



**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO**  
**CENTRO TECNOLÓGICO**  
**COLEGIADO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO**

Henrique Paulino Cruz

**Otimização e centralização das atividades  
administrativas na graduação: desenvolvimento  
de um módulo de controle no sistema Marvin**

Vitória, ES

2023

Henrique Paulino Cruz

**Otimização e centralização das atividades administrativas  
na graduação: desenvolvimento de um módulo de controle  
no sistema Marvin**

Monografia apresentada ao Curso de Engenharia de Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Colegiado do Curso de Engenharia de Computação

Orientador: Prof. Dr. Vítor E. Silva Souza

Vitória, ES

2023

Henrique Paulino Cruz

# **Otimização e centralização das atividades administrativas na graduação: desenvolvimento de um módulo de controle no sistema Marvin**

Monografia apresentada ao Curso de Engenharia de Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Trabalho aprovado. Vitória, ES, 14 de dezembro de 2023:

---

**Prof. Dr. Vítor E. Silva Souza**  
Orientador

---

**Prof<sup>a</sup>. Monalessa Perini Barcellos**  
Universidade Federal do Espírito Santo

---

**Prof. Eduardo Zambon**  
Universidade Federal do Espírito Santo

Vitória, ES  
2023

# Agradecimentos

Começo expressando profundo agradecimento aos meus pais, Flávia e Marcelo, que sempre enfatizaram a importância da educação como o caminho certo a seguir. A visão de mundo que eles me transmitiram certamente é responsável por todas as conquistas que alcancei até o momento. Não poderia ter desejado pais melhores, que sempre me proporcionaram o apoio essencial. Mesmo distante de casa, ter a certeza de um porto seguro no qual podia confiar me deu grande força. E, é claro, minha irmã caçula, Helena, que sempre foi uma luz em nossas vidas, nos alegrando e animando, sempre pronta a oferecer apoio quando necessário.

Agradeço imensamente à minha namorada, Gabrielly, por tornar a distância da família menos dolorosa. Sua companhia e auxílio nos momentos difíceis me mostraram que eu era capaz de superar qualquer desafio. A meus amigos Pablo e Victor, que ingressaram na UFES comigo e compartilhamos todos esses anos de faculdade morando juntos, foi uma experiência incrível que verdadeiramente me marcou e ensinou muito. Não poderia ter pedido por companheiros melhores.

Meu sincero agradecimento ao professor Vitor, meu orientador neste trabalho. Sua excepcional orientação, prontidão para ajudar e esclarecer dúvidas foram fundamentais para o sucesso deste projeto. Tenho certeza de que não poderia ter encontrado um professor mais alinhado com meus interesses.

A todos que conheci durante a graduação, meu muito obrigado. O departamento de informática é notável, com professores que são verdadeiras fontes de inspiração e merecem o título de educadores. Os projetos que participei durante a graduação, como PET e LabES, contribuíram significativamente para a minha formação.

Enfim, a todos que me apoiaram e acreditaram em mim, deixo aqui meu profundo agradecimento. A vida com pessoas que valorizam sua presença se torna mais leve e mais fácil de conduzir. Cada momento vivido com essas pessoas, sem dúvida, contribuiu para me conduzir até aqui.



# Resumo

Este trabalho se concentra no sistema Marvin, uma plataforma *Web* desenvolvida por alunos e professores da Universidade Federal do Espírito Santo (UFES) para simplificar diversas tarefas na comunidade acadêmica universitária. O módulo proposto neste estudo reconhece a necessidade de simplificar atividades tanto para estudantes quanto para coordenadores de curso, como a gestão do Acompanhamento de Desempenho Acadêmico (ADA), a administração de prazos e o processo de submissão de atividades acadêmicas.

Os objetivos deste estudo visam desenvolver um sistema *Web* que minimize tarefas manuais realizadas por coordenadores de cursos, centralize funções atualmente conduzidas via e-mail e simplifique os processos acadêmicos. Isso envolve a coleta e documentação de requisitos do módulo, o projeto de sua arquitetura integrada ao Marvin, bem como a implementação e teste do código-fonte.

O escopo de implementação concentra-se na criação de um ambiente organizado destinado a acadêmicos, enquanto a integração direta com os sistemas da UFES é vista como uma perspectiva futura. Isso ocorre porque a integração necessitaria de alinhamentos e discussões extensas com a administração central da universidade, o que está além do escopo deste trabalho.

**Palavras-chaves:** Acompanhamento de Desempenho Acadêmico, Marvin, Sistema de Informação, *Web*, Java.

# Lista de ilustrações

Figura 1 – Diferenças entre biblioteca e <i>framework</i> (MALDONADO et al., 2002). . . . .	20
Figura 2 – Relação entre quantidade e custo das fases do teste. . . . .	26
Figura 3 – Diagrama de Pacotes e os Subsistemas Identificados. . . . .	30
Figura 4 – Diagrama de Casos de Uso do subsistema Coordenação de Cursos de Graduação. . . . .	32
Figura 5 – Diagrama de Casos de Uso do subsistema de Submissão de Atividades Acadêmicas. . . . .	34
Figura 6 – Diagrama de classes do subsistema Coordenação de Curso de Graduação. . . . .	36
Figura 7 – Diagrama de classes do subsistema Submissão de Atividades Acadêmicas. . . . .	37
Figura 8 – Arquitetura de Software (SOUZA, 2020). . . . .	40
Figura 9 – Organização de pastas e pacotes do projeto. . . . .	41
Figura 10 – Projeto da Componente de Domínio do Problema ( <i>domain</i> ) do subsistema Coordenação de Curso de Graduação. . . . .	43
Figura 11 – Projeto da Componente de Domínio do Problema ( <i>domain</i> ) do subsistema Submissão de Atividades Acadêmicas. . . . .	44
Figura 12 – Modelo de persistência base do utilitário JButler. . . . .	45
Figura 13 – Projeto da Componente de Gerência de Dados ( <i>persistence</i> ) do subsistema Coordenação de Curso de Graduação. . . . .	46
Figura 14 – Projeto da Componente de Gerência de Dados ( <i>persistence</i> ) do subsistema Submissão de Atividades Acadêmicas. . . . .	47
Figura 15 – Projeto da Camada de Interface com o Usuário ( <i>view e controller</i> ) do subsistema Coordenação de Curso de Graduação, juntamente com a Componente de Gerência de Tarefas ( <i>application</i> ). - Parte 1. . . . .	51
Figura 16 – Projeto da Camada de Interface com o Usuário ( <i>view e controller</i> ) do subsistema Coordenação de Curso de Graduação, juntamente com a Camada de Gerência de Tarefas ( <i>application</i> ). - Parte 2. . . . .	52
Figura 17 – Projeto da Camada de Interface com o Usuário ( <i>view e controller</i> ) do subsistema Submissão de Atividades Acadêmicas, juntamente com a Camada de Gerência de Tarefas ( <i>application</i> ). . . . .	54
Figura 18 – Tela de <i>Login</i> . . . . .	55
Figura 19 – Tela de Início. . . . .	56
Figura 20 – Tela de Início - Menu “Graduação” expandido. . . . .	56
Figura 21 – Tela de Importar Alunos - Submissão de arquivo. . . . .	57
Figura 22 – Tela de Importar Alunos - Listagem e edição de dados. . . . .	58
Figura 23 – Tela de Importar Alunos - Listagem dados importados. . . . .	58
Figura 24 – Tela de Gerenciar PPCs - Listagem. . . . .	59

Figura 25 – Tela de Gerenciar PPCs - Formulário. . . . .	60
Figura 26 – Tela de ADAs - Início. . . . .	60
Figura 27 – Tela de Importar ADAs - Submissão de arquivo. . . . .	61
Figura 28 – Tela de Importar ADAs - Listagem e edição de dados. . . . .	62
Figura 29 – Tela de Importar ADAs - Conflito de versões. . . . .	62
Figura 30 – Tela de Importar ADAs - Listagem dados importados. . . . .	63
Figura 31 – Lista de ADAs para Gerenciamento. . . . .	63
Figura 32 – Tela de Gerenciar PPCs - Listagem. . . . .	64
Figura 33 – Tela de Gerenciar PPCs - Formulário. . . . .	65

# Lista de tabelas

Tabela 1 – Estórias de usuário simplificadas. . . . .	29
Tabela 2 – Subsistemas identificados e suas interdependências. . . . .	31
Tabela 3 – Descrição dos atores envolvidos nos casos de uso. . . . .	31
Tabela 4 – Rastreabilidade dos casos de uso do subsistema Coordenação de Curso de Graduação. . . . .	33
Tabela 5 – Rastreabilidade dos casos de uso do subsistema Submissão de Atividades Acadêmicas. . . . .	34
Tabela 6 – Plataforma de Desenvolvimento e Tecnologias Utilizadas. . . . .	38
Tabela 7 – Softwares de Apoio ao Desenvolvimento do Projeto . . . . .	39
Tabela 8 – Cumprimento dos objetivos delineados no Capítulo 1. . . . .	67

# Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
ADA	Acompanhamento de Desempenho Acadêmico
AJAX	Asynchronous JavaScript and XML
CCI	Componente de Controle de Interação
CDI	Contexts and Dependency Injection
CDP	Componente de Domínio do Problema
CGD	Componente de Gerência de Dados
CGT	Componente de Gerência de Tarefas
CIH	Componente de Interação Humana
CSV	Comma-separated values
DAO	Data Access Object
JPA	Java Persistence API
JSF	JavaServer Faces
JSP	JavaServer Pages
LabES	Laboratório de Práticas em Engenharia de Software “Ricardo de Almeida Falbo”
MVC	Model-View-Controller
PPC	Projeto Pedagógico de Curso
Prograd	Pró-reitoria de Graduação
SPA	Single Page Application
STI	Superintendência de Tecnologia da Informação
UFES	Universidade Federal do Espírito Santo
UML	Unified Modeling Language

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>11</b>
1.1	Motivação e Justificativa	11
1.2	Objetivos	12
1.3	Método de Desenvolvimento do Trabalho	13
1.4	Organização da Monografia	13
<b>2</b>	<b>REFERENCIAL TEÓRICO E TECNOLOGIAS UTILIZADAS</b>	<b>15</b>
2.1	Engenharia de Software	15
2.1.1	Análise e levantamento dos requisitos	17
2.1.2	Projeto	18
2.2	Desenvolvimento <i>Web</i>	19
2.2.1	Frameworks	19
2.2.2	SOLID	21
2.2.3	Arquitetura em Três Camadas	22
2.3	Testes de <i>Software</i>	24
<b>3</b>	<b>ANÁLISE E ESPECIFICAÇÃO DE REQUISITOS</b>	<b>27</b>
3.1	Descrição do Escopo	27
3.2	Estórias de Usuário	28
3.3	Identificação de Subsistemas	30
3.4	Modelos de Casos de Uso	31
3.4.1	Subsistema Coordenação de Cursos de Graduação	32
3.4.2	Subsistema de Submissão de Atividades Acadêmicas	34
3.5	Diagrama de Classes	35
3.5.1	Subsistema de Coordenação de Curso de Graduação	35
3.5.2	Subsistema de Submissão de Atividades Acadêmicas	35
<b>4</b>	<b>PROJETO DO SISTEMA</b>	<b>38</b>
4.1	Tecnologias e Ferramentas Utilizadas	38
4.2	Arquitetura de Software	39
4.3	Projeto dos Componentes da Arquitetura	41
4.3.1	Camada de Negócio	42
4.3.1.1	Subsistema Coordenação de Curso de Graduação	42
4.3.1.2	Subsistema Submissão de Atividades Acadêmicas	43
4.3.2	Camada de Acesso a Dados	44
4.3.2.1	Subsistema Coordenação de Curso de Graduação	44

4.3.2.2	Subsistema Submissão de Atividades Acadêmicas . . . . .	47
4.3.3	Camada de Apresentação . . . . .	47
4.3.3.1	Subsistema Coordenação de Curso de Graduação . . . . .	48
4.3.3.2	Subsistema Submissão de Atividades Acadêmicas . . . . .	53
<b>5</b>	<b>IMPLEMENTAÇÃO DO SISTEMA . . . . .</b>	<b>55</b>
<b>5.1</b>	<b>Importar Alunos . . . . .</b>	<b>57</b>
<b>5.2</b>	<b>Gerenciar Projetos Pedagógicos de Curso . . . . .</b>	<b>58</b>
<b>5.3</b>	<b>Acompanhamento de Desempenho Acadêmico . . . . .</b>	<b>60</b>
<b>6</b>	<b>CONCLUSÃO . . . . .</b>	<b>66</b>
<b>6.1</b>	<b>Considerações Finais . . . . .</b>	<b>66</b>
<b>6.2</b>	<b>Trabalhos Futuros . . . . .</b>	<b>67</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>69</b>
	<b>APÊNDICES . . . . .</b>	<b>71</b>
	<b>APÊNDICE A – DOCUMENTO DE REQUISITOS DE SISTEMA . . . . .</b>	<b>72</b>
	<b>APÊNDICE B – DOCUMENTO DE PROJETO DE SISTEMA . . . . .</b>	<b>114</b>

# 1 Introdução

No contexto acadêmico dos cursos de graduação da Universidade Federal do Espírito Santo (UFES), o controle das diversas tarefas necessárias para o bom funcionamento dos cursos é essencial para alcançar o objetivo de formar indivíduos preparados para o mercado de trabalho ou para a área de pesquisa.

Um dos atores com uma ampla gama de responsabilidades é o Coordenador de Curso de Graduação, que deve estar sempre ciente dos numerosos prazos estabelecidos pela Pró-Reitoria de Graduação (Prograd) para a submissão de relatórios, planejamento curricular, gestão administrativa, entre outros.

Além disso, na UFES, existe o Acompanhamento de Desempenho Acadêmico (ADA), um processo conduzido pelo Coordenador de Curso e enviado à Prograd. O ADA orienta os estudos necessários para a conclusão do curso no prazo estipulado, sendo direcionado a estudantes com baixo desempenho. Ele reúne informações sobre quem precisa de atenção especial, permitindo a implementação de medidas para apoiar esses alunos em sua jornada acadêmica.

Por fim, é fundamental que os vários atores que compõem a comunidade acadêmica interajam entre si. Por exemplo, os alunos devem submeter os comprovantes de suas horas complementares ao Coordenador de Horas do seu curso, para que sejam analisadas e adicionadas ao currículo do aluno, possibilitando sua formatura.

Considerando esses aspectos, é evidente que gerenciar, organizar e estabelecer processos manuais e demorados pode ser desafiador. Para solucionar essas questões, o módulo de gestão de cursos de graduação do Marvin<sup>1</sup> surge como uma ferramenta ágil e eficiente, voltada para estudantes, professores e coordenação da graduação. Esse módulo pode oferecer a seus usuários economia de tempo, organização e centralização na troca de informações entre os envolvidos no ambiente acadêmico.

## 1.1 Motivação e Justificativa

O Marvin é um sistema de informação que vem sendo desenvolvido pelo Laboratório de Práticas em Engenharia de Software “Ricardo de Almeida Falbo” (LabES)<sup>2</sup> ao longo dos anos com o objetivo de auxiliar atividades administrativas relacionadas aos cursos da UFES. Ele reúne diversos projetos de graduação desenvolvidos por estudantes dos cursos de Ciência da Computação e Engenharia de Computação. Neste contexto, foi identificada

<sup>1</sup> <<https://gitlab.labes.inf.ufes.br/marvin/marvin>>

<sup>2</sup> <<http://labes.inf.ufes.br>>



a necessidade de criar um novo módulo que atendesse às principais demandas dos cursos de graduação.

Ao analisar as dificuldades enfrentadas pelos estudantes e os coordenadores de graduação, identificou-se as necessidades de reduzir a carga de trabalho manual, auxiliar na organização de suas tarefas e centralizar o canal de comunicação em uma plataforma que não seja o e-mail.

Diante disso, concluiu-se que a graduação necessita de um sistema de software capaz de auxiliar algumas das tarefas administrativas associadas aos cursos como, por exemplo, gerenciar os diversos prazos que os coordenadores de curso devem cumprir, automatizar a geração do ADA e permitir que a comunidade acadêmica de graduação centralize solicitações de diversas atividades. Com o intuito de atender a essas necessidades, decidiu-se implementar um módulo do sistema Marvin.

O escopo deste módulo é proporcionar um ambiente organizado que facilite as atividades dos acadêmicos. No entanto, para a realização de ações específicas, como submeter o ADA para a Prograd e vincular horas ao currículo do aluno, é necessário utilizar o sistema oficial da UFES. A integração direta com os sistemas da UFES, sob a gestão da Superintendência de Tecnologia da Informação (STI), está fora do escopo deste projeto, sendo uma sugestão para futuras melhorias.

Além disso, no escopo de implementação, são contempladas apenas algumas das atividades de gestão de cursos de graduação, devido à grande quantidade de requisitos levantados junto aos *stakeholders*. A implementação dos demais requisitos levantados pode ser abordada em trabalhos futuros.

## 1.2 Objetivos

Este trabalho tem como objetivo geral a criação de um módulo do sistema Web Marvin que reduza as tarefas manuais executadas por coordenadores de cursos, unifique as funções agora gerenciadas por e-mail e torne os processos acadêmicos mais simples, focando em resolver três problemas principais: Gerenciamento de Prazos, Gerenciamento de ADAs e Submissão de Atividades Acadêmicas. Tal objetivo geral pode ser subdividido nos seguintes objetivos específicos:

1. Levantamento e documentação dos requisitos do módulo, abrangendo funções de oferta de disciplinas, gestão de solicitações (como aproveitamento de créditos e emissão de diplomas) e gerenciamento de prazos;
2. Projeto e documentação da arquitetura do módulo, integrando-a à arquitetura do Marvin;

3. Implementação e teste do código-fonte do módulo, integrando-o à base de código do Marvin. A implementação será restrita à funcionalidade de Acompanhamento de Desempenho Acadêmico, dada sua prioridade e adequação ao cronograma do trabalho.

## 1.3 Método de Desenvolvimento do Trabalho

O método de desenvolvimento adotado para realizar o trabalho é composto pelas seguintes atividades:

1. Revisão bibliográfica: estudo das boas práticas de Engenharia de Software e Engenharia de Requisitos, programação orientada a objetos, desenvolvimento de software Web e projeto de banco de dados relacionais;
2. Documentação do sistema: elaboração dos documentos do sistema, que incluem o Documento de Especificação de Requisitos contendo uma visão geral do produto, descrição do ambiente, *stakeholders*, histórias de usuário, regras de negócio e casos de uso; e o Documento de Projeto de Sistemas, que contém a arquitetura do software a ser desenvolvido e o projeto detalhado de seus componentes, especificando as classes, modelos, atributos e detalhando o comportamento esperado de todos os casos de uso do módulo;
3. Estudo de tecnologias: estudo das tecnologias que serão utilizadas no desenvolvimento do módulo, incluindo a linguagem de programação Java e a plataforma Jakarta EE, o ambiente de desenvolvimento IntelliJ, o banco de dados MySQL, a ferramenta DBeaver, o framework PrimeFaces para o desenvolvimento da interface e o framework JUnit para a criação de testes unitários;
4. Implementação: desenvolvimento do módulo do sistema com todas as funcionalidades descritas, utilizando os princípios do SOLID (que serão abordados ao longo deste trabalho) e realizando testes;
5. Redação da monografia: escrita da monografia utilizando o editor TeXstudio e o template abnTeX, que segue as normas da Associação Brasileira de Normas Técnicas (ABNT) para a produção de documentos técnicos e científicos brasileiros.

## 1.4 Organização da Monografia

A seguir, apresentamos a estrutura dos capítulos subsequentes desta monografia:

- O Capítulo 2 (Referencial Teórico e Tecnologias Utilizadas) explora os fundamentos

teóricos relevantes para a compreensão do trabalho, abrangendo conceitos e teorias que sustentam o seu desenvolvimento;

- O Capítulo 3 (Análise e Especificação de Requisitos) descreve o processo de levantamento e especificação dos requisitos do sistema. Isso inclui os principais recursos e funcionalidades identificados no documento de requisitos que serviram como base para o desenvolvimento do sistema;
- O Capítulo 4 (Projeto do Sistema) detalha o projeto arquitetural do sistema, destacando as tecnologias utilizadas durante o processo de desenvolvimento. Além disso, explora as etapas de implementação e como as funcionalidades foram traduzidas em código;
- O Capítulo 5 (Implementação do Sistema) oferece uma visão prática do sistema, apresentando capturas de tela que ilustram as funcionalidades desenvolvidas e a interface do usuário;
- O Capítulo 6 (Conclusão) conclui a monografia com uma revisão das principais descobertas, implicações e lições aprendidas durante o desenvolvimento do sistema. Além disso, aponta possíveis áreas para trabalhos futuros.
- O Apêndice A (Documento de Requisitos de Sistema) contém todos os documentos elaborados durante a fase de Análise e Especificação de Requisitos;
- O Apêndice B (Documento de Projeto de Sistema) engloba toda a documentação criada durante a fase de Planejamento do Sistema.

Cada capítulo contribui para uma compreensão mais profunda do projeto, abordando aspectos teóricos, técnicos e práticos relacionados ao desenvolvimento do sistema. A estrutura da monografia foi projetada para fornecer uma visão completa e organizada do trabalho realizado.

## 2 Referencial Teórico e Tecnologias Utilizadas

Para o desenvolvimento do módulo, foi realizado um levantamento bibliográfico abrangendo diversos tópicos relevantes da Engenharia de Software. Esse estudo foi essencial para garantir a aplicabilidade do resultado final e embasar teoricamente todos os conceitos utilizados no desenvolvimento.

De forma resumida, as seções foram divididas da seguinte maneira: na Seção 2.1 abordamos a Engenharia de Software e suas principais etapas; na Seção 2.2 discutimos diversos conceitos relacionados ao desenvolvimento de sistemas *Web*; e, por fim, na Seção 2.3 exploramos testes e estratégias para aumentar a confiabilidade dos códigos desenvolvidos.

### 2.1 Engenharia de Software

Os primeiros computadores construídos na década de 1940 não possuíam *software*. Os comandos eram inseridos na máquina através de ligações físicas entre os componentes. Com o tempo, surgiu a necessidade de computadores mais flexíveis, o que levou ao desenvolvimento de *softwares*, que consistem em conjuntos de instruções que permitem que a máquina execute processamentos (WAZLAWICK, 2013).

Atualmente, houve um avanço significativo no desenvolvimento de *software*. A área continua evoluindo constantemente, acompanhando a crescente demanda dos usuários. Para lidar com essas demandas, surgiu a Engenharia de Software, que engloba aspectos relacionados ao estabelecimento de processos, métodos, técnicas, ferramentas e ambientes de apoio ao desenvolvimento de *software* (BARCELLOS, 2018).

A área da Engenharia de Software é vasta, especialmente considerando a quantidade de diferentes tipos de *softwares* existentes em nosso cotidiano, muitos dos quais nem sequer imaginamos. Em geral, considerando a diversidade de tipos de *software*, existem múltiplas maneiras de classificar os sistemas. Uma dessas classificações, por exemplo, é a proposta por Wazlawick (2013):

- *Software* básico: inclui compiladores, drivers e componentes do sistema operacional;
- *Software* de tempo real: são sistemas que monitoram, analisam e controlam eventos do mundo real. Já Falbo (2018) define esta classe de sistemas como tendo fortes restrições de tempo, frequentemente projetados para operarem próximos de seus limites;
- *Software* comercial: engloba sistemas aplicados em empresas, como controle de

estoque e vendas, geralmente com acesso a bancos de dados. Também são conhecidos como sistemas de informação;

- *Software* científico e de engenharia: são sistemas que envolvem processamento intenso de números;
- *Software* embutido ou embarcado: refere-se a sistemas de *software* presentes em dispositivos como celulares, eletrodomésticos, automóveis etc. Geralmente, esses sistemas operam sob severas restrições de espaço, tempo de processamento e consumo de energia;
- *Software* pessoal: abrange sistemas usados por indivíduos no dia a dia, como processadores de texto e planilhas;
- Jogos: embora alguns jogos possuam processamento menos complexo, há aqueles que demandam alto desempenho dos computadores devido à qualidade gráfica e à necessidade de reação em tempo real. No entanto, todas as categorias de jogos possuem características intrínsecas que extrapolam o domínio da Engenharia de *Software*;
- Inteligência artificial: envolve sistemas especialistas, redes neurais e sistemas capazes de aprendizado. Além de serem sistemas independentes, com seus próprios processos de construção, também podem ser incorporados a outros sistemas.

O desenvolvimento do módulo abordado neste trabalho é classificado como um *Software* comercial (Sistema de Informação), portanto, os métodos e práticas abordados serão direcionados a esse tipo de sistema. No entanto, é importante ressaltar que algumas das técnicas apresentadas também podem ser aplicadas no desenvolvimento de outros tipos de *software*.

O desenvolvimento de *software* compreende um conjunto de atividades que têm como objetivo principal criar e entregar um produto de *software* ao cliente, juntamente com sua documentação. Esse processo geralmente envolve etapas como Análise e Especificação de Requisitos, Projeto, Implementação, Testes, Entrega e Implantação do Sistema (BARCELLOS, 2018).

Nesse contexto, o desenvolvimento do módulo em questão abrange quase todas as fases do processo de desenvolvimento de *software*, desde a fase de Análise até a Entrega. No entanto, a implantação do sistema está além do escopo deste trabalho, uma vez que é uma área vasta e que exigiria um considerável investimento de tempo e recursos adicionais.

### 2.1.1 Análise e levantamento dos requisitos

De acordo com a definição ISO/IEC/IEEE (2010) citada por [Vazquez e Simões \(2016\)](#), um requisito pode ser definido como:

- (a) Uma condição ou capacidade do sistema solicitada por um usuário para resolver um problema ou atingir um objetivo;
- (b) Uma condição ou capacidade que deve ser atendida por uma solução para cumprir um contrato, especificação, padrão ou qualquer outro documento formalmente imposto;
- (c) A documentação que representa as condições ou capacidades mencionadas nos dois itens anteriores;
- (d) Uma condição ou capacidade que deve ser alcançada ou possuída por um sistema, produto, serviço, resultado ou componente para satisfazer um contrato, padrão, especificação ou outro documento formalmente imposto. Os requisitos incluem as necessidades quantificadas e documentadas, desejos e expectativas do patrocinador, clientes e outras partes interessadas.

O levantamento de requisitos corresponde à fase inicial do processo de desenvolvimento e envolve atividades de descoberta dos requisitos. Nessa fase, é necessário um esforço conjunto entre clientes, usuários e especialistas de domínio, com o objetivo de compreender a organização, seus processos, necessidades, deficiências nos sistemas de *software* existentes, possibilidades de melhorias e restrições existentes ([BARCELLOS, 2018](#)).

Um resultado comum dessa análise de requisitos é a listagem dos requisitos funcionais e não funcionais do sistema. [Barcellos \(2018\)](#) descreve esses conceitos da seguinte forma:

- Requisitos funcionais: são declarações de serviços que o sistema deve fornecer, descrevendo o que o sistema deve fazer. Os requisitos funcionais descrevem as interações entre o sistema e seu ambiente, podendo especificar como o sistema deve reagir a entradas específicas, como deve se comportar em situações específicas e o que não deve fazer;
- Requisitos não funcionais: descrevem restrições sobre os serviços ou funções oferecidos pelo sistema, limitando as opções para criar uma solução para o problema. Esses requisitos não funcionais são de grande importância para a fase de projeto, servindo como base para tomada de decisões nessa etapa.

### 2.1.2 Projeto

O projeto de *software* é o processo criativo que transforma a especificação de um problema em uma especificação de solução. Ele utiliza a especificação de requisitos e os modelos conceituais gerados na fase de levantamento e análise de requisitos. A partir dos requisitos, várias soluções são possíveis, resultando em diferentes projetos. Uma solução é considerada adequada se satisfizer os requisitos especificados. Portanto, o projeto também envolve atividades de tomada de decisão (FALBO, 2018).

Em relação ao Projeto de *Software*, alguns aspectos relevantes são qualidade, arquitetura, padrões e documentação (BARCELLOS, 2018).

Um bom projeto de *software* deve apresentar características de qualidade, como facilidade de entendimento, implementação, testes e modificação, além de traduzir corretamente as especificações de requisitos e análise. Considerar os níveis de abstração, modularidade, ocultação de informações e independência funcional é importante para avaliar a qualidade do projeto de *software*.

A arquitetura define os elementos ou módulos de *software* e como eles se relacionam entre si. Pode envolver diferentes tipos de estruturas e relacionamentos. Em sistemas modernos, os elementos interagem por meio de interfaces que dividem detalhes em partes pública e privada. A arquitetura trata principalmente da parte pública dessa divisão.

A reutilização é fundamental no desenvolvimento de *software*, pois muitos sistemas anteriores podem ser similares ao sistema em desenvolvimento. O conhecimento prévio pode ser reutilizado para resolver problemas recorrentes. Os padrões capturam esse conhecimento, tornando-o mais geral e amplamente aplicável. Um padrão é uma solução testada e aprovada para um problema comum.

A documentação do projeto de *software* é essencial para o sucesso do projeto e para a manutenção futura do sistema. Diferentes partes interessadas requerem informações específicas e a documentação adequada é crucial para facilitar a comunicação. Além disso, um projeto de sistema é uma entidade complexa que não pode ser descrita a partir de uma única perspectiva. Portanto, a documentação deve abranger várias visões consideradas relevantes.

Por fim, é importante conhecer a tecnologia disponível e os ambientes de hardware e *software* nos quais o sistema será desenvolvido e implantado. Durante o projeto, as decisões são tomadas para resolver o problema, começando em um nível de abstração mais alto, próximo à análise, e progredindo para níveis mais detalhados, até chegar a um nível de abstração próximo à implementação (FALBO, 2018).

## 2.2 Desenvolvimento *Web*

A Internet é uma infraestrutura de informação generalizada, considerada o protótipo inicial da Infraestrutura Internacional de Informação. Sua história é complexa, abrangendo diversos aspectos tecnológicos, organizacionais e comunitários. Seu impacto se estende para além das áreas técnicas das comunicações de computador, alcançando toda a sociedade à medida que avançamos rumo ao aumento do uso de ferramentas online para realizar atividades como comércio eletrônico, busca de informações e operações comunitárias (BADALOTTI, 2018).

Ao considerar o quão diferente a Internet é hoje em relação a uma década atrás, fica evidente o nível de complexidade adquirido pelas aplicações *Web* e a rapidez com que as mudanças ocorreram. Com frequência, essa complexidade torna as aplicações difíceis de manter e aumenta seu custo ao longo do tempo. Alguns fatores que contribuem para essa complexidade, de acordo com Loudon (2018), incluem a disponibilidade contínua, uma grande base de usuários, a entrega em partes, a diversidade de tecnologias, a longevidade necessária das aplicações, a necessidade de operar em múltiplos ambientes e a demanda por atualizações em tempo real.

É evidente, portanto, a complexidade do desenvolvimento *Web* e a quantidade de aspectos a serem considerados ao iniciar um projeto de *software* para a Internet. O projetista dessas aplicações deve ter uma ampla gama de recursos e experiência, conhecendo diversas tecnologias, *frameworks* e padrões arquiteturais, a fim de ter sucesso na criação de um *software* que possa ser utilizado por um grande número de pessoas e atenda às expectativas de todos os *stakeholders*.

### 2.2.1 Frameworks

A utilização de *frameworks* no desenvolvimento de sistemas torna-se essencial em muitas ocasiões, permitindo que a equipe de desenvolvedores alcance resultados que atendam a todos os requisitos de funcionalidades e prazos de um projeto. Além desses benefícios, a adoção de *frameworks* traz outras vantagens, como código padronizado, segurança e a disponibilidade de diversos parâmetros de configuração que podem ser facilmente ajustados, o que permite que o desenvolvedor concentre-se nas regras de negócio. Maldonado et al. (2002) afirmam: “Com os *frameworks*, reutilizam-se não somente as linhas de código, mas também o projeto abstrato envolvendo o domínio de aplicação”.

De acordo com a definição de Maldonado et al. (2002), um *framework* é o projeto de um conjunto de objetos que colaboram entre si para executar um conjunto de responsabilidades. Um *framework* reutiliza análise, projeto e código. Ele reutiliza a análise, pois descreve os tipos de objetos importantes e como um problema maior pode ser dividido em problemas menores. Ele reutiliza o projeto, pois contém algoritmos abstratos e descreve



a interface que o programador deve implementar. Ele reutiliza o código, facilitando o desenvolvimento de uma biblioteca de componentes compatíveis.

Uma confusão comum é a diferenciação entre bibliotecas e *frameworks*. Uma das diferenças fundamentais entre esses dois conceitos é que o *framework* geralmente desempenha o papel de programa principal, coordenando e sequenciando as atividades da aplicação. A Figura 1 ilustra algumas diferenças básicas entre bibliotecas e *frameworks*.

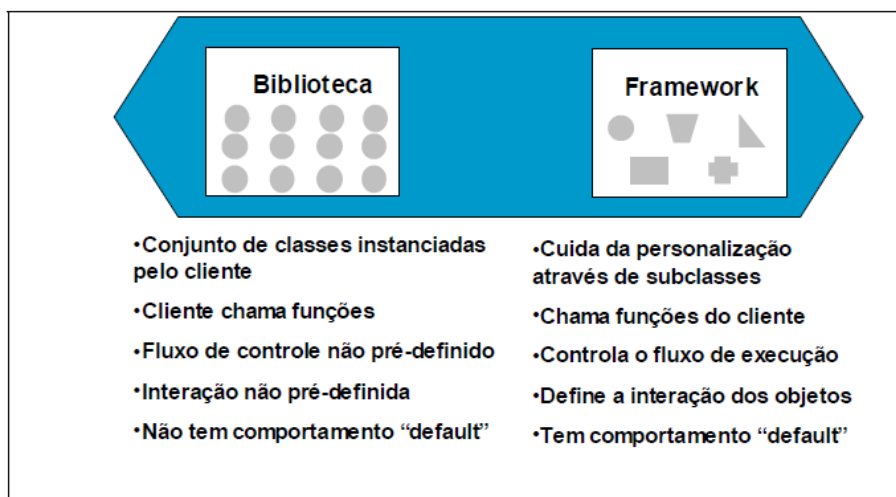


Figura 1 – Diferenças entre biblioteca e *framework* (MALDONADO et al., 2002).

Para o desenvolvimento do módulo em questão, utilizaremos a plataforma *Jakarta Enterprise Edition* (Jakarta EE), anteriormente conhecida como *Java Enterprise Edition* (Java EE). Essa mudança teve como principal objetivo manter o Java corporativo atualizado em relação às tendências de mercado, o que foi bem recebido tanto pelo mercado quanto pelos desenvolvedores.

A plataforma *Jakarta EE* é um padrão para o desenvolvimento de aplicações Java de grande porte e voltadas para a Internet, fornecendo um conjunto abrangente de bibliotecas e funcionalidades para implementar software Java distribuído (FARIA, 2015). Essa plataforma é conhecida como um “guarda-chuva”, pois engloba várias especificações e *frameworks* que auxiliam e padronizam o desenvolvimento de aplicações. Algumas das especificações mais relevantes incluem:<sup>1</sup>

- (*Jakarta*) *Servlets*: são componentes Java executados no servidor para gerar conteúdo dinâmico para a *Web*, como HTML e XML;
- *JavaServer Pages* (JSP) / *Jakarta Pages*: uma especialização de *Servlets* que simplifica a manutenção de aplicações *Web* desenvolvidas em Java;

<sup>1</sup> São apresentados os nomes criados à época do Java EE, por serem ainda amplamente utilizados, bem como os novos nomes dados às tecnologias após sua migração para Jakarta EE, c.f. <<https://jakarta.ee/specifications/>>.

- *JavaServer Faces* (JSF) / *Jakarta Faces*: é um *framework Web* baseado em Java que visa simplificar o desenvolvimento de interfaces de sistemas para a *Web*, utilizando um modelo de componentes reutilizáveis;
- *Java Persistence API* (JPA) / *Jakarta Persistence*: é uma *Application Programming Interface* (API) padrão do Java para persistência de dados, que utiliza o conceito de mapeamento objeto/relacional;
- *Enterprise Java Beans* (EJB) / *Jakarta Enterprise Beans*: são componentes executados em servidores de aplicação, projetados para facilitar o desenvolvimento de componentes distribuídos, transacionais, seguros e portáteis;
- (*Jakarta*) *Contexts and Dependency Injection* (CDI): define um conjunto de serviços que aprimoram a estrutura do código da aplicação, incluindo um ciclo de vida para objetos com estado e um mecanismo de injeção de dependência com segurança de tipos.

Essas especificações e *frameworks* fornecem uma base sólida e robusta para o desenvolvimento de aplicações Java, permitindo que os desenvolvedores se concentrem na lógica de negócios e aproveitem os benefícios de um código padronizado, segurança aprimorada e configurações flexíveis. A plataforma Jakarta EE continua evoluindo para atender às demandas em constante mudança do mercado e permanece como uma escolha confiável para o desenvolvimento de aplicações empresariais escaláveis e de alta qualidade.

