das avaliações'.

**Figura 3.** Protótipo Professor Orientador/Avaliador

#### 4.2.2 Diagramas de Casos de Uso UML e suas especificações

Para se obter uma visão das interações do sistema com seus usuários, foi utilizado o diagrama de Caso de Uso UML e suas especificações. Este tipo de diagrama consegue, de forma simples, mostrar uma funcionalidade e qual usuário, chamado de ator, deve executá-la. De forma geral, um Caso de Uso busca responder a questão: Que usos terá o sistema? [11]

Um Caso de Uso é representado por um balão com uma descrição de sua funcionalidade, e deve estar ligada a um ator do sistema, conforme a figura 4 mostra.



**Figura 4.** Exemplo Caso de Uso

A especificação de um Caso de Uso é um padrão de escrita que tenta englobar várias características necessárias para o entendimento do diagrama, como regras de negócios e requisitos não funcionais, da forma de um documento de texto. Seu objetivo é detalhar os cenários identificados no diagrama, descrevendo seu comportamento, orientando todo seu desenvolvimento, sua validação, verificação e testes [11].

A especificação utilizada neste trabalho segue a norma descrita por [11] e contém: Nome do Caso de Uso, fluxo de eventos, pré-condições, pós-condições. O fluxo de evento é composto por um fluxo principal e por fluxos secundários.

Os Casos de Uso e suas especificações deste trabalho são mostrados no Capítulo 5 – resultados e testes.

#### **4.2.3 Diagramas de Classes UML**

Para se obter uma visão detalhada dos objetos criados, seus atributos e interações, utilizam-se diagramas de Classes. Eles devem mostrar classes e um conjunto de objetos que colaboram ou interagem para a execução dos serviços oferecidos pelo sistema, e eles devem conter os atributos pertencentes àquela classe e os métodos responsáveis pela manipulação de seus atributos [12].

O diagrama de classe da ferramenta feita neste trabalho foi dividido em 4 diagramas de classe menor, para uma melhor compreensão das classes e uma simplificação do mesmo foi feita.

A figura 5 mostra a primeira parte deste diagrama, qua contém a camada de dados (data), onde aparecem as classes que representam os objetos User, Professor, Student e Project. A classe User guarda dados que são comuns a classe Professor e a classe Student, para isso utilizou-se a metodologia de Herança [12]. Além disso, é possível ver também no diagrama a associação entre professor e aluno, que pode ser dar de quatro maneira: professor orientar aluno, professor coorientar aluno, professor ser presidente de banca de aluno e professor fazer parte da banca como membro externo.

A classe Project é associada à classe Aluno como parte pelo todo, ou seja, se os dados da classe Aluno são apagados, os dados referentes à classe Project também o são, uma vez que dados como nota da defesa e se a entrega foi no prazo fazem parte do Project.

Pra finalizar, são mostradas as classes que fazem a comunicação com o Banco de Dados, ProfessorSql, StudentSql e ProjectSql. Estas classes recebem uma conexão como parâmetro e, a partir dos métodos insert, remove, get e update, chama as procedures do Banco de dados.

A classe servicesPackage é uma abstração e uma simplificação do diagrama de classe que mostra a camada de serviços do software.

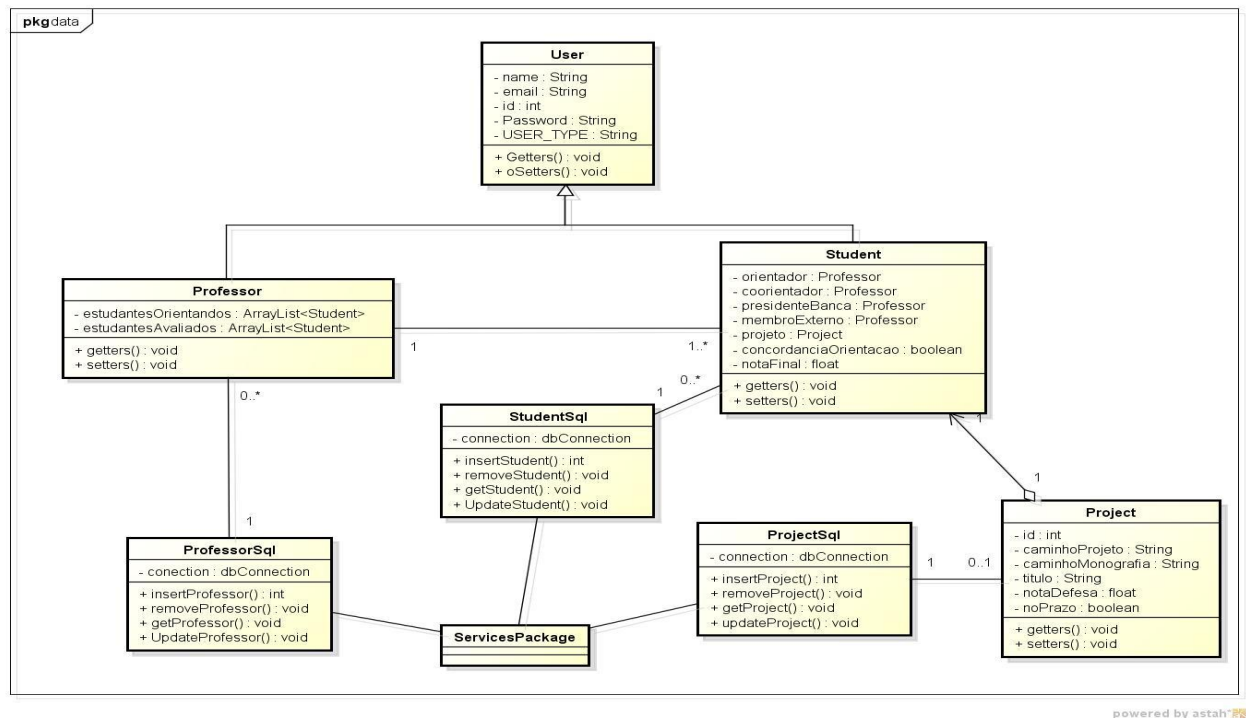


Figura 5. Diagrama Classe data

A figura 6 mostra o diagrama de classe da camada de serviços. Neste diagrama são mostrados os repositórios, estruturas de dados que receberão os dados vindo do banco. Especificando listas contendo professores, alunos e projetos. Esta camada também é responsável pela criação da conexão do banco, mostrada pela classe ConnectBd. Nesta classe está toda a configuração necessária para a conexão ao banco de dados que é passada, através dos repositórios às classes de dados responsáveis pela conexão.

As três classes principais são RepositorioAluno, RepositorioProfessor e RepositorioProjeto, que se comunicam com a classe bussiness, abstração da camada bussiness e com a classe data, uma abstração da camada data.

