



Thiago Rocha Salvatore

AlocaWeb e BibLattes - Módulos do sistema Marvin

Vitória, ES

2016

Thiago Rocha Salvatore

AlocaWeb e BibLattes - Módulos do sistema Marvin

Monografia apresentada ao Curso de Ciência da Computação do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Departamento de Informática

Orientador: Prof. Dr. Vítor E. Silva Souza

Vitória, ES

2016

Thiago Rocha Salvatore

AlocaWeb e BibLattes - Módulos do sistema Marvin

Monografia apresentada ao Curso de Ciência da Computação do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Trabalho aprovado. Vitória, ES, 31 de outubro de 2016:

Prof. Dr. Vítor E. Silva Souza
Orientador

Prof. Dr. Ricardo de Almeida Falbo
Universidade Federal do Espírito Santo

Bruno Borlini Duarte, MSc.
Universidade Federal do Espírito Santo

Vitória, ES
2016

Agradecimentos

Em aproximadamente 5 anos de curso, diversas pessoas passaram pela minha vida e fizeram a diferença, seja dentro da universidade ou até mesmo na minha vida pessoal.

Gostaria de iniciar os agradecimentos citando as duas pessoas que mais me deram forças e me ajudaram durante este período longe de casa. Meu pai Paulo e minha mãe Maria Luisa - Que estão agora em Porto Seguro - Bahia. Apenas um obrigado não seria suficiente para agradecer o carinho que vocês sempre me deram.

Agradeço a todos os amigos que fiz dentro da UFES (que não foram poucos) - Thiago, Passamani, Calmon, Igor, Murillo, Cadu, Félix, Bibi entre outros. Essa faculdade não teria sido a mesma coisa se não tivesse os conhecido.

Ao meu orientador, Vitor e a grande maioria dos professores do Departamento de Informática da UFES, por estarem sempre dispostos a ajudar, puxar a orelha quando necessário e por disponibilizar seu tempo me ajudando neste trabalho.

Por último e não menos importante queria agradecer uma pessoa que esteve comigo nos últimos meses, que me aguentou reclamando de estudar, e que me deu forças, minha namorada Lívia.

É claro que não posso esquecer de agradecer a Deus, por me ajudar quando eu estava sozinho longe de casa pela primeira vez por me ouvir quando eu precisei.

“Julgue seu sucesso pelas coisas que você teve que renunciar para conseguir.”

(Dalai Lama)

Resumo

O Departamento de Informática da Universidade Federal do Espírito Santo (DI/Ufes) precisa de sistemas capazes de apoiar certos processos internos (relacionados ao contexto acadêmico), a saber: Alocação de Salas de Aula e Controle de Publicações. O propósito do primeiro é facilitar e otimizar o processo de alocação de salas de aula a turmas, evitando conflitos de horários, por exemplo. Já o segundo tem o propósito de facilitar a extração e importação de informações de um currículo Lattes, possibilitando que o usuário gere um arquivo no formato *BibTeX*, que pode ser utilizado futuramente em *websites*, por exemplo.

Para a construção do sistema, seguiu-se um processo de Engenharia de Software realizando as etapas de levantamento de requisitos, especificação de requisitos, definição da arquitetura do sistema, implementação e testes. Foram colocadas em prática as disciplinas aprendidas no decorrer do curso, tais como Engenharia de Software, Engenharia de Requisitos, Projeto de Sistema de Software, Programação Orientada a Objetos e Desenvolvimento Web e Web Semântica. Também foram utilizados métodos e técnicas de modelagem e desenvolvimento orientado a objetos, em particular o método FrameWeb para projeto de aplicações Web baseadas em frameworks.

Palavras-chaves: Aplicação Web, Java, JSF, FrameWeb..

Lista de ilustrações

Figura 1 – Diagrama de Pacotes.	23
Figura 2 – Marvin - Core - Diagrama de Casos de Uso.	24
Figura 3 – Marvin - AlocaWeb - Diagrama de Casos de Uso.	25
Figura 4 – Marvin - BibLattes - Diagrama de Casos de Uso.	26
Figura 5 – Marvin - Core - Diagrama de Classes.	28
Figura 6 – Marvin - AlocaWeb - Diagrama de Classes.	29
Figura 7 – Marvin - BibLattes - Diagrama de Classes.	30
Figura 8 – Marvin - Arquitetura - Sistema (LIMA, 2015).	32
Figura 9 – Marvin - Implementação - Pacotes.	32
Figura 10 – Marvin - Implementação - Páginas Web	33
Figura 11 – FrameWeb - Core - Modelo Domínio.	37
Figura 12 – FrameWeb - AlocaWeb - Modelo Domínio.	38
Figura 13 – FrameWeb - BibLattes - Modelo Domínio.	38
Figura 14 – FrameWeb - <i>nemo-utils</i> - Modelo Navegação.	39
Figura 15 – Marvin - AlocaWeb - Exportar Oferta - Modelo Navegação.	40
Figura 16 – Marvin - AlocaWeb - Efetuar Alocação e Exportar Alocação - Modelo Navegação.	42
Figura 17 – Marvin - BibLattes - Enviar Currículo e Exportar Currículo - Modelo Navegação.	43
Figura 18 – FrameWeb - <i>nemo-utils</i> - Modelo Aplicação (LIMA, 2015).	44
Figura 19 – FrameWeb - Marvin - Core - Modelo Aplicação.	44
Figura 20 – FrameWeb - Marvin - AlocaWeb - Modelo Aplicação.	45
Figura 21 – FrameWeb - Marvin - BibLattes - Modelo Aplicação.	45
Figura 22 – FrameWeb - <i>nemo-utils</i> - Modelo Persistência (LIMA, 2015).	46
Figura 23 – FrameWeb - Marvin - Core - Modelo Persistência.	47
Figura 24 – FrameWeb - Marvin - AlocaWeb - Modelo Persistência.	47
Figura 25 – FrameWeb - Marvin - AlocaWeb - Modelo Persistência.	48
Figura 26 – Marvin - Tela Login.	49
Figura 27 – Marvin - Página Inicial.	49
Figura 28 – Marvin - Menu Dispositivo Móvel.	50
Figura 29 – Marvin - Lista de Turmas.	50
Figura 30 – Marvin - Tela Cadastro de Turma.	51
Figura 31 – Marvin - Tela exclusão de Período.	52
Figura 32 – Marvin - Tela Listagem Períodos Mobile.	52
Figura 33 – AlocaWeb - Gerenciar Alocações.	53
Figura 34 – AlocaWeb - Exportar Alocações.	53

Figura 35 – AlocaWeb - Relatório de Alocações.	54
Figura 36 – BibLattes - Tela Importação Currículo Lattes.	54
Figura 37 – BibLattes - Tela Gerenciamento de Publicações.	55
Figura 38 – BibLattes - Tela Gerenciamento de Publicações Mobile.	55

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
API	Application Programming Interface
CDI	Contexts and Dependency Injection
DAO	Data Access Object
EL	Expression Language
FrameWeb	Framework-based Design Method for Web Engineering
HTML	HyperText Markup Language
JAAS	Java Authentication and Authorization Service
Java EE	Java Platform, Enterprise Edition
JCP	Java Community Proces
JMS	Java Message Service
JPA	Java Persistence API
JSF	JavaServer Faces
JVM	Java Virtual Machine
MVC	Model-View-Controller
OJB	Object Relational Bridge
ORM	Object-relational Mapping
SPI	Service Provider Interface
SQL	Structured Query Language
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	eXtensible Markup Language

Sumário

1	INTRODUÇÃO	11
1.1	Objetivos	11
1.2	Metodologia	12
1.3	Organização do Texto	13
2	REFERENCIAL TEÓRICO	14
2.1	Engenharia de Software	14
2.1.1	Análise e Especificação de Requisitos	15
2.1.2	Projeto de Sistemas e Implantação	16
2.2	Desenvolvimento Web	16
2.3	O método FrameWeb	18
3	ESPECIFICAÇÃO DE REQUISITOS	21
3.1	Descrição do Escopo	21
3.1.1	Descrição Módulo AlocaWeb	21
3.1.2	Descrição Módulo BibLattes	22
3.2	Diagrama de Pacotes	22
3.3	Diagrama de Casos de Uso	23
3.3.1	Módulo Core	23
3.3.2	Módulo AlocaWeb	25
3.3.3	Módulo BibLattes	26
3.3.4	Atores x Casos de Uso	27
3.4	Diagrama de Classes	27
3.4.1	Módulo Core	27
3.4.2	Módulo AlocaWeb	28
3.4.3	Módulo BibLattes	29
4	PROJETO ARQUITETURAL E IMPLEMENTAÇÃO	31
4.1	Arquitetura do Sistema	31
4.1.1	Camada de Apresentação	32
4.1.2	Camada de Negócio	33
4.1.3	Camada de Acesso a Dados	34
4.2	Framework <i>nemo-utils</i>	35
4.3	Modelos FrameWeb	36
4.4	Apresentação do Sistema	48

5	CONSIDERAÇÕES FINAIS	56
5.1	Conclusões	56
5.2	Limitações e Perspectivas Futuras	57
	REFERÊNCIAS	58
	APÊNDICES	60

1 Introdução

O Departamento de Informática da Universidade Federal do Espírito Santo (DI/UFES) necessita de um sistema de informação para o controle de alocação de salas de aula e ofertas de disciplinas. O propósito deste sistema é facilitar a alocação de disciplinas (com seus respectivos horários, turmas e quantidade de vagas) às salas de aula disponíveis no centro, possibilitando assim uma melhor utilização do espaço, uma redução no tempo gasto para realizar tal tarefa e, ainda, a construção de um horário que facilite a vida dos estudantes.

Além disso, é importante no contexto acadêmico a existência de um sistema capaz de extrair informações relacionadas a publicações. O propósito desta ferramenta é, dado um currículo Lattes no formato *xml* (obtido no site da Plataforma Lattes), extrair todas as informações relevantes do mesmo, permitir ao usuário visualizar tais informações de uma maneira mais limpa e exportá-las em formatos adequados, para que sejam usadas, por exemplo, em suas páginas pessoais na Web.

Ambos os módulos (AlocaWeb e BibLattes) serão integrados ao Marvin. Marvin é um sistema de informação que agrega ferramentas úteis para atividades de ensino e pesquisa em uma universidade. Diversos estudantes do Departamento de Informática da Universidade Federal do Espírito Santo (DI/UFES) desenvolvem ferramentas como parte de seus projetos de graduação. Marvin é uma tentativa de integrar tais ferramentas de uma forma que possam ser realmente utilizadas por pessoas.

É muito comum, especialmente em graduações na área de informática, o desenvolvimento de ferramentas que possam apoiar certos processos de ensino e pesquisa em uma universidade. Tais ferramentas, muitas vezes, são extremamente úteis, mas acabam não sendo utilizadas pois ficam isoladas em um único sistema, que, sozinho, não é capaz de solucionar e/ou apoiar boa parte dos problemas existentes na área acadêmica. O Marvin é uma tentativa de resolver isso. Através de um sistema *web*, responsivo, o mesmo tenta integrar todas essas ferramentas em um único lugar, tornando possível a real utilização por pessoas, e otimizando diversos processos existentes dentro de uma universidade.

1.1 Objetivos

O objetivo geral deste trabalho é desenvolver um sistema Web que será utilizado por professores, técnicos e cientistas para auxiliá-los em atividades de alocação de salas de aula e de controle de publicações, utilizando os conceitos aprendidos ao longo do curso de Ciência da Computação. São objetivos específicos deste projeto:

- Produzir o documento de especificação e análise de requisitos do software;
- Produzir o documento de projeto de software, utilizando o método FrameWeb (SOUZA, 2007);
- Desenvolver o sistema de acordo com a estrutura definida no processo de Engenharia de Software, utilizando frameworks já existentes para auxiliar no desenvolvimento do sistema. Para esse objetivo exercitou-se conceitos de Programação, Linguagens de Programação, Banco de Dados e Desenvolvimento Web;
- Sugerir melhorias futuras para as limitações do sistema e para a abordagem FrameWeb.

1.2 Metodologia

A metodologia utilizada para desenvolver este trabalho foi composta pelas seguintes atividades:

1. *Revisão Bibliográfica*: estudo de boas práticas de Engenharia de Software e de Requisitos, uso e projeto de Banco de Dados Relacional, Programação Orientada a Objetos, Padrões de Projeto de Sistemas aplicados à linguagem de programação Java, entre outros;
2. *Elaboração da Documentação do Sistema*: nesta etapa, foram definidos os documentos do sistema. Em primeiro lugar, foi elaborado o Documento de Especificação de Requisitos, apresentando uma descrição geral do mundo do sistema, definição dos requisitos funcionais e não funcionais, além das regras de negócio. Também estão neste documento a apresentação dos subsistemas, casos de uso, modelo estrutural e glossário do projeto. Por fim, foi elaborado o Documento do Projeto, contendo a arquitetura do software e projeto detalhado de cada um dos seus componentes, seguindo a abordagem FrameWeb;
3. *Estudo das Tecnologias*: nesta etapa, foi necessário o estudo de tecnologias utilizadas para o desenvolvimento do sistema, tais como: linguagem de programação Java; ambiente de desenvolvimento Eclipse Java EE; banco de dados MySQL; JSF, CDI, JPA; PrimeFaces (utilizado para implementação da interface com o usuário); entre outras;
4. *Implementação e Testes*: nesta etapa, o sistema foi implementado e testado. Sempre que uma nova funcionalidade era implementada, uma série de testes era realizada para encontrar e corrigir possíveis erros;

5. *Redação da Monografia*: nesta etapa, foi realizada a escrita desta monografia. Vale ressaltar que a mesma foi escrita em *LaTeX*¹ utilizando o editor *TexStudio*² e o template *abnTeX*³ que atende os requisitos das normas da ABNT (Associação Brasileira de Normas Técnicas) para elaboração de documentos técnicos e científicos brasileiros.

1.3 Organização do Texto

Esta monografia é estruturada em cinco partes e contém, além da presente introdução, os seguintes capítulos:

- **Capítulo 2** – Referencial Teórico: apresenta uma revisão da literatura acerca de temas relevantes ao contexto deste trabalho, a saber: Engenharia de Software, FrameWeb e desenvolvimento Web;
- **Capítulo 3** – Especificação de Requisitos: apresenta a especificação de requisitos do sistema, descrevendo o minimundo e exibindo os seus diagramas de classes e casos de uso;
- **Capítulo 4** – Projeto Arquitetural e Implementação: apresenta a arquitetura do sistema, assim como as partes principais de sua implementação, ilustradas por capturas de telas do sistema;
- **Capítulo 5** – Considerações Finais: apresenta as conclusões do trabalho, dificuldades encontradas, limitações e propostas de trabalhos futuros.
- **Apêndices** – Apresenta os documentos gerados nos processos de Engenharia de Software (Documento de Requisitos, Documento de Projetos, etc.)

¹ <<http://www.latex-project.org/>>.

² <<http://www.texstudio.org/>>.

³ <<http://www.abntex.net.br>>.

2 Referencial Teórico

Este capítulo apresenta os principais conceitos teóricos que fundamentaram o desenvolvimento dos módulos AlocaWeb e BibLattes e está organizado em 3 seções. A Seção 2.1 aborda a Engenharia de Software, destacando os principais conceitos e processos utilizados. A Seção 2.3 apresenta o método FrameWeb. A Seção 2.2 apresenta os principais conceitos de desenvolvimento Web.

2.1 Engenharia de Software

O desenvolvimento de software tem crescido bastante nos últimos anos devido a sua grande importância na sociedade contemporânea. O crescimento do uso de computadores pessoais, nas organizações e nas diversas áreas do conhecimento humano gerou uma crescente demanda por soluções que realizam a automatização dos processos (LIMA, 2015).

Antes de definir o que é Engenharia de Software, é importante diferenciar o que é programação básica e o que é desenvolvimento de sistemas. Programação básica, como é visto principalmente dentro da universidade, não lida com uma enorme quantidade de linhas de código, com diferentes módulos se comunicando simultaneamente ou com integração com ferramentas externas. Já o desenvolvimento de software lida com tudo isso, e, neste momento, aparece a grande necessidade da **Engenharia de Software**, que tem como objetivo melhorar a qualidade dos produtos de software e aumentar a produtividade no processo de desenvolvimento.

A **Engenharia de Software** trata de aspectos relacionados ao estabelecimento de processos, métodos, técnicas, ferramentas e ambientes de suporte ao desenvolvimento de software (FALBO, 2014). Um processo de software pode ser visto como o conjunto de atividades, métodos e práticas que guiam os profissionais na produção de software (FALBO, 2014). Um processo de software, em uma abordagem de Engenharia de Software, envolve diversas atividades que podem ser classificadas quanto ao seu propósito em (FALBO, 2012):

- *Atividades de Desenvolvimento (ou Técnicas)*: são as atividades diretamente relacionadas ao processo de desenvolvimento do software. De maneira geral, este processo envolve as seguintes atividades: Análise e Especificação de Requisitos, Projeto, Implementação, Testes, Entrega e Implantação do Sistema. Veremos um pouco mais sobre cada uma delas a seguir;
- *Atividades de Gerência*: envolvem atividades relacionadas ao gerenciamento do

projeto de maneira abrangente. Incluem, dentre outras: processo de Gerência de Projetos, processo de Gerência de Configuração, processo de Gerência de Reutilização, etc;

- *Atividades de Controle da Qualidade*: são aquelas relacionadas com a avaliação da qualidade do produto em desenvolvimento e do processo de software utilizado.

2.1.1 Análise e Especificação de Requisitos

A especificação de requisitos compreende a primeira etapa do processo de desenvolvimento de um *software*. Nesta etapa são definidas as funcionalidades que o sistema deve conter e os critérios de qualidade sob os quais o sistema deverá operar. É caracterizada por um esforço conjunto entre clientes, usuários e especialistas de domínio a fim de traçar uma linha conjunta de entendimento entre todas as partes envolvidas a respeito de como um *software* se comportará e de como será utilizado pelo usuário. Trata-se de uma atividade complexa que não se resume somente a perguntar às pessoas o que elas desejam, mas sim analisar cuidadosamente a organização, o domínio da aplicação e os processos de negócio no qual o sistema será utilizado (SOMMERVILLE; KOTONYA, 1998).

Uma parte essencial dessa fase é a elaboração de modelos descrevendo o que o software tem de fazer (e não como fazê-lo), dita Modelagem Conceitual. Até este momento, a ênfase está sobre o domínio do problema e não se deve pensar na solução técnica, computacional a ser adotada (FALBO, 2012).

Pode-se dizer que os requisitos de um sistema incluem especificações dos serviços que o sistema deve prover, restrições sob as quais ele deve operar, propriedades gerais do sistema e restrições que devem ser satisfeitas no seu processo de desenvolvimento (FALBO, 2012). Requisitos de *software* podem ser divididos em funcionais e não funcionais.

Requisitos funcionais são declarações de serviços que o sistema deve prover, descrevendo o que o sistema deve fazer (SOMMERVILLE et al., 2003). Já os não funcionais, descrevem restrições sobre os serviços ou funções oferecidas pelo sistema (SOMMERVILLE et al., 2003), as quais limitam as opções para criar uma solução para o problema.

Um importante modelo neste processo é o modelo de casos de uso. O modelo de casos de uso é um modelo comportamental que demonstra as funções do sistema, mas de maneira estática. Ele é composto de dois tipos principais de artefatos: os diagramas de casos de uso e as descrições de casos de uso. Um diagrama de casos de uso é um diagrama bastante simples, que descreve o sistema, seu ambiente e como sistema e ambiente estão relacionados. As descrições dos casos de uso descrevem o passo a passo para a realização dos casos de uso e são essencialmente textuais (FALBO, 2012).

Os casos de uso cadastrais de baixa complexidade que envolvem as funcionalidades criar, alterar, excluir e consultar, podem ser descritos na forma de tabela como proposto

por Falbo (2012).

Ainda como produto dessa fase do desenvolvimento de um *software*, a modelagem conceitual estrutural visa modelar de forma computacional os requisitos e os casos de uso identificados anteriormente. Por meio deste modelo, é possível identificar de forma clara os tipos de entidade e os tipos de relacionamentos presentes no sistema. O propósito da modelagem conceitual estrutural, segundo o paradigma orientado a objetos, é definir as classes, atributos e associações que são relevantes para tratar o problema a ser resolvido (FALBO, 2012).

2.1.2 Projeto de Sistemas e Implantação

A fase de Projeto durante a especificação de um software tem por objetivo definir e especificar uma solução a ser implementada (FALBO, 2011). Diferentemente da etapa de Análise e Levantamento de Requisitos, onde é definido o que o sistema irá oferecer, quais as necessidades do sistema, etc., esta fase é responsável por definir como serão implementadas as funcionalidades propostas, as tecnologias a serem utilizadas para desenvolvimento e para armazenamento de dados, etc. Esta fase pode ser decomposta em duas etapas menores, a saber: (i) Projeto de Arquitetura do Sistema e (ii) Projeto Detalhado.

A etapa (i) tem como objetivo obter os relacionamentos dos componentes do sistema, assim como identificar tais componentes. Essa arquitetura deve descrever a estrutura de nível mais alto da aplicação e identificar seus principais componentes. O propósito da segunda etapa é detalhar o projeto do software para cada componente identificado na etapa anterior. Os componentes de software devem ser sucessivamente refinados em níveis maiores de detalhamento, até que possam ser codificados e testados (FALBO, 2014).

Ao final da fase de Projeto de *Software*, tem-se a arquitetura na qual o sistema será montado juntamente com o projeto dos componentes do sistema. Esses componentes, como proposto em (FOWLER, 2002), podem ser classificados da seguinte forma: interface com o usuário, lógica de negócio e persistência de dados. O primeiro é responsável por gerenciar toda a interação existente entre sistema e usuário. O segundo tem a função de implementar todas as regras de negócio e requisitos do sistema, aplicando as lógicas de negócio e repassando os dados vindos da camada de interface com o usuário para a camada de persistência. A Camada de Persistência tem como principal função armazenar os dados vindos das outras camadas e, quando necessário, deve permitir a recuperação destes dados.

2.2 Desenvolvimento Web

Aplicações Web fazem parte do cotidiano de praticamente todas as pessoas com acesso à Internet. Como qualquer outra tecnologia, a Web vem crescendo e se modificando dia após dia, tornando cada vez mais importante o desenvolvimento de aplicações que

possam ser facilmente modificadas ou atualizadas. Junto com este crescimento, uma vasta quantidade de *frameworks* acabam surgindo, alguns para solucionar problemas, outros para facilitar a implementação de certas coisas.

Com o objetivo de desenvolver tais aplicações tornou-se essencial a aplicação de conceitos existentes na Engenharia de Software, adaptados para essa nova plataforma, no desenvolvimento das aplicações Web. Características do ambiente Web, como concorrência, carga imprevisível, disponibilidade, sensibilidade ao conteúdo, evolução dinâmica, imediatismo, segurança e estética (PRESSMAN, 2011) deram origem a uma nova sub-área da Engenharia de Software, a Engenharia Web.

A Engenharia Web (*Web Engineering - WebE*), é a Engenharia de Software aplicada ao Desenvolvimento Web (PRESSMAN, 2011). Com a existência da Engenharia Web, alguns atributos de qualidade utilizados na Engenharia de Software surgiram como sendo de extrema importância no desenvolvimento de sistemas Web, são eles (OLSINA; LAFUENTE; ROSSI, 2001):

- **Usabilidade:** trata-se de um requisito de qualidade como também um objetivo de aplicações Web: permitir a acessibilidade do sistema. Logo, o site deve ter uma inteligibilidade global, permitir o feedback e ajuda online, planejamento da interface/aspectos estéticos e aspectos especiais (acessibilidade por deficientes);
- **Funcionalidade:** o software Web deve ter capacidade de busca e recuperação de informações/links/funções, aspectos navegacionais e relacionados ao domínio da aplicação;
- **Eficiência:** o tempo de resposta deve ser inferior a 10s (velocidade na geração de páginas/gráficos);
- **Confiabilidade:** relativa à correção no processamento de links, recuperação de erros, validação e recuperação de entradas do usuário;
- **Manutenibilidade:** existe uma rápida evolução tecnológica aliada à necessidade de atualização constante do conteúdo e das informações disponibilizadas na Web, logo o software Web deve ser fácil de corrigir, adaptar e estender.

A Java EE (Java Platform, Enterprise Edition) é uma plataforma padrão para desenvolver aplicações Java de grande porte e/ou para a Internet, que inclui bibliotecas e funcionalidades para implementar software Java distribuído, baseado em componentes modulares que executam em servidores de aplicações e que suportam escalabilidade, segurança, integridade e outros requisitos de aplicações corporativas ou de grande porte (FARIA, 2013).

A plataforma Java EE possui uma série de especificações (tecnologias) com objetivos distintos, por isso é considerada uma plataforma guarda-chuva. As especificações Java EE mais conhecidas são listadas em (MANZOLI, 2016):

- **Servlets:** são componentes Java executados no servidor para gerar conteúdo dinâmico para a Web, como HTML e XML;
- **JSF (JavaServer Faces):** é um framework Web baseado em Java que tem como objetivo simplificar o desenvolvimento de interfaces de sistemas para a Web, por meio de um modelo de componentes reutilizáveis;
- **JPA (Java Persistence API):** é uma API padrão do Java para persistência de dados, baseada no conceito de mapeamento objeto-relacional. Essa tecnologia traz alta produtividade para o desenvolvimento de sistemas que necessitam de integração com banco de dados;
- **EJB (Enterprise Java Beans):** são componentes que executam em servidores de aplicação e possuem como principais objetivos fornecer facilidade e produtividade no desenvolvimento de componentes distribuídos, transacionados, seguros e portáveis;
- **CDI (Contexts and Dependency Injection):** é a especificação da Java EE que trabalha com injeção de dependências.

Além disso, o Marvin também utiliza uma biblioteca de componentes de interface com o usuário chamada *Primefaces*¹. Esta biblioteca tem como principal função otimizar o tempo gasto no desenvolvimento de componentes web responsivos, como, por exemplo, tabelas, campos de formulários, botões, etc. Como resultado, a interface se torna mais robusta e agradável para o usuário.

2.3 O método FrameWeb

FrameWeb (*Framework-based Design Method for Web Engineering*) é um método de projeto para construção de sistemas de informação Web (*Web Information Systems – WISs*) baseados em *frameworks*. FrameWeb é baseado em metodologias e linguagens de modelagem bastante difundidas na área de Engenharia de Software sem, no entanto, impor nenhum processo de desenvolvimento específico (SOUZA, 2007).

A arquitetura proposta para o método *FrameWeb* foi baseada no padrão Camada de Serviço (FOWLER, 2002). Tal arquitetura é composta de três camadas (MARTINS, 2016):

¹ <<http://www.primefaces.org/>>

- Camada de Negócio (*Business Tier*): responsável pelas funcionalidades relacionadas às regras de negócio da aplicação. Esta Camada é particionada em duas: Lógica de Domínio (*Domain*) e Lógica de Aplicação (*Application*);
- Camada de Apresentação (*Presentation Tier*): responsável por funcionalidades de interface com o usuário (incluindo as telas que serão apresentadas a ele). Esta Camada, segundo o padrão proposto, é também particionada em duas outras: Visão (*View*) e Controle (*Control*);
- Camada de Acesso a Dados (*Data Access Tier*): responsável por funcionalidades relacionadas à persistência de dados.

O foco do *FrameWeb* é na etapa de Projeto de Sistemas de Informação Web, ou seja, todos os modelos utilizados estão já considerando a arquitetura a ser utilizadas, assim como frameworks, etc.

De acordo com a especificação original do método, a fase de Projeto concentra as propostas principais: (i) definição de uma arquitetura padrão que divide o sistema em camadas, de modo a se integrar bem com os *frameworks* utilizados; (ii) proposta de um conjunto de modelos de projeto que trazem conceitos utilizados pelos *frameworks* para esta fase do processo por meio da criação de um perfil UML que faz com que os diagramas fiquem mais próximos da implementação (SOUZA, 2007).

A modelagem utilizando o método *FrameWeb* é baseada em quatro principais modelos, que, juntos, são capazes de descrever todo o fluxo de execução de um sistema Web, e além disso, estrutura de banco de dados, de páginas web, etc., são eles:

- **Modelo de Domínio:** é um diagrama de classes da UML que representa o domínio do problema e o mapeamento do mesmo para a estrutura do banco de dados;
- **Modelo de Persistência:** é um diagrama utilizado para representar os objetos persistentes (que podem ser salvos no banco de dados) seguindo o padrão de projeto DAO (*Data Access Object*) (ALUR; MALKS; CRUPI, 2003).
- **Modelo de Navegação:** é utilizado para representar os componentes existentes nos pacotes de Visão (*View*) e Controles (*Controllers*), que são responsáveis pela interação com o usuário. Além disso, este modelo é capaz de mostrar o fluxo existente entre diferentes páginas de um sistema web (como certas funcionalidades existentes em uma página, levam o usuário para outra página, e assim sucessivamente).
- **Modelo de Aplicação:** é um diagrama utilizado para representar os serviços existentes no pacote de Aplicação (responsáveis por implementar os casos de uso de um sistema) e as dependências de componentes de outras camadas (Controle e Persistência).

Esses quatro modelos serão descritos na Seção 4.3 no contexto dos módulos desenvolvido neste trabalho.

3 Especificação de Requisitos

Este capítulo aborda alguns resultados da Engenharia de Requisitos para a construção dos módulos BibLattes e AlocaWeb para o sistema Marvin. Na seção 3.1, é apresentado o escopo do projeto; na seção 3.1.1 é apresentado o minimundo do módulo AlocaWeb, na seção 3.1.2 é apresentado o minimundo do módulo BibLattes, na seção 3.2, são apresentados os diagramas de pacotes, na seção 3.3, são apresentados diagramas de casos de uso e na seção 3.4, são apresentados os diagramas de classes. Os requisitos funcionais, requisitos não funcionais e regras de negócio podem ser encontrados no **Documento de Especificação de Requisitos** que está disponível no Apêndice ao final dessa monografia.

3.1 Descrição do Escopo

Esta seção descreve o funcionamento dos dois módulos desenvolvidos para o sistema Marvin, a saber: Controle de Alocação de Salas de Aula (AlocaWeb) e Extração de Dados de um CV Lattes (BibLattes).

3.1.1 Descrição Módulo AlocaWeb

O Departamento de Informática da Universidade Federal do Espírito Santo (DI/Ufes) necessita de um sistema de informação para o controle de alocação de salas de aula e ofertas de disciplinas. O propósito deste sistema é facilitar a alocação de disciplinas (com seus respectivos horários, turmas e quantidade de vagas) às salas de aula disponíveis no centro, possibilitando, assim, uma melhor utilização do espaço, uma redução no tempo gasto para realizar tal tarefa e, ainda, a construção de um horário que facilite a vida dos estudantes.

O sistema deve, portanto, permitir o cadastro de informações relevantes para o processo de alocação, como, por exemplo, dados de professores, cursos, disciplinas, turmas, salas de aula e períodos. A partir destes dados, deve ser possível realizar a alocação e, posteriormente, produzir relatórios relevantes, a saber: Ofertas de Disciplinas (lista de turmas que serão ofertadas) em um determinado período e Alocações de Salas de Aula em um determinado período.

É importante notar que antes de efetuar uma alocação, as salas de aula existentes devem ser cadastradas, assim como as turmas que serão alocadas para as salas. Depois de realizada tal alocação, o usuário poderá visualizar o relatório contendo as informações de todas as alocações em um certo período.

3.1.2 Descrição Módulo BibLattes

Pesquisadores, estudantes e professores encontram frequentemente dificuldade ao tentar produzir um website contendo informações relacionadas a seus currículos Lattes. Executar tal tarefa (extrair os dados, gerar arquivo no formato adequado e publicar o site) não é uma tarefa tão simples, principalmente para pessoas que não são da área da computação.

O propósito desta ferramenta é, dado um currículo Lattes no formato *xml* (obtido no site da Plataforma Lattes), extrair todas as informações relevantes do mesmo, permitir ao usuário visualizar tais informações de uma maneira mais limpa e exportá-las em formatos adequados, para que sejam usadas, por exemplo, em suas páginas pessoais na Web.