## 2.2.2 SOLID

Durante a fase de levantamento de requisitos, um dos requisitos não funcionais que mais são pedidos é o da Manutenibilidade. Esse requisito estabelece que o sistema deve ser facilmente mantido, seguindo padrões de *design* e utilizando controle de versões. Essa exigência é essencial, pois é comum que os *stakeholders* solicitem mudanças ou melhorias durante o desenvolvimento ou após a entrega. Para garantir essa manutenibilidade, é fundamental que o *software* tenha um código limpo, e é nesse ponto que entram em jogo os princípios SOLID (MARTIN, 2019).

Os princípios SOLID servem como um guia para orientar os programadores na organização de funções, classes e estruturas de dados, bem como nas interconexões entre esses elementos. É importante ressaltar que o termo “classe” não se limita apenas a *softwares* que utilizam linguagens orientadas a objetos. Uma classe pode ser vista como um agrupamento coeso de funções e dados. Todos os sistemas de *software* possuem esses agrupamentos, independentemente de serem ou não chamados de classes. Os princípios SOLID se aplicam a esses agrupamentos (MARTIN, 2019).

No entanto, para utilizar esses princípios, é necessário compreender cada um deles. SOLID é um acrônimo formado por cinco princípios diferentes que começaram a ser reunidos no final da década de 1980 por [Martin \(2019\)](#). Por volta de 2004, esses princípios foram reorganizados e popularizados sob o nome de SOLID. São eles:

- Princípio da Responsabilidade Única (*Single Responsibility Principle*): esse princípio é um corolário ativo da lei de Conway. Ele estabelece que a melhor estrutura para um sistema de *software* deve ser fortemente influenciada pela estrutura social da organização que o utiliza, de forma que cada módulo de *software* tenha apenas uma razão para mudar;
- Princípio do Aberto/Fechado (*Open-Closed Principle*): popularizado por Bertrand Meyer na década de 1980, esse princípio estabelece que os sistemas de *software* devem ser projetados de maneira a permitir que seu comportamento seja alterado pela adição de novo código, em vez de modificar o código existente, tornando-os mais fáceis de serem alterados;
- Princípio de Substituição de Liskov (*Liskov Substitution Principle*): esse princípio estabelece uma definição famosa de subtipos, proposta por Barbara Liskov em 1988. Resumidamente, ele afirma que, para criar sistemas de *software* com partes intercambiáveis, essas partes devem aderir a um contrato que permita sua substituição mútua;
- Princípio da Segregação de Interface (*Interface Segregation Principle*): esse princípio orienta os projetistas de *software* a evitarem depender de coisas que não utilizam;
- Princípio da Inversão de Dependência (*Dependency Inversion Principle*): esse princípio estabelece que o código que implementa políticas de alto nível não deve depender do código que implementa detalhes de baixo nível. São os detalhes que devem depender das políticas.

Portanto, é crucial que durante o processo de desenvolvimento de um *software*, a consideração da utilização dos princípios SOLID seja, no mínimo, levada em conta, a fim de possibilitar uma evolução saudável e adaptável às mudanças. Vários desses princípios serão aplicados ao longo do desenvolvimento do módulo.

### 2.2.3 Arquitetura em Três Camadas

A arquitetura de *software* engloba as decisões de projeto mais cruciais em um sistema. Essas decisões são tão significativas que, uma vez tomadas, raramente podem ser revertidas no futuro. Portanto, essa segunda definição de arquitetura é mais abrangente do

que a primeira que apresentamos. Ela considera que a arquitetura não se resume apenas a um conjunto de módulos, mas a um conjunto de decisões.

Um dos padrões arquiteturais amplamente utilizados é a Arquitetura em Três Camadas. Segundo [Valente \(2020\)](#), essas camadas são divididas da seguinte forma:

- Interface com o Usuário ou Camada de Apresentação: lida tanto com a exibição de informações quanto com a coleta e processamento de entradas e eventos de interface, como cliques em botões, marcação de texto, etc.;
- Lógica de Negócio ou Camada de Aplicação: implementa as regras de negócio do sistema;
- Banco de Dados: armazena os dados manipulados pelo sistema.

Normalmente, uma arquitetura em três camadas é distribuída. Isso significa que a camada de interface é executada nas máquinas dos clientes, a camada de negócio é executada em um servidor, frequentemente chamado de servidor de aplicação, e temos o banco de dados.

Com as camadas da nossa aplicação definidas, podemos usar padrões arquiteturais adicionais dentro de cada uma delas. Na camada de Apresentação, um padrão amplamente utilizado é o MVC (*Model-View-Controller*) ([VALENTE, 2020](#)). Esse padrão divide as classes do sistema com base em suas responsabilidades e descreve como deve ocorrer a interação entre elas.

As classes de Visão (*View*) são responsáveis pela apresentação gráfica, as classes de Controlador (*Controller*) tratam e interpretam eventos gerados por dispositivos de entrada, e as classes de Modelo (*Model*) lidam com o armazenamento dos dados manipulados. Dessa forma, estabelecemos um fluxo que se inicia na Visão, passa pelos Controladores e chega ao Modelo.

Ao considerar a camada de Banco de Dados, um padrão bastante interessante que pode ser utilizado é o *Data Access Object* (DAO). O DAO visa desacoplar o acesso aos dados do seu armazenamento subjacente. A persistência de dados depende fortemente do tipo de banco de dados utilizado, como banco de dados relacional, banco de dados orientado a objetos, arquivos, entre outros. É preferível escolher o tipo de banco de dados durante a fase de implantação, em vez da fase de design. Ao utilizar DAOs, os dados são desacoplados de sua representação, permitindo a escolha de diferentes fontes de dados, se necessário ([BERGER, 2005](#)).

Portanto, para cada entidade manipulada em nosso sistema, temos uma classe DAO correspondente que sabe exatamente como realizar a persistência e a recuperação da entidade no banco de dados escolhido. Isso significa que, se for necessário alterar a fonte

dos dados, basta adaptar o DAO para que ele seja capaz de executar todas as operações necessárias novamente.

## 2.3 Testes de *Software*

Mesmo que as técnicas de modelagem e especificação de *software* sejam avançadas e a equipe de desenvolvimento seja disciplinada e experiente, é inevitável a ocorrência de erros humanos. É um equívoco acreditar que bons desenvolvedores, mesmo altamente concentrados e utilizando boas ferramentas, possam criar *software* sem erros (WAZLAWICK, 2013).

A indústria de *software* parece ser diretamente afetada pela Lei de Murphy, que pode ser resumida da seguinte forma:

- (a) Se algo pode dar errado, dará (no pior momento possível);
- (b) Se tudo parece estar indo bem, é porque você não olhou direito;
- (c) A natureza está sempre a favor de falhas ocultas.

Nesse contexto, é necessário antecipar e mitigar os problemas que podem ocorrer em um *software* durante o seu uso. Para isso, foi desenvolvida a prática de criação de testes de *software*. Esses testes têm como objetivo mapear erros, defeitos, falhas e equívocos que um *software* possa apresentar, elaborando suítes de teste que forçam a ocorrência desses casos para verificar se o *software* se comporta conforme o esperado.

É importante ter em mente que existem diversos tipos de *softwares*, e cada um possui particularidades a serem levadas em consideração na hora de criar seus testes. Para isso, Delamaro, Jino e Maldonado (2013) apresentam diversas técnicas de teste, incluindo:

- Teste Funcional;
- Teste Baseado em Modelos;
- Teste Estrutural;
- Teste de Mutação;
- Teste Orientado a Objetos e de Componentes;
- Teste de Aspectos e Teste Apoiado por Aspectos;
- Teste de Aplicações *Web*;
- Teste de Programas Concorrentes;

- Teste em Dispositivos Móveis.

Durante o desenvolvimento do módulo proposto neste trabalho, serão aplicados os conceitos para a criação de testes orientados a objetos e de componentes, devido ao fato de o sistema utilizar esse paradigma de programação.

Quando testamos uma aplicação orientada a objetos, ou qualquer outra aplicação, é recomendável dividir a atividade de teste em diferentes fases. Essa abordagem permite que o testador concentre-se em aspectos distintos do software e em diversos tipos de defeitos, além de utilizar estratégias variadas para seleção de dados de teste e medidas de cobertura em cada fase específica (DELAMARO; JINO; MALDONADO, 2013). As principais fases de teste incluem o teste de unidade, teste de integração e teste de sistema.

O teste de unidade concentra-se nas unidades menores do programa, como funções, procedimentos, métodos ou classes. Seu objetivo é identificar erros relacionados a algoritmos incorretos, implementações malfeitas, estruturas de dados inadequadas ou erros simples de programação. O teste de unidade pode ser realizado à medida que as unidades são implementadas, permitindo que o próprio desenvolvedor conduza os testes sem a necessidade de ter o sistema completamente finalizado.

No teste de integração, que ocorre após o teste individual das unidades, o foco está na construção da estrutura do sistema. É nessa fase que as diferentes partes do *software* são combinadas e devem funcionar em conjunto, sem erros ou conflitos.

Após a conclusão do desenvolvimento do sistema, com todas as partes integradas, é realizado o teste de sistema. O objetivo dessa fase é verificar se as funcionalidades especificadas nos documentos de requisitos foram implementadas corretamente. Além disso, são explorados aspectos de correção, completude e coerência, bem como requisitos não funcionais, incluindo segurança, desempenho e robustez.

A Figura 2 ilustra os tipos de testes em forma de uma pirâmide, demonstrando que a quantidade de testes unitários em um sistema é muito maior do que a quantidade de testes de sistema. Além disso, o custo de recursos e tempo para executar os diferentes tipos de testes varia significativamente.



Figura 2 – Relação entre quantidade e custo das fases do teste.

## 3 Análise e Especificação de Requisitos

Este capítulo apresenta a especificação dos requisitos do sistema *Módulo de otimização e centralização das atividades acadêmicas da Graduação do Marvin*. Esta especificação foi construída aplicando-se técnicas de levantamento de requisitos, bem como modelagem de casos de uso e de classes utilizando a linguagem UML (FALBO, 2018).

O levantamento de requisitos foi realizado por meio de entrevistas com o cliente. Em reuniões detalhadas, o cliente demonstrava os processos que ocorriam atualmente e como imaginava um sistema que o auxiliasse nessas tarefas. Durante as reuniões, surgiam dúvidas que eram esclarecidas. Foram realizadas várias reuniões, com intervalos de dias entre elas, para consolidar todas as informações.

### 3.1 Descrição do Escopo

Este trabalho propõe um módulo destinado a simplificar a gestão das atividades acadêmicas na Universidade Federal do Espírito Santo (UFES), uma tarefa que atualmente apresenta vários desafios para toda a comunidade acadêmica. Uma base sólida para este trabalho é o projeto de graduação de Chane (2022), que resultou na criação do “Macs – Sistema de controle de acessos do Marvin”. Este projeto fornece uma descrição detalhada dos diversos papéis de usuários presentes no Marvin, contribuindo significativamente para o desenvolvimento do módulo proposto.

O módulo tem como objetivo atender às necessidades associadas ao Gerenciamento de Prazos. Considerando a diversidade de prazos com que os coordenadores precisam lidar — desde tarefas únicas até as recorrentes, com datas de início e término específicas e até prazos personalizados — o módulo propõe simplificar essa gestão complexa.

Para isso, oferecerá uma interface intuitiva que possibilitará o cadastro, edição e organização de prazos. Ademais, os coordenadores poderão importar prazos a partir de arquivos no formato CSV, o que proporcionará uma solução mais dinâmica e eficaz.

Outro aspecto crucial abordado neste módulo é o auxílio no desenvolvimento do Acompanhamento do Desempenho Acadêmico (ADA). O ADA é um programa fundamental na UFES para identificar estudantes com dificuldades em suas disciplinas e prover um acompanhamento direcionado. Contudo, a identificação dos alunos que necessitam deste suporte é um desafio que atualmente exige um esforço manual significativo por parte dos Coordenadores de Curso de Graduação.

Visando resolver essa questão, o módulo automatizará esse processo, possibilitando que os coordenadores importem informações de vários alunos e realizem cálculos auto-



máticos. Os resultados serão apresentados para validação e edição, tornando o processo mais eficiente. A análise qualitativa, indispensável para determinar as medidas a serem adotadas para cada aluno, continuará a ser realizada pelos coordenadores, preservando o elemento humano vital nesse processo.

Por último, o módulo irá lidar com a Submissão de Atividades Acadêmicas. No meio acadêmico, existem diversos processos que implicam a submissão de documentos, como o envio, aprovação, rejeição e arquivamento. Atualmente, essas submissões são realizadas por meio de caixas de e-mail específicas, o que resulta em períodos de alta demanda e dificuldades de gerenciamento.

O módulo proposto irá centralizar esses processos, fornecendo formulários personalizados para alunos e professores preencherem com as informações necessárias. Adicionalmente, o sistema automatizará as notificações por e-mail, mantendo todos os envolvidos atualizados sobre o progresso de suas submissões.

Dessa forma, o módulo proposto neste trabalho visa oferecer uma solução para atender às necessidades da comunidade acadêmica da UFES. O objetivo é simplificar suas tarefas, reduzir a carga de trabalho manual e aumentar a eficiência na gestão das atividades acadêmicas.

## 3.2 Estórias de Usuário

Uma *User Story*, ou estória de usuário, é uma descrição informal e abrangente de uma funcionalidade de *software*, escrita sob a perspectiva do usuário final. Ela visa expressar como o sistema pode beneficiá-lo. Também são definidos critérios de aceitação para cada estória, que orientam o desenvolvimento do *software*, garantindo que a funcionalidade atenda às expectativas. Esta abordagem é instrumental no levantamento de funcionalidades do sistema, fornecendo *insights* valiosos para a concepção de uma aplicação.

Os principais *stakeholders* considerados neste trabalho são coordenadores, professores e alunos. A partir dessa análise, identificamos os requisitos e as regras de negócio, que foram detalhados no Documento de Requisitos, disponível no apêndice deste trabalho.

A Tabela 1 apresenta as estórias de usuário, embora sem os critérios de aceitação correspondentes. As versões completas dessas estórias podem ser encontradas no Apêndice A. Cabe ressaltar que essas estórias foram elaboradas com base nas funcionalidades do sistema, que são descritas de forma resumida na Seção 3.1.

Tabela 1 – Estórias de usuário simplificadas.

Identificador	Descrição
US-1	Cadastro/CRUD: Como Coordenador de Curso, quero cadastrar prazos, para acompanhar suas datas e atividades relacionadas.
US-2	Como Coordenador de Curso, quero importar prazos de um arquivo CSV, para agilizar o cadastro de vários prazos.
US-3	Como Coordenador de Curso, quero visualizar meus prazos, para gerenciar e dar prioridades aos meus prazos cadastrados.
US-4	Cadastro/CRUD: Como Coordenador de Curso, quero cadastrar um Projeto Pedagógico de Curso (PPC), para ter as informações referentes às atividades que um Aluno precisa cumprir para se formar em um curso.
US-5	Como Coordenador de Curso, quero importar informações de Alunos a partir de um arquivo, para facilitar o cadastro em massa de Alunos.
US-6	Como Coordenador de Curso, quero computar as informações do Acompanhamento de Desempenho Acadêmico (ADA) de um Aluno, para identificar Alunos que precisam de acompanhamento ou desligamento.
US-7	Como Coordenador de Curso, desejo consultar todas as informações do ADA (Acompanhamento de Desempenho Acadêmico) dos Alunos cadastrados no sistema, para poder realizar o acompanhamento e observações necessárias.
US-8	Como Coordenador de Curso, quero exportar um arquivo CSV das informações computadas referentes ao ADA, para manipular e compartilhar com outras pessoas que não utilizam o sistema.
US-9	Como Coordenador de Curso, quero modificar o ADA de um Aluno, para corrigir erros, criar exceções e/ou finalizar seu status.
US-10	Como Aluno, quero realizar uma submissão de horas, para que posteriormente o Coordenador de Horas valide-as no sistema da UFES.
US-11	Como Aluno, quero consultar todas as minhas horas, para saber se elas foram aprovadas ou não.
US-12	Como Coordenador de Horas, quero consultar todas as horas que os Alunos cadastraram, para aprová-las ou não.
US-13	Como Coordenador de Horas, quero aprovar ou rejeitar uma submissão de horas, para que o Aluno tenha um feedback sobre sua solicitação.

Identificador	Descrição
US-14	Como Coordenador de Horas, quero arquivar um cadastro de horas, para finalizar o processo.
US-15	Como Aluno, quero realizar uma submissão de estágio, para que posteriormente o Coordenador de Estágio valide-as no sistema da UFES.
US-16	Como Aluno, quero consultar todos os meus estágios, para saber se eles foram aprovados ou não.
US-17	Como Coordenador de Estágio, quero consultar todos os estágios que os Alunos submeteram, para aprová-los ou não.
US-18	Como Coordenador de Estágio, quero aprovar um cadastro de estágio, para que o Aluno tenha um feedback sobre sua solicitação.
US-19	Como Coordenador de Estágio, quero arquivar um cadastro de estágio, para finalizar o processo.

### 3.3 Identificação de Subsistemas

Com o objetivo de atender aos requisitos definidos no escopo do sistema e para melhor organizar o projeto, o *Módulo de Otimização e Centralização das Atividades Acadêmicas de Graduação* foi subdividido em dois subsistemas: Coordenação de Curso de Graduação e Submissão de Atividades Acadêmicas.

A Figura 3 apresenta uma representação dos subsistemas, bem como suas interconexões com outros subsistemas já existentes no Marvin. Para uma compreensão mais detalhada de cada subsistema identificado, a Tabela 2 fornece uma breve descrição de cada um deles.

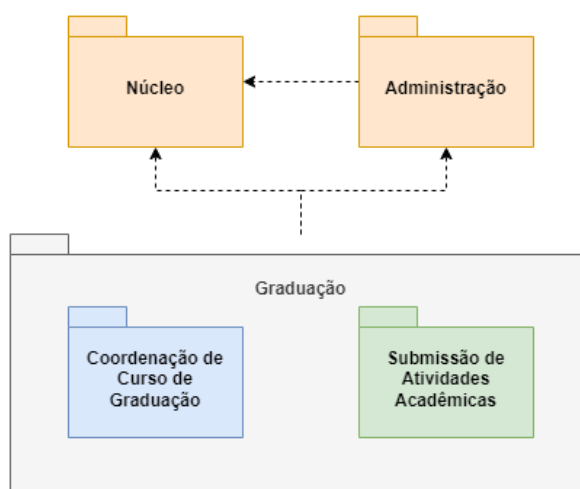


Figura 3 – Diagrama de Pacotes e os Subsistemas Identificados.

Tabela 2 – Subsistemas identificados e suas interdependências.

Subsistema	Descrição
Núcleo	Módulo responsável pelo gerenciamento do cadastro e autenticação de acadêmicos.
Administração	Módulo responsável pelo cadastro das unidades organizacionais da UFES (centros, departamentos, programas de pós-graduações, cursos) (CHANE, 2022).
Graduação	Módulo responsável pela gestão de cursos de graduação.
Coordenação de Curso de Graduação	Subsistema responsável por funcionalidades específicas para o Coordenador de Curso no sistema.
Submissão de Atividades Acadêmicas	Subsistema responsável pela gestão das submissões de atividades acadêmicas entre diferentes perfis de acadêmicos, com um fluxo específico.

### 3.4 Modelos de Casos de Uso

O modelo de casos de uso visa descrever a relação das funcionalidades do sistema com cada um de seus atores. A Tabela 3 descreve os atores identificados no contexto deste projeto.

Tabela 3 – Descrição dos atores envolvidos nos casos de uso.

Ator	Descrição
Coordenador de Curso	Acadêmico responsável por coordenar um ou mais cursos de graduação, utilizando o sistema para gerenciar os cursos sob sua supervisão.
Aluno	Acadêmico que assume o papel de aluno, utilizando o sistema para submeter atividades acadêmicas que contribuem para sua formação.
Coordenador de Horas	Acadêmico responsável por coordenar as horas complementares e de extensão de um curso de graduação, utilizando o sistema para acompanhar as submissões dessas horas feitas pelos alunos.
Coordenador de Estágio	Acadêmico responsável por coordenar os estágios de um curso de graduação, utilizando o sistema para acompanhar as submissões de estágios feitas pelos alunos.

Conforme descrito anteriormente, todos os usuários são inicialmente registrados no sistema como acadêmicos. Posteriormente, através do módulo de Chane (2022), esses acadêmicos são atribuídos a diferentes funções no sistema, relativas a certas unidades organizacionais (ex.: um curso de graduação), como as citadas na Tabela 3. Com essas

funções, os usuários podem então utilizar as funcionalidades oferecidas pelo *Módulo de Otimização e Centralização das Atividades Acadêmicas de Graduação*. Os detalhes das funcionalidades de cada um dos subsistemas serão apresentados a seguir.

### 3.4.1 Subsistema Coordenação de Cursos de Graduação

A Figura 4 apresenta a representação em conformidade com o padrão UML dos casos de uso do subsistema Coordenação de Cursos de Graduação.

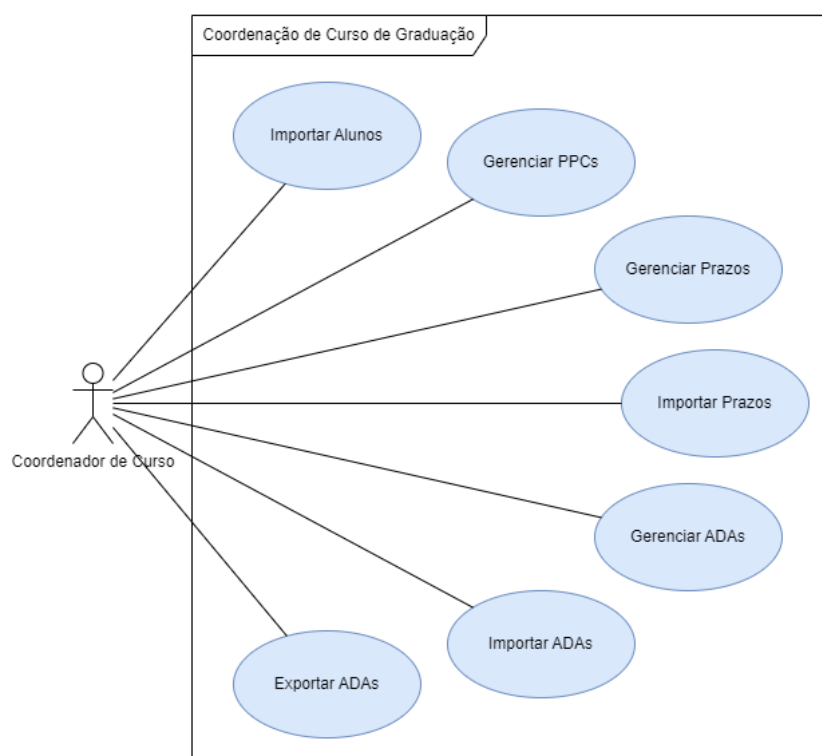


Figura 4 – Diagrama de Casos de Uso do subsistema Coordenação de Cursos de Graduação.

O caso de uso **Importar Alunos** permite que o Coordenador de Curso realize a importação em massa de alunos para um curso de graduação sob sua coordenação, utilizando um arquivo CSV. Este processo irá registrar os alunos como acadêmicos no sistema e associá-los a um curso específico.

O caso de uso **Gerenciar PPCs** é utilizado pelo Coordenador de Curso para realizar operações de CRUD (*create, read, update e delete*) em Projetos Pedagógicos de Cursos de graduação sob sua coordenação.

Os casos de uso **Importar Prazos** e **Gerenciar Prazos** estão relacionados. O primeiro é utilizado para realizar a importação em massa de vários prazos a partir de um arquivo CSV. Já o segundo permite ao coordenador realizar operações de CRUD para gerenciar todos os seus prazos.

Por fim, os casos de uso **Importar ADAs**, **Gerenciar ADAs** e **Exportar ADAs**

também estão interconectados. No primeiro caso, o Coordenador de Curso realiza a importação em lote de Acompanhamentos de Desempenho Acadêmico para um curso sob sua coordenação. Ele também precisa especificar o ano-semester em que o ADA será computado. O segundo caso de uso permite a realização de operações de leitura, edição e exclusão em ADAs que foram importados, fornecendo total autonomia ao Coordenador para gerenciar os dados da forma que considerar mais conveniente. Por fim, o terceiro caso de uso permite que, através de uma visualização, o Coordenador exporte os ADAs que está vendo para um arquivo CSV.

Na Tabela 4, é possível observar a rastreabilidade dos casos de uso em relação às histórias de usuário mencionadas na Seção 3.2.

Tabela 4 – Rastreabilidade dos casos de uso do subsistema Coordenação de Curso de Graduação.

<b>Id</b>	<b>Nome</b>	<b>Requisitos</b>
UC-1	Importar Alunos	US-5
UC-2	Gerenciar PPCs	US-4
UC-3	Gerenciar Prazos	US-1, US-3
UC-4	Importar Prazos	US-2
UC-5	Gerenciar ADAs	US-6, US-7, US-9
UC-6	Importar ADAs	US-6
UC-7	Exportar ADAs	US-8

### 3.4.2 Subsistema de Submissão de Atividades Acadêmicas

A Figura 5 ilustra os casos de uso para o subsistema de Submissão de Atividades Acadêmicas.

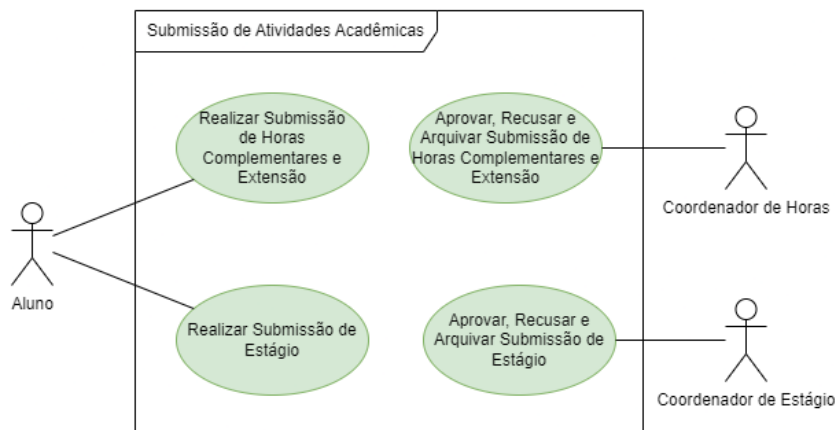


Figura 5 – Diagrama de Casos de Uso do subsistema de Submissão de Atividades Acadêmicas.

Os casos de uso **Submissão de Horas Complementares e Extensão** e **Submissão de Estágio** envolvem um Aluno de curso de graduação como ator. O primeiro caso de uso permite que o Aluno submeta horas complementares e de extensão, bem como acompanhe o progresso dessas atividades. O segundo caso é semelhante, exceto que neste caso o Aluno submete um relatório de estágio para aprovação.

O caso de uso **Aprovar, Recusar e Arquivar Submissão de Horas Complementares e Extensão** permite que o Coordenador de Horas de um curso de graduação gerencie as submissões de horas realizadas pelos Alunos.

Da mesma forma, o caso de uso **Aprovar, Recusar e Arquivar Submissão de Estágio** possibilita que um Coordenador de Estágio de um curso de graduação gerencie as submissões de relatórios de estágio feitas pelos Alunos.

Na Tabela 5, é possível verificar a rastreabilidade dos casos de uso em relação às histórias de usuário descritas na Seção 3.2.

Tabela 5 – Rastreabilidade dos casos de uso do subsistema Submissão de Atividades Acadêmicas.

Id	Nome	Requisitos
UC-8	Realizar Submissão de Horas Complementares e Extensão	US-10, US-11
UC-9	Aprovar, Recusar e Arquivar Submissão de Horas Complementares e Extensão	US-12, US-13, US-14

Id	Nome	Requisitos
UC-10	Realizar Submissão de Estágio	US-15, US-16
UC-11	Aprovar, Recusar e Arquivar Submissão de Estágio	US-17, US-18, US-19

## 3.5 Diagrama de Classes

O modelo conceitual estrutural tem como objetivo capturar e descrever as informações (classes, associações e atributos) necessárias para representar as funcionalidades descritas nos casos de uso especificados na Seção 3.4. Cada um dos subsistemas presentes no módulo possui seu próprio diagrama de classes, os quais podem ser observados nas figuras 6 e 7, descritas na sequência.

### 3.5.1 Subsistema de Coordenação de Curso de Graduação

No subsistema de Coordenação de Curso de Graduação, a Figura 6 representa a estrutura de classes. Nela, as funcionalidades descritas nos casos de uso são facilmente identificadas como classes. A parte que gerencia os ADAs dos Alunos está localizada na metade superior da figura, onde se observa que um aluno, representado por sua matrícula em um curso de graduação (*MatriculaCursoGrad*) pode possuir vários ADAs (*AcompanhamentoDesempenho*), que, por sua vez, estão vinculados a um PPC, que está relacionado a um curso de graduação (*CursoGrad*).

Na metade inferior da Figura 6, encontram-se as funcionalidades relacionadas ao controle de prazos pelos Coordenadores de Curso de Graduação (*CoordenadorCursoGrad*). Aqui, é utilizado o conceito de herança para otimizar os recursos do paradigma Orientado a Objetos. Portanto, um Coordenador simplesmente realiza o cadastro de um *Prazo*, o qual pode ser do tipo *PrazoInicioFim* (define um período em que algo deve ser feito), *PrazoUnico* (define apenas uma data final), *PrazoEncadeado* (permite definir uma sequência de prazos) ou *PrazoRecorrente* (define um prazo que se renova com uma certa frequência).

### 3.5.2 Subsistema de Submissão de Atividades Acadêmicas

No subsistema de Submissão de Atividades Acadêmicas, o diagrama de classes é mais conciso, como apresentado na Figura 7. Semelhante ao gerenciamento de prazos, o recurso de herança é aplicado aqui. Portanto, um Aluno (*Estudante*) realiza uma *Submissão*, que pode ser do tipo *SubmissaoHorasComplementares*, *SubmissaoHorasExtensao*, ou *SubmissaoEstagio*, cujos significados podem ser extraídos de seus próprios nomes. Vale ressaltar que, embora na seção de diagrama de casos de uso desse subsistema, Seção 3.4.2, os atores Coordenador de Horas e Coordenador de Estágio sejam mencionados, eles não estão representados



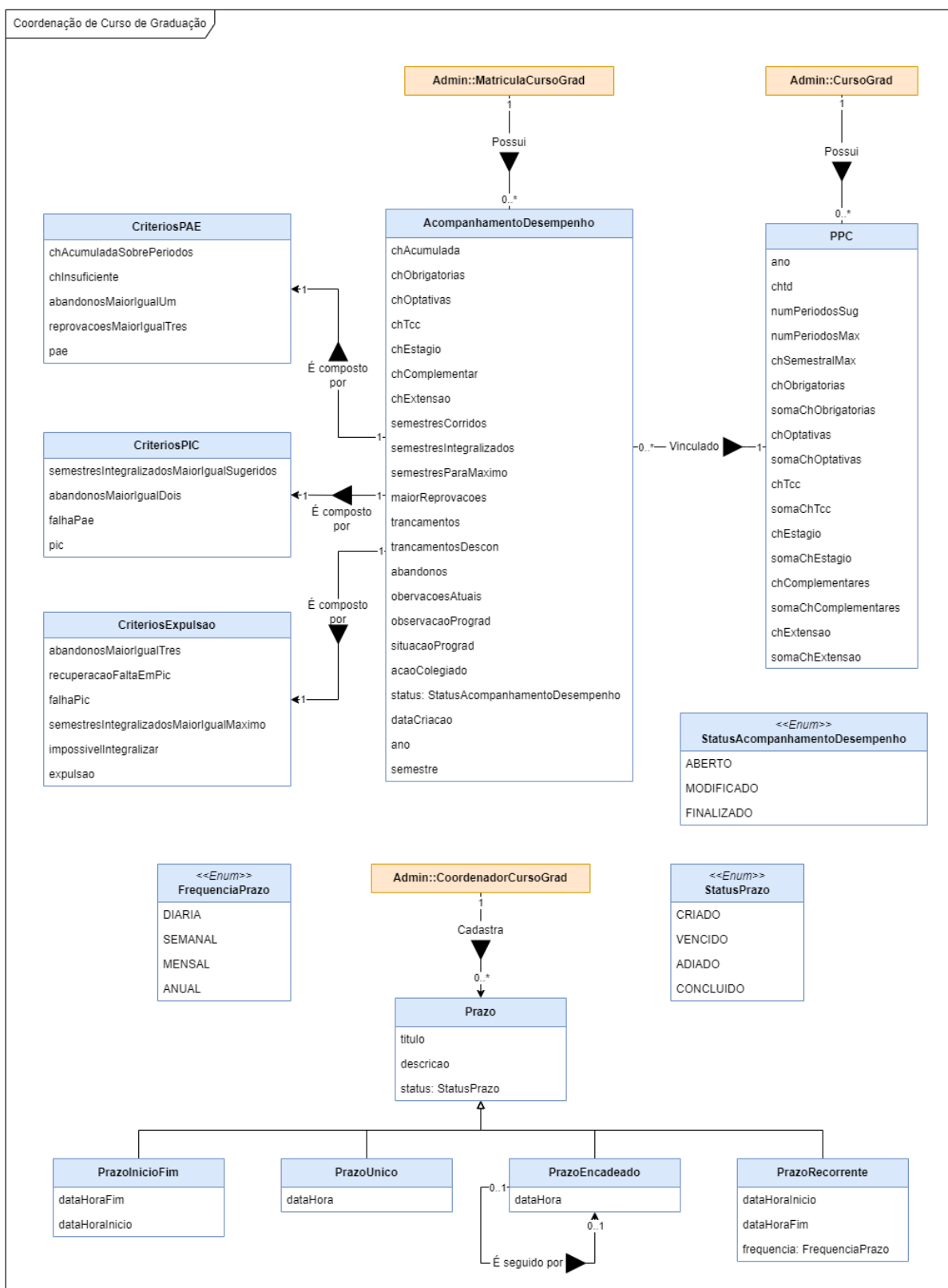


Figura 6 – Diagrama de classes do subsistema Coordenação de Curso de Graduação.

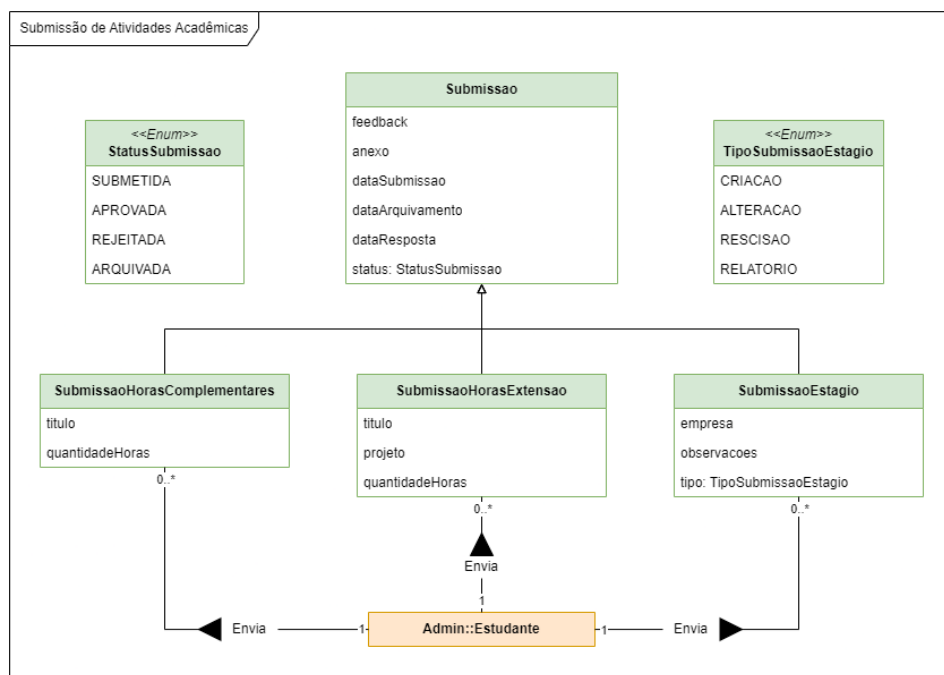


Figura 7 – Diagrama de classes do subsistema Submissão de Atividades Acadêmicas.

nesse diagrama, pois não é necessário especificar qual desses atores está acompanhando as submissões.

## 4 Projeto do Sistema

Este capítulo tem como objetivo apresentar o projeto (*design*) do sistema *Módulo de Otimização e Centralização das Atividades Acadêmicas da Graduação do Marvin* e está organizado da seguinte forma: a Seção 4.1 apresenta a plataforma de software utilizada na implementação do sistema; a Seção 4.2 apresenta a arquitetura de software; por fim, a Seção 4.3 apresenta o projeto dos componentes da arquitetura.

### 4.1 Tecnologias e Ferramentas Utilizadas

Na Tabela 6 são listadas as tecnologias utilizadas no desenvolvimento da ferramenta, bem como o propósito de sua utilização.

Tabela 6 – Plataforma de Desenvolvimento e Tecnologias Utilizadas.

