LUIZ VITOR FRANÇA LIMA

SAP - Sistema de Apoio ao Professor

VITÓRIA, ES

2015

LUIZ VITOR FRANÇA LIMA

SAP - Sistema de Apoio ao Professor

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

2015

LUIZ VITOR FRANÇA LIMA

SAP - Sistema de Apoio ao Professor

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, 15 de setembro de 2015:

Prof. Dr. Vítor E. Silva Souza
Orientador

Dr. Giancarlo Guizzardi
Universidade Federal do Espírito Santo

Danillo Ricardo Celino
Universidade Federal do Espírito Santo

VITÓRIA, ES
2015

Agradecimentos

Em primeiro lugar, agradeço a Deus por ter iluminado toda a minha caminhada.

À minha família e todos os amigos pelo apoio, força, incentivo, por suportarem as noites sem sair de casa e as viagens perdidas devido aos estudos e trabalhos.

À turma CComp2009, onde encontrei amigos para a vida toda, sempre unidos e prestativos para ajudar o próximo e por tornar o ambiente de sala de aula em um lugar mais descontraído e prazeroso de se conviver.

Ao professor Vítor, pela excelente orientação e incentivo, sempre atencioso as dúvidas e pelo exemplo de pessoa e organização.

Por fim, a todos os professores da Universidade que deram sua contribuição ao conhecimento que adquiri durante o curso e também pelas experiências de vida.

Foi um caminho longo e difícil, mas valeu a pena o esforço para vencer cada batalha. Um caminho com muito aprendizado para a vida inteira.

Muito Obrigado a todos!!!

Resumo

No meio acadêmico, é comum os professores que ministram alguma disciplina aplicarem trabalhos para os alunos que estão matriculados. Neste projeto foi desenvolvida uma aplicação Web Java para ser utilizada inicialmente pelos professores do DI/Ufes. Seu objetivo é automatizar alguns processos acadêmicos utilizando a informática para diminuir o trabalho manual do professor, assim como otimizar o seu tempo para que possa desenvolver outras atividades. Utilizando o sistema, o professor poderá cadastrar disciplinas, turmas, trabalhos, importar os dados dos alunos matriculados, além de cadastrar também as orientações que ele realiza junto aos alunos, dentre outras funcionalidades. O desenvolvimento seguiu um processo de Engenharia de Software, utilizando 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: Controle Acadêmico, Aplicação Web, Java, JSF, FrameWeb.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

No meio acadêmico, é comum os professores aplicarem trabalhos para os alunos que estão matriculados em suas turmas. Em alguns casos, os alunos podem se organizar em grupos para o desenvolvimento dos trabalhos. Atualmente, para o professor controlar esses grupos, ele deve solicitar que os alunos enviem os nomes dos participantes de cada grupo para que possa armazenar essas informações em algum lugar de forma manual.

Diante desse contexto, surgiu a motivação para a criação do sistema SAP - Sistema de Apoio ao Professor. Esse sistema foi desenvolvido para automatizar alguns processos do meio acadêmico, como o mencionado anteriormente para controlar os trabalhos e os grupos de alunos. Além disso, o sistema também permitirá que o professor possa cadastrar e controlar as informações dos alunos orientados (iniciação científica, graduação, mestrado e doutorado), tanto os atuais quanto os já formados (histórico de egressos). Desta forma, o sistema auxilia também nas atividades de pesquisa dos professores e fornece a base para que, no futuro, sejam adicionadas novas funcionalidades.

1.1 Objetivos

O objetivo geral deste trabalho é desenvolver um sistema Web que será utilizado para controlar processos do meio acadêmico de uma universidade, utilizando os conceitos aprendidos ao longo do curso de Ciência da Computação. São objetivos específicos deste projeto:

- Levantar os requisitos necessários para o sistema e criar o Documento de Requisitos. Em seguida, analisar os requisitos levantados de modo a produzir o Documento de Especificação de Requisitos. Esse objetivo trabalhou bastante com os conceitos de Engenharia de Software e, em particular, Engenharia de Requisitos;

- Produzir o Documento do Projeto, que define toda a arquitetura para o sistema que será desenvolvido, além de modelar as diferentes camadas do sistema utilizando o método FrameWeb. Esse objetivo relaciona-se com as disciplinas de Projeto de Sistemas e Desenvolvimento Web e Web Semântica (optativa);

- Desenvolver o sistema de acordo com a estrutura definida no Documento do Projeto utilizando a linguagem de programação Java e demais tecnologias de aplicações Web, tais como JSF, AJAX, entre outros. Utilizar também frameworks já existentes para auxiliar no desenvolvimento do sistema. Para esse objetivo exercitou-se conceitos de Programação, Linguagens de Programação e Banco de Dados;

- Apresentar o trabalho desenvolvido e sugerir melhorias futuras para as limitações do sistema.

1.2 Metodologia

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

1. *Revisão Bibliográfica*: Consultar boas práticas de Engenharia de Software e de Requisitos, uso 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 Levantamento de Requisitos, apresentando uma descrição geral do minimundo do sistema e definição dos requisitos funcionais e não funcionais, além das regras de negócio. Em seguida, foi elaborado o Documento de Especificação de Requisitos, apresentando os subsistemas, casos de uso, modelo estrutural, modelo dinâmico 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.

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 4.4.1; Banco de Dados mySQL; JPA e Hibernate (utilizados para realizar a persistência de dados); Primafaces (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 bugs.

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].

1.3 Organização do Texto

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

[1] LaTeX – http://www.latex-project.org/
[2] TeXstudio – http://www.texstudio.org/

- **Capítulo 2** – Referencial Teórico: apresenta uma revisão da literatura acerca de temas relevantes ao contexto deste trabalho, a saber: Engenharia de Software, FrameWeb, aplicações Web e Java;
- **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, além das principais telas do sistema;
- **Capítulo 5** – Considerações Finais: apresenta as conclusões do trabalho, dificuldades encontradas, limitações e propostas de trabalhos futuros.

2 Referencial Teórico

Nesta seção, iremos abordar os principais conceitos teóricos que foram utilizados para o desenvolvimento do sistema SAP.

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.

Pessoas que estão iniciando na área de desenvolvimento de software têm o costume de confundir desenvolvimento com programação, pois estes estão em fase de desenvolver suas habilidades no raciocínio lógico na resolução de pequenos problemas. Entretanto, ao se deparar com problemas mais complexos que requerem maior conhecimento e habilidades, uma abordagem centrada na programação não é a mais indicada.

A **Engenharia de Software** surgiu com o intuito de melhorar a qualidade dos softwares em geral e aumentar a produtividade no desenvolvimento de tais produtos, sendo responsável pelo estabelecimento de técnicas e práticas para o desenvolvimento de software cobrindo uma ampla área de aplicações e diferentes tipos de dispositivos, tais como sistemas de informação corporativos, sistemas e portais Web, aplicações em telefones celulares e tablets, entre outros (FALBO, 2012).

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, ou seja, que contribuem diretamente para o desenvolvimento do produto de software a ser entregue ao cliente. São exemplos de atividades de desenvolvimento: levantamento e análise de requisitos, projeto e implementação;

- Atividades de Gerência: envolvem atividades relacionadas ao gerenciamento do projeto de maneira abrangente. Incluem, dentre outras: atividades de planejamento e acompanhamento gerencial do projeto (processo de Gerência de Projetos), tais como realização de estimativas, elaboração de cronogramas, análise dos riscos do projeto etc.; atividades relacionadas à gerência da evolução dos diversos artefatos produzidos

nos projetos de software (processo de Gerência de Configuração); atividades relacionadas à gerência de ativos reutilizáveis de uma organização (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. Incluem atividades de verificação, validação e garantia da qualidade.

As atividades de desenvolvimento formam a espinha dorsal do desenvolvimento e, de maneira geral, são realizadas segundo uma ordem estabelecida no planejamento. As atividades de gerência e de controle da qualidade são, muitas vezes, ditas atividades de apoio, pois não estão ligadas diretamente à construção do produto final: o software a ser entregue para o cliente, incluindo toda a documentação necessária. Essas atividades, normalmente, são realizadas ao longo de todo o ciclo de vida, sempre que necessário ou em pontos preestabelecidos durante o planejamento, ditos marcos ou pontos de controle.

No que concerne às atividades técnicas, tipicamente o processo de software inicia-se com o Levantamento de Requisitos, quando os requisitos do sistema a ser desenvolvido são preliminarmente capturados e organizados. Uma vez capturados, os requisitos devem ser modelados, avaliados e documentados. Uma parte essencial dessa fase é a elaboração de modelos descrevendo o quê 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).

Com os requisitos pelo menos parcialmente capturados e especificados na forma de modelos, pode-se começar a trabalhar no domínio da solução. Muitas soluções são possíveis para o mesmo conjunto de requisitos e elas são intrinsecamente ligadas a uma dada plataforma de implementação (linguagem de programação, mecanismo de persistência a ser adotado etc.). A fase de projeto tem por objetivo definir e especificar uma solução a ser implementada. É uma fase de tomada de decisão, tendo em vista que muitas soluções são possíveis.

Uma vez projetado o sistema, pode dar-se início à implementação, quando as unidades de software do projeto são implementadas e testadas individualmente. Gradativamente, os elementos vão sendo integrados e testados (teste de integração), até se obter o sistema, quando o todo deve ser testado (teste de sistema). Por fim, uma vez testado no ambiente de desenvolvimento, o software pode ser colocado em produção. Usuários devem ser treinados, o ambiente de produção deve ser configurado e o sistema deve ser instalado e testado, agora pelos usuários no ambiente de produção (testes de homologação ou aceitação). Caso o software demonstre prover as capacidades requeridas, ele pode ser aceito e a operação iniciada.

A Engenharia de Requisitos é o processo pelo qual os requisitos de um produto de

software são coletados, analisados, documentados e gerenciados ao longo de todo o ciclo de vida do software (AURUM; WOHLIN, 2005).

Requisitos têm um papel central no desenvolvimento de software, uma vez que uma das principais medidas do sucesso de um software é o grau no qual ele atende aos objetivos e requisitos para os quais foi construído. Requisitos são a base para estimativas, modelagem, projeto, implementação, testes e até mesmo para a manutenção. Portanto, estão presentes ao longo de todo o ciclo de vida de um software (FALBO, 2012).

Sistemas de software são reconhecidamente importantes ativos estratégicos para diversas organizações. Uma vez que tais sistemas, em especial os sistemas de informação, têm um papel vital no apoio aos processos de negócio das organizações, é fundamental que os sistemas funcionem de acordo com os requisitos estabelecidos. Neste contexto, uma importante tarefa no desenvolvimento de software é a identificação e o entendimento dos requisitos dos negócios que os sistemas vão apoiar (AURUM; WOHLIN, 2005).

Nos estágios iniciais de um projeto, requisitos têm de ser levantados, entendidos e documentados (atividades de Levantamento, Análise e Documentação de Requisitos). Dada a importância dos requisitos para o sucesso de um projeto, atividades de controle da qualidade devem ser realizadas para verificar, validar e garantir a qualidade dos requisitos, uma vez que os custos serão bem maiores se defeitos em requisitos forem identificados tardiamente. Mesmo quando coletados de forma sistemática, requisitos mudam. Os negócios são dinâmicos e não há como garantir que os requisitos não sofrerão alterações. Assim, é fundamental gerenciar a evolução dos requisitos, bem como manter a rastreabilidade entre os requisitos e os demais artefatos produzidos no projeto (atividade de Gerência de Requisitos) (FALBO, 2012).

Como se pode observar, o tratamento de requisitos envolve atividades de desenvolvimento (Levantamento, Análise e Documentação de Requisitos), gerência (Gerência de Requisitos) e controle da qualidade (Verificação, Validação e Garantia da Qualidade de Requisitos). Ao conjunto de atividades relacionadas a requisitos, dá-se o nome de Processo de Engenharia de Requisitos.

Neste projeto foram utilizadas as técnicas de levantamento e especificação de requisitos aprendidas ao longo do curso, como descrição de minimundo, levantamento de requisitos funcionais e não-funcionais, modelagem de casos de uso e modelagem conceitual estrutural. Nos parágrafos que se seguem, descrevemos brevemente estas técnicas.

A descrição do minimundo apresenta, em um texto corrido, uma visão geral do domínio do problema a ser resolvido e dos processos de negócio apoiados, bem como as principais ideias do cliente sobre o sistema a ser desenvolvido.

Já os requisitos funcionais, são declarações de serviços que o sistema deve prover, descrevendo o que o sistema deve fazer (SOMMERVILLE, 2007). Um requisito funcional

descreve uma interação entre o sistema e o seu ambiente (PFLEEGER, 2004), podendo descrever, ainda, como o sistema deve reagir a entradas específicas, como o sistema deve se comportar em situações específicas e o que o sistema não deve fazer (SOMMERVILLE, 2007).

De forma simplificada, os requisitos funcionais são aqueles que fazem parte do sistema, como um relatório específico, um campo em um cadastro, etc. Eles normalmente têm a finalidade de agregar valor ao usuário ou facilitar o trabalho que ele desenvolve. Requisitos funcionais serão implementados no próprio sistema e a implementação desses requisitos caracteriza um sistema. Os requisitos funcionais identificados para o sistema SAP são descritos no Apêndice ao final dessa monografia.

Assim como os requisitos funcionais precisam ser especificados em detalhes, o mesmo acontece com os requisitos não funcionais. Para os atributos de qualidade considerados prioritários, o analista deve trabalhar no sentido de especificá-los de modo que eles se tornem mensuráveis e, por conseguinte, testáveis. Eles descrevem restrições sobre os serviços ou funções oferecidos pelo sistema (SOMMERVILLE, 2007), as quais limitam as opções para criar uma solução para o problema (PFLEEGER, 2004).

O modelo de casos de uso é um modelo comportamental, mostrando 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. Assim, ele descreve o sistema segundo uma perspectiva externa. As descrições dos casos de uso descrevem o passo a passo para a realização dos casos de uso e são essencialmente textuais (FALBO, 2012).

Tomando por base casos de uso e suas descrições, é possível passar à modelagem conceitual estrutural, quando os conceitos e relacionamentos envolvidos no domínio são capturados em um conjunto de diagramas de classes. Neste momento é importante definir, também, o significado dos conceitos e de suas propriedades, bem como restrições sobre eles. Essas definições são documentadas em um dicionário de dados do projeto.

Um diagrama de classes exibe um conjunto de classes e seus relacionamentos. Diagramas de classes proveem uma visão estática da estrutura de um sistema e, portanto, são usados na modelagem conceitual estrutural. Para tornar os modelos conceituais mais simples, de modo a facilitar a comunicação com clientes e usuários, tipos de dados de atributos podem ser omitidos do diagrama de classes. Restrições de integridade são regras de negócio e poderiam ser lançadas no Documento de Requisitos. Contudo, como elas são importantes para a compreensão e eliminação de ambiguidades do modelo conceitual, é útil descrevê-las no próprio modelo conceitual. (FALBO, 2012).

2.2 Conceitos de Java

Durante muitos anos, os usuários se habituaram com aplicações Desktop. Este tipo de aplicação é instalada no computador local e acessa diretamente um banco de dados ou gerenciador de arquivos. Entretanto, existem algumas desvantagens no desenvolvimento Desktop. Como cada usuário tem uma cópia integral da aplicação, qualquer alteração precisaria ser propagada para todas as outras máquinas. Estamos usando um cliente gordo, isto é, com muita responsabilidade no lado do cliente. Note que, aqui, estamos chamando de cliente a aplicação que está rodando na máquina do usuário. Para piorar, as regras de negócio rodam no computador do usuário. Isso faz com que seja muito mais difícil depurar a aplicação, já que não costumamos ter acesso tão fácil à maquina onde a aplicação está instalada. Em geral, enfrentamos problemas de manutenção e gerenciamento (CAELUM, 2015).

Para resolver problemas como esse, surgiram as aplicações baseadas na Web. Nessa abordagem há um servidor central onde a aplicação é executada e processada e todos os usuários podem acessá-la através de um cliente simples e do protocolo HTTP[1] (*HyperText Transfer Protocol, em português "Protocolo de Transferência de Hipertexto"*). Um navegador Web, como Firefox[2] ou Chrome[3], que fará o papel da aplicação cliente, interpretando HTML[4] (*Hypertext Markup Language, em português "Linguagem de Marcação de Hipertexto"*), CSS(*Cascading Style Sheets, em português "folhas de estilo em cascata"*) e JavaScript - que são as tecnologias que ele entende.

Enquanto o usuário usa o sistema, o navegador envia requisições (*requests*) para o lado do servidor (*server side*), que responde para o computador do cliente (*client side*). Em nenhum momento a aplicação está salva no cliente: todas as regras da aplicação estão no lado do servidor. Por isso, essa abordagem também foi chamada de cliente magro (*thin client*). A Figura 1 exibe a arquitetura Web mencionada (CAELUM, 2015).

Isso facilita bastante a manutenção e a gerenciabilidade, pois temos um lugar central e acessível onde a aplicação é executada. Contudo, note que será preciso conhecer HTML, CSS e JavaScript, para fazer a interface com o usuário, e o protocolo HTTP para entender a comunicação pela Web. E, mais importante ainda, não há mais eventos, mas sim um modelo bem diferente orientado a requisições e respostas. Toda essa base precisará ser conhecida pelo desenvolvedor.

Comparando as duas abordagens, podemos ver vantagens e desvantagens em ambas. No lado da aplicação puramente Desktop, temos um estilo de desenvolvimento orientado a eventos, usando componentes ricos, porém com problemas de manutenção e gerenciamento.

1 HTTP – https://pt.wikipedia.org/wiki/Hypertext_Transfer_Protocol
2 Firefox – https://www.mozilla.org/pt-BR/firefox/
3 Chrome – https://www.google.com/chrome/
4 HTML – https://pt.wikipedia.org/wiki/HTML

Figura 1 – Arquitetura Aplicação Web (CAELUM, 2015).

Do outro lado, as aplicações Web são mais fáceis de gerenciar e manter, mas precisamos lidar com HTML, conhecer o protocolo HTTP e seguir o modelo requisição/resposta (CAELUM, 2015).

Em vez de desenvolver puramente para Desktop, é uma tendência mesclar os dois estilos, aproveitando as vantagens de cada um. Em outras palavras, realizar um desenvolvimento Desktop para a Web, tanto central quanto com componentes ricos, aproveitando o melhor dos dois mundos e abstraindo o protocolo de comunicação. Essa é justamente a ideia dos frameworks Web baseados em componentes. No mundo Java há algumas opções como JavaServer Faces (JSF)[5], Apache Wicket[6], Vaadin[7], Tapestry[8] ou GWT da Google[9]. Todos eles são frameworks Web baseados em componentes.

JSF, parte da plataforma Java EE (*Enterprise Edition*), é uma tecnologia que nos permite criar aplicações Java para Web utilizando componentes visuais pré-prontos, de forma que o desenvolvedor não se preocupe com Javascript e HTML. Basta adicionarmos

[5] JavaServer Faces (JSF) – https://javaserverfaces.java.net/
[6] Apache Wicket – http://wicket.apache.org/
[7] Vaadin – https://vaadin.com/home
[8] Tapestry – http://tapestry.apache.org/
[9] GWT – http://www.gwtproject.org/

os componentes (calendários, tabelas, formulários) e eles serão renderizados e exibidos em formato HTML (CAELUM, 2015).

Além disso o estado dos componentes é sempre guardado automaticamente (como veremos mais à frente), criando a característica *Stateful*. Isso nos permite, por exemplo, criar formulários de várias páginas e navegar nos vários passos dele com o estado das telas sendo mantidos. Outra característica marcante na arquitetura do JSF é a separação que fazemos entre as camadas de apresentação e de aplicação. Pensando no modelo MVC (*Model-View-Controller*)[10], o JSF possui uma camada de visualização bem separada do conjunto de classes de modelo. O JSF ainda tem a vantagem de ser uma especificação do Java EE, isto é, todo servidor de aplicações Java tem que vir com uma implementação dela e há diversas outras disponíveis.

Existem diversas extensões baseadas na tecnologia JavaServer Faces que se destinam a tornar mais simples o uso de AJAX (*Asynchronous Javascript and XML, em português "Javascript e XML Assíncronos"*) e componentes em aplicações Web. Este é o caso do PrimeFaces que é uma biblioteca de componentes de código aberto para o JSF 2.0 com mais de 100 componentes, permitindo criar interfaces ricas para aplicações Web de forma simplificada e eficiente. Ele possui as seguintes vantagens em relação a outras bibliotecas de componentes JSF, tendo em vista que (CAELUM, 2015):

- Possui um rico conjunto de componentes de interface (DataTable, AutoComplete, HTMLEditor, gráficos etc);
- Nenhum XML de configuração extra é necessário e não há dependências;
- Componentes construídos com AJAX no padrão JSF 2.0 AJAX APIs;
- Mais de 25 temas templates;
- Boa documentação com exemplos de código;
- Não possui nenhuma outra dependência, ou seja, basta colocar a biblioteca do PrimeFaces em seu projeto e começar a utilizar, não existindo nenhum *overhead* (geralmente considerado como qualquer processamento ou armazenamento em excesso, seja de tempo de computação, de memória ou qualquer outro recurso que seja requerido para ser utilizado ou gasto para executar uma determinada tarefa.).

O PrimeFaces é um framework da Prime Teknoloji (empresa da Turquia) que oferece um conjunto de componentes ricos para o JavaServer Faces. Seus componentes foram construídos para trabalhar com AJAX por "default", isto é, não é necessário nenhum esforço extra por parte do desenvolvedor para realização de chamadas assíncronas ao

[10] MVC – https://pt.wikipedia.org/wiki/MVC

servidor. Além disso, o PrimeFaces permite a aplicação de temas (*skins*) com o objetivo de mudar a aparência dos componentes de forma simples.

Além do JSF, outro componente da plataforma Java EE utilizado foi o CDI. O CDI (Contexts and Dependency Injection for the Java™EE Platform, em português "Contextos e Injeção de Dependências para a plataforma Java™EE") foi proposto pela JSR 299 (*Java Specification Requests*)[11], cuja versão final aprovada pelo JCP (*Java Community Process*)[12] foi publicada em dezembro de 2009 e incluída na plataforma Java EE 6 para desenvolvimento de aplicações corporativas. Permite que declaremos uma dependência de uma classe do sistema (chamada de bean) a um EJB (*Enterprise JavaBeans*)[13] utilizando a anotação `@EJB` ou a uma classe não-EJB utilizando `@Inject`, ambos sobre o atributo que representa a associação entre o dependente e sua dependência. Provê acesso via linguagem de expressões unificada (*Expression Language*, ou EL) a beans que utilizarem a anotação `@Named` na definição da classe. Tal anotação permite definir um nome para o componente ou utilizar o nome default: o mesmo nome da classe, trocando a primeira letra para minúscula. Traz consigo alguns estereótipos que representam papéis desempenhados por determinados componentes em arquiteturas comuns e padrões de projeto, como o `@Model`, que representa o modelo em uma arquitetura MVC. Possibilita, ainda, que o desenvolvedor crie seus próprios estereótipos. As classes gerenciadas pelo CDI são associadas a determinados contextos para gerenciamento automático do seu ciclo de vida. O CDI oferece, além disso, uma série de funcionalidades como qualificadores, alternativas, decoradores, interceptadores e eventos que permitem uma grande flexibilidade no desenvolvimento da aplicação (DEVMEDIA, 2015a).

Já os objetos de acesso a dados (ou simplesmente DAO, acrônimo de *Data Access Object*), são um padrão para persistência de dados que permitem separar regras de negócio das regras de acesso a banco de dados. Numa aplicação que utilize a arquitetura MVC, todas as funcionalidades de bancos de dados, tais como obter as conexões, mapear objetos Java para tipos de dados SQL (*Structured Query Language, em português "Linguagem de Consulta Estruturada"*) ou executar comandos SQL, devem ser feitas por classes DAO.

A vantagem de usar objetos de acesso a dados é a separação simples e rigorosa entre duas partes importantes de uma aplicação que não devem e não podem conhecer quase que nada uma da outra, e que podem evoluir frequentemente e independentemente. Alterar a lógica de negócio pode esperar apenas a implementação de uma interface, enquanto que modificações na lógica de persistência não alteram a lógica de negócio, desde que a interface entre elas não seja modificada. Segue abaixo mais algumas vantagens:

- Pode ser usada em uma vasta porcentagem de aplicações;

[11] JSR – https://jcp.org/en/jsr/detail?id=299
[12] JCP – https://www.jcp.org/en/home/index
[13] EJB – https://pt.wikipedia.org/wiki/Enterprise_JavaBeans

- Esconde todos os detalhes relativos a armazenamento de dados do resto da aplicação;
- Atua como um intermediário entre a aplicação e o banco de dados;
- Mitiga ou resolve problemas de comunicação entre a base de dados e a aplicação, evitando estados inconsistentes de dados.

No contexto específico da linguagem de programação Java, um objeto de acesso a dados como padrão de projeto de software pode ser implementado de várias maneiras. Pode variar desde uma simples interface que separa partes de acesso a dados da lógica de negócio de uma aplicação até frameworks e produtos comerciais específicos.

Após diversos anos de reclamações sobre a complexidade na construção de aplicações com Java, a especificação Java EE 5 teve como principal objetivo a facilidade para desenvolver aplicações corporativas. O EJB 3 foi o grande percursor para essa mudança fazendo os Enterprise JavaBeans mais fáceis e mais produtivos de usar (DEVMEDIA, 2015c).

No caso dos Session Beans e Message-Driven Beans, a solução para questões de usabilidade foram alcançadas simplesmente removendo alguns dos mais onerosos requisitos de implementação e permitindo que os componentes sejam como POJOS (*Plain Old Java Object ou Velho e Simples Objeto Java.*

Já os Entity Beans eram um problema muito mais sério. A solução foi começar do zero. Deixou-se os Entity Beans sozinhos e introduziu-se um novo modelo de persistência. A versão atual da JPA (*Java Persistence API*) nasceu através das necessidades dos profissionais da área e das soluções proprietárias que já existiam para resolver os problemas com persistência. Com a ajuda dos desenvolvedores e de profissionais experientes que criaram outras ferramentas de persistência, chegou a uma versão muito melhor que é a que os desenvolvedores Java conhecem atualmente.

Dessa forma os líderes das soluções de mapeamento objetos-relacionais deram um passo adiante e padronizaram também os seus produtos. Hibernate[14] e TopLink[15] foram os primeiros a firmar com os fornecedores EJB.

O resultado final da especificação EJB finalizou com três documentos separados, sendo que o terceiro era o JPA. Essa especificação descrevia o modelo de persistência em ambos os ambientes Java SE (*Standard Edition*)[16] e Java EE.