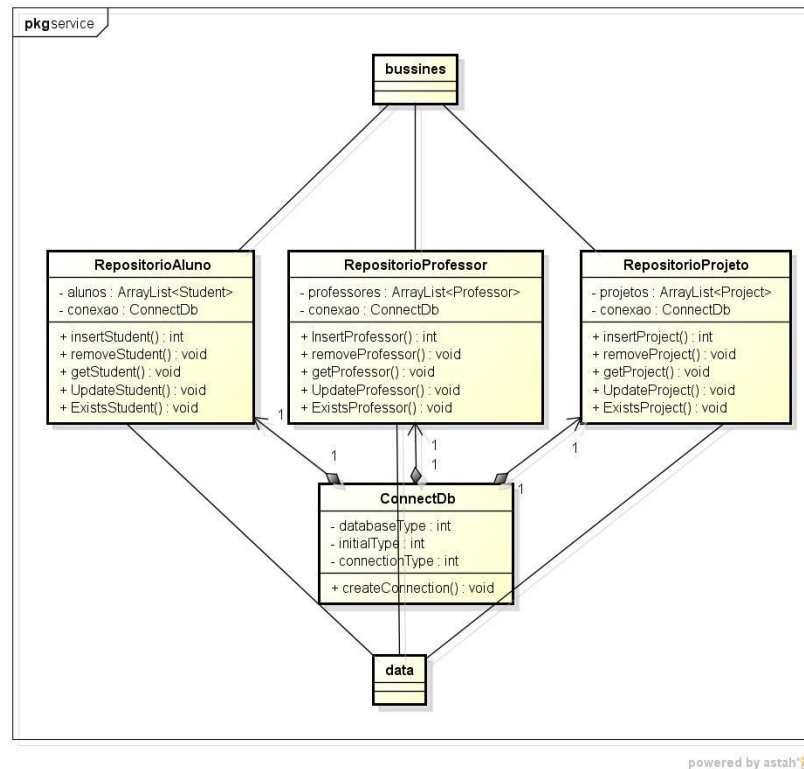
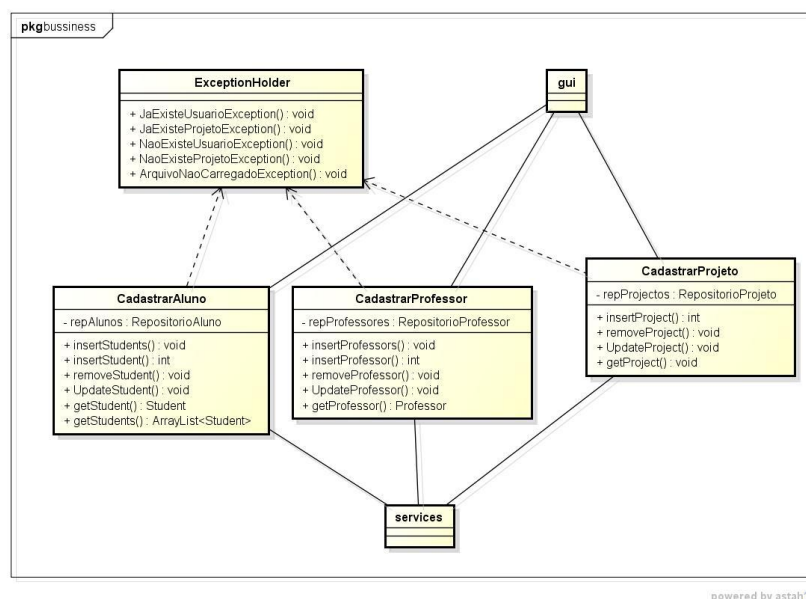


Figura 6. Diagrama classe service

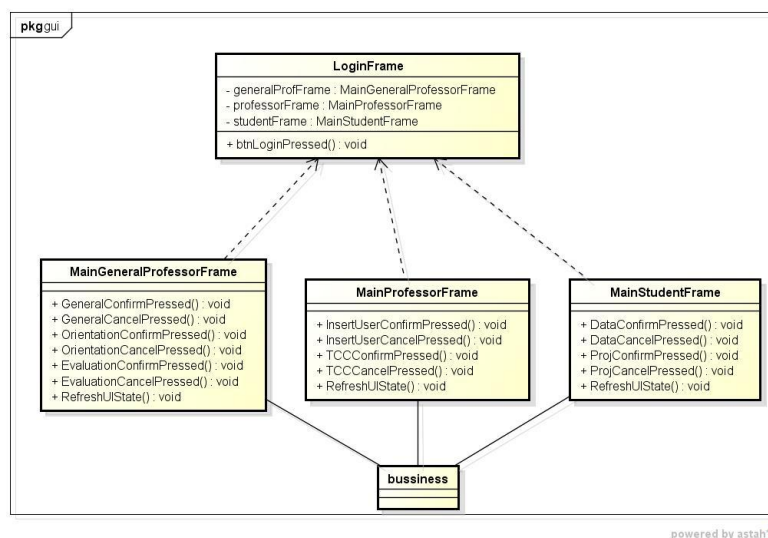
A figura 7 mostra o diagrama de classe da camada de negócios. Nesta camada toda a regra de negócio é feita. Todas as restrições do software, denominadas Exceptions são criadas nesta camada. Nem todas são executadas aqui pois algumas podem executar na camada acima de interface com o usuário (gui).

São três classes principais, CadastrarProfessor, CadastrarAluno e CadastrarProjeto. Cada uma guarda um repositório de dados e deve interagir com ele. Todo o desenvolvimento que necessite de restrições, condições e repetições deve ser executado nessa camada, nunca na camada de telas.



**Figura 7.** Diagrama classe bussiness

A figura 8 mostra a camada mais simples, a gui. Nesta camada são mostrados pelo diagrama quatro classes principais, cada uma representando uma tela. A tela LoginFrame funciona como uma tela principal e guarda a referência das outras três telas. A classe MainGeneralProfessorFrame contém as telas que serão utilizadas pelo usuário professor orientador/avaliador. A classe MainProfessorFrame contém as telas utilizadas pelo usuário professor da disciplina. Por fim, a classe MainStudentFrame contém as telas utilizadas pelo usuário aluno.



**Figura 8.** Diagrama classe gui

#### **4.2.4 Modelo ER e modelo lógico**

O modelo ER e o modelo lógico foram necessários para dar uma visão dos dados armazenados e das tabelas criadas. Este modelo é composto por entidade, representada por um retângulo, e as relações entre estas entidades, representados por um losango e os atributos de cada entidade, representados por um círculo ligado a entidade. Os números que aparecem entre parênteses no modelo representam a quantidade mínima e máxima de entidades relacionadas a uma determinada entidade.

Para a criação destes modelos, foi utilizado a ferramenta BrModelo, um software prático, sem necessidade de instalação e de fácil utilização.

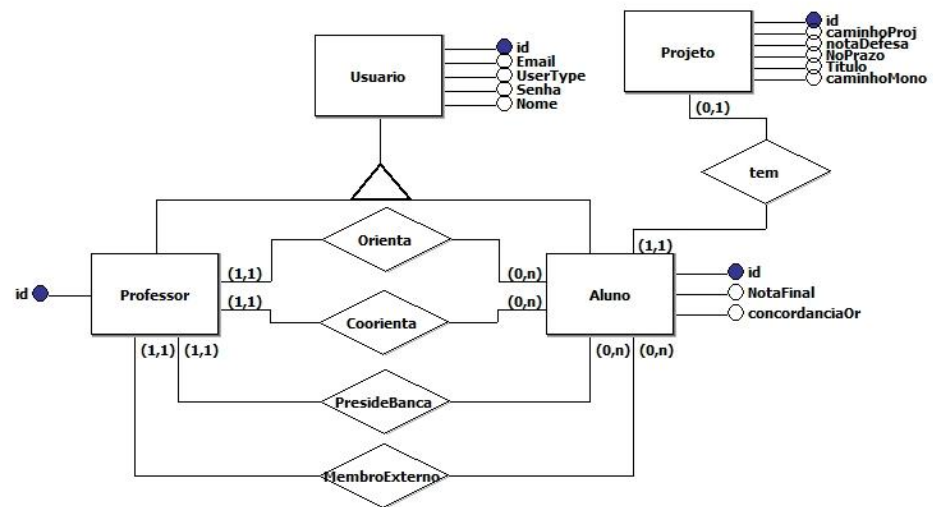
A Figura 9 mostra o diagrama ER do sistema. Ele é composto por quatro entidades: Usuario, Professor, Aluno e Projeto. Cada uma com seus atributos atômicos.