Tecnologia	Versão	Descrição	Propósito
Java	19	Linguagem de programação orientada a objetos e independente de plataforma.	Escrita do código-fonte das classes que compõem o sistema.
Jakarta EE Web Profile	9.1	Conjunto de especificação de APIs e tecnologias, que são implementadas por programas servidores de aplicação.	Redução da complexidade do desenvolvimento, implantação e gerenciamento de aplicações Web a partir de seus componentes de infra-estrutura prontos para o uso.
Jakarta Enterprise Beans (EJB) Lite	4.0	API para construção de componentes transacionais gerenciados por <i>container</i> .	Implementação das regras de negócio em componentes distribuídos, transacionais, seguros e portáteis.
Jakarta Persistence (JPA)	3.0	API para persistência de dados por meio de mapeamento objeto/-relacional.	Persistência dos objetos de domínio sem necessidade de escrita dos comandos SQL.
Jakarta Contexts and Dependency Injection (CDI)	3.0	API para injeção de dependências.	Integração das diferentes camadas da arquitetura.
Jakarta Server Faces (JSF)	3.0	API para a construção de interfaces de usuários baseada em componentes para aplicações Web	Criação das páginas Web e sua comunicação com as classes Java.
Facelets	3.0	API para definição de decoradores ( <i>templates</i> ) integrada ao JSF.	Reutilização da estrutura visual comum às páginas, facilitando a manutenção do padrão visual do sistema.

Tecnologia	Versão	Descrição	Propósito
PrimeFaces	12.0	Conjunto de componentes visuais JSF <i>open source</i> .	Reutilização de componentes visuais Web de alto nível.
AdminFaces	1.6.1	<i>Template</i> visual completo integrado ao Facelets/JSF e ao PrimeFaces.	Fornecimento do padrão visual do sistema.
JButler	2.1.2	<b>JButler</b> é um <i>mini-framework</i> para facilitar o desenvolvimento de aplicações Jakarta EE.	Fornecimento de superclasses prontas para entidades (classes de domínio) persistentes, DAOs e controladores/serviços básicos de cadastro (CRUD).
MySQL Server	8.0	Sistema Gerenciador de Banco de Dados Relacional gratuito.	Armazenamento dos dados manipulados pela ferramenta.
WildFly (Preview)	26.0	Servidor de Aplicações compatível com Jakarta EE 9.	Fornecimento de implementação das APIs citadas acima e hospedagem da aplicação Web, dando acesso aos usuários via HTTP.

Na Tabela 7 vemos os softwares que apoiaram o desenvolvimento de documentos e também do código fonte.

Tabela 7 – Softwares de Apoio ao Desenvolvimento do Projeto

Tecnologia	Versão	Descrição	Propósito
draw.io	21.6.6	Aplicativo para criação de diagramas	Criação dos modelos de Entidades, Aplicação, Persistência e Navegação.
Git	2.25.1	Sistema de controle de versões	Registro de histórico de alterações no repositório.
TeXstudio	4.5.1	Editor de $\text{\LaTeX}$ .	Escrita da documentação do sistema, sendo usado o <i>template abnTeX</i> . <sup>1</sup>
IntelliJ IDEA	2023.1.5	Ambiente de desenvolvimento (IDE) com suporte ao desenvolvimento Java EE.	Implementação, implantação e testes da aplicação Web Java EE.
Apache Maven	3.8	Ferramenta de gerência/construção de projetos de software.	Obtenção e integração das dependências do projeto.
DBeaver	23.1.1	Ferramenta para gerir bases de dados.	Visualização e edição dos dados e criação de queries SQL.
GitLab	14.4	Plataforma DevOps para gestão de projetos de software.	Controle de versão do código-fonte, gestão do time de desenvolvimento, implantação contínua.

## 4.2 Arquitetura de Software

No processo de desenvolvimento do *Módulo de Otimização e Centralização das Atividades Acadêmicas de Graduação*, empregou-se uma arquitetura fundamentada no padrão arquitetônico Camada de Serviço (*Service Layer*) conforme descrito por Fowler

<sup>1</sup> <<http://www.abntex.net.br>>.

(2002). Isso foi realizado com o auxílio de *frameworks* específicos. A Figura 8 ilustra a arquitetura do sistema, identificando os componentes presentes em cada pacote, além de destacar o papel das tecnologias Jakarta EE envolvidas.

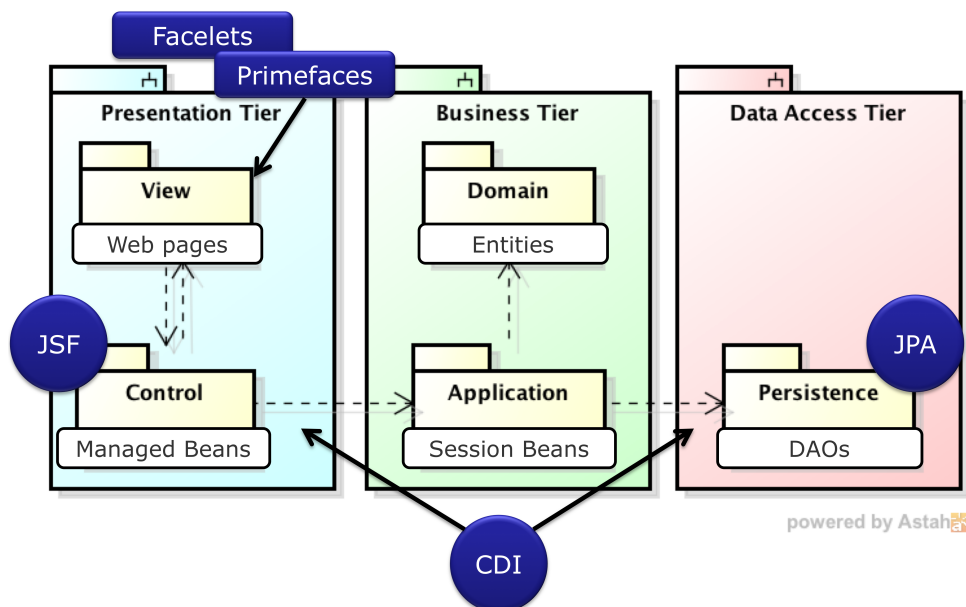


Figura 8 – Arquitetura de Software (SOUZA, 2020).

O sistema Marvin tem como objetivo centralizar uma variedade de funcionalidades em um único local. Cada conjunto de funcionalidades é organizado em módulos distintos, e cada um desses módulos adota a mesma arquitetura ilustrada na Figura 8, que se encontra subdividida em camadas.

- Camada de Apresentação (*Presentation Tier*): como o nome sugere, esta camada trata da criação da interface com a qual o usuário interage. Nela, encontramos dois componentes provenientes da arquitetura MVC: a visão (*View*), que abriga os elementos de interface, como páginas, *scripts* e *layouts*; e o controle (*Control*), que inclui as classes que gerenciam as solicitações da camada de visualização e também lida com a comunicação com o componente de aplicação (*Application*), pertencente à camada de negócios;
- Camada de Negócio (*Business Tier*): essa camada se divide em duas partes — a lógica de aplicação (*Application*) e a lógica de domínio (*Domain*). Aqui é onde as funcionalidades são oferecidas, seguindo as regras de negócios do sistema;
- Camada de Acesso a Dados (*Data Access Tier*): aqui ocorrem as operações relacionadas à persistência de informações no banco de dados. Esta camada contém apenas o componente de Persistência (*Persistence*) e é responsável por tarefas como armazenamento de arquivos e acesso a bancos de dados.

Como resultado da arquitetura proposta para o módulo, a Figura 9 ilustra a estrutura de pastas e pacotes criados em uma interface de desenvolvimento. Os nomes seguem um padrão que facilita a identificação da camada de aplicação à qual cada pacote pertence. Por exemplo, os pacotes `controller`, `application` e `persistence` correspondem, respectivamente, aos componentes pertencentes às camadas de Apresentação, Negócio e Acesso a Dados.

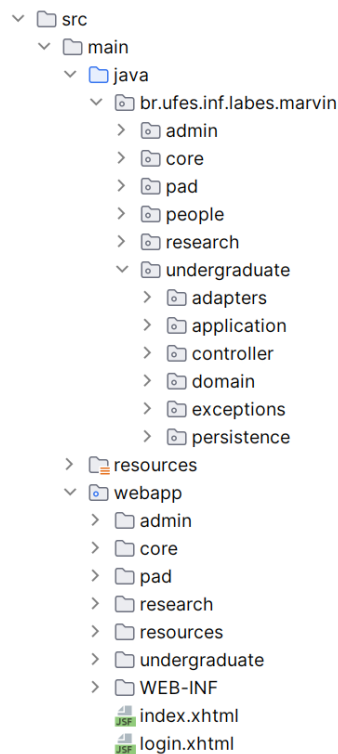


Figura 9 – Organização de pastas e pacotes do projeto.

É importante salientar que outros pacotes auxiliares podem ser criados, como é o caso dos pacotes `adapters` e `exceptions`. Esses pacotes não estão necessariamente vinculados a nenhuma das camadas mencionadas anteriormente; eles contêm classes e interfaces que auxiliam no desenvolvimento do módulo como um todo.

Por fim, vale ressaltar a existência da pasta `webapp`. Nesta pasta, ficam os arquivos `.xhtml`, que também fazem parte da Camada de Apresentação (componente de visão). Esses arquivos descrevem o que é renderizado na tela do usuário da aplicação. Eles são mantidos fora da estrutura de pacotes Java, seguindo uma padronização da ferramenta de gerência/construção de projetos de software utilizada no desenvolvimento.

### 4.3 Projeto dos Componentes da Arquitetura

Conforme mostrado na Figura 8 os principais subsistemas deste sistema estão organizados em 3 camadas: Camada de Negócio, Camada de Acesso a Dados e Camada de Apresentação, as quais são descritas em detalhes nesta seção.

### 4.3.1 Camada de Negócio

Dentro da estrutura do sistema *Otimização e centralização das atividades administrativas na graduação: desenvolvimento de um módulo de controle no sistema Marvin*, encontramos a incorporação fundamental do utilitário JButler. Especificamente, no Componente de Domínio do Problema (CDP) — *Domain*, na Figura 8, notamos que todas as classes localizadas no pacote `domain` herdam as funcionalidades e recursos da classe `PersistentObjectSupport` do JButler. Para manter a clareza nos diagramas, decidimos não representar essa herança de forma explícita, uma vez que a quantidade considerável de associações poderia tornar a visualização do modelo excessivamente complexa.

A classe `PersistentObjectSupport` desempenha um papel essencial, oferecendo atributos e métodos valiosos compartilhados por todas as classes persistentes do sistema. Alguns desses elementos foram modularizados em uma superclasse distinta chamada `DomainObjectSupport`. Ambas as classes definem interfaces que estabelecem contratos específicos, permitindo que o JButler seja reutilizado de maneira eficaz, mesmo por desenvolvedores que optem por não empregar herança.

Vale destacar que o módulo em análise mantém uma relação direta com o módulo `admin`. Essa relação pode ser observada nos relacionamentos entre as classes nos diagramas. Em resumo, as entidades essenciais abrangem inicialmente `UndergraduateDegree`, `UndergraduateDegreeEnrollment`, e `UndergraduateDegreeCoordination`, como ilustrado nas figuras 10 e 11.

Por fim, o Componente de Gerência de Tarefas (CGT) — *Application* na Figura 8, presente no pacote `application` e parte da Camada de Negócio, será analisado em conjunto com a Camada de Apresentação na Seção 4.3.3, dada a relação estreita existente o CGT e os componentes desta última camada.

#### 4.3.1.1 Subsistema Coordenação de Curso de Graduação

Ao observar o subsistema de Coordenação de Curso de Graduação, a Figura 10 apresenta o design atualizado desta camada, refletindo a implementação real. Em comparação com a Figura 10, as classes são agora representadas com nomes e atributos em inglês, seguindo o padrão do sistema em questão, e os tipos de cada atributo são os da linguagem Java. Além disso, são mostradas informações sobre a navegabilidade entre as classes, proporcionando uma visão mais completa da estrutura.

A introdução de novas classes e *enums* teve como objetivo principal aprimorar a organização do código. Essa abordagem evita que a classe `PerformanceTracking` se torne excessivamente extensa, proporcionando independência a cada critério para melhor controle. Quanto aos *enums*, sua inclusão segue boas práticas de programação, representando conjuntos fixos de valores. Isso torna o código mais claro, legível e menos propenso a erros

de implementação.

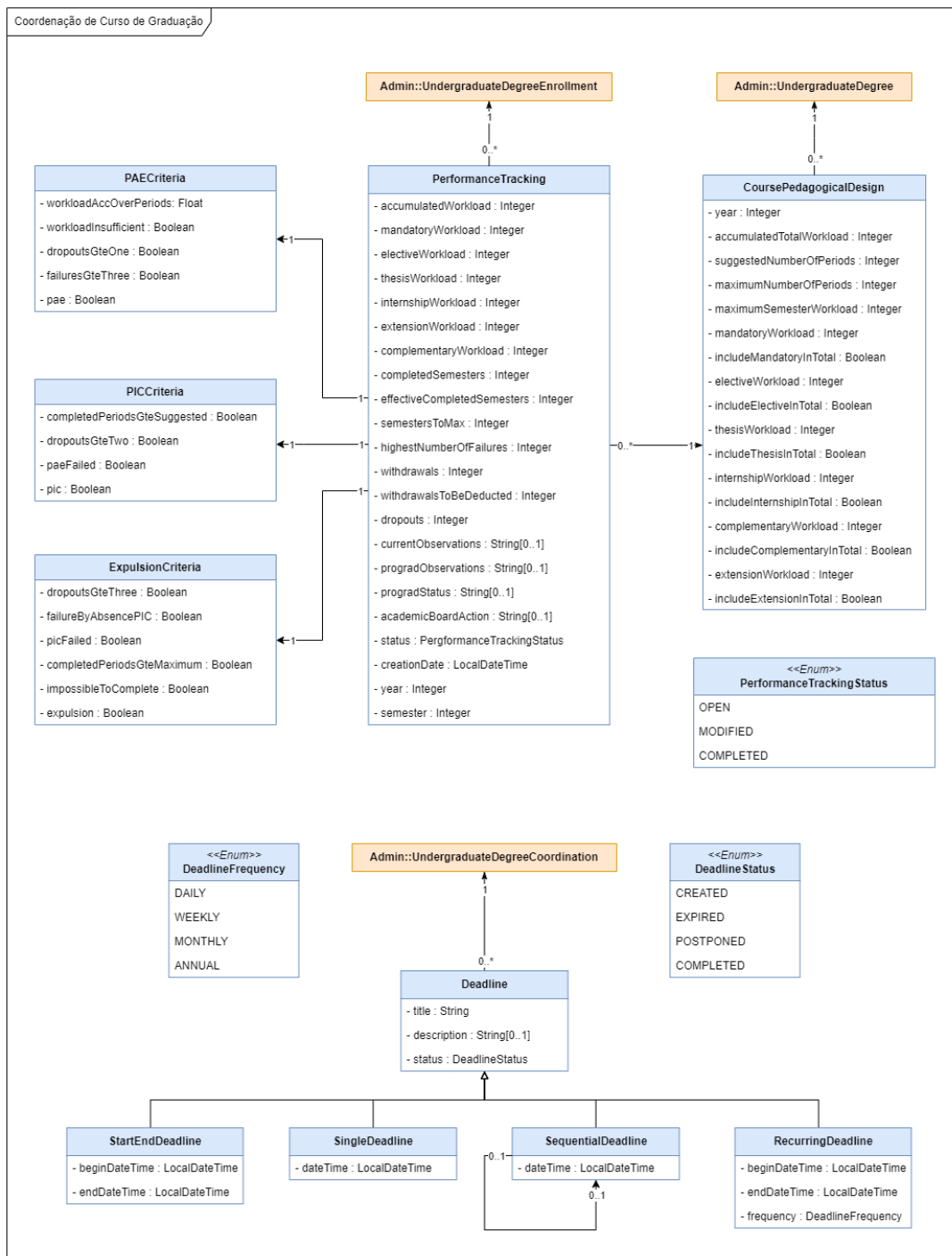


Figura 10 – Projeto da Componente de Domínio do Problema (domain) do subsistema Coordenação de Curso de Graduação.

#### 4.3.1.2 Subsistema Submissão de Atividades Acadêmicas

Quanto ao subsistema de Submissão de Atividades Acadêmicas, a Figura 11 ilustra o design atual desta camada. Mais uma vez, trata-se de uma evolução da Figura 7, seguindo uma abordagem semelhante àquela descrita na Seção 4.3.1.1.



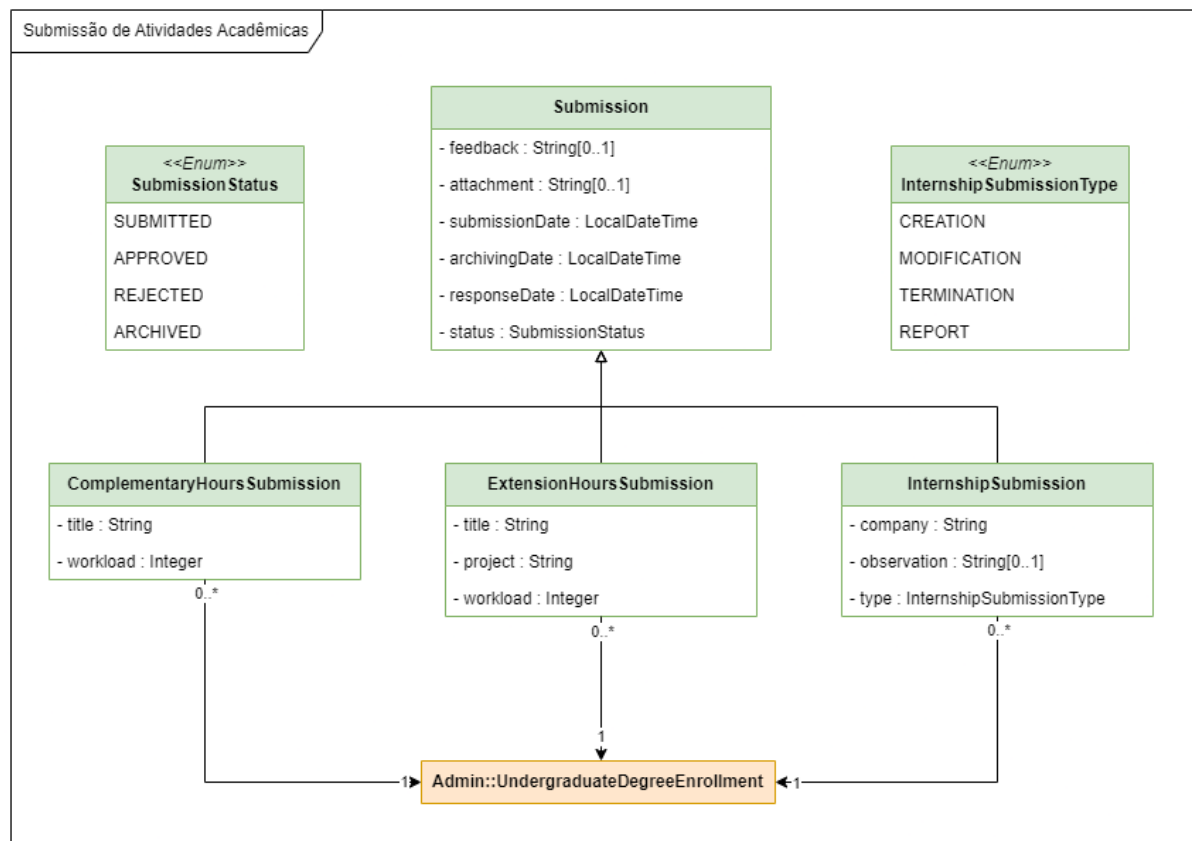


Figura 11 – Projeto da Componente de Domínio do Problema (domain) do subsistema Submissão de Atividades Acadêmicas.

### 4.3.2 Camada de Acesso a Dados

Na Camada de Acesso a Dados, estamos nos referindo ao Componente de Gerência de Dados (CGD) — *Persistence* na Figura 8 — e ao pacote `persistence` do sistema. O desenvolvimento dessa camada faz amplo uso do utilitário JButler, que oferece diversas ferramentas para facilitar o gerenciamento das entidades persistidas no banco de dados. Conforme detalhado na Seção 4.2, adotamos o padrão *Data Access Object* (DAO). Esse padrão envolve a criação de uma interface com o sufixo “DAO” e uma classe que a implementará, com o sufixo “JPADAO”.

A Figura 12 apresenta a interface e classe base do JButler, que servirão de base para todas as outras interfaces e classes neste módulo. Como é possível observar, essa abordagem oferece uma vantagem significativa para os desenvolvedores, pois ela já implementa as funcionalidades essenciais relacionadas à camada de persistência. Dessa forma, eles podem se concentrar nos detalhes específicos do domínio.

#### 4.3.2.1 Subsistema Coordenação de Curso de Graduação

No subsistema de Coordenação de Curso de Graduação, o projeto da CGD é representado na Figura 13. Um ponto importante a destacar é que a interface `DeadlineDAO`

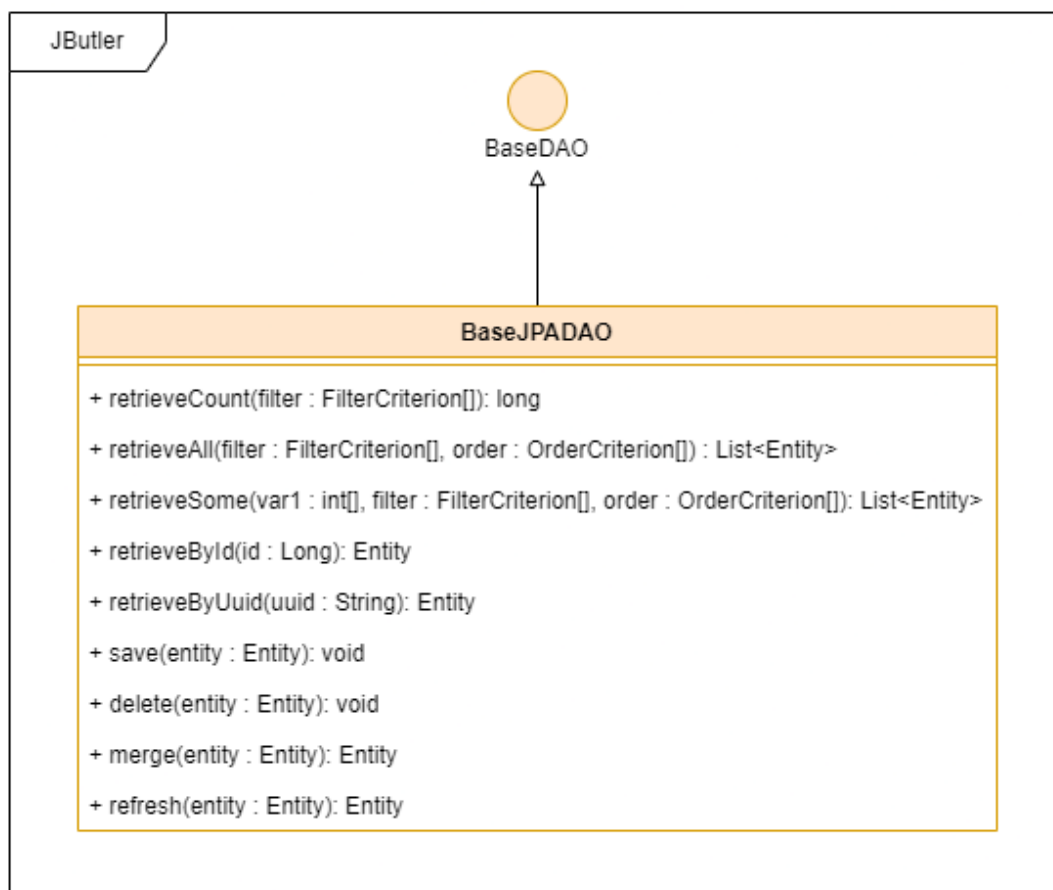


Figura 12 – Modelo de persistência base do utilitário JButler.

cuida de todas as operações relacionadas a prazos. Não foi necessário criar uma interface para cada uma das especializações de prazos que o subsistema poderia ter.

Aqui, torna-se evidente como o utilitário JButler contribui para acelerar o desenvolvimento, conforme discutido anteriormente. Entretanto, alguns métodos específicos foram necessários para atender às demandas do sistema, a saber:

- Para a classe `PerformanceTrackingJPADAO`:
  - `retrieveByYearAndSemesterAndEnrollment`: retorna um objeto `PerformanceTracking` (um ADA) filtrando por ano, semestre e matrícula;
  - `retrieveCurrent`: retorna o ADA atual de uma matrícula;
  - `findByDegreesIn`: busca todos os ADAs vinculados a cursos de graduação específicos.
- Para a classe `CoursePedagogicalDesignJPADAO`:
  - `retrieveByYearAndDegree`: retorna um objeto `CoursePedagogicalDesign` (um PPC) filtrando por ano e curso de graduação;

- `findByDegreesIn`: busca todos os PPCs vinculados a cursos de graduação específicos.
- Para a classe `DeadlineJPADAO`:
  - `retrieveByCoordination`: busca todos os prazos de um coordenador;
  - `retrieveByCoordinationAndStatus`: busca todos os prazos de um coordenador com um status específico;
  - `retrieveByCoordinationAndDate`: busca todos os prazos de um coordenador em uma data específica;
  - `retrieveByCoordinationAndMonth`: busca todos os prazos de um coordenador em um mês específico;
  - `retrieveStartEndInProgress`: busca todos os prazos do tipo de prazo de início e fim que estão em progresso em uma data específica.

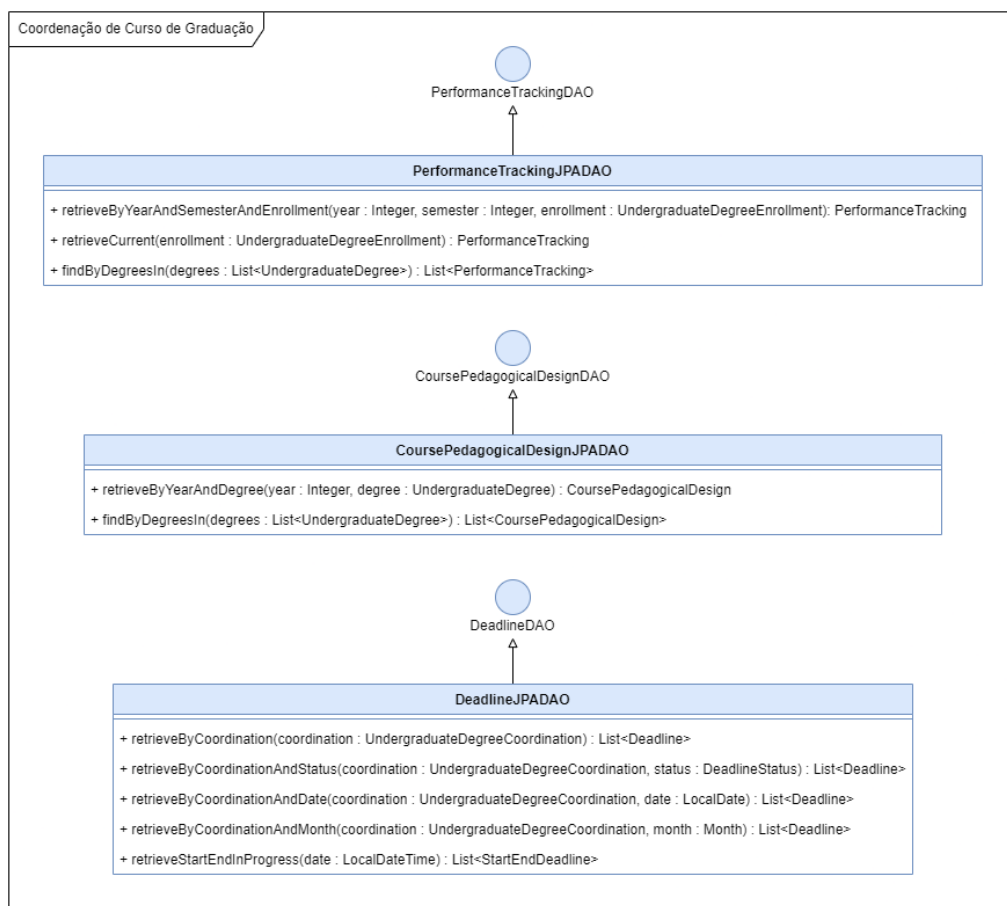


Figura 13 – Projeto da Componente de Gerência de Dados (*persistence*) do subsistema Coordenação de Curso de Graduação.

### 4.3.2.2 Subsistema Submissão de Atividades Acadêmicas

No subsistema de Submissão de Atividades Acadêmicas, a Figura 14 apresenta o projeto da CGD. Aqui, não há nenhuma peculiaridade que já não tenha sido mencionada anteriormente.

Mais uma vez, a utilização do utilitário JButler proporcionou facilidades no desenvolvimento, exigindo a implementação apenas das seguintes operações:

- Para a classe `SubmissionJPADAO`:
  - `retrieveByStatus`: busca todas as submissões com um status específico;
  - `retrieveByStudent`: busca todas as submissões de um estudante;
  - `retrieveByStudentAndStatus`: busca todas as submissões de um estudante em um status específico;
  - `retrieveInternshipByType`: busca todas as submissões do tipo de submissão de estágio com um tipo específico.

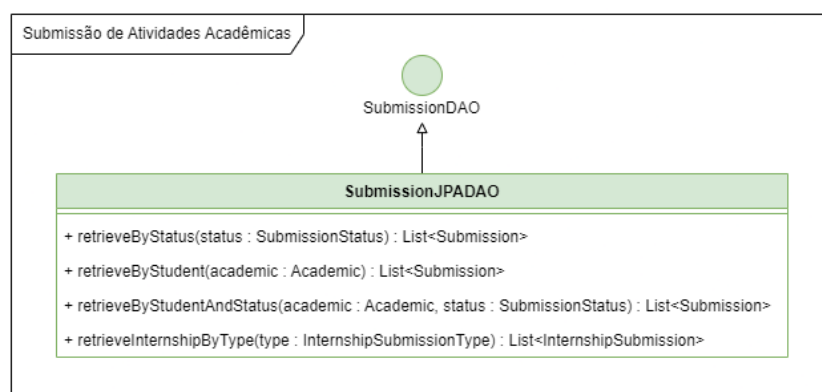


Figura 14 – Projeto da Componente de Gerência de Dados (*persistence*) do subsistema Submissão de Atividades Acadêmicas.

### 4.3.3 Camada de Apresentação

As figuras 15, 16, e 17 retratam o projeto do Componente de Interação Humana (CIH) — *View* na Figura 8, composto pela pasta `webapp`, e do Componente de Controle de Interação (CCI) — *Control* na Figura 8, representado pelo pacote `controller` do módulo. Por conta de sua dependência com o Componente de Gerência de Tarefas (CGT), presente no pacote `application`, essa relação é também destacada nas imagens. Importante notar que o projeto do subsistema Coordenação de Curso de Graduação foi dividido em duas partes para melhor visualização.

Os modelos apresentados seguem padrões específicos de representação. Os elementos que possuem o sufixo “*Controller*” representam *beans* do JSF, enquanto aqueles que

terminam em “.xhtml” representam páginas da Web renderizadas no navegador do cliente. Nesse contexto de arquitetura MVC, os controladores são a representação do *Controller* e residem no pacote `controller`, enquanto os arquivos Web compõem a *View* e estão localizados na pasta `webapp`. Juntos, formam o *front-end* da aplicação, permitindo que os usuários interajam de maneira eficaz com o sistema.

Por outro lado, as classes com o estereótipo “*interface*” são interfaces Java que representam a porta de entrada do componente *Model* no padrão MVC e são parte integrante do pacote `application`. Nesse mesmo pacote, também existem implementações para essas interfaces, embora não estejam explícitas nos diagramas. Essas interfaces trabalham em conjunto com a Componente de Domínio do Problema (CDP), que se encontra no pacote `domain`, e a Componente de Gerência de Dados (CGD), presente no pacote `persistence`, para formar o *back-end* da aplicação.

Uma característica essencial é a independência entre o *front-end* e o *back-end*. Os controladores estabelecem associações unidirecionais com as interfaces de serviço, as quais, conforme a arquitetura, não têm conhecimento prévio dos controladores. Isso mantém o *back-end* desacoplado do *front-end*. As relações bidirecionais entre controladores e páginas da Web representam o *binding* realizado pelo JSF entre as *tags* nas páginas XHTML e os atributos das classes controladoras, permitindo que os dados fluam em ambas as direções. Por fim, as associações unidirecionais entre as páginas da Web representam links entre elas, os quais, de maneira semelhante ao redirecionamento, iniciam uma nova solicitação pela página de destino.

Não são especificadas cardinalidades nas associações entre controladores e serviços, exceto na extremidade navegável, que reside no lado do serviço. Outras associações não incluem cardinalidades, visto que não envolvem classes, mas páginas da Web, tornando redundante a especificação de cardinalidades nessas relações.

Finalmente, é importante notar que em todos os modelos existe uma página da web chamada `index.xhtml`. Essa página serve como ponto de entrada no sistema para o usuário, direcionando-o para diversos módulos e outras páginas disponíveis. A partir de `index.xhtml`, o usuário começa a explorar o sistema em busca de suas necessidades.

#### 4.3.3.1 Subsistema Coordenação de Curso de Graduação

A Figura 15 representa a interface da parte do subsistema destinada ao gerenciamento de PPCs, ADAs, além da funcionalidade de importar alunos e ADAs. Nas telas de gerenciamento, identificadas pelo prefixo “*manage*”, os fluxos são notavelmente simplificados, permitindo que cada operação seja executada de maneira conveniente em um único arquivo XHTML.

Com base na página `manageCoursePedagogicalDesigns/index.xhtml`, sua estru-

tura foi concebida para operar como uma aplicação de página única (SPA, do inglês *Single Page Application*). Isso implica que os componentes da página são dinamicamente alterados à medida que o usuário interage, sem a necessidade de redirecionamento. A tecnologia AJAX (*Asynchronous JavaScript and XML*) é empregada para permitir operações assíncronas, processando requisições ao servidor em segundo plano e exibindo os dados posteriormente ao usuário.

Ao acessar a página `manageCoursePedagogicalDesigns/index.xhtml`, o usuário inicialmente visualiza uma tabela contendo todos os PPCs dos cursos que coordena. Por meio do *bind* estabelecido com o `ManageCoursePedagogicalDesignsController`, essa tabela é preenchida, pois a controladora possui um atributo que é a lista desses PPCs, populado pelo método *list* do serviço `ManageCoursePedagogicalDesignsService`. Com os PPCs listados, o coordenador tem a opção de editar ou excluir um dos registros. Ao selecionar editar, a tela é alterada dinamicamente por meio de AJAX, substituindo a tabela por um formulário preenchido com as informações do PPC. Além disso, há a opção de criar um novo registro, onde, de maneira semelhante à edição, a tabela é substituída por um formulário vazio para inserção de novos dados.

É relevante destacar que o fluxo de todas essas operações segue um padrão, onde a controladora gerencia o que deve ser exibido na tela, utilizando a camada de serviço para efetivar as operações. Por exemplo, ao excluir um registro, o usuário clica em um botão na tela, a controladora identifica o clique por meio do *bind*, e então realiza uma chamada à camada de serviço para persistir a ação no banco de dados.

A presença do utilitário `JButler` novamente simplifica o desenvolvimento. A classe `CrudService` oferece funcionalidades básicas de CRUD, como *create*, *retrieve*, *update* e *delete*. Assim, o serviço `ManageCoursePedagogicalDesignsService`, embora possua apenas o método *list*, é capaz de realizar todas as operações necessárias para o CRUD dos PPCs.

As telas de importação, identificadas pelo prefixo “*import*”, representam fluxos mais complexos, nos quais várias telas guiadas são usadas para concluir uma ação. Vale a pena observar que nesses casos, diversos arquivos XHTML usam o mesmo controlador, mantendo o mesmo escopo para o usuário ao navegar por diferentes páginas.

Analisando o fluxo de importação de alunos, notamos uma abordagem distinta em relação às telas de “*manage*”. Nesse caso, o fluxo foi concebido com o uso de redirecionamento entre as páginas `xhtml`, em vez de adotar uma abordagem de SPA (*Single Page Application*). Aqui, uma única controladora é utilizada para várias telas, sendo possível graças à anotação `@ConversationScoped` do JSF, que permite manter o estado de uma conversa do usuário ao longo de várias solicitações HTTP, garantindo que o estado persista mesmo durante a navegação entre páginas.

Iniciando na tela `importStudents/index.xhtml`, o coordenador pode selecionar um curso para o qual deseja importar estudantes e realizar o *upload* de um arquivo CSV contendo as informações dos alunos a serem importados. Após a confirmação dessas informações, a controladora `ImportStudentsController` envia o arquivo para o serviço `ImportStudentsService` por meio do método *uploadStudents*. Este método lê todas as informações dos alunos no arquivo CSV e retorna os resultados para a controladora, que, por sua vez, redireciona o usuário para a página `importStudents/list.xhtml`, exibindo uma tabela com todos os alunos identificados no arquivo CSV.

