

**CRISLAINE DA SILVA TRIPOLI
LUIS ANTONIO TAVARES**

SAS – SISTEMA DE AVALIAÇÕES E SIMULADOS

**UNIVERSIDADE DO VALE DO SAPUCAÍ
POUSO ALEGRE
2010**

**CRISLAINE DA SILVA TRIPOLI
LUIS ANTONIO TAVARES**

SAS – SISTEMA DE AVALIAÇÕES E SIMULADOS

Trabalho de Conclusão de Curso apresentado ao Departamento de Sistemas de Informação da Faculdade de Filosofia, Ciências e Letras “Eugênio Pacelli” da Universidade do Vale do Sapucaí, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Márcio Emílio Cruz Vono de Azevedo.

**UNIVERSIDADE DO VALE DO SAPUCAÍ
POUSO ALEGRE
2010**

**CRISLAINE DA SILVA TRIPOLI
LUIS ANTONIO TAVARES**

SAS – SISTEMA DE AVALIAÇÕES E SIMULADOS

Trabalho de conclusão de curso defendido e aprovado em ___/___/_____ pela banca examinadora constituída pelos professores:

Professor MÁRCIO EMÍLIO CRUZ VONO DE AZEVEDO
Orientador

Professora MS. VALÉRIA PADUAN DA SILVA
Examinadora

Professor FABIO AUGUSTO OLIVEIRA
Examinador

Dedicamos

Aos nossos amigos e familiares por estarem sempre presente durante mais essa etapa da nossa vida.

AGRADECIMENTOS

Agradecemos a DEUS pela sabedoria, saúde e possibilidade de finalização do trabalho.

Agradecemos ao nosso professor orientador Márcio pelo tempo despendido com nossas dúvidas e orientações, e por todo conhecimento que nos propiciou durante o curso.

Agradecemos à Prof.^a Ms. Valéria por sua atenção e ajuda nas definições do projeto.

Agradecemos à Prof.^a Dra. Joelma e ao Prof. José Luiz pelo empenho no percurso deste trabalho.

Agradecemos à UNIVÁS por nos propiciar as condições e conhecimentos necessários.

"Julgue um homem pelas suas perguntas, não pelas suas respostas."

(Voltaire)

TAVARES, Luis Antonio; TRIPOLI, Crislaine da Silva. **SAS - Sistema de Avaliações e Simulados**: Trabalho de Conclusão de Curso. Graduação. Sistemas de Informação. Universidade do Vale do Sapucaí. Pouso Alegre. 2010

RESUMO

Este trabalho tem como objetivo apresentar uma proposta de *software* para auxiliar nos processos avaliativos. A ferramenta desenvolvida possibilita ao professor a geração de avaliações e simulados, correção e análise de desempenho dos mesmos. Os testes poderão ser realizados *online* ou poderão ser impressos. Alunos também poderão acessar relatórios de desempenho. A modelagem do sistema foi feita utilizando a metodologia de desenvolvimento ICONIX que é apresentada no decorrer do trabalho. O sistema foi desenvolvido utilizando a linguagem Java, e alguns *frameworks* como Spring e EclipseLink. Além das tecnologias, são abordados conceitos relacionados à avaliação, ao ENADE e à avaliação institucional.

Palavras-chave: Avaliação. Simulado. Ferramenta. Spring. Java. Iconix.

TAVARES, Luis Antonio; TRIPOLI, Crislaine da Silva. **EST – Exam Simulation Tool**: Trabalho de Conclusão de Curso. Graduação. Sistemas de Informação. Universidade do Vale do Sapucaí. Pouso Alegre. 2010

ABSTRACT

This project presents a proposal for software that will assist in educational activities and student preparation through simulation of educational exams. The developed tool allows teachers to generate and to correct simulations of educational tests. The tests can be done online or can be printed and students and teachers can access performance reports. The modeling of the system was designed using the ICONIX modeling process that is presented in this paper. The system was developed using the Java programming language and frameworks such as Spring and Eclipse Link. This paper also discusses the concepts related to educational tests, the ENADE exam and institutional evaluation.

Keywords: School Test. Exam Simulator Tool. Spring. Java. Iconix.