Ao efetuar login no sistema, uma tela será mostrada instruindo o usuário a acessar seu CV Lattes e fazer o *download* do arquivo *xml*. Com este arquivo em mãos, o usuário poderá fazer *upload* do mesmo no sistema que irá, então, efetuar a leitura do arquivo e extrair as informações necessárias, a saber: nome do(s) autor(es) de cada publicação, título, informações sobre onde e quando foi publicado, DOI, e URL para acessá-la.

Caso o usuário já tenha um CV cadastrado na base de dados, o sistema deverá fazer uma atualização de tal CV, substituindo entradas repetidas, inserindo novas entradas e removendo aquelas que não mais existem no arquivo *xml*.

Ao final de todo o processo, o usuário poderá visualizar como as informações serão exportadas, podendo, assim, corrigir o que for necessário em seu CV Lattes (é importante notar que o usuário não poderá alterar seu currículo neste sistema, apenas na própria plataforma Lattes).

A princípio, a exportação de dados será feita para o formato BibTeX, para que o usuário possa usá-la em seu site pessoal por meio de ferramentas que já suportam este tipo de formato, como, por exemplo, o *plug-in* *papercite*¹ da plataforma WordPress.

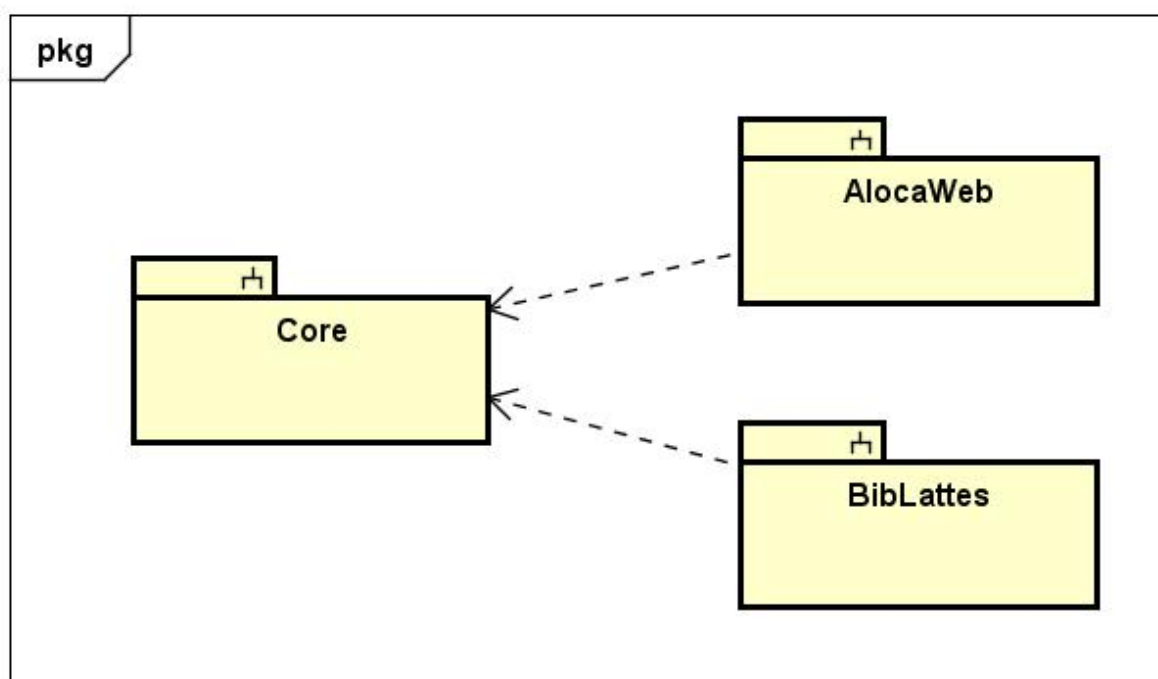
3.2 Diagrama de Pacotes

Dada a descrição do minimundo, foi possível estruturar o sistema em três diferentes pacotes, como pode ser visto na Figura 1.

O pacote *Core*, é responsável por funcionalidades que podem ser comuns a diferentes módulos do Marvin, por exemplo, Professor, Turmas, Cursos, etc.

O pacote *AlocaWeb* é responsável pelas funcionalidades relacionadas à alocação de salas de aula, e o pacote *BibLattes* é responsável por funcionalidades de importação, exportação e visualização de curriculum lattes.

¹ <<https://br.wordpress.org/plugins/papercite/>>



powered by Astah

Figura 1 – Diagrama de Pacotes.

3.3 Diagrama de Casos de Uso

Como visto na Subseção 3.2, este projeto foi dividido em dois subsistemas (módulos): AlocaWeb e BibLattes. Além disso, funcionalidades que são fortemente relacionadas ao contexto acadêmico, como Gerenciamento de Disciplinas, Cursos, Professores, Turmas, Períodos e Disciplinas acabaram ficando em um terceiro módulo: Core. Veremos na Subseção 3.3.1 os casos de uso do módulo Core, na Subseção 3.3.2 os casos de uso do módulo AlocaWeb e na Subseção 3.3.3 os casos de uso do módulo BibLattes.

3.3.1 Módulo Core

A Figura 2 mostra os casos de uso do módulo Core que serão descritos a seguir. O módulo Core foi criado para gerenciar as funcionalidades mais básicas do sistema Marvin. Os casos de uso **Gerenciar Cursos**, **Gerenciar Turma**, **Gerenciar Disciplina** e **Gerenciar Período** são do tipo cadastrais e incluem alteração, inclusão, consulta e exclusão. Já o caso de uso Exportar Oferta lida com a exportação de oferta de disciplinas em um determinado período, não sendo, portanto, um caso de uso cadastral.

O DI/Ufes possui cursos de graduação e de pós-graduação. Então, foi criado o caso de uso **Gerenciar Cursos** para que o responsável possa inserir novos cursos, possa também consultar, alterar e até mesmo excluí-lo. Para inserir um novo curso, basta informar o código, e o nome do mesmo.

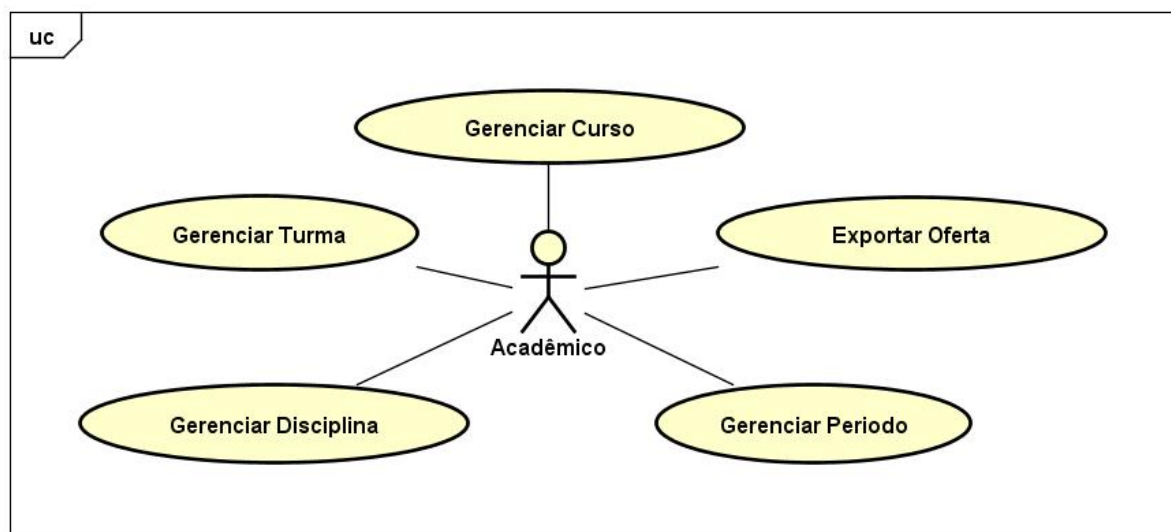


Figura 2 – Marvin - Core - Diagrama de Casos de Uso.

Um curso por si só não é suficiente para um sistema acadêmico, é necessário também a existência de períodos. Para isso, o caso de uso **Gerenciar Período** foi criado. Este caso de uso tem a função de permitir ao usuário adicionar, excluir, consultar ou até mesmo editar um determinado período. Para inserir um novo período, basta informar o semestre (primeiro ou segundo), o ano (por exemplo, 2016), a data de início e a data de término prevista para o período.

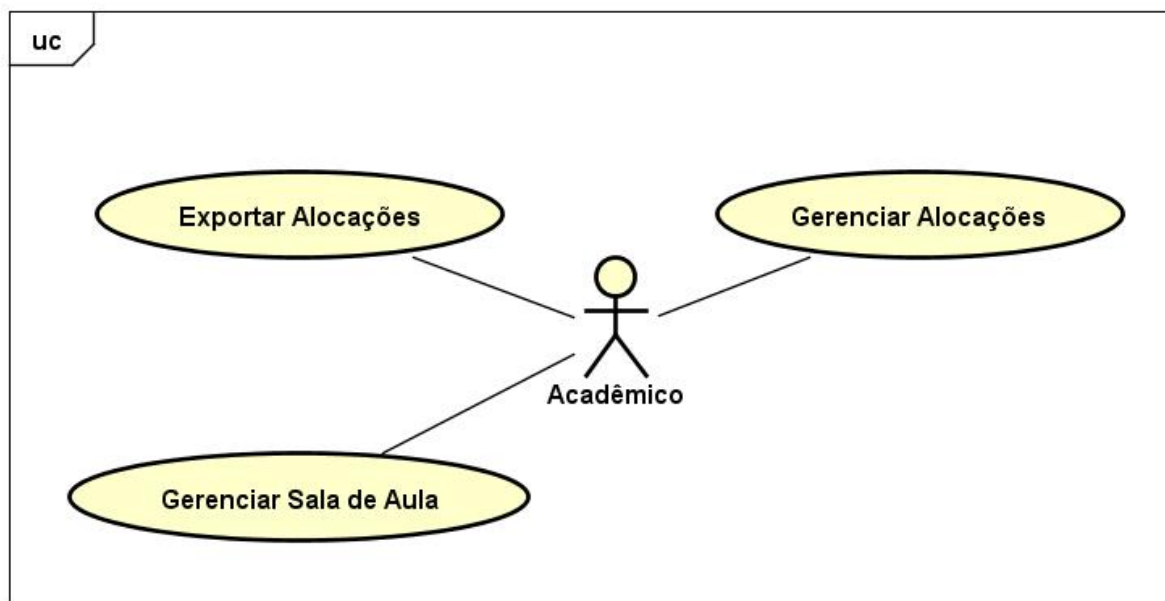
Junto com Período e Curso, as disciplinas formam a grade curricular de cursos dentro da universidade, portanto, é extremamente importante que haja um caso de uso para controle das mesmas. O caso de uso **Gerenciar Disciplina** tem tal função. Este caso de uso controla a inclusão, edição, remoção e visualização de disciplinas. Para adicionar uma disciplina, o usuário deverá informar o nome, código e curso ao qual a mesma está relacionada.

Um período, um curso, uma disciplina e um conjunto de horários formam uma turma, ou seja, em um determinado período, uma disciplina será ofertada para um determinado curso e será lecionada em certos horários - formando então uma turma. Para controlar isso, temos o caso de uso **Gerenciar Turma**. Sendo possível então a inclusão, remoção, edição e visualização de turmas. Para criar uma nova turma, é necessário informar a quantidade de vagas ofertadas, o período, uma disciplina, um curso e caso já esteja definido, um professor.

Por último temos o caso de uso **Exportar Oferta**. Dado que é possível criar turmas, é importante que o usuário possa gerar um relatório com as turmas existentes em cada período, facilitando assim a vida do estudante que deseja saber quais as disciplinas que poderão ser cursadas em um certo período e eventualmente quem irá leciona-las.

3.3.2 Módulo AlocaWeb

A Figura 3 mostra os casos de uso do módulo **AlocaWeb** que serão descritos abaixo. Este módulo foi criado para gerenciar as funcionalidades de alocação de turmas às salas de aula e exportação de alocações por período. Os casos de uso responsáveis por tais funcionalidades são: **Exportar Alocações**, **Gerenciar Alocações** e **Gerenciar Sala de Aula**.



powered by Astah

Figura 3 – Marvin - AlocaWeb - Diagrama de Casos de Uso.

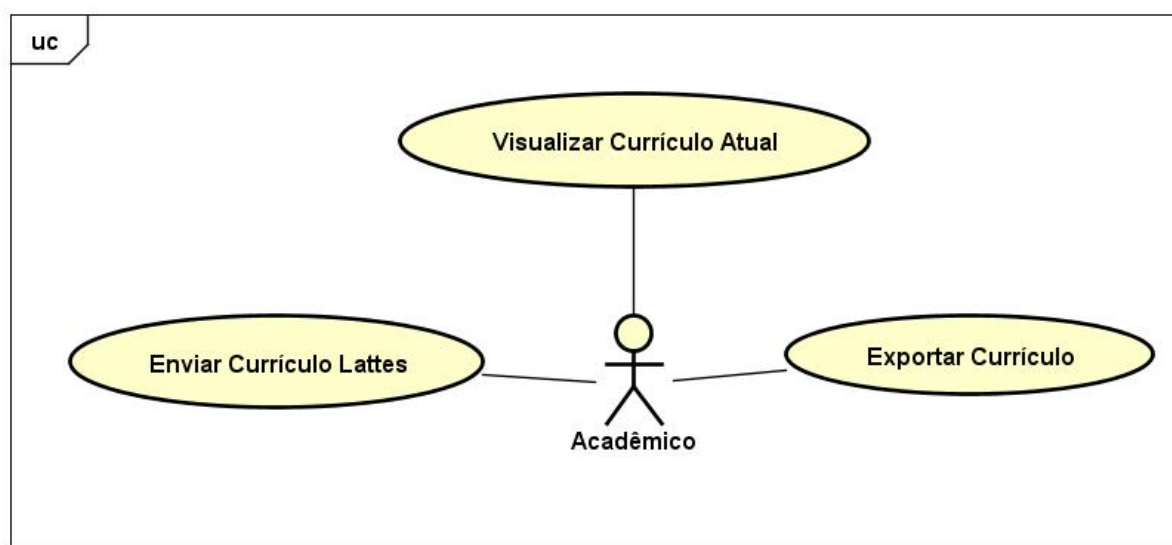
Dentro de uma universidade existem salas de aula. Como este módulo irá lidar com alocação de turmas às salas de aula, é necessário a existência de um caso de uso para lidar com tais salas, portanto temos então o caso de uso **Gerenciar Sala de Aula**. Este também é um caso de uso Cadastral, permitindo o usuário adicionar, remover, consultar e editar salas de aula. Para criar uma sala de aula, é necessário informar prédio, andar, número de cadeiras e número da sala.

No caso de uso **Gerenciar Alocações** o usuário poderá incluir, excluir, consultar e deletar alocações existentes. Para criar uma nova alocação, ele deverá informar a Turma à qual tal alocação se refere e a sala de aula que será utilizada.

Além de criar alocações, o sistema irá permitir que seja gerado um relatório com as mesmas. O caso de uso **Exportar Alocações** é o responsável por isso. Através deste caso de uso, o usuário poderá exportar todas as alocações existentes em um determinado período.

3.3.3 Módulo BibLattes

A Figura 4 mostra os casos de uso do módulo **BibLattes**. Este módulo é responsável por todas as funcionalidades relacionadas ao controle de produções bibliográficas. Através dele o usuário poderá enviar um currículo Lattes ao sistema, visualizá-lo de forma amigável e, caso deseje, exportar este currículo no formato *BibTeX*. Existem diversos tipos de produções bibliográficas existentes. As suportadas por este módulo são: Artigos, Livros, Capítulos de Livros e Trabalhos em Eventos. Os casos de uso responsáveis por tais funcionalidades são: **Visualizar Currículo Atual**, **Exportar Currículo** e **Enviar Currículo Lattes**.



powered by Astah

Figura 4 – Marvin - BibLattes - Diagrama de Casos de Uso.

Através do caso de uso **Enviar Currículo Lattes**, o usuário poderá efetuar o *upload* de um currículo lattes no formato *xml* extraído da plataforma Lattes. O sistema irá então atualizar o currículo caso já exista um cadastrado ou adicionar um novo. Com o objetivo de tornar o sistema mais completo, existe o caso de uso **Visualizar Currículo Atual**. Este caso de uso permite ao usuário visualizar o currículo que foi adicionado no sistema, possibilitando ao mesmo visualizar possíveis erros no que foi enviado. É importante notar que o sistema não irá permitir a edição de itens do currículo. Caso deseje, o usuário poderá efetuar as alterações na Plataforma Lattes, obter um novo arquivo *xml* e então enviá-lo novamente para o sistema.

Por último, temos o caso de uso **Exportar Currículo**. Este caso de uso irá extrair do banco de dados todas as produções bibliográficas do usuário e produzir um arquivo no formato *BibTeX*, que pode ser utilizado em websites, por exemplo.

3.3.4 Atores x Casos de Uso

O modelo de casos de uso visa capturar e descrever as funcionalidades que um sistema deve prover para os atores que interagem com o mesmo. No contexto deste projeto, temos apenas um ator, que é o Acadêmico, portanto, sempre que nos referirmos a usuário, estaremos nos referindo a um Acadêmico. Todos os casos de uso existentes neste sistema serão realizados por este ator.

Um Acadêmico pode ser um professor, técnico administrativo ou até mesmo um aluno. Para evitar que um aluno, por exemplo, possa cadastrar um curso, (MANZOLI, 2016) implementou a funcionalidade de controle de permissões para o Marvin. Esse controle foi implementado em forma de papéis, onde um Acadêmico pode ter diferentes papéis (Professor, técnico, etc). No contexto deste projeto este controle não existe, por este motivo todos os atores são do tipo Acadêmico, e não existem papéis definidos.

Maiores informações e detalhes sobre os casos de uso poderão ser consultados no **Documento de Análise de Requisitos** que está disponível no Apêndice ao final dessa monografia.

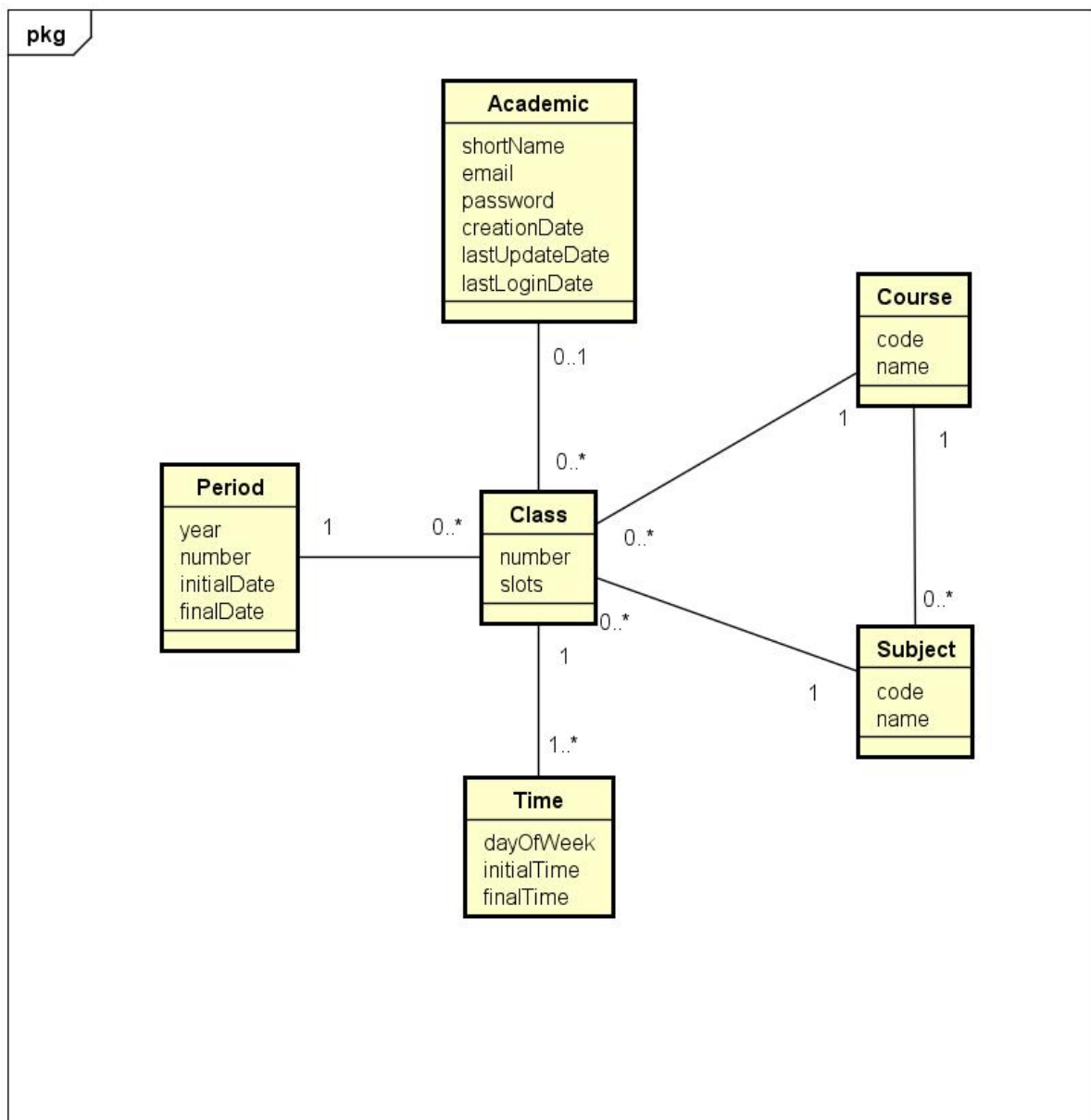
3.4 Diagrama de Classes

Assim como os casos de uso na Seção 3.3, os diagramas de classes estão divididos de acordo com a divisão dos subsistemas. Na Subseção 3.4.1 estão as classes pertencentes ao subsistema **Core**, na Subseção 3.4.2 estão as classes pertencentes ao módulo **AlocaWeb** e na Subseção 3.4.3 estão as classes pertencentes ao módulo **BibLattes**.

3.4.1 Módulo Core

A Figura 5 exibe o diagrama de classes do subsistema **Core**. A classe mais importante deste subsistema é a classe **Class** (Turma). Uma *turma* é o ponto central dentro de uma universidade. Ela irá existir em um *período* específico (Period), está relacionada a um *curso* (Course), uma *disciplina* (Subject), será lecionada em certos *horários* (Time), e pode ou não ter um *professor* (Academic) alocado à ela (o professor pode ser definido em um outro momento) - Neste caso, o Acadêmico *Academic* teria o papel de professor.

É importante notar que, apesar de não existir o caso de uso Gerenciar Acadêmico (*Manage Academic*), a Figura 5 apresenta a classe Acadêmico (*Academic*). Isso acontece porque a base do Marvin já possui tal funcionalidade. Esta classe está sendo apresentada aqui apenas para deixar claro como uma turma é definida.



powered by Astah

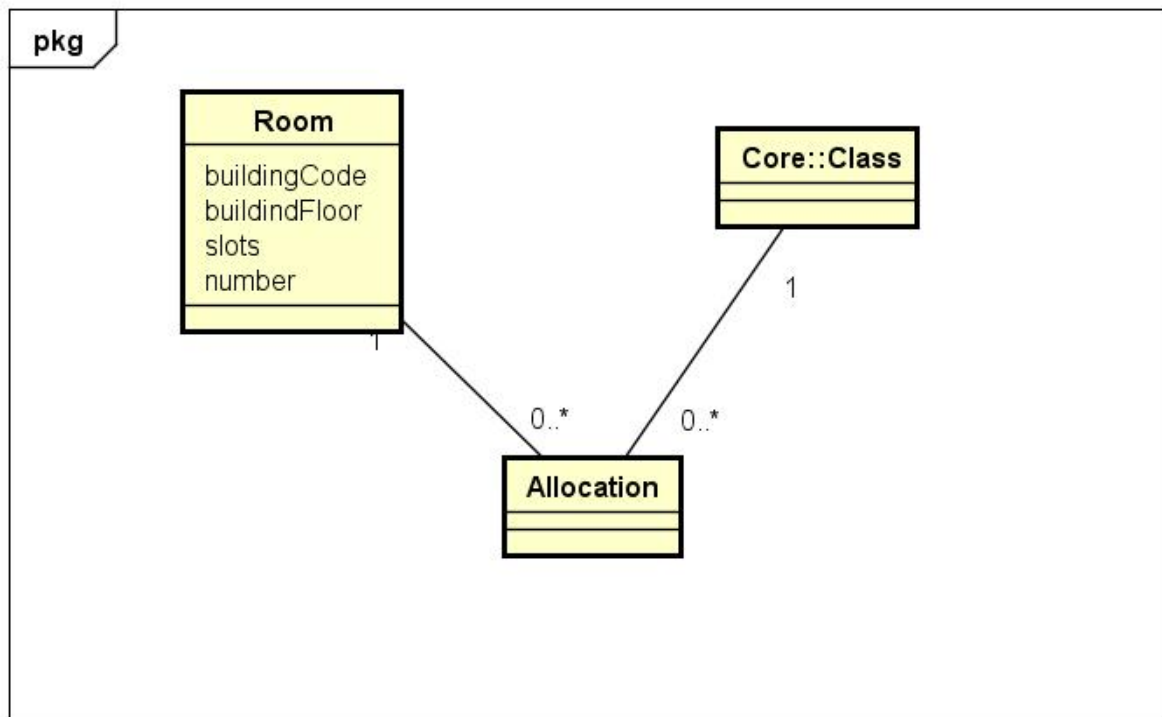
Figura 5 – Marvín - Core - Diagrama de Classes.

3.4.2 Módulo AlocaWeb

A Figura 6 exibe o diagrama de classes do subsistema **AlocaWeb**. A classe **Class** (turma) pertence ao módulo **Core**, como pode ser visto na figura. Faz parte então deste subsistema a classe **Allocation** (alocação) e **Room** (sala de aula). A classe **Allocation** desempenha um papel bastante importante. Uma *alocação* é composta por uma *sala de aula* e uma *turma*, ou seja, estamos definindo qual *turma* irá ocupar uma determinada *sala*.

Uma *sala de aula* (**Room**) está localizada em um prédio (**buildingCode**), em um certo andar (**buildingFloor**), possui um certo número de cadeiras (**slots**) e possui também

um número.



powered by Astah

Figura 6 – Marvin - AlocaWeb - Diagrama de Classes.

É importante notar neste diagrama que uma turma só pode estar alocada a uma sala em um certo espaço de tempo. Além disso, uma sala não pode ser alocada para duas turmas diferentes, caso elas tenham alguma interseção de horário. Outro detalhe importante é que caso um professor esteja alocado a uma turma, a alocação de sala só poderá ser feita em horários que não coincidam com outras alocações para tal professor.

Maiores informações sobre o diagrama de classes poderão ser consultadas no Documento de Especificação de Requisitos, disponível no Apêndice ao final dessa monografia.

3.4.3 Módulo BibLattes

A Figura 7 exhibe o diagrama de classes do subsistema **BibLattes**. Cada usuário do sistema (Academic - Lembrando que no contexto deste projeto não existem papéis específicos para um Acadêmico) pode possuir apenas um currículo cadastrado no sistema, portanto, cada *publicação* possui um dono (Academic). Além disso, uma publicação possui também *autores* (Author), que podem ser autores de mais de uma publicação, por exemplo.

Livros (Book), *Capítulos de Livros* (Collection), *Artigos* (Article) e *Trabalhos em Eventos* (Proceeding) são tipos de publicação (todos possuem um título e um ano de publicação).

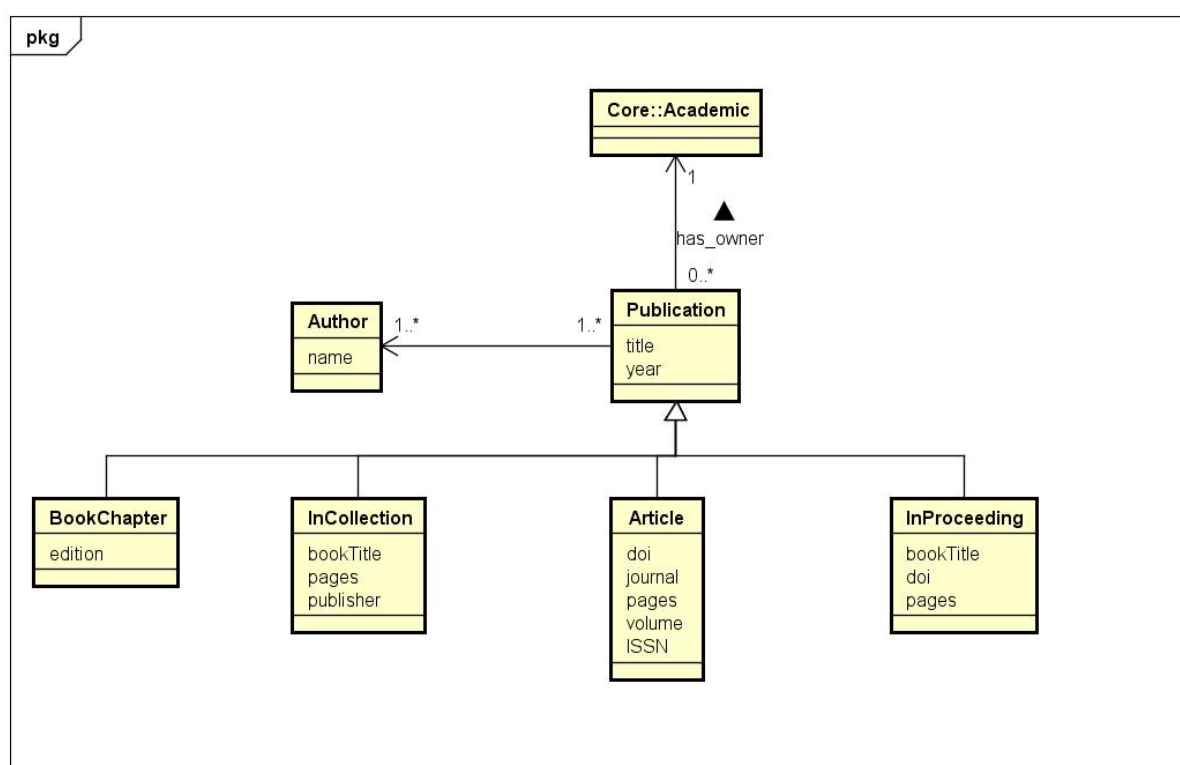


Figura 7 – Marvin - BibLattes - Diagrama de Classes.

4 Projeto Arquitetural e Implementação

Como proposto na Seção 2.1, após as fases de especificação e análise de requisitos, ocorre a fase de projeto do sistema. Esta etapa é responsável pela modelagem de como será a implementação do sistema, incorporando, aos requisitos, as tecnologias a serem utilizadas.

Neste capítulo iremos mostrar a arquitetura do sistema, assim como algumas partes de sua implementação e apresentar as principais telas do sistema. Na Seção 4.1, a arquitetura do sistema é descrita; na Seção 4.2, o framework nemo-utils é apresentado; na Seção 4.3 os modelos FrameWeb são apresentados. Por fim, na Seção 4.4, são apresentadas algumas telas e características da ferramenta.

4.1 Arquitetura do Sistema

O sistema Marvin, inicialmente, era composto apenas do subsistema *core* (contendo as principais funcionalidades do mesmo). Com a criação de dois novos módulos (*BibLattes* e *AlocaWeb*), foi necessária, então, a criação de dois novos subsistemas (implementados como pacotes Java), seguindo a divisão de subsistemas feita na análise dos requisitos e apresentada no Capítulo 3.

O módulo `br.ufes.inf.nemo.marvin.core` contém as classes mais importantes do sistema, a saber: Acadêmico, Turmas, Cursos, Períodos, e Disciplinas, e seus respectivos cadastros. Já o módulo `br.ufes.inf.nemo.marvin.alocaweb` contém as funcionalidades do subsistema **AlocaWeb**, e o módulo `br.ufes.inf.nemo.marvin.biblattes` contém as funcionalidades do subsistema **BibLattes**. Mais à frente iremos detalhar um pouco mais as subdivisões desses módulos.