Com o coordenador visualizando os dados a serem importados, ele tem a capacidade de editar as informações de cada aluno ou remover um aluno indesejado. Todas essas modificações ocorrem exclusivamente entre a tela e a controladora, sem a necessidade de envolver a camada de serviço, uma vez que nenhum dado foi persistido até esse momento. Ao confirmar os dados, a controladora valida a consistência com o método *validateEnrollStudents* do serviço. Se tudo estiver correto, a ação é efetivada pelo método *enrollStudents* do serviço. A controladora, então, obtém uma lista com todas as novas matrículas geradas pelo sistema e encaminha o coordenador para a tela `importStudents/done.xhtml`, onde é possível visualizar todos os alunos que foram importados com sucesso.

A representação de prazos, que também faz parte deste subsistema, é controlada por meio do modelo apresentado na Figura 16. O fluxo é semelhante ao já mostrado, e é importante mencionar os componentes `CalendarDeadlinesController` e `deadlines/index.xhtml`, que são responsáveis por uma tela mais voltada para a visualização dos prazos do coordenador de curso.

Além disso, vale destacar que, embora o modelo da Camada de Negócios (Figura 10) inclua três especializações para a entidade `Deadline`, no contexto da interface, todas são tratadas como uma classe generalizada `Deadline`. Essa abordagem é adotada devido à semelhança significativa em termos de atributos e funcionalidades entre todas as especializações. Dessa forma, é mais eficiente ter uma única tela para lidar com todas elas, mantendo apenas algumas especificidades para atender a cada caso.

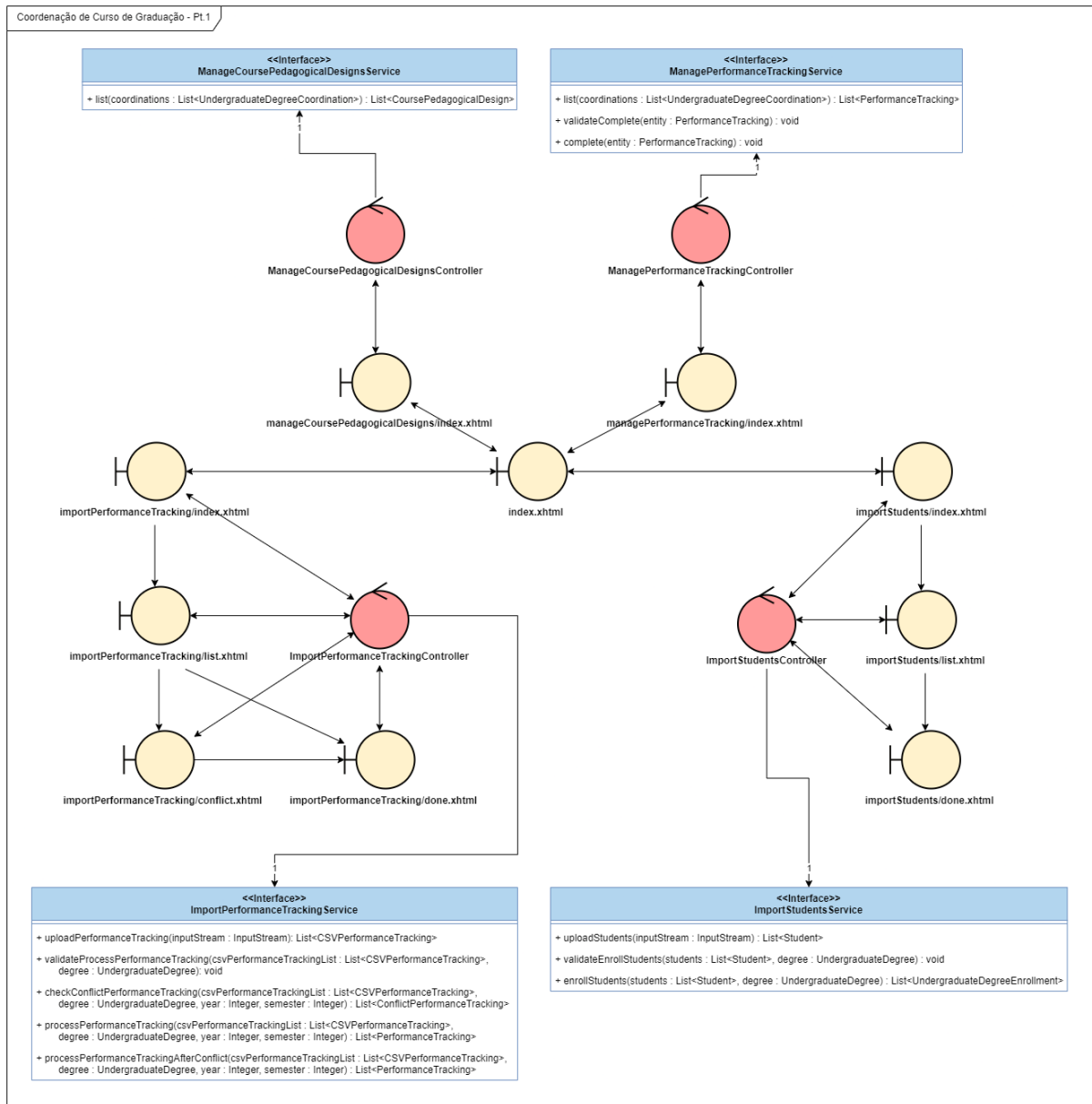


Figura 15 – Projeto da Camada de Interface com o Usuário (view e controller) do sub-sistema Coordenação de Curso de Graduação, juntamente com a Componente de Gerência de Tarefas (application). - Parte 1.



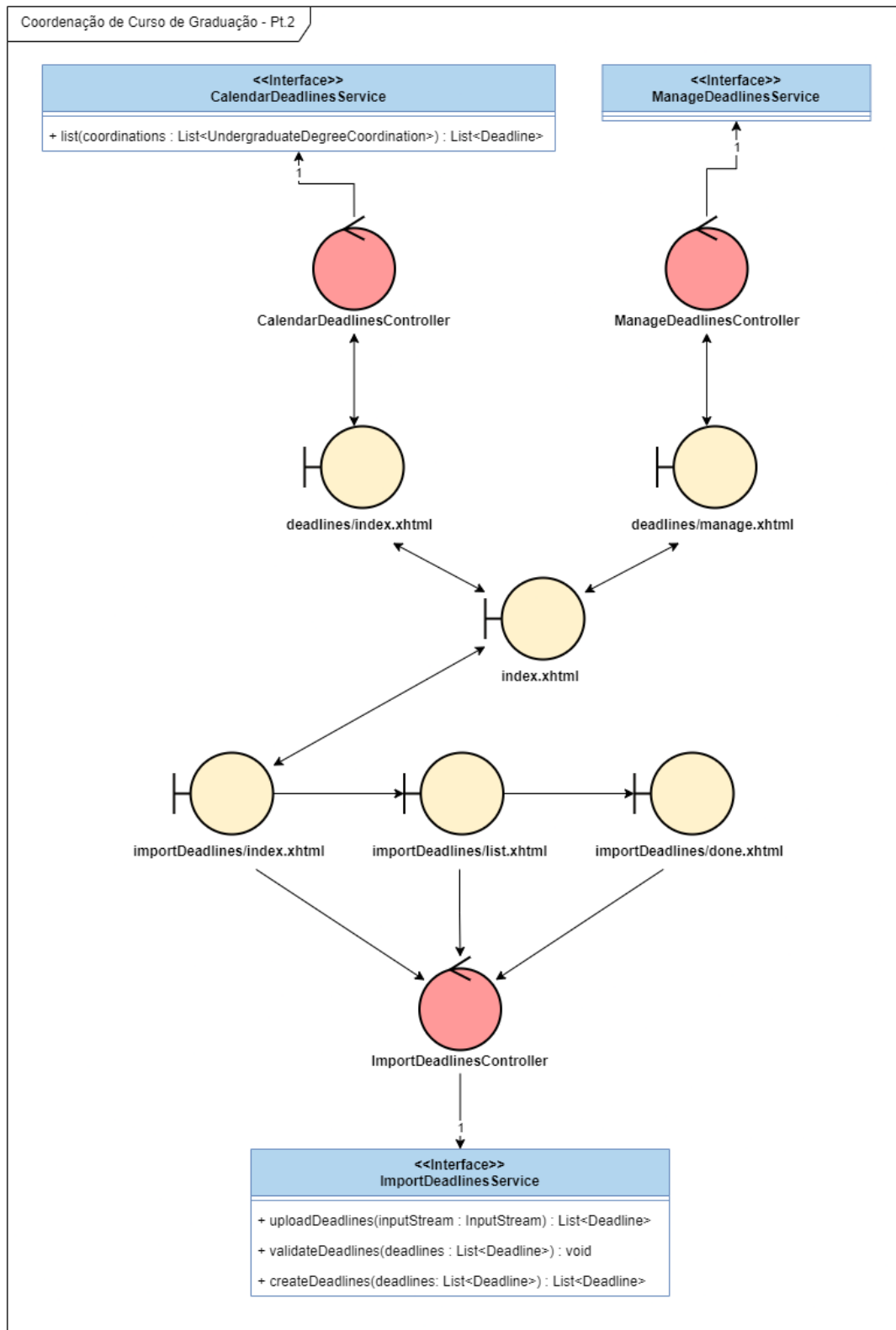


Figura 16 – Projeto da Camada de Interface com o Usuário (view e controller) do subsistema Coordenação de Curso de Graduação, juntamente com a Camada de Gerência de Tarefas (application). - Parte 2.

#### 4.3.3.2 Subsistema Submissão de Atividades Acadêmicas

Concluindo a Camada de Apresentação, a Figura 17 descreve o modelo do subsistema de Submissão de Atividades Acadêmicas. A principal distinção em relação aos modelos anteriores (Seção 4.3.3.1) é a presença de uma tela intermediária que organiza outras antes do usuário acessar as páginas nas quais ele pode realizar operações. Essa tela é `submissions/index.xhtml`, adotada por questões de organização para melhorar a experiência do usuário no sistema.

Aqui, nos deparamos novamente com várias telas prefixadas por “*manage*”, e, seguindo o padrão do sistema, essas telas operam da mesma maneira descrita na Seção 4.3.3.1. Em outras palavras, elas funcionam como uma SPA, utilizando a tecnologia AJAX para proporcionar uma interatividade mais eficaz com o usuário.

Considerando a página `manageComplementaryHoursSubmissions/index.xhtml`, pois todas elas compartilham o mesmo funcionamento, variando apenas em alguns campos a serem preenchidos e no perfil de usuário que pode acessá-las. Quando um estudante acessa a tela, todas as suas submissões de horas complementares são exibidas. Ele pode apenas visualizar as informações da submissão para acompanhar o progresso ou cadastrar uma nova submissão. Se um coordenador de horas complementares acessa a página, ele vê todas as submissões feitas pelos alunos sob sua coordenação e pode visualizar as informações de cada uma, além de fornecer *feedbacks* positivos ou negativos.

Todas as operações descritas anteriormente ocorrem por meio da interação entre a controladora `ManageComplementaryHoursSubmissionsController` e ao componente da camada de serviço `ManageComplementaryHoursSubmissionsService`. Partindo da listagem, quando a primeira chama o método `list` do segundo, todos os dados relevantes para cada um dos perfis de usuário são retornados. A visualização de uma submissão utiliza o método `retrieve` do serviço, que também herda da classe `CrudService` do utilitário `JButler`, proporcionando diversas outras funcionalidades, como `create`, que também é invocado quando o aluno cadastra uma nova submissão. Por fim, o coordenador pode enviar feedback ao preencher um campo na tela e confirmar a ação, permitindo que a controladora execute o método `sendFeedback` do serviço. Assim, apesar de nem todas as operações de CRUD serem contempladas, uma parte significativa delas é, conforme levantado nos requisitos do sistema.

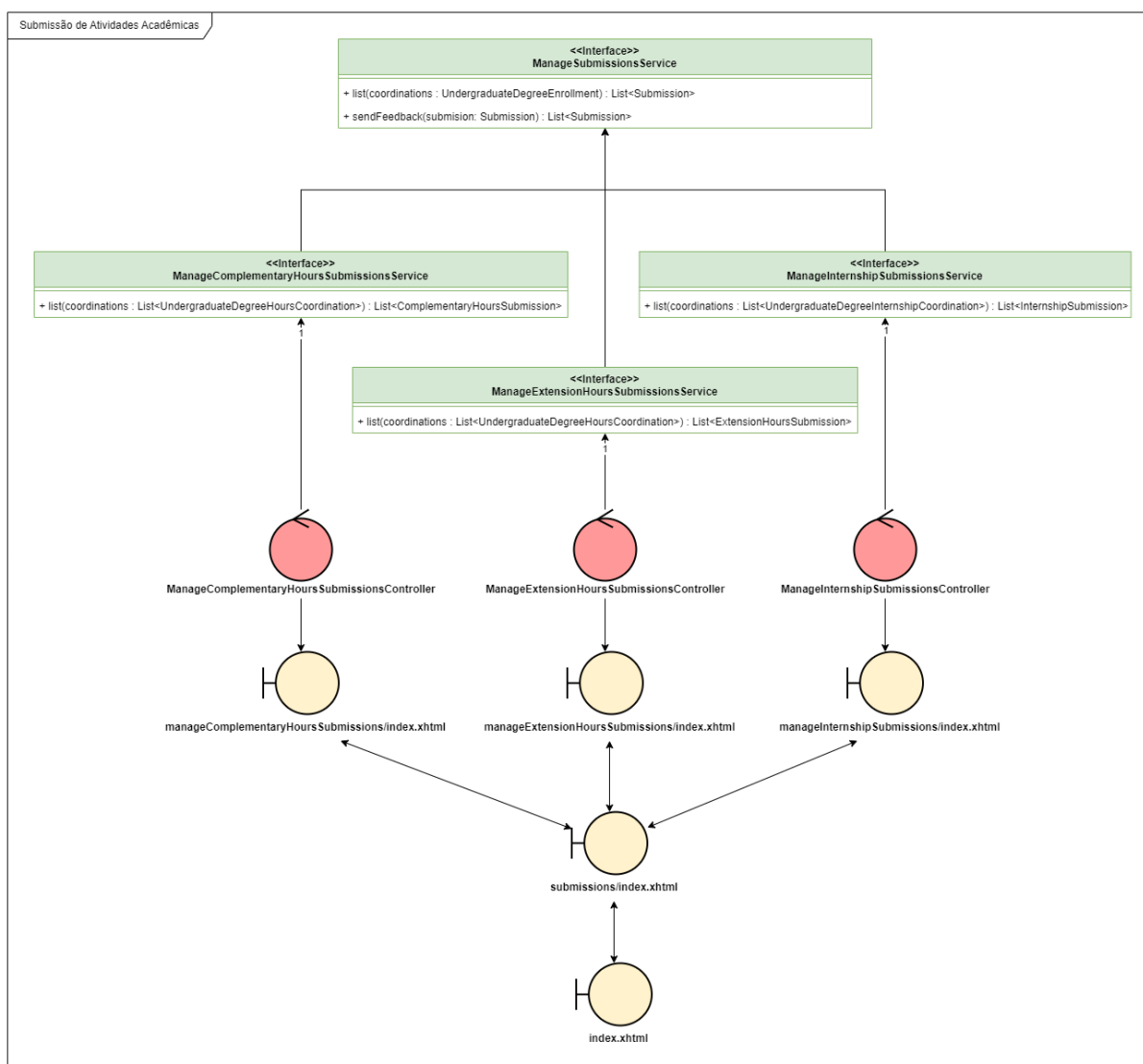


Figura 17 – Projeto da Camada de Interface com o Usuário (view e controller) do subsistema Submissão de Atividades Acadêmicas, juntamente com a Camada de Gerência de Tarefas (application).

## 5 Implementação do Sistema

Neste capítulo, apresentaremos os resultados do sistema desenvolvido, utilizando várias capturas de tela. Dado o amplo escopo do *Módulo de Otimização e Centralização das Atividades Acadêmicas de Graduação*, conforme discutido na Seção 3.1, a implementação completa do módulo exigiria mais tempo do que o disponível para este trabalho. Portanto, limitamos a implementação aos casos de uso de Importar Alunos, Gerenciar PPCs, Gerenciar ADAs, Importar ADAs e Exportar ADAs, conforme descrito na Seção 3.4.1. A versão completa da implementação pode ser acessada no repositório GitLab do Marvin, disponível em <https://gitlab.labes.inf.ufes.br/marvin/marvin/>.

Em relação ao funcionamento do sistema implementado, ao acessar a página do Marvin, o usuário é recebido com uma tela de *Login*, como ilustrado na Figura 18. Ressalta-se que o sistema de criação e autenticação de usuários já está disponível no Marvin, e o módulo utiliza esses sistemas existentes.

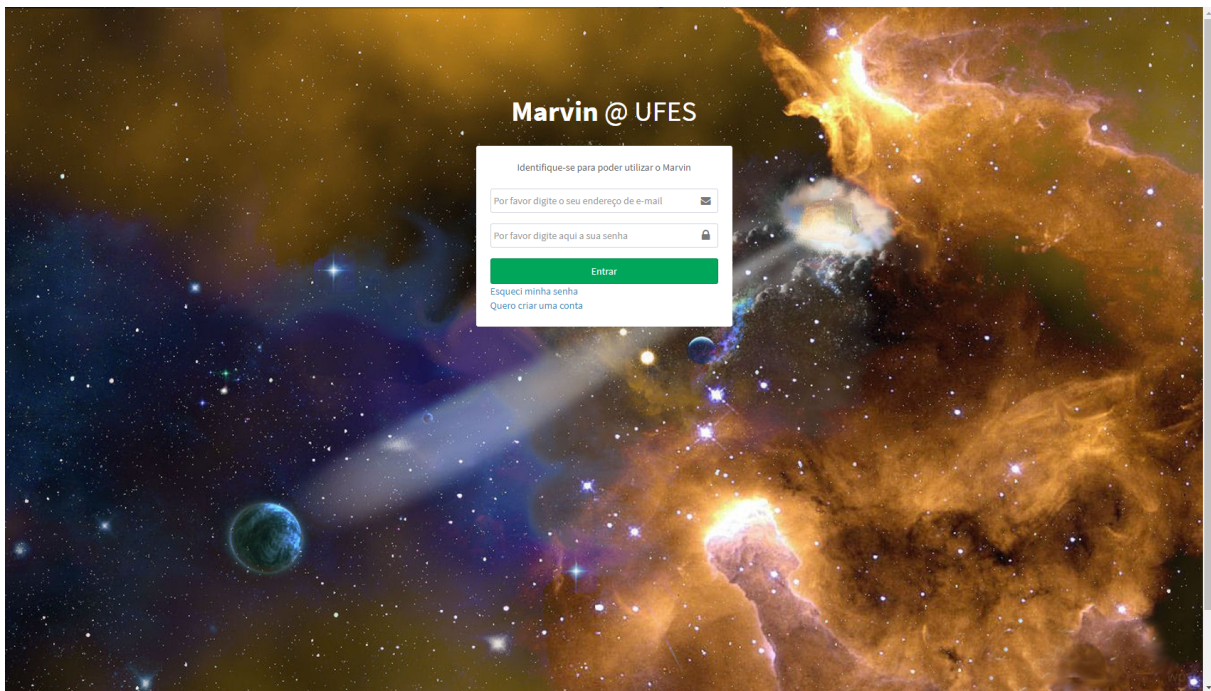


Figura 18 – Tela de *Login*.

Após a autenticação, o usuário, que ocupa o cargo de Coordenador de Curso de Graduação, é redirecionado para a tela principal do Marvin, conforme ilustrado na Figura 19. A partir daqui, o usuário tem acesso ao menu lateral com várias opções de acordo com seu perfil.

Todas as funcionalidades dos casos de uso implementados são agrupadas no menu “Graduação”. Ao clicar neste menu, ele se expande e exibe todas as funcionalidades disponí-

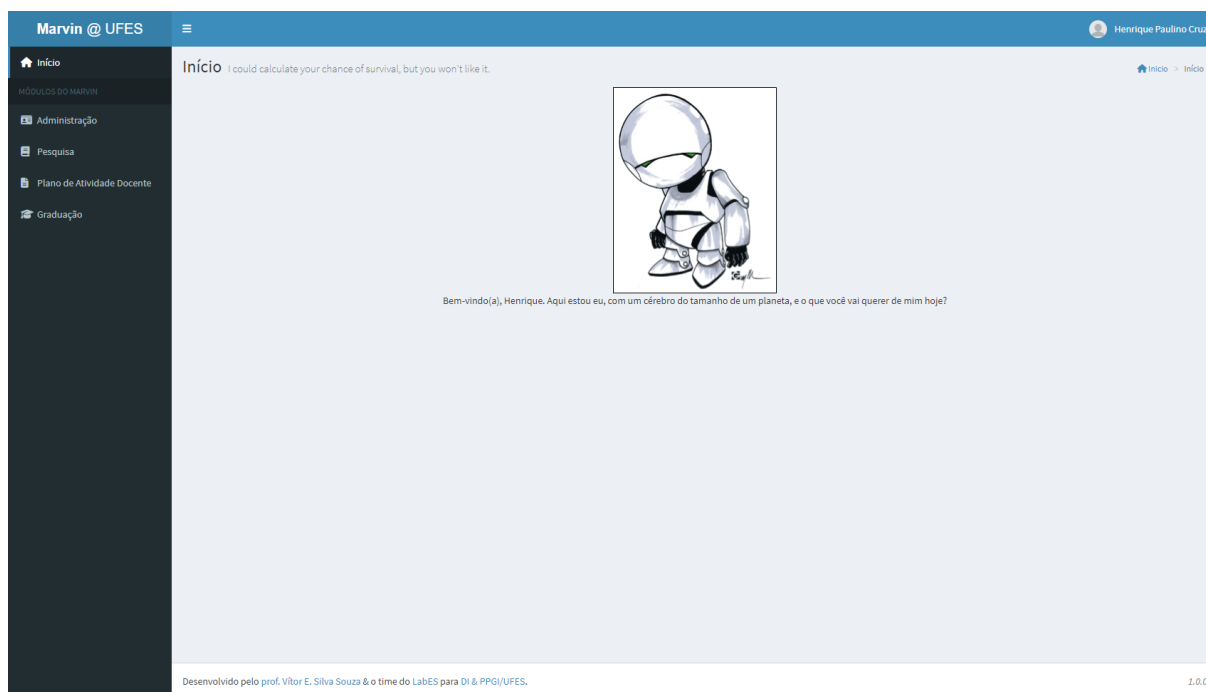


Figura 19 – Tela de Início.

veis para o Coordenador, como mostrado na Figura 20. Cada uma dessas funcionalidades será detalhadamente apresentada a seguir.

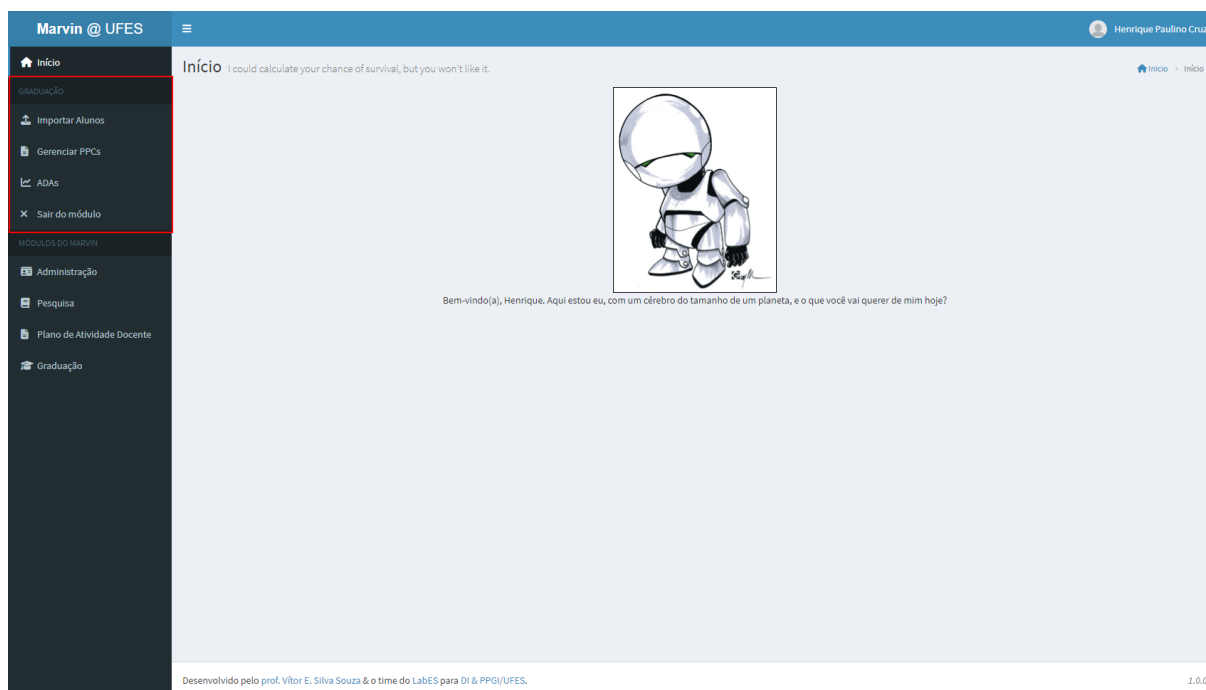


Figura 20 – Tela de Início - Menu “Graduação” expandido.

## 5.1 Importar Alunos

O primeiro caso de uso implementado foi o de Importar Alunos. Para concluir essa operação, o Coordenador passa por três telas distintas. Na Figura 21, ilustra-se a primeira etapa, onde o Coordenador pode escolher o curso de graduação e um arquivo CSV contendo os dados dos alunos a serem importados e matriculados no curso desejado. Com o formulário preenchido, o Coordenador pode prosseguir para a próxima etapa.

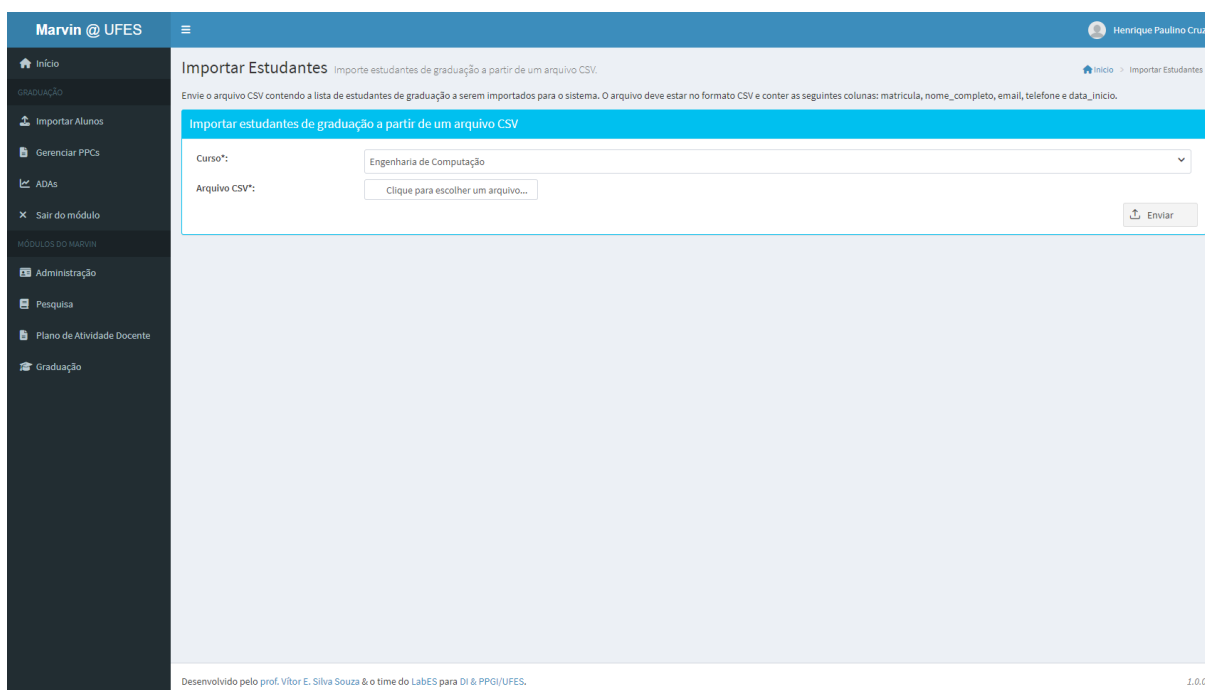


Figura 21 – Tela de Importar Alunos - Submissão de arquivo.

A Figura 22 representa a tela em que o Coordenador visualiza todos os dados dos alunos encontrados no arquivo CSV submetido. A célula destacada na Figura 22 mostra que é possível clicar em cada célula da tabela para corrigir ou alterar informações lidas do arquivo. O Coordenador pode remover uma linha da tabela, impedindo a importação do aluno. Além disso, o sistema valida os dados, alertando sobre erros, como dados ausentes ou matrículas duplicadas. Com todos os dados validados e preenchidos, o Coordenador pode avançar para a última tela.

O caso de uso conclui com a Figura 23, mostrando ao Coordenador os dados efetivamente importados. Nessa tela, não é possível editar os alunos, apenas visualizá-los.

**Importar Estudantes** importe estudantes de graduação a partir de um arquivo CSV.

Por favor, revise as informações extraídas do arquivo CSV enviado e confirme a importação dos estudantes.

**Estudantes presentes no CSV enviado**

Matricula	Nome Completo	E-mail	Telefone	Data de Início	
2018100333	Aluno 3	aluno3@email.com	28 99999-9999	05/03/2016	
2018100444	Aluno 4	aluno4@email.com	28 99999-9999	05/03/2018	
2019100555	Aluno 5	aluno5@email.com	28 99999-9999	05/03/2019	
2021100111	Aluno 1	aluno1@email.com	28 99999-9999	03/03/2021	
2023100222	Aluno 2	aluno2@email.com	28 99999-9999	04/03/2023	

Desenvolvido pelo prof. Vítor E. Silva Souza & o time do LabES para DI & PPGI/UFES. 1.0.0

Figura 22 – Tela de Importar Alunos - Listagem e edição de dados.

**Importar Estudantes** importe estudantes de graduação a partir de um arquivo CSV.

**Estudantes importados com sucesso**

Matricula	Nome Completo	E-mail	Data de Início
2018100333	Aluno 3	aluno3@email.com	05/03/2016
2018100444	Aluno 4	aluno4@email.com	05/03/2018
2019100555	Aluno 5	aluno5@email.com	05/03/2019
2021100111	Aluno 1	aluno1@email.com	03/03/2021
2023100222	Aluno 2	aluno2@email.com	04/03/2023

Desenvolvido pelo prof. Vítor E. Silva Souza & o time do LabES para DI & PPGI/UFES. 1.0.0

Figura 23 – Tela de Importar Alunos - Listagem dados importados.

## 5.2 Gerenciar Projetos Pedagógicos de Curso

O caso de uso Gerenciar PPCs é mais simples e padrão em relação ao restante do sistema do Marvin. Este caso de uso consiste basicamente em um CRUD (*create*, *read*, *update* e *delete*), permitindo ao Coordenador de Curso de Graduação visualizar, cadastrar, editar e excluir PPCs, utilizando duas telas distintas.

A Figura 24 mostra a tela inicial da funcionalidade, a lista de PPCs do Coordenador. Nesta tela, é possível criar visualizações específicas usando os filtros, facilitando a busca de um PPC específico. Existem botões que desencadeiam operações do CRUD, como o botão “Novo” para cadastrar um novo PPC, o botão para excluir um registro e a opção de selecionar vários itens para exclusão em lote, além do botão para permitir a edição de um registro.

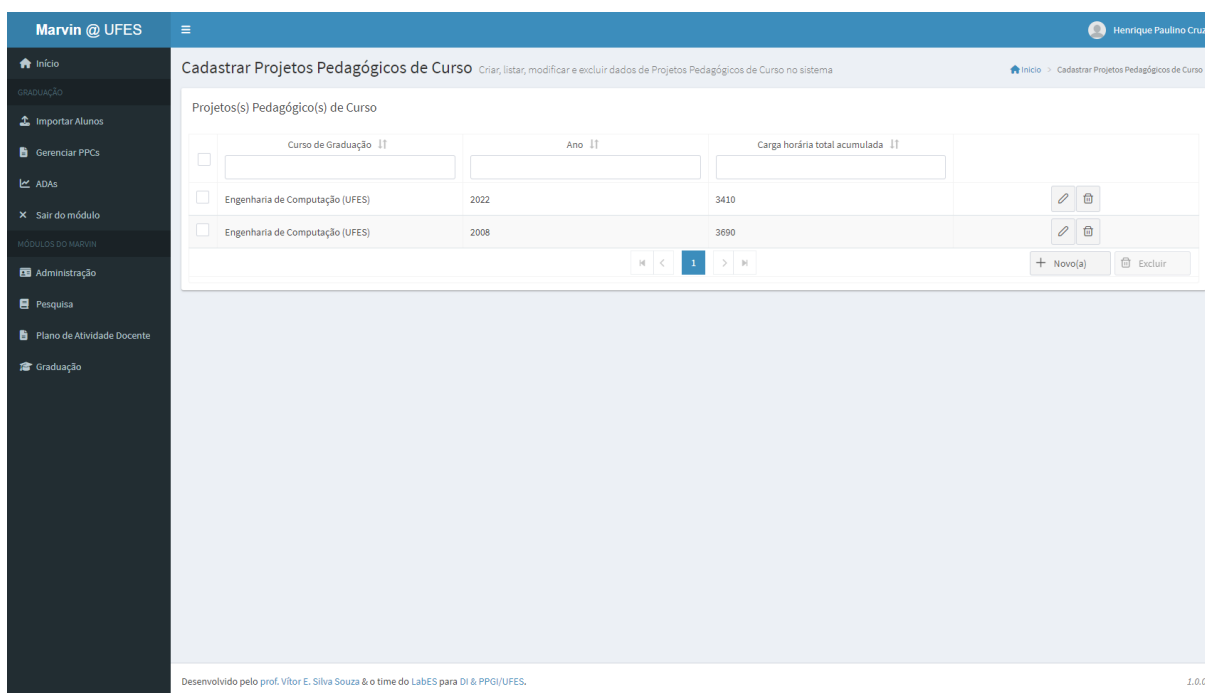


Figura 24 – Tela de Gerenciar PPCs - Listagem.

Para realizar outras operações do CRUD, o Coordenador utiliza a tela representada na Figura 25. Este formulário possui todas as informações de um PPC e tem dois estados iniciais.

O primeiro estado ocorre quando o Coordenador acessa o formulário para cadastrar um novo PPC. Nesse caso, o formulário está em branco, e o Coordenador preenche os dados do novo PPC desejado. Após o preenchimento, ao submeter o formulário, se houver algum erro, o sistema sinaliza o campo com erro e informa sobre o ocorrido; caso contrário, o PPC é cadastrado e o usuário é redirecionado para a tela de listagem.

O segundo estado acontece quando o Coordenador deseja editar um PPC. Nesse caso, o formulário é preenchido com as informações do PPC, permitindo ao Coordenador editar qualquer dado. Se alguma informação não for válida, o sistema informa ao Coordenador; caso contrário, as informações são alteradas e o usuário é redirecionado para a tela de listagem.



The screenshot shows a web interface for managing Pedagogical Projects (PPCs). The header includes 'Marvin @ UFES' and the user name 'Henrique Paulino Cruz'. The main content area is titled 'Cadastrar Projetos Pedagógicos de Curso' and contains a form with the following fields:

- Curso de Graduação: Engenharia de Computação (dropdown)
- Ano: [input field]
- Carga horária total acumulada: [input field]
- Número de períodos sugeridos: [input field]
- Número máximo de períodos: [input field]
- Carga horária máxima por semestre: [input field]
- Carga horária obrigatória: [input field]
- Incluir carga horária obrigatória no total:
- Carga horária optativa: [input field]
- Incluir carga horária optativa no total:
- Carga horária de TCC: [input field]
- Incluir carga horária de TCC no total:
- Carga horária de estágio: [input field]
- Incluir carga horária de estágio no total:
- Carga horária complementar: [input field]
- Incluir carga horária complementar no total:
- Carga horária de extensão: [input field]
- Incluir carga horária de extensão no total:

Buttons for 'Salvar' and 'Cancelar' are located at the bottom of the form.

Figura 25 – Tela de Gerenciar PPCs - Formulário.

### 5.3 Acompanhamento de Desempenho Acadêmico

Por fim, os casos de uso Gerenciar ADAs, Importar ADAs e Exportar ADAs foram agrupados no menu “ADAs”. Ao clicar nele, o usuário visualiza a tela da Figura 26. Esta tela serve apenas para que o Coordenador tenha visibilidade das operações que pode realizar, com uma descrição para facilitar o uso do sistema.