LISTA DE FIGURAS

Figura 1: Interação entre um Aplicativo Java e um Sistema Operacional	25
Figura 2: Java EE Server, Componentes e <i>Containers</i>	27
Figura 3: Interação entre Servlets, Servidor, <i>Container</i> e acesso a Regras de Negócio ..	28
Figura 4: Processo da primeira execução de uma página JSP	31
Figura 5: Funcionamento do Container IoC do Spring	36
Figura 6: Estrutura dos módulos do Spring Framework.	39
Figura 7: O ciclo de vida de uma requisição no Spring MVC	41
Figura 8: Aplicação utilizando JPA	43
Figura 9: Elementos Chaves da Abordagem do ICONIX	45
Figura 10: Exemplo de Modelo de Domínio.....	49
Figura 11: Análise de Robustez	51
Figura 12: Tela de Cadastro de Professor	54
Figura 13: Tela para realização de Simulados	54
Figura 14: Diagrama de Fluxo de Navegação.....	55
Figura 15: Modelo de domínio.....	56
Figura 16: Componentes do Caso de Uso.....	57
Figura 17: Casos de Uso.	58
Figura 18: Estereótipos da análise de robustez.	61
Figura 19: Diagrama de robustez do caso de uso Gerar Simulados Impressos e Online	62
Figura 20: Diagrama de robustez do caso de uso Realizar Simulado <i>Online</i>	62
Figura 21: Atualização do Modelo de Domínio adicionando os atributos das Entidades	63
Figura 22: Diagrama de Sequência Cadastrar Questões	65
Figura 23: Diagrama de Sequência Gerar Simulados Impressos e <i>Online</i>	65
Figura 24: Conclusão do Modelo Estático	67

LISTA DE ABREVIATURAS E SIGLAS

EAD	Educação à Distância
ENADE	Exame Nacional de Desempenho dos Estudantes
HTML	<i>Hyper Text Markup Language</i>
IES	Instituição de Ensino Superior
Java EE	<i>Java Enterprise Edition</i>
JMS	<i>Java Message Service</i>
JMX	<i>Java Management Extensions</i>
JPA	<i>Java Persistence API</i>
JSP	<i>Java Servers Pages</i>
LDB	Lei de Diretrizes e Bases da Educação Nacional
MVC	<i>Model-View-Controller</i>
OO	Orientação a Objetos
POA	Programação Orientada a Aspecto
RUP	<i>Rational Unified Process</i>
SAS	Sistema de Avaliações e Simulados
SINAES	Sistema Nacional de Avaliação da Educação Superior
SCJP	<i>Sun Certified Java Programmer</i>
SpEL	<i>Spring Expression Language</i>
XML	<i>Extensible Markup Language</i>
XP	<i>Extremme Programming</i>
UML	<i>Unified Modeling Language</i>

LISTA DE TABELAS

Tabela 1: Lista de requisitos funcionais.....	53
Tabela 2: Fluxo de eventos para caso de uso Cadastrar Professor.....	59
Tabela 3: Fluxo de eventos para caso de uso Realizar Simulado Online.....	60
Tabela 4: Caso de teste para Realização de Simulados ou Avaliações.....	70

SUMÁRIO

INTRODUÇÃO	13
2 AVALIAÇÃO	17
2.1 Tipos de avaliação.....	17
2.2 Como a tecnologia pode auxiliar o processo avaliativo.....	18
2.3 Avaliação institucional.....	19
2.3.1 Objetos e objetivos da avaliação institucional	20
2.3.2 Funções da avaliação institucional.....	21
2.4 Exame Nacional de Desempenho dos Estudantes - ENADE.....	22
2.5 Cursos pré-vestibulares	22
3 TECNOLOGIAS	24
3.1 Java.....	24
3.1.1 Java Enterprise Edition - Java EE	25
3.1.1.1 Servlet	27
3.1.1.2 Java Server Pages - JSP	30
3.2 Spring 3.0	32
3.2.1 Injeção de dependência	35
3.2.2 Módulos	39
3.2.3 Spring MVC	40
3.3 Java Persistence API	41
3.3.1 EclipseLink	43
4 METODOLOGIA DE DESENVOLVIMENTO	44
4.1 Origem do ICONIX e questões fundamentais	45
4.2 Fases e marcos do ICONIX.....	46
4.2.1 Análise de requisitos	46
4.2.2 Análise e projeto preliminar.....	47
4.2.3 Projeto	47
4.2.4 Implementação	48
4.3 Técnicas usadas no ICONIX.....	48
4.3.1 Modelo de domínio	48
4.3.2 Casos de uso.....	49