Estes três módulos são, ainda, subdivididos em camadas segundo a arquitetura proposta na Seção 2.3. A Figura 8 mostra tal arquitetura. Os módulos *AlocaWeb* e *BibLattes* foram divididos em três camadas, sendo elas: apresentação (*Presentation Tier*), negócio (*Business Tier*) e acesso a dados (*Data Access Tier*). Esta figura mostra, também, as tecnologias Java associadas a cada pacote. Tais tecnologias foram citadas na Seção 2.2. É importante notar a utilização do CDI na ligação entre o pacote *Control* e o pacote *Application* e entre o pacote *Application* e o pacote *Persistence*. A função desta tecnologia é fazer com que um *Managed Bean*, por exemplo, tenha acesso a um *Session Bean* sem ser necessário explicitar isso no código Java. Tudo que o programador tem que fazer é declarar uma variável com a anotação específica, e o próprio CDI irá injetar o *Bean* correspondente.

A Figura 9 exibe os pacotes que foram criados para os módulos *BibLattes* e *AlocaWeb*.

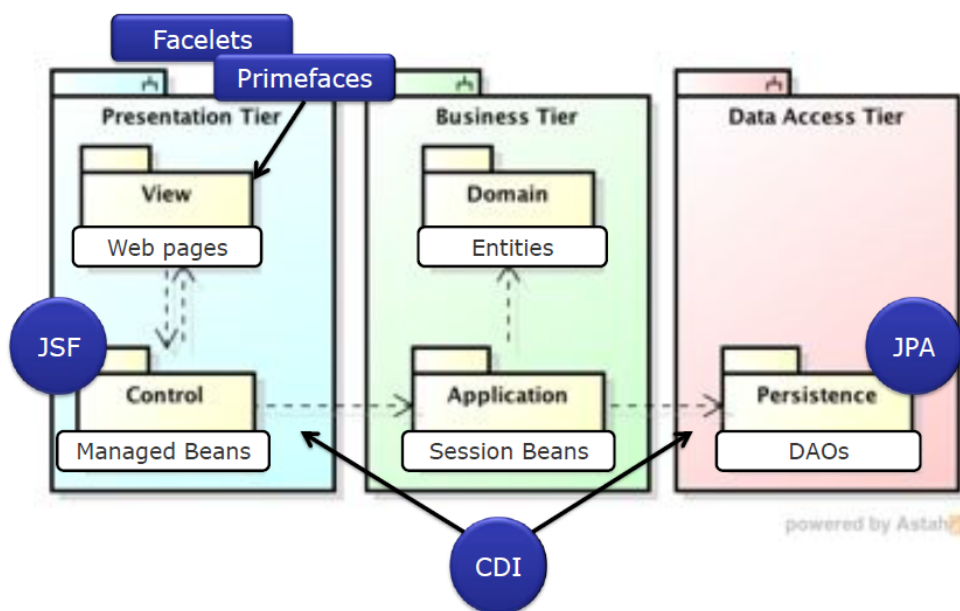


Figura 8 – Marvin - Arquitetura - Sistema (LIMA, 2015).

Como podemos perceber, os pacotes foram agrupados por módulos e em seguida de acordo com a arquitetura apresentada na Figura 8. A seguir iremos detalhar um pouco mais cada um deles.

```

br.ufes.inf.nemo.marvin.alocaweb.application
> br.ufes.inf.nemo.marvin.alocaweb.controller
br.ufes.inf.nemo.marvin.alocaweb.domain
br.ufes.inf.nemo.marvin.alocaweb.persistence
br.ufes.inf.nemo.marvin.biblattes.application
> br.ufes.inf.nemo.marvin.biblattes.controller
> br.ufes.inf.nemo.marvin.biblattes.domain
br.ufes.inf.nemo.marvin.biblattes.persistence
br.ufes.inf.nemo.marvin.biblattes.sparql
br.ufes.inf.nemo.marvin.converters
> br.ufes.inf.nemo.marvin.core.application
> br.ufes.inf.nemo.marvin.core.controller
br.ufes.inf.nemo.marvin.core.domain
br.ufes.inf.nemo.marvin.core.exceptions
br.ufes.inf.nemo.marvin.core.persistence

```

Figura 9 – Marvin - Implementação - Pacotes.

4.1.1 Camada de Apresentação

A **camada de apresentação** foi subdividida em visão (*View*) e controle (*Control*). A parte da visão é formada pelas páginas Web. A parte de controle contém os controladores que realizam a comunicação entre a interface e a aplicação.

A estrutura Web do sistema Marvin, cujas páginas Web fazem parte da visão

da camada de apresentação está organizada conforme a Figura 10. Existe uma pasta raiz chamada `WebContent` que contém todos os arquivos da visão. Esta pasta possui três subpastas que representam os módulos do Marvin, a saber: `alocaWeb`, `bibLattes` e `core`. Dentro da pasta `core`, foi criada uma subpasta para cada um dos casos de uso básicos (CRUD). Isso ajuda na organização e caso seja necessário criar um novo caso de uso, basta adicionar uma nova pasta e os arquivos necessários.

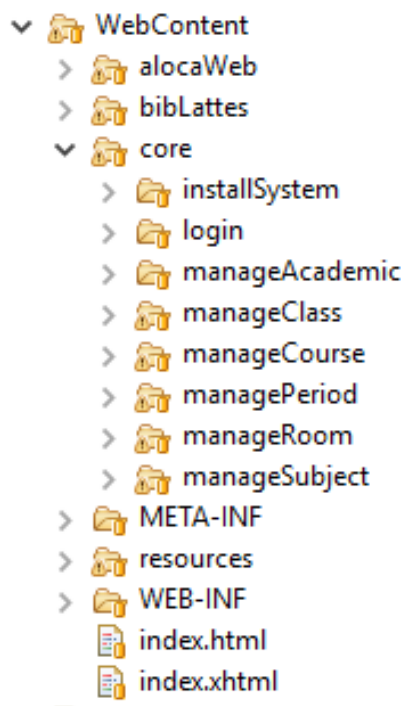


Figura 10 – Marvin - Implementação - Páginas Web

As pastas que implementam os casos de uso seguem um padrão que foi definido no framework do *nemo-utils* (cf. Seção 4.2). Esse padrão de visão consiste em duas páginas, sendo a primeira chamada `form.xhtml`, que é responsável por elencar os dados das entidades para que possam ser modificados e armazenados no banco de dados. Já a página `list.xhtml` é responsável por recuperar e exibir para o usuário as informações das entidades que estão armazenadas no banco de dados.

Dentro da pasta raiz `WebContent`, temos as páginas `index.html` e `index.xhtml` que são as páginas iniciais do sistema.

Um decorador é utilizado para definir o layout da página e o menu que está sendo utilizado. A pasta `resources` contém a subpasta `default` que contém o decorador.

4.1.2 Camada de Negócio

A **camada de negócio** foi subdividida em domínio (*Domain*) e aplicação (*Application*). A parte do domínio é formada pelas entidades de negócio (classes de domínio),

enquanto a aplicação contém as validações dos dados e implementação dos casos de uso (funcionalidades do sistema).

Os pacotes `core.domain`, `alocaweb.domain` e `biblattes.domain` contém a definição de cada uma das classes do sistema Marvin. Cada uma dessas entidades está definida em um arquivo `*.java` e, através de anotações, identificam como o mapeamento objeto-relacional irá ocorrer. Através desse mapeamento, o JPA irá gerenciar os objetos no banco de dados automaticamente, sem precisar de nenhuma intervenção do desenvolvedor. É nesse momento que é realizado também o relacionamento entre as classes do sistema utilizando as anotações `@OneToMany`, `@ManyToOne` ou `@ManyToMany` de JPA. Por fim, nesse pacote existe um arquivo para cada classe com o mesmo nome e um “_” no final (chamada de *static meta-model* ou metamodelo estático), que declara os atributos que poderão ser utilizados para realizar as consultas no banco de dados utilizando os conceitos de Criteria API. Essas consultas serão implementadas na camada de persistência.

Os pacotes `core.application`, `alocaweb.application` e `biblattes.application` contém os componentes que fazem a comunicação entre a apresentação (controladores) e a persistência (DAOs), implementando as funcionalidades do sistema descritas em seus casos de uso (cf. Cap. 3). Faz também as validações das informações antes de chamar os métodos de acesso a dados. Essas validações serão feitas ao tentar criar, modificar ou deletar uma entidade.

4.1.3 Camada de Acesso a Dados

A **camada de acesso a dados** possui uma única parte responsável pela persistência (*Persistence*) representada pelos pacotes `core.persistence`, `alocaweb.persistence` e `biblattes.persistence`, que contêm os objetos responsáveis por fazer a comunicação com o banco de dados. Esses objetos são conhecidos como DAO (*Data Access Object*) (ALUR; MALKS; CRUPI, 2003) e serão responsáveis por armazenar e recuperar os dados do banco de dados.

Sobre a arquitetura do banco de dados, conforme explicado anteriormente, o sistema Marvin utiliza o JPA para fazer o mapeamento objeto relacional e, através desse mapeamento, o próprio JPA irá gerenciar os objetos no banco de dados automaticamente. Com isso, foi utilizada a anotação `@Entity` nas classes do domínio para realizar a persistência dos dados.

4.2 Framework *nemo-utils*

O framework *nemo-utils*¹, fornece ao usuário uma série de facilidades no que diz respeito a funcionalidades do tipo CRUD (*create, read, update, delete*). Este framework implementa, genericamente, as operações básicas existentes entre a aplicação e o banco de dados, fazendo com que o desenvolvedor se preocupe apenas em adaptar e indicar quais as entidades de domínio que serão utilizadas e/ou mapeadas, diminuindo, assim, drasticamente o tempo gasto para implementar tais funcionalidades.

O módulo *AlocaWeb* e seus casos de uso são, em boa parte, cadastrais. Por este motivo, praticamente todos os controladores utilizadas pelo módulo *AlocaWeb* herdam da classe `CrudController` do *nemo-utils*. Uma outra vantagem deste *framework* é a facilidade na criação de filtros, por exemplo. Através do método `initFilters`, implementado na classe `CrudController`, é possível adicionar filtros em uma página de uma maneira extremamente simples.

O *nemo-utils* é composto de algumas classes básicas:

1. **CrudController** - Classe responsável por armazenar dados das páginas *Web* e realizar a comunicação com a camada de aplicação;
2. **CrudServiceBean** - Classe responsável por realizar as validações e por fazer a comunicação com a camada de acesso a dados (ao implementar esta classe, o usuário é obrigado a implementar também os métodos de validação existentes nela);
3. **PersistentObjectSupport** - Todas as entidades do domínio herdam dessa classe. É uma implementação do padrão para objetos persistentes, que utiliza **EJB 3** como padrão de anotações para persistência;
4. **BaseJPADAO** - Classe responsável por realizar as operações no banco de dados, tais como: consultas, modificação, inserção e exclusão de dados.

Praticamente todas as classes existentes nos módulos *AlocaWeb* e *BibLattes* utilizaram o *nemo-utils* de alguma forma. A principal razão disso foi atender ao requisito não funcional **RNF - 4 - Reusabilidade**. Este requisito diz que o sistema deve explorar o potencial de reutilização. Como dito anteriormente, o objetivo do *nemo-utils* é simplificar a criação de casos de uso cadastrais, evitando que o usuário precise recriar tais funcionalidades para cada entidade de domínio. Por este motivo o uso deste *framework* se tornou essencial.

¹ *nemo-utils* – <https://github.com/nemo-ufes/nemo-utils>

4.3 Modelos FrameWeb

Nesta seção, são exibidos os modelos FrameWeb, citados anteriormente na Seção 2.3. Esses modelos também estão divididos nas camadas da arquitetura do sistema, conforme citado na Seção 4.1.

Apresentamos primeiramente o **Modelo de Domínio**. Os mapeamentos de persistência são meta-dados das classes de domínio que permitem que os *frameworks* ORM (*Object-Relational Mapping*), *frameworks* que realizam o mapeamento objeto-relacional automaticamente, transformem objetos que estão na memória, em linhas de tabelas no banco de dados relacional. Por meio de mecanismos leves de extensão da UML, como estereótipos e restrições, adicionamos tais mapeamentos ao diagrama de classes de domínio, guiando os desenvolvedores na configuração do *framework* ORM. Apesar de tais configurações serem relacionadas mais à persistência do que ao domínio, elas são representadas no Modelo de Domínio porque as classes que são mapeadas e seus atributos são exibidas neste diagrama.

A Figura 11 mostra o modelo de domínio para o módulo **Core**. Assim como na implementação do banco de dados, boa parte dos atributos na imagem possuem tamanho (**size**) definido. As classes **Academic** e **Period** possuem atributos do tipo data. Na restrição destes atributos informamos se precisão vai ser *time*, armazenando no banco de dados somente a hora, *date*, apenas a data, ou *timestamp* armazenado ambos, data e hora. Neste último caso não é preciso colocar na restrição pois é a opção *default*. Além disso, o valor *null* no atributo do tipo Data em **Academic** significa que o mesmo pode ser nulo no banco de dados.

A Figura 12 mostra o modelo de domínio para o módulo **AlocaWeb**. A classe **Allocation** está relacionada com as classes **Room** e **Class** do módulo **Core**, que foi definida na Figura 11.

Por último, a Figura 13 fornece o modelo de domínio para o módulo **BibLattes**. As classes **BookChapter**, **InCollection**, **Article** e **InProceeding** são todas tipos específicos de publicação (**Publication**). Além disso, uma publicação pode possuir diversos autores (**Author**), mas apenas um dono (*owner*). No contexto deste módulo, dono é a pessoa que fez *upload* do currículo Lattes. Portanto, se dois currículos iguais forem enviados por duas pessoas diferentes, teremos duas publicações iguais no banco, mas com *owners* diferentes.

Todas as classes de domínio estendem de *PersistentObjectSupport* do framework *nemo-utils*, sendo que essa herança não é mostrada no diagrama com o intuito de não poluí-lo com várias associações.

Diferente da abordagem original do FrameWeb proposto em 2007, todos os atributos que são não nulos tiveram a *tag not null* omitida.

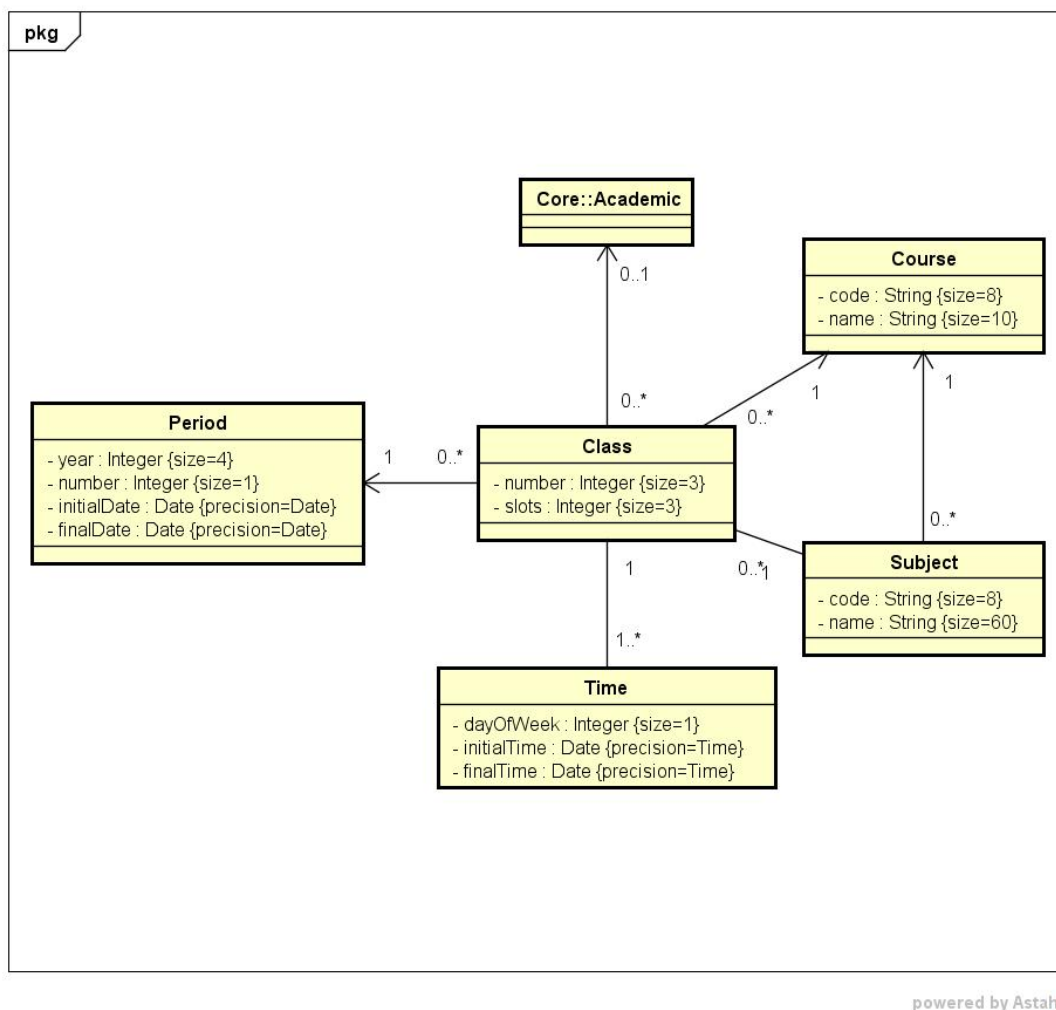
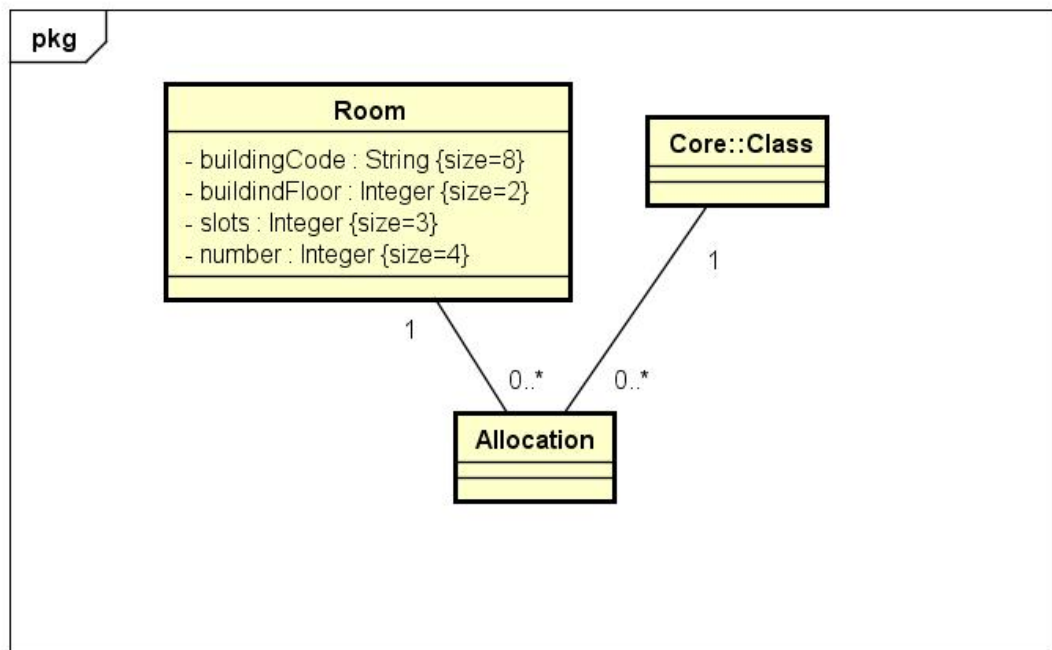


Figura 11 – FrameWeb - Core - Modelo Domínio.

Já o **Modelo de Navegação** é um diagrama de classe da UML que representa os diferentes componentes que formam a camada de Apresentação, como páginas Web, formulários HTML e classes de ação. Esse modelo é utilizado pelos desenvolvedores para guiar a codificação das classes e componentes dos pacotes **Visão** e **Controle**.

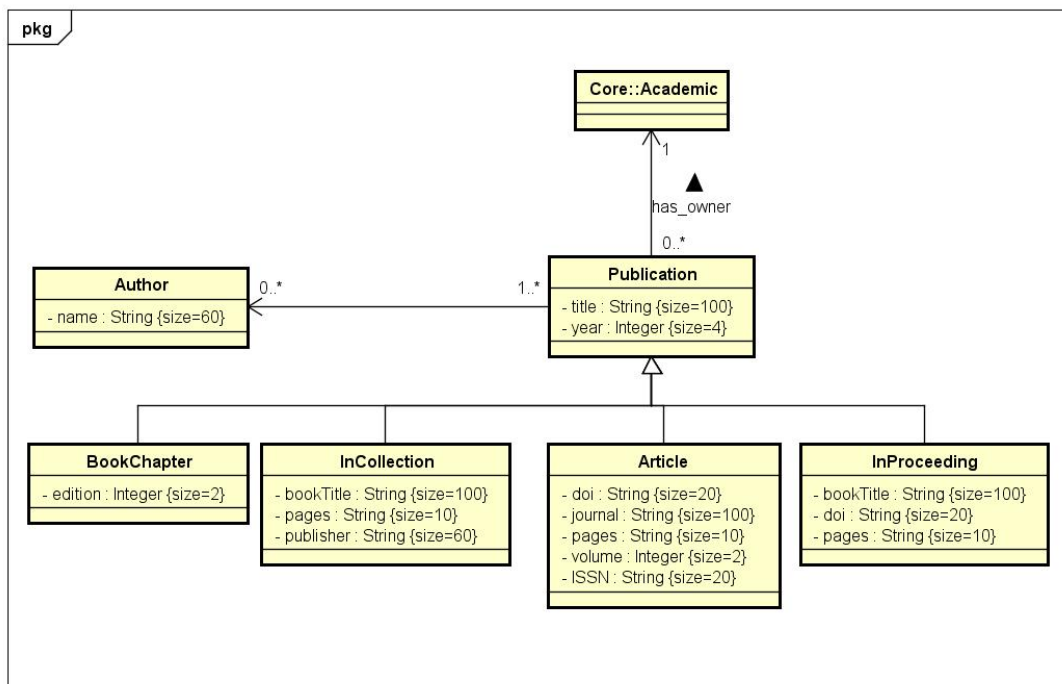
Em formulários HTML, atributos representam campos do formulário, que devem ter seus tipos definidos de acordo com a biblioteca de componentes utilizada. Como neste trabalho foi utilizado PrimeFaces, os tipos ficarão com o prefixo “p” (ex.: `p.inputText`, `p.commandButton`, etc.). A classe de ação é o principal componente do modelo. Suas associações de dependência ditam o controle de fluxo quando uma ação é executada.

As funcionalidades criar, editar, excluir e visualizar (abreviadas de CRUD, do inglês *create, read, update e delete*), seguem um mesmo fluxo de execução e de interação com o usuário. Tais funcionalidades são similares para todos os casos de uso cadastrais devido à utilização da ferramenta *nemo-utils*. Esse fluxo de execução similar é representado pela Figura 14 que é um modelo de apresentação genérico.



powered by Astah

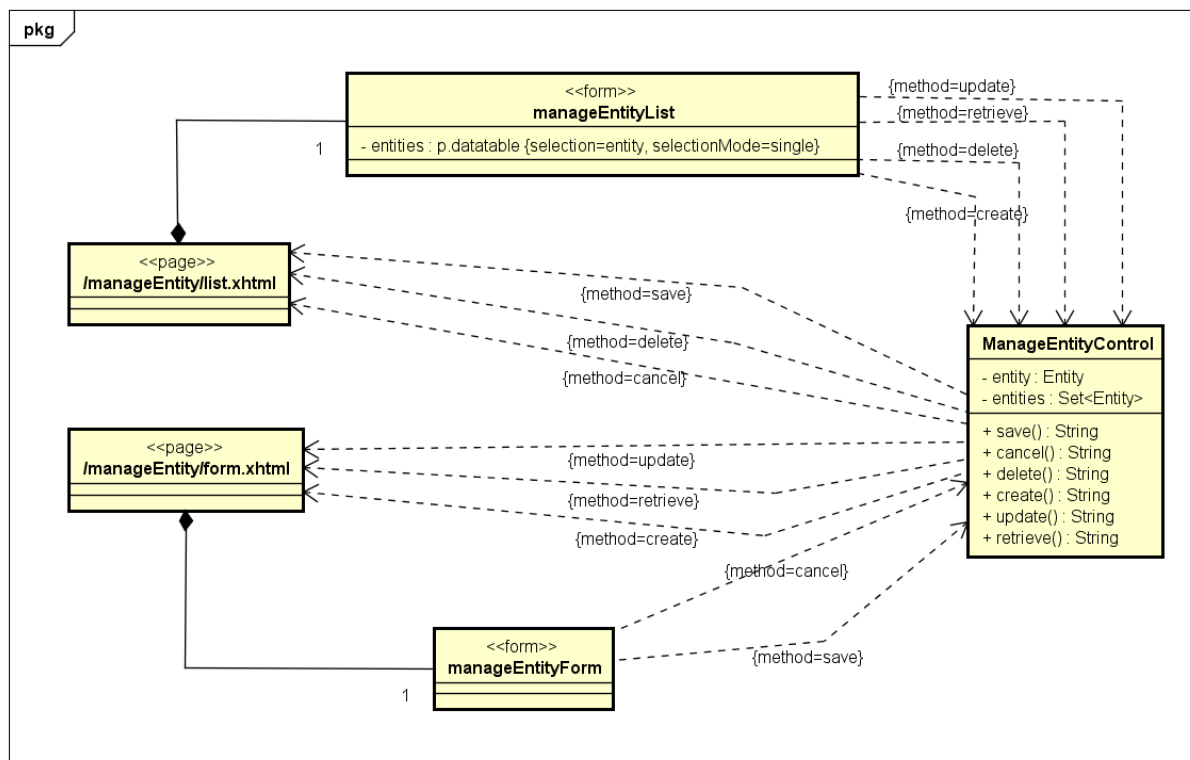
Figura 12 – FrameWeb - AlocaWeb - Modelo Domínio.



powered by Astah

Figura 13 – FrameWeb - BibLattes - Modelo Domínio.

A Figura 14 apresenta o modelo de navegação do framework nemo-utils. Este é um modelo genérico para todas as entidades que são do tipo CRUD (e que utilizam o nemo-utils). Deve existir, para cada entidade, uma página chamada *list.xhtml*, que irá conter a listagem de todos os objetos deste tipo existentes no banco. Essa página possui



powered by Astah

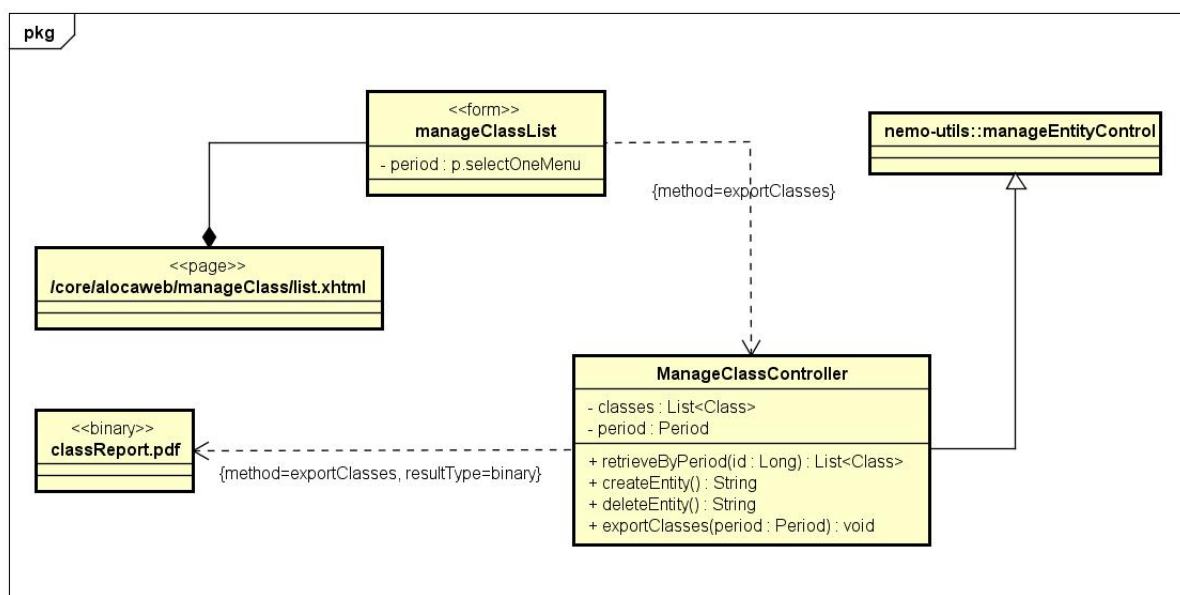
Figura 14 – FrameWeb - *nemo-utils* - Modelo Navegação.

um formulário relacionado (*manageEntityList*), que armazena todas as entidades sendo mostradas na pagina citada anteriormente. Ao selecionar uma entidade da tabela, a mesma é atribuída ao atributo *entity* existente em *ManageEntityControl*, sendo possível então executar certas ações, tais como: atualizar o objeto selecionado (*update*), ler o objeto selecionado (*retrieve*), e excluir o objeto selecionado (*delete*). Além disso, o usuário pode criar um novo objeto (*create*). Todas estas funcionalidades, ao serem executadas, irão chamar os respectivos métodos existentes em *ManageEntityControl*. Ao executar tais métodos, existem certos retornos, identificados nas relações de dependência saindo de *ManageEntityControl*. Por exemplo, ao executar a funcionalidade *create*, o usuário será redirecionado para a pagina *form.xhtml*, onde poderá incluir os dados do novo objeto, e então executar o método *save*, que irá salvar o objeto, e como retorno, irá ser redirecionado para a página inicial (contendo a listagem de todas as entidades existentes no banco).

Para os casos de uso que apresentam funções diferentes das CRUDs, o modelo anterior não pode ser aplicado. A Figura 15 apresenta o modelo de navegação para o fluxo *Exportar Oferta de Disciplinas* do caso de uso *Exportar Oferta de Disciplinas*. É importante notar que a listagem de disciplinas e o filtro por período serão implementados utilizando o *nemo-utils* e, portanto, o passo de filtragem por período não será mostrado no modelo.

Podemos perceber que o modelo possui uma página web marcada com estereótipo

«page», Esta página possui um formulário (*manageClassList*), marcado aqui como o estereótipo «form» que possui o atributo *period*. Este atributo é utilizado em conjunto com o filtro do *nemo-utils*. Ao selecionar um período, o usuário poderá exportar as ofertas, fazendo com o que o método `exportClasses()` seja executado. Este método se encontra na classe `ManageClassController`. O mesmo irá processar a requisição e como resultado, retornará um arquivo no formato PDF (por este motivo o estereótipo «binary» no resultado do método `exportClasses()`).



powered by Astah

Figura 15 – Marvin - AlocaWeb - Exportar Oferta - Modelo Navegação.

A Figura 16 exibe o modelo de navegação para o fluxo *Efetuar Alocação* do caso de uso *Gerenciar Alocações* e o fluxo *Exportar Alocação* do caso de uso *Exportar Alocação*. É importante notar neste modelo que, quando o *form* tem um atributo **X** (neste caso *class*, por exemplo) e o controlador tem um método que recebe tal parâmetro (neste caso o controlador *ManageRoomController* tem o método *updateRooms* que tem como parâmetro *class*), a passagem da informação deve ser feita via chamada do método. Este fluxo não é proposto pelo *FrameWeb*, sendo, portanto, uma sugestão de extensão para o mesmo.

Outro importante detalhe é que o método *createPDF*, como pode ser visto na imagem, realiza a exportação de alocações baseadas em um período (caso selecionado), mas não recebe nenhum parâmetro. Isso acontece porque o *nemo-utils* possui sua própria maneira de implementar os filtros (como pode ser visto na herança existente na Figura 16). Dentro de cada controlador que herda de *CrudController*, existe um atributo *filterParam*. Este atributo é usado dentro da função `createPDF` com o objetivo de retornar (caso selecionado) o período que será utilizado na exportação. O fluxo *Consultar Alocação* do caso de uso *Gerenciar Alocação* corresponde exatamente a esta filtragem por período, portanto também não é exibido no modelo.