No momento em que a primeira versão do JPA foi iniciada, outros modelos de persistência ORM (*Object Relational Mapping, em português "Mapeamento Objeto Relaci-*

[14] Hibernate – http://hibernate.org/
[15] TopLink – http://www.oracle.com/technetwork/middleware/toplink/overview/index.html
[16] Java SE – http://www.oracle.com/technetwork/pt/java/javase/overview/index.html

onal") já haviam evoluído. Mesmo assim muitas características foram adicionadas nesta versão e outras foram deixadas para uma próxima versão.

A versão JPA 2.0 incluiu um grande número de características que não estavam na primeira versão, especialmente as mais requisitadas pelos usuários, entre elas a capacidade adicional de mapeamento, expansões para a Java Persistence Query Language (JPQL), a API Criteria para criação de consultas dinâmicas, entre outras características.

Entre as principais inclusões na JPA destacam-se (DEVMEDIA, 2015c):

- **POJOS Persistentes:** Talvez o aspecto mais importante da JPA seja o fato que os objetos são POJOs, significando que os objetos possuem design simples que não dependem da herança de interfaces ou classes de frameworks externos. Qualquer objeto com um construtor default pode ser feito persistente sem nenhuma alteração numa linha de código. Mapeamento Objeto-Relacional com JPA é inteiramente dirigido a metadados. Isto pode ser feito através de anotações no código ou através de um XML definido externamente;
- **Consultas em Objetos:** As consultas podem ser realizadas através da Java Persistence Query Language (JPQL), uma linguagem de consulta que é derivada do EJB QL e transformada depois para SQL. As consultas usam um esquema abstraído que é baseado no modelo de entidade como oposto às colunas na qual a entidade é armazenada;
- **Configurações simples:** Existe um grande número de características de persistência que a especificação oferece, todas são configuráveis através de anotações, XML ou uma combinação das duas. Anotações são simples de usar, convenientes para escrever e fácil de ler. Além disso, JPA oferece diversos valores defaults, portanto para já sair usando JPA é simples, bastando algumas anotações;
- **Integração e Teste:** Atualmente as aplicações normalmente rodam num Servidor de aplicação, sendo um padrão do mercado hoje. Testes em servidores de aplicação são um grande desafio e normalmente impraticáveis. Efetuar teste de unidade e teste caixa branca em servidores de aplicação não é uma tarefa tão trivial. Porém, isto é resolvido com uma API que trabalha fora do servidor de aplicação. Isto permite que a JPA possa ser utilizada sem a existência de um servidor de aplicação. Dessa forma, testes unitários podem ser executados mais facilmente.

O Hibernate provê três tipos diferentes para retornar informações, uma é a HQL, outra forma é através de consultas nativas e a terceira forma é através da API Criteria (DEVMEDIA, 2015b).

A API Criteria nos permite construir consultas estruturadas utilizando Java, e ainda provê uma checagem de sintaxe em tempo de compilação que não é possível com

uma linguagem de consulta como HQL (*Hibernate Query Language*) ou SQL. A API Criteria também fornece projeção, agregação e outras funcionalidades. Possui a interface `org.hibernate.Criteria` que está disponível com todos os métodos fornecidas pela API. Através dessa API podemos construir consultas de forma programática. A interface `Session` do Hibernate contém vários métodos `createCriteria()` que recebem como parâmetro uma classe de um objeto persistente ou o nome da entidade e, dessa forma, o Hibernate irá criar um objeto Criteria que retorna instâncias das classes de objetos persistentes quando a aplicação executa uma consulta com a API.

A API Criteria também permite que possamos criar restrições onde restringimos os objetos retornados, podendo ter mais que uma restrição para uma consulta. Embora possamos criar nossos próprios objetos implementando o objeto `Criterion` ou estendendo um objeto `Criterion` já existente, recomenda-se que os desenvolvedores usem esses objetos embutidos na lógica de negócio da aplicação. Também podemos criar nossa própria classe `Factory` que retorna instâncias do objeto `Criterion` do Hibernate apropriadamente configurados para as restrições da nossa aplicação.

2.3 FrameWeb

FrameWeb (*Framework-based Design Method for Web Engineering*) é baseada em metodologias e linguagens de modelagens bastante difundidas na área de Engenharia de Software sem, no entanto, impor nenhum processo de desenvolvimento específico. É esperado, apenas, que determinadas atividades comuns à maioria dos processos de software, como levantamento de requisitos, análise, projeto, codificação, testes e implantação, sejam conduzidas pela equipe de desenvolvimento (SOUZA, 2007).

O método, portanto, sugere um processo de software orientado a objetos contendo as fases descritas anteriormente, mas podendo ser estendido e adaptado pela equipe de desenvolvimento. A linguagem de modelagem UML (*Unified Modeling Language*) (BOOCH G.; RUMBAUGH, 2005) é utilizada durante todo o processo. Podemos dizer também que é um método de projeto para construção de sistemas de informação Web (*Web Information Systems – WISs*) baseado em *frameworks*. Abaixo apresentamos algumas motivações principais desse método:

- O uso de *frameworks* ou arquiteturas baseadas em *containers* similares a eles se tornou o padrão de fato para o desenvolvimento de aplicações distribuídas, em especial os baseados na plataforma Web;
- O uso de métodos que se adequam diretamente à arquitetura de software adotada promove uma agilidade maior ao processo, característica que é bastante desejada na maioria dos projetos Web (PRESSMAN, 2005).

Em linhas gerais, FrameWeb assume que determinados tipos de *frameworks* serão utilizados durante a construção da aplicação, define uma arquitetura básica para o WIS e propõe modelos de projeto que se aproximam da implementação do sistema usando esses *frameworks*.

Sendo um método para a fase de Projeto, não prescreve um processo de software completo. No entanto, sugere o uso de um processo de desenvolvimento que contemple as fases apresentadas na Figura 2. Para uma melhor utilização de FrameWeb, espera-se que sejam construídos diagramas de casos de uso e de classes de domínio (ou similares) durante as fases de Requisitos e Análise. Além disso, como já mencionado anteriormente, agilidade é uma característica desejada num processo de software para a Web e, portanto, sugere-se que princípios de agilidade sejam seguidos, em especial os seguintes propostos pela Modelagem Ágil (AMBLER S.; JEFFRIES, 2002):

- **Modele com um propósito:** crie apenas os modelos que adicionam informação útil;
- **Viaje com pouca bagagem:** ao passar de uma fase para outra no processo de desenvolvimento, alguns modelos precisarão ser adequados à nova fase. Leve apenas aqueles que serão úteis na fase seguinte;
- **Conteúdo é mais importante que apresentação:** utilize a notação e a linguagem mais adequadas ao objetivo principal de um modelo, que é a transmissão da informação;
- **Conheça os modelos e as ferramentas:** equipes de modelagem e desenvolvimento devem ser altamente proficientes nas linguagens e ferramentas utilizadas na modelagem;
- **Adapte-se localmente:** as equipes devem ser capazes de se adaptar às necessidades específicas de um projeto.

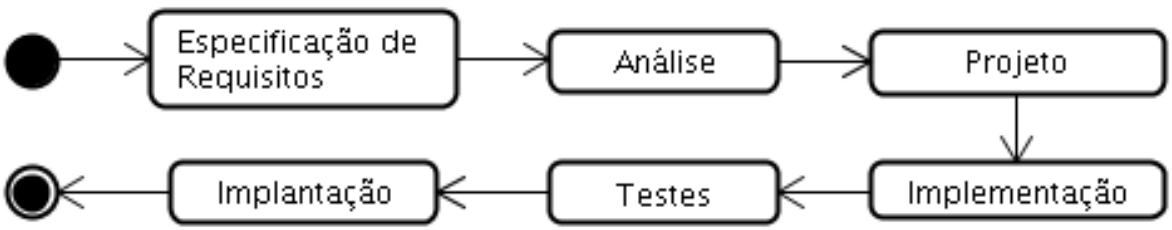

Figura 2 – Processo de Desenvolvimento de Software.

A fase de Projeto concentra as propostas principais do método: (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.

O FrameWeb define extensões leves (*lightweight extensions*) ao meta-modelo da UML para representar componentes típicos da plataforma Web e dos *frameworks* utilizados, criando um perfil UML que é utilizado para a construção de diagramas de quatro tipos:

- **Modelo de Domínio:** é um diagrama de classes da UML que representa os objetos de domínio do problema e seu mapeamento para a persistência em banco de dados relacional;
- **Modelo de Aplicação:** é um diagrama de classes da UML que representa as classes de serviço, que são responsáveis pela codificação dos casos de uso, e suas dependências;
- **Modelo de Navegação:** é um diagrama de classe da UML que representa os diferentes componentes que formam a camada de Lógica de Apresentação, como páginas Web, formulários HTML e classes controladoras;
- **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.

Esses quatro modelos serão descritos com maiores detalhes na Seção 4.2.

3 Especificação de Requisitos

No levantamento de requisitos, o minimundo foi definido da seguinte forma:

No DI/Ufes, é comum os professores que ministram alguma disciplina aplicarem trabalhos para os alunos que estão matriculados. O sistema que será desenvolvido busca automatizar alguns processos do meio acadêmico utilizando a informática para diminuir o trabalho manual do professor, assim como otimizar o seu tempo para que possa desenvolver outras atividades.

Utilizando o sistema o professor inicialmente deverá associar a disciplina ou cadastrá-la no sistema, caso ainda não exista, para depois fazer a associação. Feito isso, deverá fazer a importação com os dados dos alunos matriculados. Poderá também cadastrar cada trabalho com sua devida especificação (data de entrega, valor, etc.).

Também será possível criar grupos de trabalho que irão possuir uma quantidade máxima de alunos definida pelo professor. Os próprios alunos que serão responsáveis por criarem os seus grupos através do sistema SAP utilizando login e senha. O professor poderá cadastrar um valor de penalização para cada dia de atraso do recebimento do trabalho após a data de entrega. Além disso, o sistema também permitirá que o professor possa cadastrar e controlar as informações dos alunos orientados (iniciação científica, graduação, mestrado e doutorado), tanto os atuais quanto os já formados (histórico de egressos). Desta forma, o sistema auxilia também nas atividades de pesquisa dos professores e fornece a base para que, no futuro, sejam adicionadas funcionalidades que auxiliem no gerenciamento de grupos de pesquisa.

É interessante notar que na universidade há um sistema para gerenciamento de matrículas de alunos em disciplinas, com possibilidade de cadastro de diários de classe, envio de e-mail para alunos, registro de faltas e notas, etc. Por esse motivo, tais funcionalidades não foram incluídas no SAP. Idealmente, os sistemas seriam integrados, porém tal integração é algo muito difícil de acontecer, visto que o Núcleo de Tecnologia da Informação (NTI) não pode abrir seus sistemas e bases de dados para alunos e professores que não fazem parte de seu quadro de funcionários. Portanto, trabalharemos a princípio com o fato que os sistemas são disjuntos, promovendo integração manual sempre que possível (ex.: leitura de arquivos estruturados com a lista de alunos matriculados para uma disciplina, produzido pelo sistema da Ufes).

Todos os requisitos funcionais, não-funcionais e as regras de negócios estão listados no **Documento de Requisitos** que está disponível no Apêndice ao final dessa monografia.

O sistema SAP foi dividido em dois subsistemas, conforme a Tabela 1.

Tabela 1 – Subsistemas.

Subsistema	**Descrição**
Controle Trabalho	Envolve as funcionalidades relacionadas ao controle dos estudantes, trabalhos, turmas e disciplinas.
Controle Laboratório	Envolve as funcionalidades relacionadas ao controle dos professores, orientados e grupos de pesquisa.

Os atores identificados no contexto do sistema SAP estão definidos na Tabela 2.

Tabela 2 – Atores.

Ator	**Descrição**
Professor	Representa os professores do Departamento de Informática da Universidade Federal do Espírito Santo (DI/Ufes).
Administrador	Possui as mesmas funcionalidades do Professor, além de funções de administração do sistema como, por exemplo, cadastro de professores e grupos de pesquisa.
Estudante	Representa os estudantes cadastrados no sistema.

A seguir, veremos os diagramas de casos de uso e de classes do sistema SAP produzidos durante a atividade de análise dos requisitos. Os documentos de requisitos e de especificação de requisitos completos podem ser encontrado nos apêndices desta monografia.

3.1 Diagrama de Casos de Uso

A Figura 3 mostra os casos de uso do subsistema **Controle Trabalho** que serão descritos abaixo.

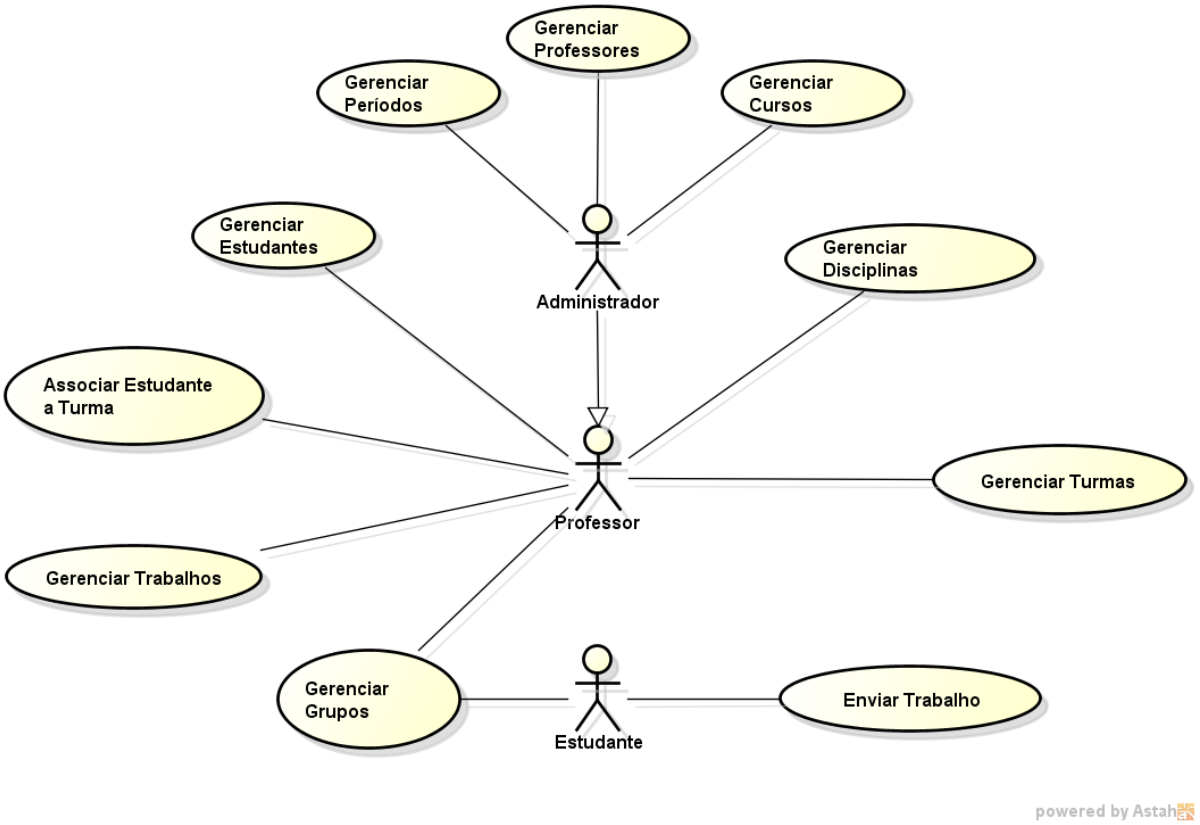

Figura 3 – Controle Trabalho - Diagrama de Casos de Uso.

O subsistema **Controle Trabalho** foi criado para gerenciar todos os dados e funcionalidades das turmas, incluindo estudantes, trabalhos e grupos de trabalhos. A maior parte do sistema que foi desenvolvido está nesse subsistema.

O meio acadêmico possui cursos, por exemplo: ciência da computação e engenharia da computação. Com isso, foi criado o caso de uso **Gerenciar Cursos** para que o professor possa, por exemplo, inserir novos cursos. Para inserir um novo curso, basta informar o código e o nome do mesmo.

Além dos cursos, temos também os períodos, que no caso da Ufes são dois por ano, um para cada semestre. Com isso, criamos o caso de uso **Gerenciar Períodos** para realizar esse controle. As informações de um período são: ano, número, data inicial e data final.

Uma informação crucial para o sistema são os professores, que serão controlados pelo caso de uso **Gerenciar Professores**. Nesse caso, deve ser informado: nome, email (será o login para acessar o sistema), senha e indicar se terá perfil de administrador no sistema ou não. Todos os professores efetivos do DI/Ufes já estão cadastrados previamente no sistema.

Os casos de uso citados anteriormente podem ser realizados apenas por professores que possuem o perfil de administrador. Já os casos de uso que serão citados abaixo

podem ser realizados por qualquer professor. O subsistema Controle Laboratório também possui alguns casos de uso que são realizados apenas por administradores e serão descritos posteriormente.

Uma das peças chaves do sistema são os estudantes, que serão controlados pelo caso de uso **Gerenciar Estudantes**. As informações necessárias de um estudante são: matrícula, nome, e-mail, senha e telefone. Os estudantes farão parte das turmas, grupos de trabalhos e servirão de base para criar uma orientação que serão explicados posteriormente.

O caso de uso **Gerenciar Disciplinas** foi criado para o professor poder controlar as disciplinas que serão lecionadas. Nesse caso, basta informar o código e o nome da disciplina.

Um dos casos de uso mais importantes é o **Gerenciar Turmas**, que será o responsável por controlar os dados de uma turma. Esse caso de uso irá usar uma série de informações obrigatórias de outros casos de uso, por exemplo, período. Com isso, caso não exista nenhum período cadastrado no sistema, não será possível criar uma turma. As informações necessárias para uma turma são: período, professor, curso, disciplina, número da turma e os estudantes. Nesse caso, apenas a informações do número da turma poderá ser digitada, enquanto as demais informações são carregadas e deverão ser apenas selecionadas.

Após ter as turmas criadas, o professor poderá utilizar o caso de uso **Associar Estudante a Turma** para incluir estudantes em uma turma. Essa operação poderá ser feita de duas formas distintas:

- No próprio cadastro da turma, o professor poderá selecionar o estudantes e incluir ou retirar o mesmo da turma;
- Importar os dados de uma planilha com os campos previamente definidos, devendo selecionar o período e a turma desejada.

Normalmente os professores aplicam trabalhos nas turmas que administram. Com isso, foi criado o caso de uso **Gerenciar Trabalhos** para fazer esse controle. As informações de um trabalho são: período, turma, número do trabalho, assunto, data de entrega, valor, número máximo de participantes e valor de desconto por atraso (opcional). Como podemos perceber, esse caso de uso é o que mais possui informações. Esse caso de uso também será utilizado pelos estudantes, mas apenas para visualização das informações.

Os trabalhos serão formados por grupos de alunos, que serão controlados pelo caso de uso **Gerenciar Grupos**. Nesse caso, devemos informar: período, turma, trabalho, número do grupo e selecionar os estudantes que farão parte do mesmo. Esse caso de uso será utilizado pelos professores e os próprios estudantes. Entretanto, apenas os professores

possuem acesso para cadastrar mais estudantes no grupo do que o permitido (configurado no cadastro do trabalho).

A Figura 4 mostra os casos de uso do subsistema **Controle Laboratório** que serão descritos abaixo. Esse subsistema é muito menor em relação ao subsistema Controle Trabalho, pois o foco da monografia está no controle das turmas e trabalhos, mas ele poderá servir como base para melhorias futuras.

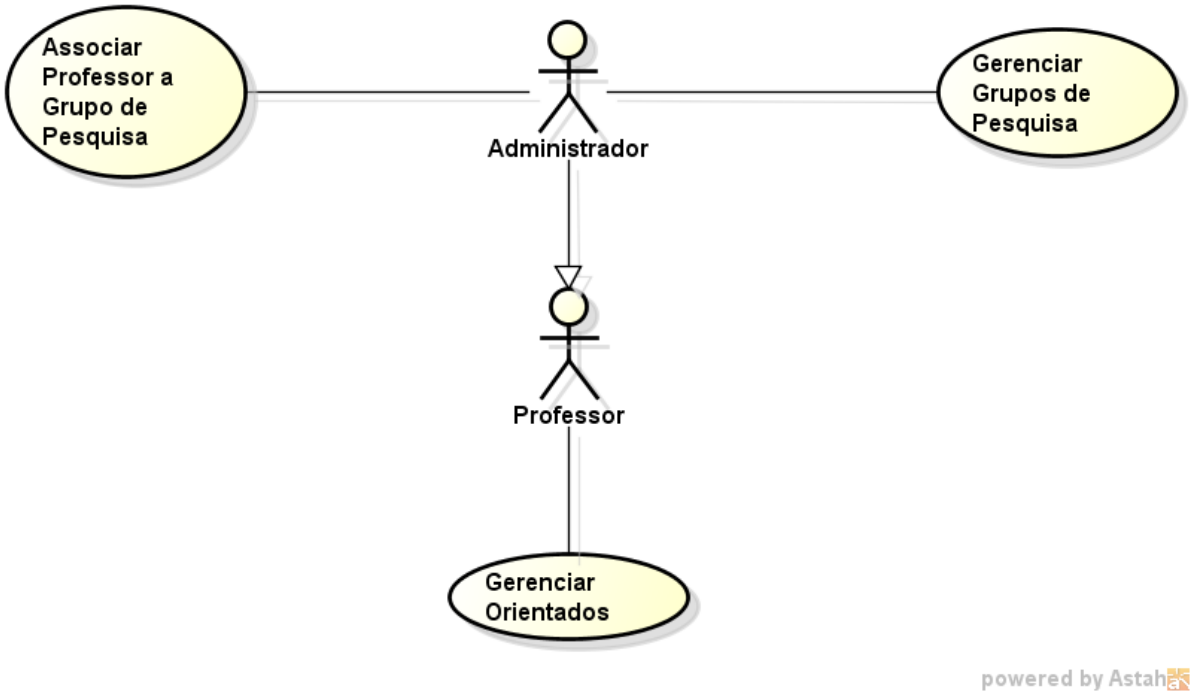

Figura 4 – Controle Laboratório - Diagrama de Casos de Uso.

Foi criado também o conceito de grupo de pesquisa, que nada mais é do que a reunião de alguns professores em um grupo para realizar pesquisas. Com isso, foi criado o caso de uso **Gerenciar Grupos de Pesquisa**, onde deve ser informado: nome do grupo, site e os professores que participam do mesmo. Além disso, o caso de uso **Associar Professor a Grupo de Pesquisa** será responsável por incluir ou remover professores dos grupos de pesquisa. Esses casos de uso serão utilizados apenas pelos professores que possuem perfil de administrador.

Por fim, o caso de uso **Gerenciar Orientados** foi criado para permitir que qualquer professor gerencie os estudantes orientados. As informações necessárias nesse caso são: estudante (previamente cadastrado no sistema que será apenas selecionado), tema, tipo da orientação, orientador, co-orientador, data inicial e data final.

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. A Tabela 3 resume os casos de uso que foram citados anteriormente e que podem ser realizados por cada um dos atores do sistema.

Tabela 3 – Atores x Casos de Uso.

Ator	**Casos de Uso**
Professor	Gerenciar Disciplinas, Gerenciar Turmas, Gerenciar Estudantes, Gerenciar Trabalhos, Gerenciar Grupos, Associar Estudante a Turma e Gerenciar Orientados.
Administrador	Além dos casos de uso do professor, os administradores incluem: Gerenciar Professores, Gerenciar Períodos, Gerenciar Cursos, Gerenciar Grupos de Pesquisa e Associar Professor a Grupo de Pesquisa.
Estudante	Gerenciar Grupos, Visualizar Trabalho e Enviar Trabalho.

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

3.2 Diagrama de Classes

A Figura 5 exibe o diagrama de classes do subsistema **Controle Trabalho**. Assim como no diagrama casos de uso, o diagrama de classes do subsistema Controle Trabalho também possui a maior parte das classes do sistema.

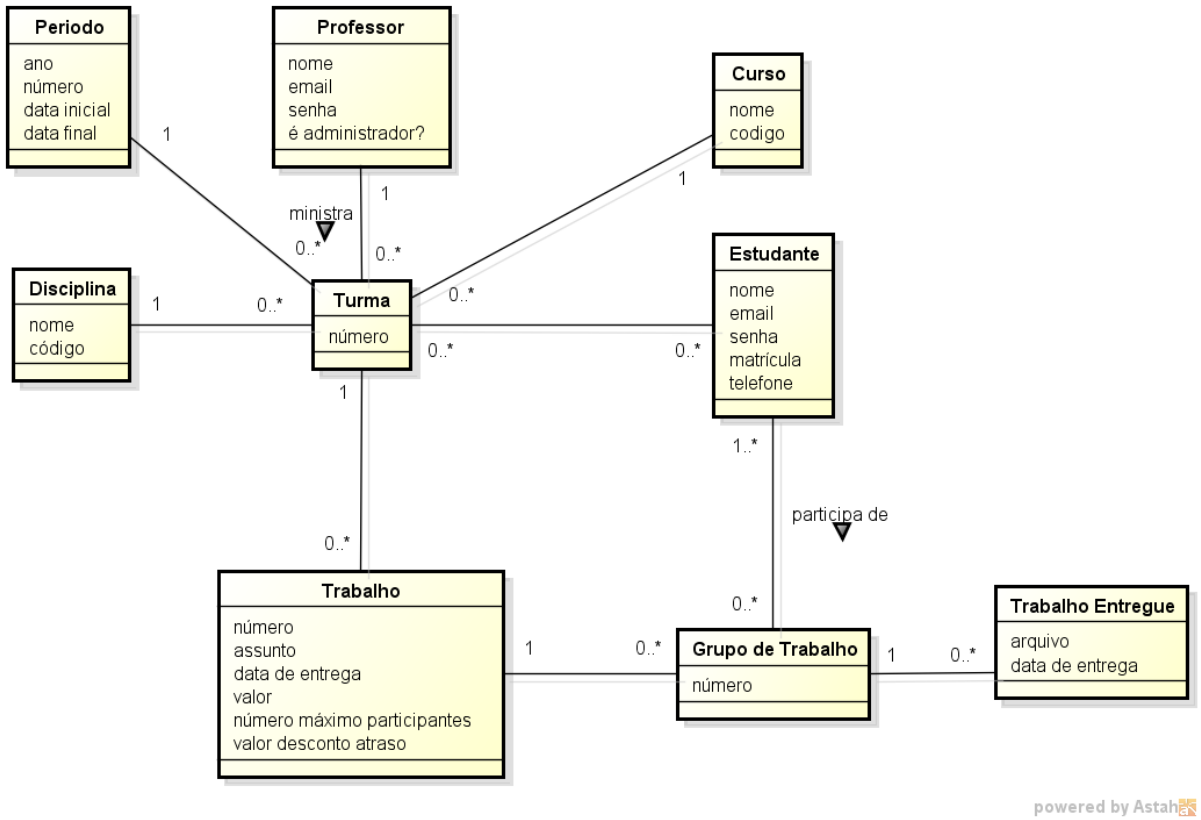

Figura 5 – Controle Trabalho - Diagrama de Classes.