4.3.3 Diagrama de sequência	50
4.3.4 Análise de robustez	50
5 DESENVOLVIMENTO DO PROJETO UTILIZANDO ICONIX.....	52
5.1 Análise de requisitos	52
5.1.1 Prototipação	53
5.1.2 Modelo de domínio	55
5.1.3 Identificação dos casos de uso	56
5.2 Análise e projeto preliminar.....	58
5.2.1 Descrição dos casos de uso	58
5.2.2 Análise de robustez	60
5.2.3 Atualização do modelo de domínio	63
5.3 Projeto	64
5.3.1 Diagramas de sequência.....	64
5.3.2 Conclusão do modelo estático.....	66
5.4 Desenvolvimento do projeto	68
5.4.1 Codificação	68
5.4.2 Testes	69
6 DISCUSSÃO DOS RESULTADOS	71
CONCLUSÃO	73
REFERÊNCIAS	74
APÊNDICES	77
ANEXOS	105

INTRODUÇÃO

Este trabalho tem como objetivo apresentar a implementação do Sistema de Avaliações e Simulados - SAS¹, uma ferramenta especialmente voltada para o ambiente *web*² na qual o professor pode gerar simulados impressos ou online com questões provenientes do banco de dados do sistema. Os objetivos específicos do trabalho são: a) demonstrar alguns recursos do *framework*³ Spring; b) criar a modelagem e a documentação de engenharia de software do sistema e; c) implementar o sistema.

Como este é um *software* voltado para a realização de avaliações e para preparação dos alunos para a realização das mesmas, um dos conceitos importantes a serem abordados é a avaliação.

Avaliar é uma prática constante na vida das pessoas. Avalia-se a todo momento. No convívio social, no trabalho, no lazer, sempre há julgamentos sendo feitos sobre atos, comportamentos, decisões que foram tomadas e outras inúmeras situações cotidianas.

Este cenário não poderia ser diferente no ambiente educacional, muito embora seja tomada de maneira formalizada. De acordo com a Lei de Diretrizes e Bases da Educação Nacional - LDB⁴, a avaliação é garantida nos níveis de ensino fundamental, médio e superior (LDB, 1996).

As práticas avaliativas nos níveis de ensino citados acima, assim como nos cursos preparatórios e outros afins, comumente fazem uso de simulados e provas objetivas, onde recai o foco deste projeto que propõe uma ferramenta para apoiar estes processos, abordando o aspecto operacional (realização e correção) e o aspecto gerencial (relatórios de desempenho) dos mesmos.

A aplicação foi desenvolvida em ambiente *web* por este proporcionar grandes diferenciais, alguns dos quais são apresentados no decorrer do trabalho, como por exemplo, a facilidade de poder ser acessado em qualquer computador que disponha de acesso a internet.

¹ O termo Sistema de Avaliações e Simulados será, doravante, designado por SAS.

² *Web* é um termo proveniente de World Wide Web que significa “rede de alcance mundial” em inglês.

³ *Framework* é uma abstração que une códigos comuns entre vários projetos de *software* provendo uma funcionalidade genérica.

⁴ O termo Lei de Diretrizes e Bases da Educação Nacional será, doravante, designado por LDB.

Por esta característica, esta ferramenta pode também ser utilizada em avaliações na modalidade de ensino à distância – EAD. “Pensar a avaliação num cenário de expansão da educação a distância parece essencial para que tenhamos mais garantias de um futuro de sucesso para essa modalidade educacional.” (MAXIMO *et al*, 2008, p.2).

A necessidade da implementação de um sistema para auxiliar instituições de ensino na realização de provas objetivas e simulados foi notada pelos próprios participantes deste projeto, que observaram nos ambientes educacionais em que trabalharam e estudaram a carência por um *software* que ajude nos processos avaliativos. Essa necessidade também foi relatada por professores da universidade e por professores de cursos técnicos de outras instituições.