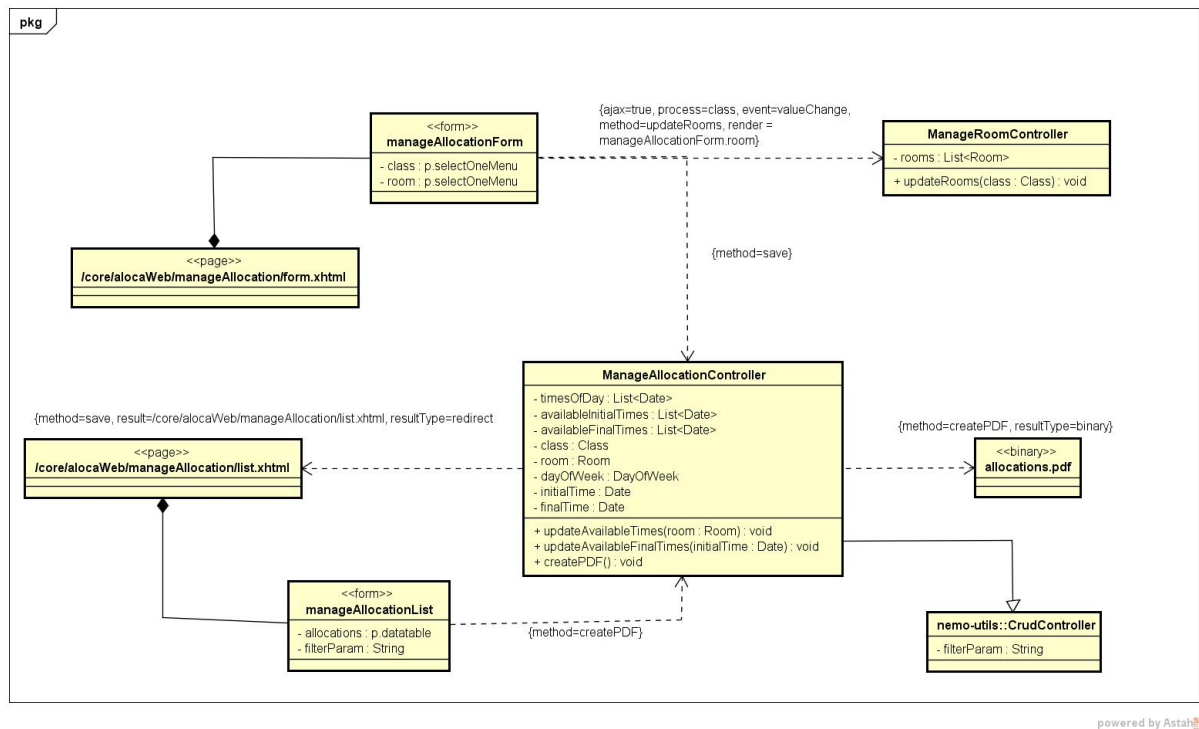


Figura 16 – Marvin - AlocaWeb - Efetuar Alocação e Exportar Alocação - Modelo Navegação.

O modelo apresentado na Figura 16 é um pouco mais complexo, pois faz uso de *AJAX* para popular os valores dos campos do formulário. Podemos ver que `manageAllocationForm` possui o estereótipo `<<form>>`, e que praticamente todos os métodos disparados a partir dele são do tipo *AJAX*. A ideia do fluxo acima é que o sistema diminua as opções de seleção do usuário de acordo com o que ele seleciona, ou seja: o usuário seleciona uma turma, o sistema então dispara o método `updateRooms` do controlador `ManageRoomController`. Este método irá buscar todas as salas de aula com capacidade maior ou igual à quantidade de vagas ofertadas na turma (*class*) selecionada, e irá renderizar o campo `room` existente em `manageAllocationForm`.

Um outro fluxo importante neste modelo é o fluxo de exportar alocação. O form `manageAllocationList` é responsável por tal. O usuário irá selecionar um período (como foi explicado anteriormente, o atributo `filterParam` do `nemo-utils` é responsável por isso), e irá exportar a alocação. O método `createPDF` irá, então, gerar um arquivo `allocations.pdf` contendo todas as alocações para o período selecionado.

Por fim, a Figura 17 mostra o modelo de navegação para o fluxo *Enviar Currículo* do caso de uso *Enviar Currículo Lattes* e o fluxo *Exportar Currículo* do caso de uso *Exportar Currículo*.

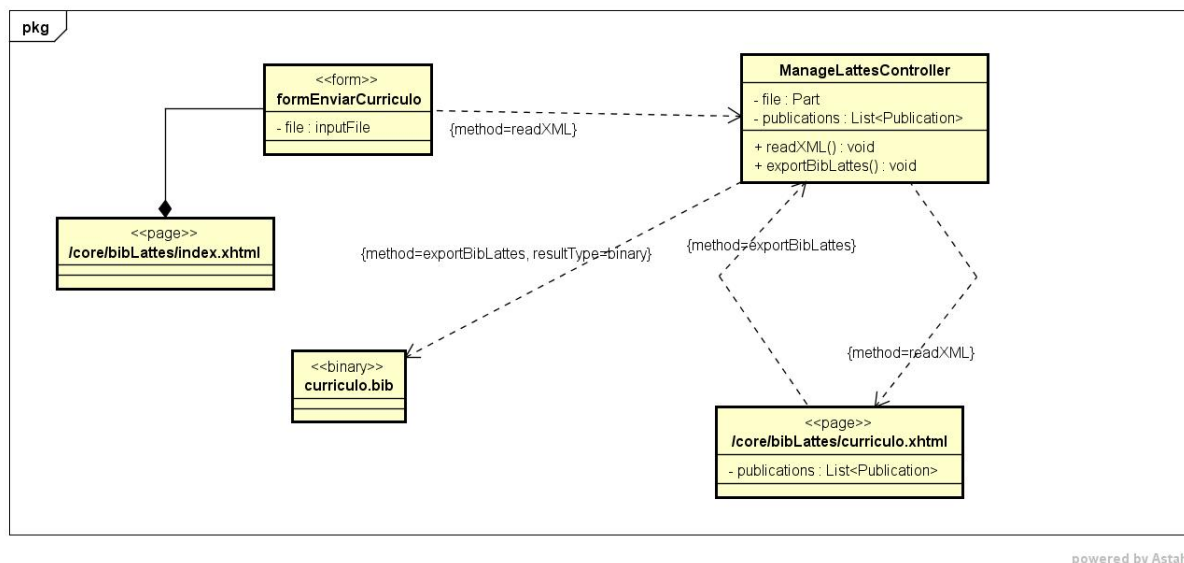


Figura 17 – Marvin - BibLattes - Enviar Currículo e Exportar Currículo - Modelo Navegação.

No modelo acima temos o form `formEnviarCurrículo` com atributo `file` do tipo *inputFile*. Isso significa que o campo do formulário é um campo para *upload* de arquivo. Ao efetuar o *upload*, o form dispara o método `readXML` do controlador `ManageLattesController`. Este método irá ler o arquivo XML contendo o currículo Lattes e popular a lista de publicações (*publications*). Ao final, a página `currículo.xhtml` será renderizada, contendo a listagem de todas as publicações que foram enviadas.

Além disso, o usuário poderá também exportar a lista de publicações no formato BibTex. Ao realizar tal exportação, a página `currículo.xhtml` chama o método `exportBibLattes`. Este método irá gerar o arquivo `currículo.bib` com todas as publicações.

O próximo modelo FrameWeb a ser mostrado é o **Modelo de Aplicação**, que consiste em um diagrama de classes da UML que representa as classes de serviço, responsáveis pela codificação dos casos de uso, e suas dependências. Esse diagrama é utilizado para guiar a implementação das classes do pacote **Aplicação** e a configuração das dependências entre os pacotes **Controle**, **Aplicação** e **Persistência**, ou seja, quais classes de ação dependem de quais classes de serviço e quais DAOs são necessários para que as classes de serviço alcancem seus objetivos (SOUZA, 2007).

Todas as classes de aplicação que são de casos de uso cadastrais estendem de `CrudServiceBean` do pacote *nemo-utils*. A Figura 18 apresenta o modelo de aplicação do *nemo-utils*.

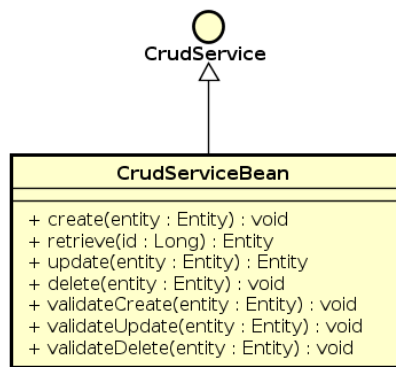


Figura 18 – FrameWeb - *nemo-utils* - Modelo Aplicação (LIMA, 2015).

A Figura 19 mostra o modelo de aplicação para o módulo *core*. A Figura 20 mostra o modelo de aplicação para o módulo *AlocaWeb* e, por último, a Figura 21 mostra o modelo de aplicação para o módulo *BibLattes*.

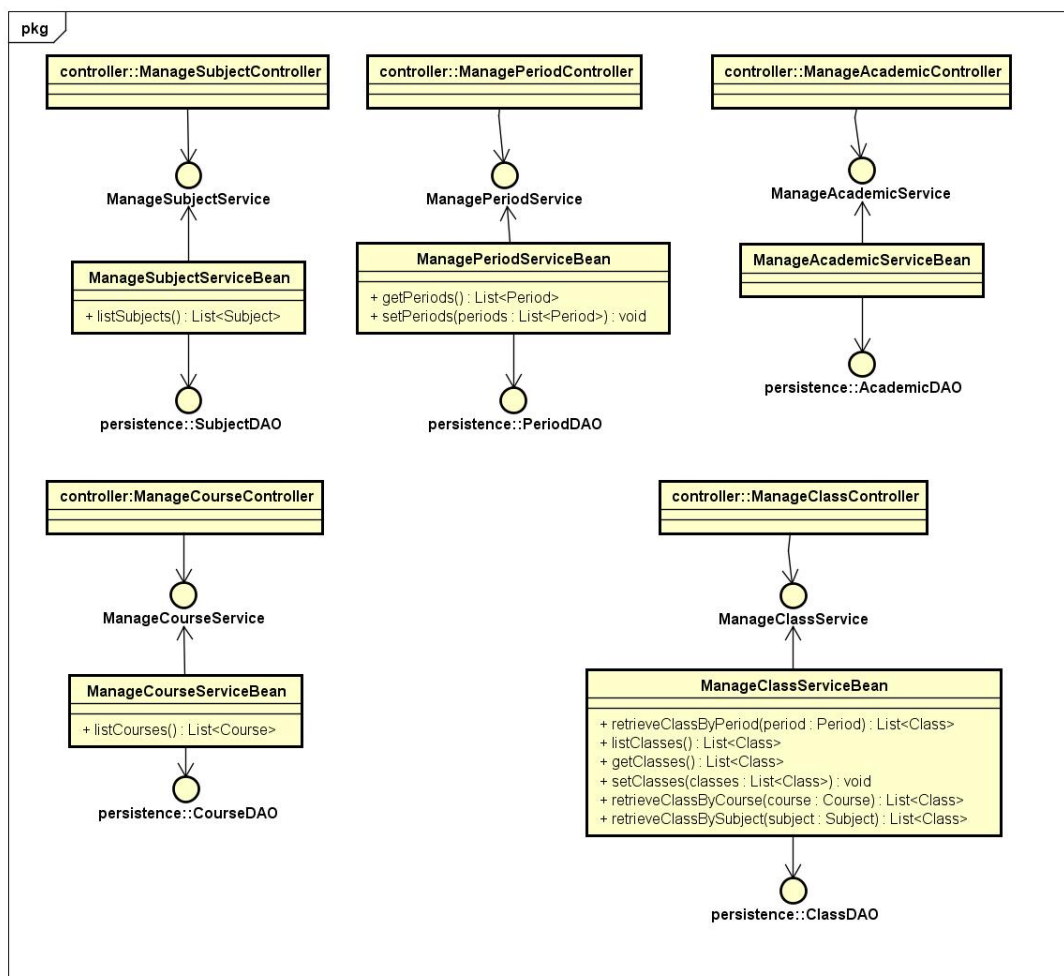
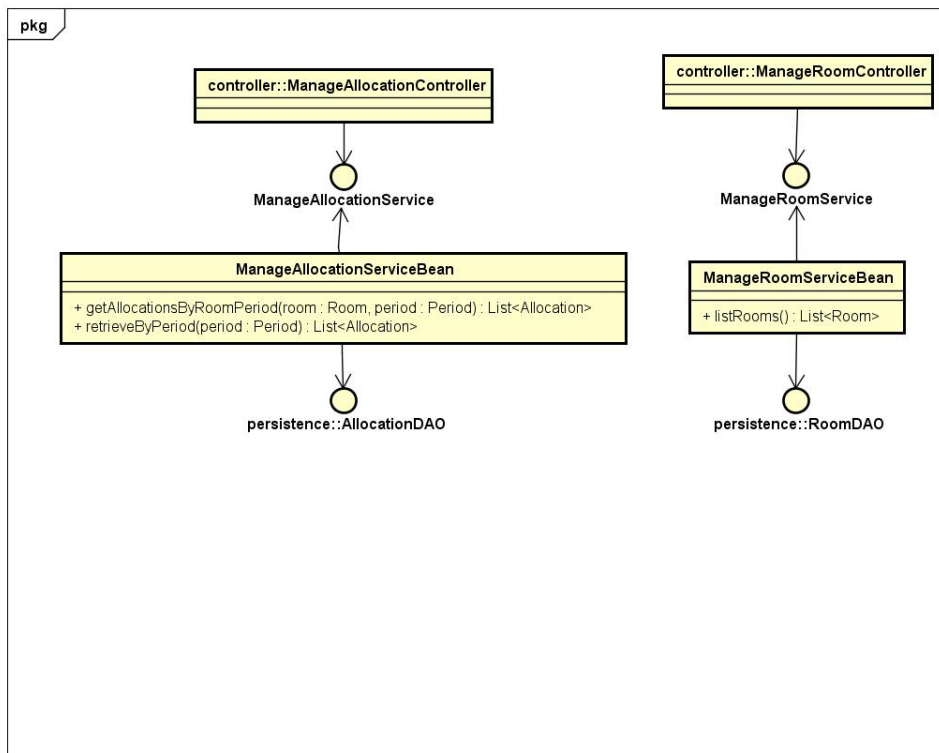
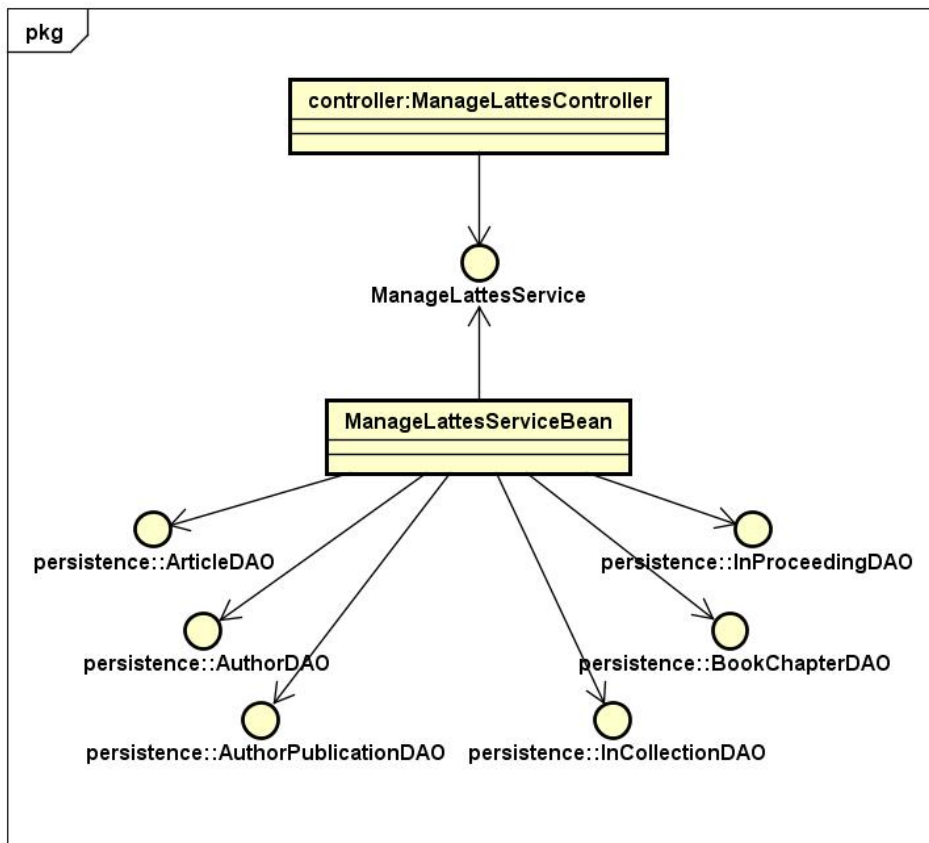


Figura 19 – FrameWeb - Marvin - Core - Modelo Aplicação.



powered by Astah

Figura 20 – FrameWeb - Marvin - AlocaWeb - Modelo Aplicação.



powered by Astah

Figura 21 – FrameWeb - Marvin - BibLattes - Modelo Aplicação.

Outro importante padrão apresentado e sugerido pelo FrameWeb é o DAO. Tal padrão de desenvolvimento é baseado na existência de uma camada de acesso a dados, fazendo com que o controlador não tenha acesso direto ao banco de dados. O **Modelo de Persistência** é um diagrama de classes da UML que representa as classes DAO existentes, responsáveis pela persistência das instâncias das classes de domínio. Esse diagrama é utilizado para auxiliar no processo de desenvolvimento do pacote de persistência (onde os DAOs ficam localizados).

Existem diversas funcionalidades que são comuns a qualquer classe (ou objeto) que possa ser salvo em um banco de dados, são elas: retornar todos os objetos, deletar um objeto, salvar um objeto, atualizar um objeto, retornar um objeto dado um id, etc. Com o objetivo de evitar a repetição destas funcionalidades em cada DAO, o *nemo-utils* nos provê um DAO base que implementa as principais operações utilizadas em um banco de dados. Além disso, a utilização de interfaces na criação dos DAOs é importante para evitar que qualquer modificação tecnológica (API de persistência, por exemplo), influencie drasticamente em uma aplicação. A Figura 22 exibe as classes bases do *nemo-utils*.

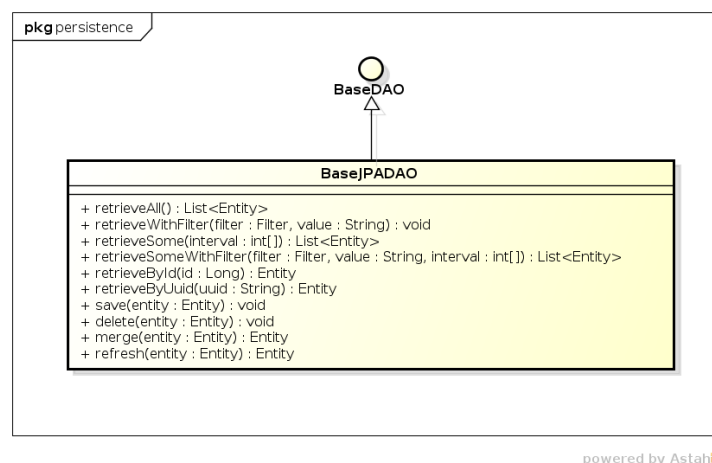


Figura 22 – FrameWeb - *nemo-utils* - Modelo Persistência (LIMA, 2015).

Com o objetivo de permitir que qualquer classe herde as funcionalidades existentes em seus DAOs base, o *nemo-utils* declara tanto a interface **BaseDAO** como a classe **BaseJPADAO** usando tipos genéricos, fazendo com que as classes e interfaces que herdem deles tenham que especificar qual a classe que será gerenciada pelos mesmos. Também não será necessário exibir os métodos do DAO na implementação e na interface, basta modelá-los em apenas um dos dois. No caso do DAO Base, subentende-se que todos os métodos públicos de BaseJPADAO são definidos na interface BaseDAO.

Segundo os padrões estabelecidos por FrameWeb, todas as interfaces DAO são subinterfaces de BaseDAO, enquanto todas as implementações JPA são subclasses de BaseJPADAO, herdando todos os métodos básicos, por exemplo: `retrieveAll()`, `save()`, `delete()`, `retrieveById()`. Os demais métodos que foram declarados no diagrama se

referem a consultas específicas que devem ser disponibilizadas para o funcionamento de determinados casos de uso.

As Figuras 23, 24 e 25 são os modelos de persistência para os módulos Core, AlocaWeb e BibLattes, respectivamente. É importante notar aqui que existem alguns modelos que possuem operações específicas (que foram implementadas neste trabalho), como por exemplo `CourseJPADAO` que possui a operação `retrieveCourseByNameCode`. Este método busca um curso dado o nome e o código do mesmo. Ele teve de ser implementado pois o *framework* `nemo-utils` não possui essa operação, visto ser ela necessária para atender um determinado caso de uso específico do domínio de nossa aplicação.

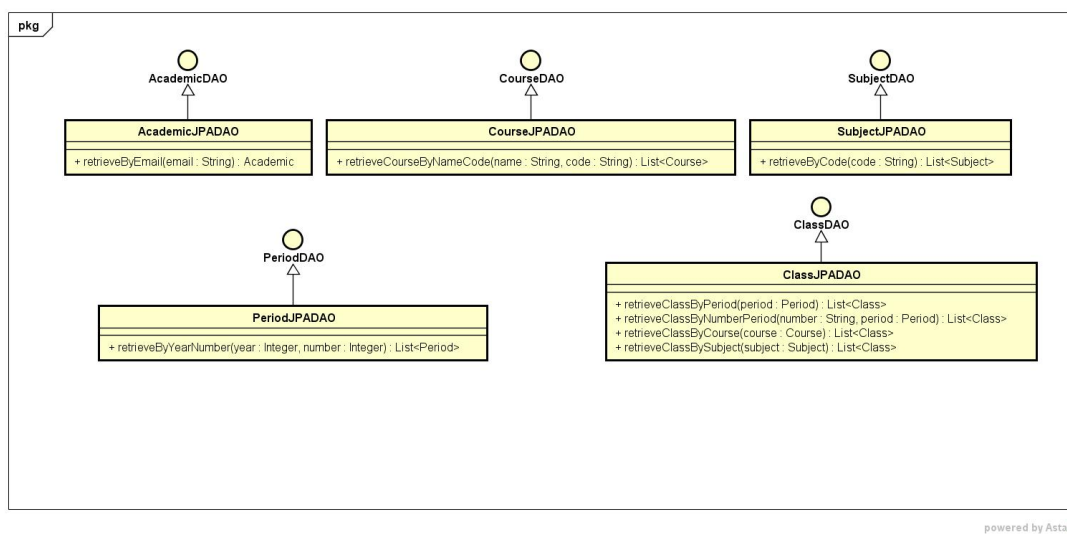


Figura 23 – FrameWeb - Marvin - Core - Modelo Persistência.

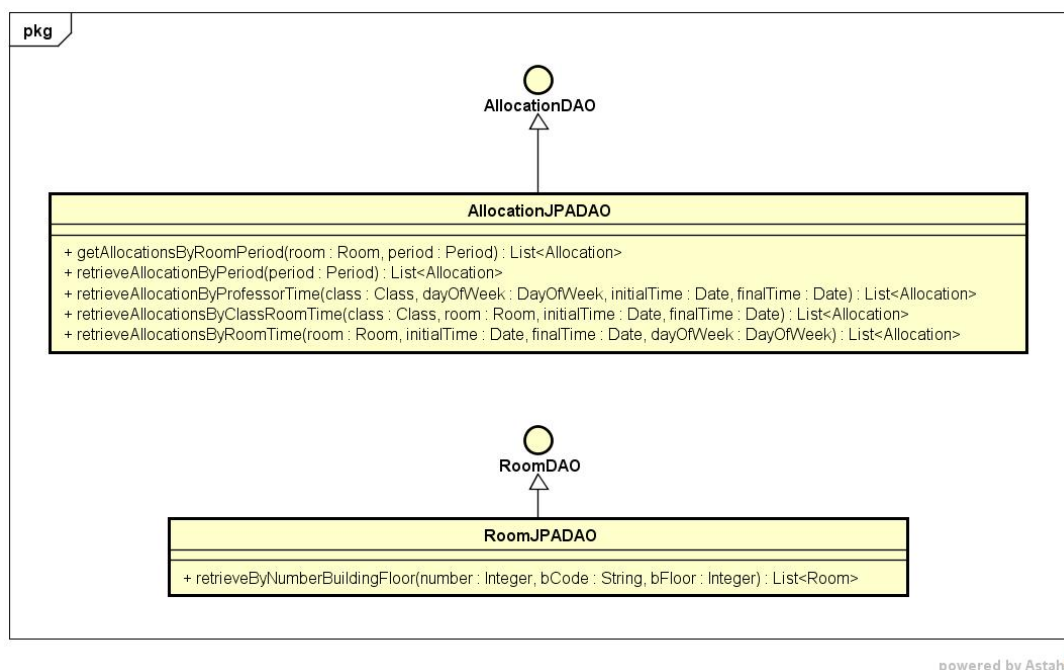
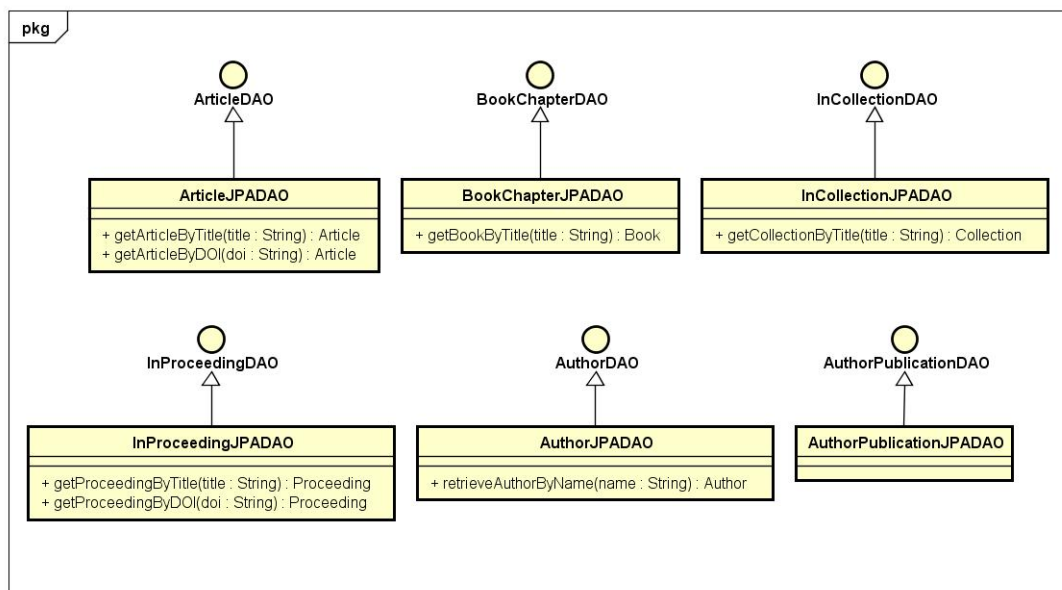


Figura 24 – FrameWeb - Marvin - AlocaWeb - Modelo Persistência.



powered by Astah

Figura 25 – FrameWeb - Marvin - AlocaWeb - Modelo Persistência.

Como é possível perceber, o Modelo de Persistência não define nenhuma extensão da UML para representar os conceitos necessários da camada de acesso a dados, mas apenas regras que tornam essa modelagem mais simples e rápida, por meio da definição de padrões.

4.4 Apresentação do Sistema

Nesta seção, apresentamos o sistema por meio de uma série de capturas de tela, a Figura 26 mostra a tela inicial de login no sistema. As telas de *Login* e *Academics* não foram implementadas neste trabalho, já estavam prontas quando iniciamos a implementação dos módulos AlocaWeb e BibLattes. Elas serão mostradas apenas para que o fluxo não fique sem sentido.

O login utilizará email e senha. O campo do e-mail possui validação para verificar se o mesmo é válido. O campo da senha aceita qualquer caractere alfanumérico e possui tamanho máximo 15.

Do lado esquerdo da tela é possível ver um menu. Inicialmente este menu possui apenas a funcionalidade Login. Depois de efetuado o mesmo, o usuário poderá ver todas as funcionalidades implementadas e que já existiam no sistema Marvin, como pode ser visto na Figura 27.

O sistema possui um layout responsivo, por este motivo, a visualização do menu irá ser diferente em um celular e em um *Desktop*, por exemplo. A Figura 28 exibe como fica o menu em um celular. Ao contrário da aplicação Desktop, que o menu está sempre

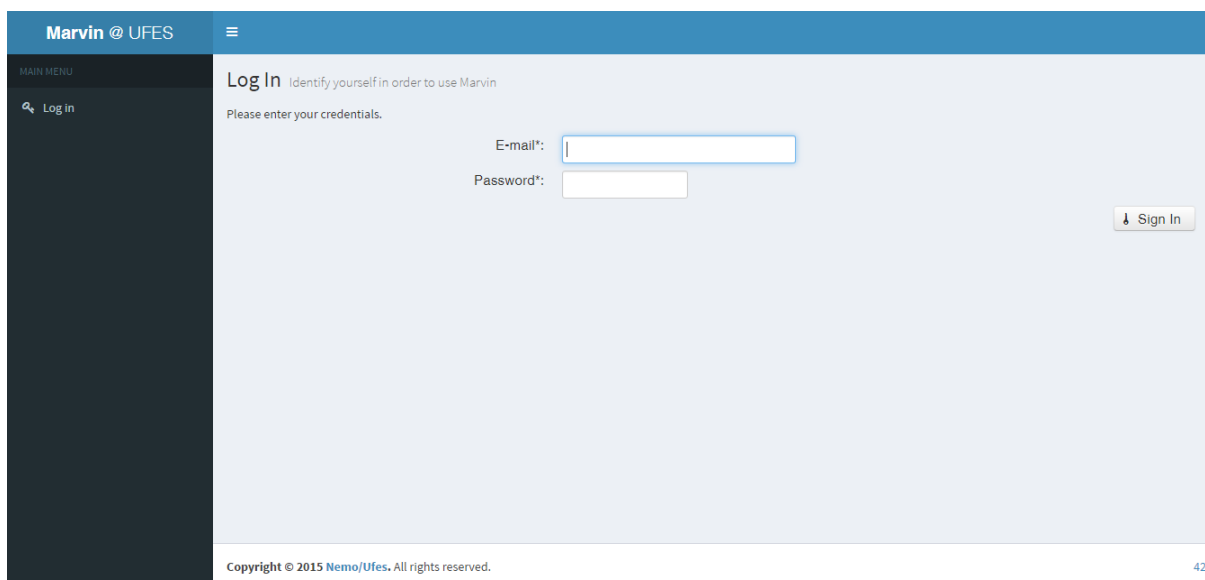


Figura 26 – Marvin - Tela Login.