Podemos verificar facilmente que a classe mais importante é a classe Turma, pois

ela possui a maior quantidade de ligações com as demais classes. É obrigatório que uma classe Turma possua uma ligação com as classes Período, Professor, Curso e Disciplina. Entretanto, com as classes Trabalho e Estudante essa ligação já é opcional e pode ocorrer mais de uma vez. Já as classes Período, Professor, Curso, Disciplina e Estudante podem ter registros no sistema e mesmo assim não estarem ligadas a nenhuma outra classe.

Toda classe Trabalho deve estar associada a uma classe Turma. Vale ressaltar que nesse caso, quando for preciso descobrir quem é o professor que ministra uma turma ou o período de um determinado trabalho, essa informação não estará presente na classe Trabalho mas sim na classe Turma. Um trabalho pode possuir vários grupos ou até mesmo nenhum. Já a classe Grupo de Trabalho deve estar ligada a um trabalho e deve possuir pelo menos um estudante. Por fim, a classe Trabalho Entregue estará associada a uma classe Grupo de Trabalho.

Podemos ressaltar aqui também algumas restrições de integridade, sendo elas:

- Um estudante não pode participar de mais de um grupo de trabalho para o mesmo trabalho de uma turma;
- Um trabalho não pode ser entregue após o término do período letivo da turma;
- Um grupo de trabalho não pode ter mais participantes do que o valor definido na especificação do trabalho, exceto nos casos que forem alterados pelo professor;
- Um trabalho não pode ser entregue por um estudante que não esteja matriculado na turma;
- A data final de um período não pode ser anterior à data inicial do mesmo.

A Figura 6 exibe o diagrama de classes do subsistema **Controle Laboratório**. Esse diagrama possui poucas classes se for comparado com o subsistema Controle Trabalho, pois o subsistema Controle Laboratório tem poucas funcionalidades.

Podemos notar que as classes Professor e Estudante foram referenciadas do subsistema Controle Trabalho. Portanto, fazem parte desse susbistema as classes Grupo de Pesquisa e Orientação. A classe Grupo de Pesquisa pode ter vários ou até nenhum professor. Enquanto a classe Orientação deve possuir um Estudante e um Professor, podendo ter ainda um co-orientador que também é um professor mas é opcional.

Esse diagrama possui as seguintes restrições de integridade:

- Um professor não pode ser um orientador e coorientador ao mesmo tempo em uma orientação;
- A data final de uma orientação não pode ser anterior à data inicial da mesma.

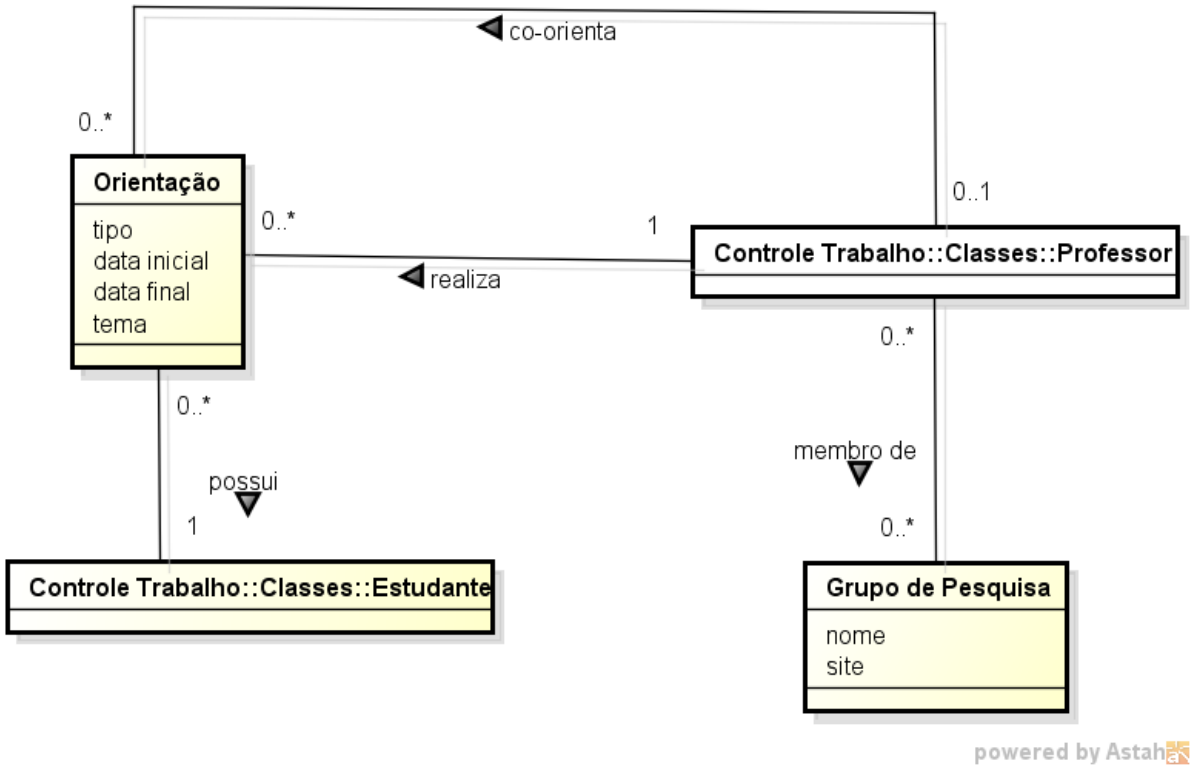

Figura 6 – Controle Laboratório - Diagrama de Classes.

Maiores informações e detalhes sobre o diagrama de classes poderão ser consultados no Documento de Especificação de Requisitos que está disponível no Apêndice ao final dessa monografia.

4 Projeto Arquitetural e Implementação

Nesta seção iremos mostrar a arquitetura do projeto, assim como algumas partes de sua implementação e apresentar as principais telas do sistema. A apresentação do projeto do SAP foi dividida em três partes: arquitetura, modelos FrameWeb e o framework *nemo-utils*.

4.1 Arquitetura do Sistema

No projeto arquitetural, o SAP foi dividido em dois módulos principais (implementados como pacotes Java), seguindo a divisão de subsistemas feita na análise dos requisitos e apresentada no Capítulo 3. A Figura 7 mostra os módulos que formam a arquitetura do SAP.

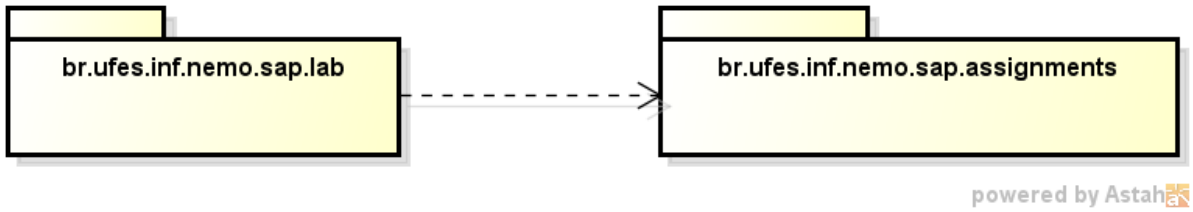

Figura 7 – Pacotes que formam a arquitetura do SAP.

O módulo `br.ufes.inf.nemo.sap.assignments` contém as funcionalidades do subsistema **Controle Trabalho**, enquanto o módulo `br.ufes.inf.nemo.sap.lab` contém as funcionalidades do subsistema **Controle Laboratório**. Mais à frente iremos detalhar um pouco mais as subdivisões desses módulos.

Os módulos principais do SAP são ainda subdivididos em camadas segundo a arquitetura que pode ser verificada na Figura 8. O sistema SAP foi divido em três camadas, sendo elas: apresentação (*Presentation Tier*), negócio (*Business Tier*) e acesso a dados (*Data Access Tier*). A camada de apresentação por sua vez foi subdividida em duas partes, visão (*View*) e controle (*Control*). A camada de negócio também foi subdividida em duas partes, domínio (*Domain*) e aplicação (*Application*). Por fim, a camada de acesso a dados possui uma única parte responsável pela persistência (*Persistence*). A seguir iremos detalhar um pouco mais cada uma dessas camadas.

A **camada de apresentação** foi subdividida em visão e controle. A parte da visão é formada pelas páginas Web que serão detalhadas no final da seção. A parte de controle contém os controladores que realizam a comunicação entre a interface e a aplicação.

A **camada de negócio** foi subdividida em domínio e aplicação. A parte do domínio

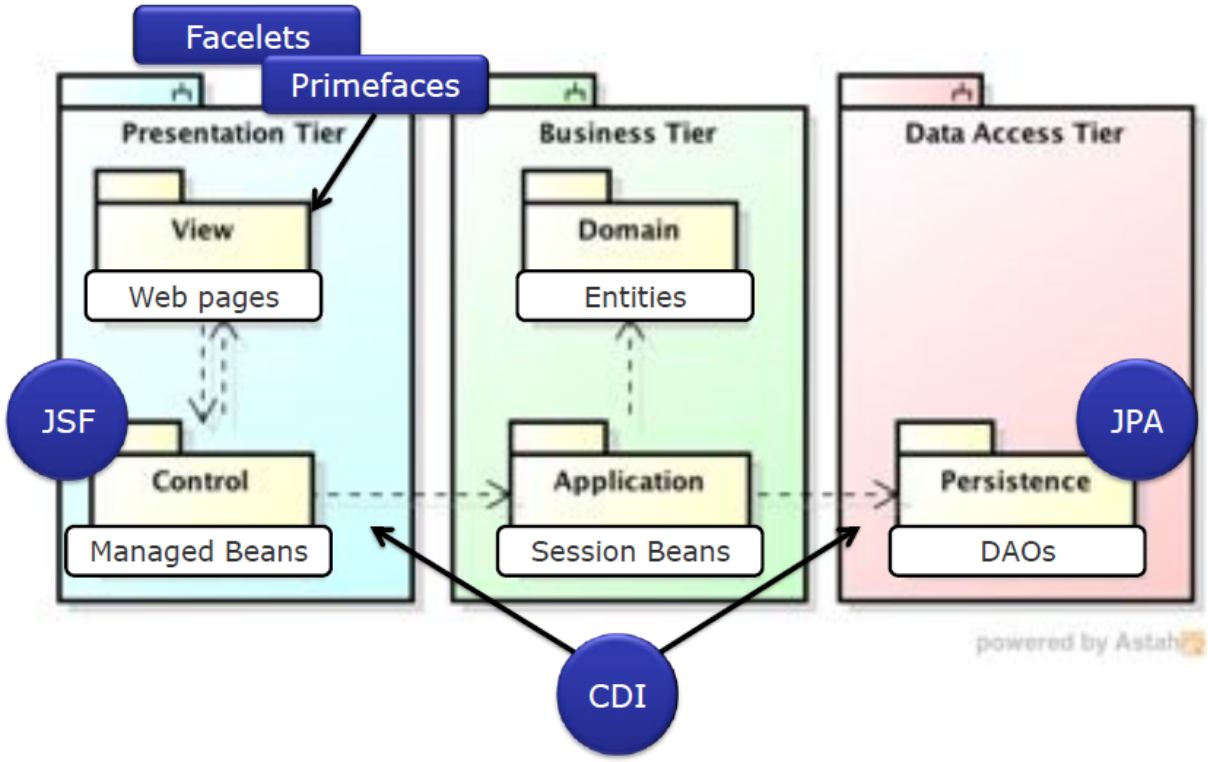

Figura 8 – SAP - Arquitetura - Sistema.

é formada pelas entidades do negócio, enquanto a aplicação contém as validações dos dados e implementação dos casos de uso (funcionalidades do sistema).

A **camada de acesso a dados** contém os objetos responsáveis por fazer a comunicação com o banco de dados. Esses objetos são conhecidos como DAO (*Data Access Object* ou objeto de acesso a dados) e são um padrão para persistência de dados que permite separar regras de negócio das regras de acesso a banco de dados. Eles serão responsáveis por armazenar e recuperar os dados do banco de dados.

A Figura 8 mostra, também, as tecnologias Java associadas a cada pacote. Tais tecnologias foram abordadas na Seção 2.2.

Iremos falar agora sobre os pacotes que formam cada uma das camadas da arquitetura do sistema SAP.

A Figura 9 exibe os pacotes do sistema SAP. Como podemos perceber, os pacotes foram agrupados pelos módulos principais e pelas camadas da arquitetura que foram explicados anteriormente. A seguir iremos detalhar um pouco mais cada um deles.

O processo de transformar a aplicação Web para que suporte mais que um idioma e formato de dados é conhecido como **internacionalização, ou i18n**. Para internacionalizar uma aplicação são utilizados arquivos de propriedades, que contêm pares de chave-valor. Para cada localidade é criado um arquivo de propriedades, com a tradução correspondente. O conteúdo da página será exibido de acordo com o idioma configurado no navegador.

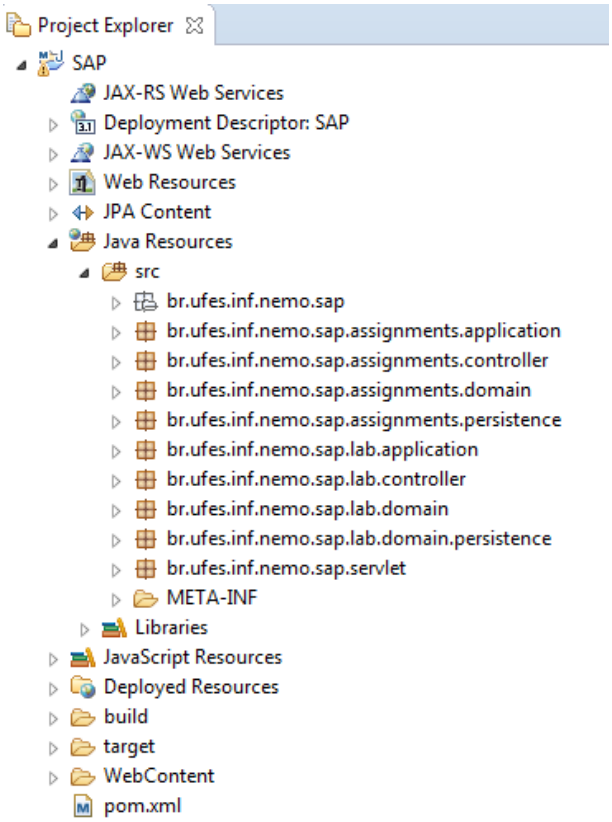

Figura 9 – SAP - Implementação - Pacotes.

No caso do sistema SAP, o pacote `br.ufes.inf.nemo.sap` contém os arquivos de propriedade `faces_pt_BR.properties` e `messages_pt_BR.properties` com as traduções para a língua portuguesa. Essas propriedades também devem ser definidas dentro do arquivo `faces-config.xml` que fica dentro da subpasta `WEB-INF` da pasta `WebContent` no projeto SAP.

O pacote `br.ufes.inf.nemo.sap.assignments.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 ou modificar uma entidade.

O pacote `br.ufes.inf.nemo.sap.assignments.domain` contém a definição das entidades do sistema SAP. Cada uma dessas entidades está definida em um arquivo `*.java` e já realiza o mapeamento objeto-relacional para o banco de dados. Através desse mapeamento, o Hibernate irá criar 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. São através dessas anotações que as tabelas associativas serão criadas automaticamente pelo JPA/Hibernate no banco de dados. Por um lado essa é uma boa prática, visto que diminui consideravelmente o esforço do desenvolvedor, entretanto, a

estrutura criada pode não ser a mais otimizada se for analisada por administradores de banco de dados (DBA, *DataBase Administrator*). 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.

O pacote `br.ufes.inf.nemo.sap.assignments.domain.controller` contém os arquivos que implementam a camada de controle, enquanto o pacote `br.ufes.inf.nemo.sap.assignments.persistence` contém os arquivos que implementam a camada de persistência.

Por fim, o pacote `br.ufes.inf.nemo.sap.servlet` contém apenas o arquivo `ErrorPageFilter.java` que é um filtro responsável por fazer o tratamento de sessão expirada. Essa configuração deve ser feita também no arquivo `web.xml` que fica dentro da subpasta `WEB-INF` da pasta `WebContent` no projeto SAP. Nesse arquivo também é configurado a página inicial do sistema.

Sobre a arquitetura do banco de dados, conforme explicado anteriormente, o sistema SAP utiliza o Hibernate para fazer o mapeamento objeto relacional, através desse mapeamento o JPA/Hibernate irá criar os objetos no banco de dados automaticamente. Com isso, foi utilizada a anotação `@Entity` nas classes do domínio para realizar a persistência dos dados. As tabelas do banco de dados são exibidas na Figura 10.

Podemos notar que o nome das tabelas são iguais aos nomes das entidades, exceto nas tabelas associativas que são utilizadas para fazer os mapeamentos muitos para muitos. Por exemplo, uma turma pode possuir vários estudantes, assim como um estudante pode estar matriculado em diversas turmas. Nesses caso, foi necessária a criação da tabela `schoolroom_student` para armazenar essa informação. Na Figura 11 podemos observar que na turma com `id` **191** estão matriculados os estudantes com `ids` **192, 193, 194, 195 e 196**.

Além disso, o Hibernate cria automaticamente uma tabela chamada `hibernate_sequence` que possui uma única coluna chamada `next_val`, que contém o número do `id` que será utilizado para a próxima entidade criada pelo sistema. Vale ressaltar também que todas as entidades possuem uma coluna chamada `uuid` que é um número gerado aleatoriamente para diferenciar unicamente uma entidade, e uma coluna chamada `version` que é inicialmente preenchida com 0 e esse valor é incrementado posteriormente caso a entidade tenha algum de seus atributos atualizados. Por fim, a configuração do Hibernate fica no arquivo `persistence.xml` dentro de `JPA Content` no projeto SAP.

Vamos falar agora sobre a estrutura Web do sistema SAP, onde as páginas Web fazem parte da visão da camada de apresentação. A organização dessas páginas pode ser

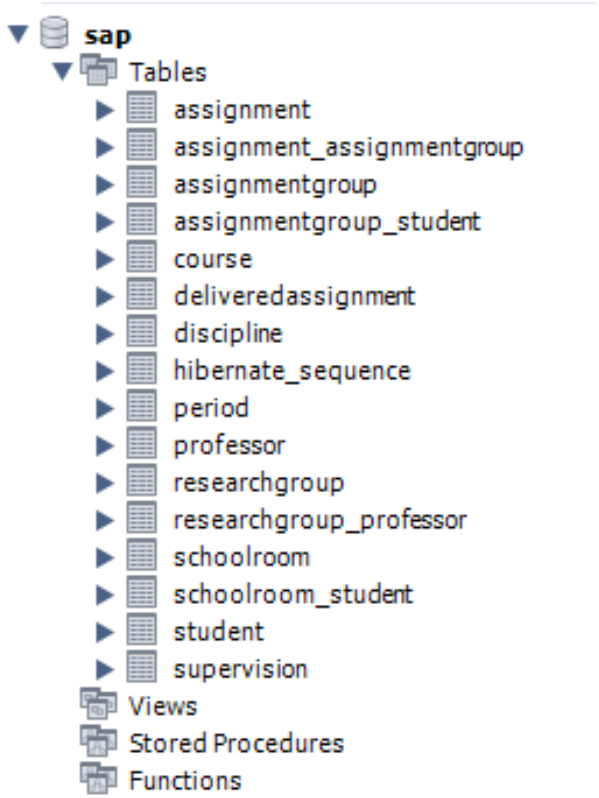

Figura 10 – SAP - Arquitetura - Banco de Dados.

SchoolRoom_id	students_id
191	192
191	193
191	194
191	195
191	196

Figura 11 – SAP - Tabela Associativa.

verificada na Figura 12. Existe uma pasta raiz chamada `WebContent` que contém todos os arquivos da visão. Ela possui duas subpastas que representam os módulos do SAP: `assignments` e `lab`. Dentro de cada uma dessas, uma nova pasta foi criada para tratar cada caso de uso de forma separada. Isso ajuda na organização e caso seja necessário criar um novo caso de uso, basta adicionar uma nova pasta e os arquivos necessários.

As pastas que implementam os casos de uso seguem um padrão que foi definido no framework do *nemo-utils* (cf. Seção 4.3). 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 da entidade que estão armazenadas no banco de dados. Por exemplo, podemos verificar essas

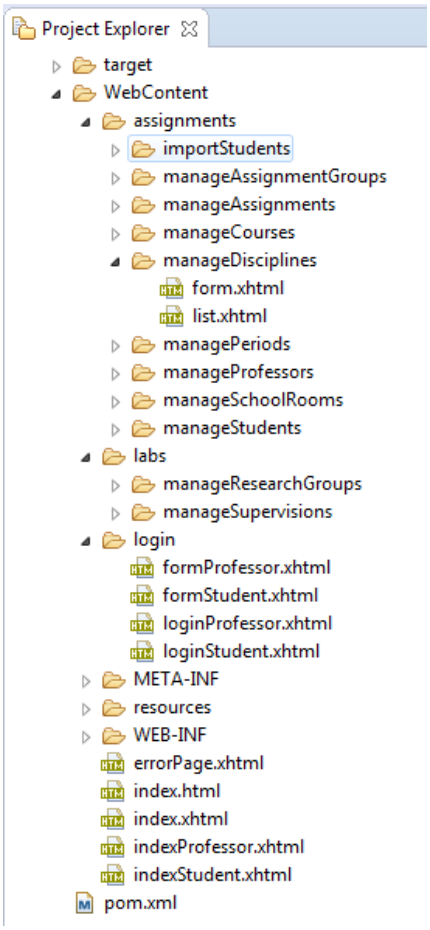

Figura 12 – SAP - Implementação - Páginas Web

duas páginas dentro da pasta `manageDisciplines` na Figura 12.

A pasta `login` contém uma página para realizar o login e outra para poder alterar os dados do usuário. Essas páginas são diferentes tanto para os professores quanto para os estudantes.

Ainda dentro da pasta raiz `WebContent`, temos a página `errorPage.xhtml` que será utiliza no tratamento de sessão expirada. Temos também as páginas `index.html` e `index.xhtml` que são as páginas iniciais do sistema. Por fim, temos as páginas `indexProfessor.xhtml` e `indexStudent.xhtml` que são as páginas iniciais dos professores e estudantes após a realização do login. Uma observação importante é que tanto os professores como os administradores utilizam a mesma página inicial, porém os menus são diferentes devido ao `decorator.xhtml` (será explicado a seguir).

A pasta `resources` contém a subpasta `decorator`. O decorador será utilizado para definir o layout da página e o menu que está sendo utilizado. Como existem diferentes tipos de usuários para o sistema, foram criados diferentes decoradores para atender a cada tipo de usuário. A pasta `default` contém o decorador quando o usuário ainda não está logado no sistema. Já as pastas `administrator`, `professor` e `student` definem o layout depois que o usuário realiza o login no sistema. Vale ressaltar que o administrador e o

professor utilizam a mesma tela para fazer o login, e somente após fazer a verificação do usuário que o sistema irá verificar qual decorador será utilizado para poder diferenciar os menus.

4.2 Modelos FrameWeb

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

Vamos começar falando sobre o **Modelo de Domínio** que é um diagrama de classes da UML que representa os objetos de domínio do problema e seu mapeamento para a persistência em banco de dados relacional. A partir dele são implementadas as classes da camada de domínio na atividade de implementação. Os passos para sua construção são (SOUZA, 2007):

- A partir do modelo de classes construído na fase de análise de requisitos, adequar o modelo à plataforma de implementação escolhida, indicando os tipos de dados de cada atributo, promovendo a classes atributos que devam ser promovidos, definindo a navegabilidade das associações etc.
- Adicionar os mapeamentos de persistência.

Os mapeamentos de persistência são meta-dados das classes de domínio que permitem que os *frameworks* ORM (*object-relational mapping*) 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.

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

Todas as classes de domínio estendem de *PersistentObjectSupport* do framework *nemo-utils*, sendo que essa herança não é mostrada no diagrama acima com o intuito de não polui-lo com várias associações. As Figuras 15 e 16 são os modelos de domínio para os módulos “Controle Trabalho” e “Controle Laboratório”, respectivamente.

Já o **Modelo de Navegação** é um diagrama de classe da UML que representa

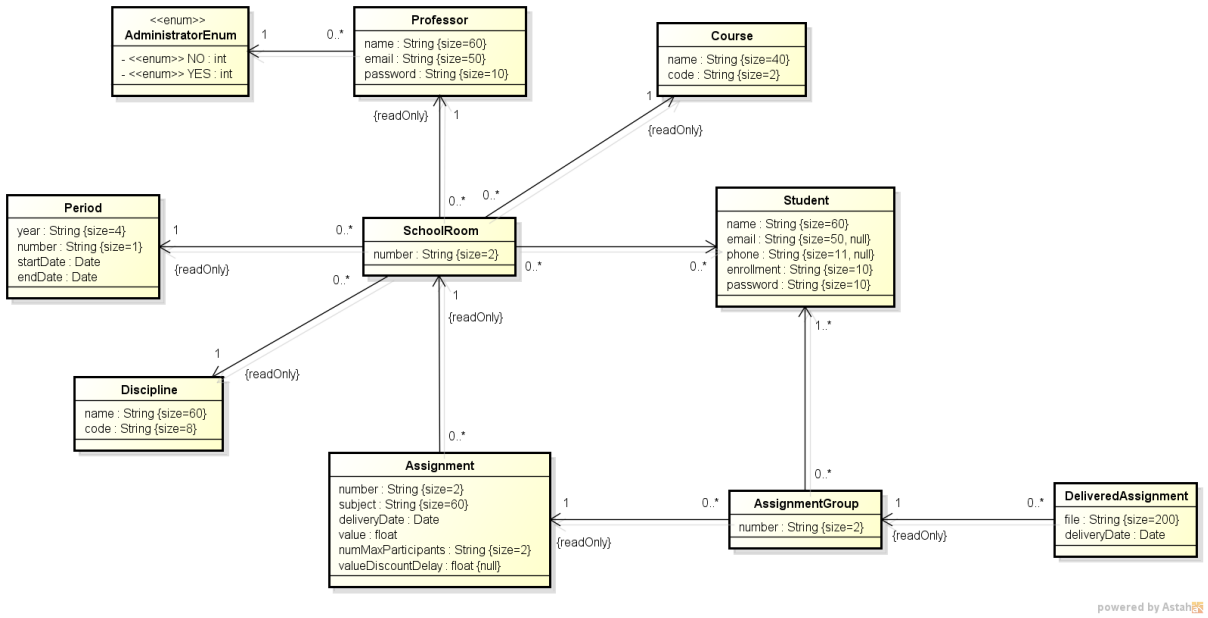

Figura 13 – FrameWeb - Controle Trabalho - Modelo Domínio.