O sistema proposto por este trabalho abrange turmas de nível superior, médio e fundamental que utilizam provas objetivas no seu processo de aprendizagem. Em cursos pré-vestibulares e preparatórios para concursos este projeto também possui uma grande aplicabilidade, pois são cursos que trabalham com um número grande de questões e a maioria delas são objetivas, o que faz com que um banco de questões seja de extrema utilidade. O sistema também pode ser usado em simulados para o ENADE e em avaliações institucionais, estas situações são abordadas no decorrer do trabalho.

Em muitas das aplicações demandadas atualmente, há uma certa complexidade envolvida em sua implementação, com isso os desenvolvedores necessitam de soluções que auxiliem no processo de desenvolvimento de forma que o mesmo possa ser simplificado. Com este objetivo surgiram vários frameworks, cada qual direcionado para uma ou mais etapas ou camadas de um projeto.

Para este projeto o *framework* Spring foi utilizado em grande parte de sua implementação e alguns de seus conceitos e recursos são abordados do decorrer do trabalho.

Diante deste contexto, a relevância deste trabalho se justifica por propor uma ferramenta que auxilie nos processos avaliativos das instituições de ensino e por demonstrar os recursos do *framework* Spring utilizados no desenvolvimento da mesma.

Este trabalho configura-se como pesquisa aplicada porque tem como finalidade o estudo dos processos avaliativos e de tecnologias para desenvolvimento de aplicações *web* visando a implementação de uma solução que auxilie as instituições de ensino nessas atividades.

Trata-se de uma pesquisa aplicada por ter um resultado prático visível em termos de utilidade e não apenas pela aquisição e disseminação do conhecimento envolvido.

Segundo BARROS (2002), a pesquisa aplicada, conhecida também como pesquisa prática é caracterizada por quando o pesquisador é movido pela necessidade de conhecer, para aplicação imediata de seus resultados e tal pesquisa contribui para fins práticos.

O sistema está voltado para o ambiente educacional, podendo ser implantado em turmas de nível superior, nível médio e fundamental, sendo usado na criação, realização e correção de provas objetivas aplicadas para estas turmas.

Este trabalho tem como participantes os acadêmicos Crislaine da Silva Tripoli e Luis Antonio Tavares, e o orientador Prof. Márcio Emílio Cruz Vono de Azevedo. Crislaine atualmente é estagiária em desenvolvimento de *software*, atuando com projetos *web* e Java desde do início 2008. Luis também tem trabalhado em aplicativos Java e aplicações para *web* durante o mesmo período. O professor Márcio é certificado SCJP (*Sun Certified Java Programmer*) e trabalha com desenvolvimento de aplicativos *web*. Diante desse cenário, o lado profissional foi um grande motivador para a escolha por uma aplicação Java EE, mas esta escolha também se deve a outros grandes diferenciais já citados que uma aplicação deste tipo proporciona.

Todos os participantes já trabalharam em ambiente educacional, o que também motivou a escolha por um *software* educativo.

Os procedimentos para a realização do trabalho foram: levantar dados sobre os processos avaliativos, levantar dados sobre as tecnologias utilizadas no desenvolvimento do sistema, levantar dados sobre os procedimentos metodológicos para o desenvolvimento do projeto, desenvolver a modelagem do sistema e, por fim, a implementação do sistema.

Os dados utilizados foram coletados de livros, artigos e sites especializados. Sendo que para o desenvolvimento da aplicação foram necessárias algumas ferramentas das quais pode-se citar: o PostgreSQL (sistema de gerenciamento de banco de dados objeto-relacional), o Eclipse IDE (uma plataforma para desenvolvimento de aplicações em diversas linguagens), o Java Developing Kit (conjunto de APIs Java para desenvolvimento de aplicativos) e o JUDE (Ferramenta para desenvolvimento de diagramas de modelagem de *software*).

Nos capítulos seguintes são apresentados os conceitos e teorias que foram utilizados no desenvolvimento deste trabalho. Os assuntos abordados são avaliação educacional, as tecnologias utilizadas na implementação do projeto e a metodologia de desenvolvimento utilizada. Dentre os autores consultados pode-se citar Perrenoud, Dornelles, Levy, Walls, Barros, Sun Microsystems e ICONIXPROCESS. Algumas das tecnologias abordadas são: orientação a objetos, Java, Java EE e o Spring *framework*. Também são apresentados os detalhes de funcionamento do sistema e seu processo de desenvolvimento.