The screenshot shows the 'Acompanhamento de Desempenho Acadêmico' (ADA) start screen. The header includes 'Marvin @ UFES' and the user name 'Henrique Paulino Cruz'. The main content area is titled 'Acompanhamento de Desempenho Acadêmico' and contains two buttons:

- Importar Acompanhamento de Desempenho Acadêmico: Importe Acompanhamentos de Desempenho Acadêmico a partir de um arquivo CSV.
- Gerenciar Acompanhamento de Desempenho Acadêmico: Criar, listar, modificar e excluir dados de Acompanhamentos de Desempenho Acadêmico no sistema.

The footer of the page reads: 'Desenvolvido pelo prof. Vítor E. Silva Souza & o time do LabES para DI & PPGI/UFES. 1.0.0'.

Figura 26 – Tela de ADAs - Início.

O caso de uso Importar ADAs é dividido em várias etapas, semelhantes ao processo de Importar Alunos.

A Figura 27 representa a primeira etapa da funcionalidade Importar ADAs. Aqui, o Coordenador informa o curso, o ano e período de importação, e faz o upload de um arquivo CSV contendo os dados dos ADAs a serem importados. Com o formulário preenchido, o Coordenador pode prosseguir para a próxima etapa.

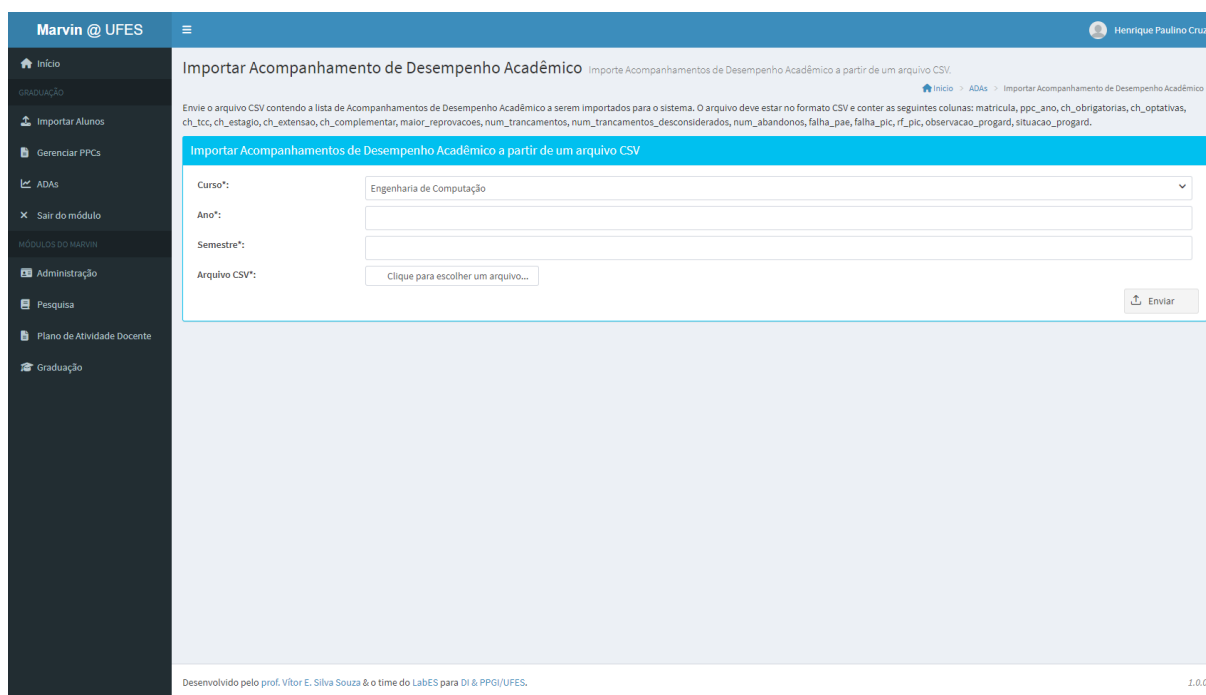


Figura 27 – Tela de Importar ADAs - Submissão de arquivo.

A Figura 28 mostra a tela em que o Coordenador pode visualizar todos os dados dos ADAs encontrados no arquivo CSV submetido. Assim como em Importar Alunos, o Coordenador pode clicar em cada célula da tabela para corrigir ou alterar informações. Existem validações nos dados, como a detecção de conflitos, i.e., quando um ADA já foi submetido para o mesmo aluno no mesmo ano e período.

A tela de conflito, mostrada na Figura 29, exibe as importações que deram conflito. O Coordenador pode corrigir conflitos escolhendo os registros que deseja sobrescrever.

A Figura 30 representa a última etapa da funcionalidade Importar ADAs. Aqui, o Coordenador visualiza todos os dados efetivamente importados. Nessa tela, não é possível mais realizar edições, apenas visualizações.

O caso de uso Gerenciar ADAs é semelhante ao Gerenciar PPCs, mas não permite a criação de novos registros.

A Figura 31 ilustra a tela inicial da funcionalidade de gerenciamento de Acompanhamentos de Desempenho Acadêmico (ADAs), onde o Coordenador pode visualizar, editar, excluir registros e criar visualizações específicas usando filtros. Além disso, a funcionalidade

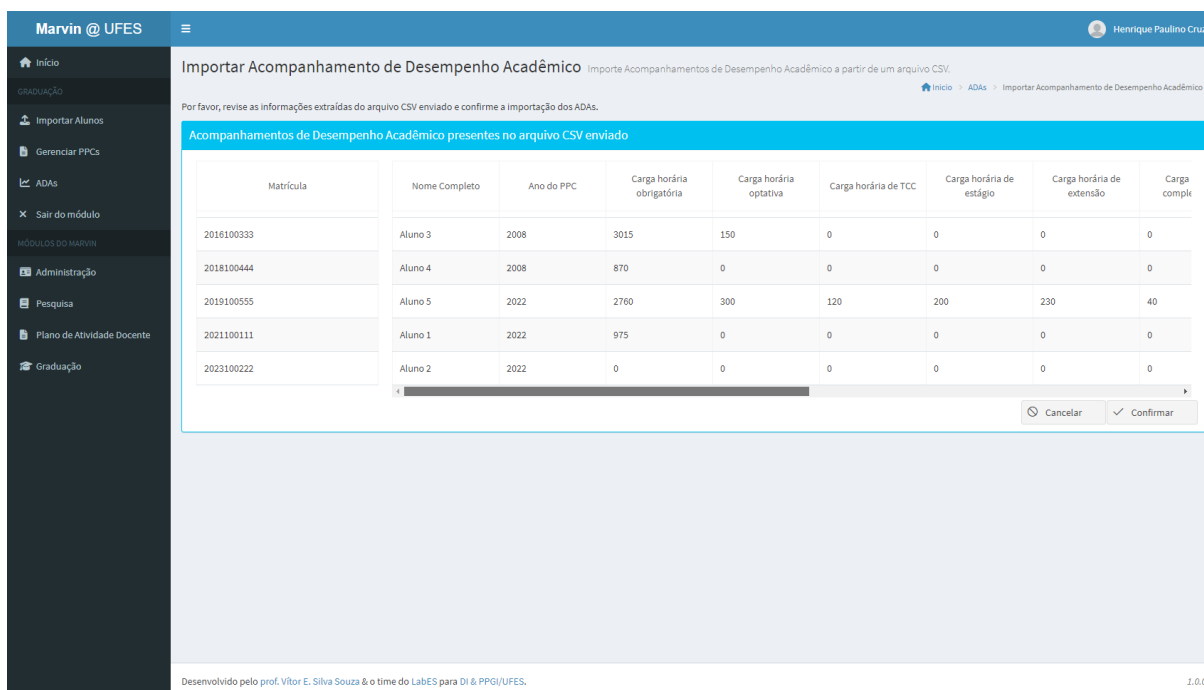


Figura 28 – Tela de Importar ADAs - Listagem e edição de dados.

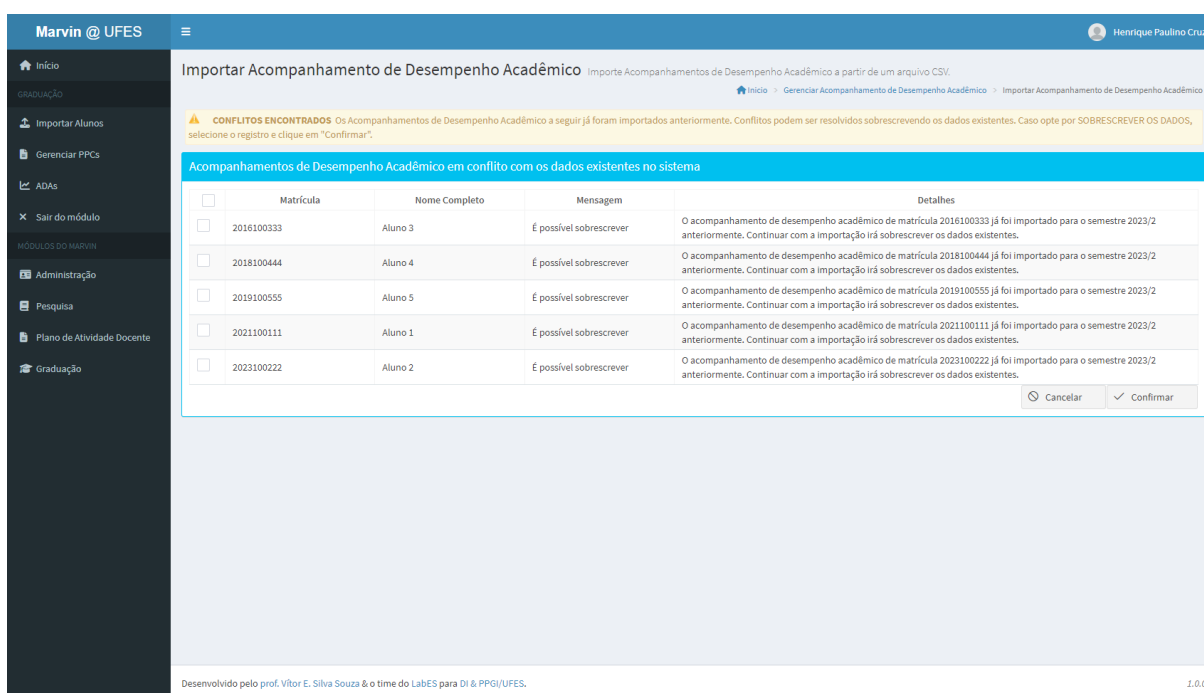


Figura 29 – Tela de Importar ADAs - Conflito de versões.

permite a exportação das visualizações para um arquivo CSV (resultado pode ser visto na Figura 32), proporcionando ao Coordenador a capacidade de baixar todas as informações contidas nos ADAs dos alunos, facilitando o compartilhamento de dados e a criação de *backups* externos.

A tela do formulário, ilustrada na Figura 33, permite ao Coordenador visualizar, editar e finalizar ADAs. Existe a opção de “Finalizar”, que coloca o registro como finalizado,

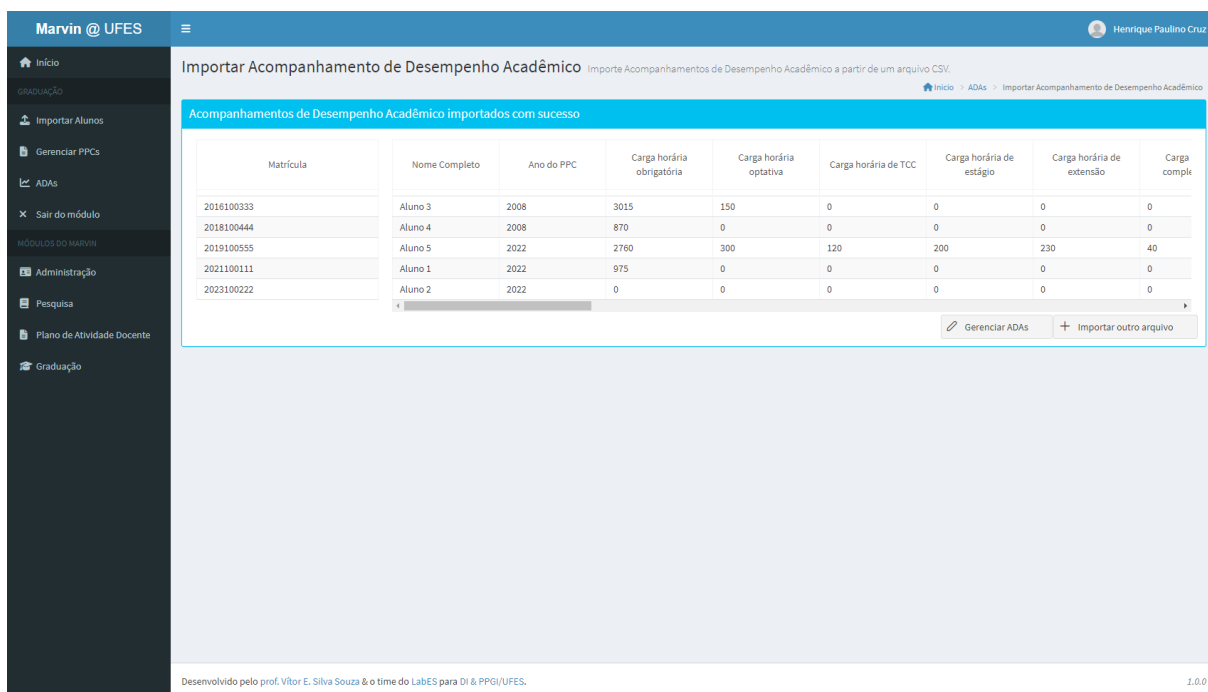


Figura 30 – Tela de Importar ADAs - Listagem dados importados.

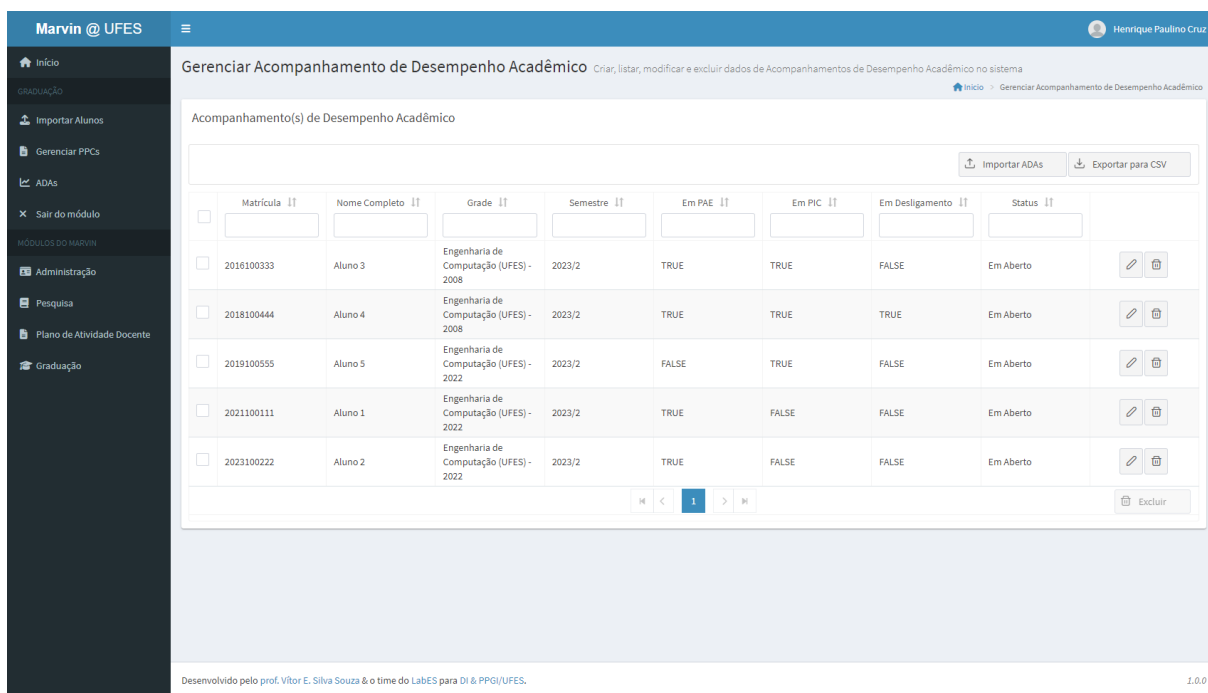


Figura 31 – Lista de ADAs para Gerenciamento.

impedindo modificações futuras, a menos que seja reaberto através da tela de modificações.

The screenshot shows a Google Sheets interface with a spreadsheet titled "Exportação". The spreadsheet contains a table with the following data:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	
1	Matricula	Nome Completo	Grade	Semestre	Carga horária ot	Carga horária oç	Carga horária de	Carga horária de	Carga horária de	Carga horária de	Carga horária de	Carga horária ac	Maior número de	Trancamentos	Trancamentos a Ab
2	2018100333	Aluno 3	Engenharia de C	2023/2	3015	150	0	0	0	0	0	3165	2	0	2
3	2018100444	Aluno 4	Engenharia de C	2023/2	870	0	0	0	0	0	0	870	5	0	2
4	2019100555	Aluno 5	Engenharia de C	2023/2	2760	300	120	200	230	40	3410	0	0	0	2
5	2021100111	Aluno 1	Engenharia de C	2023/2	975	0	0	0	0	0	975	1	0	0	0
6	2023100222	Aluno 2	Engenharia de C	2023/2	0	0	0	0	0	0	0	0	0	0	0
7															
8															

Below the table, there is a button labeled "Adicionar" with a text input field containing "1000" and the text "mais linhas na parte de baixo".

Figura 32 – Tela de Gerenciar PPCs - Listagem.

Marvin @ UFES
Henrique Paulino Cruz

Gerenciar Acompanhamento de Desempenho Acadêmico
Criar, listar, modificar e excluir dados de Acompanhamentos de Desempenho Acadêmico no sistema

Inicio > Gerenciar Acompanhamento de Desempenho Acadêmico

Detalhes do Acompanhamento de Desempenho Acadêmico

**Informações Gerais**

Matrícula	2016100333
Nome Completo	Aluno 3
Grade	Engenharia de Computação (UFES) - 2008
Status	Em Aberto
Carga horária obrigatória	3015
Carga horária optativa	150
Carga horária de TCC	0
Carga horária de estágio	0
Carga horária de extensão	0
Carga horária complementar	0
Carga horária acumulada	3165
Maior número de reprovações	2
Trancamentos	0
Trancamentos a serem desconsiderados	2
Abandonos	0
Semestres Integralizados	10
Semestres Corridos	10
Faltam para NPM	2

**Critérios PAE**

Caraga horária acumulada sobre o número de semestres integralizados	243
Carga horária insuficiente	<input checked="" type="checkbox"/> Sim
Número de abandonos maiores ou iguais a 1	<input type="checkbox"/> Não
Número de reprovações maiores ou iguais a 3	<input type="checkbox"/> Não
Em PAE	<input checked="" type="checkbox"/> Sim

**Critérios PIC**

Número de semestres integralizados maior que o número de semestres sugeridos	<input checked="" type="checkbox"/> Sim
Número de abandonos maiores ou iguais a 2	<input type="checkbox"/> Não
Falha PAE	<input type="checkbox"/> Não
Em PIC	<input checked="" type="checkbox"/> Sim

**Critérios Desligamento**

Número de abandonos maiores ou iguais a 3	<input type="checkbox"/> Não
Recuperação por falta em PIC	<input type="checkbox"/> Não
Falha PIC	<input type="checkbox"/> Não
Número de semestres integralizados maior que o número de semestres máximos	<input type="checkbox"/> Não
Impossibilidade de integralização	<input type="checkbox"/> Não
Em Desligamento	<input type="checkbox"/> Não

**Resultados**

Situação da PROGRAD	PIC
Observações da PROGRAD	Aluno em PIC
Ação da Colegiado	
Observações Atuais	

Finalizar
 Salvar
 Cancelar

Desenvolvido pelo prof. Vítor E. Silva Souza & o time do LabES para DI & PPGI/UFES.
1.0.0

Figura 33 – Tela de Gerenciar PPCs - Formulário.

## 6 Conclusão

Neste capítulo, abordam-se as considerações finais acerca do trabalho desenvolvido, destacando suas principais contribuições, limitações, lições aprendidas ao longo do desenvolvimento, desafios enfrentados e perspectivas para trabalhos futuros. A Seção 6.1 abrange as contribuições do trabalho e descreve as dificuldades enfrentadas, enquanto a Seção 6.2 oferece uma reflexão pessoal sobre a ferramenta, estabelecendo um paralelo entre os objetivos iniciais e o resultado alcançado. Por fim, são apresentadas opiniões sobre possíveis melhorias futuras com base na experiência adquirida.

### 6.1 Considerações Finais

A criação do *Módulo de Otimização e Centralização das Atividades Acadêmicas da Graduação do Marvin* representa um avanço importante para consolidar o Marvin como um sistema dedicado a auxiliar a comunidade acadêmica. Esse módulo oferece a oportunidade de simplificar e otimizar diversas atividades que costumam ser realizadas de maneira diversificada por diferentes atores. No entanto, é crucial destacar suas limitações, uma vez que, apesar da maior centralização das atividades, ainda é necessário utilizar o sistema da UFES para concluir efetivamente algumas operações.

Além disso, considerando que nem todas as funcionalidades levantadas nos requisitos no Capítulo 3 foram implementadas no Marvin, o trabalho realizado abre caminho para que tais recursos sejam incorporados por meio de contribuições de alunos e projetos de extensão. Não obstante, com as funcionalidades já desenvolvidas, o Coordenador pode conduzir uma rodada de Acompanhamento de Desempenho Acadêmico de um curso de forma mais prática do que pelos métodos anteriores.

Destaca-se a importância do Capítulo 2 na fundamentação e no desenvolvimento do módulo. A elaboração desse capítulo, após revisar uma ampla variedade de materiais acadêmicos e da comunidade de programação, proporciona uma confiança substancial na qualidade do trabalho. O resultado foi a criação de documentos sólidos de Requisitos de Sistema e de Projeto de Sistema, que oferecem clareza abrangente sobre a implementação das funcionalidades do módulo.

Sem dúvidas, as maiores dificuldades enfrentadas durante o trabalho surgiram na fase de implementação dos requisitos. A *stack* de tecnologias usada para desenvolver o Marvin não é a que eu geralmente utilizo no meu dia a dia. Então, problemas básicos acabavam aparecendo com frequência e levando um tempo considerável para serem resolvidos.

Apesar desses obstáculos, com uma base sólida adquirida ao longo do curso de

Engenharia de Computação, abrangendo disciplinas como Programação, Programação Orientada a Objetos, Engenharia de Software, Estrutura de Dados, Banco de Dados e participação em projetos de extensão como o LabES,<sup>1</sup> foi possível superar todos os desafios e desenvolver um software de alta qualidade.

A Tabela 8 proporciona uma visão concisa da realização dos objetivos delineados no Capítulo 1. Cada etapa, desde o levantamento de requisitos até a implementação e teste do código-fonte, foi totalmente cumprida. Adicionalmente, a tabela esclarece em qual parte do trabalho cada um dos objetivos foi abordado.

Tabela 8 – Cumprimento dos objetivos delineados no Capítulo 1.

Objetivos	Satisfação	Evidência
Levantamento e documentação dos requisitos	Completamente Satisfeito	Capítulo 3
Projeto e documentação da arquitetura	Completamente Satisfeito	Capítulo 4
Implementar e testar o código-fonte	Completamente Satisfeito	Capítulo 5

## 6.2 Trabalhos Futuros

Um *software* é uma entidade dinâmica e, portanto, requer manutenções contínuas para garantir seu desempenho ideal, além de melhorias contínuas para se adaptar às evoluções nas necessidades dos usuários. Reconhecer essa natureza vital do *software* é crucial, destacando a necessidade de futuros trabalhos que aprimorem o que já foi desenvolvido. Após uma análise aprofundada e diálogo com os *stakeholders* originais, vários pontos foram identificados como áreas potenciais para desenvolvimentos futuros, que poderiam ser facilmente abordados por estudantes em trabalhos subsequentes. Esses pontos incluem:

- Implementar integralmente casos de uso previamente identificados como requisitos, como a Submissão de Atividades Acadêmicas e o Gerenciamento de Prazos para o Coordenador de Curso. Essa expansão do sistema busca oferecer uma cobertura mais abrangente das necessidades acadêmicas, proporcionando aos usuários uma variedade completa de funcionalidades;
- Aperfeiçoar a funcionalidade de visualização dos Acompanhamentos de Desempenho Acadêmico (ADAs), permitindo que os usuários personalizem as colunas exibidas.

<sup>1</sup> <<https://labes.inf.ufes.br/projetos/marvin/>>



Essa melhoria proporcionará uma experiência mais flexível e adaptável, permitindo que cada usuário ajuste a visualização de acordo com suas preferências e necessidades específicas;

- Aprimorar a funcionalidade de importação completa de alunos, incluindo todo o histórico acadêmico. Essa melhoria visa uma migração de dados mais abrangente e eficiente, garantindo que o sistema tenha acesso a informações detalhadas sobre a trajetória acadêmica dos alunos, contribuindo para uma gestão mais completa e informada;
- Reforçar a funcionalidade de visualização dos Acompanhamentos de Desempenho Acadêmico (ADAs), permitindo que os usuários personalizem as colunas exibidas. Isso proporcionará uma experiência mais flexível e adaptável, permitindo que cada usuário ajuste a visualização de acordo com suas preferências e necessidades específicas;
- Aplicar os princípios da ciência de dados para desenvolver um modelo que contribua para a avaliação dos alunos de cursos de graduação. Esse modelo tem o potencial de gerar análises e previsões cruciais sobre o desempenho dos alunos, incluindo probabilidades de evasão e outras métricas relevantes;
- Realizar testes de esforço envolvendo membros experientes da comunidade acadêmica em segurança para avaliar a robustez do sistema. Esses testes buscam identificar potenciais brechas e validar a eficácia das medidas de segurança implementadas. Observar minuciosamente os usuários finais durante a interação com o sistema, com foco especial na autenticação por login e senha, assegurando que esse processo seja isento de erros e que os usuários tenham acesso adequado às funcionalidades correspondentes ao seu perfil.

# Referências

- BADALOTTI, G. M. Introdução ao desenvolvimento de sistemas web. UNIASSELVI, 2018. Citado na página 19.
- BARCELLOS, M. P. Engenharia de software, notas de aula. Vitória: UFES, 2018. Disponível em: <<https://nemo.inf.ufes.br/wp-content/uploads/Monalessa/EngSoftware/NotasDeAula-EngSw-EngComp-v2018.pdf>>. Citado 4 vezes nas páginas 15, 16, 17 e 18.
- BERGER, M. *Data Access Object Pattern*. 2005. <<https://api.semanticscholar.org/CorpusID:8351558>>. Citado na página 23.
- CHANE, V. A. *Macs - Sistema de controle de acessos do Marvin*. Vitória, ES, Brasil, 2022. Citado 2 vezes nas páginas 27 e 31.
- DELAMARO, M.; JINO, M.; MALDONADO, J. *Introdução ao teste de software*. [S.l.]: Elsevier Brasil, 2013. Citado 2 vezes nas páginas 24 e 25.
- FALBO, R. d. A. *Projeto de Sistema de Software*. 2018. <[http://www.inf.ufes.br/~vitorsouza/falbo/Notas\\_Aula\\_Projeto\\_Sistemas\\_Falbo\\_2018.pdf](http://www.inf.ufes.br/~vitorsouza/falbo/Notas_Aula_Projeto_Sistemas_Falbo_2018.pdf)>. Notas de Aula. Citado 3 vezes nas páginas 15, 18 e 27.
- FARIA, T. Java EE 7 com JSF, PrimeFaces e CDI. *SI sn*, 2015. Citado na página 20.
- FOWLER, M. *Patterns of Enterprise Application Architecture*. 1. ed. [S.l.]: Addison-Wesley, 2002. ISBN 9780321127426. Citado na página 40.
- LOUDON, K. Desenvolvimento de grandes aplicações web. *Revista Telfrac*, v. 1, n. 1, 2018. Citado na página 19.
- MALDONADO, J. C. et al. Padrões e frameworks de software. *Notas Didáticas, Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, ICMC/USP, São Paulo, SP, Brasil*, p. 28, 2002. Citado 3 vezes nas páginas 5, 19 e 20.
- MARTIN, R. *Arquitetura Limpa: O guia do artesão para estrutura e design de software*. Alta Books, 2019. (Robert C. Martin). ISBN 9788550808161. Disponível em: <<https://books.google.com.br/books?id=BOaPDwAAQBAJ>>. Citado 2 vezes nas páginas 21 e 22.
- SOUZA, V. E. S. The FrameWeb Approach to Web Engineering: Past, Present and Future. In: ALMEIDA, J. P. A.; GUIZZARDI, G. (Ed.). *Engineering Ontologies and Ontologies for Engineering*. 1. ed. Vitória, ES, Brazil: NEMO, 2020. cap. 8, p. 100–124. ISBN 9781393963035. Disponível em: <<http://purl.org/nemo/celebratingfalbo>>. Citado 2 vezes nas páginas 5 e 40.
- VALENTE, M. T. *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. [S.l.]: Editora: Independente, 2020. Citado na página 23.
- VAZQUEZ, C. E.; SIMÕES, G. S. *Engenharia de Requisitos: software orientado ao negócio*. [S.l.]: Brasport, 2016. Citado na página 17.

WAZLAWICK, R. S. *Engenharia de Software, Conceitos e Práticas*. [S.l.]: Elsevier Editora Ltda, 2013. Citado 2 vezes nas páginas [15](#) e [24](#).

# Apêndices

# APÊNDICE A – Documento de Requisitos de Sistema



Documento de Requisitos de Sistema

# **Módulo de otimização e centralização das atividades acadêmicas da Graduação do Marvin**

Vitória, ES

2023

## Registro de Alterações:

<b>Versão</b>	<b>Responsável</b>	<b>Data</b>	<b>Alterações</b>
1.0	Henrique Paulino Cruz	09/03/2023	Versão inicial das estórias de usuário.
1.1	Henrique Paulino Cruz	16/03/2023	Adiciona critérios de aceitação nas estórias de usuário.
1.1	Vitor Estevão Silva Souza	22/03/2023	Revisão das estórias de usuário.
1.2	Henrique Paulino Cruz	27/04/2023	Finaliza capítulos 2, 3 e 4.
1.2	Vitor Estevão Silva Souza	02/05/2023	Revisão dos capítulos 2, 3 e 4.
1.3	Henrique Paulino Cruz	18/05/2023	Finaliza capítulo 1.
1.3	Vitor Estevão Silva Souza	25/05/2023	Revisão do capítulo 1.
1.4	Henrique Paulino Cruz	29/05/2023	Finaliza capítulo 5.
1.4	Vitor Estevão Silva Souza	15/06/2023	Revisão do capítulo 5.
1.5	Henrique Paulino Cruz	10/07/2023	Finaliza capítulo 6.

# 1 Introdução

Este documento apresenta a especificação dos requisitos do sistema *Módulo de otimização e centralização das atividades acadêmicas da Graduação do Marvin*. Esta especificação foi construída aplicando-se técnicas de levantamento de requisitos, bem como modelagem de casos de uso e de classes utilizando a linguagem UML.

A Seção 2 descreve os requisitos levantados junto aos *stakeholders*. A Seção 3 explica a divisão em subsistemas, descrevendo brevemente cada um deles. A Seção 4 apresenta o modelo de casos de uso, incluindo descrições de atores, os diagramas de casos de uso e suas respectivas descrições. A Seção 5 traz os modelos conceituais estruturais do sistema na forma de diagramas de classes. Por fim, a Seção 6 detalha o dicionário do projeto, contendo as definições das classes identificadas.

## 2 Definição de Requisitos

Esta seção descreve o resultado da atividade de levantamento de requisitos. A Subseção 2.1 resume de forma geral o propósito do sistema e a motivação para seu desenvolvimento. A Subseção 2.2 descreve o minimundo do sistema, apresentando superficialmente suas principais características. A Subseção 2.3 lista os requisitos de usuário do sistema, na forma de histórias de usuário e requisitos não-funcionais.

### 2.1 Descrição do Propósito do Sistema

Após análise do sistema atualmente utilizado pela comunidade acadêmica, mais especificamente pelos envolvidos na graduação, para gerenciamento das demandas acadêmicas, constatou-se que os acadêmicos precisam realizar uma grande quantidade de trabalho manual e que há dificuldades na comunicação entre as diversas entidades envolvidas.

Essa situação tem gerado sobrecarga de tarefas para os Coordenadores de Curso de Graduação, que precisam cumprir prazos e fornecer relatórios à Pró-Reitoria de Graduação (Prograd). Além disso, há um grande volume de e-mails recebidos, que é o único canal disponibilizado para solicitações de demandas e esclarecimento de dúvidas dos acadêmicos.

Diante disso, concluiu-se que a Graduação necessita de um sistema de software que possa gerenciar os diversos prazos que os Coordenadores de Curso devem cumprir, automatizar a geração do Acompanhamento do Desempenho Acadêmico (ADA) e permitir que a comunidade acadêmica de graduação centralize solicitações de diversas atividades



acadêmicas essenciais para a formação dos alunos. Para atender a essa necessidade, pretende-se implementar um módulo do sistema Marvin.<sup>1</sup>

## 2.2 Descrição do Minimundo

O Marvin é um sistema Web desenvolvido desde 2019 com o objetivo de gerenciar atividades no meio acadêmico. Ele integra módulos criados por estudantes de graduação da UFES ao longo dos anos, sob a orientação do professor Vítor E. Silva Souza.

O propósito deste módulo, implementado neste estudo, é atender às necessidades da graduação e reduzir a sobrecarga de trabalho manual e o uso excessivo de e-mails. Para alcançar esse objetivo, foram identificadas as seguintes necessidades.

### 2.2.1 Gerenciamento de Prazos

O cargo de Coordenador de Curso de Graduação requer o cumprimento de diversos prazos, com diferentes características, como prazos únicos, recorrentes, de início e fim, e de recorrência customizada.

Para simplificar o controle desses prazos, é necessário um sistema que permita ao Coordenador gerenciá-los de forma simples e objetiva. Esse sistema deve fornecer uma visualização classificada por ordem de prioridade, uma interface fácil para cadastrar e editar prazos, além de permitir a importação de prazos a partir de um arquivo CSV para uma usabilidade mais dinâmica.

### 2.2.2 Computação do Acompanhamento do Desempenho Acadêmico

Na UFES, o Acompanhamento do Desempenho Acadêmico (ADA) é um programa que identifica alunos enfrentando dificuldades em disciplinas de graduação e oferece um acompanhamento mais direto em diferentes níveis de atuação, visando ajudá-los a obter a formação desejada.

No entanto, identificar quais alunos precisam desse acompanhamento é uma tarefa desafiadora, exigindo um trabalho manual do Coordenador de Curso de Graduação. Essa tarefa pode ser automatizada, restando ao Coordenador apenas a análise qualitativa para tomar a decisão final sobre as medidas a serem adotadas para cada aluno.

O Apêndice C fornece uma explicação detalhada sobre a computação do ADA para um aluno. Resumidamente, existem dois tipos de acompanhamento: o Plano de Acompanhamento de Estudos (PAE) e o Plano de Integralização Curricular (PIC).

Diante dessa alta demanda de trabalho manual, é necessário um sistema que permita

---

<sup>1</sup> <<https://gitlab.labes.inf.ufes.br/marvin/marvin>>

ao Coordenador importar informações de diversos alunos, realizar cálculos automáticos e apresentar os resultados para validação e edição. Além disso, o Coordenador deve realizar uma análise qualitativa, indicando a situação final do aluno e fazendo observações, a fim de concluir o processo.

### 2.2.3 Submissão de Atividades Acadêmicas

Na comunidade acadêmica, há vários processos que envolvem submissões, com etapas de envio, aprovação, rejeição e arquivamento. Alguns exemplos recorrentes dessas atividades são:

- Submissão de Horas: os alunos submetem suas horas de extensão e complementares para aprovação pelo coordenador do curso.
- Submissão de Estágio: os alunos submetem seus termos de compromisso, aditivo, rescisão ou relatórios de estágio para aprovação pelo coordenador de estágios do curso.

Atualmente, essas submissões são realizadas por meio de caixas de e-mail específicas. No entanto, em determinados períodos, há uma alta demanda nesses canais de comunicação, o que dificulta a realização e o gerenciamento dessas tarefas.

Diante dessa situação, surge a necessidade de um sistema centralizado, que simplifique e objetive as submissões para todos os envolvidos. Nesse sistema, alunos e professores preencheriam formulários com todas as informações necessárias para cada tipo de submissão. Além disso, eles receberiam notificações por e-mail sobre o andamento das submissões.

Os coordenadores teriam acesso ao sistema, onde poderiam visualizar, aprovar, rejeitar e arquivar as submissões. Também seriam notificados por e-mail sempre que uma nova submissão fosse enviada.

## 2.3 Requisitos de Usuário

Tomando por base o contexto do sistema descrito na Seção 2.2 e considerando como principais *stakeholders*, coordenadores, professores e alunos, foram identificados os seguintes requisitos — na forma de histórias de usuário — e regras do negócio.

As histórias de usuário são apresentadas na Tabela 1 e os requisitos não-funcionais globais (ou seja, aqueles que não são caracterizados como critérios de aceitação de histórias de usuário específicas) na Tabela 2.