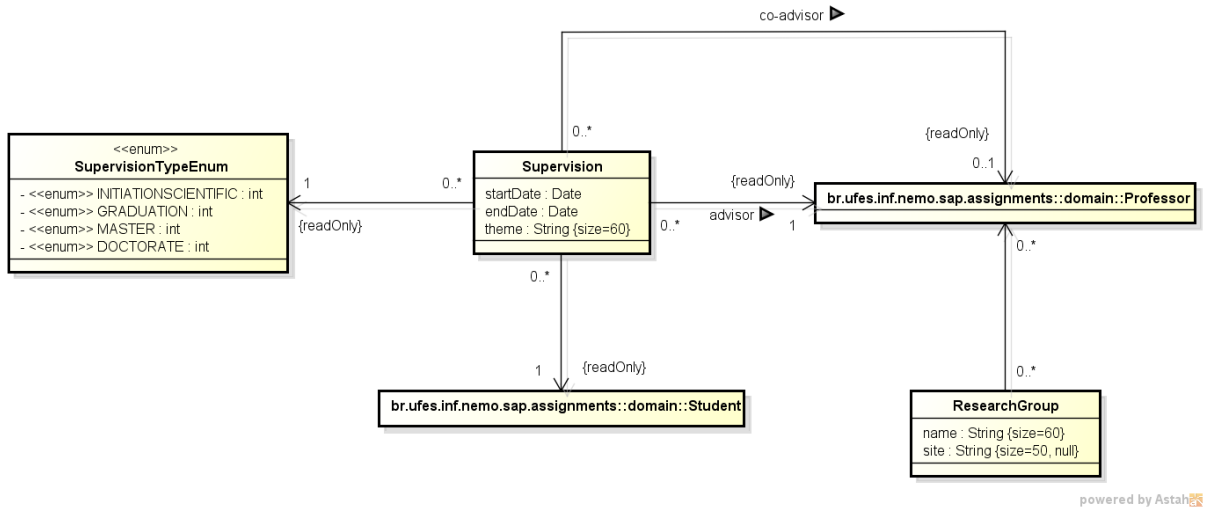

Figura 14 – FrameWeb - Controle Laboratório - Modelo Domínio.

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**.

Para páginas Web e modelos, atributos representam informações que são exibidas ao usuário na página. Os tipos possíveis são os mesmos tipos definidos pela plataforma de implementação. Relacionamentos de dependência entre páginas e modelos indicam um link HTML entre as mesmas, enquanto associações de composição entre páginas ou modelos e formulários denotam a presença daquele formulário dentro da página ou modelo.

Em formulários HTML, atributos representam campos do formulário, que devem ter seus tipos definidos com o nome do campo segundo o padrão HTML (ex.: input, checkbox,

button 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 a utilização da ferramenta *nemo-utils*. Esse fluxo de execução similar é representado pela Figura 13 que é um modelo de apresentação genérico.

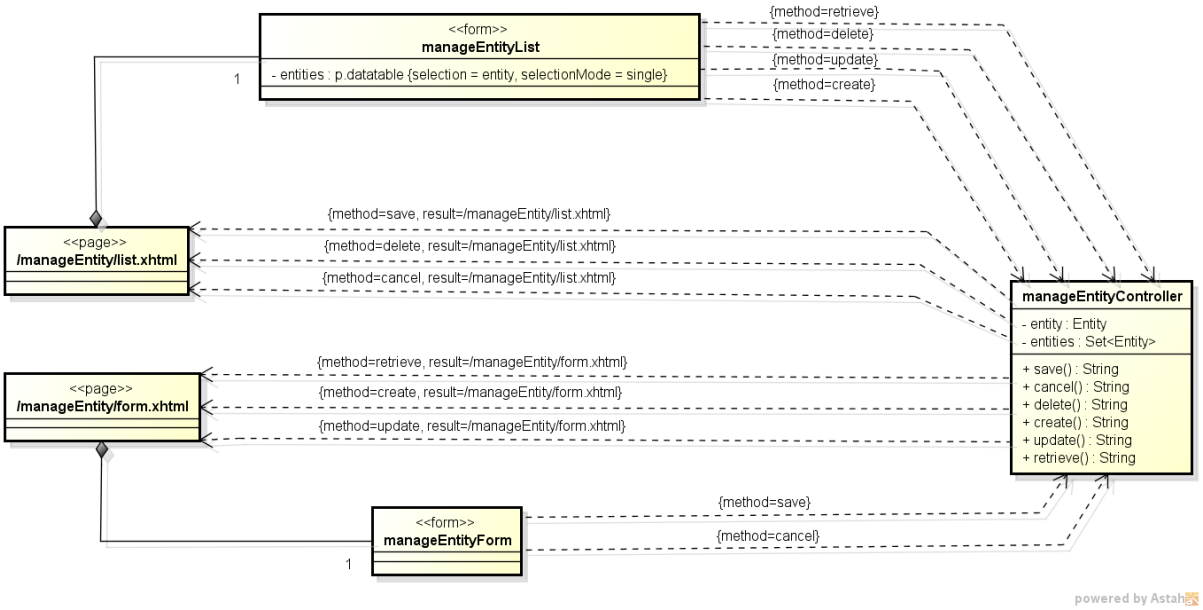

Figura 15 – FrameWeb - *nemo-utils* - Modelo Navegação.

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 14 é um modelo de navegação para o caso de uso "Associar Estudante a Turma". Nesse caso, podemos perceber que o modelo possui uma classe para representar a página web (esteriótipo `«page»`), outra para representar um formulário dentro dessa página (esteriótipo `«form»`) que possui alguns componentes como atributos. Por último, temos uma classe chamada `ImportStudentsController` para representar o controlador, que também possui ligações de dependência com o formulário representando a submissão dos dados e a chamada da funcionalidade AJAX.

No **Modelo de Aplicação** é um diagrama de classes da UML que representa as classes de serviço, que são 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. Os passos para a construção do Modelo de Aplicação são (SOUZA, 2007):

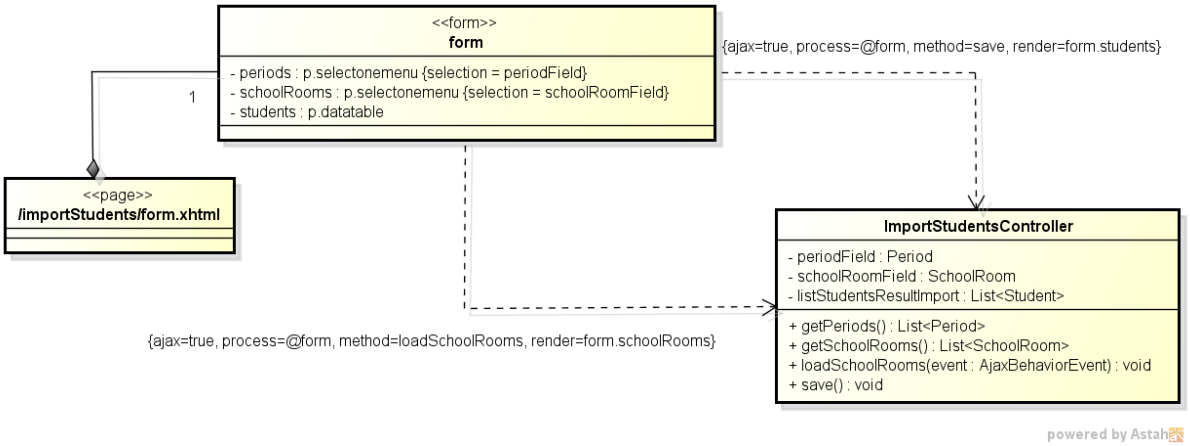

Figura 16 – FrameWeb - Importar Estudantes - Modelo Navegação.

- Analisar os casos de uso modelados durante a Especificação de Requisitos, definir a granularidade das classes de serviço e criá-las. Utilizar, preferencialmente, nomes que as relacionem com os casos de uso ou cenários que representam;
- Adicionar às classes/interfaces os métodos que implementam a lógica de negócio, com atenção ao nome escolhido (preferencialmente relacionar o método ao cenário que implementa), aos parâmetros de entrada e ao retorno (observar a descrição do caso de uso);
- Por meio de uma leitura da descrição dos casos de uso, identificar quais DAOs são necessários para cada classe de aplicação e modelar as associações;
- Voltar ao modelo de navegação (se já foi construído), identificar quais classes de ação dependem de quais classes de serviço e modelar as associações.

Todas as classes de aplicação estendem de *CrudServiceBean* do pacote *nemo-utils*, tal classe está representada abaixo de forma genérica (`Entity` é implementado como um politipo/tipo genérico `T` no código da classe). Da mesma forma dos diagramas anteriores essa herança não é mostrada no diagrama acima com o intuito de não poluir o diagrama com várias associações. A Figura 17 é o modelo de aplicação genérico do *nemo-utils*, enquanto as Figuras 18 e 19 são os modelos de aplicação para os módulos "Controle Trabalho" e "Controle Laboratório", respectivamente.

O FrameWeb indica a utilização do padrão de projeto DAO para a construção da camada de acesso a 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 guia a construção das classes DAO, que pertencem ao pacote de persistência.

Os passos para a construção desse modelo são:

- Criar as interfaces e implementações concretas dos DAOs base;

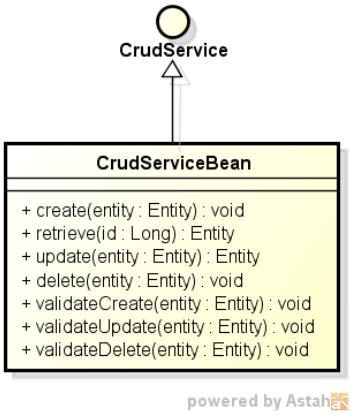

Figura 17 – FrameWeb - *nemo-utils* - Modelo Aplicação.

- Definir quais classes de domínio precisam de lógica de acesso a dados e, portanto, precisam de um DAO;
- Para cada classe que precisa de um DAO, avaliar a necessidade de consultas específicas ao banco de dados, adicionando-as como operações nos respectivos DAOs.

O Modelo de Persistência apresenta, para cada classe de domínio que necessita de lógica de acesso a dados, uma interface e uma classe concreta DAO que implementa a interface. A interface, que é única, define os métodos de persistência existentes para aquela classe, a serem implementados por uma ou mais classes concretas, uma para cada tecnologia de persistência diferente (ex.: um DAO para o *framework* Hibernate, outro para o *framework* OJB etc.).

Para que não seja necessário repetir em cada interface DAO operações que são comuns a todas elas (ex.: `save()`, `delete()`, `retrieveById()`, etc.), podemos apresentar DAOs base que declaram esses métodos – novamente, uma interface e várias implementações. Automaticamente, todas as interfaces DAO de todos os diagramas herdam as definições da interface base, ocorrendo o mesmo com as implementações concretas de cada tecnologia de persistência, sem que isso precise estar explícito no diagrama. A Figura 20 exibe as classes bases do *nemo-utils*.

Tanto a interface **BaseDAO** quanto a classe **BaseJPADAO** são declaradas usando tipos genéricos, deixando a cargo de suas sub-interfaces e sub-classes a especificação da classe gerenciada por cada DAO. O DAO base define métodos para recuperar todos os objetos de uma determinada classe, recuperar um objeto dado seu identificador, salvar e excluir um objeto. 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

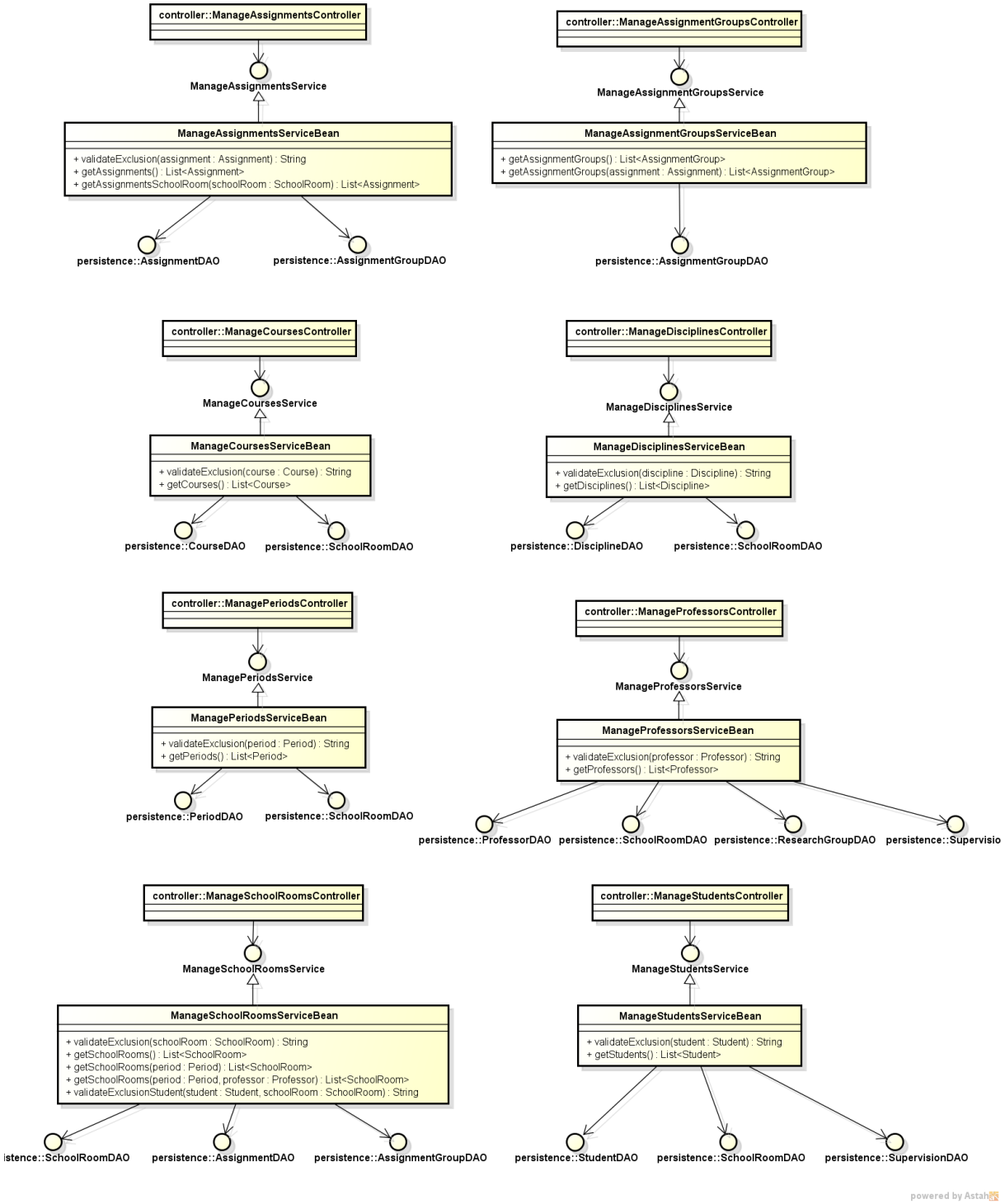

Figura 18 – FrameWeb - Controle Trabalho - Modelo Aplicação.

subinterfaces de BaseDAO, enquanto todas as implementações JPA são sub-classes 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 (último passo do processo de construção do Modelo de Persistência). As Figuras 21 e 22 são os modelos de persistência para os módulos “Controle

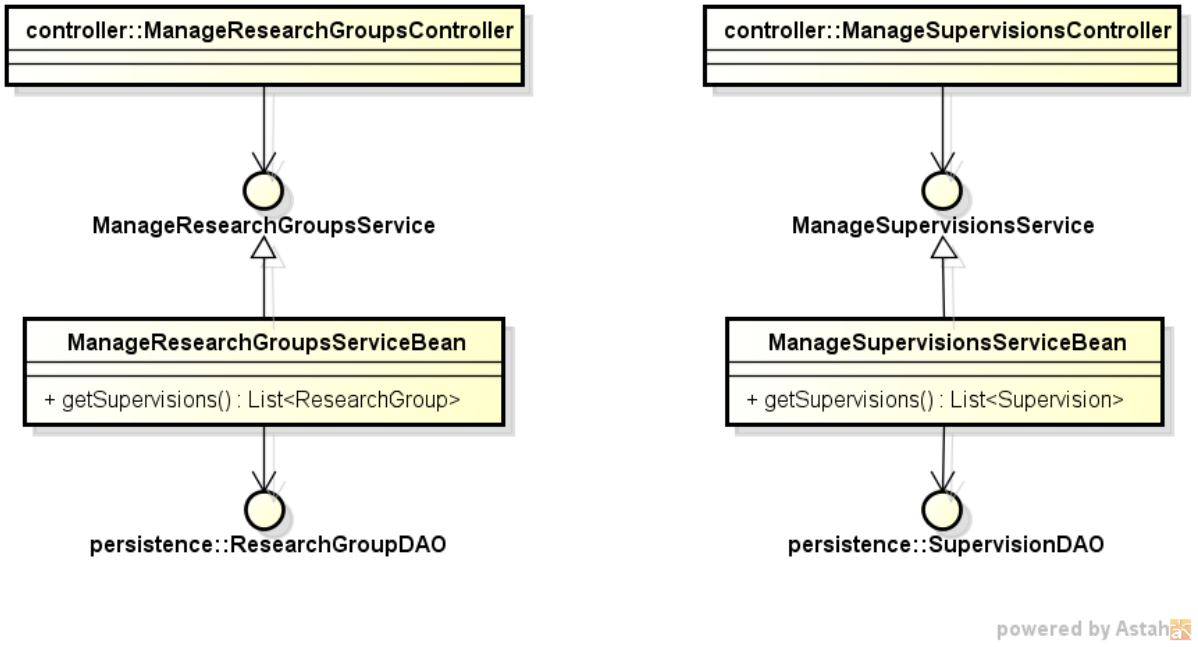

Figura 19 – FrameWeb - Controle Laboratório - Modelo Aplicação.

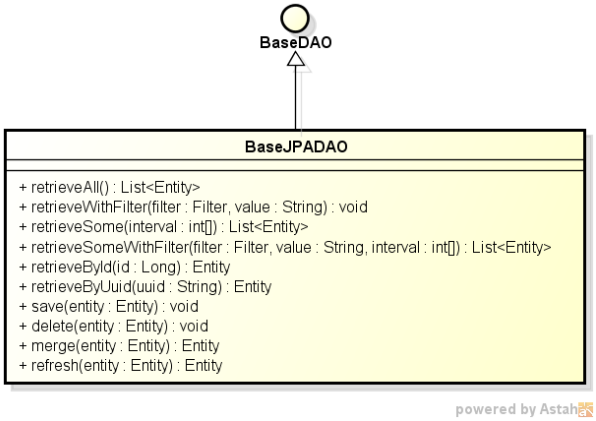

Figura 20 – FrameWeb - *nemo-utils* - Modelo Persistência.

Trabalho" e "Controle Laboratório", respectivamente.

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.

4.3 Framework *nemo-utils*

Nesta seção vamos falar um pouco sobre o framework *nemo-utils*, que foi utilizado para implementar o sistema SAP. No Documento de Requisitos, a **RNF03** diz que "O sistema deve reutilizar componentes e frameworks existentes, se for possível.", pensando nisso, resolvemos utilizar o framework *nemo-utils* que provê uma série de facilidades, pois

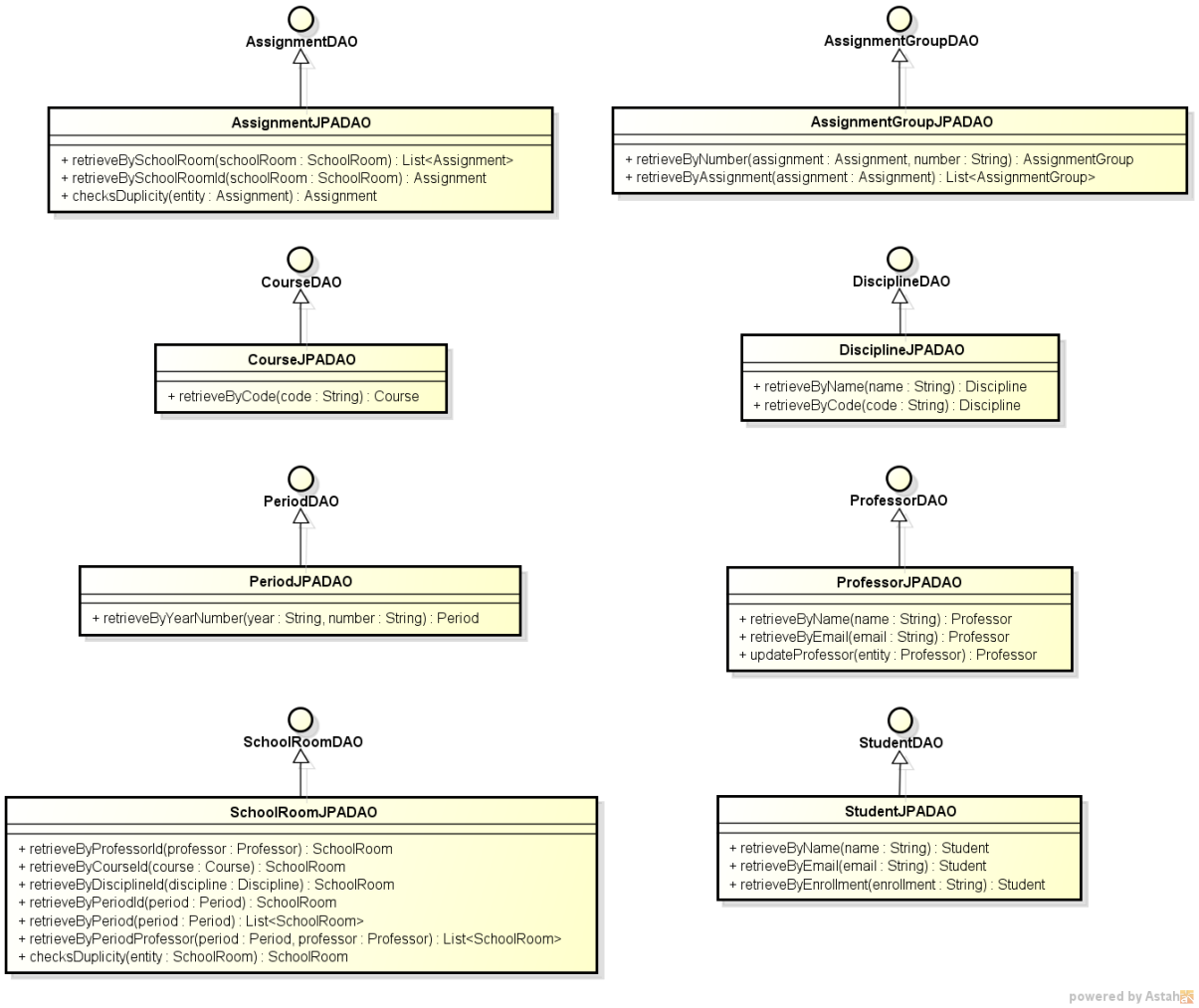

Figura 21 – FrameWeb - Controle Trabalho - Modelo Persistência.

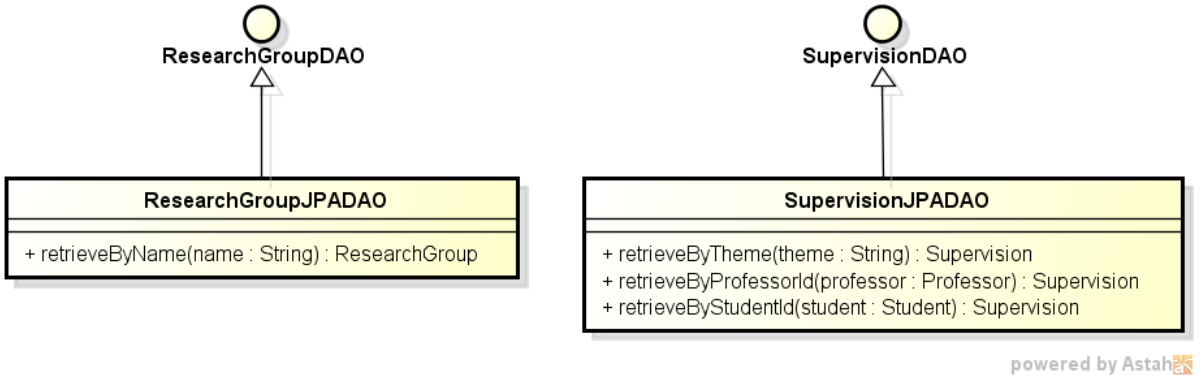

Figura 22 – FrameWeb - Controle Laboratório - Modelo Persistência.

ele já implementa as operações básicas entre a aplicação e o banco de dados de uma forma genérica, bastando ao desenvolver adaptar os códigos para as entidades do domínio do seu problema. Com isso, não foi necessário perder tempo criando funcionalidades que já estavam implementadas no framework.

Os arquivos do pacote `br.ufes.inf.nemo.sap.assignments.controller` herdam

da classe `CrudController` do Nemo Utils. Essa classe é responsável por armazenar temporariamente os dados das páginas Web e depois fazer a comunicação com a camada de aplicação. Em algumas páginas, também são responsáveis por carregar os dados dos componentes `selectOneMenu` do **PrimeFaces**. Além disso, realizam os filtros de pesquisa através do método `initFilters`.

Os arquivos do pacote `br.ufes.inf.nemo.sap.assignments.application` herdam da classe `CrudServiceBean` do *nemo-utils*. Essa classe é responsável realizar as validações e por fazer a comunicação com a camada de acesso a dados. Essa classe possui alguns métodos responsáveis pelas validações, sendo eles:

- `validateCreate` - responsável por fazer as validações ao tentar criar uma nova entidade no sistema. Também possui validações para evitar que dados duplicados sejam inseridos no sistema;
- `validateUpdate` - responsável por fazer as validações ao tentar atualizar os dados de uma entidade já existente no sistema. Também possui validações para evitar que dados duplicados sejam inseridos no sistema;
- `validateDelete` - responsável por fazer as validações ao tentar excluir os dados de uma entidade já existente no sistema. Em alguns casos, algumas classes não podem ser excluídas se tiverem algum relacionamento com outra classe no sistema. Por exemplo, não é possível excluir um professor que possua uma turma.