2 AVALIAÇÃO

A avaliação é um processo que remota ao século XVII e foi a partir do século XIX que se tornou inseparável do ensino, com a chegada da escolaridade obrigatória, conforme explica PERRENOUD (1999).

Segundo DORNELLES (2004), a avaliação é um elemento integrante e regulador das práticas pedagógicas, pois se constitui em um certificador da aprendizagem, em um direcionador de decisões para o planejamento e como um avaliador do sistema educacional. SINAES (2004) afirma que a LDB consolidou a necessidade dos processos de avaliação como pilar para a educação.

Diante do que foi explicado por esses autores, fica evidente a importância da avaliação no ensino e sua obrigatoriedade perante a legislação. Porém não há uma regulamentação sobre qual tipo de avaliação ou método avaliativo a ser usado. A sessão seguinte mencionará as diferentes classificações de avaliação.

2.1 Tipos de avaliação

A avaliação pode ser classificada em quatro tipos, que são:

- **Diagnóstica:** “Tem como propósito realizar um levantamento dos conhecimentos prévios, com o objetivo de fornecer, ao professor, elementos que lhe permitam adequar o planejamento à prática educativa” (DORNELLES, 2004, p.3);
- **Emancipadora:** Tem por objetivos a modificação e a melhoria contínua, sendo um instrumento educativo da emancipação do aluno, do seu senso de autocrítica e autodesenvolvimento conforme explica DORNELLES (2004);
- **Formativa:** Trata-se de um processo de avaliação realizado no decorrer de um programa instrucional com o objetivo de aperfeiçoá-lo, afirma FERREIRA (1999);

- **Somativa:** “É a conhecida como clássica, tem por objetivo refletir os resultados obtidos em momentos específicos, traduzindo, de forma breve, codificada, o quanto de uma meta foi atingido” (DORNELLES, 2004, p.3).

Cada classificação tem seus instrumentos disponíveis para realizar a avaliação. Por exemplo, em processos diagnósticos pode-se utilizar o pré-teste e a observação. Enquanto que na avaliação somativa há a prova e o questionário. No caso de uma avaliação formativa, utilizam-se acompanhamentos, discussões e relatórios. E finalmente, numa avaliação emancipadora utilizam-se a figura do tutor como relator do processo evolutivo, a auto-avaliação e discussões com os pares.

2.2 Como a tecnologia pode auxiliar o processo avaliativo

O processo avaliativo tem sofrido profundas mudanças provocadas pela evolução tecnológica. “Com a sociedade da informação, nascem novos paradigmas de educação, que podemos batizar de educação não-tradicional ou alternativa.” (MÁTTAR NETO, 2003, p.117).

Atualmente, com a disseminação de novas tecnologias, inúmeras ferramentas têm sido desenvolvidas para auxiliar as atividades acadêmicas e até mesmo criar novos conceitos de aprendizagem, rompendo assim, com tradicionais métodos utilizados. Desta forma, pode-se dizer que cada vez mais a educação está se apoiando nesses novos recursos, ganhando assim, agilidade, eficiência e praticidade em muitos processos.

Segundo LEVY (1999), cada indivíduo inserindo neste ambiente de transformações que está sendo vivenciado nos dias atuais se encontra em um estado de desapossamento, pois os métodos de trabalho estão sendo bruscamente alterados.

Diante desta situação o ambiente educacional também se depara com a necessidade de se adaptar e de participar desse desenvolvimento, criando soluções que atendam as suas necessidades e otimize suas atividades, contexto no qual se encaixa o escopo deste projeto.

A ferramenta desenvolvida neste trabalho pode ser utilizada para os diversos tipos de avaliações e simulados que utilizem questões fechadas. Dentre algumas das situações em que se poderá fazer uso da ferramenta são abordadas: avaliações institucionais, simulados para pré-vestibulares e simulados para o Exame Nacional de

Desempenho dos Estudantes - ENADE⁵. A próxima sessão tratará de avaliação institucional.

2.3 Avaliação institucional

“A avaliação institucional é o instrumento central, organizador da coerência do conjunto.” (SINAES, 2004, p.94). É a ferramenta pela qual a instituição é avaliada para que possam ser identificados pontos de deficiência, e assim, possibilitar que a qualidade do ensino seja melhorada.