É importante ressaltar que este módulo do Marvin parte do pressuposto que há outro módulo responsável pelos cadastros básicos das unidades administrativas da

universidade (ex.: centros de ensino, cursos de graduação) bem como dos acadêmicos e suas relações com tais unidades administrativas (ex.: coordenador/secretário de curso de graduação, estudante de graduação). A lista abaixo indica as estórias de usuário que pressupomos já tenham sido implementadas nesse outro módulo:

- US-1: como Administrador do sistema, quero cadastrar centros de ensino, para que cursos de graduação possam ser associados a eles;
- US-2: como Administrador do sistema, quero cadastrar cursos de graduação, para que os acadêmicos possam se associar de alguma maneira a eles (ex.: estudante matriculado, etc.).
- US-3: como Administrador do sistema, quero cadastrar Acadêmicos no sistema, para que eles possam usar as funcionalidades do mesmo;
- US-4: como Administrador do sistema, quero cadastrar Professores em cargos de coordenação de cursos de graduação, para que esses possam ter acesso a funcionalidades específicas deste papel;
- US-5: como Administrador do sistema, quero cadastrar Professores em cargos de coordenação de horas complementares e extensão de cursos de graduação, para que esses possam ter acesso a funcionalidades específicas deste papel;
- US-6: como Administrador do sistema, quero cadastrar Professores em cargos de coordenação de estágio de cursos de graduação, para que esses possam ter acesso a funcionalidades específicas deste papel;
- US-7: como Administrador do sistema ou Coordenador de Curso, quero cadastrar a matrícula de Alunos em cursos de graduação, para que eles possam ter acesso às funcionalidades correspondentes aos estudantes no sistema;

Tabela 1 – Estórias de Usuário.

<b>ID:</b>	US-8	<b>Depende:</b>	US-4	<b>Prioridade:</b>	Alta
<b>Descrição:</b>	Cadastro/CRUD: Como Coordenador de Curso, quero cadastrar prazos, para acompanhar suas datas e atividades relacionadas.				

<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"><li>– CA1: O sistema deve permitir o cadastro de prazos com quatro tipos diferentes: prazos únicos, prazos recorrentes, prazos com data de início e fim ou prazos encadeados;</li><li>– CA2: Para cadastrar um prazo único, o Coordenador deve informar a data-hora final, descrição e título do prazo;</li><li>– CA3: Para cadastrar um prazo recorrente, o Coordenador deve informar a data-hora inicial e final, descrição, título e a frequência em que o prazo irá ocorrer sendo ela: “Diária”, “Semanal”, “Mensal”, “Anual”. Nesse tipo de prazo, a data-hora final marca até quando a recorrência ocorrerá, caso esse campo esteja vazio, a recorrência será para sempre;</li><li>– CA4: Para cadastrar um prazo com data de início e fim, o Coordenador deve informar a data-hora inicial e final, descrição e título do prazo;</li><li>– CA5: Para cadastrar um prazo encadeado, o Coordenador deve informar a data-hora final, descrição e título do prazo;</li><li>– CA6: O status de um prazo pode ter os valores: “Criado”, “Vencido”, “Adiado” ou “Concluído”;</li><li>– CA7: Ao cadastrar um prazo, seu status é definido como “Criado”;</li><li>– CA8: É possível cadastrar um prazo sem informar uma hora, caso em que o sistema assumirá a hora de 23:59;</li><li>– CA9: O Coordenador deve ter a capacidade de realizar alterações ou exclusões em prazos existentes;</li><li>– CA10: Caso o Coordenador edite a data-hora final qualquer prazo, para um valor superior ao anterior, o prazo ficará com o status de “Adiado”;</li><li>– CA11: Caso o Coordenador altere o status de um prazo encadeado para “Concluído”, deverá ser disponibilizada a opção de inserir uma nova data-hora final. Isso permitirá a criação de um novo prazo encadeado, com as mesmas informações do prazo anterior, porém com uma nova data-hora final;</li><li>– CA12: A consulta de prazos pode ser feita por título e data-hora final;</li><li>– CA13: Os prazos são de propriedade do Coordenador que os cadastrou, ou seja, não devem estar visíveis ou editáveis para qualquer outro Coordenador;</li><li>– CA14: Todas as alterações ou exclusões de prazos devem ser comunicadas e/ou requerer confirmação do Coordenador antes de serem efetivadas.</li></ul>
--------------------------------	--

<b>ID:</b>	US-9	<b>Depende:</b>	US-8	<b>Prioridade:</b>	Média
<b>Descrição:</b>	Como Coordenador de Curso, quero importar prazos de um arquivo CSV, para agilizar o cadastro de vários prazos.				
<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: A importação de prazos deve ser feita a partir de um arquivo CSV, seguindo o padrão descrito no Apêndice A;</li> <li>– CA2: Ao importar um arquivo, o sistema deve mostrar todos os prazos identificados nele, para que o Coordenador indique quais devem ser importados;</li> <li>– CA3: O sistema deve permitir que o Coordenador modifique cada um dos prazos que serão importados, antes de efetivar a operação;</li> <li>– CA4: Para salvar os dados importados, é necessário confirmação.</li> </ul>				

<b>ID:</b>	US-10	<b>Depende:</b>	US-8	<b>Prioridade:</b>	Alta
<b>Descrição:</b>	Como Coordenador de Curso, quero visualizar meus prazos, para gerenciar e dar prioridades aos meus prazos cadastrados.				
<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: O sistema deve exibir os prazos com vencimento nas próximas semanas (uma semana por padrão com possibilidade de modificar);</li> <li>– CA2: Devem ser apresentadas as seguintes informações para o usuário: título, descrição e data-hora;</li> <li>– CA3: Os prazos devem vir ordenados do que possui a data-hora de vencimento mais próxima para a mais longe;</li> <li>– CA4: Prazos que passaram da data-hora de finalização devem ter um enfoque de alerta;</li> <li>– CA5: Deve ser possível atualizar o status de um prazo a partir dessa visualização.</li> </ul>				

<b>ID:</b>	US-11	<b>Depende:</b>	US-4, US-7	<b>Prioridade:</b>	Alta
<b>Descrição:</b>	Cadastro/CRUD: Como Coordenador de Curso, quero cadastrar um Projeto Pedagógico de Curso (PPC), para ter as informações referentes às atividades que um Aluno precisa cumprir para se formar em um curso.				

<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: Para cadastrar um PPC, é necessário fornecer as seguintes informações: ano, curso sob sua coordenação, carga horária total das disciplinas, número de períodos sugeridos e máximo para conclusão, carga horária máxima semestral permitida, carga horária das disciplinas obrigatórias e optativas, além da carga horária destinada ao Trabalho de Conclusão de Curso (TCC), estágio e atividades complementares e de extensão;</li> <li>– CA2: Se o Coordenador estiver responsável por somente um curso, o campo correspondente a essa informação já estará preenchido com esse curso;</li> <li>– CA3: Ainda no cadastro, para cada uma das cargas horárias, é necessário informar se ela deve ou não ser considerada para o cálculo da carga horária acumulada, as cargas horárias que contemplam essa situação são: disciplinas obrigatórias, disciplinas optativas, TCC, estágio, atividades complementares e atividades de extensão;</li> <li>– CA4: Não deve ser possível cadastrar um PPC com o par ano e curso já informados em outro PPC;</li> <li>– CA5: Um PPC pode ter todas as informações alteradas;</li> <li>– CA6: A consulta pode ser feita por curso e ano;</li> <li>– CA7: Para excluir um PPC, nenhuma outra entidade do sistema pode estar dependendo dele;</li> <li>– CA8: Alterações devem ser comunicadas e/ou deve ser requerida confirmação.</li> </ul>
--------------------------------	--

<b>ID:</b>	US-12	<b>Depende:</b>	US-4, US-7	<b>Prioridade:</b>	Alta
<b>Descrição:</b>	Como Coordenador de Curso, quero importar informações de Alunos a partir de um arquivo, para facilitar o cadastro em massa de Alunos.				

<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: A importação de Alunos deve ser feita a partir de um arquivo CSV, seguindo o padrão descrito no Apêndice B;</li> <li>– CA2: Antes de realizar a operação, o Coordenador deve identificar claramente o curso específico sob sua coordenação ao qual a importação se refere;</li> <li>– CA3: Ao importar um arquivo, o sistema deve mostrar todos os Alunos identificados nele, para que o Coordenador indique quais devem ser importados;</li> <li>– CA4: O sistema deve permitir que o Coordenador modifique cada um dos Alunos que serão importados, antes de efetivar a operação;</li> <li>– CA5: Para salvar os dados importados, é necessária confirmação.</li> </ul>
--------------------------------	---

<b>ID:</b>	US-13	<b>Depende:</b>	US-12	<b>Prioridade:</b>	Alta
<b>Descrição:</b>	Como Coordenador de Curso, quero computar as informações do Acompanhamento de Desempenho Acadêmico (ADA) de um Aluno, para identificar Alunos que precisam de acompanhamento ou desligamento.				

<b>Crerios de Aceitaço:</b>	<ul style="list-style-type: none"> <li>– CA1: O sistema deve armazenar o histrico de todos os ADAs de um Aluno, diferenciados pelo ano-semester vigente;</li> <li>– CA2: Antes de realizar a operao, o Coordenador deve informar o ano-semester vigente do ADA e identificar o curso especifico sob sua coordenao ao qual a importao se refere;</li> <li>– CA3: Se o Coordenador informar o ano-semester do ADA de um Aluno que j possui um ADA, o sistema deve avisar que o ADA daquele ano-semester j foi computado e dar a opo ao Coordenador de continuar, sobrescrevendo as informoes j existentes;</li> <li>– CA4: Quando o sistema computar as informoes do ADA de um Aluno com um novo ano-semester vigente, o ADA atual passa a ser o anterior, e as novas informoes sero referentes ao ADA atual;</li> <li>– CA5: O ADA atual e caracterizado como o ADA com a data de criao mais recente;</li> <li>– CA6: Para computar o ADA de Alunos, ser feita a importao de um arquivo CSV conforme o Apndice C;</li> <li>– CA7: Apes a concluso da operao, o sistema dever registrar no ADA as informoes computadas do Aluno, indicando claramente se ele atendeu aos critrios para ser classificado como “PAE”, “PIC” e/ou “Desligamento”, conforme detalhadamente descrito no Apndice D. Cada um dos critrios deve ser explicitamente apresentado.</li> <li>– CA8: Ao finalizar a operao, o sistema deve gravar no ADA o status como “Aberto”. Esse status pode assumir os valores de “Aberto”, “Modificado” ou “Finalizado”;</li> <li>– CA9: Apes a concluso da operao, o Coordenador deve ser redirecionado para uma tela que exibe todas as informoes dos Alunos que foram usadas como insumo e os resultados computados.</li> </ul>
-----------------------------	--

<b>ID:</b>	US-14	<b>Depende:</b>	US-13	<b>Prioridade:</b>	Alta
<b>Descrio:</b>	Como Coordenador de Curso, desejo consultar todas as informoes do ADA (Acompanhamento de Desempenho Acadmico) dos Alunos cadastrados no sistema, para poder realizar o acompanhamento e observoes necessrias.				



<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: O sistema deve exibir os Alunos do curso de forma paginada e ordenada alfabeticamente;</li> <li>– CA2: Deve ser exibida para o Coordenador o status referente ao ADA e a ação do colegiado de cada um dos Alunos;</li> <li>– CA3: O Coordenador pode informar parâmetros como nome, matrícula e status do ADA para realizar a consulta;</li> <li>– CA4: O Coordenador deve ser capaz de navegar e visualizar todas as informações de um Aluno específico a partir da visualização;</li> <li>– CA5: O Coordenador deve ser capaz de visualizar os ADAs anteriores, fornecendo o ano-semester do ADA desejado, na tela de visualização individual.</li> </ul>
--------------------------------	---

<b>ID:</b>	US-15	<b>Depende:</b>	US-14	<b>Prioridade:</b>	Baixa
<b>Descrição:</b>	Como Coordenador de Curso, quero exportar um arquivo CSV das informações computadas referentes ao ADA, para manipular e compartilhar com outras pessoas que não utilizam o sistema.				
<b>Critérios de Aceitação:</b>	– CA1: Deve ser possível exportar os dados de todos os Alunos de uma vez, de um grupo selecionado ou de forma individual;				

<b>ID:</b>	US-16	<b>Depende:</b>	US-14	<b>Prioridade:</b>	Alta
<b>Descrição:</b>	Como Coordenador de Curso, quero modificar o ADA de um Aluno, para corrigir erros, criar exceções e/ou finalizar seu status.				

<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: Não deve ser possível fazer modificações no ADA com o status de “Finalizado”;</li> <li>– CA2: Todas as informações do ADA podem ser modificadas pelo Coordenador, possibilitando a correção de dados incorretos ou desatualizados;</li> <li>– CA3: Ao modificar qualquer informação do ADA atual, o status do ADA será automaticamente definido como “Modificado”, indicando que foi alterado pelo Coordenador;</li> <li>– CA4: Para atribuir o valor de “Finalizado” para o status do ADA, o Coordenador deve preencher as informações de observações atuais e ação do colegiado, garantindo a conclusão do processo;</li> <li>– CA5: Se o status do ADA já estiver marcado como “Finalizado”, o sistema deve oferecer a opção ao Coordenador de reverter o status para “Modificado”;</li> <li>– CA6: O sistema deve exigir confirmação do Coordenador para qualquer modificação feita, evitando alterações acidentais.</li> </ul>
--------------------------------	---

<b>ID:</b>	US-17	<b>Depende:</b>	US-7, US-5	<b>Prioridade:</b>	Baixa
<b>Descrição:</b>	Como Aluno, quero realizar uma submissão de horas, para que posteriormente o Coordenador de Horas valide-as no sistema da UFES.				

<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: O sistema deve permitir a submissão de horas com dois tipos diferentes: complementares e de extensão;</li> <li>– CA2: Para submeter uma hora complementar devem ser informados título, quantidade de horas e certificado de participação (anexo);</li> <li>– CA3: Para submeter uma hora de extensão devem ser informados título, projeto vinculado, quantidade de horas e certificado de participação (anexo);</li> <li>– CA4: Ao submeter uma hora, o status da submissão automaticamente terá o valor de “Submetida” e deve ser registrada a data da submissão;</li> <li>– CA5: Ao submeter uma hora, o Coordenador de Horas deve receber um e-mail informando o novo registro;</li> <li>– CA6: Alterações podem ser feitas apenas durante fluxos de aprovar e rejeitar ou arquivar, descritos nas estórias <a href="#">US-20</a> e <a href="#">US-21</a>;</li> <li>– CA7: Exclusões não são permitidas.</li> </ul>
--------------------------------	--

<b>ID:</b>	US-18	<b>Depende:</b>	<a href="#">US-17</a>	<b>Prioridade:</b>	Baixa
<b>Descrição:</b>	Como Aluno, quero consultar todas as minhas horas, para saber se elas foram aprovadas ou não.				
<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: Para realizar a consulta, o usuário deve informar como parâmetros: título e status de submissão;</li> <li>– CA2: Devem ser apresentadas as seguintes informações para o usuário: título, quantidade de horas, status e feedbacks da submissão.</li> </ul>				

<b>ID:</b>	US-19	<b>Depende:</b>	<a href="#">US-17</a>	<b>Prioridade:</b>	Baixa
<b>Descrição:</b>	Como Coordenador de Horas, quero consultar todas as horas que os Alunos cadastraram, para aprová-las ou não.				
<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: Para realizar a consulta, o usuário deve informar como parâmetros: título, aluno que submeteu, tipo, status e data da submissão;</li> <li>– CA2: Devem ser apresentadas as seguintes informações para o usuário: título, aluno que submeteu, tipo, anexos, status e data da submissão.</li> </ul>				

<b>ID:</b>	US-20	<b>Depende:</b>	US-19	<b>Prioridade:</b>	Baixa
<b>Descrição:</b>	Como Coordenador de Horas, quero aprovar ou rejeitar uma submissão de horas, para que o Aluno tenha um feedback sobre sua solicitação.				
<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: Para aprovar ou rejeitar uma submissão, é necessário que o status da mesma esteja como “Submetida”;</li> <li>– CA2: Em caso de rejeição, deve ser obrigatório escrever um feedback da submissão;</li> <li>– CA3: Após rejeitar uma submissão, o status da mesma será de “Rejeitada”;</li> <li>– CA4: Após aprovar uma submissão, o status da mesma será de “Aprovada”;</li> <li>– CA5: Aprovando ou rejeitando uma submissão, deve ser registrada a data da resposta;</li> <li>– CA6: Após aprovar ou rejeitar uma submissão, o Aluno deve receber um e-mail informando alteração nos status.</li> </ul>				

<b>ID:</b>	US-21	<b>Depende:</b>	US-20	<b>Prioridade:</b>	Baixa
<b>Descrição:</b>	Como Coordenador de Horas, quero arquivar um cadastro de horas, para finalizar o processo.				
<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: Para arquivar uma submissão, é necessário que o status da mesma esteja como “Aprovada” ou “Rejeitada”;</li> <li>– CA2: Após arquivar uma submissão, o status da mesma será de “Arquivada”, e deve ser registrada a data de arquivamento;</li> </ul>				

<b>ID:</b>	US-22	<b>Depende:</b>	US-7, US-6	<b>Prioridade:</b>	Baixa
<b>Descrição:</b>	Como Aluno, quero realizar uma submissão de estágio, para que posteriormente o Coordenador de Estágio valide-as no sistema da UFES.				

<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: Uma submissão de estágio pode ser de quatro tipos diferentes, sendo eles: “Criação”, “Alteração”, “Rescisão” ou “Relatório”;</li> <li>– CA2: A diferença entre os tipos de submissão de estágio reside no tipo de anexo que o Aluno deve enviar em cada um. Ou seja, essa diferenciação serve apenas para informar ao Coordenador de Estágio como ele deve interpretar aquela submissão;</li> <li>– CA3: Para cadastrar um estágio devem ser informados empresa, observações e anexo;</li> <li>– CA4: Ao submeter um estágio, o status da submissão automaticamente terá o valor de “Submetida”, e deve ser registrada a data da submissão;</li> <li>– CA5: Ao submeter um estágio, o Coordenador de Estágio deve receber um e-mail informando o novo registro;</li> <li>– CA6: Alterações podem ser feitas apenas durante fluxos de aprovar e rejeitar ou arquivar, descritos nas estórias <a href="#">US-25</a> e <a href="#">US-26</a>;</li> <li>– CA7: Exclusões não são permitidas.</li> </ul>
--------------------------------	---

<b>ID:</b>	US-23	<b>Depende:</b>	<a href="#">US-22</a>	<b>Prioridade:</b>	Baixa
<b>Descrição:</b>	Como Aluno, quero consultar todos os meus estágios, para saber se eles foram aprovados ou não.				
<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: Para realizar a consulta, o usuário deve informar como parâmetros: período e empresa;</li> <li>– CA2: Devem ser apresentadas as seguintes informações para o usuário: tipo, período, empresa, status e feedback da submissão.</li> </ul>				

<b>ID:</b>	US-24	<b>Depende:</b>	<a href="#">US-22</a>	<b>Prioridade:</b>	Baixa
<b>Descrição:</b>	Como Coordenador de Estágio, quero consultar todos os estágios que os Alunos submeteram, para aprová-los ou não.				

<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: Para realizar a consulta, o usuário deve informar como parâmetros: período, aluno, empresa, status e data da submissão;</li> <li>– CA2: Devem ser apresentadas as seguintes informações para o usuário: tipo, período, aluno, empresa, anexos, status e data da submissão.</li> </ul>
--------------------------------	---

<b>ID:</b>	US-25	<b>Depende:</b>	US-24	<b>Prioridade:</b>	Baixa
<b>Descrição:</b>	Como Coordenador de Estágio, quero aprovar um cadastro de estágio, para que o Aluno tenha um feedback sobre sua solicitação.				
<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: Para aprovar ou rejeitar uma submissão, é necessário que o status da mesma esteja como “Submetida”;</li> <li>– CA2: Em caso de rejeição, deve ser obrigatório escrever um feedback da submissão;</li> <li>– CA3: Após rejeitar uma submissão, o status da mesma será de “Rejeitada”;</li> <li>– CA4: Após aprovar uma submissão, o status da mesma será de “Aprovada”;</li> <li>– CA5: Aprovando ou rejeitando uma submissão, deve ser registrada a data da resposta;</li> <li>– CA6: Após aprovar ou rejeitar uma submissão, o Aluno deve receber um e-mail informando alteração nos status.</li> </ul>				

<b>ID:</b>	US-26	<b>Depende:</b>	US-25	<b>Prioridade:</b>	Baixa
<b>Descrição:</b>	Como Coordenador de Estágio, quero arquivar um cadastro de estágio, para finalizar o processo.				
<b>Critérios de Aceitação:</b>	<ul style="list-style-type: none"> <li>– CA1: Para arquivar uma submissão, é necessário que o status da mesma esteja como “Aprovada” ou “Rejeitada”;</li> <li>– CA2: Após arquivar uma submissão, o status da mesma será de “Arquivada”, e deve ser registrada a data de arquivamento;</li> </ul>				

Tabela 2 – Requisitos Não Funcionais.

ID	Descrição	Categoria	Prioridade
<b>RNF-01</b>	O sistema deve ser compatível com os principais navegadores Web.	Portabilidade	Alta
<b>RNF-02</b>	O sistema deve garantir a consistência dos dados, exigindo o preenchimento de todos os campos obrigatórios ao realizar uma inserção.	Confiabilidade	Alta
<b>RNF-03</b>	O sistema deve ser de fácil manutenção, seguindo padrões de design e utilizando controle de versões.	Manutenibilidade	Alta
<b>RNF-04</b>	As páginas do módulo devem ter tempos de carregamento de dados inferiores a 3 segundos, garantindo rápida execução.	Eficiência de tempo	Média
<b>RNF-05</b>	O sistema deve ser intuitivo e de fácil aprendizado, não exigindo a realização de tutoriais ou treinamentos extensivos para seu uso.	Facilidade de Aprendizado	Média
<b>RNF-06</b>	A ferramenta deve ser de fácil utilização, com funcionalidades acessíveis sem a necessidade de percorrer várias páginas.	Facilidade de Operação	Alta
<b>RNF-07</b>	O sistema deve ter medidas de segurança efetivas contra os ataques mais comuns na Internet.	Segurança	Média
<b>RNF-08</b>	O sistema deve ser desenvolvido utilizando a linguagem Java e plataforma Jakarta EE.	Tecnologia	Alta

### 3 Identificação de Subsistemas

Para atender aos requisitos elencados na Seção 2 e a fim de facilitar o gerenciamento do projeto, o sistema foi dividido em 2 subsistemas: Coordenação de Curso de Graduação e Submissão de Atividades Acadêmicas.

A Figura 1 ilustra os subsistemas e suas interdependências, enquanto a Tabela 3 apresenta uma breve descrição de cada subsistema identificado.

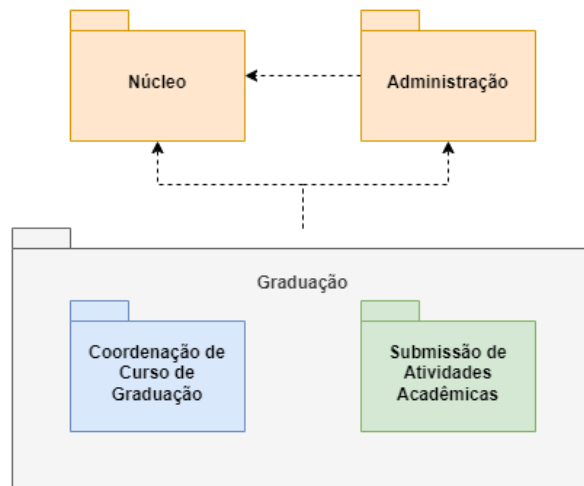


Figura 1 – Diagrama de Pacotes e os Subsistemas Identificados.

Tabela 3 – Subsistemas identificados e suas interdependências.

Subsistema	Descrição
Núcleo	Módulo responsável pelo gerenciamento do cadastro e autenticação de acadêmicos.
Administração	Módulo responsável pelo cadastro das unidades organizacionais da UFES (centros, departamentos, programas de pós-graduações, cursos).
Graduação	Módulo responsável pela gestão de cursos de graduação.
Coordenação de Curso de Graduação	Subsistema responsável por funcionalidades específicas para o Coordenador de Curso no sistema.
Submissão de Atividades Acadêmicas	Subsistema responsável pela gestão das submissões de atividades acadêmicas entre diferentes perfis de acadêmicos, com um fluxo específico.

## 4 Modelo de Casos de Uso

O modelo de casos de uso corresponde a uma tentativa de descrever a relação das funcionalidades do sistema com cada um de seus atores. Os atores identificados no contexto deste projeto estão descritos na Tabela 4.



Tabela 4 – Descrição dos atores envolvidos nos casos de uso.

Ator	Descrição
Coordenador de Curso	Acadêmico responsável por coordenar um ou mais cursos de graduação, utilizando o sistema para gerenciar os cursos sob sua supervisão.
Aluno	Acadêmico que assume o papel de aluno, utilizando o sistema para submeter atividades acadêmicas que contribuem para sua formação.
Coordenador de Horas	Acadêmico responsável por coordenar as horas complementares e de extensão de um curso de graduação, utilizando o sistema para acompanhar as submissões dessas horas feitas pelos alunos.
Coordenador de Estágio	Acadêmico responsável por coordenar os estágios de um curso de graduação, utilizando o sistema para acompanhar as submissões de estágios feitas pelos alunos.

A seguir, são apresentados os diagramas de casos de uso e a relação de cada caso de uso com as histórias de usuário apresentadas anteriormente, organizados por subsistema.

## 4.1 Subsistema Coordenação de Curso de Graduação

A Figura 2 apresenta o diagrama de casos de uso do Coordenação de Curso de Graduação.

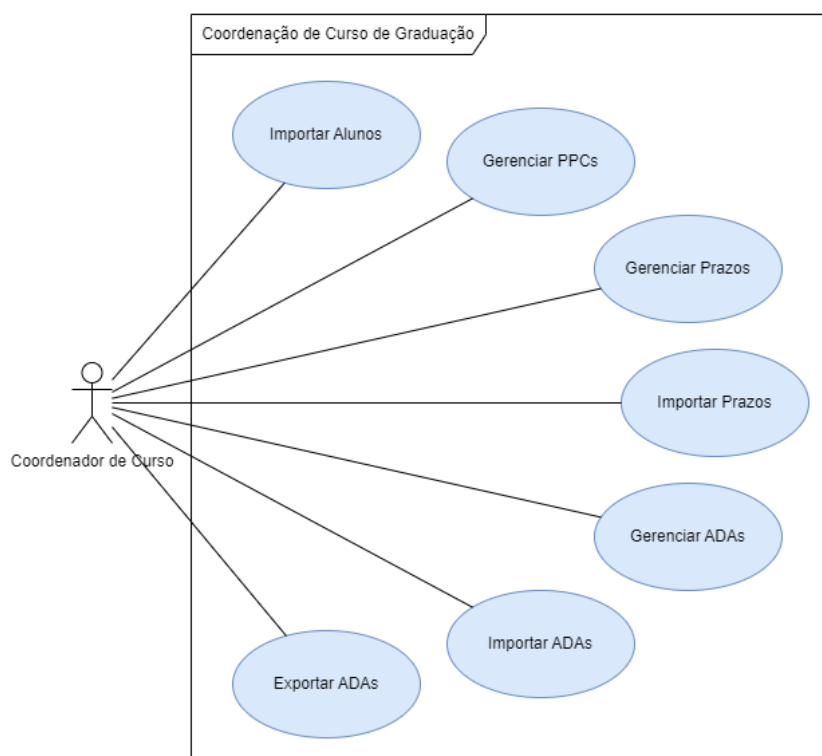


Figura 2 – Diagrama de Casos de Uso do subsistema Coordenação de Curso de Graduação.

Na Tabela 5, é possível observar a rastreabilidade dos casos de uso em relação às estórias de usuário mencionadas na Seção 2.3. Essas estórias já foram detalhadas de maneira suficiente, levando em consideração seus critérios de aceitação. Dessa forma, não há necessidade de fornecer descrições adicionais sobre as mesmas.

Tabela 5 – Rastreabilidade dos casos de uso do subsistema Coordenação de Curso de Graduação.

<b>Id</b>	<b>Nome</b>	<b>Requisitos</b>
UC-1	Importar Alunos	US-12
UC-2	Gerenciar PPCs	US-11
UC-3	Gerenciar Prazos	US-8, US-10
UC-4	Importar Prazos	US-9
UC-5	Gerenciar ADAs	US-13, US-14, US-16
UC-6	Importar ADAs	US-13
UC-7	Exportar ADAs	US-15

## 4.2 Subsistema Submissão de Atividades Acadêmicas

A Figura 3 apresenta o diagrama de casos de uso do subsistema Submissão de Atividades Acadêmicas.

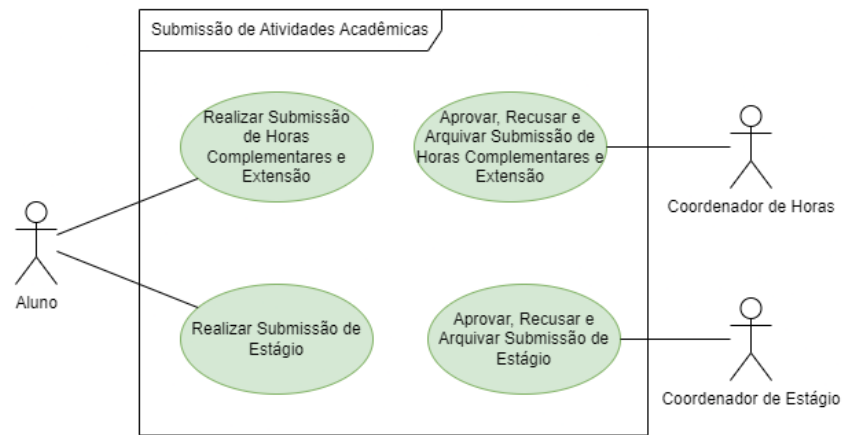


Figura 3 – Diagrama de Casos de Uso do subsistema Submissão de Atividades Acadêmicas.

Na Tabela 6, é possível verificar a rastreabilidade dos casos de uso em relação às estórias de usuário descritas na Seção 2.3. Nessa seção, as estórias de usuário foram apresentadas com detalhamento adequado, levando em conta seus critérios de aceitação. Dessa forma, não há necessidade de fornecer descrições adicionais sobre as mesmas.

Tabela 6 – Rastreabilidade dos casos de uso do subsistema Submissão de Atividades Acadêmicas.

Id	Nome	Requisitos
UC-8	Realizar Submissão de Horas Complementares e Extensão	US-17, US-18
UC-9	Aprovar, Recusar e Arquivar Submissão de Horas Complementares e Extensão	US-19, US-20, US-21
UC-10	Realizar Submissão de Estágio	US-22, US-23
UC-11	Aprovar, Recusar e Arquivar Submissão de Estágio	US-24, US-25, US-26

## 5 Modelo Estrutural

O modelo conceitual estrutural visa capturar e descrever as informações (classes, associações e atributos) que o sistema deve representar para prover as funcionalidades

descritas nos casos de uso especificados na Seção 4.

A Figura 4 apresenta o diagrama de classes do subsistema Coordenação de Curso de Graduação. A Figura 5 apresenta o diagrama de classes do subsistema Submissão de Atividades Acadêmicas.

Na sequência, a Seção 6 — Dicionário de Projeto — apresenta as descrições das classes, atributos e operações presentes nos diagramas apresentados nesta seção.

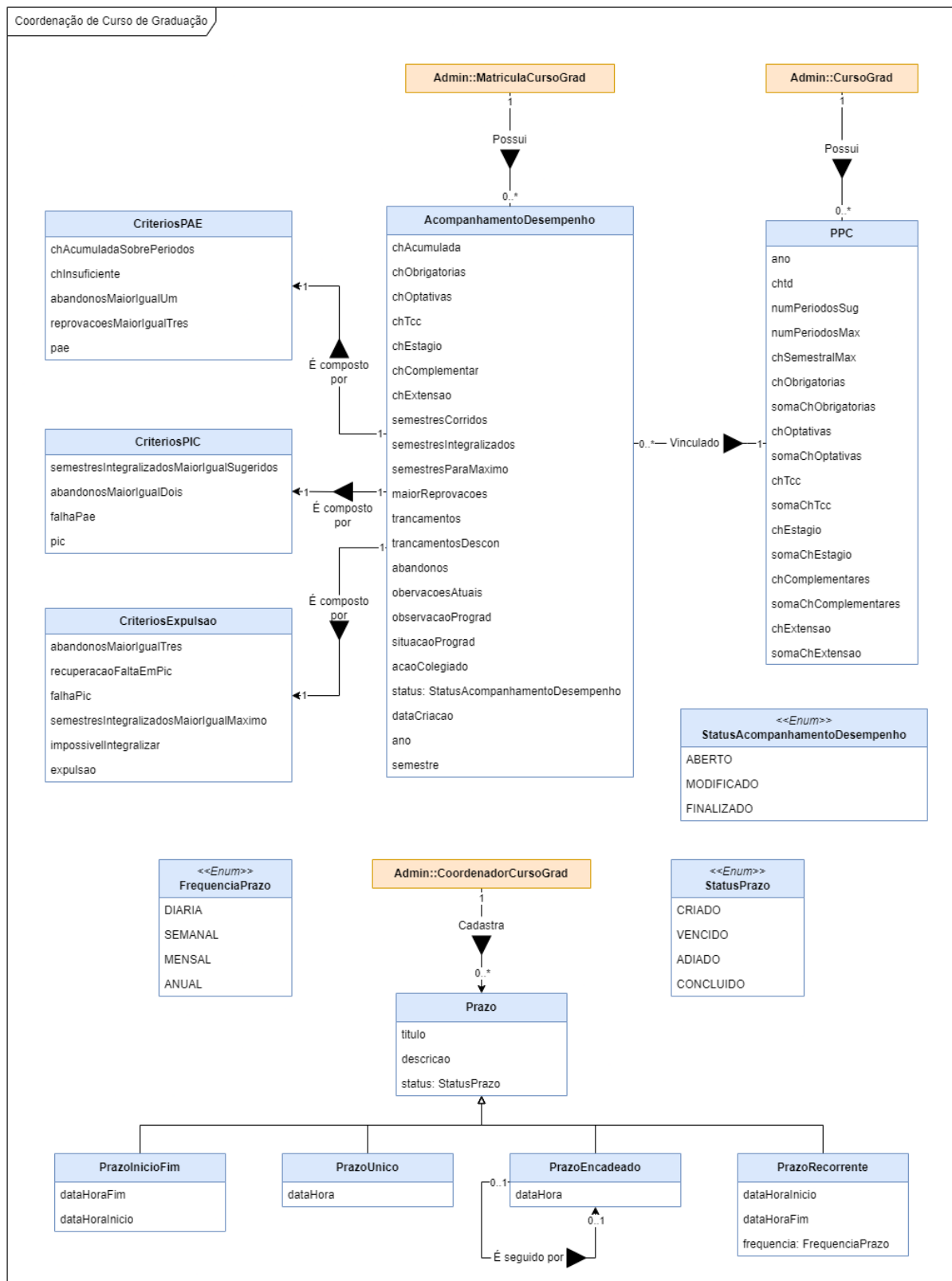


Figura 4 – Diagrama de classes do subsistema Coordenação de Curso de Graduação.

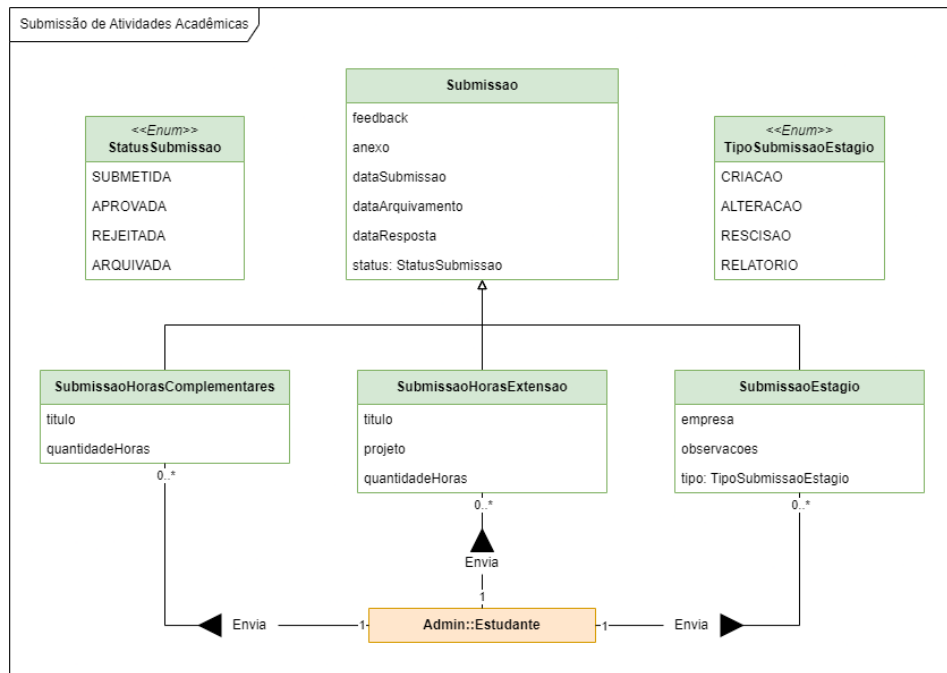


Figura 5 – Diagrama de classes do subsistema Submissão de Atividades Acadêmicas.