Professor e Aluno são entidades filhas de Usuario, por possuírem dados que são análogos. Existem quatro relações entre elas, de orientação, coorientação, presidente da banca e Membro externo, todas com um professor podendo ter n alunos e um aluno tendo apenas um professor.

Um Aluno pode ter nenhum ou algum Projeto, enquanto um Projeto deve ser ligado a um Aluno. Projeto é uma entidade fraca de Aluno, ou seja, se os dados do Aluno for apagado, o Projeto vinculado a ele também é apagado.

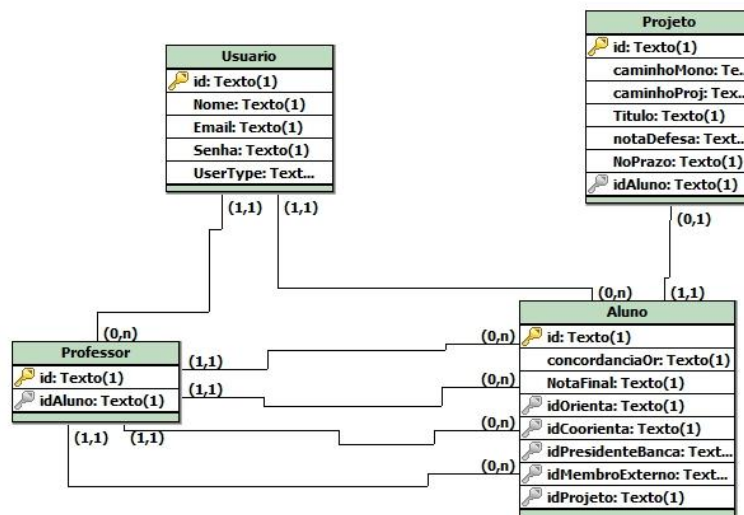
O modelo lógico é uma representação gráfica das tabelas desenvolvidas para a ferramenta, representadas por retângulos, e composta pelos dados de cada tabela.





**Figura 9.** Modelo Entidade-Relacionamento

A Figura 10 mostra o modelo lógico do sistema criado. Apresenta quatro tabelas: Usuario, Professor, Aluno e Projeto. A partir deste modelo foi feito todo o desenvolvimento do Banco de Dados.



**Figura 10.** Modelo Lógico

## 4.3 Desenvolvimento

### 4.3.1 Linguagem de Programação

A linguagem de programação escolhida para o desenvolvimento do trabalho foi a linguagem JAVA. Esta escolha se deu pela familiaridade dos envolvidos com a mesma, além de JAVA ser uma linguagem orientada a objetos, sendo ideal em conjunto com metodologias ágeis, e pela vasta quantidade e qualidade de suas bibliotecas nativas.

Foi utilizado a IDE Eclipse versão Europa, também devido à familiaridade de uso e por ser possível integrar nela a criação dos algoritmos e das telas desenvolvidas.

### 4.3.2 Estruturação

A estruturação do código foi feita por camadas. Uma camada para dados, uma para serviços, uma para negócios e uma para a interface.

O desenvolvimento foi estruturado utilizando o padrão de projeto Fachada, este padrão, como o nome sugere, representa a parte externa de um edifício. Ao passar e olhar este prédio, é apenas possível ver as paredes, janelas e portas deste edifício. Tudo que é interno ao prédio, como tubos, fiação e outras complexidades, são deixadas de lado por quem está olhando [13].

Esta é a forma como o padrão é usado, ela esconde a complexidade da próxima camada, e fornece uma interface para a camada atual, onde ela se comunica com a outra camada [13].

- **Camada de Dados**

A camada de dados é responsável por guardar os dados atômicos do desenvolvimento, ou seja, os objetos que devem ser utilizados pelo sistema. Além disso, é responsável pela interação direta com o Banco de Dados, conectando-se as procedures para atualizar os dados.

- **Camada de Serviço**

A camada de serviço é responsável por criar a conexão com o banco de dados, para que a regra de negócio não se preocupe em como isso é feito e para que a camada de dados já receba toda a configuração de conexão, bastando apenas fazer a conexão, sem configurar nada mais. Esta camada guarda os dados em uma estrutura de dados. Neste trabalho foi utilizado uma LinkedList por sua simplicidade de utilização. A cada modificação desta estrutura, um evento é disparado ativando a conexão com a Base de Dados.

- **Camada de Negócios**

A camada de negócio é responsável por todas as regras de negócios do sistema. Todas as restrições e exceções que o software possa ter devem ser cheçadas nesta camada. Isto é feito para que a camada de interface com o usuário não se preocupe em como um erro ou uma restrição acontece, ele deve apenas informar ao usuário. Esta camada liga-se também à camada de serviço, onde ela guarda a interface, sem se preocupar qual a estrutura de dados utilizada pelo repositório da camada abaixo.

- **Camada de Interface com o Usuário**

A camada de interface com o usuário (gui), é responsável por lançar os eventos executados pelo usuário, como clicar em um botão ou modificar algum dado de alguma tabela. Estes objetos são criados de forma automática pela IDE gráfica utilizada, por isso não é viável colocar um exemplo do seu código, mas as telas do sistema serão mostradas e explicadas na seção a seguir.

## **4.4 Funcionamento do sistema**

Como mostrado nos diagramas de classe, o sistema possui quatro telas principais: login, tela do professor da disciplina, tela do professor orientador/avaliador e a tela do aluno. Nesta seção serão mostrados essas telas e seu funcionamento.

#### 4.4.1 Tela de Login

É a tela principal do sistema, sendo utilizada por todos os usuários do sistema. Para logar e chegar a sua tela inicial, o usuário deve inserir seu login e sua senha, o sistema identificará qual o tipo de usuário e abrirá a tela correspondente, com os dados do usuário, como mostrado na Figura 11.

Existem links na parte de baixo da tela, estes links devem ser introduzidos pelo professor da disciplina, devem ser atalhos para arquivos, sites ou qualquer coisa que o professor ache ser necessário para o desenvolver da disciplina.



**Figura 11.** Tela Login

#### 4.4.2 Tela do professor orientador/avaliador

Esta tela é dividida em três abas:

- **Dados Gerais:** esta aba mostra os dados necessários do professor, nome e email, e duas listas, mostrando todos os alunos orientandos e os alunos que devem ser avaliados. Estas listas foram colocadas para agilizar a visão de seus alunos pelo professor. Esta tela está mostrada na Figura 12.

Professor

Dados Gerais   Orientações   Avaliações

Nome: Professor 1

Email: ccbll@ecomp.poli.br

Alunos Orientados:

Aluno 1

Alunos Avaliados:

Aluno 2

Confirmar Alterações   Cancelar Alterações

**Figura 12.** Dados Gerais Professor Orientador

- **Orientações:** Esta aba mostra uma tabela com todas as informações necessárias sobre os alunos orientados pelo professor. É nesta tela onde o professor pode modificar os dados de seus alunos, basta selecionar a célula, por exemplo, se ele remover a monografia o aluno deve mandar um novo documento. A Figura 13 ilustra essa tela.

Professor

Dados Gerais   Orientações   Avaliações

Aluno	Co-Orientador	Título Monogra...	Monografia	Nota Defesa	Nota Apresent...
Aluno 1		Título	Monografia.pdf		

Confirmar Alterações   Cancelar Alterações

**Figura 13.** Tela Orientações Professor Orientador

- **Avaliações:** Tela análoga a anterior, porém as células possíveis de serem editadas são diferentes. Nesta tela, o professor pode modificar, por exemplo, a nota da defesa do aluno. É ilustrada na Figura 14.