O processo de avaliação institucional é assegurado pelo Sistema Nacional de Avaliação da Educação Superior – SINAES⁶. Este sistema foi criado pela lei nº 10.861 e tem como finalidades:

(...) a melhoria da qualidade da educação superior, a orientação da expansão da sua oferta, o aumento permanente da sua eficácia institucional e efetividade acadêmica e social e, especialmente, promoção do aprofundamento dos compromissos e responsabilidades sociais das instituições de educação superior (...) (LEI 10.861, 2004, Art.1 §1)

Os processos de avaliação institucional têm como foco as Instituições de Ensino Superior – IES⁷. Segundo a regulamentação do SINAES, esses processos abordam três aspectos:

- a) o objeto de análise é o conjunto de dimensões, estruturas, relações, atividades, funções e finalidades de uma IES; dentre outros aspectos, ensino-pesquisa-extensão, administração, responsabilidade e compromissos sociais, formação, etc;
- b) os sujeitos da avaliação são os conjuntos de professores, estudantes funcionários e membros da comunidade externa especialmente convidados ou designados; e
- c) os processos avaliativos seguem os procedimentos institucionais e se utilizam da infra-estrutura da própria instituição. (SINAES, 2004, p. 94)

⁵ O termo Exame Nacional de Desempenho dos Estudantes será, doravante, designado por ENADE.

⁶ O termo Sistema Nacional de Avaliação da Educação Superior será, doravante, designado por SINAES.

⁷ O termo Instituição de Ensino Superior será, doravante, designado por IES.

A avaliação institucional deve envolver ampla participação da comunidade acadêmica e de membros da administração, fundamentando-se nos objetivos e princípios vinculados ao ideal da educação superior.

Comumente, as avaliações institucionais fazem uso de questões fechadas para que sejam avaliados os vários aspectos da instituição, principalmente no que se refere ao perfil do corpo docente. “A avaliação dos cursos de graduação resultará na atribuição de conceitos, ordenados em uma escala com 5 (cinco) níveis, a cada uma das dimensões e ao conjunto das dimensões avaliadas.” (LEI 10.861, 2004, Art.4 §2)

Deste fato surge uma aplicabilidade para o SAS, como uma ferramenta para a execução desses processos, possibilitando que, uma vez cadastradas as questões referentes à instituição, as avaliações possam ser feitas de maneira prática e rápida, assim como, corrigidas automaticamente.

2.3.1 Objetos e objetivos da avaliação institucional

O foco da avaliação institucional é o trabalho pedagógico e científico, em seu sentido técnico e formativo, e as várias atividades relacionadas aos compromissos sociais da instituição. Contudo, também é necessário compreender as relações sociais e as condições de trabalho, a eficiência administrativa e a eficácia dos processos interpessoais da instituição, explica SINAES (2004).

É imprescindível conhecer as condições de sustentabilidade e continuidade e todos os dados importantes da infra-estrutura, especialmente aqueles mais diretamente relacionados com a pesquisa e com o ensino, como laboratórios, bibliotecas, equipamentos, instrumentos técnicos, etc., sem nunca perder de vista as finalidades e objetivos primordiais da instituição educativa. É também de enorme importância a apreciação crítica dos fluxos de informação, bem como a análise do funcionamento das câmeras, conselhos, comissões e outras estrutura colegiadas da instituição (SINAES, 2004, p.97).

Dentre os objetivos da avaliação institucional estão: encontrar os pontos fracos e fortes da instituição; descobrir o grau de envolvimento dos alunos, professores e

funcionários; e, verificar se a instituição está cumprindo seu papel social. É necessário avaliar a forma como o ensino está se desenvolvendo, qual o papel social e profissional que o aluno está exercendo, os critérios de aprovação dos alunos e os critérios de promoção dos professores na carreira docente.

É fundamental não só identificar as falhas e carências, mas encontrar a causa das mesmas, apontando alternativas para sanar os problemas e expondo ações adequadas para promover as mudanças necessárias. Conhecer as qualidades e aspectos fortes da IES também é muito importante na avaliação institucional, assim como, além da comunidade interna, outros participantes devem conhecer e julgar o processo, participar da escolha dos temas da instituição, a forma como se constituem os grupos de pesquisa, a política de formação continuada dos docentes, a colaboração entre instituições e as necessidades de laboratórios, bibliotecas e outras estruturas básicas.