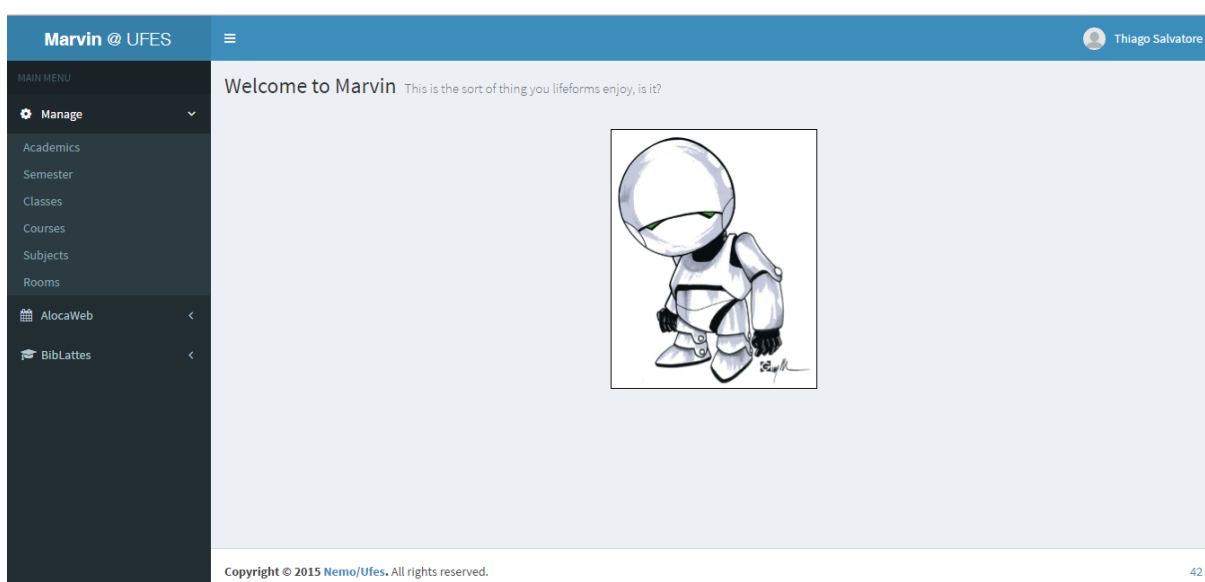


Figura 27 – Marvin - Página Inicial.

visível, no celular o usuário deve clicar no botão com três traços para ativar o mesmo. Nos próximos passos serão mostradas outras telas em um dispositivo móvel.

As funcionalidades do tipo CRUD possuem telas que seguem exatamente o mesmo padrão e fluxo. Para evitar repetição, apenas uma das telas será mostrada, exibindo as funcionalidades de listagem, cadastro, leitura, atualização e exclusão de uma turma. A Figura 29 exibe a tela que lista os cursos cadastrados no sistema.

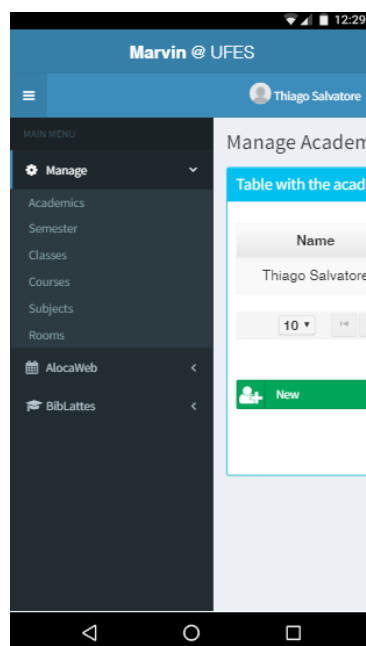


Figura 28 – Marvin - Menu Dispositivo Móvel.

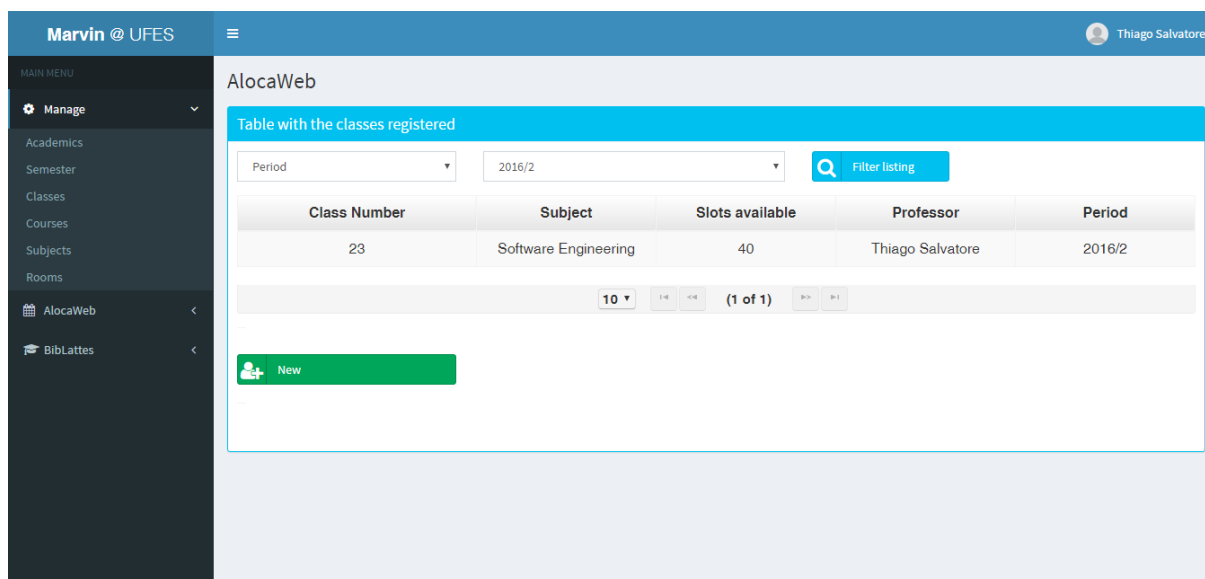
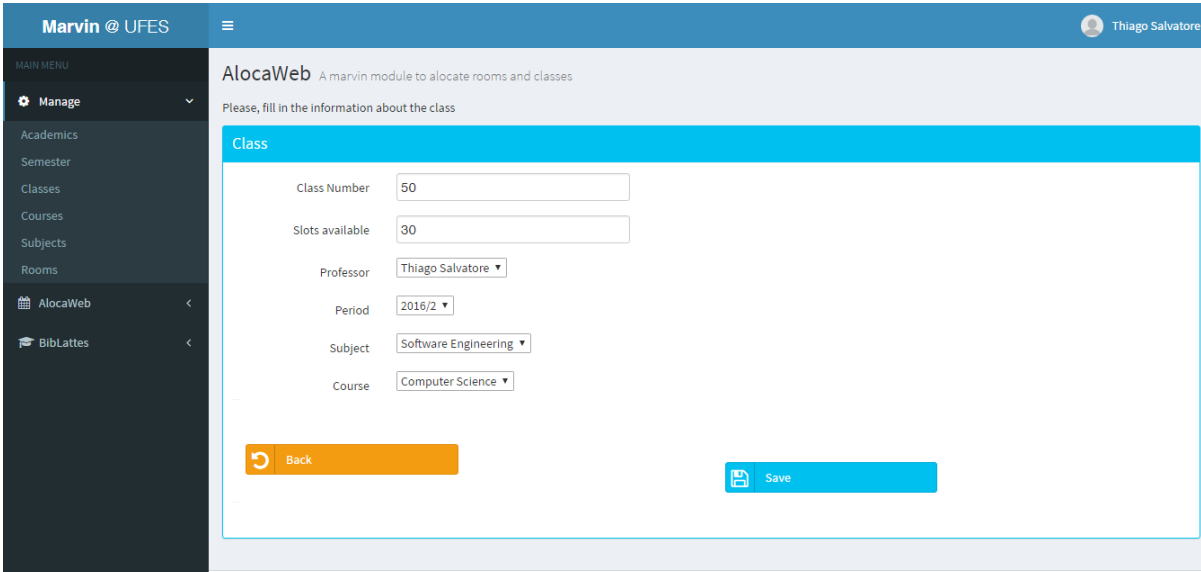


Figura 29 – Marvin - Lista de Turmas.

Ao selecionar um item na listagem, por exemplo, a turma de *Software Engineering*, uma lista de funcionalidades aparece, além da funcionalidade **New** que já estava presente, a saber: *View*, *Modify* e *Delete*. Ao clicar nos botões **New** (novo), **View** (visualizar) ou **Modify** (alterar), o usuário será redirecionado para a página de cadastro de um item conforme a Figura 30. É importante notar que ao clicar em *View*, os campos do formulário serão somente leitura. Ao clicar em *Modify* os campos do formulário já virão preenchidos com as informações da turma selecionada, sendo possível editar as mesmas. E por último, ao clicar em *New*, o formulário será exibido em branco, possibilitando a criação de uma nova turma.



The screenshot displays the AlocaWeb interface. At the top, the user is identified as 'Marvin @ UFES' and 'Thiago Salvatore'. The main menu on the left includes 'Manage', 'Academics', 'Semester', 'Classes', 'Courses', 'Subjects', 'Rooms', 'AlocaWeb', and 'BibLattes'. The main content area is titled 'AlocaWeb - A marvin module to allocate rooms and classes' and contains a form for creating a class. The form fields are: Class Number (50), Slots available (30), Professor (Thiago Salvatore), Period (2016/2), Subject (Software Engineering), and Course (Computer Science). There are 'Back' and 'Save' buttons at the bottom of the form.

Figura 30 – Marvin - Tela Cadastro de Turma.

A funcionalidade *Delete* possibilita a exclusão de um item (neste caso, de uma turma). Ao clicar na mesma, um novo painel será exibido, para confirmação da exclusão. O usuário poderá, então, confirmar a mesma, excluindo o item, ou cancelar. A Figura 31 exhibe tal comportamento para o CRUD *Semester*.

A Figura 32 apresenta a tela de listagem de períodos para quem está utilizando um dispositivo móvel. Ao contrário do Desktop, o menu, como dito anteriormente, não fica visível, e os botões de ação estão agora em posição vertical (não mais horizontal).

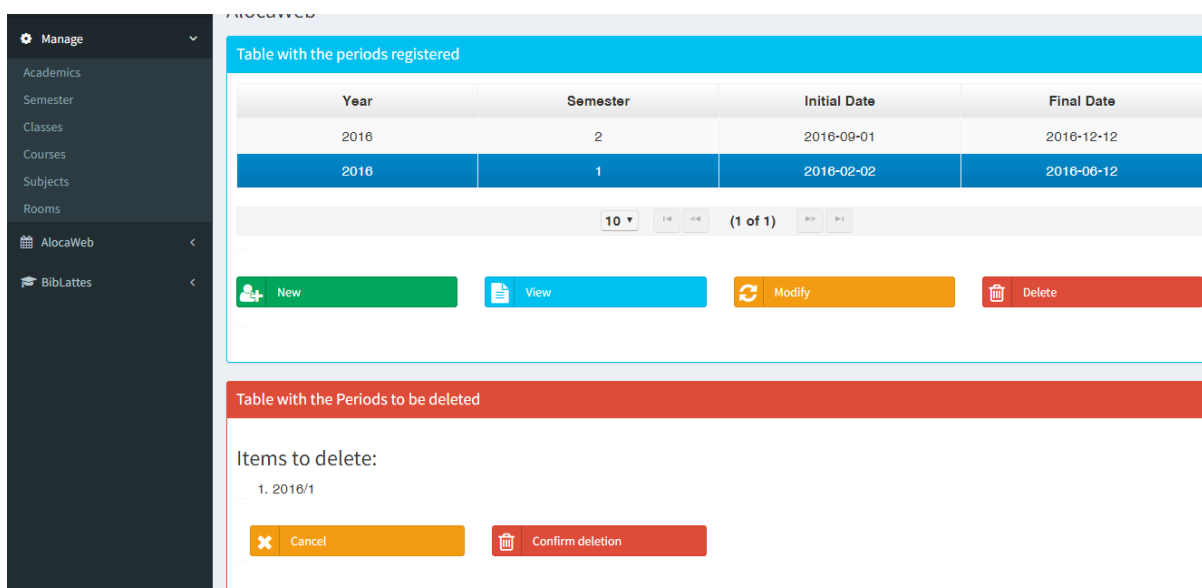


Figura 31 – Marvin - Tela exclusão de Período.

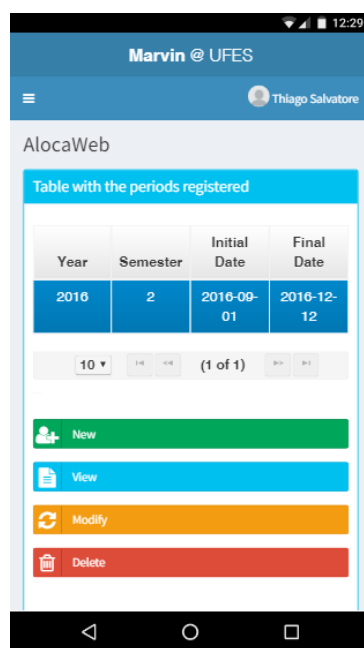


Figura 32 – Marvin - Tela Listagem Períodos Mobile.

A Figura 33 exibe a tela relacionada ao caso de uso **Gerenciar Alocações**. Esta tela é bastante semelhante às outras, com a diferença de que é possível filtrar os resultados por período. As funcionalidades *New*, *View*, *Modify* e *Delete* não serão exibidas novamente.

A Figura 34 mostra a tela responsável por implementar o caso de uso **Exportar Alocação**. Nesta tela, o usuário irá selecionar um período e clicar em exportar. O sistema irá, então, gerar um arquivo PDF contendo todas as alocações para o período selecionado. A Figura 35 exibe um modelo deste documento PDF.

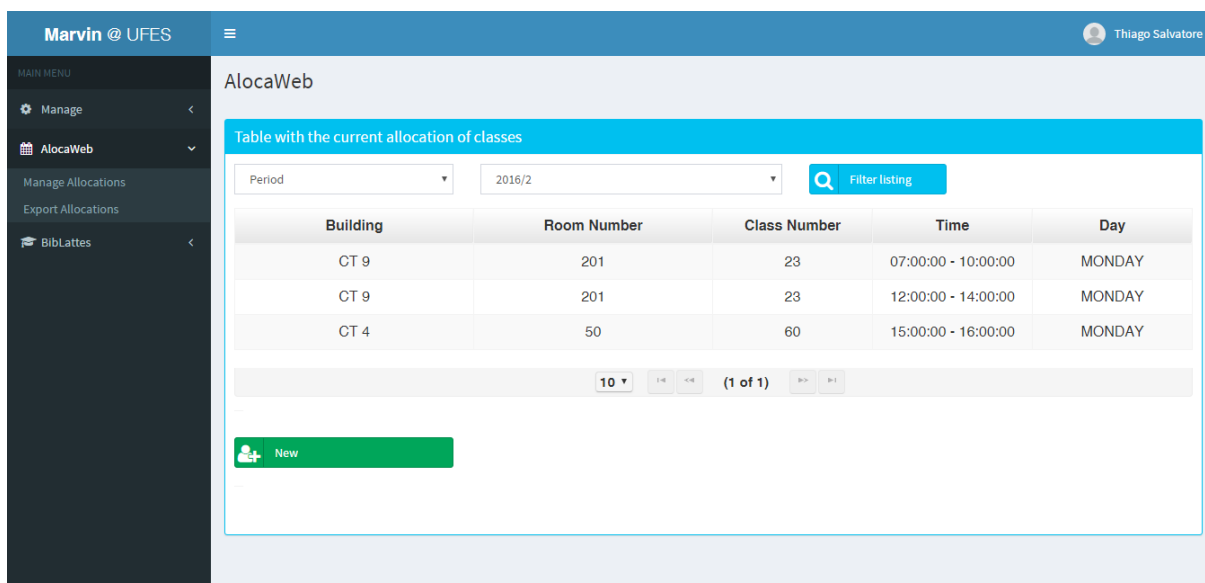


Figura 33 – AlocaWeb - Gerenciar Alocações.

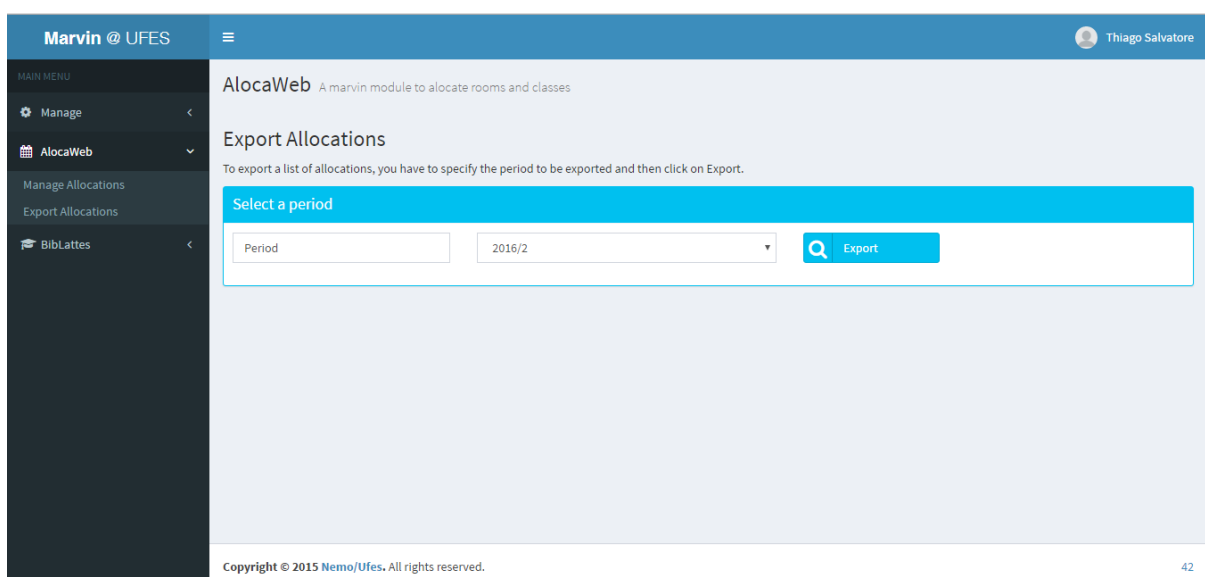


Figura 34 – AlocaWeb - Exportar Alocações.

Outra importante funcionalidade do sistema está relacionada ao BibLattes. A Figura 36 mostra a tela para importação de um currículo lattes. Nesta tela é possível ver o botão *Choose* e *Import*. O primeiro permite ao usuário selecionar o currículo que será enviado, e o segundo executa a ação de importar o currículo. Ao final do processamento, o usuário é automaticamente redirecionado para a tela de gerenciamento de publicações (que pode também ser acessada através do menu BibLattes > Manage Publications).

A Figura 37 exhibe a tela para visualização de publicações. Esta tela possui uma das principais funcionalidades do módulo *BibLattes* que é a exportação de publicações para o formato *BibTeX*. Para executar tal ação, o usuário precisa apenas clicar em *Export to BibTeX*.

Room allocation for 2016/2						
Course Name	Day of Week	Time	Building	Room #	Subject	Professor
Computer Science	MONDAY	15:00-16:00	CT 4	50	Fenomenos de Transporte	Thiago Salvatore
Computer Science	MONDAY	07:00-10:00	CT 9	201	Software Engineering	Thiago Salvatore
Computer Science	MONDAY	12:00-14:00	CT 9	201	Software Engineering	Thiago Salvatore

Figura 35 – AlocaWeb - Relatório de Alocações.

Figura 36 – BibLattes - Tela Importação Currículo Lattes.

A Figura 38 exibe a tela de gerenciamento de publicações em um dispositivo mobile. É possível perceber que apenas a coluna *Title* é exibida. Como o dispositivo possui uma tela menor, é melhor mostrar ao usuário apenas a informação mais importante, neste caso, o título da publicação.

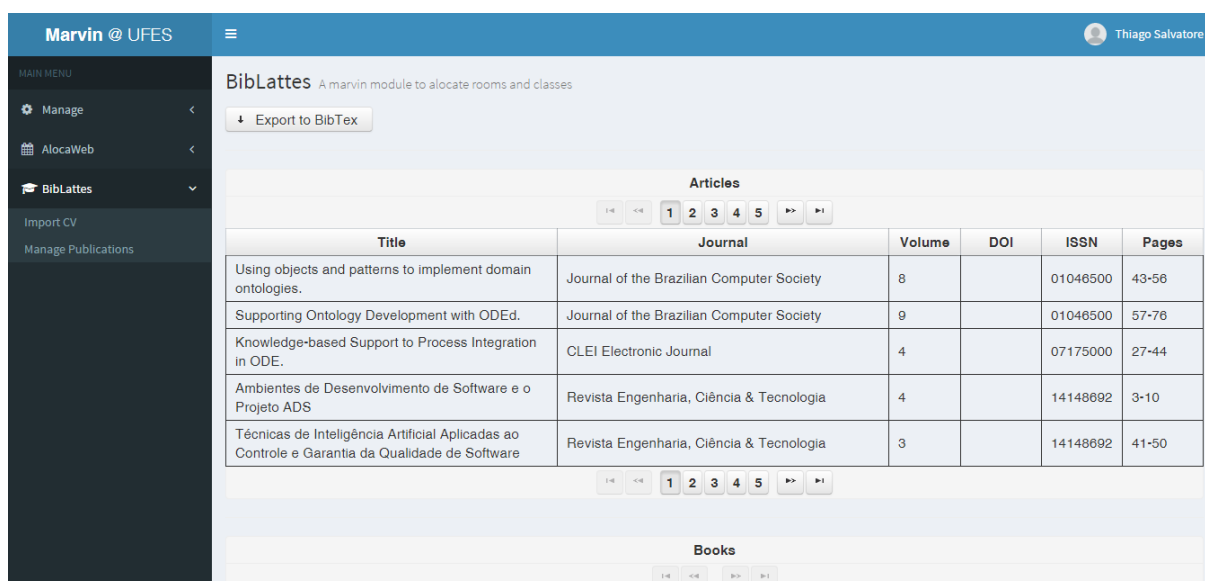


Figura 37 – BibLattes - Tela Gerenciamento de Publicações.



Figura 38 – BibLattes - Tela Gerenciamento de Publicações Mobile.

5 Considerações Finais

Este capítulo apresenta as conclusões do trabalho realizado, mostrando suas contribuições. Por fim, são apresentadas suas limitações e perspectivas de trabalhos futuros.

5.1 Conclusões

Dada a importância da alocação de salas de aula e do controle de publicações no contexto acadêmico, e visto que tais sistemas não existiam dentro do DI/UFES, surgiu a necessidade da criação de sistemas web capazes de atender a essas necessidades. Além disso, como muitos estudantes do DI/UFES desenvolvem ferramentas como parte de seu projeto final de graduação, viu-se a necessidade de integrar essas ferramentas de forma a serem realmente usadas.

Todos os objetivos levantados no Capítulo 1 foram atingidos, sendo que toda a documentação proposta foi produzida de acordo com os padrões de Engenharia de Software. Foram levantados os requisitos, em seguida foi feita a análise de tais requisitos, produzindo, então, o Documento de Especificação de Requisitos. Este documento contém informações sobre requisitos funcionais, não funcionais, regras de negócio, descrição do sistema, definição de atores, casos de uso e diagramas de classe. Com o documento de requisitos finalizado, foi criado o Documento de Projeto, contendo todas as informações relacionadas à arquitetura do sistema e foram criados os modelos propostos na Seção 2.3, seguindo a abordagem FrameWeb.

Uma das maiores dificuldades encontradas foi adaptar o conhecimento existente em desenvolvimento web utilizando linguagens como JavaScript e implementar um sistema utilizando Java EE e suas tecnologias (JSF, CDI, JPA). Para resolver esse problema, foi feita a leitura de diversos tutoriais e artigos envolvendo desenvolvimento web utilizando Java. Além disso, o *FrameWeb* não é um conceito fácil de ser consolidado em tão pouco tempo, principalmente no que diz respeito à construção dos modelos.

Apesar da dificuldade, o *FrameWeb* foi um método bastante útil, dado que sua modelagem é bem próxima de como o sistema irá se comportar na *web*. O uso de modelos da UML facilitou o desenvolvimento e entendimento dos mesmos.

Por fim, é importante citar como as disciplinas cursadas durante o curso ajudaram no desenvolvimento deste trabalho, Engenharia de Software, Projeto de Sistemas, Programações, Lógica, dentre outras. Uma das coisas mais difíceis no final de um curso é conseguir utilizar todos os conhecimentos adquiridos em 5 anos e colocá-los em prática. Este trabalho proporcionou isso, sendo possível colocar em prática boa parte das disciplinas

estudadas durante o curso.

5.2 Limitações e Perspectivas Futuras

Assim como qualquer software, é sempre possível melhorar alguns detalhes e adicionar novas funcionalidades. O ciclo de desenvolvimento nunca acaba, é importante que haja uma manutenção constante, que seja capaz de identificar problemas, verificar quais tecnologias não estão mais sendo utilizadas, mantendo, assim, o sistema o mais atualizado possível.

A partir dos resultados alcançados, algumas limitações podem ser observadas, o que dá margem para a realização de trabalhos futuros. Sendo assim, alguns trabalhos surgirão a partir deste. Essas limitações são apresentadas nos itens abaixo.

- Especificar no *FrameWeb* as maneiras sugeridas para a representação de requisições AJAX;
- Integrar o AlocaWeb e o BibLattes com a nova versão do Marvin (que está utilizando o framework JButler¹, sucessor do nemo-utils);
- Adicionar as funcionalidades de controle de permissão sugeridas por Manzoli (2016);
- Adicionar a funcionalidade de alocação esporádica (utilização de uma sala de aula em apenas um dia específico, e não durante todo o período), proposta inicialmente mas que não foi implementada;
- Melhorar a maneira como é gerado o relatório de alocações - Agrupar as salas e separar as alocações por dia da semana.

¹ <<https://github.com/dwws-ufes/jbutler/>>

Referências

- ALUR, D.; MALKS, D.; CRUPI, J. *Core J2EE Patterns: Best Practices and Design Strategies*. 2. ed.. ed. [S.l.]: Prentice Hall, 2003. ISBN 0131422464. Citado 2 vezes nas páginas 19 e 34.
- FALBO, R. A. *Projeto de Sistemas*. [s.n.], 2011. 68 p. Disponível em: <http://www.inf.ufes.br/~falbo/files/Notas_Aula_Projeto_Sistemas_2.pdf>. Citado na página 16.
- FALBO, R. A. *Engenharia de Requisitos*. [s.n.], 2012. 179 p. Disponível em: <http://www.inf.ufes.br/~falbo/files/Notas_Aula_Engenharia_Requisitos.pdf>. Citado 3 vezes nas páginas 14, 15 e 16.
- FALBO, R. A. *Engenharia de Software*. [s.n.], 2014. 141 p. Disponível em: <http://www.inf.ufes.br/~falbo/files/Notas_Aula_Engenharia_Software.pdf>. Citado 2 vezes nas páginas 14 e 16.
- FARIA, T. *Java EE 7 com JSF, PrimeFaces e CDI*. [S.l.: s.n.], 2013. Citado na página 17.
- FOWLER, M. *Patterns of Enterprise Application Architecture*. 2. ed.. ed. [S.l.]: Addison-Wesley, 2002. Citado 2 vezes nas páginas 16 e 18.
- LIMA, L. V. F. *SAP - Sistema de Apoio ao Professor*. [S.l.], 2015. Citado 5 vezes nas páginas 6, 14, 32, 44 e 46.
- MANZOLI, B. *SAE - Sistema de Acompanhamento de Egressos*. [S.l.], 2016. Citado 3 vezes nas páginas 18, 27 e 57.
- MARTINS, B. F. S. *Uma abordagem dirigida a modelos para o projeto de Sistemas de Informação Web com base no método FrameWeb*. [S.l.], 2016. Citado na página 18.
- OLSINA, L.; LAFUENTE, G.; ROSSI, G. Specifying quality characteristics and attributes for websites. In: _____. *Web Engineering: Managing Diversity and Complexity of Web Application Development*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 266–278. ISBN 978-3-540-45144-0. Disponível em: <http://dx.doi.org/10.1007/3-540-45144-7_26>. Citado na página 17.
- PRESSMAN, R. S. *Engenharia de Software - Uma abordagem profissional*. 7ª edição. ed. [S.l.]: McGraw-Hill, 2011. ISBN 9788563308337. Citado na página 17.
- SOMMERVILLE, I.; KOTONYA, G. *Requirements engineering: processes and techniques*. [S.l.]: John Wiley & Sons, Inc., 1998. Citado na página 15.
- SOMMERVILLE, I. et al. *Engenharia de software*. [S.l.]: Addison Wesley São Paulo, 2003. v. 6. Citado na página 15.
- SOUZA, V. E. S. *FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web*. 2007. Disponível em: <http://nemo.inf.ufes.br/wp-content/papercite-data/pdf/frameweb__um_metodo_baseado_em_frameworks_para_o_

[projeto_de_sistemas_de_informacao_web_2007.pdf](#)>. Citado 4 vezes nas páginas 12, 18, 19 e 43.

Apêndices



Documento de Especificação de Requisitos

AlocaWeb e BibLattes – Módulos do sistema Marvin

Registro de Alterações:

Versão	Responsável	Data	Alterações
1.0	Thiago Rocha Salvatore	10/03/2016	<i>versão inicial</i>
1.1	Vítor E. Silva Souza	14/03/2016	<i>primeira revisão</i>
1.2	Thiago Rocha Salvatore	20/03/2016	<i>versão corrigida</i>
1.3	Vítor E. Silva Souza	25/03/2016	<i>segunda revisão</i>
1.4	Thiago Rocha Salvatore	23/06/2016	<i>versão corrigida</i>
1.5	Vítor E. Silva Souza	27/06/2016	<i>terceira revisão</i>
1.6	Thiago Rocha Salvatore	04/07/2016	<i>versão corrigida e junção documentos</i>
1.7	Vítor E. Silva Souza	15/07/2016	<i>quarta revisão</i>
2.0	Thiago Rocha Salvatore	18/07/2016	<i>versão final corrigida</i>

Vitória, ES

2016

1 Introdução

Este documento apresenta os requisitos de usuário e a análise dos requisitos do sistema AlocaWeb e BibLattes – Módulos do sistema Marvin. A atividade de análise de requisitos foi conduzida aplicando-se técnicas de modelagem de casos de uso, modelagem de classes e modelagem de comportamento dinâmico do sistema. Os modelos apresentados foram elaborados usando a UML.

Este documento está organizado da seguinte forma: a Seção 2 contém uma descrição do propósito do sistema; a Seção 3 apresenta uma descrição do minimundo apresentando o problema; a Seção 4 apresenta as listas de requisitos de usuário levantados junto ao cliente; a Seção 5 apresenta os subsistemas identificados, mostrando suas dependências na forma de um diagrama de pacotes; a Seção 6 apresenta o modelo de casos de uso, incluindo descrições de atores, os diagramas de casos de uso e descrições de casos de uso; a Seção 7 apresenta o modelo conceitual estrutural do sistema, na forma de diagramas de classes; finalmente, a Seção 8 apresenta o dicionário do projeto, contendo as definições das classes identificadas.

2 Descrição do Propósito do Sistema

O Departamento de Informática da Universidade Federal do Espírito Santo (DI/Ufes) necessita de um sistema de informação para o controle de alocação de salas de aula e ofertas de disciplinas. O propósito deste sistema é facilitar a alocação de disciplinas (com seus respectivos horários, turmas e quantidade de vagas) às salas de aula disponíveis no centro, possibilitando assim uma melhor utilização do espaço, uma redução no tempo gasto para realizar tal tarefa e, ainda, a construção de um horário que facilite a vida dos estudantes.

Além disso, é importante no contexto acadêmico a existência de um sistema capaz de extrair informações relacionadas às publicações. O propósito desta ferramenta é, dado um currículo Lattes no formato *xml* (obtido no site da Plataforma Lattes), extrair todas as informações relevantes do mesmo, permitir ao usuário visualizar tais informações de uma maneira mais limpa e exportá-las em formatos adequados, para que sejam usadas,

por exemplo, em suas páginas pessoais na Web.

3 Descrição do Minimundo

Marvin é um sistema de informação que agrega ferramentas úteis para atividades de ensino e pesquisa em uma universidade. Com o objetivo de tornar o Marvin ainda mais completo, dois novo sub-sistemas devem ser adicionados ao mesmo, a saber: Controle de Alocação de Salas de Aula (AlocaWeb) e Extração de Dados de um CV Lattes (BibLattes).

3.1 AlocaWeb – Controle de Alocação de Salas de Aula

O Departamento de Informática da Universidade Federal do Espírito Santo (DI/Ufes) necessita de um sistema de informação para o controle de alocação de salas de aula e ofertas de disciplinas. O propósito deste sistema é facilitar a alocação de disciplinas (com seus respectivos horários, turmas e quantidade de vagas) às salas de aula disponíveis no centro, possibilitando assim uma melhor utilização do espaço, uma redução no tempo gasto para realizar tal tarefa e, ainda, a construção de um horário que facilite a vida dos estudantes.