Utilizando o conceito de herança da programação orientada a objetos, quase todas as entidades do domínio herdam da classe `PersistentObjectSupport`, que é uma implementação padrão para objetos persistentes que utiliza EJB 3 como padrão de anotações para persistência. Essas classes estão no pacote `br.ufes.inf.nemo.sap.assignments.domain` e possuem os seguintes atributos: `serialVersionUID, id e version`. Nesse caso, é importante saber que o campo `id` será usado para identificar unicamente uma entidade no banco de dados e o campo `version` identifica a versão da entidade, para implementação de trancamento otimista (*optimistic lock*).

Por último, os arquivos do pacote `br.ufes.inf.nemo.sap.assignments.persistence` herdam da classe `BaseJPADAO` do *nemo-utils*. Essa classe é responsável por realizar as operações no banco de dados, sendo elas: consulta, modificação, inserção e exclusão de dados. Todas as consultas são realizadas utilizando os conceitos de Criteria API do JPA. Como as consultas que foram implementadas são bem simples utilizando poucas restrições, grande parte do código foi reaproveitado para todas as classes, alterando apenas o tipo e os atributos.

4.4 Apresentando o Sistema

Vamos começar apresentando as telas iniciais do sistema SAP. A Figura 23 mostra a tela inicial do sistema. Ela é composta pela logo do sistema na parte superior, mais abaixo possui uma barra horizontal para exibir os menus. Por fim, todo o espaço abaixo será utilizado para exibir as informações do sistema.

Na barra de menus, podemos observar uma opção para voltar a página inicial e outras duas para efetuar o login no sistema, uma para o professor e outra para o estudante. A parte inferior possui uma breve descrição do sistema e suas principais funcionalidades.

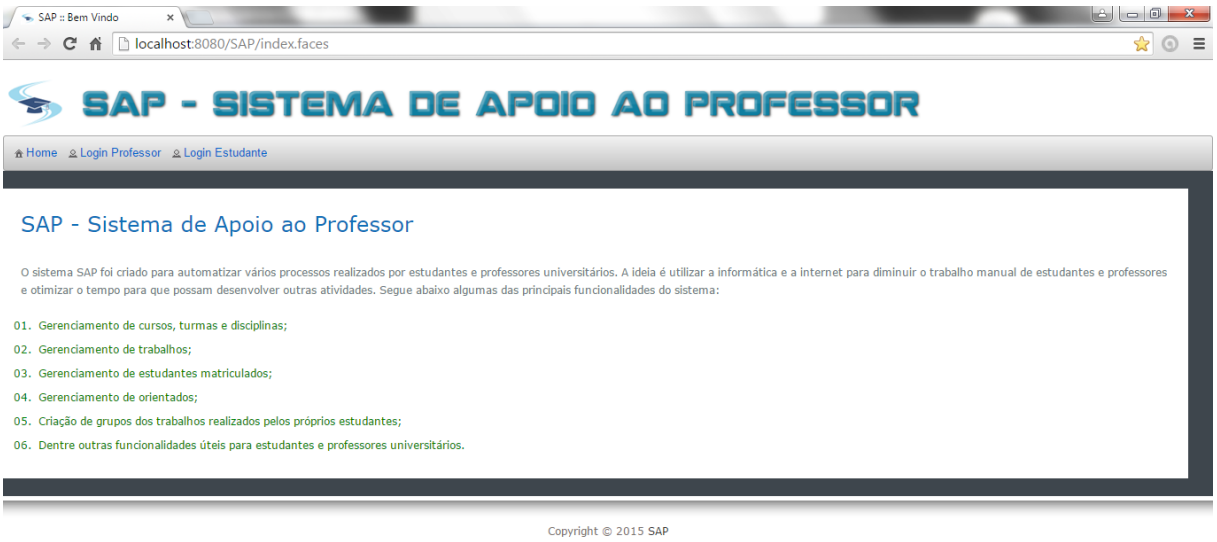

Figura 23 – SAP - Tela Inicial.

No Documento de Requisitos, a **RNF04** diz que "O sistema deve possuir um controle de acesso dos usuários, permitindo apenas o acesso de usuários autorizados. Para isso, os professores e alunos usarão Login e Senha". Pensando nisso, o sistema SAP implementou login e senha para que os seus usuários realizem o acesso e também trata a questão da sessão expirada. A seguir iremos explicar essas questões.

O login dos professores utilizará email e senha. Apesar do professor e do administrador possuírem acessos diferentes, a tela para efetuar o login será a mesma. O que irá diferenciar depois será as opções do menu que será explicado mais a frente. 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 10. Caso o e-mail e senha informados não correspondam a nenhum professor, será exibida uma mensagem de erro no canto superior direito, conforme a Figura 24.

Já o login dos estudantes utilizará matrícula e senha. No campo da matrícula, deve ser informado um número com exatamente 10 caracteres. Caso a matrícula e senha informadas não correspondam a nenhum estudante, será exibida uma mensagem de erro,

Figura 24 – SAP - Erro Login Professor.

semelhante ao login do professor.

Para evitar que um usuário fique com uma sessão aberta infinitamente, foi definido que toda sessão que estiver inativa por 30 minutos será expirada. Isso ajuda na segurança, evitando que uma sessão fique aberta mesmo depois do usuário não estar utilizando, assim como no desempenho do servidor, visto que não será necessário deixar recursos alocados para sessões que não estão sendo utilizadas. A Figura 25 exibe a tela quando a sessão for expirada.

Figura 25 – SAP - Sessão Expirada.

Os menus do sistema irão variar de acordo com o usuário, que são: professor, administrador e estudante. Apesar do professor e o administrador possuírem acessos diferentes, os itens do menu principal são os mesmos, o que irá mudar são os subitens. A Figura 26 exibe a tela inicial do professor após realizar o login.

A Figura 27 exibe os itens do menu do professor. No canto direito, podemos verificar

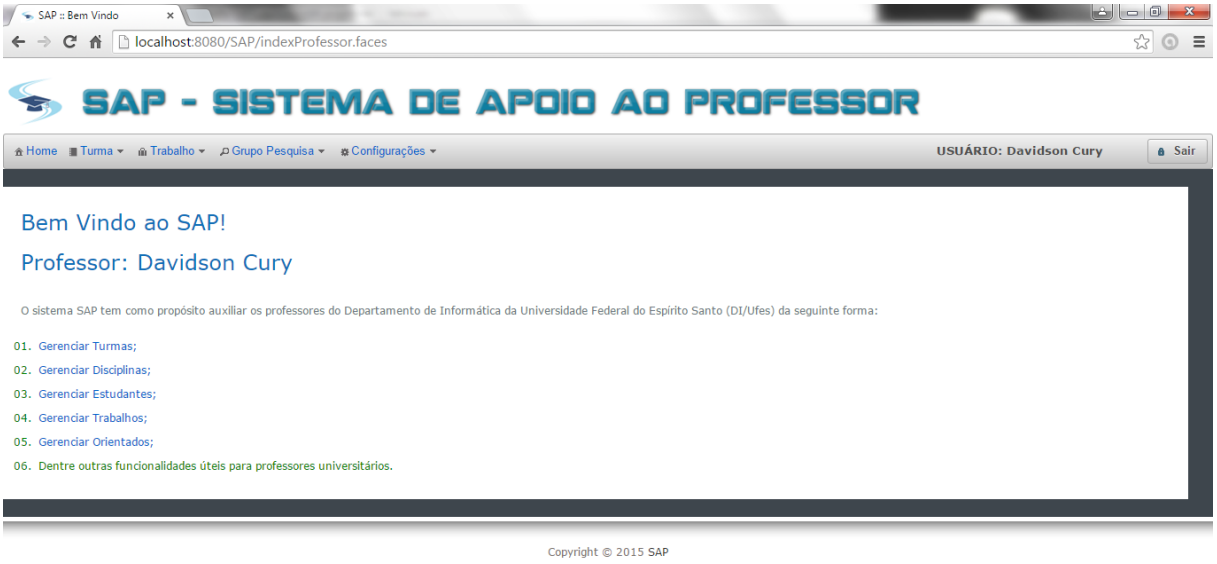

Figura 26 – SAP - Tela inicial Professor.

o usuário que está logado e o botão para sair do sistema.

Figura 27 – SAP - Menu Professor.

A Figura 28 exibe os itens do menu do administrador que é muito semelhante ao do professor, a diferença é que o administrador inclui os itens: Gerenciar Períodos, Gerenciar Cursos, Grupos de Pesquisa e Gerenciar Professores.

Figura 28 – SAP - Menu Administrador.

A tela inicial do estudante e semelhante a tela inicial do professor, diferenciando apenas os menus. A Figura 29 exibe os itens do menu do estudante. Conforme verifi-

cado nas seções anteriores, tanto os professores quanto os estudantes possuem o menu **Configurações -> Alterar Dados**.

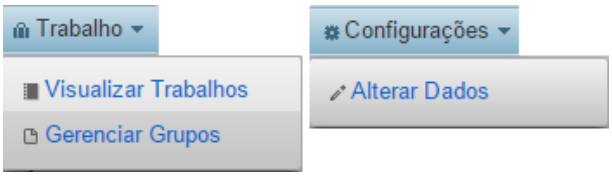

Figura 29 – SAP - Menu Estudante.

A Figura 30 exibe a tela para alteração de dados do professor, onde ele pode alterar o seu nome, e-mail ou senha. No caso dos estudantes, os campos do nome e matrícula não poderão ser alterados por eles mesmos, pois somente o professor possui acesso a essa alteração através do caso de uso Gerenciar Estudantes.

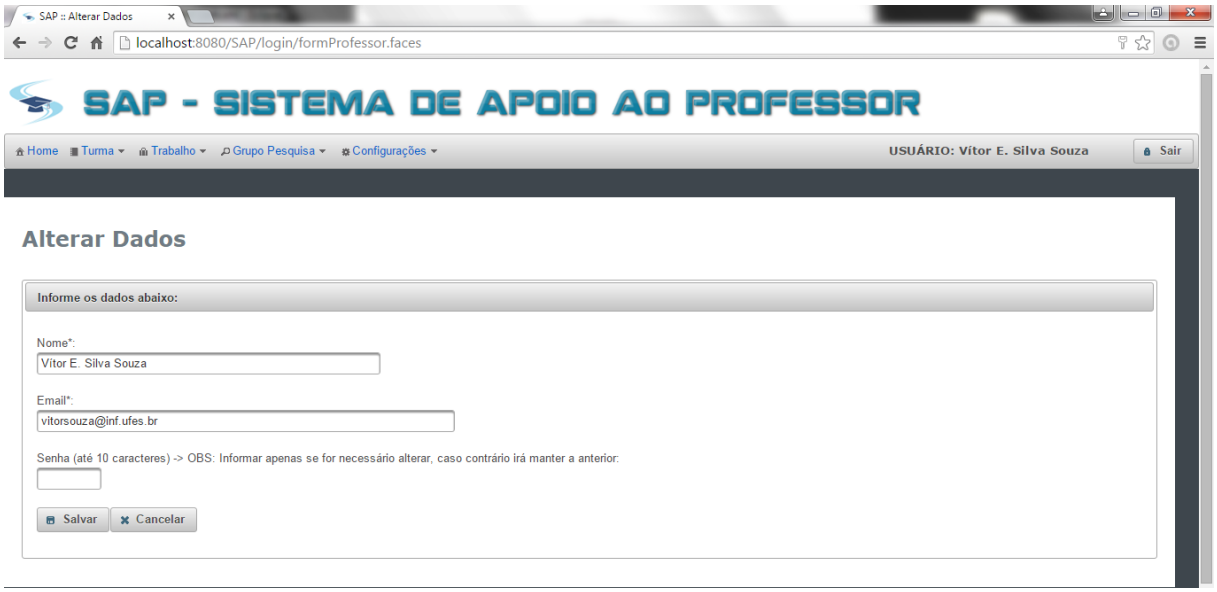

Figura 30 – SAP - Alterar Dados Professor.

A maioria das telas do sistema SAP seguem um padrão, sendo uma para listar as entidade já cadastradas no sistema e outra para visualizar ou informar os dados. A Figura 31 exibe a tela que gerencia as disciplinas. Podemos notar que já existem várias disciplinas cadastradas previamente no sistema. A configuração padrão é exibir 10 registros por página, mas pode ser alterada para 5 ou 20 também. Nesse caso, podemos notar que existem 16 páginas com 10 registros cada uma.

Ao clicar em cima de uma disciplina, quatro botões ficarão disponíveis, são eles: **Novo, Detalhes, Modificar e Deletar**. Esses botões realizam as operações básicas no banco de dados. Ao clicar no botão **Novo**, uma tela será exibida com os campos em branco para preencher com os dados da nova entidade.

Ao clicar no botão **Detalhes**, uma tela será exibida com todos os campos de preenchimento desabilitados, pois a ideia nesse caso é apenas visualizar os dados sem

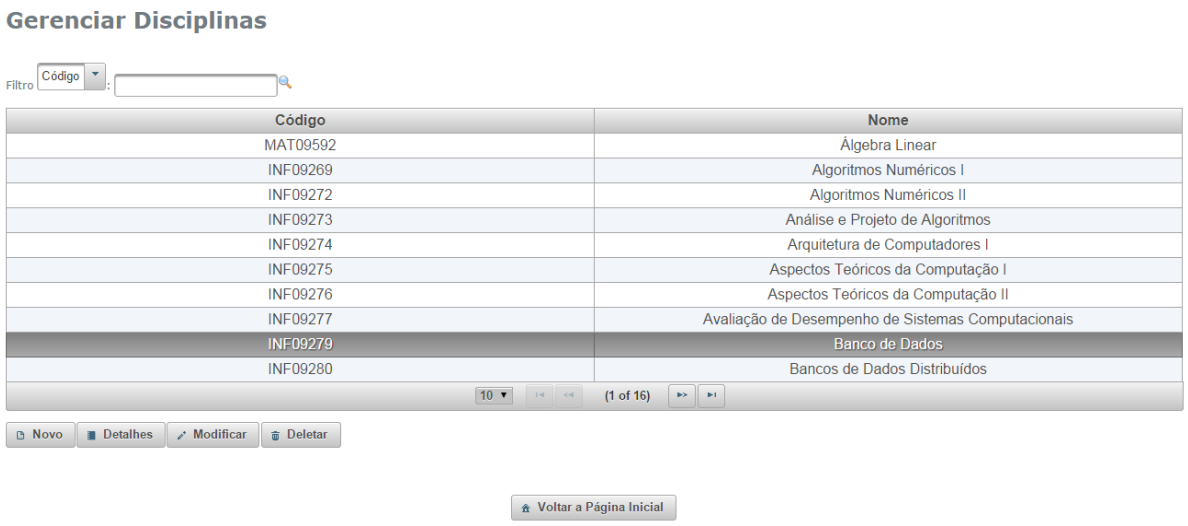

Figura 31 – SAP - Gerenciar Disciplina.

poder fazer qualquer tipo de alteração.

Ao clicar no botão **Modificar**, uma tela será exibida com todos os campos de preenchimento habilitados para edição. No caso da modificação, existem várias validações que podem ser feitas ao tentar salvar os novos dados das entidades para evitar duplicação. Por exemplo, a disciplina "Algoritmos Numéricos I" possui código "INF09269" e a disciplina "Algoritmos Numéricos II" possui código "INF0927". Com isso, se tentarmos colocar um código já existente o sistema irá apresentar uma mensagem de erro, conforme a Figura 32. Isso irá acontecer para todos os campos que são chaves primárias das entidades.

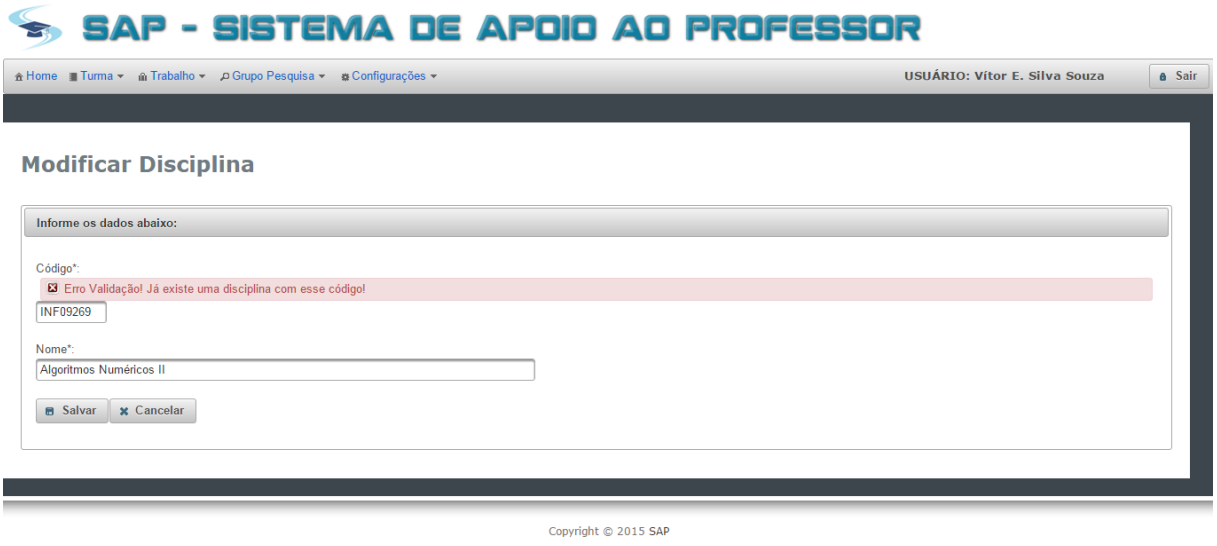

Figura 32 – SAP - Validação Disciplina.

Por último, temos o botão **Deletar** que é responsável por excluir a entidade do banco de dados. Ao clicar nesse botão, um novo quadro será exibido (conforme a Figura 33)

na parte inferior da tela informando a entidade que será excluída e dois botões, um para confirmar e outro para cancelar. Ao clicar no botão **Confirmar Exclusão**, a entidade será excluída (caso passe nas validações) e os dados da tela serão carregados novamente. Ao clicar no botão **Cancelar**, o quadro inferior será escondido e a entidade não será excluída.

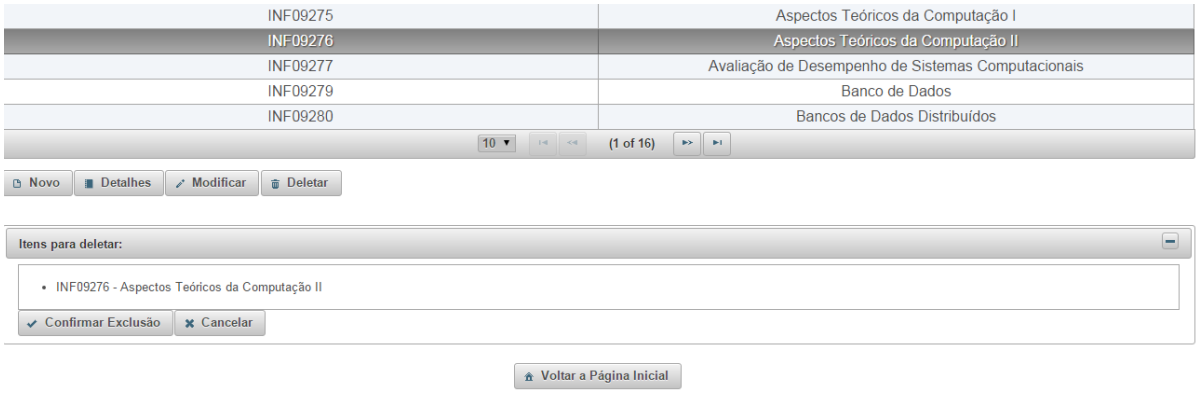

Figura 33 – SAP - Deletar Disciplina.

Também é possível pesquisar uma entidade informando uma parte do texto de um determinado campo. Por exemplo, na Figura 34 filtramos todas as disciplinas que possuem o texto "prog" no campo Nome.

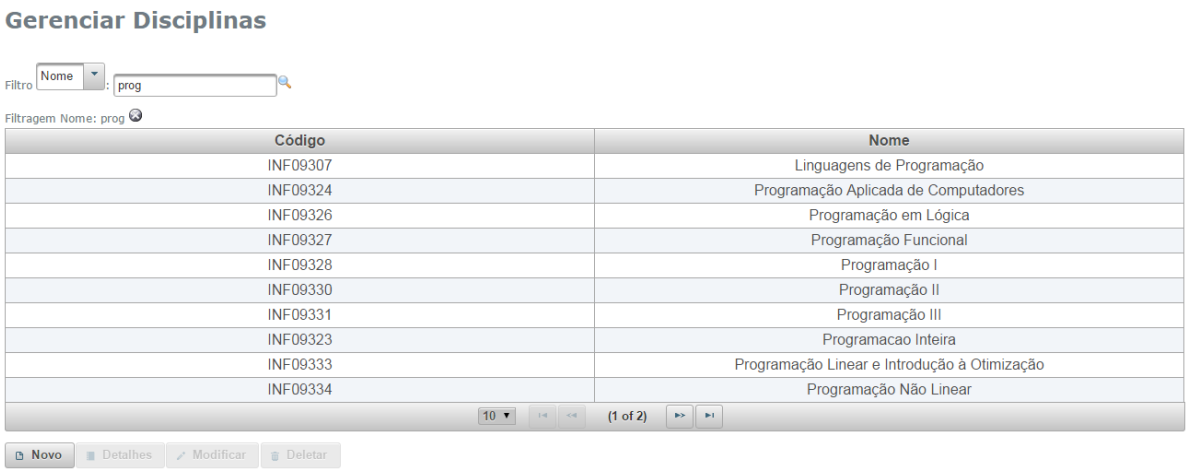

Figura 34 – SAP - Filtrar Disciplina.

Iremos falar agora sobre a funcionalidade **Gerenciar Grupos** que poderá ser feita tanto por professores quanto pelos estudantes. No caso dos estudantes, irá existir uma trava que limita o número de integrantes do grupo pelo número máximo informado no cadastro do trabalho. A Figura 35 exibe a tela com essa funcionalidade. Para criar um grupo, primeiramente devem ser selecionados o período, turma e o trabalho. Feito isso, deve ser escolhido um número para o grupo. Por último, existem duas listas, sendo uma com os estudantes que estão matriculados na turma e que ainda não estão em nenhum

grupo e a outra com os estudantes que foram selecionados para fazer parte do grupo. Ao salvar os dados e tentar criar um novo grupo, os estudantes que foram selecionados no grupo anterior não serão mais listados na tela, visto que já fazem parte de um grupo.

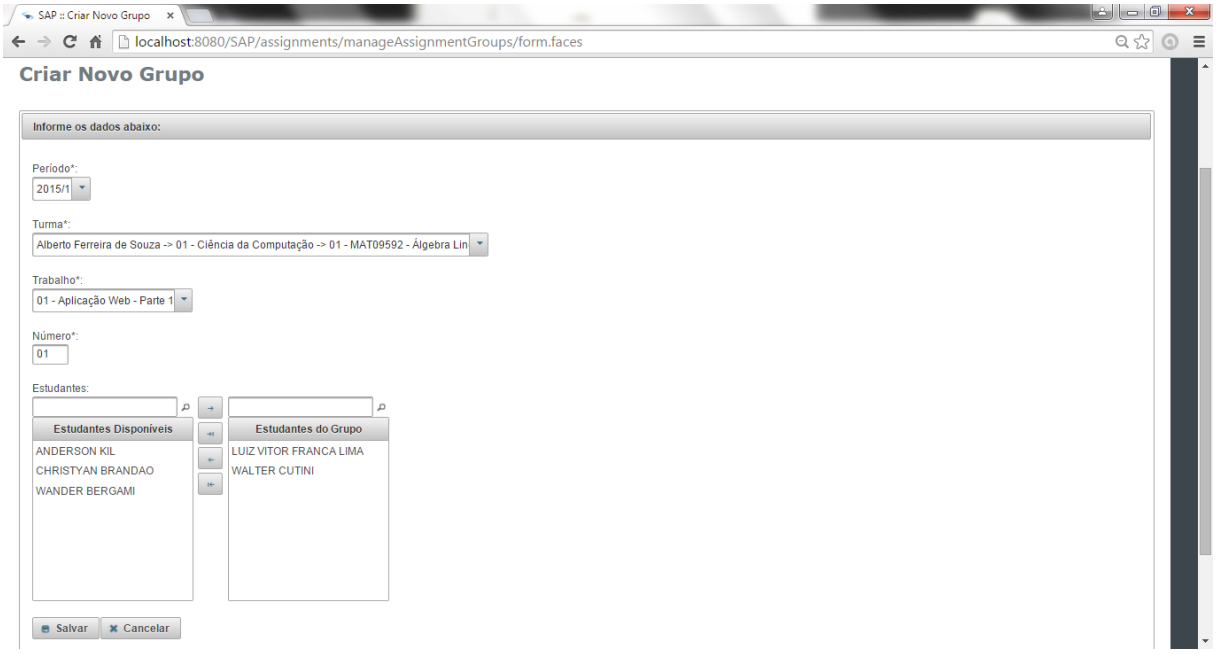

Figura 35 – SAP - Gerenciar Grupos.

Outra funcionalidade importante no sistema é a **Importar Estudantes** que implementa o caso de uso "Associar Estudante a Turma". Como o próprio nome já diz, essa funcionalidade é responsável por importar os dados dos estudantes de um arquivo com extensão ***.csv** e matricular os mesmos na turma que foi selecionada. Nesse ponto temos uma pequena integração com o Portal do Professor da Ufes, visto que, o arquivo é retirado desse sistema com os dados dos estudantes que estão matriculados na turma e possui um layout pré-definido para ser importado no sistema SAP. Existem três casos que podem ocorrer nessa importação que estão listados abaixo. A Figura 36 exibe o resultado de uma importação de exemplo.

- **"Novo estudante registrado no sistema."** – o estudante não estava cadastrado no sistema e nem matriculado na turma;
- **"Estudante já registrado no sistema e matriculado na turma."** – o estudante já estava cadastrado no sistema e matriculado na turma. Nesse caso, nenhuma alteração será realizada e essa mensagem é apenas informativa;
- **"Estudante já registrado no sistema. Foi matriculado na turma."** – o estudante estava cadastrado no sistema mas não estava matriculado na turma selecionada.