2.3.2 Funções da avaliação institucional

Os processos avaliativos realizados por cada instituição estão relacionados com funções de regulação e auto-regulação. A auto-avaliação é um instrumento básico obrigatório e seu exercício é prerrogativa do Estado.

Por meio dela, as instituições conhecerão melhor a sua própria realidade e poderão praticar os atos regulatórios internos que considerarem necessários para cumprir com mais qualidade e pertinência os seus objetivos e suas missões (SINAES, 2004, p.98).

A auto-avaliação tem como funções mais importantes: produzir conhecimentos, realizar as finalidades essenciais da instituição, identificar a causa das deficiências encontradas no processo, aumentar a consciência pedagógica e a capacidade profissional dos professores, tornar mais efetiva e expandir a relação da instituição com o meio social, discutir sobre a relevância científica e social de suas atividades e fornecer todas as informações que sejam necessárias ao conhecimento do Estado e da sociedade.

2.4 Exame Nacional de Desempenho dos Estudantes - ENADE

O ENADE é um exame nacional realizado nas instituições de ensino superior que tem como objetivo verificar o desempenho dos alunos em relação aos conteúdos previstos nas diretrizes curriculares dos respectivos cursos, explica ENADE(2004). Assim como a avaliação institucional, o ENADE é regulamentado pelo SINAES.

Os resultados obtidos no exame permitem com que sejam constituídos referenciais para a definição de medidas que visem à melhoria da qualidade dos cursos de graduação, por parte dos professores, técnicos, dirigentes e autoridades educacionais. Estes resultados são expressos numa escala de cinco níveis e divulgados aos alunos, à instituição, aos órgãos de regulação e à sociedade em geral.

“O ENADE será aplicado periodicamente, admitida a utilização de procedimentos amostrais, aos alunos de todos os cursos de graduação, ao final do primeiro e do último ano de curso” (LEI 10.861, 2004, Art.5 §2).

Este exame é componente curricular obrigatório dos cursos de graduação, sendo que no histórico escolar do estudante deverá ser atestada sua situação como regular, quando ele participou efetivamente do exame ou, quando for o caso, dispensa oficial pelo Ministério da Educação.

Muitas das questões utilizadas no ENADE são objetivas, o que faz com que o exame seja outro caso no qual o SAS pode ser empregado, trazendo como benefícios a possibilidade dos alunos terem acesso a uma base com uma variada quantidade de questões e todas as outras vantagens que o sistema em si já contempla.

2.5 Cursos pré-vestibulares

Os pré-vestibulares são cursos realizados por estudantes com ensino médio em andamento ou concluído e têm por objetivo preparar o aluno para o vestibular. Nestes cursos o estudante revisa todo o conteúdo que aprendeu durante sua vida escolar, visando obter sua classificação nos tradicionais exames vestibulares.

Nos cursos pré-vestibulares é muito comum o uso de questões de exames anteriores para estudo. Desta forma, um banco de questões como o proposto pelo sistema SAS pode facilitar o processo de aprendizado e enriquecer o conteúdo ministrado na aulas.

O capítulo seguinte apresentará as tecnologias utilizadas no desenvolvimento do projeto.

3 TECNOLOGIAS

Neste capítulo são apresentadas as tecnologias usadas na implementação do sistema assim como conceitos relacionados às mesmas. São abordados assuntos como a tecnologia Java, o *framework* Spring e JPA - *Java Persistence API*.

3.1 Java

Criada pela Sun Microsystems em 1995, Java é uma tecnologia que basicamente consiste em uma linguagem de programação e uma plataforma denominada de Máquina Virtual Java ou *Java Virtual Machine* - JVM⁸, usada para execução dos aplicativos desenvolvidos na linguagem Java e outras linguagens suportadas.

Segundo a SUN MICROSYSTEMS (2010a), a tecnologia Java possibilita aos desenvolvedores escrever aplicativos uma vez, para rodar em qualquer computador. Isto se deve a sua característica multiplataforma, pois uma vez desenvolvido, o software poderá rodar em qualquer computador e sistema operacional que já disponha de uma Máquina Virtual Java instalada, diferentemente de um software em Pascal ou C por exemplo, onde o código fonte é compilado para um sistema operacional específico.