Para isso, o usuário forneceria dados como período em que disciplinas serão ofertadas, professor responsável por cada disciplina, curso ao qual ela está relacionada, quantidade de vagas existentes, salas de aula disponíveis e espaço disponível em cada uma das salas de aula.

Ao abrir o sistema, o usuário será redirecionado para a página inicial, onde poderá escolher o que deseja fazer, a saber: Cadastrar Professor, Cadastrar Curso, Cadastrar Disciplina, Cadastrar Período, Cadastrar Turma, Cadastrar Oferta, Exportar Oferta, Cadastrar Sala de Aula, Alocar Salas de Aula, Exportar Alocação. Para cadastrar um professor, o usuário deverá informar dados como nome, departamento, e sexo. Outra atividade que também pode ser executada é o Cadastro de Cursos, onde deverá ser informado o nome e o código do curso. O Cadastro de Disciplinas é semelhante ao Cadastro de Cursos. Para cadastra-la, o usuário deverá informar o nome e o código da mesma. Como este é um sistema para uma universidade, é necessário que exista o Cadastro de Período. Um período é composto de ano, número, data inicial e data final. Uma das principais atividades é o cadastro de Turma. Para cadastrar uma turma, o usuário deve primeiramente cadastrar um Professor, um Curso, uma Disciplina, e um Período. Além

disso, o mesmo deve informar qual o número desta turma e o número de vagas que serão ofertadas.

É importante notar que uma Turma é relacionada a um período, professor, curso e disciplina. Ou seja, é possível existir turmas de mesmo curso e disciplinas, ministradas pelo mesmo professor, mas somente em períodos distintos. Além disso, uma sala de aula não pode estar alocada a mais de uma turma em um mesmo espaço de tempo.

Para produzir uma Oferta de Disciplinas, é preciso inicialmente cadastrar os professores do departamento e as disciplinas do currículo dos cursos de Ciência e Engenharia de Computação (no caso do DI/Ufes). Uma oferta é uma lista de turmas que serão ofertadas em um determinado período. O usuário poderá visualizar ou exportar tal oferta.

Antes de efetuar uma alocação, as salas de aula disponíveis devem ser cadastradas. Uma sala de aula é composta de número, prédio, e número de vagas disponíveis. A principal funcionalidade deste sistema é a Alocação de Turmas a Salas de Aula. Para efetuar a alocação, o usuário poderá escolher se essa alocação será semestral ou esporádica (i.e., irá se repetir durante todo o semestre ou será apenas um dia, por exemplo) e deverá selecionar uma turma (ou no caso da alocação ser esporádica, indicar a razão desta alocação), dias, um horário e uma determinada sala de aula. Após fazer toda a alocação, o mesmo poderá exportar ou visualizar a configuração atual.

3.2 BibLattes – Extração de dados de um CV Lattes

Pesquisadores, estudantes e professores encontram frequentemente dificuldade ao tentar produzir um website contendo informações relacionadas a seus currículo Lattes. Executar tal tarefa (extrair os dados, gerar arquivo no formato adequado e publicar o site) não é uma tarefa tão simples, principalmente para pessoas que não são da área da computação.

O propósito desta ferramenta é, dado um currículo Lattes no formato *xml* (obtido no site da Plataforma Lattes), extrair todas as informações relevantes do mesmo, permitir ao usuário visualizar tais informações de uma maneira mais limpa e exportá-las em formatos adequados, para que sejam usadas, por exemplo, em suas páginas pessoais na Web.

Ao efetuar login no sistema, uma tela será mostrada instruindo o usuário a acessar seu CV Lattes e fazer o *download* do arquivo *xml*. Com este arquivo em mãos, o usuário poderá fazer *upload* do mesmo no sistema que irá, então, efetuar a leitura do arquivo e extrair as informações necessárias, a saber: nome do(s) autor(es) de cada publicação, título, informações sobre onde e quando foi publicado, DOI, e URL para acessá-la.

Caso o usuário já tenha um CV cadastrado na base de dados, o sistema deverá fazer uma atualização de tal CV, substituindo entradas repetidas, inserindo novas entradas

e removendo aquelas que não mais existem no arquivo *xml*.

Ao final de todo o processo, o usuário poderá visualizar como as informações serão exportadas, podendo assim corrigir o que for necessário em seu CV Lattes (é importante notar que o usuário não poderá alterar seu currículo neste sistema, apenas na própria plataforma Lattes).

A princípio, a exportação de dados será feita para o formato BibTeX, para que o usuário possa usá-la em seu site pessoal por meio de ferramentas que já suportam este tipo de formato, como, por exemplo, o *plug-in* `papercite`¹ da plataforma WordPress.

¹ <<https://br.wordpress.org/plugins/papercite/>>

4 Requisitos de Usuário

Tomando por base o contexto dos dois sub-sistemas acima, foram identificados os seguintes requisitos de usuário e regras de negócio:

4.1 AlocaWeb – Controle de Alocação de Salas de Aula

Tabela 1 – Requisitos Funcionais do módulo AlocaWeb

ID	Descrição	Prioridade	Depende
RF-1	O sistema deve permitir o gerenciamento de cursos, contendo: nome, código e quem é o coordenador do curso.	Alta	
RF-2	O sistema deve permitir o cadastro de disciplinas, armazenando informações como nome e código.	Alta	
RF-3	O sistema deve permitir o cadastro de períodos. Um período é composto de ano, número, data inicial e data final.	Alta	
RF-4	O sistema deve permitir o gerenciamento de turmas. Uma turma possui número, quantidade de vagas e é relacionada a um período, disciplina, professor e curso.	Alta	RF-2, RF-3, RF-1
RF-5	O sistema deve permitir o gerenciamento de salas de aula, contendo informações como: prédio, andar, número de cadeiras, número da sala.	Alta	
RF-6	O sistema deve permitir a geração de relatórios de ofertas de disciplinas de um determinado período.	Alta	RF-4
RF-7	O sistema deve permitir que usuários aloquem turmas às salas de aula. Uma alocação é composta de: turma, horário de início, horário de fim, dia da semana e sala de aula.	Alta	RF-4, RF-5
RF-8	O sistema deve permitir a geração de relatórios da alocação de salas em um determinado período.	Alta	RF-7

Tabela 2 – Regras de Negócio do Módulo AlocaWeb

ID	Descrição	Prioridade	Depende
RN-1	Um professor não pode estar alocado a mais de uma turma no mesmo horário e dia da semana.	Alta	RF-7

RN-2	Uma turma só poderá ser alocada a uma sala de aula se não houver nenhuma outra turma naquela sala no horário, dia da semana especificado, período especificado e se o número de vagas naquela turma for menor ou igual ao número de vagas disponíveis na sala.	Alta	RF-7
RN-3	Não podem existir duas salas de aula no mesmo prédio, andar e com o mesmo número.	Alta	RF-5
RN-4	Uma turma pode ser alocada a mais de uma sala de aula, desde que em horários e dias da semana diferentes.	Alta	RF-7
RN-5	Dois turmas de mesmo número não podem existir em um mesmo período.	Alta	RF-4
RN-6	Não podem existir dois períodos com mesmo ano e semestre (por exemplo 2016/1).	Alta	RF-3
RN-7	Dois cursos com mesmo Código ou Nome não podem existir.	Alta	RF-1
RN-8	Não podem existir duas disciplinas com o mesmo código.	Alta	RF-2

4.2 BibLattes – Extração de Dados Currículo Lattes

Tabela 3 – Requisitos Funcionais do Módulo BibLattes

ID	Descrição	Prioridade	Depende
RF-9	O sistema deve prover ao usuário um passo a passo sobre como obter o arquivo XML de um currículo Lattes.	Média	
RF-10	O sistema deve permitir ao usuário fazer o <i>upload</i> do currículo Lattes no formato <i>XML</i> .	Alta	
RF-11	O sistema deve permitir a visualização do currículo existente na base de dados.	Alta	RF-10
RF-12	O sistema deve permitir ao usuário a exportação de seu currículo Lattes no formato BibTeX, contendo informações como: Autores, Local onde foi publicado, data da publicação, páginas (caso exista), editora (caso exista), título, título do livro (caso exista), entre outras.	Alta	RF-10

Tabela 4 – Regras de Negócio do Módulo BibLattes

ID	Descrição	Prioridade	Depende
----	-----------	------------	---------

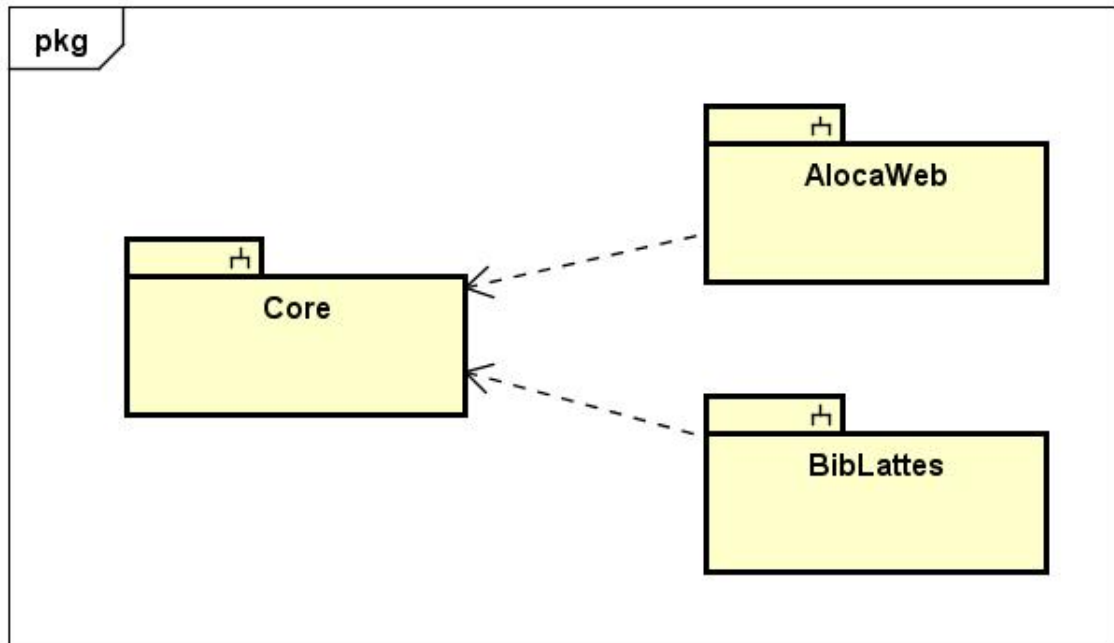
RN-9	Um usuário pode ter no máximo um currículo cadastrado na base de dado em um determinado período de tempo.	Alta	RF-10
RN-10	Usuários não podem efetuar alterações no currículo diretamente pelo sistema, apenas pela plataforma Lattes e/ou alterando o arquivo XML.	Alta	RF-11
RN-11	Um currículo só será enviado para a base de dados depois de o usuário verificar que as informações estão corretas e confirmar a inserção.	Alta	RF-10

Tabela 5 – Requisitos Não Funcionais dos Módulos Allocaweb e BibLattes

ID	Descrição	Categoria	Escopo	Prioridade
RNF-1	A ferramenta deve estar disponível como uma aplicação Web, acessível a partir dos principais navegadores disponíveis no mercado.	Portabilidade	Sistema	Alta
RNF-2	A ferramenta dever ser de aprendizado fácil, não sendo necessário nenhum treinamento especial para seu uso.	Facilidade de Aprendizado	Sistema	Alta
RNF-3	A ferramenta deve sempre gerar um arquivo BibTex contendo informações corretas e atualizadas.	Confiabilidade	Sistema	Alta
RNF-4	O desenvolvimento do sistema deve explorar o potencial de reutilização de componentes, tanto no que se refere ao desenvolvimento com reuso quanto ao desenvolvimento para reuso.	Reusabilidade	Sistema	Média
RNF-5	A ferramenta deve ser de fácil operação, não sendo necessário uso contínuo para uma boa operação do sistema.	Facilidade de Operação	Sistema	Média
RNF-6	O desenvolvimento do sistema deve facilitar manutenções futuras, utilizando padrões de design de software.	Sistema	Média	

5 Identificação de Subsistemas

Tal projeto é composto por dois módulos - BibLattes e AlocaWeb, originando assim diferentes subsistemas. A Figura 1 mostra os subsistemas identificados no contexto do presente projeto, os quais são descritos na tabela abaixo.



powered by Astah

Figura 1 – Diagrama de Pacotes e os Subsistemas Identificados.

Tabela 6 – Subsistemas

Subsistema	Descrição
Core	Envolve todas as funcionalidades centrais do Marvin e que serão utilizadas pelos módulos, tais como: Cadastro de Cursos, Disciplinas, Egressos, Turmas, Salas de Aula, dentre outros.
AlocaWeb	Envolve toda a funcionalidade relacionada à Alocação de Salas de Aula. Este pacote é fortemente dependente do pacote Core, dado que todas as funcionalidades necessárias para alocação de salas de aula (Cadastro de Turmas e Salas de Aula) se encontram nele.
BibLattes	Pacote do módulo BibLattes. Este pacote contém as funcionalidades responsáveis pelo Upload, Exportação e Armazenamento de informações advindas de um currículo lattes.

6 Modelo de Casos de Uso

O modelo de casos de uso visa capturar e descrever as funcionalidades que um sistema deve prover para os atores que interagem com o mesmo. Os atores identificados no contexto deste projeto estão descritos na Tabela 7.

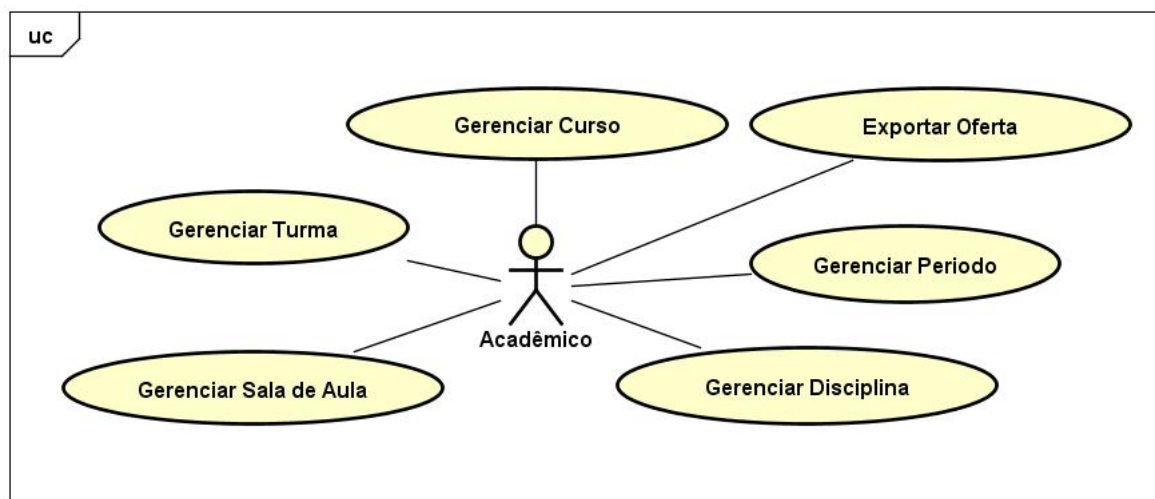
Tabela 7 – Atores

Ator	Descrição
Acadêmico	Qualquer funcionário da UFES com acesso ao sistema

A seguir, são apresentados os diagramas de casos de uso e descrições associadas, organizados por subsistema.

6.1 Subsistema Core

A Figura 2 apresenta o diagrama de casos de uso do subsistema Core.



powered by Astah

Figura 2 – Diagrama de Casos de Uso do Subsistema Core.

A seguir, são apresentadas as descrições de cada um dos casos de uso identificados. Os casos de uso cadastrais de baixa complexidade, envolvendo inclusão, alteração, consulta e exclusão, são descritos na Tabela 8.

Tabela 8 – Casos de Uso Cadastrais

Id	Nome	Ações	Observações	Requisitos	Classes
UC-1	Gerenciar Período	I	Informar: ano, e semestre (selecionar entre primeiro ou segundo), data inicial e data final.	RF-3,RN-6	Semester
		A			
		C			
		E	A exclusão de um periodo resultará na exclusão de todas as Turmas e Alocações ligadas à ele.		
UC-2	Gerenciar Cursos	I	Informar: nome e código	RF-1, RN-7	Course
		A			
		C			
		E	A exclusão de um curso resultará na remoção de todas as Turmas relacionadas à ele.		
UC-3	Gerenciar Disciplina	I	Informar: nome e código	RF-2, RN-8	Subject
		A			
		C			
		E	A exclusão de uma disciplina resultará na remoção de todas as Turmas relacionadas à ela.		
UC-4	Gerenciar Turma	I	Informar: número, quantidade de vagas, período, disciplina, professor e curso	RF-4	Semester, Course, Subject, Academic, Class
		A			
		C			
		E	A exclusão de uma disciplina resultará na remoção de todas as Turmas relacionadas à ela.		
UC-5	Gerenciar Sala de Aula	I	Informar: prédio, andar, número de cadeiras, número da sala.	RF-5, RN-3	Room
		A			
		C			
		E	A exclusão de uma sala irá remover todas as alocações existentes para a mesma.		

Descrição de Caso de Uso

Projeto: AlocaWeb e BibLattes – Módulos do sistema Marvin

Identificador do Caso de Uso: UC-6

Caso de Uso: Exportar Oferta de Disciplinas

Descrição Sucinta: Este caso de uso é responsável por exportar a atual oferta de disciplinas no formato PDF.

Tabela 9 – Fluxos de Eventos Normais

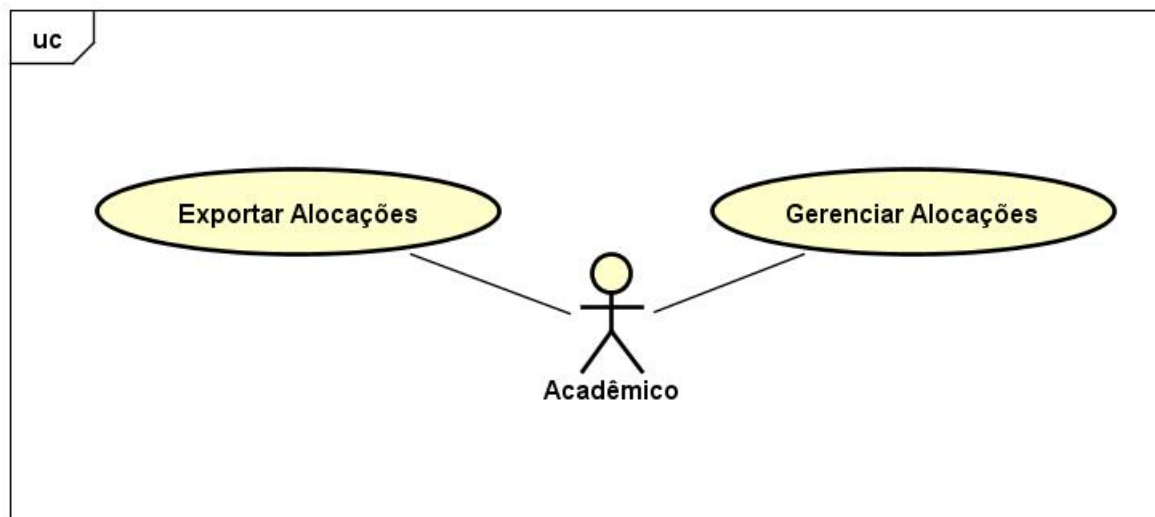
Nome do Fluxo	Precondição	Descrição
Exportar Oferta Disciplinas		<ol style="list-style-type: none">1. O usuário irá selecionar o período o qual deseja realizar a exportação.2. O sistema irá buscar todas as turmas criadas no período selecionado.3. O sistema irá gerar um arquivo no formato PDF com as informações a respeito de cada uma das turmas encontradas.

Requisitos Relacionados: [RF-6](#) , [RN-5](#)

Classes Relacionadas: Class.

6.2 Subsistema AlocaWeb

A Figura 3 apresenta o diagrama de casos de uso do subsistema AlocaWeb.



powered by Astah

Figura 3 – Diagrama de Casos de Uso do Subsistema AlocaWeb.

Descrição de Caso de Uso

Projeto: AlocaWeb e BibLattes – Módulos do sistema Marvin

Identificador do Caso de Uso: UC-7

Caso de Uso: Gerenciar Alocações

Descrição Sucinta: Este caso de uso é responsável por gerenciar as alocações de salas de aula.

Tabela 10 – Fluxos de Eventos Normais

Nome do Fluxo	Precondição	Descrição
Efetuar Alocação		<ol style="list-style-type: none"> 1. O usuário informa a turma. 2. Baseado no número de vagas existentes na turma, o sistema irá mostrar as possíveis salas para tal alocação (que tenha um número de cadeiras maior ou igual ao número de vagas). 3. O usuário irá selecionar uma sala de aula. 4. O usuário irá selecionar um dia da semana 5. O sistema irá verificar os horários disponíveis para essa sala nesse período e irá exibir ao usuário. 6. O usuário deverá selecionar um horário de início e um horário para o termino da alocação. 7. O sistema irá verificar se não existe nenhuma alocação para essa turma dentro do espaço de tempo selecionado e registrar a alocação.
Consultar Alocação		<ol style="list-style-type: none"> 1. O usuário informa o período referente à alocação. 2. O sistema irá retornar todas as alocações relacionadas a tal período.

Tabela 11 – Fluxos de Eventos Variantes

Nome do Fluxo	Variante	Descrição
Efetuar Alocação	2. Não existe sala de aula disponível	2. O sistema irá exibir uma mensagem informando que não existe sala de aula disponível para esta turma.
Efetuar Alocação	5. Todos os horários já estão ocupados para esta sala	4. O sistema irá exibir uma mensagem informando que não há horários disponíveis.
Efetuar Alocação	6. Existe alguma alocação dentro do espaço de tempo selecionado	6. O sistema irá informar o usuário que já existe uma alocação no espaço de tempo escolhido.

Requisitos Relacionados: [RF-7](#), [RN-2](#), [RN-4](#)

Classes Relacionadas: Allocation, Class, Semester, Subject, Course, Room, Academic.

Descrição de Caso de Uso

Projeto: AlocaWeb e BibLattes – Módulos do sistema Marvin

Identificador do Caso de Uso: UC-8

Caso de Uso: Exportar Alocação

Descrição Sucinta: Este caso de uso é responsável pela exportação de alocações existentes.

Tabela 12 – Fluxos de Eventos Normais

Nome do Fluxo	Precondição	Descrição
Exportar Alocação		<ol style="list-style-type: none"> 1. O usuário seleciona o período desejado para exportação. 2. O sistema irá buscar todas as alocação existentes para o período selecionado. 3. O sistema irá gerar um arquivo no formato PDF contendo as alocações.

Tabela 13 – Fluxos de Eventos Variantes

Nome do Fluxo	Variante	Descrição
Exportar Alocação	1. Usuário não seleciona o período para exportação	1. O sistema irá exportar todas as alocações existentes no banco de dados.

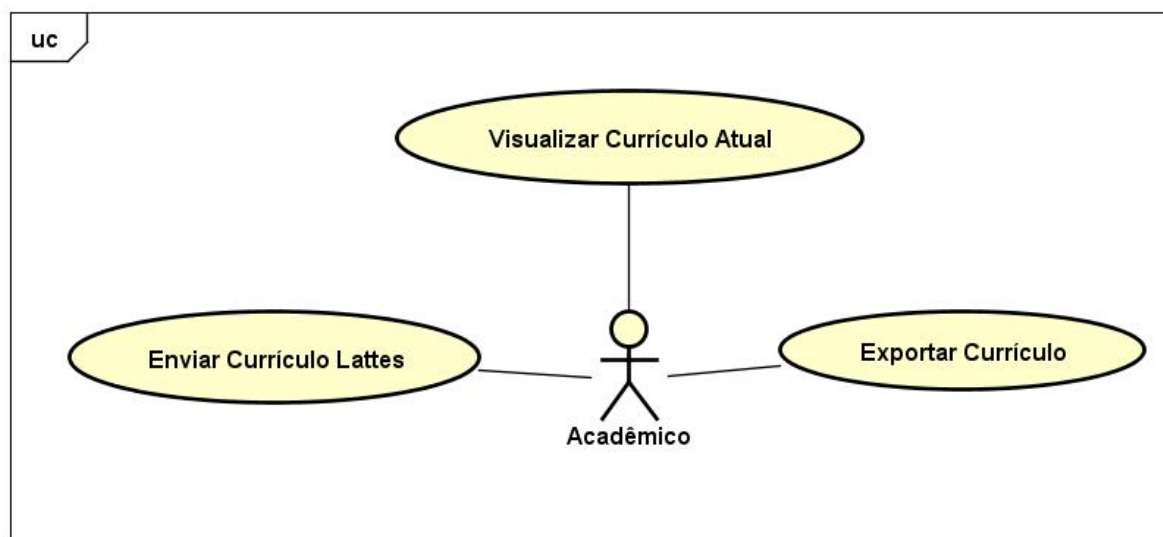
Requisitos Relacionados: [RF-8](#), [RN-2](#)

Classes Relacionadas: Allocation, Class, Semester, Subject, Course, Room, Academic.

6.3 Subsistema BibLattes

A Figura 4 apresenta o diagrama de casos de uso do subsistema BibLattes.

Os casos de uso de consulta mais abrangente que as consultas a um único objeto, mas ainda de baixa complexidade, tais como consultas que combinam informações de vários objetos envolvendo filtros, estão descritos na Tabela 14.



powered by Astah

Figura 4 – Diagrama de Casos de Uso do Subsistema BibLattes.

Tabela 14 – Casos de Uso de Consulta

Id	Nome	Observações	Requisitos	Classes
UC-9	Visualizar Currículo Atual	Depois de efetuar o <i>upload</i> de seu currículo, o usuário poderá visualizar de forma amigável as informações que foram enviadas para o banco de dados, para futuramente exportá-las em formato BibTeX.	RF-11	Book, Collection, Article, Proceeding, Author, Academic

Descrição de Caso de Uso

Projeto: AlocaWeb e BibLattes – Módulos do sistema Marvin

Identificador do Caso de Uso: UC-10

Caso de Uso: Enviar Currículo Lattes

Descrição Sucinta: Este caso de uso é responsável por efetuar o *upload* de um currículo lattes no formato *xml*.

Tabela 15 – Fluxos de Eventos Normais

Nome do Fluxo	Precondição	Descrição
Enviar Currículo Lattes		<ol style="list-style-type: none"> 1. O usuário irá selecionar o currículo lattes no formato <i>xml</i> para <i>upload</i>. 2. O sistema irá apagar o atual currículo Lattes do usuário. 3. O sistema irá extrair as informações das publicações (Trabalhos em Eventos, Livros, Capítulos de Livros e Artigos) existentes neste currículo. 4. O sistema irá, através da web semântica, verificar se existem mais informações sobre tal publicação no DBLP (<https://pt.wikipedia.org/wiki/Digital_Bibliography_%26_Library_Project>). 5. O sistema irá salvar cada uma das publicações extraídas do currículo lattes no banco de dados.

Tabela 16 – Fluxos de Eventos Variantes

Nome do Fluxo	Variante	Descrição
Enviar Currículo Lattes	1. O usuário seleciona um arquivo em outro formato	1. O sistema irá exibir uma mensagem informando que o arquivo selecionado está no formato incorreto.

Requisitos Relacionados: RF-9, RF-10, RN-9, RN-11

Classes Relacionadas: Author, Book, Collection, Proceeding, Article, AuthorPublication.

Descrição de Caso de Uso

Projeto: AlocaWeb e BibLattes – Módulos do sistema Marvin

Identificador do Caso de Uso: UC-11

Caso de Uso: Exportar Currículo

Descrição Sucinta: Este caso de uso é responsável por exportar as informações do Currículo de um usuário no formato BibTeX.

Tabela 17 – Fluxos de Eventos Normais

Nome do Fluxo	Precondição	Descrição
Exportar Currículo		<ol style="list-style-type: none">1. O usuário irá selecionar o currículo lattes que deverá ser exportado.2. O sistema irá buscar no banco de dados todas as publicações relacionadas à este currículo.3. O sistema irá, baseado no formato BibTeX, salvar estas informações em um arquivo <i>.bib</i>.

Requisitos Relacionados: [RF-12](#)

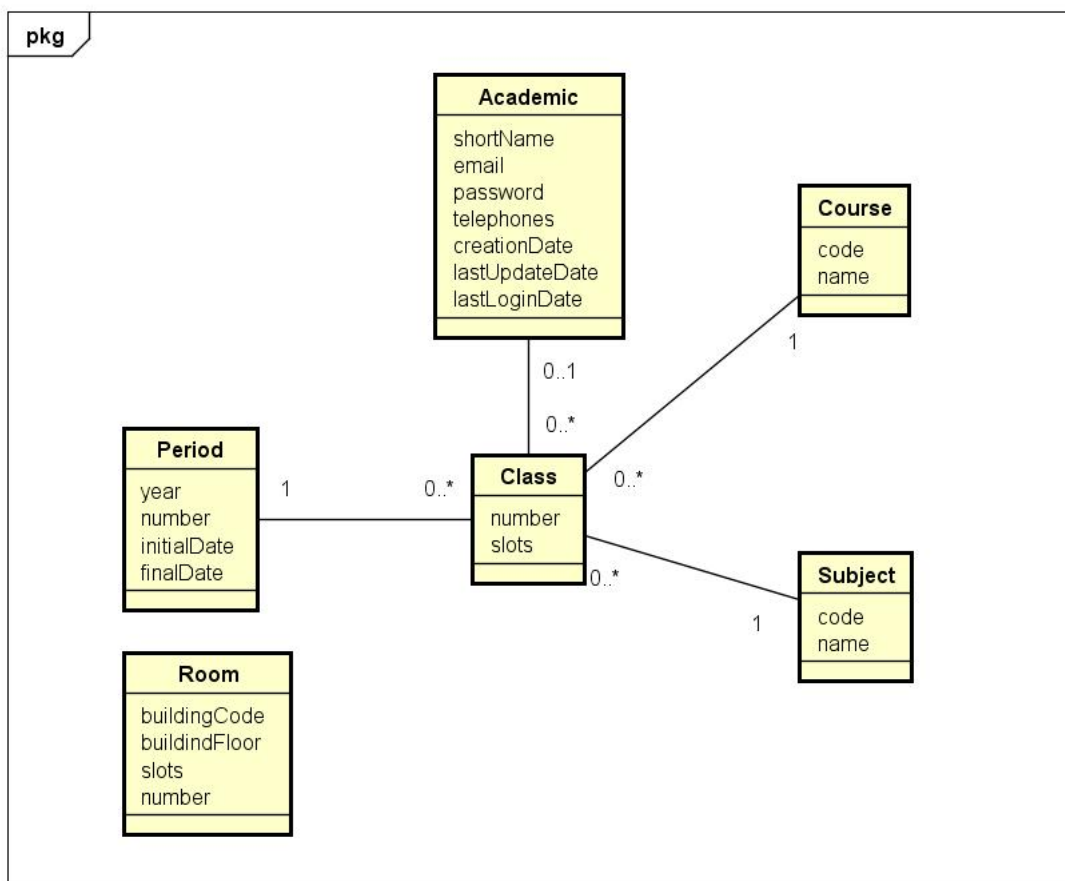
Classes Relacionadas: Author, Book, Collection, Proceeding, Article, AuthorPublication.

7 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 na seção anterior. A seguir, são apresentados os diagramas de classes de cada um dos subsistemas identificados no contexto deste projeto. Na Seção 8 – Dicionário de Projeto – são apresentadas as descrições das classes, atributos e operações presentes nos diagramas apresentados nesta seção.

7.1 Subsistema Core

A Figura 5 apresenta o diagrama de classes do subsistema Core.