Aluno	Co-Orientador	Título Monogra...	Monografia	Nota Defesa	Nota Apresent...
Aluno 2		Título	Monografia.pdf		

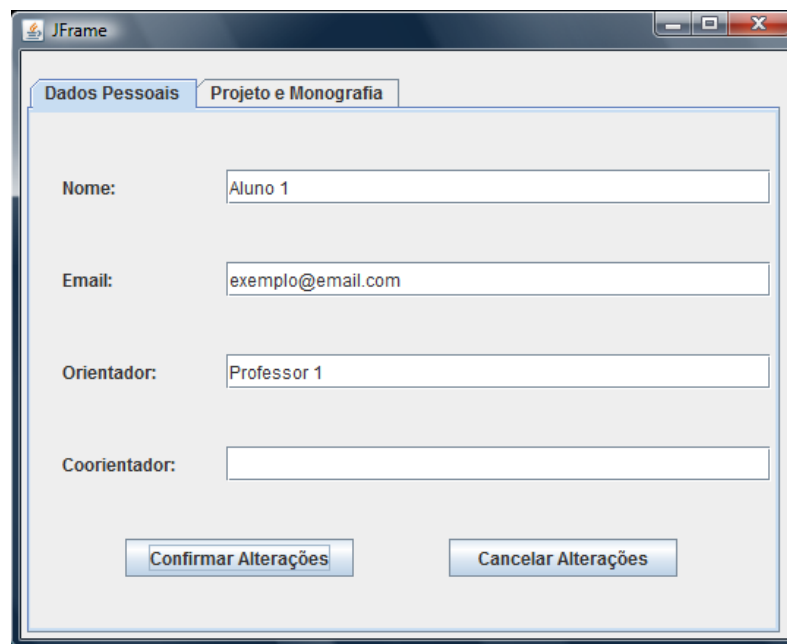
Confirmar Alterações      Cancelar Alterações

**Figura 14.** Tela Avaliação Professor Orientador

#### 4.4.3 Tela do Aluno

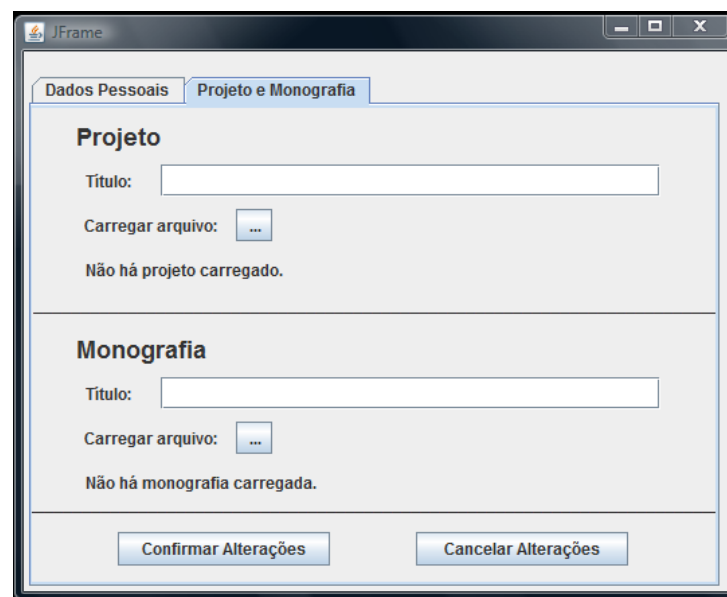
Esta tela é dividida em duas abas:

- **Dados Pessoais:** Nesta aba o aluno pode inserir e modificar todos os seus dados, nome, email, o seu orientador e seu coorientador. O nome do orientador tem *autocomplete*, ou seja, ao começar a inserir o nome do orientador, o sistema sugere, pelos nomes que tem, aquele orientador. Se o estudante tentar salvar um orientador não sugerido, o sistema não permite. A Figura 15 mostra um exemplo desta tela.



**Figura 15.** Tela Dados Pessoais Aluno

- **Projeto e Monografia:** Esta tela é possível o aluno carregar os arquivos referentes ao projeto e a monografia. Ao carregar o arquivo, se houver um professor orientador, um email é enviado a ele para que possa avaliar o arquivo, caso o professor reprove, o aluno deve carregar outro arquivo novamente. É possível também inserir e modificar o título do projeto e da monografia, como mostra a Figura 16.

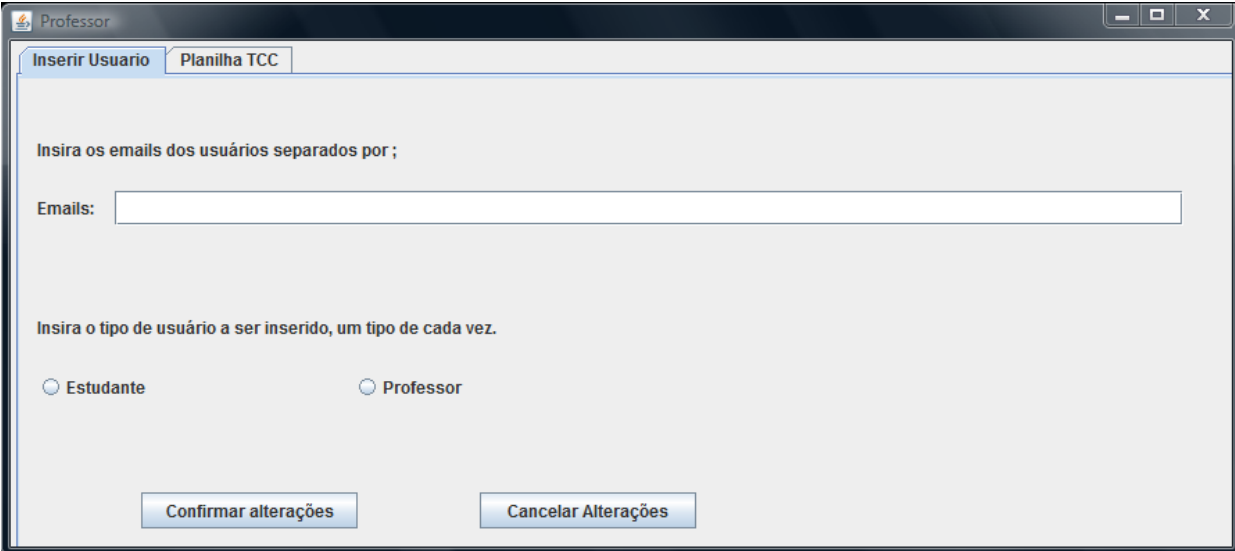


**Figura 16.** Tela Projeto e Monografia Aluno

#### 4.4.4 Tela do Professor da disciplina

Esta tela também é dividida em três abas principais:

- **Inserir Usuário:** Nesta aba o professor da disciplina deve inserir os usuários do sistema. Basta ir no campo de email e colocar os emails dos usuários separados por um ';', após isso selecionar qual o tipo de usuário que se está inserindo, Estudante ou Professor, para então apertar em confirmar. Esta tela esta ilustrada na Figura 17.