Essa funcionalidade pode ser melhorada em trabalhos futuros, visto que, o caminho da planilha para a importação dos dados está implementado com um caminho fixo. O

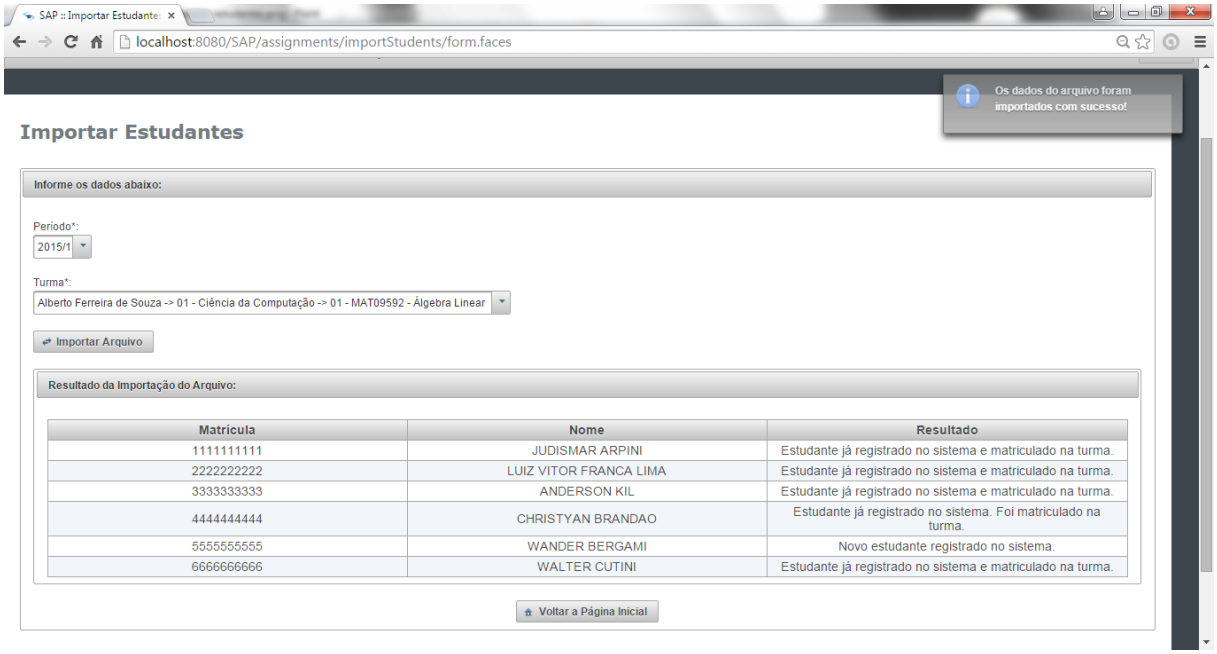

Figura 36 – SAP - Importar Estudantes.

ideal seria que o professor pudesse selecionar a planilha a partir de qualquer local e fazer o upload do mesmo para o servidor para depois realizar a importação dos dados.

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

Com a motivação de atender a uma demanda do DI/Ufes, viu-se a oportunidade de automatizar alguns processos do meio acadêmico através do sistema que foi desenvolvido. Com isso, atividades que antes eram controladas manualmente agora podem ser gerenciadas através do sistema SAP. Como o trabalho que foi desenvolvido é apenas um protótipo, muitas funcionalidades ainda podem ser adicionadas ao sistema.

Todos os objetivos identificados no Capítulo 1 foram atingidos, exceto algumas partes da implementação do sistema devido ao curto período de tempo e a pouca experiência do autor na linguagem Java. Os requisitos foram levantados e analisados, gerando os documentos de Requisitos, Especificação de Requisitos e de Projeto para o sistema SAP. A experiência adquirida com o desenvolvimento desse trabalho foi enorme e bastante proveitosa. Foi possível conhecer novas tecnologias que aplicam na prática os conceitos aprendidos em sala de aula para resolver os problemas que podemos encontrar no dia-a-dia.

Dentre as dificuldades encontradas para o desenvolvimento desse trabalho podemos destacar: estudo e entendimento da linguagem Java aplicada ao desenvolvimento Web, tendo em vista o curto período de tempo para o desenvolvimento do projeto e escrita da monografia e demais documentos, assim como assimilar os conceitos de FrameWeb, que até então eram desconhecidos para o autor desse trabalho.

Vale ressaltar, no entanto, que após um período inicial de aprendizado os modelos FrameWeb auxiliaram bastante, por eles aproximarem o modelo de projeto arquitetural da implementação do sistema. Isso reduziu o tempo de desenvolvimento, visto que os componentes necessários já estão especificados nos modelos. Entretanto, o FrameWeb possui pouca documentação atualizada, visto que sua última publicação é de 2007. Para contornar esse problema, foram utilizados alguns materiais que estão sendo aplicados em sala de aula e em novas pesquisas que estão sendo feitas justamente para corrigir as limitações identificadas nos modelos anteriores do método.

Por fim, a integração das diferentes disciplinas trabalhadas durante o curso de Ciência da Computação foi um desafio, visto que, na graduação, cada disciplina foi vista separadamente e na maioria das vezes de modo teórico, o que gerou uma certa dificuldade ao tentar colocar em prática. Com o desenvolvimento desse sistema, foi possível reforçar e

aplicar todos esses conceitos na prática e enfrentar e superar as dificuldades que apareceram no decorrer do caminho.

5.2 Limitações e Perspectivas Futuras

Algumas limitações podem ser encontradas no sistema SAP, o que dá margem para a realização de trabalhos futuros. São elas:

- Implementar a funcionalidade para o próprio estudante enviar o trabalho através do sistema, onde o sistema deveria armazenar o arquivo enviado, assim como as informações da data de envio e o estudante que enviou.
- Melhorar a funcionalidade que importa os dados dos estudantes para as turmas. Da forma atual, o arquivo está em um local fixo do servidor. Deveria ter uma opção para o professor selecionar e fazer o upload do arquivo.
- Controlar os acessos dos professores em relação aos dados de outros professores. Da forma atual, um professor pode alterar os dados da turma de um outro professor, sendo que isso deveria ser feito apenas pelos administradores.
- Criar relatórios para o sistema. Por exemplo, relatório informando os grupos que enviaram ou não o trabalho para uma determinada turma, relatório com todas as informações sobre uma turma, estudante ou professor.
- Correção automática dos trabalhos enviados pelos estudantes. Assim que o estudante enviasse o trabalho, o sistema deveria executar uma bateria de testes previamente cadastrada pelo professor. No final, o sistema deveria informar quantos testes foram executados com sucesso e uma possível nota para o estudante.
- Integração dos dados do sistema com outros sistemas externos. Por exemplo, consumir dados de artigos escritos por estudantes que estejam cadastrados no sistema SAP.

Referências

AMBLER S.; JEFFRIES, R. *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. 1ª edição. ed. [S.l.]: John Wiley and Sons, 2002. ISBN 0471202827. Citado na página 28.

AURUM, A.; WOHLIN, C. *Engineering and Managing Software Requirements*. [S.l.: s.n.], 2005. ISBN 978-3-540-28244-0. Citado na página 19.

BOOCH G.; RUMBAUGH, J. J. I. *The Unified Modeling Language User Guide*. 2ª edição. ed. [S.l.]: Addison-Wesley Professional„ 2005. ISBN 0321267974. Citado na página 27.

CAELUM. *Capítulo 7 - Introdução ao JSF e Primefaces*. 2015. Disponível em: <http://www.caelum.com.br/apostila-java-testes-jsf-web-services-design-patterns/introducao-ao-jsf-e-primefaces/>. Citado 4 vezes nas páginas 7, 21, 22 e 23.

DEVMEDIA. *CDI – Contextos e Dependências – Parte 2 - Java Magazine 85*. 2015. Disponível em: <http://www.devmedia.com.br/cdi-contextos-e-dependencias-parte-2-java-magazine-85/18492>. Citado na página 24.

DEVMEDIA. *Hibernate API Criteria: Realizando consultas*. 2015. Disponível em: <http://www.devmedia.com.br/hibernate-api-criteria-realizando-consultas/29627>. Citado na página 26.

DEVMEDIA. *Introdução à JPA - Java Persistence API*. 2015. Disponível em: <http://www.devmedia.com.br/introducao-a-jpa-java-persistence-api/28173>. Citado 2 vezes nas páginas 25 e 26.

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 4 vezes nas páginas 17, 18, 19 e 20.

PFLEEGER, S. *Engenharia de Software: Teoria e Prática*. 2ª edição. ed. [S.l.]: Prentice Hall, 2004. Citado na página 20.

PRESSMAN, R. S. *Software Engineering: A Practitioner's Approach*. 6ª edição. ed. [S.l.]: McGraw Hill, 2005. ISBN 007301933X. Citado na página 27.

SOMMERVILLE, I. *Engenharia de Software*. 8ª edição. ed. [S.l.]: Addison Wesley, 2007. Citado 2 vezes nas páginas 19 e 20.

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 3 vezes nas páginas 27, 46 e 47.

Apêndices

Documento de Requisitos

Projeto: "**SAP** - Sistema de Apoio aos Professores"

Registro de Alterações:

Versão	Responsáveis	Data	Alterações
1.0	Luiz Vitor França Lima	30/10/2014	Versão inicial.
2.0	Luiz Vitor França Lima	09/11/2014	Alterações na descrição e Minimundo. Exclusão de algumas funcionalidades que não serão necessárias nesse momento. Reescrita de algumas RF, alterando letras maiúsculas para minúsculas. Reescrita da RF05 para "criação de turmas". Criação da RF12 para cadastrar alunos nas disciplinas. Inclusão das RF13, RF14, RF15 e RF16 sobre os relatórios do sistema.
3.0	Luiz Vitor França Lima	04/12/2014	Pequenas alterações no final da descrição do Minimundo. Retirada da "matricula" do professor na RF01 Alteração na RF05 e criação da RN05. Inclusão do "assunto" no cadastro do trabalho na RF07. RF08 e RF09 foram reescritos.
4.0	Luiz Vitor França Lima	09/03/2015	Pequenas modificações na descrição de alguns requisitos funcionais. Inclusão da RGN17 para cadastro de cursos.
5.0	Luiz Vitor França Lima	11/06/2015	Adaptações no escopo do projeto.

1. Introdução

Este documento apresenta os requisitos de usuário do sistema "**SAP** - Sistema de Apoio aos Professores" e 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; e a seção 4 apresenta a lista de requisitos de usuário levantados junto ao cliente.

2. Descrição do Propósito do Sistema

O sistema "**SAP**" tem como propósito auxiliar os professores do Departamento de Informática da Universidade Federal do Espírito Santo (DI/Ufes) no gerenciamento das suas disciplinas, controlando os alunos matriculados, envio e recebimento de trabalhos, assim como os grupos formados. Também auxilia no controle dos alunos orientados, tanto os atuais como os já formados (histórico de egressos), dentre outras funcionalidades úteis para professores universitários.

3. Descrição do Minimundo

No DI/Ufes, é comum os professores que ministram alguma disciplina aplicarem trabalhos para os alunos que estão matriculados. Com isso, quando os alunos enviam seus trabalhos o professor precisa salvá-los manualmente, assim como verificar quem enviou antes ou depois do prazo estipulado e quem não enviou. O sistema que será desenvolvido busca automatizar os processos acima utilizando a informática para diminuir o trabalho manual do professor, assim como otimizar o seu tempo para que possa desenvolver outras atividades.

Utilizando o sistema, o professor inicialmente deverá associar a disciplina ou cadastrá-la no sistema, caso ainda não exista, para depois fazer a associação. Feito isso, deverá fazer a importação com os dados dos alunos matriculados. Poderá também cadastrar cada trabalho com sua devida especificação (data de entrega, valor, etc.).

Também será possível criar grupos de trabalho que irão possuir uma quantidade máxima de alunos definida pelo professor. Os próprios alunos que serão responsáveis por criarem os seus grupos através do sistema SAP utilizando login e senha. O professor poderá cadastrar um valor de penalização para cada dia de atraso do recebimento do trabalho após a data de entrega.

Além disso, o sistema também permitirá que o professor possa cadastrar e controlar as informações dos alunos orientados (iniciação científica, graduação, mestrado e doutorado), tanto os atuais quanto os já formados (histórico de egressos). Desta forma, o sistema auxilia também nas atividades de pesquisa dos professores e fornece a base para que, no futuro, sejam adicionadas funcionalidades que auxiliem no gerenciamento de grupos de pesquisa.

É interessante notar que na universidade há um sistema para gerenciamento de matrículas de alunos em disciplinas, com possibilidade de cadastro de diários de classe, envio de e-mail para alunos, registro de faltas e notas, etc. Por esse motivo, tais funcionalidades não foram incluídas no SAP. Idealmente, os sistemas seriam integrados, porém tal integração é algo muito difícil de acontecer, visto que o Núcleo de Tecnologia da Informação (NTI) não pode abrir seus sistemas e bases de dados para alunos e professores que não fazem parte de seu quadro de funcionários. Portanto, trabalharemos a princípio com o fato que os sistemas são disjuntos, promovendo integração manual sempre que possível (ex.: leitura de arquivos estruturados com a lista de alunos matriculados para uma disciplina, produzido pelo sistema da Ufes).

4. Requisitos de Usuário

Tomando por base o contexto do sistema, foram identificados os seguintes requisitos de usuário:

Requisitos Funcionais

Identificador	Descrição	Prioridade	Depende de
RF01	O sistema deve permitir o cadastro de professores, com as seguintes informações: nome, e-mail e grupos de pesquisa. Deve ser informado também se o professor é ou não administrador. Apenas os coordenadores poderão realizar esse cadastro.	Alta	RF04
RF02	O sistema deve permitir o cadastro de disciplinas, com as seguintes informações: nome e código.	Alta	
RF03	O sistema deve permitir o cadastro de alunos, com as seguintes informações: nome, matrícula, telefone e e-mail.	Alta	
RF04	O sistema deve permitir o cadastro de grupos de pesquisas, com as seguintes informações: nome e site.	Alta	
RF05	O sistema deve permitir o cadastro de turmas, com as seguintes informações: número, período, alunos, curso e disciplina.	Alta	RF02
RF06	O sistema deve permitir a associação de alunos a disciplinas. O professor será responsável por fazer essa associação realizando uma importação com os dados dos alunos.	Alta	RF01, RF02, RF03
RF07	O sistema deve permitir o cadastro de trabalhos, com as seguintes informações: número, turma, assunto, valor, data de entrega e número máximo de participantes por grupo. Esse cadastro deve ser associado a uma disciplina. Para controlar os envios com atraso, o professor também poderá cadastrar um valor para desconto por dia de atraso, mas essa informação é opcional, ficando a critério do professor poder utilizar ou não.	Alta	RF01, RF02
RF08	O sistema deve permitir o cadastro de grupos de trabalhos, com as seguintes informações: número e participantes (será uma lista de alunos). Esse cadastro deve ser associado a um trabalho e poderá ser feito por um aluno ou professor, bastando selecionar os alunos listados na tela como integrantes do grupo. Somente o professor tem acesso para associar mais alunos do que o permitido para o grupo.	Alta	RF03, RF04, RF07
RF09	O aluno deverá enviar o trabalho através do sistema informando, opcionalmente, um código MD5. O sistema, por sua vez, deverá conferir se o código MD5 informado é correto e registrar a data de recebimento do trabalho.	Alta	RF03, RF07
RF10	O sistema deve permitir a associação de professores a grupos de pesquisa.	Alta	RF01, RF04

RF11	O sistema deve permitir o cadastro de orientados, com as seguintes informações: nome, matrícula, e-mail, telefone, orientador, co-orientador (opcional), data inicial, data final, tipo de orientação e tema.	Alta	RF01
RF12	O sistema deve permitir a associação de alunos individualmente a disciplinas, que poderá ser realizada pelo professor.	Alta	RF01, RF02, RF03
RF17	O sistema deve permitir o cadastro de cursos, com as seguintes informações: nome e código. Apenas os coordenadores poderão realizar esse cadastro.	Alta	
RF18	O sistema deve permitir o cadastro de períodos, com as seguintes informações: ano, número, data inicial e data final. Apenas os coordenadores poderão realizar esse cadastro.	Alta	

Regras de Negócio

Identificador	**Descrição**	**Prioridade**	**Depende de**
RN01	Um aluno não pode estar associado a mais de um grupo para um mesmo trabalho.	Alta	RF08
RN02	Para os Alunos Orientados, os possíveis Tipos de Orientação são: • "Iniciação Científica" (duração máxima de 1 ano) • "Graduação" (duração prevista de 1 ano) • "Mestrado" (duração máxima de 3 anos e 6 meses) • "Doutorado" (duração máxima de 5 anos e 6 meses)	Alta	RF11

Requisitos Não Funcionais

Identificador	**Descrição**	**Categoria**	**Escopo**	**Prioridade**	**Depende de**
RNF01	O sistema deve ter seu acesso disponível pela internet, a partir dos principais navegadores.	Portabilidade	Sistema	Alta	
RNF02	O sistema deve ser de aprendizado fácil, não sendo necessário nenhum treinamento especial para seu uso.	Facilidade de Aprendizado	Sistema	Alta	
RNF03	O sistema deve reutilizar componentes e *frameworks* existentes, se for possível.	Reusabilidade	Sistema	Média	
RNF04	O sistema deve possuir um controle de acesso dos usuários, permitindo	Segurança	Sistema	Alta	

	apenas o acesso de usuários autorizados. Para isso, os professores e alunos usarão Login e Senha.				

Documento de Especificação de Requisitos

Projeto: "**SAP -** Sistema de Apoio aos Professores"

Registro de Alterações:

Versão	Responsáveis	Data	Alterações
1.0	Luiz Vitor França Lima	14/12/2014	Versão inicial.
2.0	Luiz Vitor França Lima	10/02/2015	Caso de Uso "Cadastrar Aluno" movido para o subsistema "Controle Laboratório". Alteração na descrição dos casos de uso cadastrais e de consulta. Correção na descrição das classes, alteração de iniciais maiúsculas para minúsculas e colocação de acentos. Inclusão do UC13 para cadastrar curso. Falta preencher apenas a seção 5. Modelo Dinâmico.
3.0	Luiz Vitor França Lima	11/03/2015	Exclusão da classe Orientado e criação da classe Matrícula que será usada tanto para graduação quanto para pós-graduação. Criação do Modelo Dinâmico (seção 5).
4.0	Luiz Vitor França Lima	22/07/2015	Atualização dos diagramas de classes e casos de uso. Atualizando os casos de uso de acordo com a implementação do sistema SAP. Exclusão dos casos de uso de geração de relatórios (melhorias futuras). Movimentação de alguns casos de uso do pacote de laboratório para o pacote de trabalho. Inclusão do caso de uso "Gerenciar Período".

1. Introdução

Este documento apresenta a especificação dos requisitos da ferramenta "**SAP -** Sistema de Apoio aos Professores". 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 apresenta os subsistemas identificados, mostrando suas dependências na forma de um diagrama de pacotes; a seção 3 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 4 apresenta o modelo conceitual estrutural do sistema, na forma de diagramas de classes; a seção 5 apresenta o modelo comportamental dinâmico do sistema, na forma de diagramas de atividades; finalmente, a seção 6 apresenta o glossário do projeto, contendo as definições das classes identificadas.

2. Identificação de Subsistemas

A Figura 1 mostra os subsistemas identificados no contexto do presente projeto, os quais são descritos na tabela abaixo.

Figura 1 – Diagrama de Pacotes e os Subsistemas Identificados.

Tabela 1 – Subsistemas

Subsistema	Descrição
Controle Trabalho	Envolve as funcionalidades relacionadas ao controle dos estudantes, trabalhos, turmas e disciplinas.
Controle Laboratório	Envolve as funcionalidades relacionadas ao controle dos professores, orientados e grupos de pesquisa.

3. 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 abaixo.

Tabela 2 – Atores.

Ator	Descrição
Professor	Representa os professores do Departamento de Informática da Universidade Federal do Espírito Santo (DI/Ufes).
Administrador	Possui as mesmas funcionalidades do Professor, entretanto, acrescenta algumas funcionalidades especiais relacionadas aos professores e grupos de pesquisa.
Estudante	Representa os estudantes cadastrados no sistema.

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

3.1 - Subsistema Controle Trabalho

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

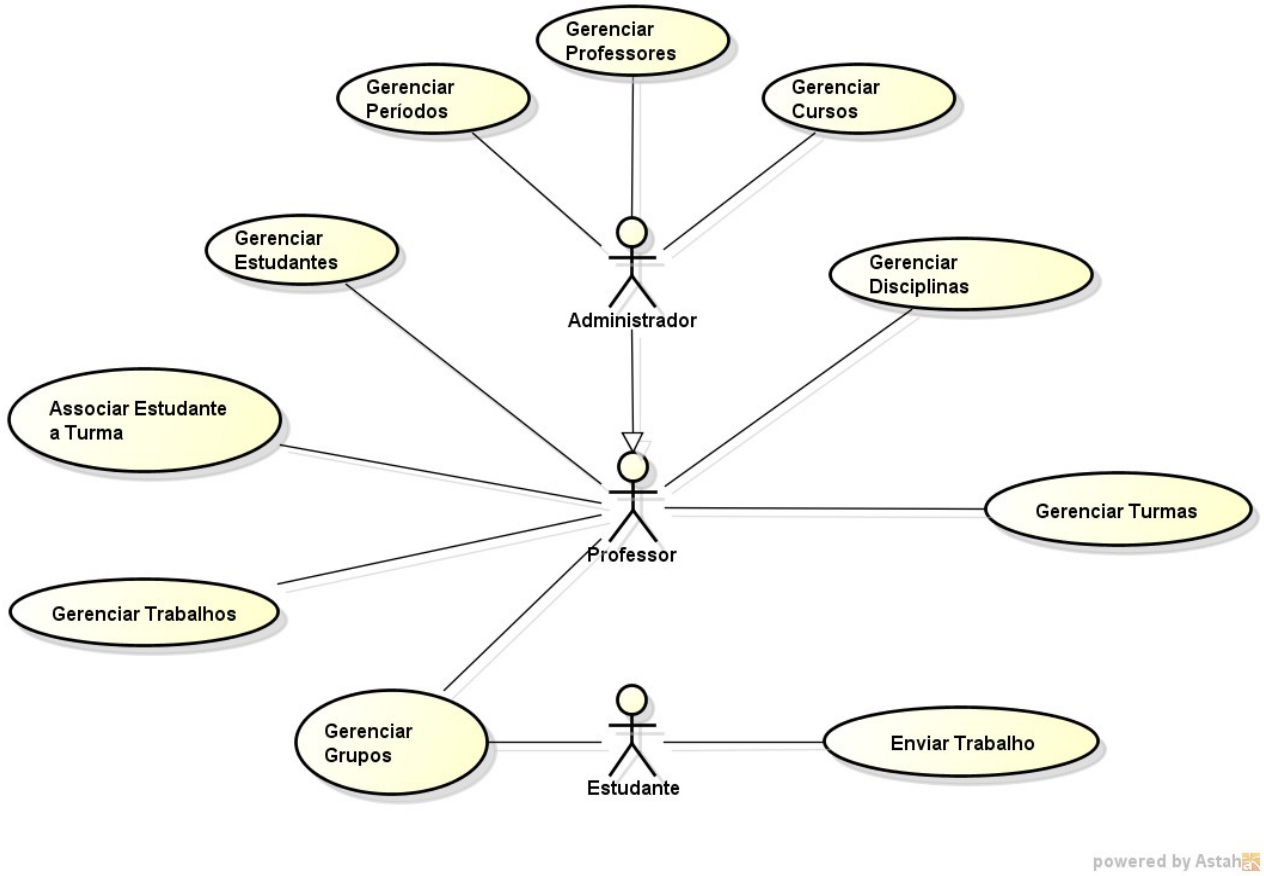

Figura 2 – Diagrama de Casos de Uso do Subsistema Controle Trabalho.

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 abaixo, segundo o padrão da organização.

Tabela 3 – Casos de Uso Cadastrais

Subsistema	**Controle Trabalho**				
Identificador	**Caso de Uso**	**Ações Possíveis**	**Observações**	**Requisitos**	**Classes**
UC01	**Gerenciar Disciplinas**	I, A, C, E	[I] Informar: nome e código. [E] Caso a disciplina possua alguma turma, a mesma não poderá ser excluída.	RF02	Disciplina
UC02	**Gerenciar Turmas**	I, A, C, E	[I] Informar: número, período, horário, curso e disciplina. A turma é associada automaticamente ao ator Professor.	RF01, RF02, RF03,	Disciplina, Turma, Período,

			[E] Caso a turma possua algum trabalho associado, a mesma não poderá ser excluída.	RF05, RF07, RF08	Professor, Trabalho, Trabalho Entregue, Grupo de Trabalho, Estudante
UC03	**Gerenciar Trabalhos**	I, A, C, E	[I] Informar: turma, assunto, valor, data de entrega, descrição, número máximo de participantes por grupo e valor para desconto por dia de atraso. [E] Caso o trabalho já possua algum grupo cadastrado, o mesmo não poderá ser excluído.	RF05, RF07, RF08	Turma, Trabalho, Grupo de Trabalho, Trabalho Entregue
UC04	**Gerenciar Grupos**	I, A, C, E	[I] Informar: número e estudantes participantes (dentre os estudantes matriculados na turma daquele trabalho). Não é permitido a um estudante participar de mais de um grupo para o mesmo trabalho. OBS: Se o ator for “Professor” ou “Administrador”, será possível cadastrar mais estudantes do que o número máximo de participantes do trabalho. [E] Caso o grupo já possua algum trabalho entregue, o mesmo não poderá ser excluído.	RF03, RF07, RF08	Estudante, Grupo Trabalho, Trabalho, Trabalho Entregue
UC09	**Gerenciar Estudantes**	I, A, C, E	[I] Informar: nome, matrícula, e-mail e senha. [E] Caso o estudante esteja matriculado em alguma turma ou grupo, o mesmo não poderá ser excluído.	RF03, RF05, RF08	Estudante, Turma, Grupo de Trabalho
UC10	**Gerenciar Professores**	I, A, C, E	[I] Informar: nome, e-mail, senha e se é administrador. [E] Caso o professor possua alguma turma ou orientado, o mesmo não poderá ser excluído.	RF01, RF04, RF11	Professor, Grupo Pesquisa, Orientado, Orientação, Turma
UC13	**Gerenciar Cursos**	I, A, C, E	[I] Informar: código e nome. [E] Caso o curso possua alguma turma, o mesmo não poderá ser excluído.	RF17	Curso
UC16	**Gerenciar Períodos**	I, A, C, E	[I] Informar: ano, número, data inicial e data final. [E] Caso o período possua alguma turma, o mesmo não poderá ser excluído.	RF18	Período

A seguir, são apresentados os casos de uso de maior complexidade que não puderam ser descritos segundo os formatos tabulares simplificados. Esses casos de uso são descritos segundo o padrão de descrição completa de casos de uso definido.