## 6 Dicionário de Projeto

Esta seção apresenta as definições detalhadas das classes, descrevendo seus atributos e associações e servindo como um glossário do projeto. As definições são organizadas por subsistema, cada classe sendo apresentada em uma tabela separada. A coluna “Obr.?” indica com um “x” se o atributo é obrigatório (deve possuir um valor para se criar um objeto da classe).

Vale destacar que eventuais operações que estas classes vierem a ter não são listadas e descritas nesta fase do projeto. Além disso, na Seção 5, algumas classes podem ser incluídas nos diagramas de outros subsistemas para ilustrar a relação entre eles. No dicionário de projeto, no entanto, classes são descritas apenas em seus subsistemas de origem.

### 6.1 Subsistema Coordenação de Curso de Graduação

Tabela 7 – Detalhamento da classe *AcompanhamentoDesempenho*.

Propriedade	Tipo	Obr.?	Descrição
chAcumulada	Inteiro	x	Carga horária acumulada que o aluno possui até o momento.
chObrigatorias	Inteiro	x	Carga horária em disciplinas obrigatórias que o aluno possui até o momento.
chOptativas	Inteiro	x	Carga horária em disciplinas optativas que o aluno possui até o momento.
chTec	Inteiro	x	Carga horária em disciplinas de TCC que o aluno possui até o momento.
chEstagio	Inteiro	x	Carga horária em estágios que o aluno possui até o momento.
chExtensao	Inteiro	x	Carga horária em projetos de extensão que o aluno possui até o momento.
maiorReprovacoes	Inteiro	x	Número máximo de reprovações em uma mesma matéria que o aluno possui até o momento.
trancamentos	Inteiro	x	Número de semestres que o aluno trancou até o momento.
trancamentosDescon	Inteiro	x	Número de semestres que devem ser descontados na contagem de trancamentos do aluno.
abandonos	Texto	x	Número de semestres que o aluno abandonou até o momento.
observacoesAtuais	Texto		Observações que o coordenador de curso fará sobre o ADA do aluno.
observacoesPrograd	Texto		Observações que a Prograd fez sobre o ADA do aluno.
situacaoPrograd	Texto		Situação que a Prograd definiu para o aluno.
acaoColegiado	Texto		Ação que o colegiado irá tomar após a análise do ADA do aluno.
status	Enumerado	x	Status que o ADA se encontra, indicando se está pronto ou requer análise.
dataCriacao	Data e Hora	x	Data de criação do ADA.
anoSemestre	Texto	x	Ano e semestre no padrão “2023/1”, identificando o semestre ao qual o ADA se refere.

Tabela 8 – Detalhamento da classe *PPC*.

Propriedade	Tipo	Obr.?	Descrição
ano	Inteiro	x	Ano em que o PPC foi criado.
chtd	Inteiro	x	Carga horária total acumulada necessária para a formação do aluno.
numPeriodosSug	Inteiro	x	Número de períodos sugerido para a conclusão do curso pelo aluno.
numPeriodosMax	Inteiro	x	Número máximo de períodos para a conclusão do curso pelo aluno.
chObrigatorias	Inteiro	x	Carga horária mínima de disciplinas obrigatórias exigidas para a formação no PPC.
somaChObrigatorias	Booleano	x	Indica se o valor de <i>chObrigatorias</i> deve ser somado à carga horária total ( <i>chtd</i> ).
chOptativas	Inteiro	x	Carga horária mínima de disciplinas optativas exigidas para a formação no PPC.
somaChOptativas	Booleano	x	Indica se o valor de <i>chOptativas</i> deve ser somado à carga horária total ( <i>chtd</i> ).
chTcc	Inteiro	x	Carga horária mínima de disciplinas de TCC exigidas para a formação no PPC.
somaChTcc	Booleano	x	Indica se o valor de <i>chTcc</i> deve ser somado à carga horária total ( <i>chtd</i> ).
chEstagio	Inteiro	x	Carga horária mínima de estágio exigida para a formação no PPC.
somaChEstagio	Booleano	x	Indica se o valor de <i>chEstagio</i> deve ser somado à carga horária total ( <i>chtd</i> ).
chComplementares	Inteiro	x	Carga horária mínima de horas complementares exigidas para a formação no PPC.
somaChComplementares	Booleano	x	Indica se o valor de <i>chComplementares</i> deve ser somado à carga horária total ( <i>chtd</i> ).
chExtensao	Inteiro	x	Carga horária mínima de horas de extensão exigidas para a formação no PPC.
somaChExtensao	Booleano	x	Indica se o valor de <i>chExtensao</i> deve ser somado à carga horária total ( <i>chtd</i> ).

Tabela 9 – Detalhamento da classe *Prazo*.

Propriedade	Tipo	Obr.?	Descrição
titulo	Texto	x	Título do prazo.



Propriedade	Tipo	Obr.?	Descrição
descricao	Texto		Descrição para preencher informações adicionais.
status	Enumerado	x	Informa qual o status atual do prazo.

Tabela 10 – Detalhamento da classe *PrazoInicioFim*.

Propriedade	Tipo	Obr.?	Descrição
dataHoraInicio	Data e Hora	x	Data e hora que o prazo se inicia.
dataHoraFim	Data e Hora	x	Data e hora que o prazo termina.

Tabela 11 – Detalhamento da classe *PrazoUnico*.

Propriedade	Tipo	Obr.?	Descrição
dataHora	Data e Hora	x	Data e hora para quando o prazo está programado.

Tabela 12 – Detalhamento da classe *PrazoEncadeado*.

Propriedade	Tipo	Obr.?	Descrição
dataHora	Data e Hora	x	Data e hora para quando o prazo está programado.

Tabela 13 – Detalhamento da classe *PrazoRecorrente*.

Propriedade	Tipo	Obr.?	Descrição
dataHoraInicio	Data e Hora	x	Data e hora de quando o prazo irá começar a repetir.
dataHoraFim	Data e Hora	x	Data e hora de até quando o prazo irá repetir.
frequencia	Enumerado	x	Frequência em que o prazo irá se repetir.

## 6.2 Subsistema Submissão de Atividades Acadêmicas

Tabela 14 – Detalhamento da classe *Submissao*.

Propriedade	Tipo	Obr.?	Descrição
feedback	Texto		Texto escrito por quem responde a submissão para dar alguma informação relevante.
anexo	Byte		Arquivo que pode ser anexado por quem cria a submissão, utilizado principalmente para comprovar certificação, estágio, entre outros.

Propriedade	Tipo	Obr.?	Descrição
dataSubmissao	Data e Hora	x	Data que a submissão foi criada.
dataArquivamento	Data e Hora	x	Data que a submissão foi arquivada.
dataResposta	Data e Hora	x	Data que a submissão foi respondida.
status	Enumerado	x	Status que a submissão se encontra atualmente.

Tabela 15 – Detalhamento da classe *SubmissaoHorasComplementares*.

Propriedade	Tipo	Obr.?	Descrição
titulo	Texto	x	Título da submissão.
quantidadeHoras	Inteiro	x	Quantidade de horas complementares recebidas com a atividade.

Tabela 16 – Detalhamento da classe *SubmissaoHorasExtensao*.

Propriedade	Tipo	Obr.?	Descrição
titulo	Texto	x	Título da submissão.
projeto	Texto	x	Nome do projeto de extensão que está provendo as horas de extensão.
quantidadeHoras	Inteiro	x	Quantidade de horas complementares recebidas com o projeto de extensão.

Tabela 17 – Detalhamento da classe *SubmissaoEstagio*.

Propriedade	Tipo	Obr.?	Descrição
empresa	Texto	x	Empresa que o aluno realizou o estágio.
observacoes	Texto		Observações que o aluno acha relevantes para a submissão.
tipo	Enumerado	x	Tipo da submissão, se é uma submissão de criação, alteração, rescisão ou relatório.

# Apêndices

# APÊNDICE A – Especificação do arquivo CSV para importar prazos

O arquivo CSV deve conter uma linha com o cabeçalho, e as demais linhas devem ser informações de cada um dos prazos a serem importados. O separador de colunas deve ser vírgula, e as colunas do cabeçalho devem ter os mesmos nomes e ordem da apresentada abaixo:

- **tipo**
  - Tipo do prazo
  - Campo textual
  - Opções: “UNICO”, “RECORRENTE” e “INICIO\_FIM”
- **titulo**
  - Título do prazo
  - Campo textual
- **descricao**
  - Descrição do prazo
  - Campo textual
- **data\_hora\_inicio**
  - Data e hora de início do prazo
  - Caso seja um prazo do tipo “ÚNICO”, deixar em branco
  - Campo de data-hora no formato: dd/MM/yyyy ou dd/MM/yyyy hh:mm
  - Ex: 08/03/2023, 08/03/2023 14:30
- **data\_hora\_fim**
  - Data e hora de fim do prazo
  - Caso seja um prazo do tipo “RECORRENTE”, preencher significa que o prazo vai se repetir até essa data, e deixar em branco, significa que o prazo vai se repetir para sempre
  - Campo de data-hora no formato: dd/MM/yyyy ou dd/MM/yyyy hh:mm
  - Ex: 08/03/2023, 08/03/2023 14:30

- **recorrecia**

- Recorrência de um prazo
- Caso seja um prazo do tipo “ÚNICO” ou “INICIO\_FIM”, deixar em branco
- Campo de textual
- Opções: “DIARIO”, “SEMANAL”, “MENSAL” e “ANUAL”

# APÊNDICE B – Especificação do arquivo CSV para importar alunos

O arquivo CSV deve conter uma linha com o cabeçalho, e as demais linhas devem ser informações de cada um dos alunos a serem importados. O separador de colunas deve ser vírgula, e as colunas do cabeçalho devem ter os mesmos nomes e ordem da apresentada abaixo:

- **matricula**

- Matrícula do aluno
- Campo textual

- **nome\_completo**

- Nome do aluno
- Campo textual

- **email**

- Email do aluno
- Campo textual

- **telefone**

- Telefone do aluno
- Campo textual

- **data\_inicio**

- Data que o aluno começou o curso
- Campo de data no formato: dd/MM/yyyy
- Ex: 08/03/2019

# APÊNDICE C – Especificação do arquivo CSV para computar o ADA

O arquivo CSV deve conter uma linha com o cabeçalho, e as demais linhas devem ser informações de cada um dos alunos que o ADA será computado. O separador de colunas deve ser vírgula, e as colunas do cabeçalho devem ter os mesmos nomes e ordem da apresentada abaixo:

- **matricula**
  - Matrícula do aluno
  - Campo textual
- **ppc\_ano**
  - Ano do PPC que o aluno está vinculado
  - Campo numérico
- **ch\_obrigatorias**
  - Carga horária de disciplinas obrigatórias
  - Campo numérico
- **ch\_optativas**
  - Carga horária de disciplinas optativas
  - Campo numérico
- **ch\_tcc**
  - Carga horária de Trabalho de Conclusão de Curso
  - Campo numérico
- **ch\_estagio**
  - Carga horária de estágio
  - Campo numérico
- **ch\_extensao**
  - Carga horária de projetos de extensão
  - Campo numérico

- **ch\_complementar**
  - Carga horária complementar
  - Campo numérico
- **maior\_reprovacoes**
  - Maior quantidade de reprovações em uma disciplina, não precisam ser consecutivas
  - Campo numérico
- **num\_trancamentos**
  - Número de semestres que o aluno trancou
  - Campo numérico
- **num\_trancamentos\_desconsiderados**
  - Número de semestres que o aluno trancou que devem ser desconsiderados, por exemplo, trancamento por motivo de pandemia
  - Campo numérico
- **num\_abandonos**
  - Número de semestres que o aluno abandonou, ou seja, não fez matrícula ou fez e depois cancelou todas as disciplinas
  - Campo numérico
- **falha\_pae**
  - Descumpriu ou não fez/respondeu o PAE
  - Campo numérico
  - Se falhou, preencher com 1, caso contrário, 0
- **falha\_pic**
  - Descumpriu ou não fez/respondeu o PIC
  - Campo numérico
  - Se falhou, preencher com 1, caso contrário, 0
- **rf\_pic**
  - Reprovou por falta durante PIC
  - Campo numérico



- Se teve reprovação, preencher com 1, caso contrário, 0

- **observacao\_progard**

- Observações feitas pela progard
- Campo textual
- Pode estar vazio

- **situacao\_progard**

- Situação que o aluno se encontra de acordo com a progard
- Campo textual
- Pode estar vazio

# APÊNDICE D – Resumo da Legislação sobre Acompanhamento do Desempenho Acadêmico

As próximas páginas são referentes à um documento elaborado pelo professor e atual coordenador do curso de Engenharia da Computação Eduardo Zambon, que apresenta as informações sobre Acompanhamento do Desempenho Acadêmico.

# Acompanhamento do Desempenho Acadêmico

---

## RESOLUÇÃO No. 68/2017

- Dois tipos de acompanhamento (Art. 5o.):
    - **PAE**: Plano de Acompanhamento de Estudos
    - **PIC**: Plano de Integralização Curricular
  - Critérios para aluno entrar em **PAE** (Art. 6o.):
    - Abandonou o curso por 1 semestre  
OU
    - 3 reprovações na mesma disciplina (não precisam ser consecutivas)  
OU
    - Satisfaz o critério de baixa CH concluída dada pela fórmula abaixo
$$\frac{CHA}{NSI} < \frac{2*CHTd}{NPS+NPM}$$
- Onde:
- **CHA**: Carga Horária Acumulada
    - Atualmente conta qualquer coisa que o aluno fez no curso, exceto atividade complementar
  - **NSI**: Número de Semestres Integralizados
    - Número total de semestres concluídos, não contam os semestres com trancamento (**nem 2020 inteiro**) mas contam os em abandono
  - **CHTd**: Carga Horária Total de Disciplinas do Curso
    - No Acadêmico selecionou-se as opções para excluir *Atividades Complementares e Estágio*. Valores para as grades:
      - Grade 2008: **3690** (OBR+OPT)
      - Grade 2022: **3410** (OBR+OPT+TCC+Extensão)
  - **NPS**: Número de Períodos Sugeridos
    - 10 semestres, para ambas as grades
  - **NPM**: Número de Períodos Máximo
    - 15 semestres, para ambas as grades
- Critérios para aluno entrar em **PIC** (Art. 7o.):
    - Abandonou o curso por 2 semestres (não precisam ser consecutivos)  
OU
    - Extrapolou o número de períodos sugeridos (NPS)  
OU
    - Descumpriu ou não fez/respondeu o PAE
      - Logo, se não são aplicáveis os dois primeiros pontos, precisa estar em PAE para entrar em PIC
  - Definição **abandono** (Art. 9o.):

- Não fez matrícula ou fez e depois cancelou todas as disciplinas
- Tempo de abandono conta sobre o NSI
- Procedimento do ADA (Art. 11):
  - Prograd gera relatórios e os envia ao colegiado
  - Colegiado convoca o aluno (**via Portal da Ufes**) para fazer relatório de PAE ou PIC
  - Relatórios de PAE e PIC devem ficar arquivados no colegiado
    - Também devem ser registrados no sistema Acadêmico da Ufes
- Critérios para o aluno ser **desligado** (Art. 12):
  - Abandonou o curso por 3 semestres (não precisam ser consecutivos)  
OU
  - Teve RF durante PIC  
OU
  - Descumpriu ou não fez/respondeu o PIC  
OU
  - Impossibilidade de integralização curricular dentro de NPM
    - Como determinar a impossibilidade?  
Calcular:  
**(CH não cumprida) / (no. de períodos que faltam até o NPM)**  
Se for maior que a CH máxima semestral é impossível formar => desligamento
- Recurso de desligamento I (Art. 12 - Par. 3o.):
  - Colegiado informa o estudante do desligamento
  - Aluno pode retornar com um recurso
  - Colegiado avalia o recurso
    - Se aceitar, entra em PIC
    - Se rejeitar, envia desligamento para Prograd junto com o recurso indeferido
- Recurso de desligamento II (Art. 14):
  - Aluno pode fazer recurso de desligamento para CCG (Câmara Central de Graduação)
- Recurso de desligamento III (Art. 14 - Par. 5o.):
  - Aluno pode fazer recurso de desligamento para CEPE
    - Até 30 dias após recurso indeferido pela CCG
    - **Recurso só se aplica se for comprovado que a resolução foi descumprida**
- Dilatação do prazo máximo de conclusão (Art. 15):
  - Colegiado pode dilatar o prazo máximo de conclusão para os alunos:
    - Com deficiências físicas
    - Com afecções que levem a limitações da capacidade de aprendizagem
    - Casos de força maior previstos em lei com comprovação adequada
  - Dilatação só pode ir até  $0.5 \cdot \text{NPM} = 7.5$  semestres
    - Prazo absolutamente máximo então fica em 23 semestres (11.5 anos)
  - Se teve dilatação de prazo, entra imediatamente em PIC
- Sobre o trancamento (Art. 16):

- Não pode trancar a matrícula se está em processo de desligamento
  - Idem se estiver em PIC, exceto se o colegiado autorizar
  - Ainda é permitido o trancamento por motivos de saúde previstos em lei
- 

## RESOLUÇÃO No. 40/2021

- Criou o Trancamento de Matrícula por Motivo de Pandemia (TMP)
    - Não conta na integralização curricular (NSI) (Art. 3o.)
    - TMP não conta no limite de TMA (Trancamento de Matrícula pelo estudante) ou TMJ (Trancamento de Matrícula Justificado) (Art. 4o. § 8o.)
- 

## INSTRUÇÃO NORMATIVA No 6, DE 1º DE ABRIL DE 2022

- **Art. 8o.** A Coordenação do Curso de Graduação deverá notificar o estudante, via Portal do Aluno, sobre a sua inclusão no ADA, devendo mencionar a Resolução n° 68/2017 - Cepe/Ufes, a presente Instrução Normativa (IN) e convocá-lo para preenchimento do Anexo I (PAE) ou Anexo II (PIC) da Resolução n° 68/2017 - Cepe/Ufes, conforme os prazos definidos no Calendário de Procedimentos.
  - **§ 1o.** As convocações devem ser claras no que tange à situação do estudante, devendo informar a base legal e se o enquadramento feito pelo Colegiado de Curso de Graduação é PAE ou PIC.
  - **§ 2o.** As comunicações devem preservar a identidade dos estudantes inseridos em ADA.
- **Art. 10.** As medidas pedagógicas para estudantes em PAE serão estabelecidas pelo Colegiado de Curso de Graduação.
- **Art. 12.** A aprovação do PIC pelo Colegiado de Curso fica condicionada à análise da duração sugerida e da duração máxima prevista no Projeto Pedagógico do Curso (PPC), não sendo possível ultrapassar esse prazo, exceto nos casos aprovados pela CCG ou pelo Cepe.
- **Art. 13.** Se o estudante não atender à convocação para ADA em primeira chamada, as Coordenações de Curso poderão proceder a novas notificações, dentro do prazo estabelecido pelo Calendário de Procedimentos.
  - *Parágrafo único.* O não atendimento às convocações deverá ser documentado por meio de declaração do coordenador ou registro em ata do Colegiado de Curso.
- **Art. 14.** A Coordenação de Curso deverá registrar o PAE e o PIC no Sistema Acadêmico conforme os prazos estabelecidos no Calendário de Procedimentos, podendo delegar tal função à Secretaria de Graduação.
  - *Parágrafo único.* A Coordenação de Curso deverá encaminhar, para o e-mail do estudante, uma cópia do documento registrado no portal acadêmico assinado

eletronicamente e solicitando a ciência do interessado.

- VER TÍTULOS III, IV, V, VI, VII, VIII e IX (Sobre desligamento)
- **Art. 33.** Caso o estudante possua um processo de desligamento aberto anteriormente, este documento deve ser atualizado, não devendo protocolar nova solicitação.

# APÊNDICE B – Documento de Projeto de Sistema



Documento de Projeto de Sistema

# **Módulo de otimização e centralização das atividades acadêmicas da Graduação do Marvin**

Vitória, ES

2023



## Registro de Alterações:

<b>Versão</b>	<b>Responsável</b>	<b>Data</b>	<b>Alterações</b>
1.0	Henrique Paulino Cruz	15/08/2023	Finaliza capítulos 1, 2 e 3.
1.0	Vitor Estevão Silva Souza	06/09/2023	Revisão dos capítulos 1, 2 e 3.
1.0	Henrique Paulino Cruz	08/09/2023	Correções apontadas nos capítulos 1, 2 e 3.
1.1	Henrique Paulino Cruz	26/10/2023	Finaliza capítulos 4 e 5.
1.1	Vitor Estevão Silva Souza	27/10/2023	Revisão dos capítulos 4 e 5.

# 1 Introdução

Este documento apresenta o projeto (*design*) do sistema *Módulo de otimização e centralização das atividades acadêmicas da Graduação do Marvin*. O sistema tem como objetivos principais possibilitar o preenchimento do Acompanhamento de Desempenho Acadêmico (ADA) pelos Coordenadores de Curso da UFES e facilitar a interação entre os diversos atores da comunidade acadêmica, como alunos, coordenadores e demais envolvidos, visando identificar e auxiliar os estudantes que necessitam de suporte especial.

Além desta introdução, este documento está organizado da seguinte forma: a Seção 2 apresenta a plataforma de software utilizada na implementação do sistema; a Seção 3 apresenta a especificação dos requisitos não funcionais (atributos de qualidade), definindo as táticas e o tratamento a serem dados aos atributos de qualidade considerados condutores da arquitetura; a Seção 4 apresenta a arquitetura de software; por fim, a Seção 5 apresenta o projeto dos componentes da arquitetura.

## 2 Plataforma de Desenvolvimento

Na Tabela 1 são listadas as tecnologias utilizadas no desenvolvimento da ferramenta, bem como o propósito de sua utilização.

Tabela 1 – Plataforma de Desenvolvimento e Tecnologias Utilizadas.

Tecnologia	Versão	Descrição	Propósito
Java	19	Linguagem de programação orientada a objetos e independente de plataforma.	Escrita do código-fonte das classes que compõem o sistema.
Jakarta EE Web Profile	9.1	Conjunto de especificação de APIs e tecnologias, que são implementadas por programas servidores de aplicação.	Redução da complexidade do desenvolvimento, implantação e gerenciamento de aplicações Web a partir de seus componentes de infra-estrutura prontos para o uso.
Jakarta Enterprise Beans (EJB) Lite	4.0	API para construção de componentes transacionais gerenciados por <i>container</i> .	Implementação das regras de negócio em componentes distribuídos, transacionais, seguros e portáteis.
Jakarta Persistence (JPA)	3.0	API para persistência de dados por meio de mapeamento objeto/-relacional.	Persistência dos objetos de domínio sem necessidade de escrita dos comandos SQL.

Tecnologia	Versão	Descrição	Propósito
Jakarta Contexts and Dependency Injection (CDI)	3.0	API para injeção de dependências.	Integração das diferentes camadas da arquitetura.
Jakarta Server Faces (JSF)	3.0	API para a construção de interfaces de usuários baseada em componentes para aplicações Web	Criação das páginas Web e sua comunicação com as classes Java.
Facelets	3.0	API para definição de decoradores ( <i>templates</i> ) integrada ao JSF.	Reutilização da estrutura visual comum às páginas, facilitando a manutenção do padrão visual do sistema.
PrimeFaces	12.0	Conjunto de componentes visuais JSF <i>open source</i> .	Reutilização de componentes visuais Web de alto nível.
AdminFaces	1.6.1	<i>Template</i> visual completo integrado ao Facelets/JSF e ao PrimeFaces.	Fornecimento do padrão visual do sistema.
JButler	2.1.2	<a href="#">JButler</a> é um mini-framework para facilitar o desenvolvimento de aplicações Jakarta EE.	Fornecimento de superclasses prontas para entidades (classes de domínio) persistentes, DAOs e controladores/serviços básicos de cadastro (CRUD).
MySQL Server	8.0	Sistema Gerenciador de Banco de Dados Relacional gratuito.	Armazenamento dos dados manipulados pela ferramenta.
WildFly (Preview)	26.0	Servidor de Aplicações compatível com Jakarta EE 9.	Fornecimento de implementação das APIs citadas acima e hospedagem da aplicação Web, dando acesso aos usuários via HTTP.

Na Tabela 2 vemos os softwares que apoiaram o desenvolvimento de documentos e também do código fonte.

Tabela 2 – Softwares de Apoio ao Desenvolvimento do Projeto

Tecnologia	Versão	Descrição	Propósito
draw.io	21.6.6	Aplicativo para criação de diagramas	Criação dos modelos de Entidades, Aplicação, Persistência e Navegação.
Git	2.25.1	Sistema de controle de versões	Registro de histórico de alterações no repositório.
TeXstudio	4.5.1	Editor de LaTeX.	Escrita da documentação do sistema, sendo usado o <i>template abnTeX</i> . <sup>1</sup>
IntelliJ IDEA	2023.1.5	Ambiente de desenvolvimento (IDE) com suporte ao desenvolvimento Java EE.	Implementação, implantação e testes da aplicação Web Java EE.
Apache Maven	3.8	Ferramenta de gerência/construção de projetos de software.	Obtenção e integração das dependências do projeto.

<sup>1</sup> <<http://www.abntex.net.br>>.

Tecnologia	Versão	Descrição	Propósito
DBeaver	23.1.1	Ferramenta para gerir bases de dados.	Visualização e edição dos dados e criação de queries SQL.
GitLab	14.4	Plataforma DevOps para gestão de projetos de software.	Controle de versão do código-fonte, gestão do time de desenvolvimento, implantação contínua.

### 3 Requisitos Não Funcionais

A Tabela 3 apresenta a especificação dos requisitos não funcionais identificados no Documento de Especificação de Requisitos, os quais foram considerados condutores da arquitetura.

Para efeito de validação, considera-se que as medidas indicadas na tabela serão verificadas de acordo com seus critérios de aceitação durante um período de homologação do sistema junto aos seus principais *stakeholders*, após o qual será considerado entregue e em atendimento aos seus RNFs principais.

Tabela 3 – Especificação de Requisitos Não Funcionais.

RNF-1 – O sistema deve ser compatível com os principais navegadores Web.	
Categoria:	Portabilidade
Tática / Tratamento:	Implementação de códigos responsivos e utilização de bibliotecas suportadas pelos principais navegadores.
Medida:	Garantir que o sistema seja acessível nos principais navegadores, como Google Chrome, Mozilla Firefox e Microsoft Edge, sem distorções no layout.
Critério de Aceitação:	Realizar testes nos navegadores mencionados para verificar se não ocorrem alterações indesejadas no layout ou comportamento do sistema.

RNF-2 – O sistema deve garantir a consistência dos dados, exigindo o preenchimento de todos os campos obrigatórios ao realizar uma inserção.	
Categoria:	Confiabilidade
Tática / Tratamento:	Implementação de validações em todas as interações do sistema que possam resultar em inconsistências nos registros, abrangendo campos obrigatórios e restrições de integridade.
Medida:	Elaboração de testes automatizados de <i>software</i> .
Critério de Aceitação:	Alcance de cobertura de testes superior a 80%. Os usuários devem ser capazes de utilizar o sistema sem a ocorrência de erros inesperados devido a dados inválidos. Deve ser fornecido feedback ao usuário sobre as restrições por meio de mensagens de validação e realce dos campos obrigatórios.

RNF-3 – O sistema deve ser de fácil manutenção, seguindo padrões de design e utilizando controle de versões.	
Categoria:	Manutenibilidade
Tática / Tratamento:	Implementação de práticas baseadas em <i>Design Patterns</i> e incorporação de ferramentas de controle de versão, como o Git.
Medida:	Estratégias como revisões de código devem ser empregadas para garantir a qualidade e uniformidade do código.
Critério de Aceitação:	Qualquer novo código deve ser revisado e aprovado por pelo menos um revisor, antes de ser incorporado na <i>branch</i> principal.

RNF-4 – As páginas do módulo devem ter tempos de carregamento de dados inferiores a 3 segundos, garantindo rápida execução.	
Categoria:	Eficiência
Tática / Tratamento:	Adoção de práticas de codificação simples, normalização em estruturas de tabelas e aplicação de <i>Design Patterns</i> e estruturas de dados adequadas para cada contexto.
Medida:	Conduzir análises de desempenho do sistema através de ferramentas apropriadas que gerem relatórios detalhados sobre cada operação.
Critério de Aceitação:	Com base na análise de desempenho obtida, o tempo de carregamento para cada ação realizada pelo usuário deve ser inferior a 3 segundos.

RNF-5 – O sistema deve ser intuitivo e de fácil aprendizado, não exigindo a realização de tutoriais ou treinamentos extensivos para seu uso.	
Categoria:	Usabilidade
Tática / Tratamento:	A interface deve adotar padrões de <i>design</i> que enfatizem a usabilidade e a experiência do usuário.
Medida:	Avaliar junto aos usuários finais se o sistema se mostrou intuitivo e de fácil utilização, sem a necessidade de orientações.
Critério de Aceitação:	O tempo necessário para que usuários novatos executem tarefas não deve exceder, em média, o dobro do tempo empregado por usuários familiarizados.

RNF-6 – A ferramenta deve ser de fácil utilização, com funcionalidades acessíveis sem a necessidade de percorrer várias páginas.	
Categoria:	Usabilidade
Tática / Tratamento:	O desenvolvimento do sistema deve se pautar em conceitos de <i>User Experience</i> (UX), aproveitando ao máximo a gama de componentes disponíveis. Deve-se modularizar de maneira estratégica cada conjunto de recursos do sistema, buscando centralizar as funcionalidades.
Medida:	Calcular a média de páginas distintas que um usuário precisa percorrer para atingir seus objetivos.
Critério de Aceitação:	A média de páginas diferentes que um usuário precisa navegar para concluir uma operação específica deve ser inferior a 5.

RNF-7 – O sistema deve ter medidas de segurança efetivas contra os ataques mais comuns na Internet.	
Categoria:	Segurança

Tática / Tratamento:	Utilizar um <i>framework</i> de autenticação e autorização moderno, aderindo às melhores práticas, como a utilização de criptografia robusta para senhas. Um tempo considerável será destinado ao estudo aprofundado desse <i>framework</i> para sua configuração e implementação adequadas.
Medida:	Caso seja possível, reunir membros da comunidade acadêmica com experiência em segurança para tentar obter acesso não autorizado ao sistema. E durante a utilização dos usuários finais, observar se os mesmos possuem acesso às funções do sistema relativas aos seus perfis.
Critério de Aceitação:	Não deve ocorrer acesso não autorizado durante os testes de segurança. Durante a observação do uso pelos usuários finais, todos devem ter acesso somente às funções apropriadas aos seus perfis.

## 4 Arquitetura de Software

No processo de desenvolvimento do *Módulo de otimização e centralização das atividades acadêmicas da Graduação do Marvin*, empregou-se uma arquitetura fundamentada no padrão arquitetônico Camada de Serviço (*Service Layer*) conforme descrito por Fowler (2002). Isso foi realizado com o auxílio de *frameworks* específicos. A Figura 1 ilustra a arquitetura do sistema, identificando os componentes presentes em cada pacote, além de destacar o papel das tecnologias Jakarta EE envolvidas.

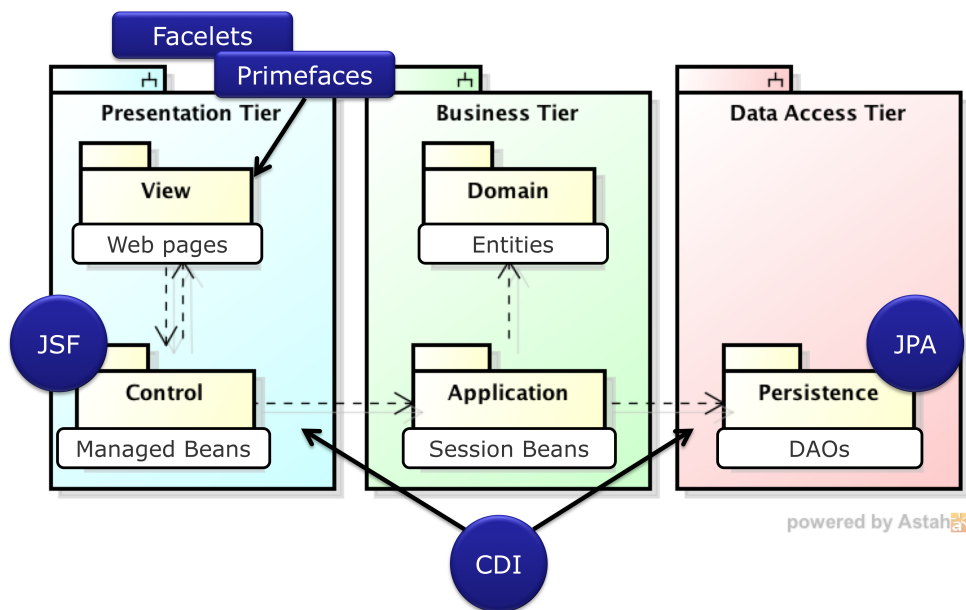


Figura 1 – Arquitetura de Software (SOUZA, 2020).

O sistema Marvin tem como objetivo centralizar uma variedade de funcionalidades em um único local. Cada conjunto de funcionalidades é organizado em módulos distintos, e

cada um desses módulos adota a mesma arquitetura ilustrada na Figura 1, que se encontra subdividida em camadas.

- Camada de Apresentação (*Presentation Tier*): como o nome sugere, esta camada trata da criação da interface com a qual o usuário interage. Nela, encontramos dois componentes provenientes da arquitetura MVC: a visão (*View*), que abriga os elementos de interface, como páginas, *scripts* e *layouts*; e o controle (*Control*), que inclui as classes que gerenciam as solicitações da camada de visualização e também lida com a comunicação com o componente de aplicação (*Application*), pertencente à camada de negócios;
- Camada de Negócio (*Business Tier*): essa camada se divide em duas partes — a lógica de aplicação (*Application*) e a lógica de domínio (*Domain*). Aqui é onde as funcionalidades são oferecidas, seguindo as regras de negócios do sistema;
- Camada de Acesso a Dados (*Data Access Tier*): aqui ocorrem as operações relacionadas à persistência de informações no banco de dados. Esta camada contém apenas o componente de Persistência (*Persistence*) e é responsável por tarefas como armazenamento de arquivos e acesso a bancos de dados.

Como resultado da arquitetura proposta para o módulo, a Figura 2 ilustra a estrutura de pastas e pacotes criados em uma interface de desenvolvimento. Os nomes seguem um padrão que facilita a identificação da camada de aplicação à qual cada pacote pertence. Por exemplo, os pacotes `controller`, `application` e `persistence` correspondem, respectivamente, aos componentes homônimos pertencentes às camadas de Apresentação, Negócio e Acesso a Dados, respectivamente.

É importante salientar que outros pacotes auxiliares podem ser criados, como é o caso dos pacotes `adapters` e `exceptions`. Esses pacotes não estão necessariamente vinculados a nenhuma das camadas mencionadas anteriormente; eles contêm classes e interfaces que auxiliam no desenvolvimento do módulo como um todo.

Por fim, vale ressaltar a existência da pasta `webapp`. Nesta pasta, ficam os arquivos `.xhtml`, que também fazem parte da Camada de Apresentação (componente de visão). Esses arquivos descrevem o que é renderizado na tela do usuário da aplicação. Eles são mantidos fora da estrutura de pacotes Java, seguindo uma padronização do *framework* utilizado para o desenvolvimento.

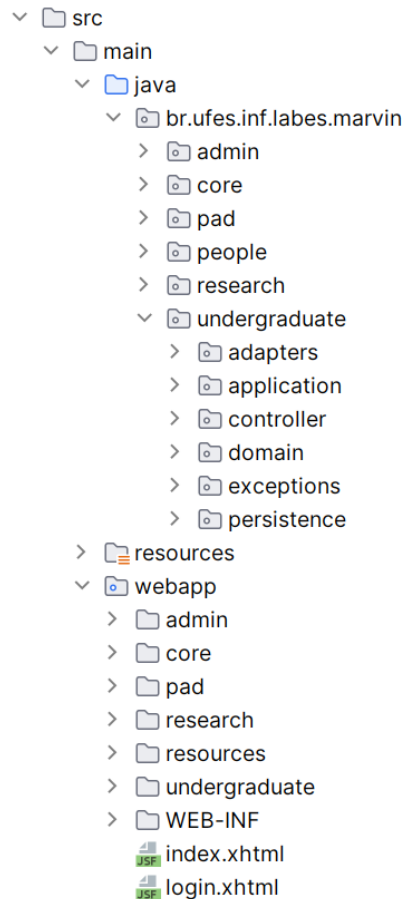


Figura 2 – Organização de pastas e pacotes do projeto.