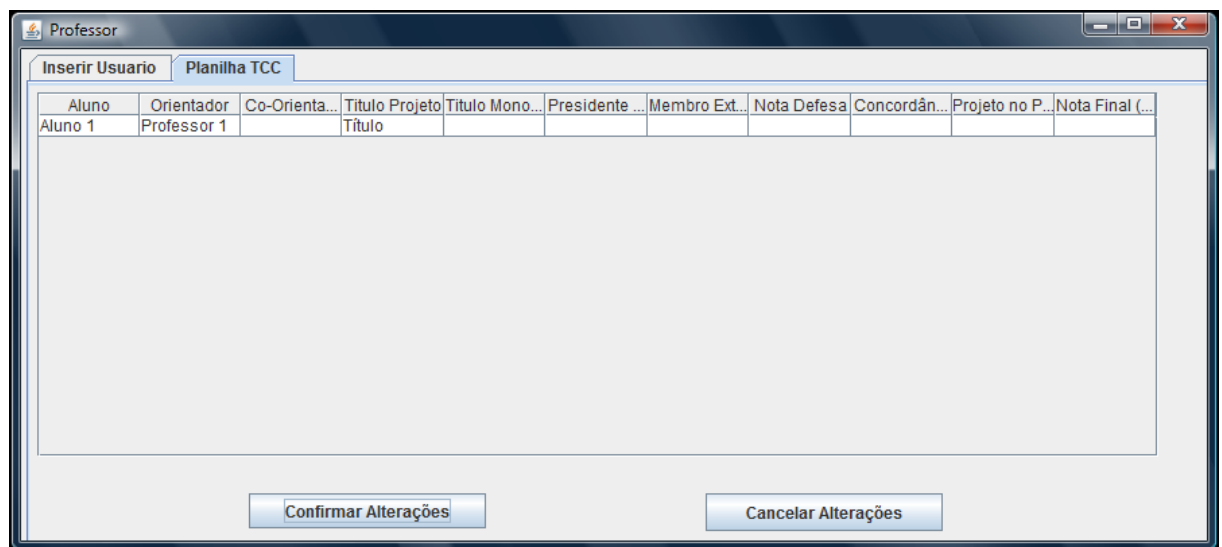


A imagem mostra uma janela de software intitulada "Professor". No topo, há duas abas: "Inserir Usuario" (selecionada) e "Planilha TCC". O conteúdo da aba "Inserir Usuario" inclui o texto "Insira os emails dos usuários separados por ;" seguido de um campo de entrada "Emails:". Abaixo, há o texto "Insira o tipo de usuário a ser inserido, um tipo de cada vez." e duas opções de rádio: "Estudante" e "Professor". No rodapé da janela, há dois botões: "Confirmar alterações" e "Cancelar Alterações".

**Figura 17.** Tela Inserir Usuário Professor da disciplina

- **Planilha TCC:** Esta tela apresenta, em um formato de tabela, todas as informações dos projetos da disciplina de Projeto de Final de Curso. Todos os campos desta tabela podem ser modificados pelo professor da disciplina, desde o nome do aluno à nota atribuída. O sistema calcula e mostra nesta tabela, na coluna Nota Final, a nota do aluno. É importante salientar que esta tabela pode ser facilmente exportada para outras ferramentas de planilha, um simples copiar e colar é suficiente. A figura 18 ilustra esta tela.





**Figura 18.** Tela Planilha TCC

- **Configurações:** Esta tela conta com as configurações do sistema que podem ser modificadas pelo professor da disciplina: Resetar senhas, inserir prazos de entrega e os links da tela principal. Esta tela não é mostrada nem detalhada pois suas funções não foram finalizadas até o fim da escrita deste documento.

# Capítulo 5

## Resultados e Testes

Este capítulo mostra a modelagem do sistema através de Casos de Uso e suas especificações. Indo além, mostra também os testes feitos a partir desta modelagem, estes testes estão descritos em conjunto com as especificações de casos de uso, onde observa-se a descrição do que deve ocorrer em cada caso de uso e, a partir dos fluxos alternativos, o que aconteceu ao executar os testes da ferramenta para determinado Caso de Uso.

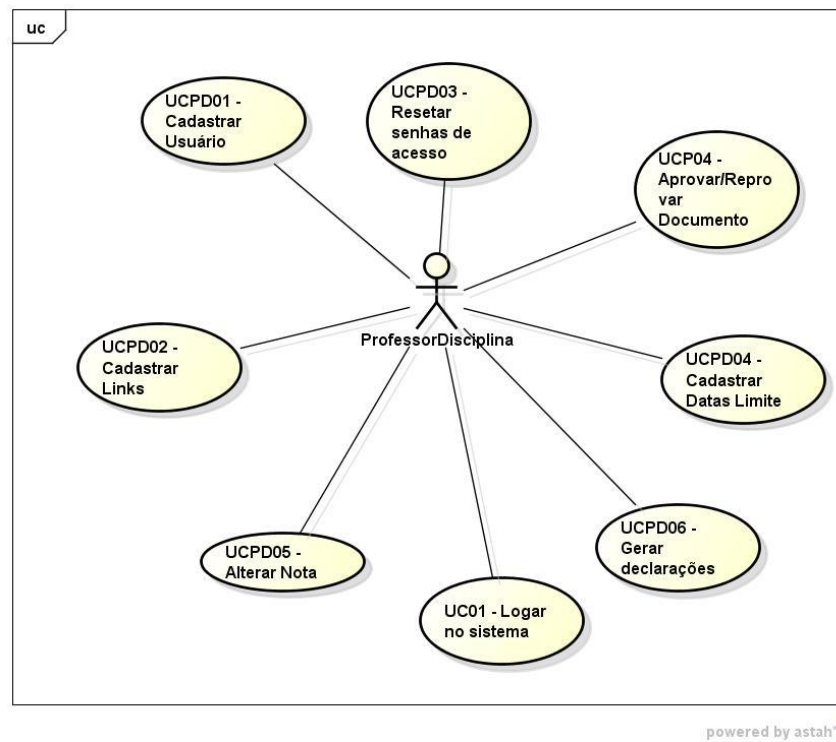
### 5.1 Modelagem do Sistema

#### 5.1.1 Diagramas de Casos de Uso

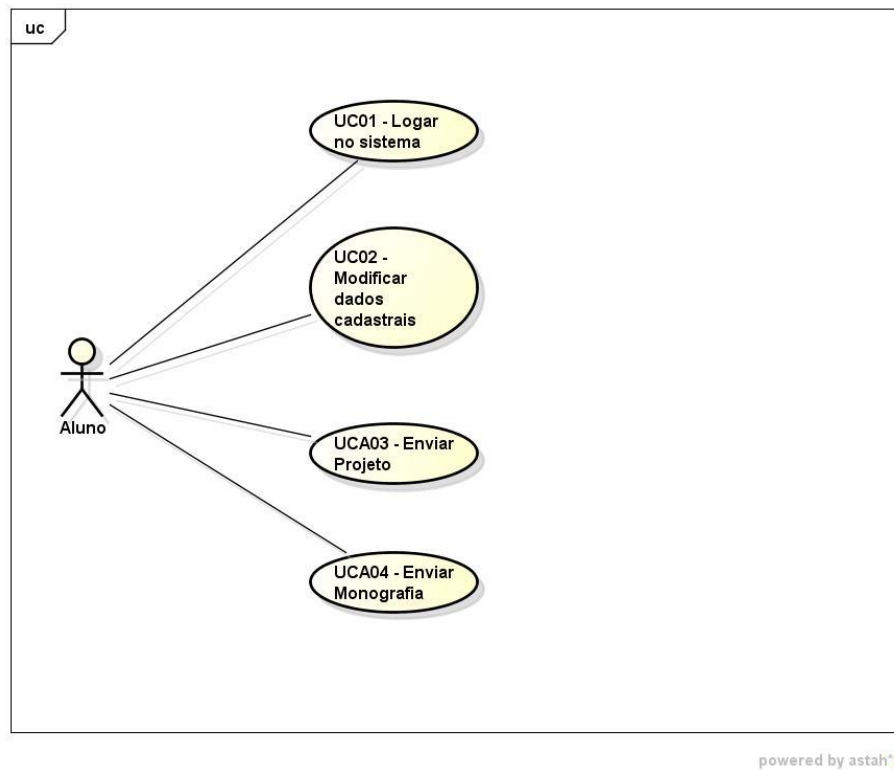
Foram gerados três diagramas de Casos de Uso, um para cada usuário do sistema. Para cada caso de uso, suas especificações foram construídas.