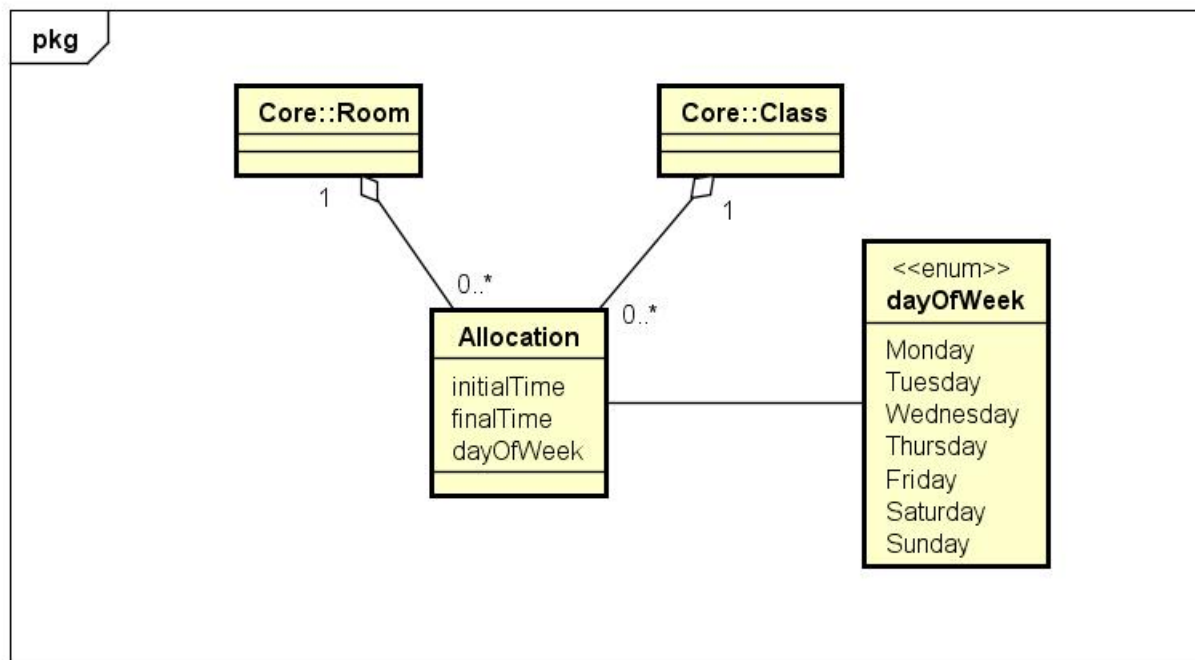


powered by Astah

Figura 5 – Diagrama de Classes do Subsistema Core.

7.2 Subsistema AlocaWeb

A Figura 6 apresenta o diagrama de classes do subsistema AlocaWeb.

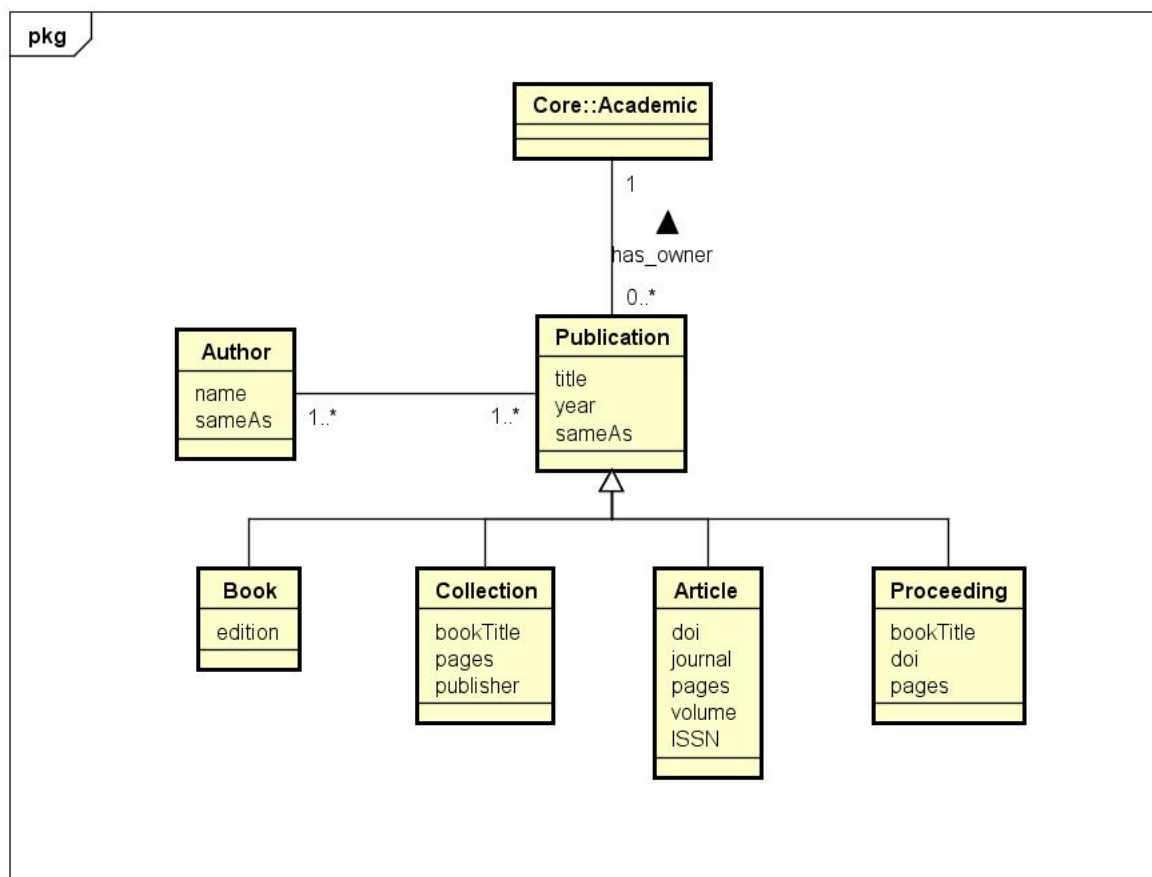


powered by Astah

Figura 6 – Diagrama de Classes do Subsistema AlocaWeb.

7.3 Subsistema BibLattes

A Figura 7 apresenta o diagrama de classes do subsistema BibLattes.



powered by Astah

Figura 7 – Diagrama de Classes do Subsistema BibLattes.

8 Dicionário de Projeto

Esta seção apresenta as definições das classes (e seus atributos), servindo como um glossário do projeto. As definições são organizadas por subsistema. Vale destacar que eventuais operações que estas classes vierem a ter não são listadas e descritas nesta fase do projeto.

8.1 Classes

8.1.1 Course

Propriedade	Tipo	Obrigatório?	Descrição
name	Texto	x	Nome do curso.
code	Texto	x	Código do curso.

8.1.2 Period

Propriedade	Tipo	Obrigatório?	Descrição
initialDate	Data	x	Data de início de um período.
finalDate	Data	x	Data de término de um período.
number	Número	x	Semestre correspondente ao período (Primeiro ou Segundo)
year	Número	x	Ano correspondente ao período.

8.1.3 Subject

Propriedade	Tipo	Obrigatório?	Descrição
code	Texto	x	Código da disciplina.
name	Texto	x	Nome da disciplina.

8.1.4 Class

Propriedade	Tipo	Obrigatório?	Descrição
number	Texto	x	Número da turma.
slots	Número	x	Número de vagas disponíveis nesta turma.
academic	Academic	x	Professor responsável pela turma.
course	Course	x	Curso relacionado à esta turma.
period	Period	x	Periodo ao qual esta turma se refere.
subject	Subject	x	Matéria à qual esta turma está relacionada.

8.1.5 Room

Propriedade	Tipo	Obrigatório?	Descrição
buildingCode	Texto	x	Código do prédio onde esta sala esta localizada.
buildingFloor	Número	x	Andar onde esta sala se encontra.
number	Número	x	Número da sala.
slots	Número	x	Número de cadeiras existentes nesta sala.

8.1.6 Allocation

Propriedade	Tipo	Obrigatório?	Descrição
dayOfWeek	ENUM	x	Dia da semana referente à esta alocação.
initialTime	Número		Horário de início da aula para esta alocação.
finalTime	Número	x	Horário final da aula para esta alocação.
class	Class	x	Turma que está sendo alocada.
room	Room	x	Sala que está sendo utilizada pela turma.

8.1.7 Author

Propriedade	Tipo	Obrigatório?	Descrição
name	Texto	x	Nome do autor.
sameAs	Texto		Recurso DBLP ao qual este autor se refere.

8.1.8 Article

Propriedade	Tipo	Obrigatório?	Descrição
DOI	Texto		DOI do Artigo.
ISSN	Texto		ISSN do Artigo.
journal	Texto	x	Local onde o artigo foi publicado.
pages	Texto	x	Páginas do artigo no Journal.
title	Texto	x	Título do artigo.
volume	Número	x	Volume do journal onde o artigo foi publicado.
year	Número	x	Ano em que o artigo foi publicado.
sameAs	Texto		Recurso DBLP ao qual este artigo se refere.

8.1.9 Proceeding

Propriedade	Tipo	Obrigatório?	Descrição
bookTitle	Texto	x	Livro onde foi publicado este trabalho.
DOI	Texto		DOI do trabalho.
year	Número	x	Ano em que o trabalho foi publicado.
pages	Texto	x	Páginas do trabalho no livro.
title	Texto	x	Título do trabalho.
sameAs	Texto		Recurso DBLP ao qual este trabalho se refere.

8.1.10 Collection

Propriedade	Tipo	Obrigatório?	Descrição
bookTitle	Texto	x	Título do Livro onde o capítulo se encontra.
pages	Texto	x	Páginas referentes ao capítulo.
publisher	Texto	x	Editora que publicou o livro.
title	Texto	x	Título do capítulo.
year	Número	x	Ano em que o livro foi publicado.
sameAs	Texto		Recurso DBLP ao qual este capítulo se refere.

8.1.11 Book

Propriedade	Tipo	Obrigatório?	Descrição
title	Texto	x	Título do Livro.
edition	Número		Edição do Livro.
year	Número	x	Ano em que o livro foi publicado.
sameAs	Texto		Recurso DBLP ao qual este livro se refere.



Documento de Projeto de Sistema

AlocaWeb e BibLattes - Módulos do sistema Marvin

Registro de Alterações:

Versão	Responsável	Data	Alterações
1.0	Thiago Rocha Salvatore	28/07/2016	Versão inicial
1.1	Vítor E. Silva Souza	01/08/2016	Primeira revisão
1.2	Thiago Rocha Salvatore	01/08/2016	Ajustes primeira revisão
1.3	Vítor E. Silva Souza	11/08/2016	Segunda revisão
1.4	Thiago Rocha Salvatore	11/08/2016	Ajustes segunda correção
1.5	Vítor E. Silva Souza	12/08/2016	Terceira revisão
1.7	Vítor E. Silva Souza	15/08/2016	Revisão final

Vitória, ES

2016

1 Introdução

Este documento apresenta o documento de projeto (*design*) do sistema AlocaWeb e BibLattes - Módulos do sistema Marvin. Este documento está organizado da seguinte forma: a Seção 2 apresenta a plataforma de software utilizada na implementação da ferramenta; a Seção 3 trata de táticas utilizadas para tratar requisitos não funcionais (atributos de qualidade); por fim, a Seção 4 apresenta o projeto da arquitetura de software e suas subseções explicam cada uma de suas camadas.

2 Plataforma de Desenvolvimento

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

Tecnologia	Versão	Descrição	Propósito
JavaEE	7	Conjunto de especificação de APIs e tecnologias, que são implementadas por programas servidores de aplicação.	Reduzir a complexidade do desenvolvimento, implantação e gerenciamento de aplicações, de modo que o desenvolvedor não se preocupe demasiadamente com segurança, escalabilidade e desempenho.
Java	8	Linguagem de programação orientada a objetos e independente de plataforma.	Desenvolvimento de aplicativos em linguagem de programação orientada a objetos e independente de plataforma.
JSF	2.2	Framework web baseado em Java que tem como objetivo simplificar o desenvolvimento de interfaces de sistemas para a web.	Melhorar a produtividade, permitindo a construção de interfaces para web usando um conjunto de componentes pré-construídos, ao invés de criar interfaces inteiramente do zero.
EJB	3.2	Componente da plataforma JEE que roda em um container de um servidor de aplicação.	Fornecer um desenvolvimento rápido e simplificado de aplicações Java, com base em componentes distribuídos, transacionais, seguros e portáteis.
JPA	2.1	API para persistência de dados por meio de mapeamento objeto-relacional.	Eliminar muito do trabalho com consultas SQL e facilitar a manutenção visto que menos linhas de código são necessárias.
CDI	1.1	API para injeção de dependências.	Integração das diferentes camadas da arquitetura e serviços de transação.
AdminLTE	2.3.0	Template Bootstrap 3 responsivo	Utilizar um template responsivo open source que seja facilmente personalizado, engloba scripts JS e folhas de estilos CSS para prover um layout responsivo.
Facelets	2.0	Sistema de template Web de código aberto.	Reusar estrutura comum às páginas e facilitar futura manutenção do padrão visual do sistema.
PrimeFaces	5.1	Conjunto de componentes JSF open source com várias extensões.	Reutilizar componentes avançados de interface gráfica.
MySQL Server	5.6.23	Sistema Gerenciador de Banco de Dados Relacional gratuito.	Persistência dos dados manipulados pela ferramenta.
WildFly	9.0.2	Servidor de Aplicações para Java EE.	Prover acesso a aplicações web por meio do protocolo HTTP (HyperText Transfer Protocol).

Tabela 1 – Plataforma de Desenvolvimento e Tecnologias Utilizadas

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

Tecnologia	Versão	Descrição	Propósito
Eclipse Java EE IDE for Web Developers	4.5.1	Ambiente de desenvolvimento (IDE) para a linguagem Java.	Facilitar a atividade de implementação de software.
Astah Community	7.0.0	Ferramenta para modelagem em UML	Modelar os diagramas de classes, casos de uso, etc.
Apache Maven	3.2.5	Ferramenta de gerência de projeto baseada em project object model (POM).	Simplificar o download das dependências do projeto.
TeXstudio	5.5.1	Editor de LaTeX.	Escrever a documentação do sistema.

Tabela 2 – Softwares de Apoio ao Desenvolvimento do Projeto

3 Atributos de Qualidade e Táticas

Na Tabela 3 são listados os atributos de qualidade considerados neste projeto, com uma indicação se os mesmos são condutores da arquitetura ou não e as táticas a serem utilizadas para tratá-los.

Categoria	Requisitos Não Funcionais	Condutor da Arquitetura	Tática
Portabilidade	RNF-1	Sim	1- A camada de lógica de negócio deve ser organizada segundo o padrão Camada de Serviço. 2- Utilizar uma linguagem que seja compatível com os principais navegadores do mercado.
Manutenibilidade	RNF-6	Sim	1- Organizar a arquitetura da ferramenta segundo uma combinação de camadas e partições. 2- A camada de gerência de dados deve ser organizada segundo o padrão DAO.
Facilidade de Aprendizado, Facilidade de Operação	RNF-2, RNF-5	Sim	Prover ao usuário a capacidade de entrar com comandos que permitam operar o sistema de modo mais amigáveis. Para tal, as interfaces do sistema devem permitir, sempre que possível, a entrada por meio de seleção ao invés da digitação de campos.
Confiabilidade	RNF-3	Sim	Sempre que o usuário efetuar o <i>upload</i> de um novo currículo, o sistema irá limpar todas as informações armazenadas e atualizar com as extraídas do mesmo, evitando assim qualquer tipo de inconsistência.
Reusabilidade	RNF-4	Sim	Reutilizar componentes e <i>frameworks</i> existentes. No caso, foi reutilizado o <i>nemo-utils</i> . Quando não houver componentes disponíveis e houver potencial para reuso, devem-se desenvolver novos componentes para reuso.

Tabela 3 – Atributos de Qualidade e Táticas Utilizadas

4 Arquitetura de Software

Antes de falar da arquitetura cabe destacar que AlocaWeb e BibLattes, serão implementados como módulos do Marvin, que é um Sistema de Informação baseado na Web e que agrega ferramentas úteis para o gerenciamento de tarefas de ensino e pesquisa em uma universidade. Marvin é uma tentativa de unificar ferramentas desenvolvidas por estudantes do DI/Ufes durante seus projetos finais de graduação, possibilitando a real utilização por pessoas.

A arquitetura de software destes subsistemas, baseia-se na combinação de camadas e módulos. Cada um desses módulos (AlocaWeb e BibLattes), está organizado em três camadas seguindo o proposto pelo FrameWeb, a saber: Camada de Apresentação (*Presentation Tier*), Camada de Negócios (*Business Tier*) e Camada de Acesso a Dados (*Data Access Tier*). De forma a dar suporte para a construção da aplicação, a ferramenta de apoio nemo-utils será utilizada. Tal ferramenta provê classes que auxiliam na implementação dos casos de uso cadastrais que seguem o modelo de arquitetura a ser utilizado.

A primeira camada contém os pacotes de Visão (*View*) e Controle (*Control*), a segunda contém o de Domínio (*Domain*) e o de Aplicação (*Application*) e a terceira somente o pacote de Persistência (*Persistence*). Cada pacote será explicado melhor nas próximas seções onde serão descritos os módulos *AlocaWeb* e *BibLattes*. A Figura 1 apresenta a visão geral das camadas e seus pacotes juntamente com o relacionamento que existe entre eles e as tecnologias Java EE utilizadas em cada pacote.

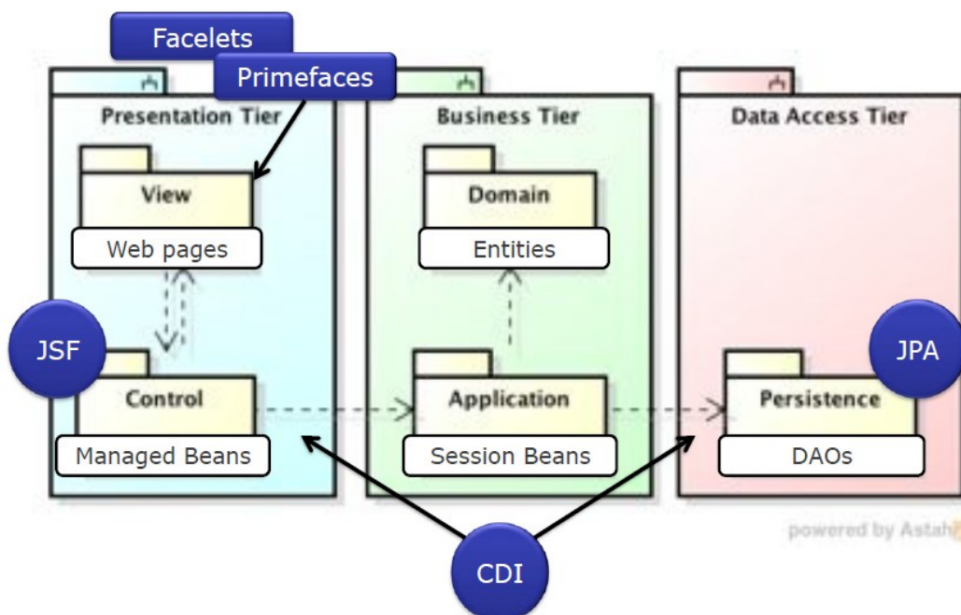


Figura 1 – Arquitetura de Software do Sistema (LIMA, 2015)

A Figura 2 apresenta a subdivisão de cada módulo nas camadas descritas acima, a saber a Camada de Apresentação (**control**), Camada de Negócios (**domain** e **application**) e Camada de Acesso a Dados (**persistence**). A camada de visão (**view**) corresponde às páginas Web, que encontram-se em uma outra pasta, separada das classes Java.

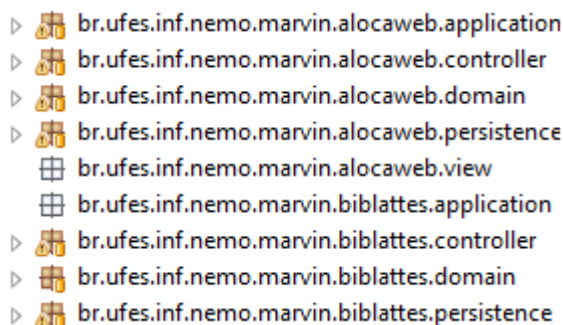


Figura 2 – Subdivisão dos módulos AlocaWeb e BibLattes.

Os diagramas das subseções a seguir seguem o padrão proposto pelo FrameWeb (SOUZA, 2007).

4.1 Camada de Apresentação

As funcionalidades criar, visualizar, editar e excluir (abreviadas de CRUD, do inglês *create, read, update and delete*), seguem um mesmo fluxo de execução e de interação com o usuário. Tais funcionalidades são similares para todos os casos de uso cadastrais devido a utilização da ferramenta nemo-utils. Esse fluxo de execução similar é representado na Figura 3 através de um modelo de apresentação genérico.

Para os casos de uso que apresentam funções diferentes de apenas as básicas de cadastro, o modelo de navegação mostrado anteriormente não pode ser aplicado. A Figura 4 apresenta o modelo de navegação para o fluxo *Exportar Oferta de Disciplinas* do caso de uso *Exportar Oferta de Disciplinas*. É importante notar que a listagem de disciplinas e o filtro por período serão implementados utilizando o nemo-utils, e portanto, o passo de filtragem por período não será mostrado no modelo, já que a Figura 3 exibe tal comportamento.

A Figura 5 exibe o modelo de navegação para o fluxo *Efetuar Alocação* do caso de uso *Gerenciar Alocações* e o fluxo *Exportar Alocação* do caso de uso *Exportar Alocação*. Pode ser percebido neste modelo que, quando o form tem um atributo **x** (neste caso class, por exemplo) e o controlador tem um método que tem tal parâmetro (neste caso o controlador `ManageRoomController` tem o método `updateRooms` que tem como parâmetro class), a passagem da informação deve ser feita via chamada do método. Este fluxo não é proposto pelo FrameWeb, sendo portanto uma sugestão de extensão para o mesmo.

É importante notar que o método *createPDF*, como foi descrito anteriormente, realiza a exportação de alocações baseadas em um período (caso selecionado), mas não

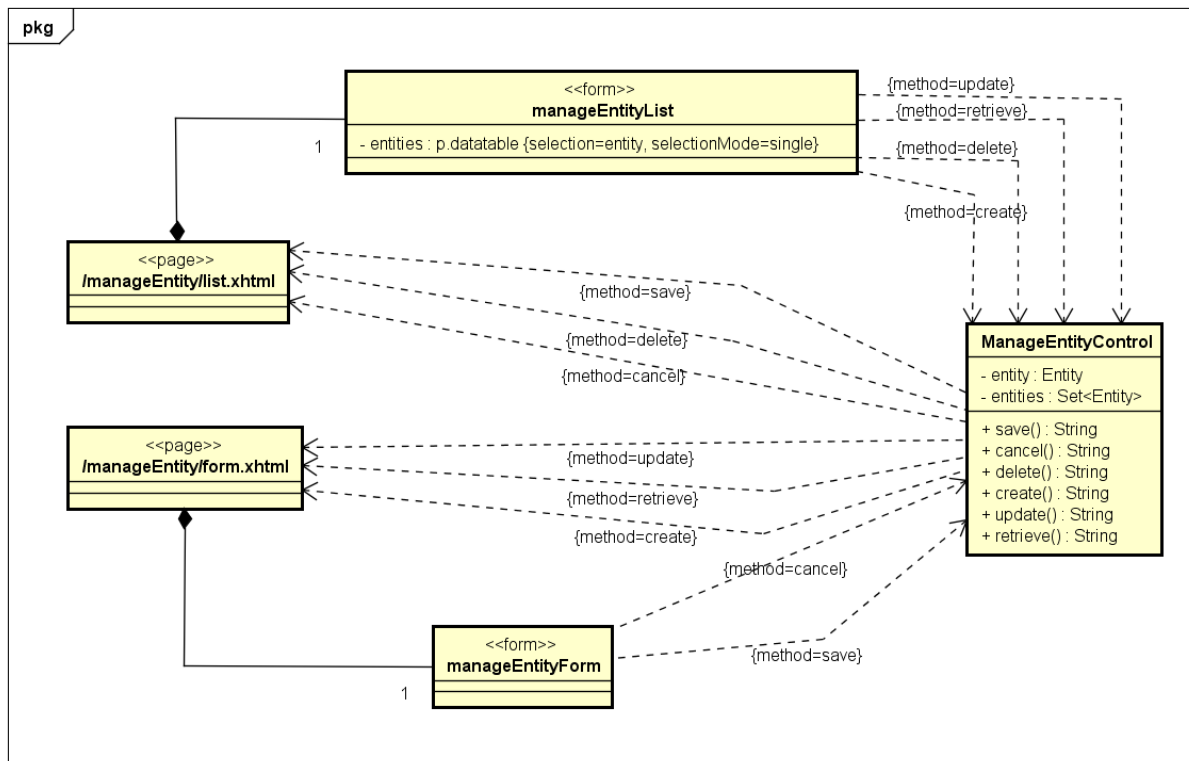
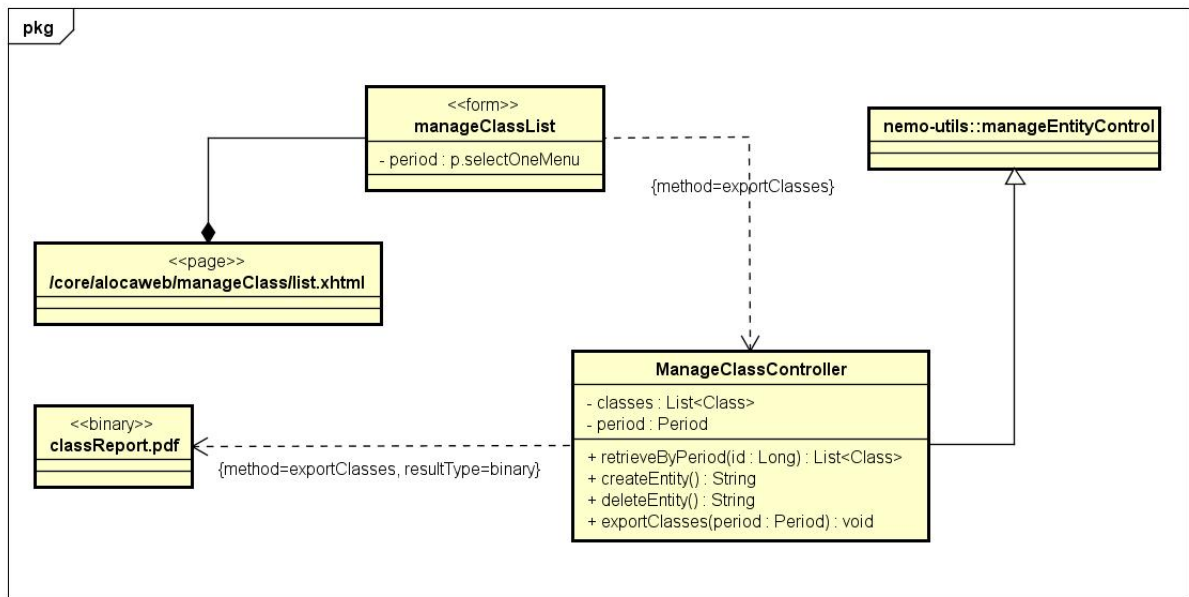


Figura 3 – Modelo de Navegação de um CRUD nemo-utils, usado como base para funcionalidades dos cadastros dos sistema AlocaWeb e BibLattes.

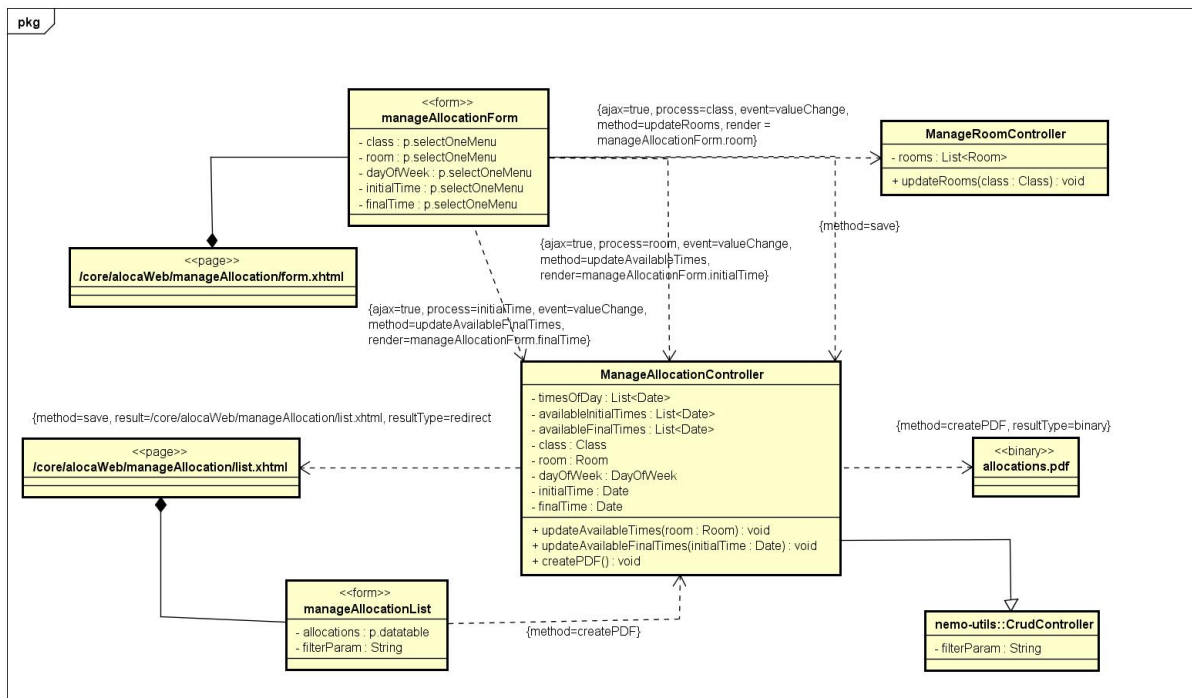
recebe nenhum parâmetro. Isso acontece porque o *nemo-utils* possui sua própria maneira de implementar os filtros (como visto na herança existente na Figura 5). Dentro de cada controlador que herda de **CrudController** existe um atributo *filterParam*. Este atributo é usado dentro da função *createPDF* com o objetivo de retornar (caso selecionado) o período que será utilizado na exportação. O fluxo *Consultar Alocação* do caso de uso *Gerenciar Alocações* corresponde exatamente a esta filtragem por período, portanto também não é exibido no modelo.

Por fim, a Figura 6 mostra o modelo de navegação para o fluxo *Enviar Currículo* do caso de uso *Enviar Currículo Lattes* e o fluxo *Exportar Currículo* do caso de uso *Exportar Currículo*.



powered by Astah

Figura 4 – Modelo de Navegação - Exportar Oferta Disciplinas.



powered by Astah

Figura 5 – Modelo de Navegação - Efetuar Alocação e Exportar Alocação.