Projeto: SAP - Sistema de Apoio aos Professores
Subsistema: Controle Trabalho
Identificador do Caso de Uso: UC07
Caso de Uso: Associar Estudante a Turma

Descrição Sucinta: Utilizado pelo professor para associar o estudante a uma turma.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Associar Estudante a Turma	A turma deve estar registrada no sistema.	1 A associação do estudante com a turma pode ser feita de duas formas distintas: 2 2.1 Tela Gerenciar Turmas: 2.1.1 Selecionar a turma desejada. 2.1.2 O professor pode selecionar um ou mais estudantes que já estejam cadastrados no sistema. 2.1.3 Após selecionar os estudantes, basta salvar as alterações da turma. 2.2 Tela Importar Estudantes: 2.2.1 Selecionar o período desejado. 2.2.2 Selecionar a turma desejada. 2.2.3 O professor importa os dados de vários alunos de um arquivo com extensão no formato “*.csv”. Esse arquivo contêm a matrícula e o nome dos estudantes . Nesse caso, a senha cadastrada será “123456” e o aluno deverá mudar posteriormente.
Desassociar Estudante a Turma	O estudante deve estar matriculado na turma.	1. O professor deve entrar na tela “Gerenciar Turmas”. 2. Selecionar a turma desejada. 3. Selecionar os estudantes que devem ser retirados da turma. 4. Após selecionar os estudantes, basta salvar as alterações da turma.

Requisitos Relacionados: RF01, RF03, RF05

Classes Relacionadas: Professor, Estudante, Turma

Descrição de Caso de Uso

Projeto: SAP - Sistema de Apoio aos Professores
Subsistema: Controle Trabalho
Identificador do Caso de Uso: UC08
Caso de Uso: Enviar Trabalho

Descrição Sucinta: Utilizado pelo estudante para poder enviar um trabalho através do sistema.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Enviar Trabalho	O aluno deve estar registrado no sistema e pertencer a algum grupo. O trabalho também deve estar registrado no sistema.	1 O aluno deve entrar no sistema e selecionar o trabalho e o grupo. 2 Existem duas possibilidades para o envio do trabalho: 2.1 Se for a primeira vez que o aluno envia o trabalho, o mesmo será armazenado normalmente. 2.2 Se não for a primeira vez, o sistema deve manter os arquivos que foram enviados anteriormente e salvar o novo também. 3 Clicar em um botão para enviar o trabalho. 4 Deverá ser exibida a mensagem “Deseja confirmar o envio do trabalho?” com as opções “SIM” ou “NÃO”. 5 Caso escolha a opção “SIM”, o trabalho deve ser enviado e exibida a mensagem “Trabalho enviado com sucesso!”. Também deverá ser exibida uma outra mensagem com o código MD5 do arquivo recebido pelo sistema para que o aluno possa verificar se foi enviado corretamente. 6 Caso escolha a opção “NÃO”, deve voltar para a tela anterior.

Requisitos Relacionados: RF03, RF07, RF09

Classes Relacionadas: Estudante, Trabalho, Grupo Trabalho, Trabalho Entregue

3.2 - Subsistema Controle Laboratório

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

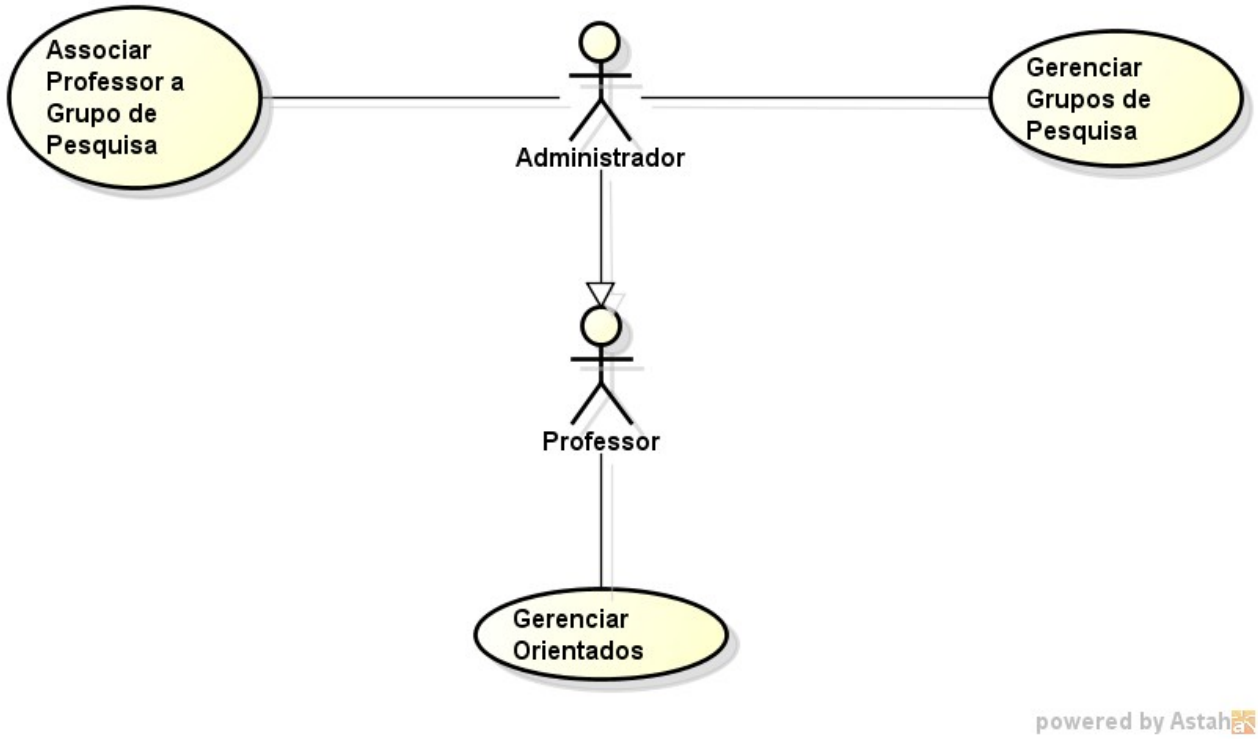

Figura 3 – Diagrama de Casos de Uso do Subsistema Controle Laboratório.

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 abaixo, segundo o padrão da organização.

Tabela 5 – Casos de Uso Cadastrais

Subsistema	**Controle Laboratório**				
Identificador	**Caso de Uso**	**Ações Possíveis**	**Observações**	**Requisitos**	**Classes**
UC11	**Gerenciar Orientados**	I, A, C, E	[I] Informar: estudante, orientador, co-orientador (opcional), data inicial, data final, tipo de orientação e tema.	RF01, RF03, RF11	Professor, Estudante, Orientação
UC12	**Gerenciar Grupos de Pesquisa**	I, A, C, E	[I] Informar: nome e site. [E] Caso o grupo de pesquisa possua algum professor associado, o mesmo não poderá ser excluído.	RF01, RF04	Grupo Pesquisa, Professor

A seguir, são apresentados os casos de uso de maior complexidade que não puderam ser descritos segundo os formatos tabulares simplificados. Esses casos de uso são descritos segundo o padrão de descrição completa de casos de uso definido.

Projeto: SAP - Sistema de Apoio aos Professores
Subsistema: Controle Laboratório
Identificador do Caso de Uso: UC15
Caso de Uso: Associar Professor a Grupo de Pesquisa

Descrição Sucinta: Utilizado pelo administrador para associar professores nos grupos de pesquisa.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Associar professor a grupo de pesquisa	O professor e o grupo de pesquisa já devem estar registrados no sistema.	1. O administrador deve entrar na tela “Grupos de Pesquisa”. 2. Selecionar o grupo de pesquisa desejado. 3. Selecionar os professores que deseja associar ao grupo de pesquisa. 4. Salvar as alterações do grupo de pesquisa.
Desassociar professor a grupo de pesquisa	O professor e o grupo de pesquisa devem estar associados.	1. O administrador deve entrar na tela “Grupos de Pesquisa”. 2. Selecionar o grupo de pesquisa desejado. 3. Selecionar os professores que deseja retirar do grupo de pesquisa 4. Salvar as alterações do grupo de pesquisa.

Requisitos Relacionados: RF01, RF04

Classes Relacionadas: Professor, Grupo de Pesquisa

4. 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 6 – Glossário de Projeto – são apresentadas as descrições das classes presentes nos diagramas apresentados nesta seção.

4.1 - Subsistema Controle Trabalho

A Figura 4 apresenta o diagrama de classes do subsistema Controle Trabalho.

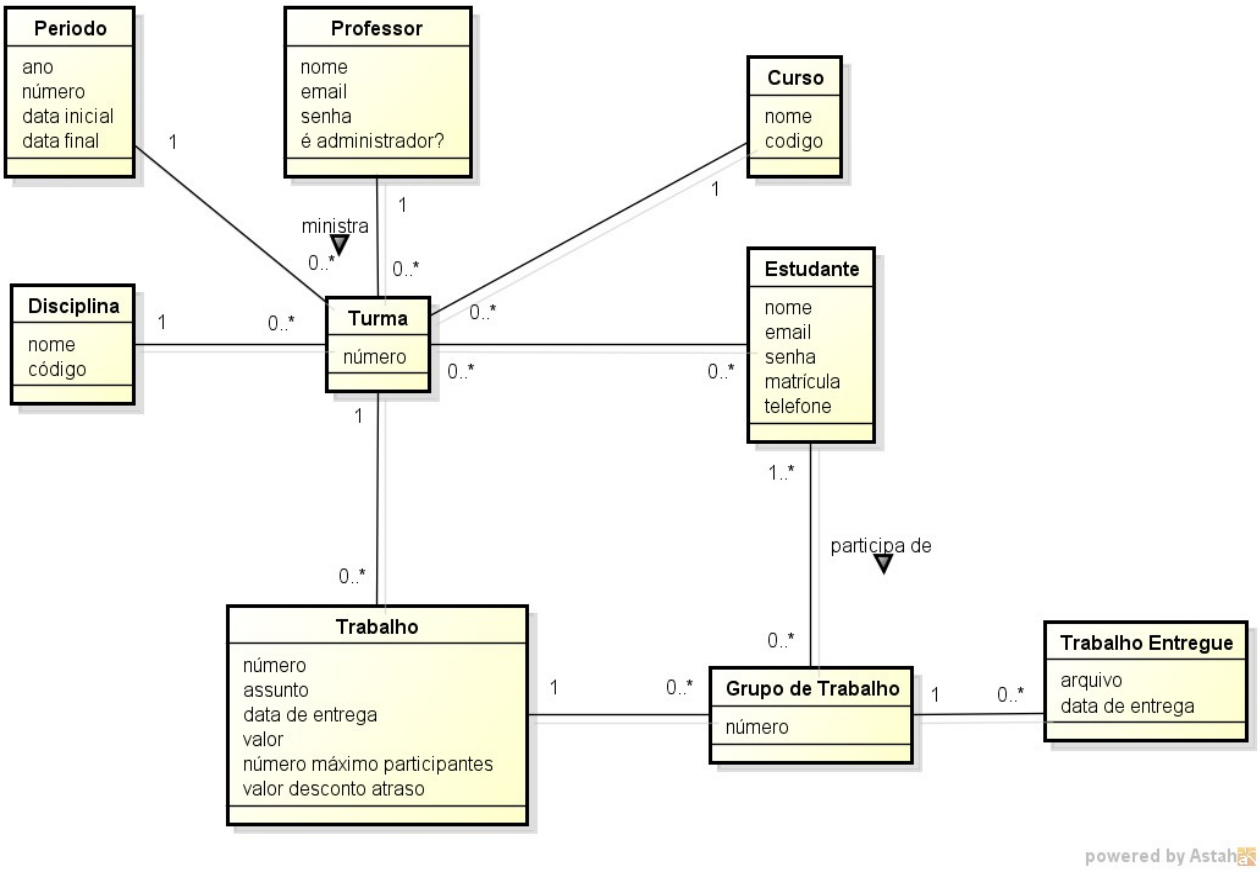

Figura 4 – Diagrama de Classes do Subsistema Controle Trabalho.

As seguintes restrições de integridade devem ser observadas:

- Um estudante não pode participar de mais de um grupo de trabalho para o mesmo trabalho de uma turma.
- Um trabalho não pode ser entregue após o término do período letivo da turma.
- Um grupo de trabalho não pode ter mais participantes do que o valor definido na especificação do trabalho, exceto nos casos que forem alterados pelo professor.
- Um trabalho não pode ser entregue por um estudante que não esteja matriculado na turma.
- A data final de um período não pode ser anterior à data inicial do mesmo.

4.2 - Subsistema Controle Laboratório

A Figura 5 apresenta o diagrama de classes do subsistema Controle Laboratório.

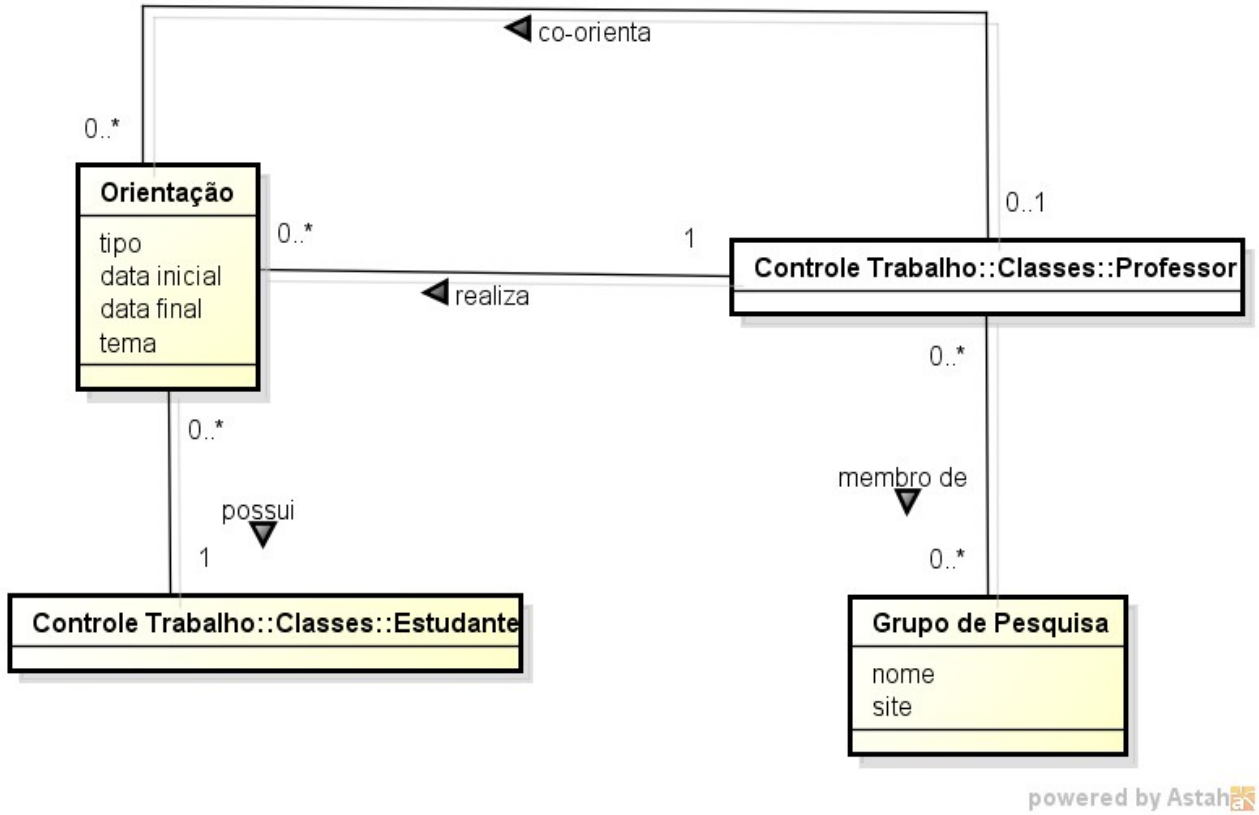

Figura 5 – Diagrama de Classes do Subsistema Controle Laboratório.

As seguintes restrições de integridade devem ser observadas:

- Um professor não pode ser um orientador e coorientador ao mesmo tempo em uma orientação.
- A data final de uma orientação não pode ser anterior à data inicial da mesma.

5. Modelo Dinâmico

O modelo dinâmico visa capturar o comportamento dinâmico do sistema. A seguir, são apresentados os diagramas de atividades elaborados no contexto deste projeto.

A Figura 6 apresenta o diagrama de atividades da classe Trabalho Entregue do subsistema Controle Trabalho.

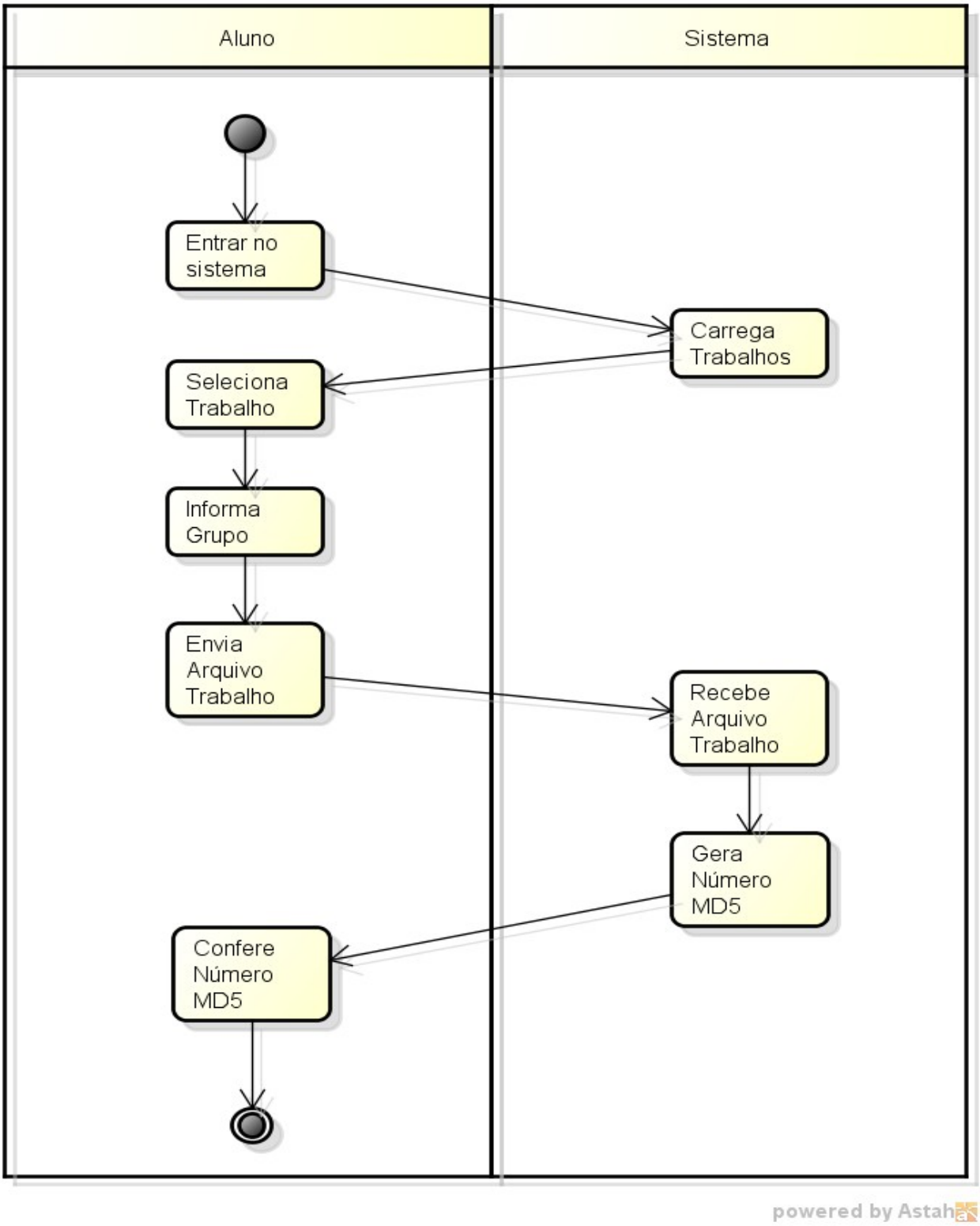

Figura 6 – Diagrama de Atividades da Classe Trabalho Entregue.

6. Glossário de Projeto

Esta seção apresenta as definições dos principais conceitos envolvidos no projeto. Essas definições estão organizadas por subsistema.

6.1 - Subsistema Controle Trabalho

- **Professor:** professores que utilizam o sistema.
  - *nome*: nome do professor.
  - *email*: email do professor.
  - *senha*: senha para acessar o sistema.
  - é administrador?: campo que irá indicar se o professor possui as funcionalidades de administrador ou não.
  - *Grupo de Pesquisa*: grupos de pesquisa do professor.

- **Disciplina:** disciplinas oferecidas.
  - *nome*: nome da disciplina.
  - *código*: código da disciplina.

- **Curso:** cursos disponíveis.
  - *nome*: nome do curso.
  - *código*: código do curso.

- **Turma:** turmas ofertadas.
  - *Periodo*: período da turma.
  - *Curso*: cursos associados a turma.
  - *Disciplina*: disciplina associada a turma.
  - *Estudante*: estudantes matriculados na turma.
  - *Professor*: professor responsável pela turma.
  - *Trabalho*: trabalhos associados a turma.

- **Periodo:** períodos do ano letivo.
  - *ano*: ano do período.
  - *número*: número do período.
  - *data inicial*: data inicial do período.
  - *data final*: data final do período.

- **Estudante:** estudantes registrados no sistema.
  - *nome*: nome do estudante.
  - *matrícula*: matrícula do estudante.
  - *email*: e-mail do estudante.
  - *senha*: senha para acessar o sistema.
  - *telefone*: telefone do estudante.

- **Trabalho:** trabalhos práticos das disciplinas.
  - *número*: número do trabalho.
  - *assunto*: assunto do trabalho.
  - *valor*: valor do trabalho.
  - *data de entrega*: data de entrega do trabalho.
  - *número máximo participantes*: número máximo de participantes do trabalho.
  - *valor desconto atraso*: valor que poderá ser descontado por dia de atraso na entrega do trabalho. Essa informação é opcional, fica a critério do professor poder utilizar ou não.

- **Grupo Trabalho:** grupos de estudantes para um trabalho.
  - *número*: número do grupo de trabalho.
  - *Estudante*: lista de estudantes que integram o grupo de trabalho.

- **Trabalho Entregue:** trabalhos que foram enviados pelos grupos.
  - *arquivo*: caminho completo onde o trabalho entregue foi salvo pelo sistema.
  - *data de entrega*: data de entrega do trabalho.

    md5: número verificador calculado pelo sistema no recebimento do trabalho. Poderá ser utilizado pelo aluno para verificar se o trabalho foi entregue corretamente.

6.2 - Subsistema Controle Laboratório

- **Grupo de Pesquisa:** grupos de pesquisas existentes no Departamento de Informática da Universidade Federal do Espírito Santo (DI/Ufes).
  - *nome*: nome do grupo de pesquisa.
  - *site*: site do grupo de pesquisa.

- **Orientação:** informações sobre a orientação do estudante.
  - *orientador:* professor que realiza a orientação do estudante.
  - *coOrientador*: professor que realiza a co-orientação do estudante.
  - *tipo*: tipo da orientação (IC, TCC, mestrado ou doutorado).

- *data inicial*: data inicial da orientação.
- *data final*: data final da orientação.
- *tema*: tema da orientação do estudante.

Documento de Projeto de Sistema

Projeto: “**SAP** - Sistema de Apoio aos Professores”

Registro de Alterações:

Versão	Responsáveis	Data	Alterações
1.0	Luiz Vitor	05/08/2015	Versão inicial.
2.0	Luiz Vitor	28/08/2015	Alteração na tabela com as tecnologias utilizadas. Modelos FrameWeb corrigidos.

1. Introdução

Este documento apresenta o documento de projeto (*design*) do sistema “**SAP** - Sistema de Apoio aos Professores”. 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.

Tabela 1 – Plataforma de Desenvolvimento (Utilizadas em Runtime).

Tecnologia	Versão	Descrição	Propósito
Java EE	7	Linguagem de programação orientada a objetos e independente de plataforma.	Desenvolvimento de aplicativos em linguagem de programação orientada a objetos e independente de plataforma.
MySQL Server	5.6.23	Sistema Gerenciador de Banco de Dados Relacional gratuito.	Persistência dos dados manipulados pela ferramenta.
Java Persistence API (JPA)	2.1	API para persistência de dados por meio de mapeamento objeto-relacional.	Padronizar o mapeamento objeto-relacional por meio de uma interface de persistência comum aos principais

			frameworks de mapeamento objeto-relacional.
Contexts and Dependency Injection for Java EE (CDI)	1.1	*Framework* gratuito para injeção de dependências.	Integração das diferentes camadas da arquitetura e serviços de transação.
Hibernate	3.4	*Framework* de mapeamento objeto-relacional gratuito que implementa a JPA.	Fazer mapeamento objeto-relacional.
WildFly	8.1.0.Final	Servidor Web para Java.	Prover acesso a aplicações web por meio do protocolo HTTP (*HyperText Transfer Protocol*).

Na Tabela 2 são listadas as tecnologias utilizadas na implementação da ferramenta, bem como o propósito de sua utilização.

Tabela 2 – Tecnologias Utilizadas Durante a Implementação.

Tecnologia	Versão	Descrição	Propósito
Eclipse Java EE IDE for Web Developers	4.4.1	Ambiente de desenvolvimento (IDE) para a linguagem Java.	Facilitar a atividade de implementação de software.
Astah Community	6.9.0	Ferramenta para modelagem em UML.	Ferramenta de modelagem UML (*Unified Modeling Language*).
BrOffice	3.3.1	*Suite* gratuita de aplicativos para escritório.	Apoiar a elaboração da documentação.
Apache Maven	3.2.5	Ferramenta de gerência de projeto baseada em *project object model* (POM).	Simplificar o download das dependências 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 .

Tabela 3 – Atributos de Qualidade e Táticas Utilizadas.

Categoria	Requisitos Não Funcionais Considerados	Condutor da Arquitetura	Tática
Facilidade de Aprendizado	RNF02	Sim	• Prover ao usuário a capacidade de entrar com comandos que permitam operar o sistema de modo mais eficiente. Para tal, as interfaces do sistema devem permitir, sempre que possível, a entrada por meio de seleção ao invés da digitação de campos.

Segurança de Acesso	RNF04	Sim	• Identificar usuários usando *login* e autenticá-los por meio de senha.
Portabilidade	RNF01	Sim	• Organizar a arquitetura da ferramenta segundo uma combinação de camadas e partições. • A camada de lógica de negócio deve ser organizada segundo o padrão Camada de Serviço. • A camada de gerência de dados deve ser organizada segundo o padrão DAO. • Utilizar uma linguagem que seja compatível com os principais navegadores do mercado.
Reusabilidade	RNF03	Sim	• Reutilizar componentes e *frameworks* existentes. No caso, foi reutilizado o nemo-utils. • Quando não houver componentes disponíveis e houver potencial para reúso, devem-se desenvolver novos componentes para reúso.

4. Arquitetura de Software

A arquitetura de software do sistema SAP, baseia-se na combinação de camadas e módulos. Cada um desses módulos, por sua vez, 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 dois módulos do SAP (Controle Trabalho e Controle Laboratório). 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.