O Caso de Uso “Professor Orientador” mostra as interações do professor orientador com o sistema. O modelo, ilustrado na Figura 19, apresenta oito casos de uso que interagem com o ator, são eles: Cadastrar Usuario, Cadastrar Links, Resetar senhas de acesso, Alterar notas, Logar no sistema, Aprovar/Reprovar Documento, Cadastrar DatasLimite, Gerar Declarações.

O Caso de Uso “Aluno” mostra as interações do usuário Aluno com o sistema. Como se vê na Figura 20, este modelo apresenta quatro casos de interação: Logar no sistema, Modificar dados cadastrais, Enviar Projeto, Enviar Monografia.



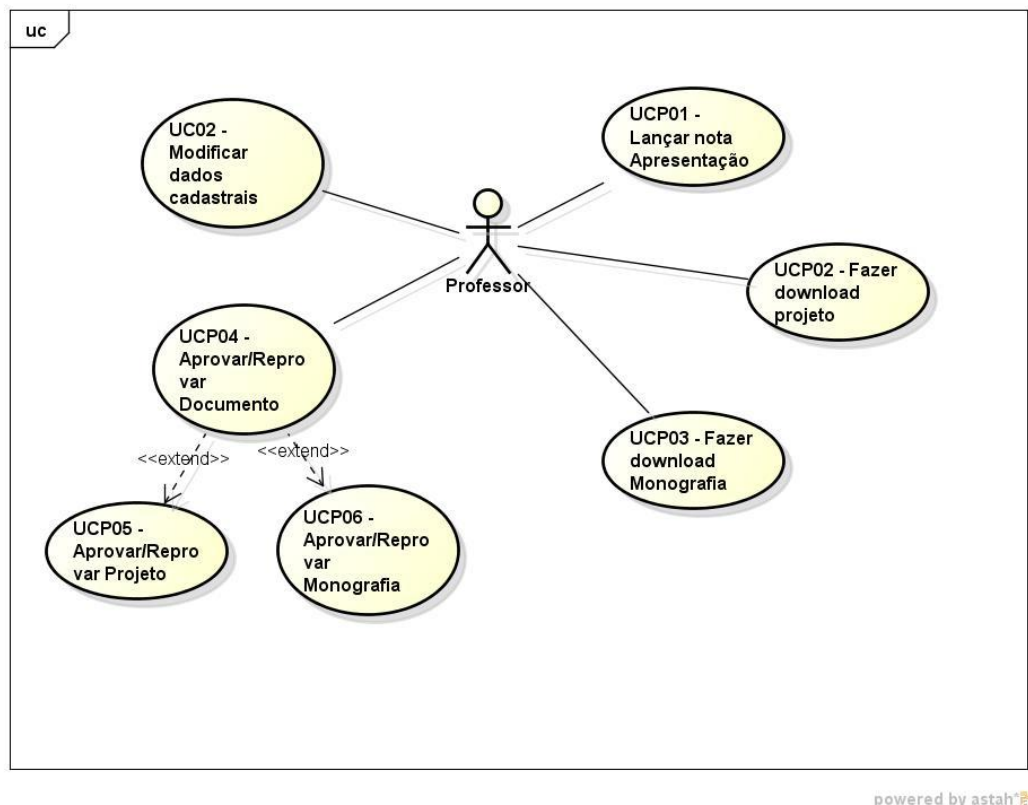
**Figura 19.** Caso de Uso Professor Orientador



**Figura 20.** Caso de Uso Aluno

O Caso de Uso “ProfessorGeral” ilustrado na Figura 21, mostra as interações do usuário Professor orientador/avaliador com o sistema. Apresenta sete possíveis interações com o sistema: Modificar dados cadastrais, Lançar Nota apresentação, Fazer download Projeto, Fazer download Monografia, Aprovar/Reprovar Documento que estende nos casos de uso Aprovar/Reprovar Projeto e Aprovar/Reprovar Monografia.

Esta extensão descrita no parágrafo anterior se dá quando existe, na modelagem, dois casos de uso que utilizam as mesmas formas de descrição, mudando apenas algum detalhe, no caso qual o documento a ser aprovado ou reprovado.



**Figura 21.** Caso de Uso Professor Disciplina

### 5.1.2 Especificações de Casos de Uso

As especificações criadas no desenvolver da ferramenta seguiram as especificações de CockBurn[11], e contêm: Identificador, Nome do Caso de Uso, pré-condição, Descrição, Pós-condição e fluxos alternativos. Estas especificações

foram numeradas e serão mostradas. Primeiramente as especificações gerais, ou seja, onde mais de um usuário a segue, depois as especificações que tem como ator o usuário professor da disciplina. Não serão mostradas todas as especificações criadas, apenas aquelas que ocorreram algum tipo de teste, ou seja, que fluxos alternativos foram encontrados e detalhados diretamente nas especificações.

- **UC01 – Logar no sistema**

**Pré-Condição:**

O sistema ter sido iniciado por um usuário (professor da disciplina, professor geral ou aluno).

**Descrição:**

1. O usuário insere seu email no campo email.
2. O usuário insere sua senha no campo senha e clica em 'entrar'.
3. O sistema valida o email e a senha.

**Pós-Condição:**

O sistema mostra a tela principal do usuário.

**Fluxos alternativos:**

Em 3, o sistema não consegue validar a senha ou o email do usuário.

**3A.** O sistema exibe um alerta ao usuário pedindo para inserir novamente seu email e senha.

**3B.** Caso o usuário seja aluno ou professor geral e erre mais de 3 vezes, sua senha é bloqueada e deve ser resetada junto ao professor da disciplina (UCPD03).

- **UC02 – Modificar dados cadastrais**

**Pré-Condição:**

Um usuário (aluno ou professor) estar logado no sistema.

**Descrição:**

1. O usuário seleciona 'Modificar dados cadastrais'.
2. O sistema exibe uma nova tela com as informações do usuário já cadastradas (Nome, email, senha, título projeto, nome orientador, nome co-orientador, etc).
3. O usuário modifica os campos que quiser e clica em 'Confirmar'.
4. O sistema salva os novos dados e exibe uma tela de sucesso.

**Pós-Condição:**

Dados cadastrais do usuário atualizados.

**Fluxos alternativos:**

FA01 : Em 3, caso o usuário seja aluno e o campo modificado seja nome orientador.

**4A.** Uma mensagem de alerta é exibida ao usuário dizendo que um email será mandado ao antigo professor avisando sobre o desligamento, e um email será enviado ao novo orientador, pedindo sua confirmação.

**4B.** O aluno confirma a modificação.

**4C.** O sistema envia os emails e salva os novos dados.

- **UCPD01 – Cadastrar Usuário**

**Pré-Condição:**

Professor da disciplina estar logado ao sistema.

**Descrição:**

1. O professor clica em 'Cadastrar usuário'.
2. O sistema exibe a tela de Cadastro de usuários.
3. O professor insere no campo 'email' o email do usuário a ser cadastrado.
4. O professor escolhe a ação 'Inserir novo usuário'.
5. O professor seleciona qual o tipo de usuário a ser cadastrado (Professor ou aluno).
6. O sistema checa se não existe um cadastro com aquele email.
7. O sistema valida a inserção e envia uma mensagem de sucesso ao professor.

**Pós-Condição:**

Um novo usuário foi inserido no sistema. Se entrar no Fluxo alternativo FA1, a pós-condição muda para – Um novo usuário foi removido do sistema.