4.2 Camada de Negócios

4.2.1 Dominio

Diferente da abordagem FrameWeb original proposta em 2007, todos os atributos que não podem ser nulos tiveram a *tag not null* omitida e os que podem tiveram a *tag null* acrescida de forma a diminuir a poluição visual com repetições desnecessárias no

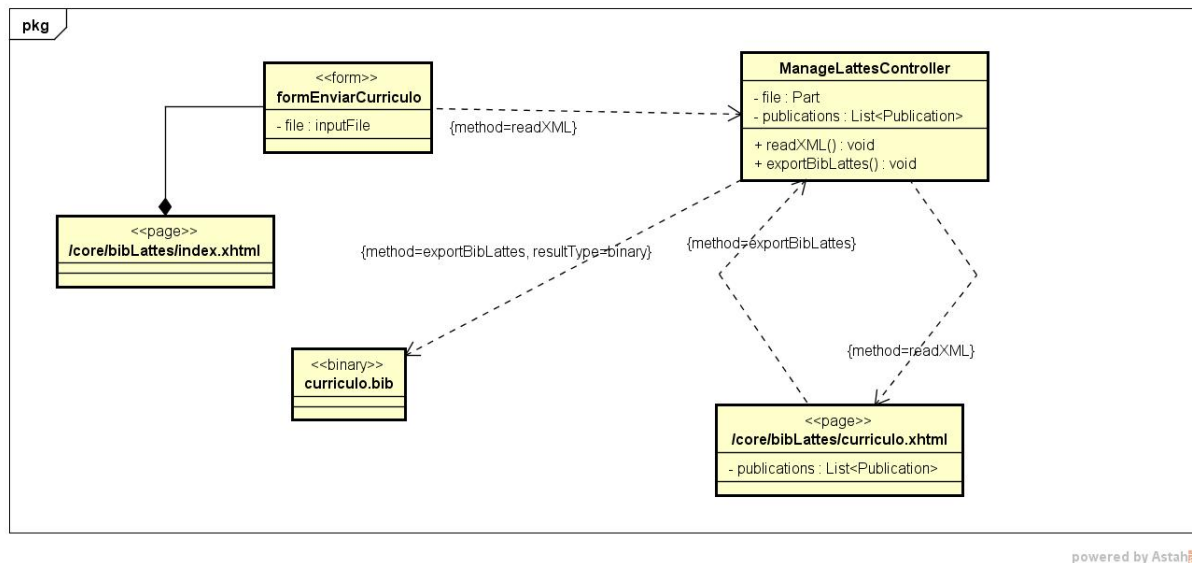
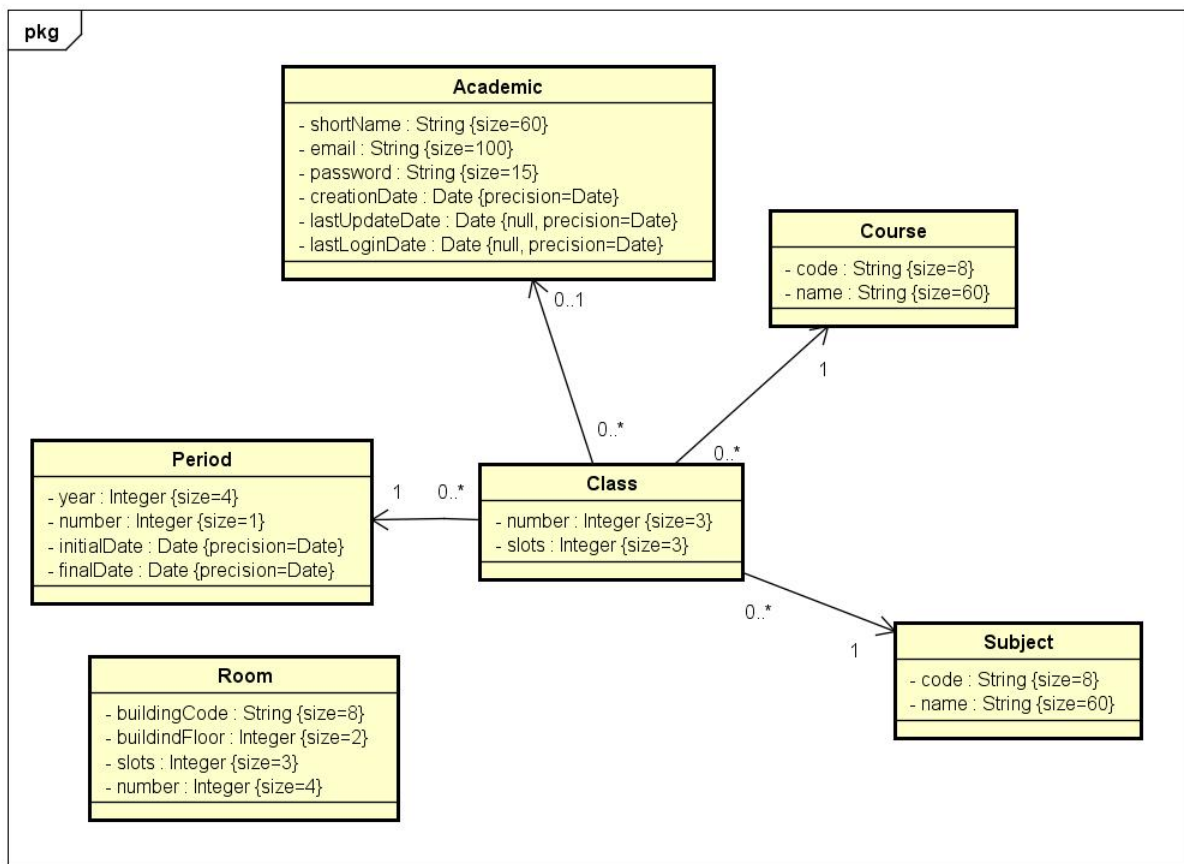


Figura 6 – Modelo de Navegação - Exportar Currículo e Enviar Currículo.

diagrama.

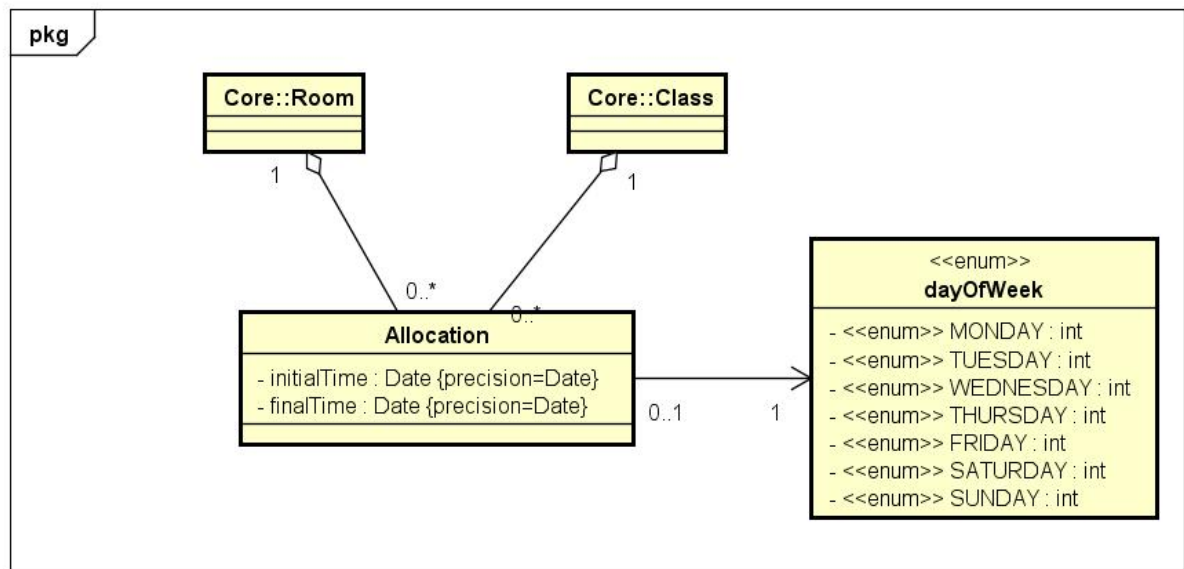
Todas as classes de domínio estendem `PersistentObjectSupport` do pacote `nemo-utills`, sendo que essa herança não é mostrada nos diagramas com o intuito de não poluí-los com várias associações.

A Figura 7 mostra o Modelo de Domínio para o módulo `core`, a Figura 8 mostra o modelo de domínio para o módulo `alocaWeb` e a Figura 9 mostra o modelo de domínio para o `bibLattes`.



powered by Astah

Figura 7 – Modelo de Domínio do Marvin para o módulo *core*.

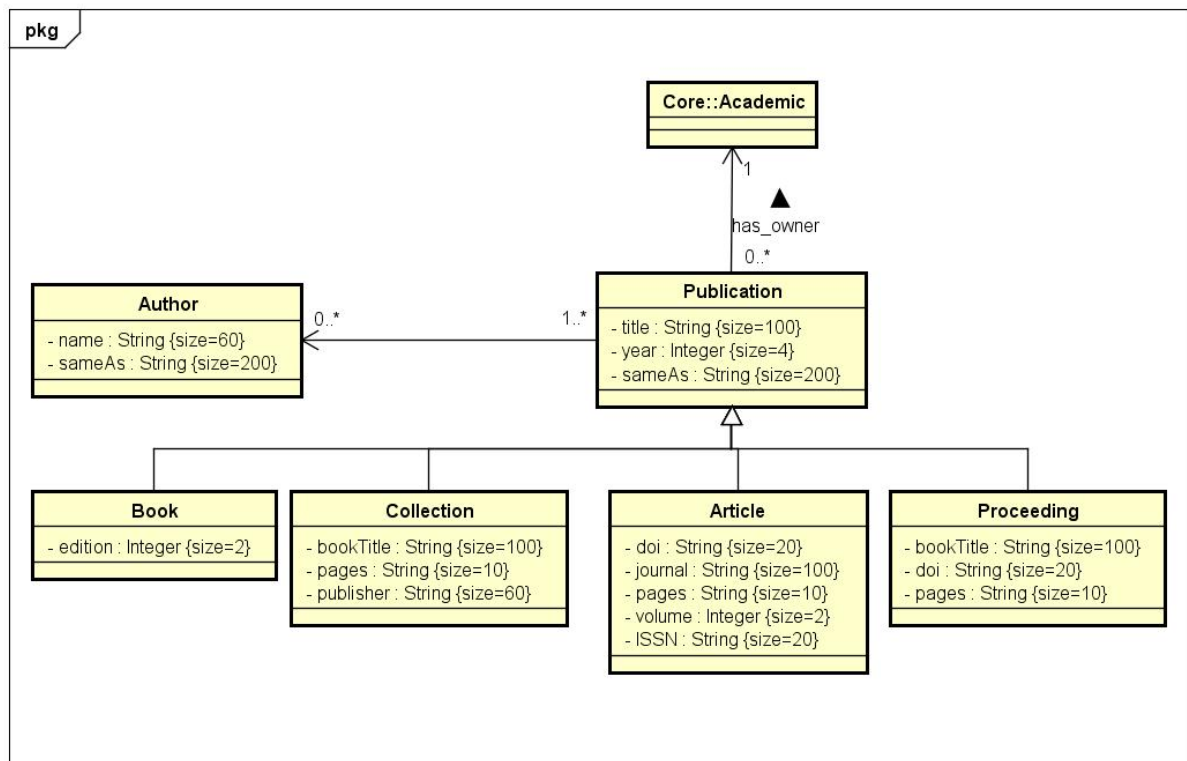


powered by Astah

Figura 8 – Modelo de Domínio do Marvin para o módulo *AlocaWeb*.

4.2.2 Aplicação

Todas as classes de aplicação que são de casos de uso cadastrais estendem de *CrudServiceBean* do pacote *nemo-utils*. Tal classe está representada na Figura 10 de

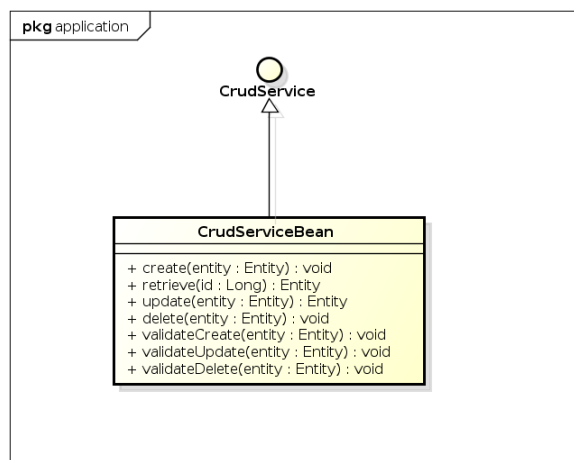


powered by Astah

Figura 9 – Modelo de Domínio do Marvin para o módulo *BibLattes*.

forma genérica. Da mesma forma dos diagramas anteriores essa herança não é mostrada no diagrama com o intuito de não poluí-lo com várias associações.

Os casos de uso não cadastrais Confirmar Seminário e Convidar Palestrante, devido sua baixa complexidade e sua alta relação com o caso de uso gerenciar seminário, foram adicionados dentro de *ManageSeminario*.

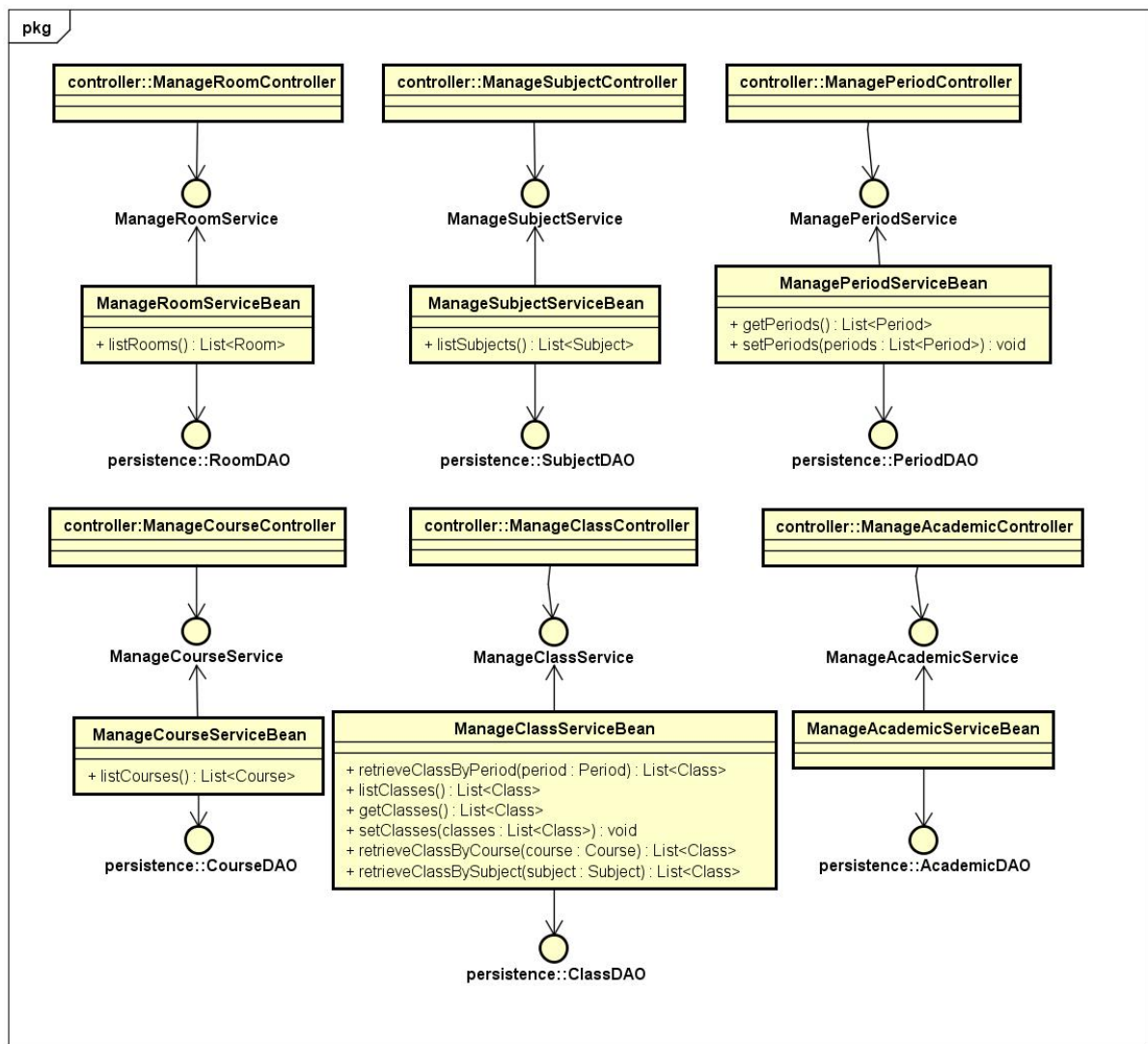


powered by Astah

Figura 10 – Modelo de Aplicação genérica da ferramenta nemo-utils (LIMA, 2015).

A Figura 11 mostra o modelo de aplicação para o módulo *core*, a Figura 12

representa o modelo de aplicação para o módulo *AlocaWeb* e a Figura 13 apresenta o modelo de aplicação para o módulo *BibLattes*.



powered by Astah

Figura 11 – Modelo de Aplicação do Marvin para o módulo *core*.

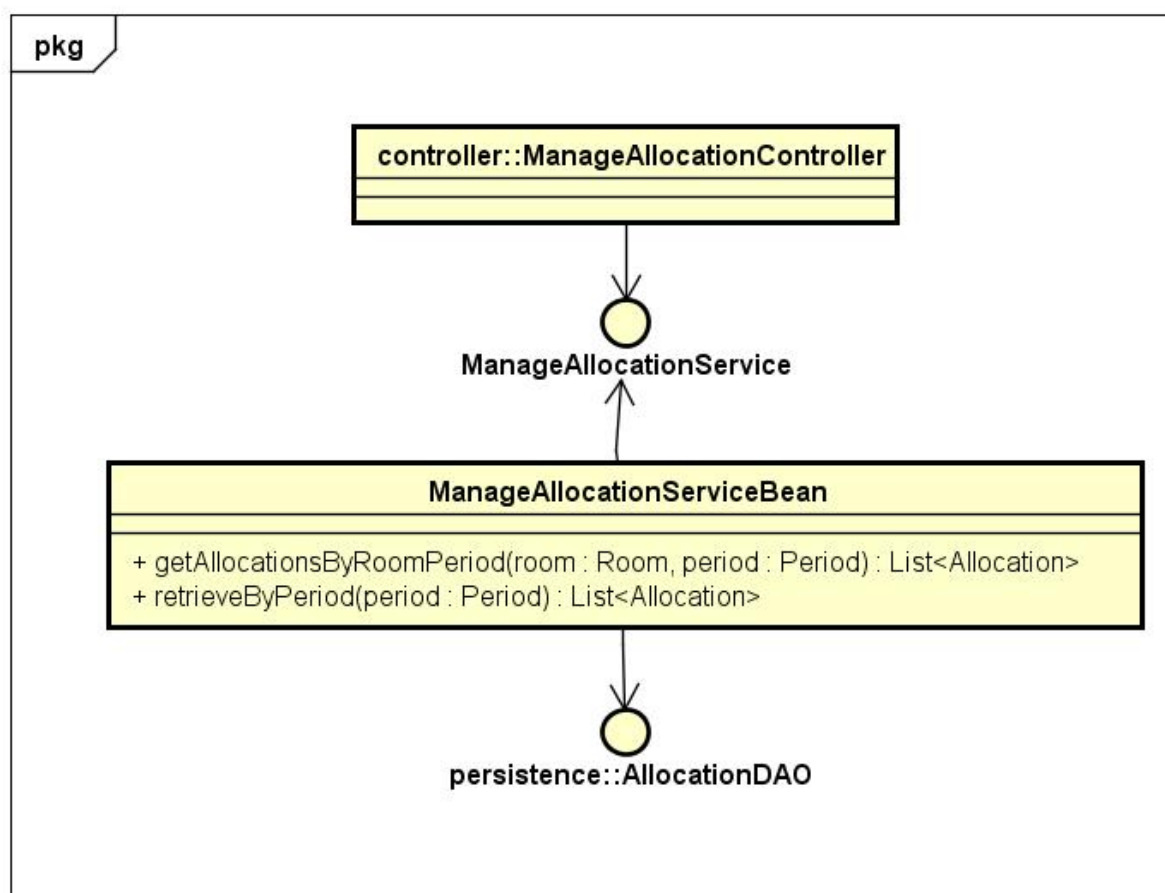
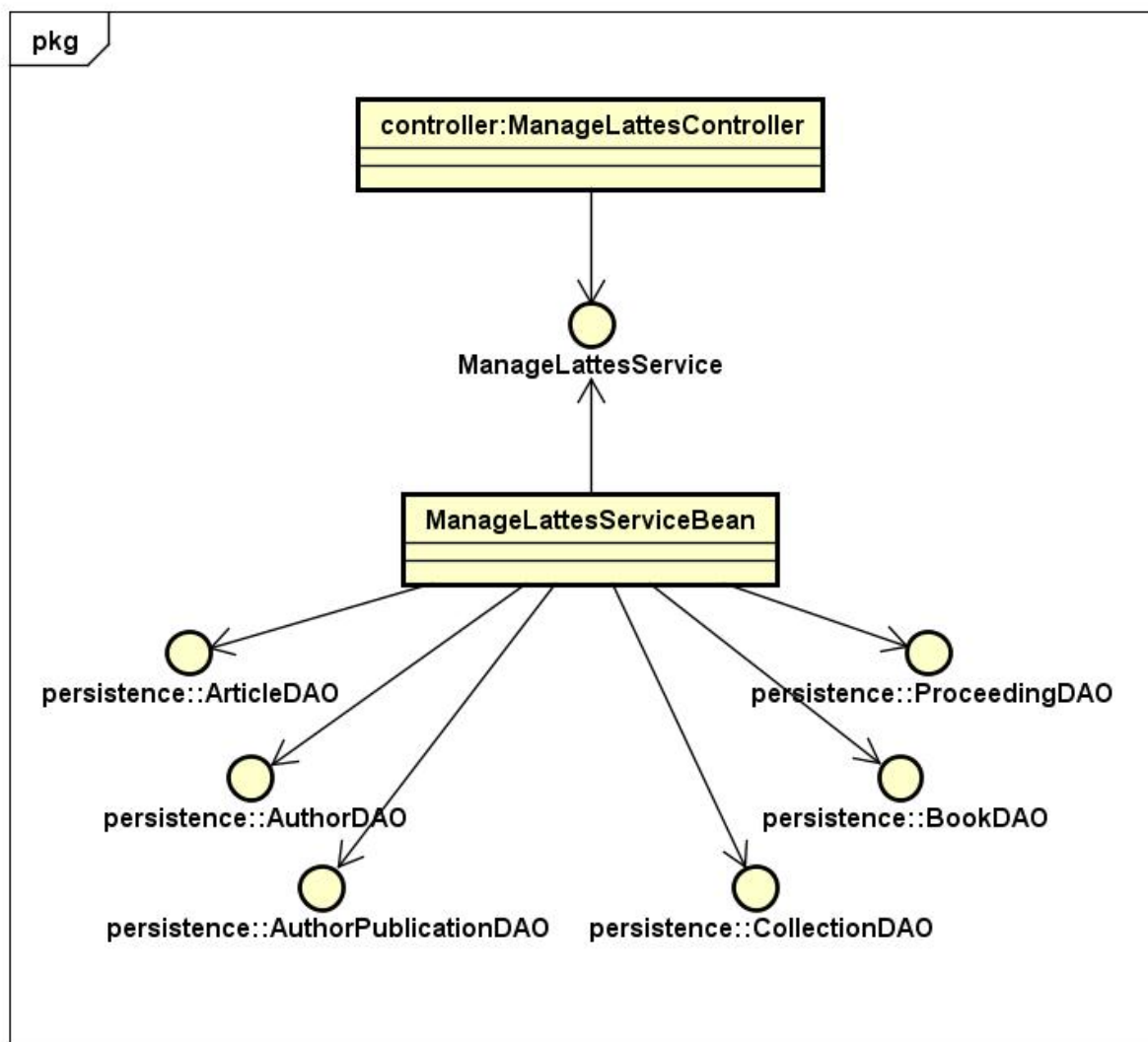


Figura 12 – Modelo de Aplicação do Marvin para o módulo *Aloca Web*.



powered by Astah

Figura 13 – Modelo de Aplicação do Marvin para o módulo *BibLattes*.

4.3 Camada de Acesso a Dados

Nesta seção são apresentados os Modelos de Persistência desenvolvidos para os módulos AlocaWeb e BibLattes e que foram usados na implementação do pacote de persistência.

Vale notar que o nome das classes já indica qual tecnologia de persistência foi utilizada, esse sistema de nomenclatura é mais uma sugestão do FrameWeb para simplificar o processo de software. Vale notar também que na Figura 14 está representado o diagrama de persistência genérico provido pela ferramenta o nemo-utils.

Será possível perceber que a relação de herança entre os DAOs específicos e o DAO base não é representada explicitamente nos diagramas para evitar poluição visual. Esta também é uma recomendação do FrameWeb, ficando, portanto, o desenvolvedor incumbido de derivar essa relação implicitamente ao analisar o modelo.

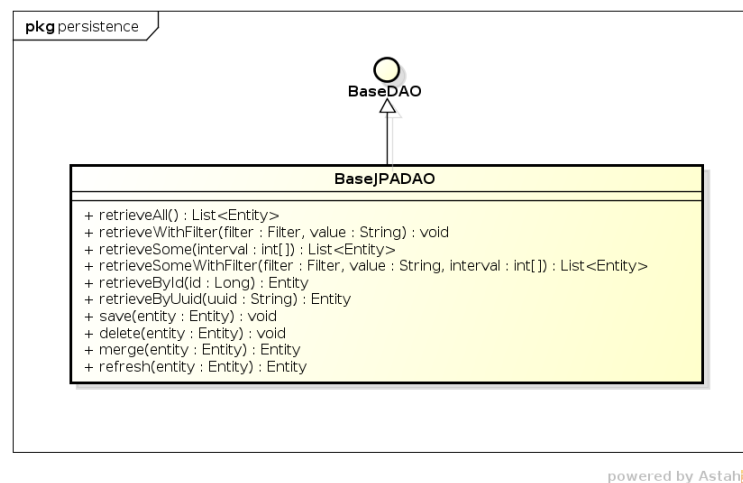
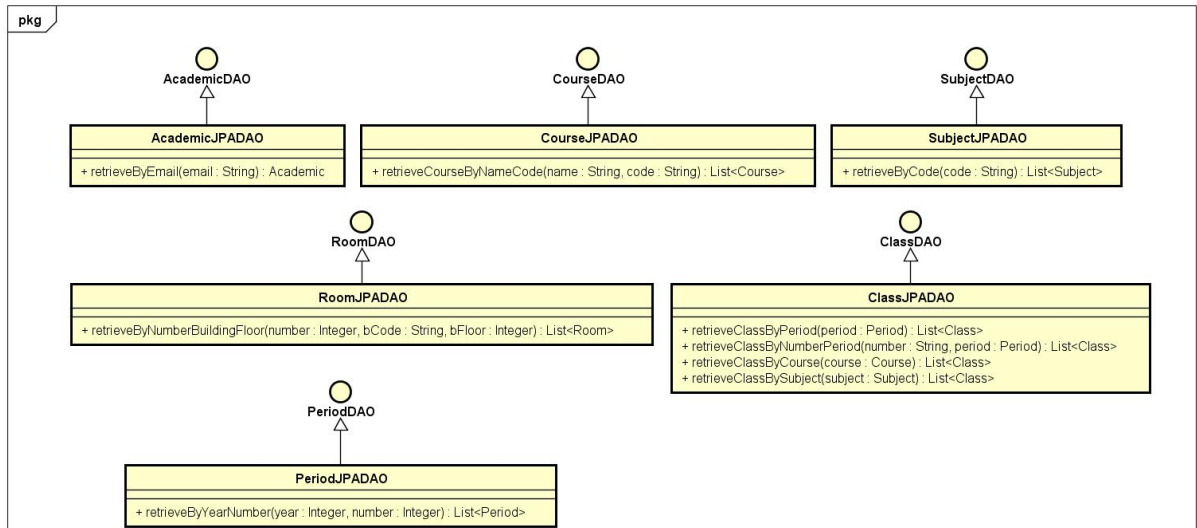


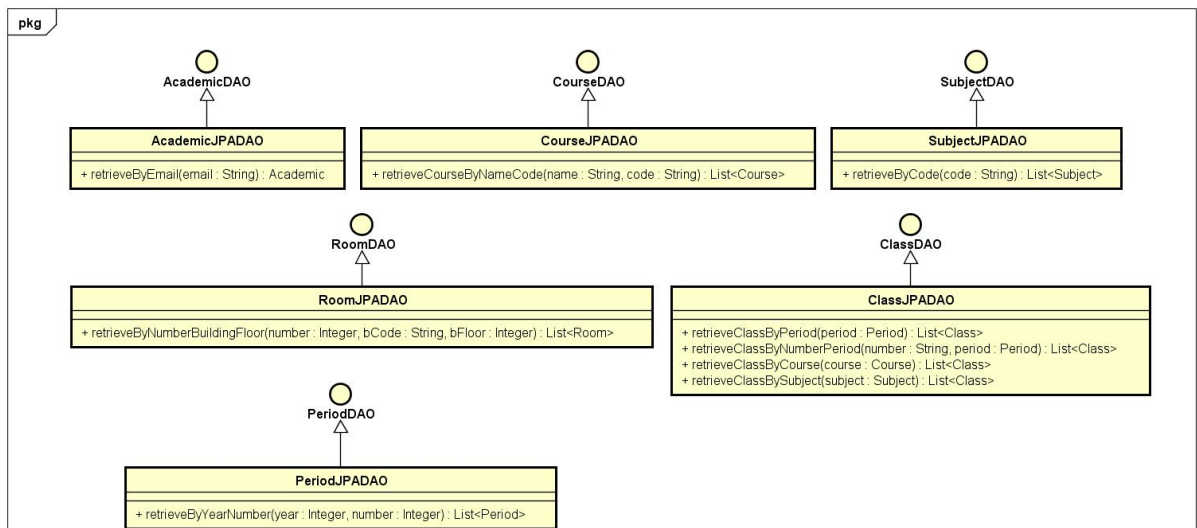
Figura 14 – Modelo de Persistência genérico da ferramenta nemo-utils (LIMA, 2015).

Temos na Figura 15 o modelo para o módulo *core*, na Figura 16 o modelo para o módulo *AlocaWeb* e na Figura 17 o modelo para o módulo *BibLattes*.



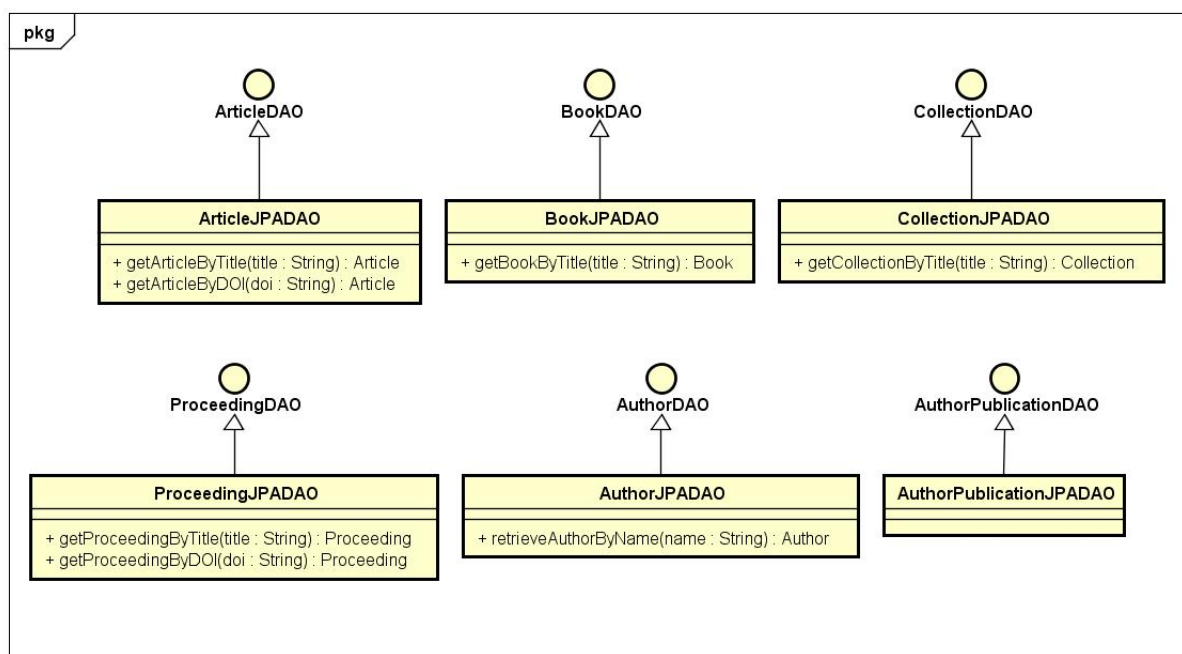
powered by Astah

Figura 15 – Modelo de Persistência do Marvin para o módulo *core*.



powered by Astah

Figura 16 – Modelo de Persistência do Marvin para o módulo *AlocaWeb*.



powered by Astah

Figura 17 – Modelo de Persistência do Marvin para o módulo *BibLattes*.

Referências

LIMA, L. V. F. *SAP - Sistema de Apoio ao Professor*. [S.l.], 2015. Citado 3 vezes nas páginas 5, 11 e 16.

FrameWeb – A Framework-based Design Method for Web Engineering. Citado na página 6.