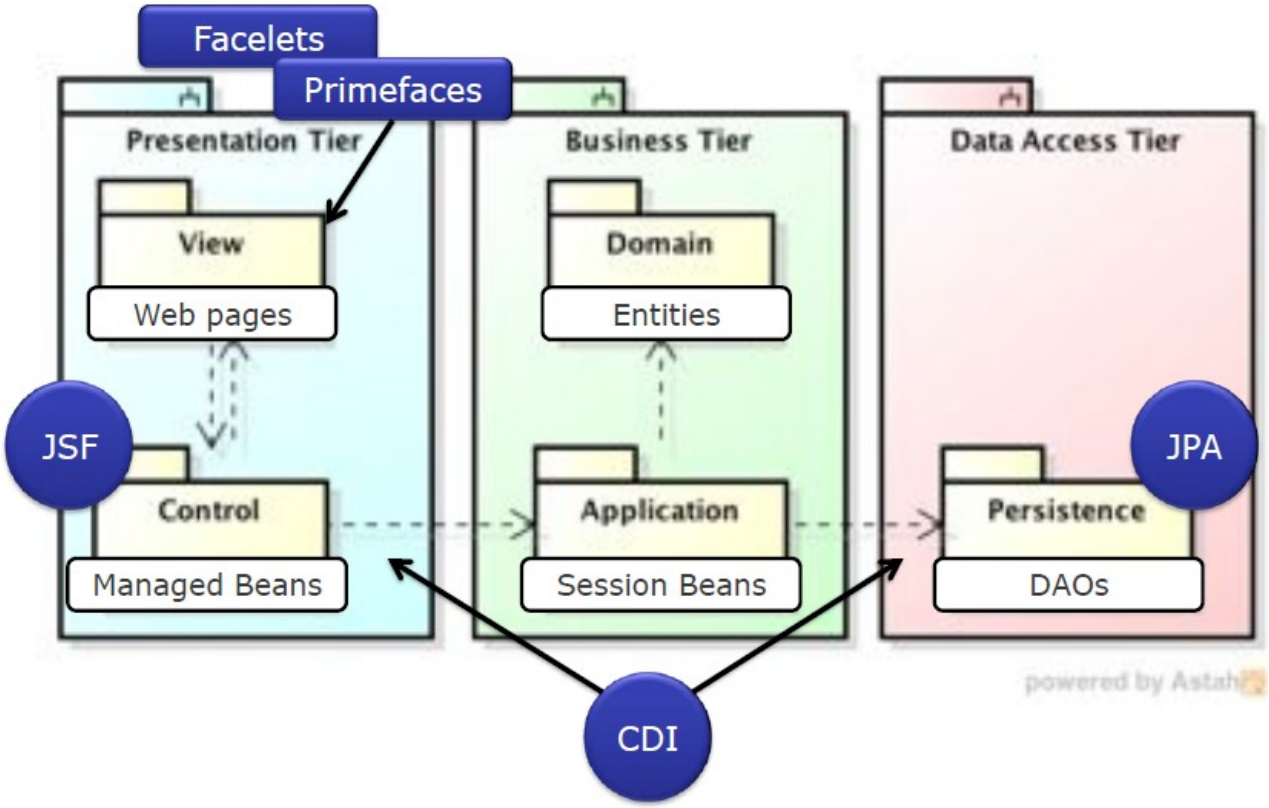

Figura 1 – Arquitetura de Software do Sistema SAP.

A figura 2 apresenta a subdivisão de cada módulo nas camadas descritas acima, a saber a Camada de Apresentação (*controller*), Camada de Negócios (*domain e application*) e Camada de Acesso a Dados (*persistence*).

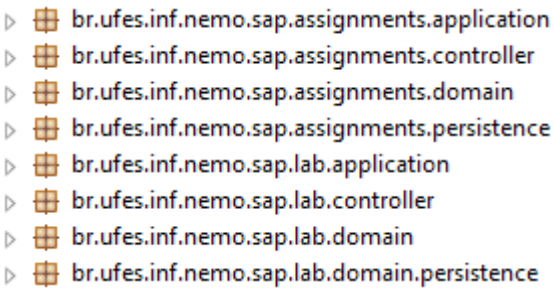

Figura 2 - Subdivisão dos módulos "Controle Trabalho (*assignments*)" e "Controle Laboratório (*lab*)" de acordo com a arquitetura de software.

4.1 - Camada de Apresentação

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 a utilização da ferramenta *nemo-utils*. Esse fluxo de execução similar é representado abaixo através de um modelo de apresentação genérico.

Figura 3 – Modelo de Navegação de um CRUD *nemo-utils*, usado como base para funcionalidades dos cadastros do sistema SAP.

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. Segue abaixo o modelo de navegação para o caso de uso “Associar Estudante a Turma”.

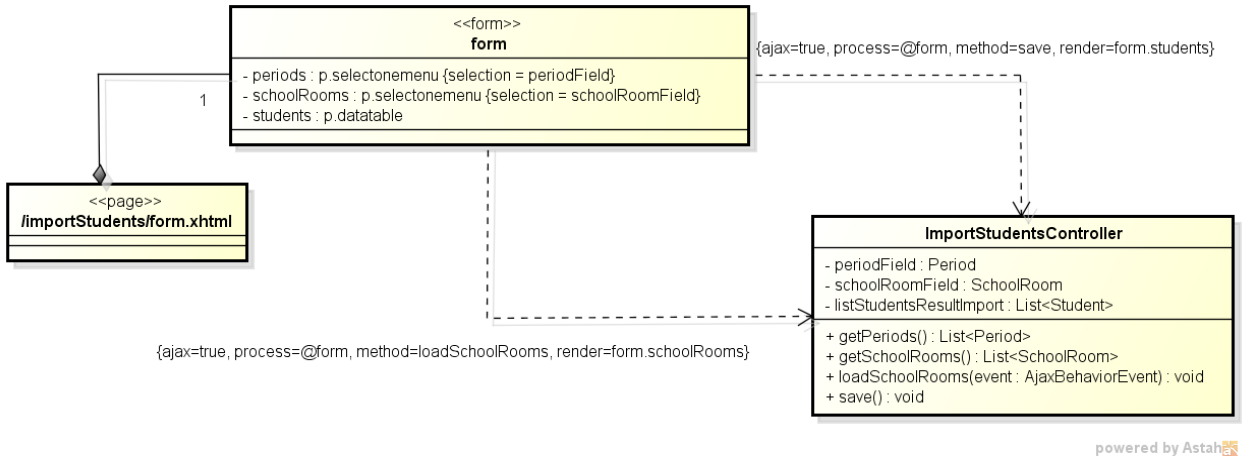

Figura 4 – Modelo de Navegação do caso de uso “Associar Estudante a Turma”.

4.2 – Camada de Negócios

Diferente da abordagem original do *FrameWeb* original proposto em 2007, todos os atributos que são não nulos tiveram a *tag* {not null} omitida e os que são nulos tiveram a *tag* {null} acrescida de forma a diminuir a poluição visual com repetições desnecessárias no diagrama.

Todas as classes de domínio estendem de *PersistentObjectSupport* do pacote *nemo-utils*, onde essa herança não é mostrada no diagrama acima com o intuito de não poluir o diagrama com várias associações.

Abaixo segue o modelo de domínio para os módulos “Controle Trabalho” e “Controle Laboratório”.

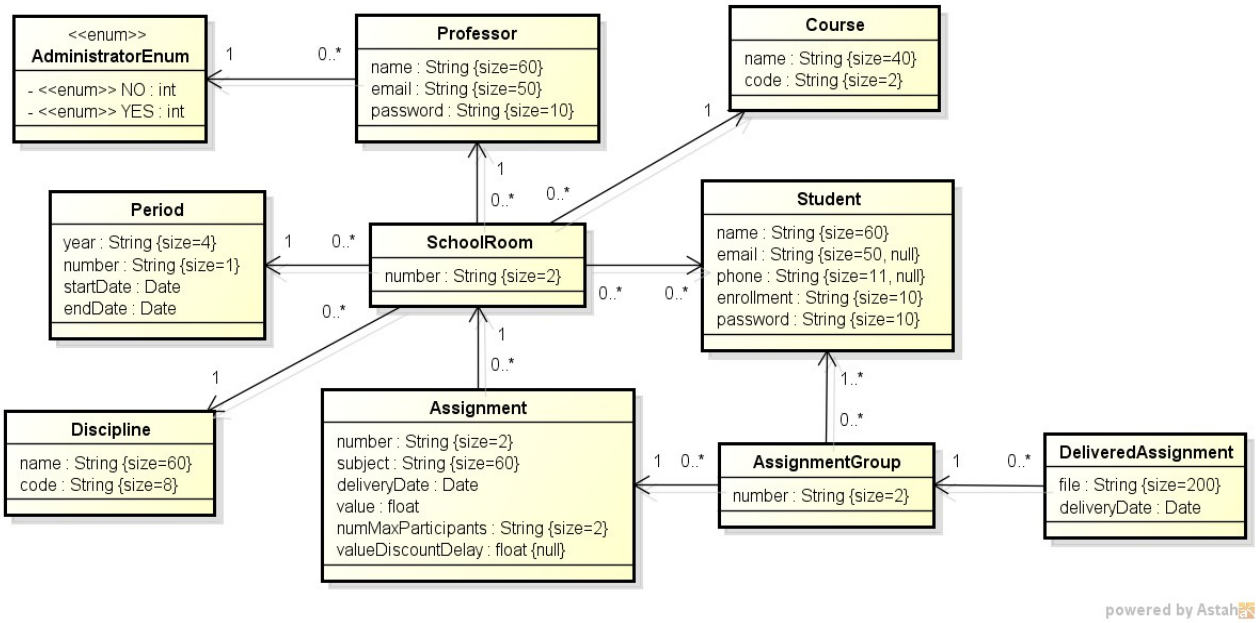

Figura 5 – Modelo de Domínio do SAP para o módulo “Controle Trabalho”.

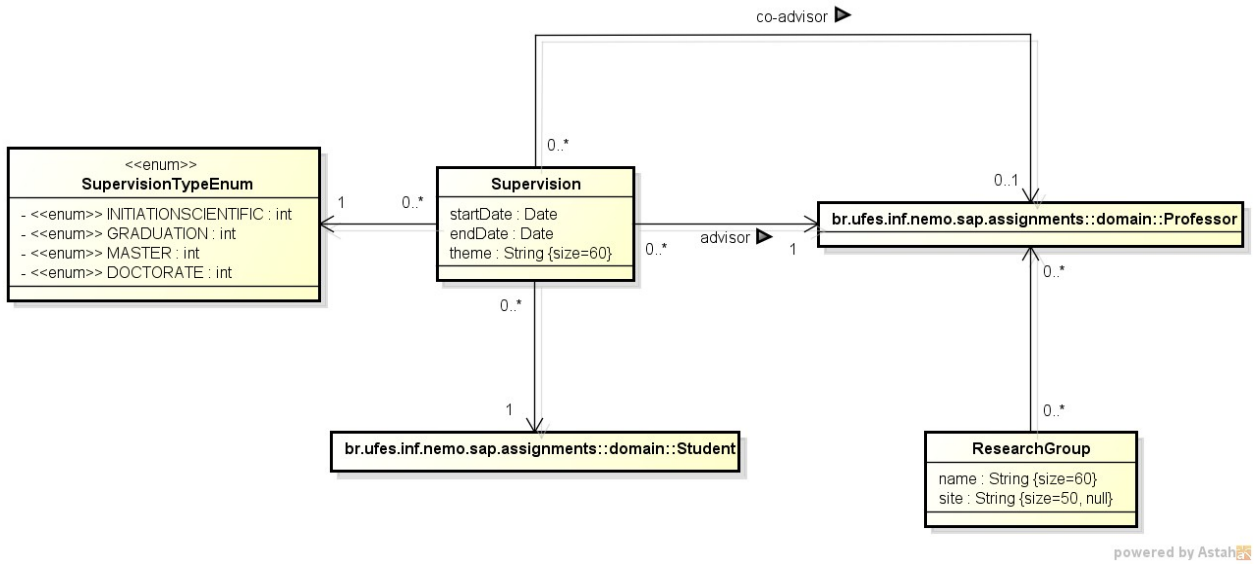

Figura 6 – Modelo de Domínio do SAP para o módulo “Controle Laboratório”.

Todas as classes de aplicação estendem de *CrudServiceBean* do pacote *nemo-utils*, tal classe está representada abaixo de forma genérica. Da mesma forma dos diagramas anteriores essa herança não é mostrada no diagrama acima com o intuito de não poluir o diagrama com várias associações. Abaixo, segue o modelo de aplicação para os módulos “Controle Trabalho” e “Controle Laboratório”.

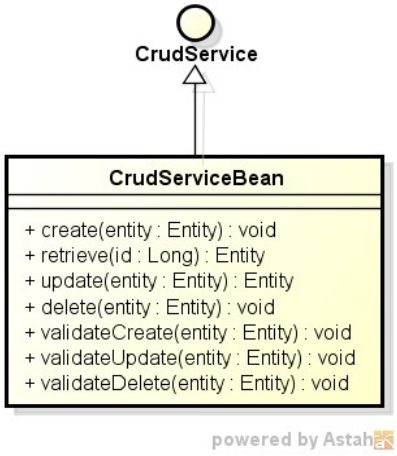

Figura 7 – Modelo de Aplicação genérica da ferramenta *nemo-utils*.

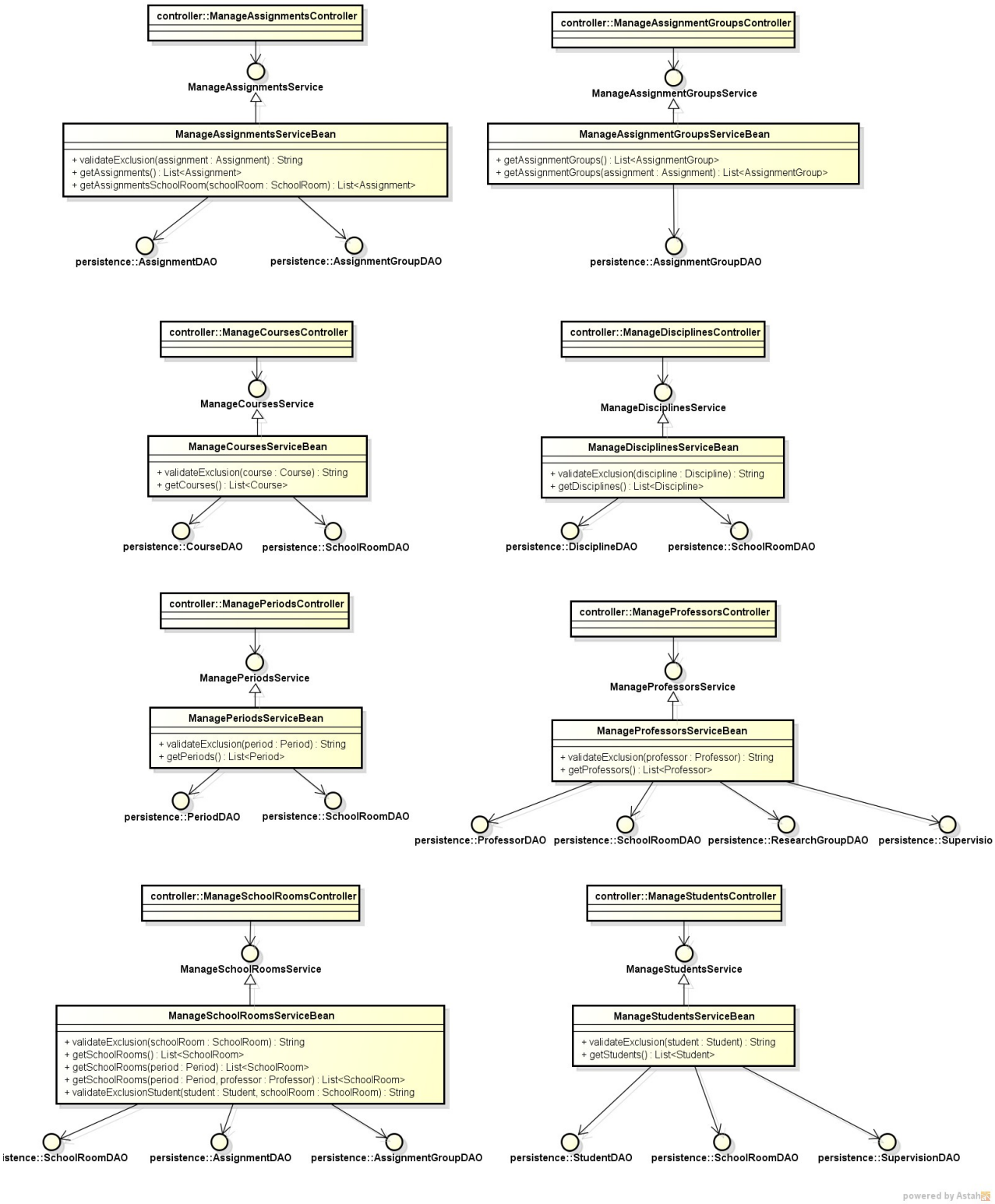

Figura 8 – Modelo de Aplicação do SAP para o módulo “Controle Trabalho”.

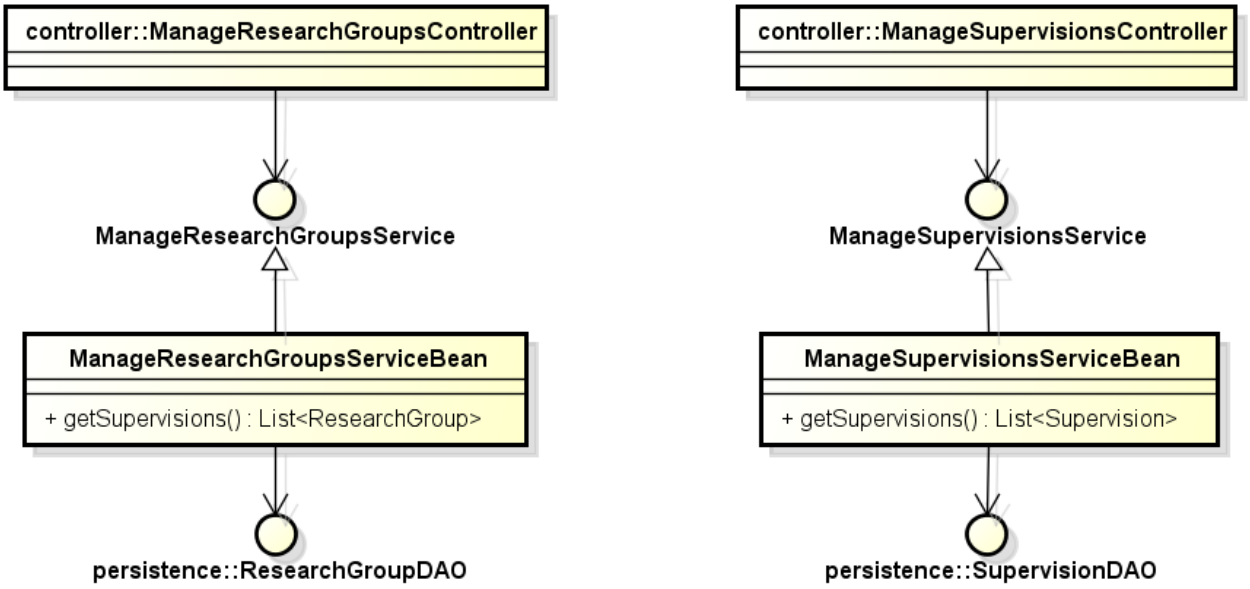

Figura 9 – Modelo de Aplicação do SAP para o módulo "Controle Laboratório".

4.3 – Camada de Acesso a Dados

Nas figuras abaixo são apresentados os Modelos de Persistências desenvolvidos para o projeto SAP, os mesmos 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 abaixo está representado o diagrama de persistência genérico provido pela ferramenta o *nemo-utils* e o modelo para os módulos "Controle Trabalho" e "Controle Laboratório".

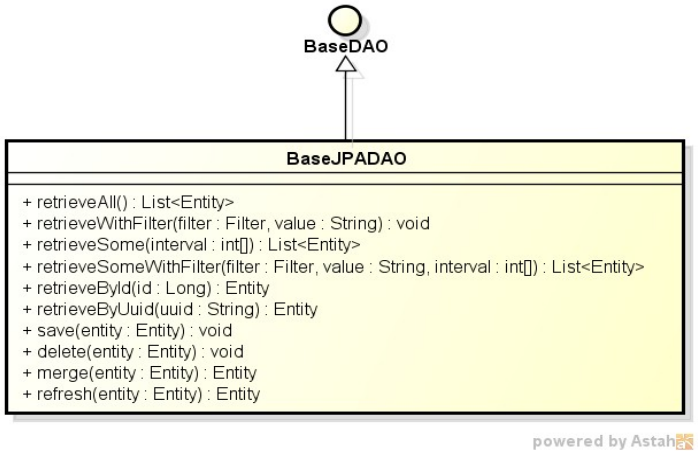

Figura 10 – Modelo de Persistência genérico da ferramenta *nemo-utils*.

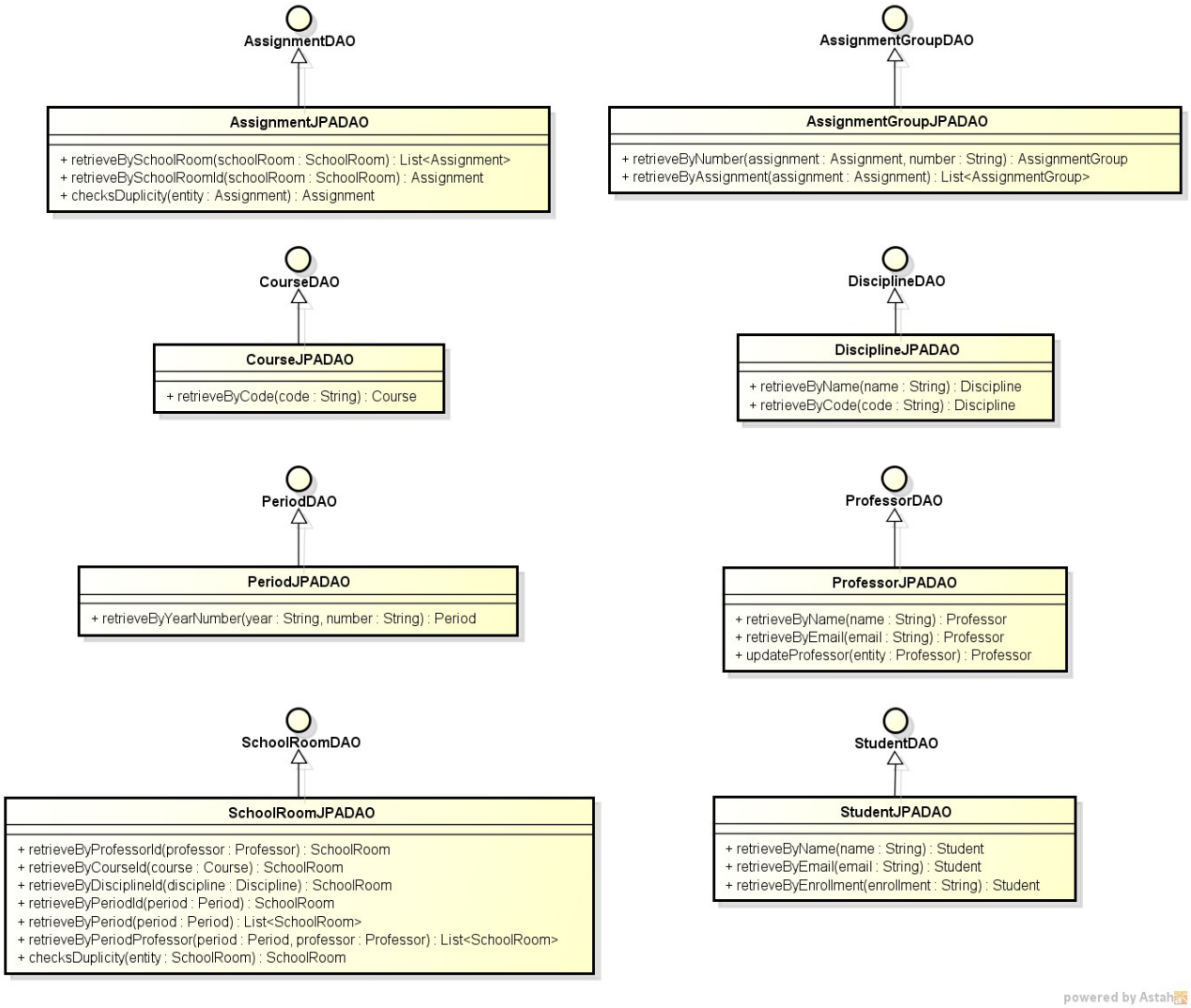

Figura 11 – Modelo de Persistência do SAP para o módulo “Controle Trabalho”.

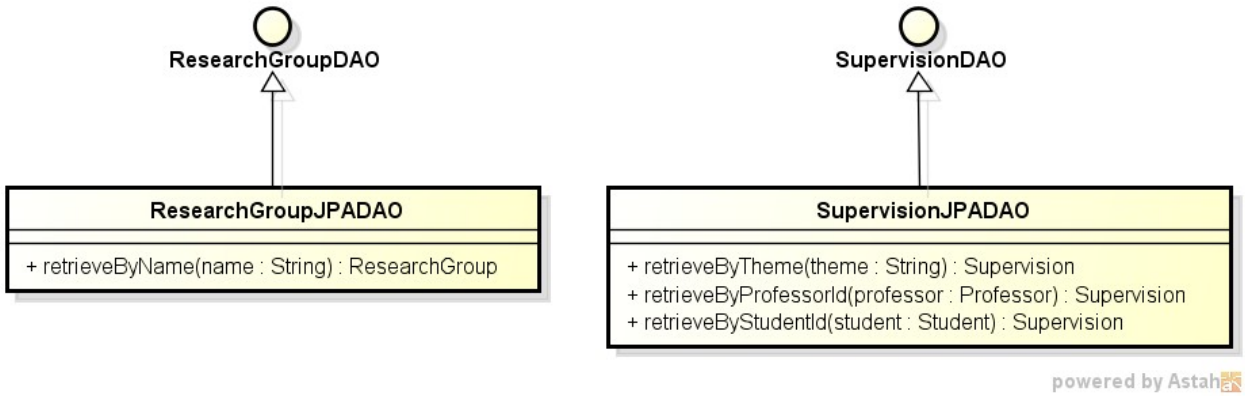

Figura 12 – Modelo de Persistência do SAP para o módulo “Controle Laboratório”.

Note 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.