**Fluxos alternativos:**

**FA1** : Em 4, o professor selecionar a ação 'Remover um usuário'.

**4A.** O sistema checa a existência do cadastro com aquele email.

**4B.** O sistema valida a remoção e envia uma mensagem de sucesso ao professor.

**FA2** : Em 6, caso o sistema encontre um usuário já cadastrado com aquele email.

**6A.** O sistema envia um alerta ao professor informando 'Este email já está cadastrado'.

**FA3** : Em 4A, caso o sistema não encontre um usuário cadastrado com aquele email.

**4AA.** O sistema envia uma mensagem de erro ao professor informando 'Email não cadastrado'.

- **UCPD02 – Cadastrar Links**

**Pré-Condição:**

O professor da disciplina estar Logado.

**Descrição:**

1. O professor clica em 'Cadastrar Links'.
2. O sistema exibe a tela de cadastro de links, com quatro campos de links possíveis.
3. O professor insere os links em cada campo, podendo deixar algum em branco e clica no botão 'Confirmar'.
4. O sistema checa a conectividade dos links.
5. O sistema salva os links e exibe uma mensagem de sucesso ao professor.

**Pós-Condição:**

A tela inicial mostra os links cadastrados.

**Fluxos alternativos:**

**FA1** : Em 4, caso o sistema não consiga checar os links.

**4A.** O sistema salva os links e exibe uma mensagem de alerta ao professor 'Link sem acesso'.



- **UCPD03 – Resetar senhas de acesso**

**Pré-Condição:**

O professor da disciplina estar logado e ter pelo menos um usuário além dele cadastrado.

**Descrição:**

1. O professor clica em 'Resetar senha de usuário'.
2. O sistema exibe uma nova tela, com um campo 'email do usuário' e um botão 'confirmar'.
3. O professor insere o email do usuário a ter sua senha resetada e clica em 'confirmar'.
4. O sistema valida o pedido e reseta a senha do usuário para a senha padrão.

**Pós-Condição:**

O usuário tem sua senha resetada para a senha padrão.

**Fluxos alternativos:**

**FA01** : Em 4, caso o usuário não seja encontrado.

**4A.** Uma mensagem de erro é mostrada na tela 'Email não cadastrado'.

- **UCPD04 – Cadastrar Datas Limite**

**Pré-Condição:**

O professor da disciplina estar logado.

**Descrição:**

1. O professor clica em 'Datas Limite'

2. O sistema exibe uma nova tela com 3 campos – Data Limite Termo de concordância de orientação; Data Limite Projeto; Data Limite Monografia.
3. O professor insere as datas de cada campo, podendo deixar algum em branco e clica em ‘confirmar’.
4. O sistema salva as Datas Limite.

**Pós-Condição:**

As datas limite terem sido modificadas.

**Fluxos alternativos:**

N/A

- **UCPD05 – Alterar Notas**

**Pré-Condição:**

O professor da disciplina estar logado e ter pelo menos um aluno com os documentos projeto e monografia salvo e aprovado.

**Descrição:**

1. O professor vai à tela ‘TCC’.
2. O professor seleciona qual o TCC a ter sua nota alterada pelo nome do aluno.
3. O professor vai ao campo ‘Nota’ e modifica-a.
4. O professor clica em ‘confirmar’.
5. O sistema salva a nova nota do TCC.

**Pós-Condição:**

Nota do aluno alterada.

**Fluxos alternativos:**

**FA01** : Em 5, caso o aluno esteja com alguma pendência – Entrega de Projeto/Monografia ou Aprovação de Projeto/Monografia.

**5A.** Uma mensagem de alerta é exibida ‘O aluno ainda tem pendências’.

**5B.** O sistema salva a nova nota.

## Capítulo 6

# Conclusão e Trabalhos Futuros

### 6.1 Conclusão

A ferramenta desenvolvida e demonstrada neste trabalho contribui para a otimização do tempo gasto nos processos que envolvem o gerenciamento da disciplina Projeto Final de Curso (PFC), focando nas entregas dos projetos e das monografias dos alunos, e nas atribuições das notas.

Esta ferramenta tenta automatizar algumas tarefas desta disciplina, como as entregas de documentos e a interação do aluno com o seu orientador.

A ajuda maior do software produzido é para o professor da disciplina, o qual consegue gerir e observar, apenas pelas telas do software, todas as etapas da disciplina.

Os algoritmos aqui criados servem para ser usados em ambientes *desktops* e em ambientes virtuais, devido à portabilidade da linguagem escolhida. A conexão com o banco de dados, independentemente do ambiente escolhido para a execução, se dará da mesma maneira, sempre procurando o mesmo servidor.

Algumas novas oportunidades foram descobertas durante a elaboração da ferramenta tanto em relação à sua finalidade quanto à sua codificação. Na seção a seguir estas descobertas são detalhadas, bem como outros possíveis trabalhos futuros.

### 6.2 Trabalhos Futuros

Introduzir o banco de dados no servidor do ECOMP – POLI para testes mais aprofundados de suas funcionalidades e para encontrar novas funcionalidades.

Tornar esta ferramenta totalmente online, onde o usuário possa executá-la a partir de um browser da internet qualquer, sem a necessidade de fazer o download da ferramenta ou executá-la através de uma máquina virtual JAVA.

Terminar a funcionalidade de Geração de Certificados, onde os certificados da banca examinadora, do orientador e de membros externos sejam feitas de forma automática.

Terminar a funcionalidade de enviar email ao professor orientador, nesta ferramenta o email é enviado, mas não existe uma padronização neste envio, ou seja, se um link para a ferramenta deve ser exibido ou apenas um aviso da atualização.

Fazer um módulo para a ferramenta onde seja possível seu uso a partir de dispositivos móveis (celulares, tablets, etc).

# Bibliografia

- [1] SANCHES, Aglay. **Tecnologia Educacional e os recursos pedagógicos**. São Paulo: Anuário de Produção acadêmica, 2009.
- [2] SIG@ 2012-2013. Disponível em: <http://elkras.org/ufpe-siga-2012-2013-www-siga-ufpe-br/#ixzz23ZSArMFY>. Acesso em: 20/11/2012.
- [3] Moodle. Disponível em: <http://docs.moodle.org/dev/> Acesso em: 20/11/2012.
- [4] Normas do Projeto de Final de Curso. Disponível em: <http://www2.ecomp.poli.br/wp-content/uploads/2010/09/normasTCC.pdf>. Acesso em: 20/11/2012.
- [5] SOMMERVILLE, I. **Software Engineering**. Edinburgh: Addison-Wesley, 1995.
- [6] MAZZOLA, V. **Engenharia de Software**. Sem editora. 2010.
- [7] MYERS, J., John Wiley & Sons, **The Art of Software Testing**, Nova Jersey: 2004
- [8] Agile Manifesto (2004). Disponível em <http://agilemanifesto.org/>, acessado em 03 de Dezembro de 2012.
- [9] SOARES, M. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software**. UNIPAC – Universidade Presidente Antonio Carlos
- [10] Schwaber, K. e Beedle, M. **Agile Software Development with SCRUM**, Prentice-Hall, (2002)
- [11] COCKBURN, W. **Structuring use cases with goals**. Humans and Technology technical report, IEEE, USA, 1995.
- [12] JACOBSON. **UML User's Guide**. Addison-Wesley, 2010.

- [13] JAVA DESIGN PATTERNS. Disponível em:  
**[http://www.allapplabs.com/java\\_design\\_patterns/facade\\_pattern.htm](http://www.allapplabs.com/java_design_patterns/facade_pattern.htm)**,  
acessado em 03 de dezembro de 2012.
- [14] Balsamiq Mockups. Disponível em:  
**<http://www.balsamiq.com/products/mockups>**, acessado em 03 de  
dezembro de 2012.