## 5 Projeto dos Componentes da Arquitetura

Conforme mostrado na Figura 1 os principais subsistemas deste sistema estão organizados em 3 camadas: Camada de Negócio, Camada de Acesso a Dados e Camada de Apresentação, as quais são descritas em detalhes nesta seção.

### 5.1 Camada de Negócio

Dentro da estrutura do sistema *Módulo de otimização e centralização das atividades acadêmicas da Graduação do Marvin*, encontramos a incorporação fundamental do utilitário JButler. Especificamente, no Componente de Domínio do Problema (CDP), notamos que todas as classes localizadas no pacote `domain` herdaram as funcionalidades e recursos da classe `PersistentObjectSupport` do JButler. Para manter a clareza nos diagramas, decidimos não representar essa herança de forma explícita, uma vez que a quantidade considerável de associações poderia tornar a visualização do modelo excessivamente complexa.

A classe `PersistentObjectSupport` desempenha um papel essencial, oferecendo atributos e métodos valiosos compartilhados por todas as classes persistentes do sistema.



Alguns desses elementos foram modularizados em uma superclasse distinta chamada `DomainObjectSupport`. Ambas as classes definem interfaces que estabelecem contratos específicos, permitindo que o `JButler` seja reutilizado de maneira eficaz, mesmo por desenvolvedores que optem por não empregar herança.

Vale destacar que o módulo em análise mantém uma relação direta com o módulo `admin`. Essa relação pode ser observada nos relacionamentos entre as classes nos diagramas. Em resumo, as entidades essenciais abrangem inicialmente `UndergraduateDegree`, `UndergraduateDegreeEnrollment`, e `UndergraduateDegreeCoordination`, como ilustrado nas figuras 3 e 4.

Por fim, na Seção 5.3, investigaremos o Componente de Gerenciamento de Tarefas, presente no pacote de aplicação (`application`). Além disso, analisaremos os componentes da Camada de Interface com o Usuário.

### 5.1.1 Subsistema Coordenação de Curso de Graduação

Ao observar o subsistema Coordenação de Curso de Graduação, a Figura 3 destaca o projeto dessa camada, desempenhando um papel vital no gerenciamento dos Projetos Pedagógicos dos Cursos (PPC), dos prazos para o coordenador de curso e no processamento dos Acompanhamentos de Desempenho Acadêmico (ADA).

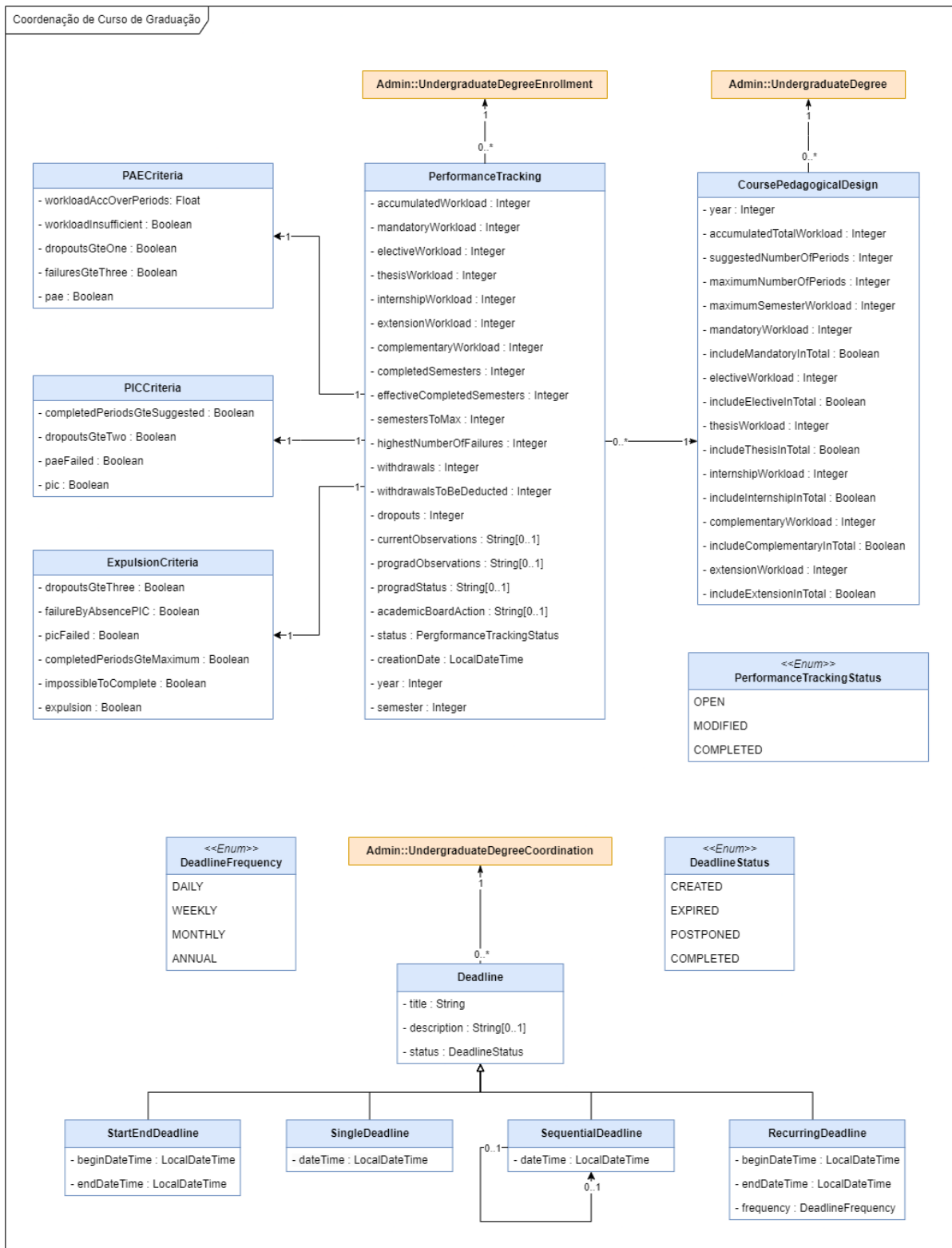


Figura 3 – Projeto da Componente de Domínio do Problema (domain) do subsistema Coordenação de Curso de Graduação.

### 5.1.2 Subsistema Submissão de Atividades Acadêmicas

Quanto ao subsistema de Submissão de Atividades Acadêmicas, a Figura 4 apresenta o projeto dessa camada, desempenhando um papel crítico na gestão e processamento das

atividades acadêmicas submetidas pelos usuários.

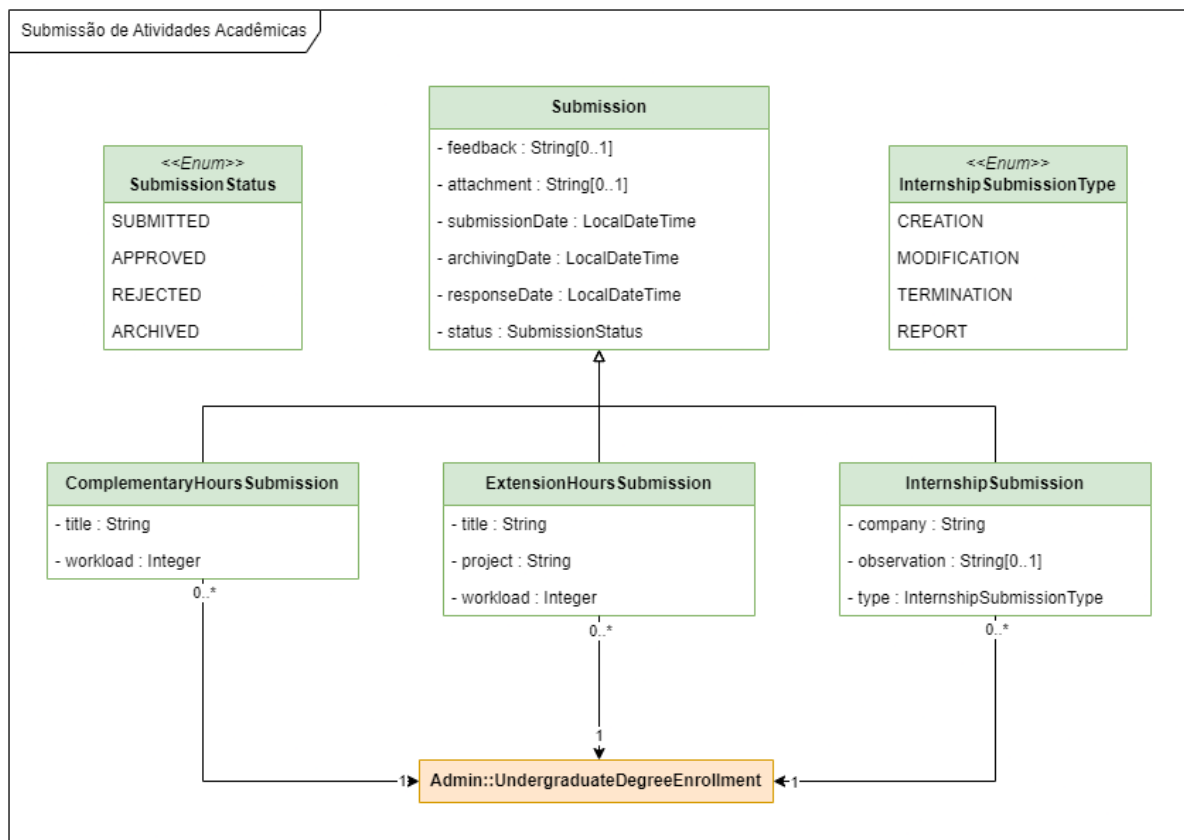


Figura 4 – Projeto da Componente de Domínio do Problema (domain) do subsistema Submissão de Atividades Acadêmicas.

## 5.2 Camada de Acesso a Dados

Na Camada de Acesso a Dados, estamos nos referindo ao Componente de Gerência de Dados (CDG) e ao pacote `persistence` do sistema. O desenvolvimento dessa camada faz amplo uso do utilitário JButler, que oferece diversas ferramentas para facilitar o gerenciamento das entidades persistidas no banco de dados. Conforme detalhado na Seção 4, adotamos o padrão *Data Access Object* (DAO). Esse padrão envolve a criação de uma interface com o sufixo “DAO” e uma classe que a implementará, com o sufixo “JPADAO”.

A Figura 5 apresenta a interface e classe base do JButler, que servirão de base para todas as outras interfaces e classes neste módulo. Como é possível observar, essa abordagem oferece uma vantagem significativa para os desenvolvedores, pois ela já implementa as funcionalidades essenciais relacionadas à camada de persistência. Dessa forma, eles podem se concentrar nos detalhes específicos do domínio.

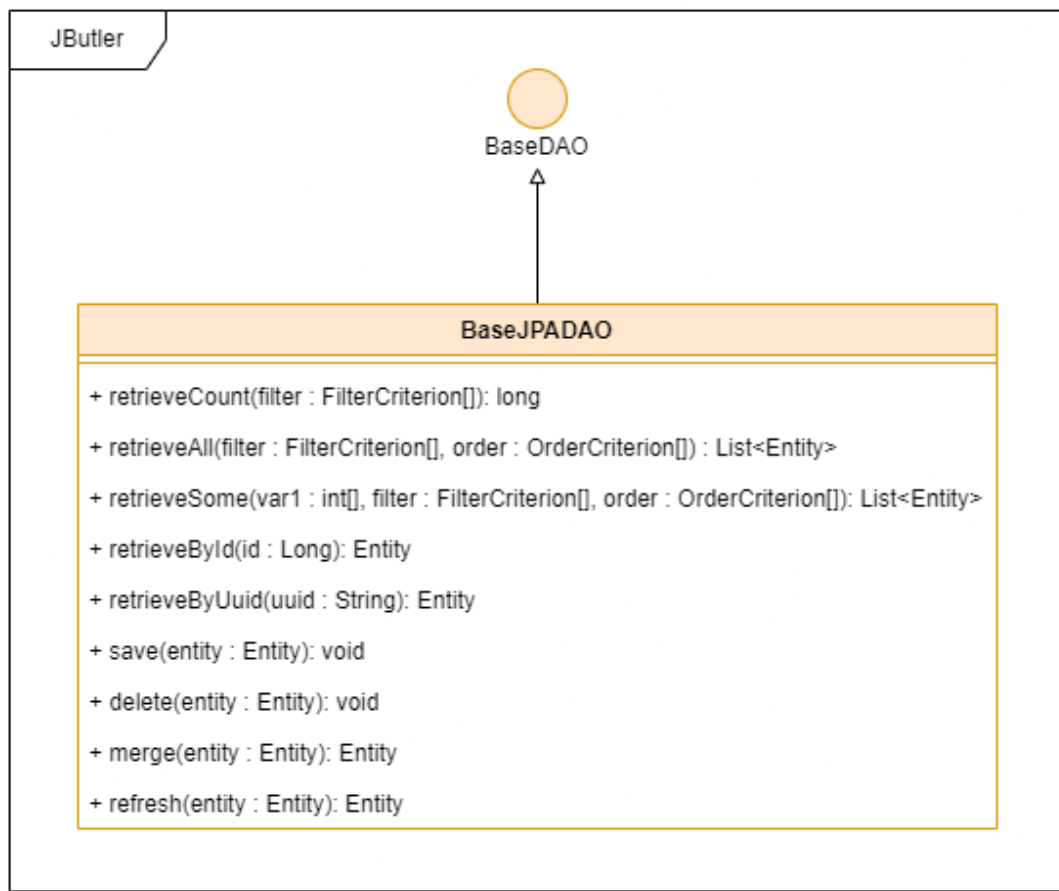


Figura 5 – Modelo de persistência base do utilitário JBButler.

### 5.2.1 Subsistema Coordenação de Curso de Graduação

No subsistema de Coordenação de Curso de Graduação, o projeto da CGD é representado na Figura 6. Um ponto importante a destacar é que a interface **DeadlineDAO** cuida de todas as operações relacionadas a prazos. Não foi necessário criar uma interface para cada uma das especializações de prazos que o subsistema poderia ter.

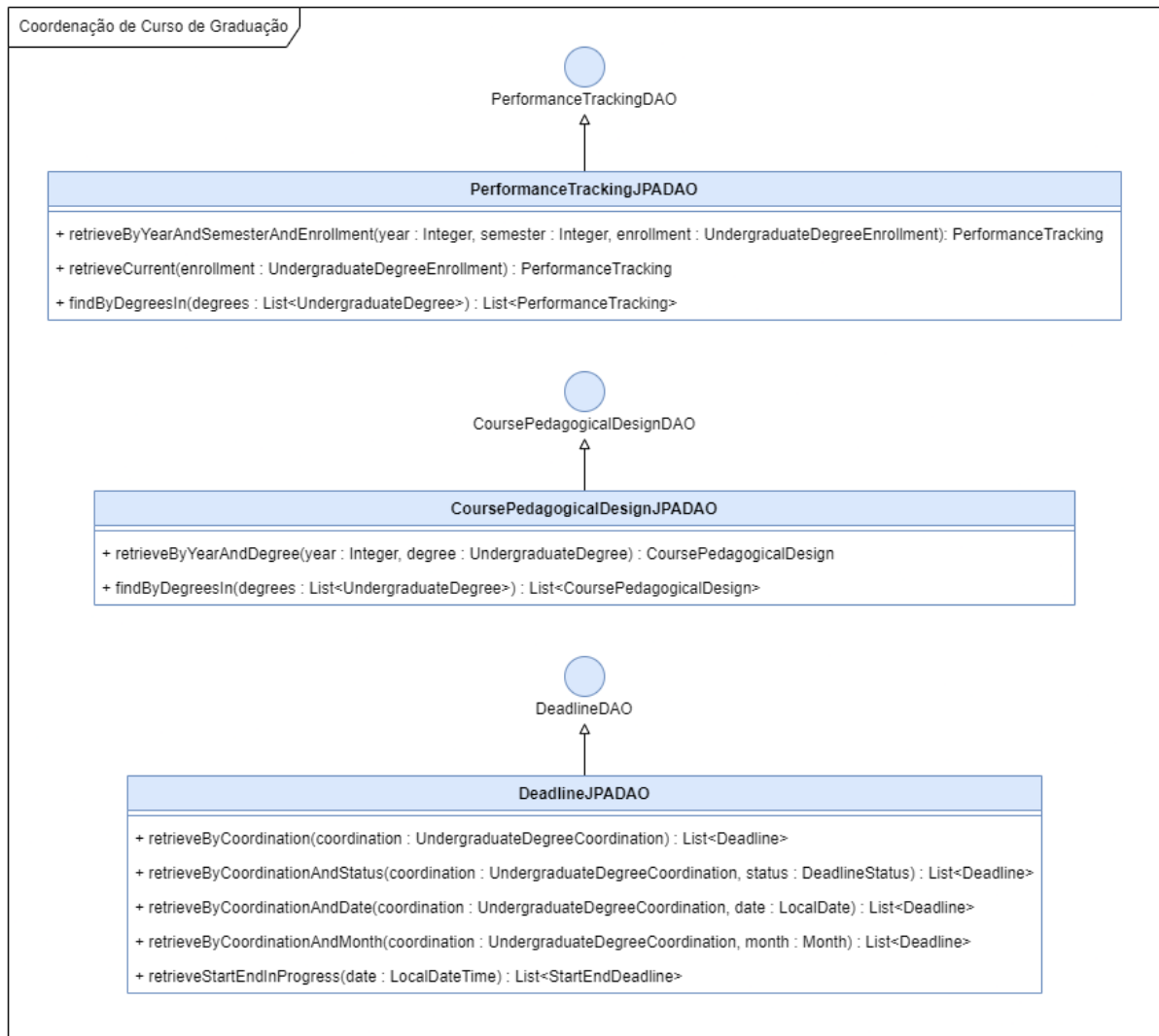


Figura 6 – Projeto da Componente de Gerência de Dados (*persistence*) do subsistema Coordenação de Curso de Graduação.

### 5.2.2 Subsistema Submissão de Atividades Acadêmicas

No subsistema de Submissão de Atividades Acadêmicas, a Figura 7 apresenta o projeto da CGD. Aqui, não há nenhuma peculiaridade que já não tenha sido mencionada anteriormente, e o modelo é claro e de fácil compreensão.

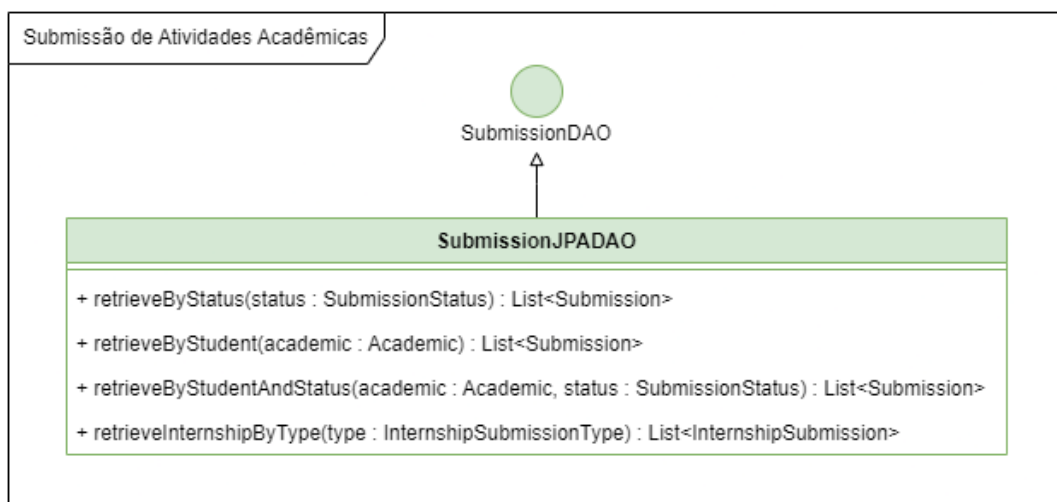


Figura 7 – Projeto da Componente de Gerência de Dados (*persistence*) do subsistema Submissão de Atividades Acadêmicas.

### 5.3 Camada de Apresentação

As figuras 8, 9, e 10 retratam o projeto do Componente de Interação Humana, composto pela pasta `webapp`, e do Componente de Controle de Interação, representado pelo pacote `controller` do módulo. Por conta de sua dependência com o Componente de Gerência de Tarefas, presente no pacote `application`, essa relação é também destacada nas imagens. Importante notar que o projeto do Subsistema Coordenação de Curso de Graduação foi dividido em duas partes para melhor visualização.

Os modelos apresentados seguem padrões específicos de representação. Os elementos que possuem o sufixo “*Controller*” representam *beans* do JSF, enquanto aqueles que terminam em “.*xhtml*” representam páginas da Web renderizadas no navegador do cliente. Nesse contexto de arquitetura MVC, os controladores são a representação do *Controller* e residem no pacote `controller`, enquanto os arquivos Web compõem a *View* e estão localizados na pasta `webapp`. Juntos, formam o *front-end* da aplicação, permitindo que os usuários interajam de maneira eficaz com o sistema.

Por outro lado, as classes com o estereótipo “*interface*” são interfaces Java que representam a porta de entrada do componente *Model* no padrão MVC e são parte integrante do pacote `application`. Nesse mesmo pacote, também existem implementações para essas interfaces, embora não estejam explícitas nos diagramas. Essas interfaces trabalham em conjunto com a Componente de Domínio do Problema, que se encontra no pacote `domain`, e a Componente de Gerência de Dados, presente no pacote `persistence`, para formar o *back-end* da aplicação.

Uma característica essencial é a independência entre o *front-end* e o *back-end*. Os controladores estabelecem associações unidirecionais com as interfaces de serviço, as quais,

conforme a arquitetura, não têm conhecimento prévio dos controladores. Isso mantém o *back-end* desacoplado do *front-end*. As relações bidirecionais entre controladores e páginas da Web representam o *binding* realizado pelo JSF entre as *tags* nas páginas XHTML e os atributos das classes controladoras, permitindo que os dados fluam em ambas as direções. Por fim, as associações unidirecionais entre as páginas da Web representam links entre elas, os quais, de maneira semelhante ao redirecionamento, iniciam uma nova solicitação pela página de destino.

Não são especificadas cardinalidades nas associações entre controladores e serviços, exceto na extremidade navegável, que reside no lado do serviço. Outras associações não incluem cardinalidades, visto que não envolvem classes, mas páginas da Web, tornando redundante a especificação de cardinalidades nessas relações.

Finalmente, é importante notar que em todos os modelos existe uma página da web chamada `index.xhtml`. Essa página serve como ponto de entrada no sistema para o usuário, direcionando-o para diversos módulos e outras páginas disponíveis. A partir de `index.xhtml`, o usuário começa a explorar o sistema em busca de suas necessidades.

### 5.3.1 Subsistema Coordenação de Curso de Graduação

A Figura 8 representa a interface da parte do subsistema destinada ao gerenciamento de PPCs, ADAs, além da funcionalidade de importar alunos e ADAs. Nas telas de gerenciamento, identificadas pelo prefixo “*manage*”, os fluxos são notavelmente simplificados, permitindo que cada operação seja executada de maneira conveniente em um único arquivo XHTML.

As telas de importação, identificadas pelo prefixo “*import*”, representam fluxos mais complexos, nos quais várias telas guiadas são usadas para concluir uma ação. Vale a pena observar que nesses casos, diversos arquivos XHTML usam o mesmo controlador, mantendo o mesmo escopo para o usuário ao navegar por diferentes páginas.

A representação de prazos, que também faz parte deste subsistema, é controlada por meio do modelo apresentado na Figura 9. O fluxo é semelhante ao já mostrado, e é importante mencionar os componentes `CalendarDeadlinesController` e `deadlines/index.xhtml`, que são responsáveis por uma tela mais voltada para a visualização dos prazos do coordenador de curso.

Além disso, vale destacar que, embora o modelo da Camada de Negócios (Figura 3) inclua três especializações para a entidade *Deadline*, no contexto da interface, todas são tratadas como uma classe generalizada *Deadline*. Essa abordagem é adotada devido à semelhança significativa em termos de atributos e funcionalidades entre todas as especializações. Dessa forma, é mais eficiente ter uma única tela para lidar com todas elas, mantendo apenas algumas especificidades para atender a cada caso.

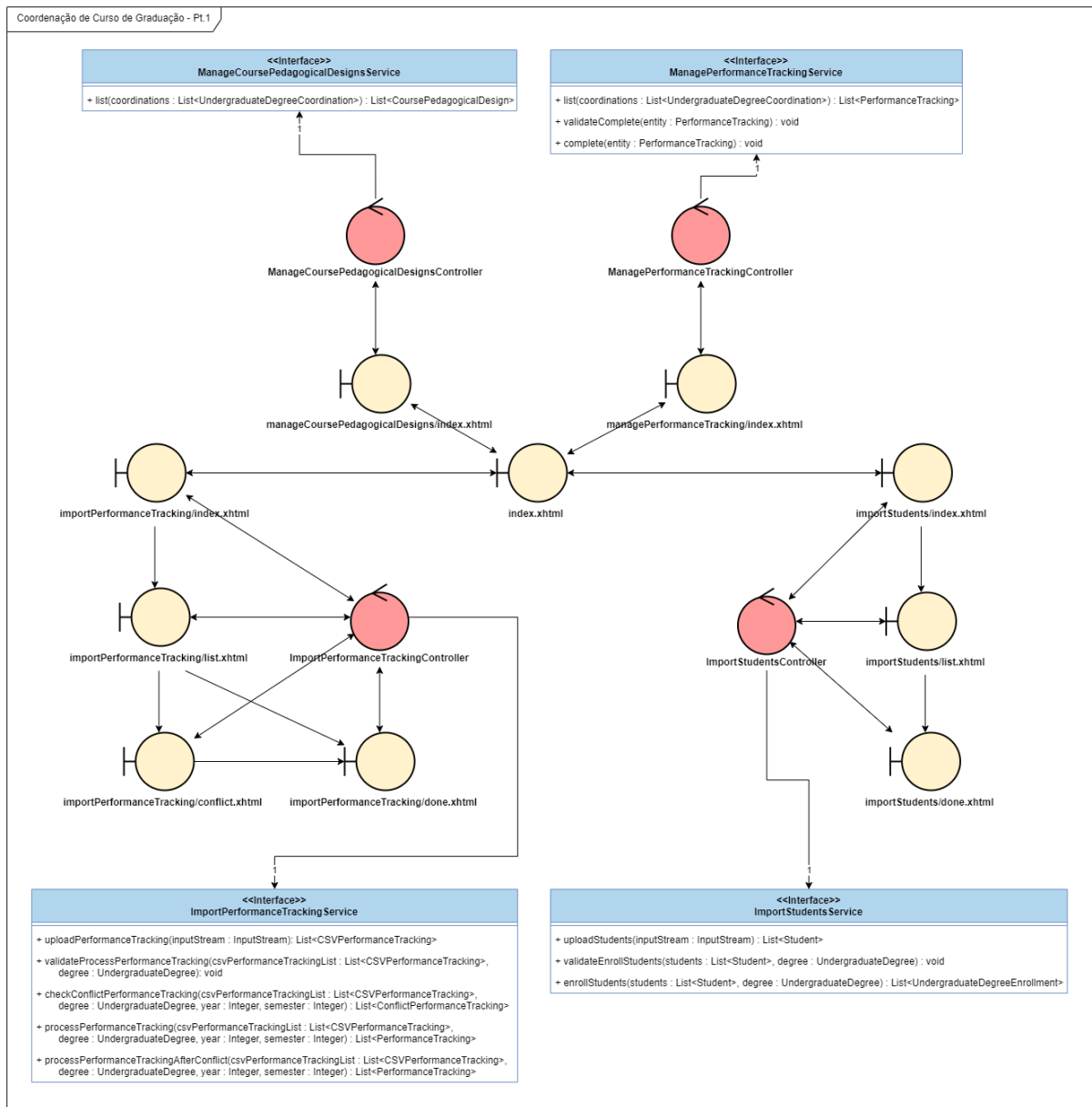


Figura 8 – Projeto da Camada de Interface com o Usuário (view e controller) do subsistema Coordenação de Curso de Graduação, juntamente com a Componente de Gerência de Tarefas (application). - Parte 1.

### 5.3.2 Subsistema Submissão de Atividades Acadêmicas

Concluindo a Camada de Apresentação, a Figura 10 descreve o modelo do subsistema de Submissão de Atividades Acadêmicas. A principal distinção em relação aos modelos anteriores (Seção 5.3.1) é a presença de uma tela intermediária que organiza outras antes do usuário acessar as páginas nas quais ele pode realizar operações. Essa tela é `submissions/index.xhtml`, adotada por questões de organização para melhorar a experiência do usuário no sistema.



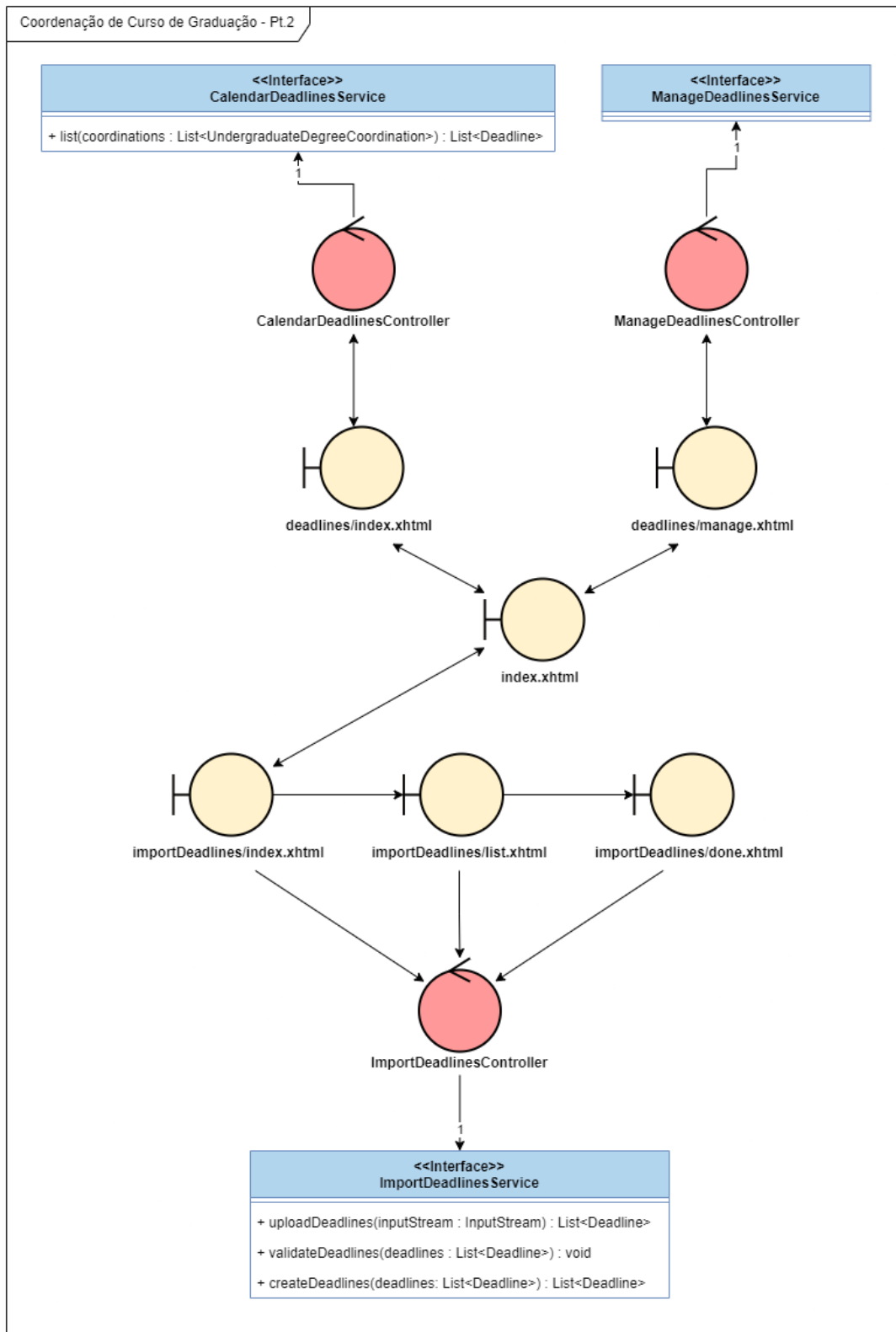


Figura 9 – Projeto da Camada de Interface com o Usuário (view e controller) do subsistema Coordenação de Curso de Graduação, juntamente com a Camada de Gerência de Tarefas (application). - Parte 2.

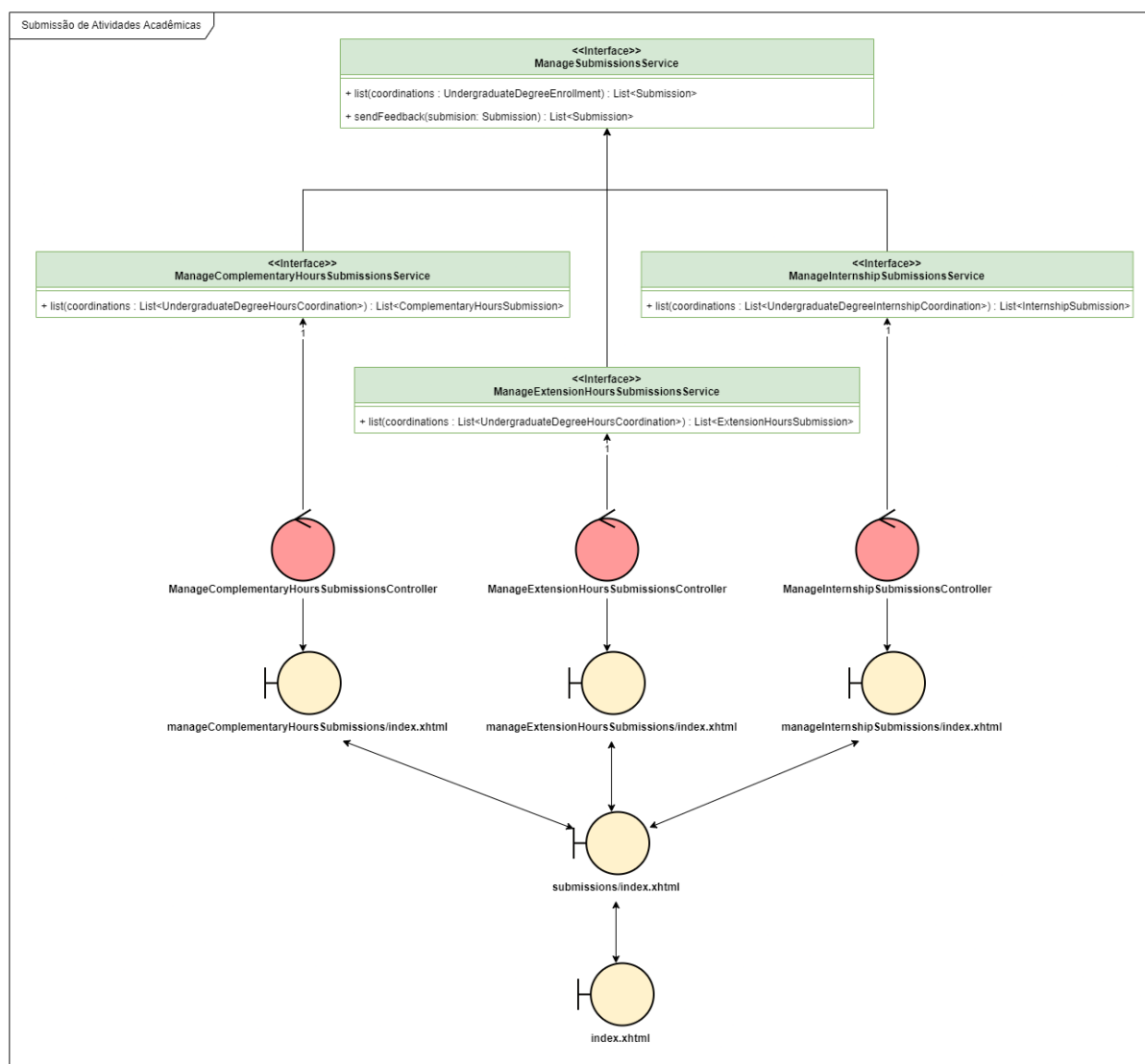


Figura 10 – Projeto da Camada de Interface com o Usuário (view e controller) do subsistema Submissão de Atividades Acadêmicas, juntamente com a Camada de Gerência de Tarefas (application).

# Referências

FOWLER, M. *Patterns of Enterprise Application Architecture*. 1. ed. [S.l.]: Addison-Wesley, 2002. ISBN 9780321127426. Citado na página 6.

SOUZA, V. E. S. The FrameWeb Approach to Web Engineering: Past, Present and Future. In: ALMEIDA, J. P. A.; GUIZZARDI, G. (Ed.). *Engineering Ontologies and Ontologies for Engineering*. 1. ed. Vitória, ES, Brazil: NEMO, 2020. cap. 8, p. 100–124. ISBN 9781393963035. Disponível em: <<http://purl.org/nemo/celebratingfalbo>>. Citado na página 6.