# Apêndice A

## Diagrama de Classe completo

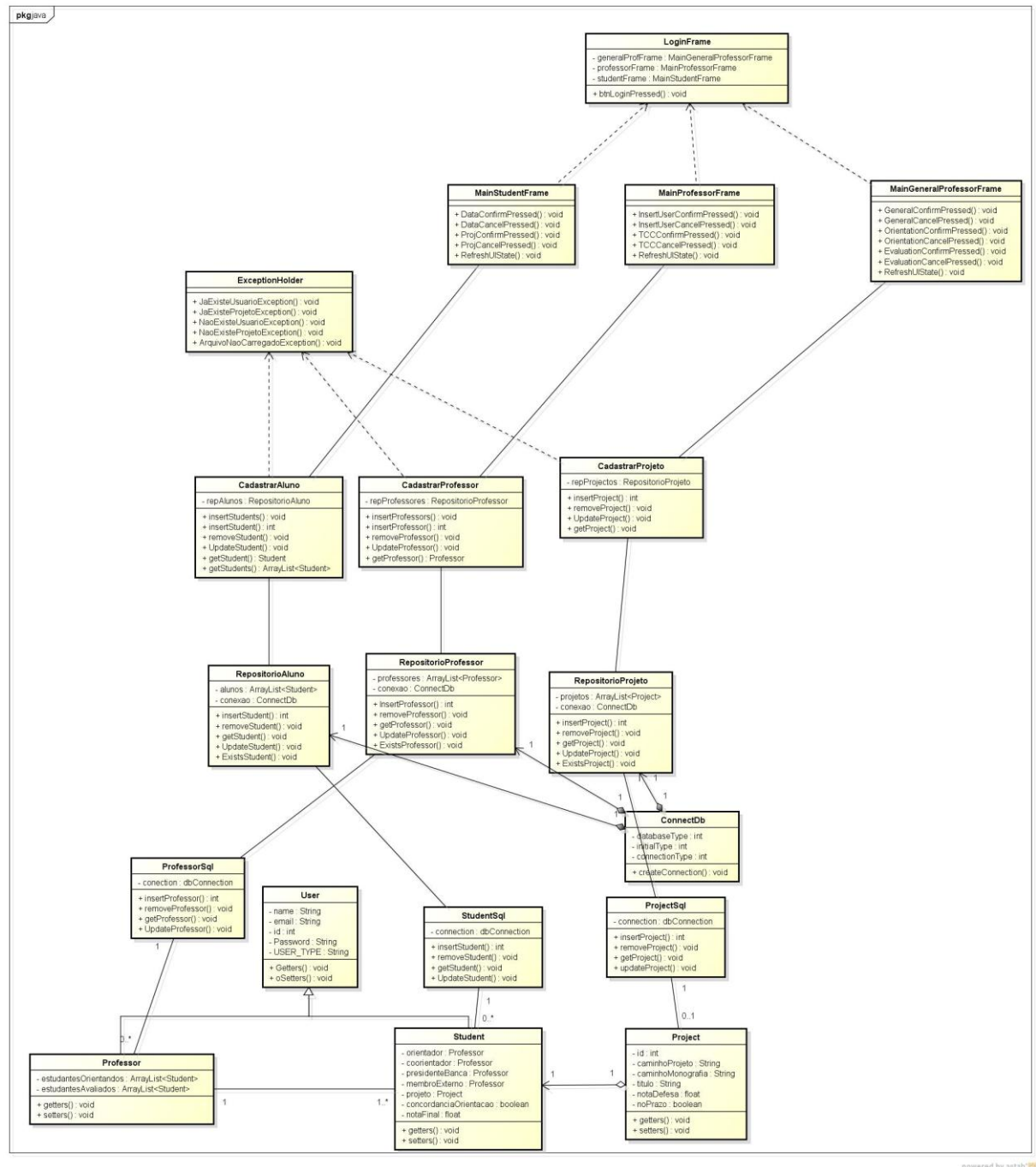


Figura 22. Diagrama de Classe completo



# Apêndice B

## Exemplos de código

- Código de um dos objetos da camada de dados:

```
package data;

public class Project {

    private int id;
    private byte[] projeto;
    private byte[] monografia;
    private String tituloProjeto;
    private String tituloMonografia;
    private float notaDefesa;
    private boolean noPrazo;

    public float getNotaDefesa() {
        return notaDefesa;
    }

    public void setNotaDefesa(float notaDefesa) {
        this.notaDefesa = notaDefesa;
    }

    public boolean isNoPrazo() {
        return noPrazo;
    }

    public void setNoPrazo(boolean noPrazo) {
        this.noPrazo = noPrazo;
    }

    public Project() {
        notaDefesa = -1;
        tituloProjeto = "";
        tituloMonografia = "";
        noPrazo = true;
    }

    public String getTituloProjeto() {
        return tituloProjeto;
    }

    public void setTituloProjeto(String tituloProjeto) {
        this.tituloProjeto = tituloProjeto;
    }

    public String getTituloMonografia() {
        return tituloMonografia;
    }

    public void setTituloMonografia(String tituloMonografia) {
```

```
        this.tituloMonografia = tituloMonografia;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public byte[] getProjeto() {
        return projeto;
    }

    public void setProjeto(byte[] projeto) {
        this.projeto = projeto;
    }

    public byte[] getMonografia() {
        return monografia;
    }

    public void setMonografia(byte[] monografia) {
        this.monografia = monografia;
    }

    public boolean equals(Object p){
        if(p instanceof Project){
            return ((Project)p).getId() == this.getId();
        }
        else
            return false;
    }
}
```

- Código de um dos objetos da camada de serviço:

```
package service;

import java.util.ArrayList;
import data.Project;

public class RepositorioProjeto {

    private ArrayList<Project> projetos;

    public RepositorioProjeto(ArrayList<Project> projetos) {
        this.projetos = projetos;
    }

    public int insertProject(Project projeto){
        //Insere o projeto no banco de dados.
        this.projetos.add(projeto);
        return projeto.getId();
    }

    public void removeProject(int id){
        this.projetos.remove(projetos.get(id));
    }
}
```

```
}

public Project getProject(int id){
    return this.projetos.get(id);
}

public void updateProject(Project p){
    Project oldProject = this.projetos.get(p.getId());
    oldProject.setMonografia(p.getMonografia());
    oldProject.setProjeto(p.getProjeto());
    oldProject.setTituloMonografia(p.getTituloMonografia());
    oldProject.setTituloProjeto(p.getTituloProjeto());
    oldProject.setNoPrazo(p.isNoPrazo());
    oldProject.setNotaDefesa(p.getNotaDefesa());
}

public boolean existsProject(int id){
    if(this.projetos != null && this.projetos.size() > 0){
        return this.projetos.contains(id);
    }
    else
        return false;
}
}
```

- Código de um dos objetos da camada de negócios

```
package bussiness;

import java.util.ArrayList;
import service.RepositorioProjeto;
import data.Project;

public class CadastrarProjeto {

    private RepositorioProjeto repProj;

    public CadastrarProjeto(){
        repProj = new RepositorioProjeto(new
        ArrayList<Project>());
    }

    public void insertProject(Project p){
        if(repProj.existsProject(p.getId())){
            throw new JáExisteProjetoException();
        }
        else
            repProj.insertProject(p);
    }

    public void removeProject(int id){
        if(!repProj.existsProject(id)){
            throw new NãoExisteProjetoException();
        }
        else
            repProj.removeProject(id);
    }

    public void updateProject(Project p){
        if(!repProj.existsProject(p.getId()))
```

```
        throw new NãoExisteProjetoException();
    else
        repProj.updateProject(p);
}

public void getProject(int id) {
    if(!repProj.existsProject(id))
        throw new NãoExisteProjetoException();
    else
        repProj.getProject(id);
}
}
```