A figura 1 ilustra como a JVM faz a interação com o sistema operacional e o aplicativo em Java. O compilador Java gera um código binário chamado de *bytecode*, a JVM irá “traduzir” este código para o sistema operacional, desta forma o mesmo código que é executado em sistema operacional Windows poderá também ser executado em um Linux por exemplo.

⁸ O termo *Java Virtual Machine* será, doravante, designado por JVM.

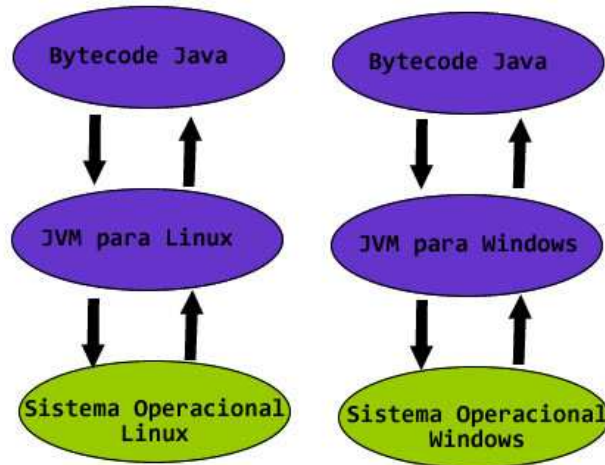


Fig. 1: Interação entre um Aplicativo Java e um Sistema Operacional
Fonte: Elaborado pelos autores.

Pode-se também destacar outras importantes características da linguagem como a orientação a objetos - OO⁹, o ambiente para programação *multithreaded* e a portabilidade.

De acordo com a SUN MICROSYSTEMS (2010b), a tecnologia Java se divide em 3 plataformas: tecnologia Java para dispositivos pequenos e móveis, tecnologia Java para PC desktops e tecnologia Java para médios e grandes negócios. A plataforma utilizada neste trabalho é a última citada, também conhecida por Java Enterprise Edition (Java EE).

3.1.1 Java Enterprise Edition - Java EE

A plataforma Java EE é o padrão utilizado para o desenvolvimento de aplicações em ambientes corporativos com Java. Conforme explica a SUN MICROSYSTEMS (2010c) esta plataforma provê aos desenvolvedores um poderoso conjunto de APIs, reduzindo o tempo de desenvolvimento, reduzindo a complexidade e aumentando a performance da aplicação.

Segundo ORACLE (2010), os componentes base e a plataforma independente desta tecnologia, fazem com que se torne fácil escrever aplicações Java EE, uma vez

⁹ O termo orientação a objetos será, doravante, designado por OO.

que a lógica de negócio está organizada em componentes reutilizáveis. Além disso, o servidor Java fornece serviços sob a forma de *container* para cada tipo de componente.

Na arquitetura Java EE a finalidade básica de um *container* é fornecer o ambiente de execução para os componentes. A especificação Java EE define um contrato entre componentes e *container* e especifica o modelo de desenvolvimento destes componentes. Este contrato especifica como estes componentes devem ser desenvolvidos e implementados e como os componentes podem acessar serviços fornecidos pelo *container*.

De acordo com a ORACLE (2010), a especificação Java EE destaca cinco componentes ou *containers* suportados em um produto Java EE. São eles:

- Java EE Server: é a parte de execução de um produto Java EE. Um Java EE Server provê serviços EJB (*Enterprise Java Beans*) e *web containers*.
- *Enterprise Java Beans* (EJB): gerencia a execução de *enterprise beans* para aplicações Java EE. *Enterprise bean* e outros *containers* rodam dentro de um Java EE Server.
- Web Container: gerencia a execução de páginas, *servlets* e alguns componentes EJB para aplicações Java EE. Componentes web rodam dentro do Java EE Server.
- Aplicação Cliente: gerencia a execução de componentes de aplicações cliente, que podem ser desenvolvidos em Swing, AWT ou outra tecnologia desktop, porém com acesso a todos os recursos do Java EE.
- Applet: gerencia a execução de Applets que consistem de um *Plug-in* Java e um *browser* rodando juntos no lado cliente.

A figura 2 ilustra o servidor Java EE, seus *containers* e componentes.

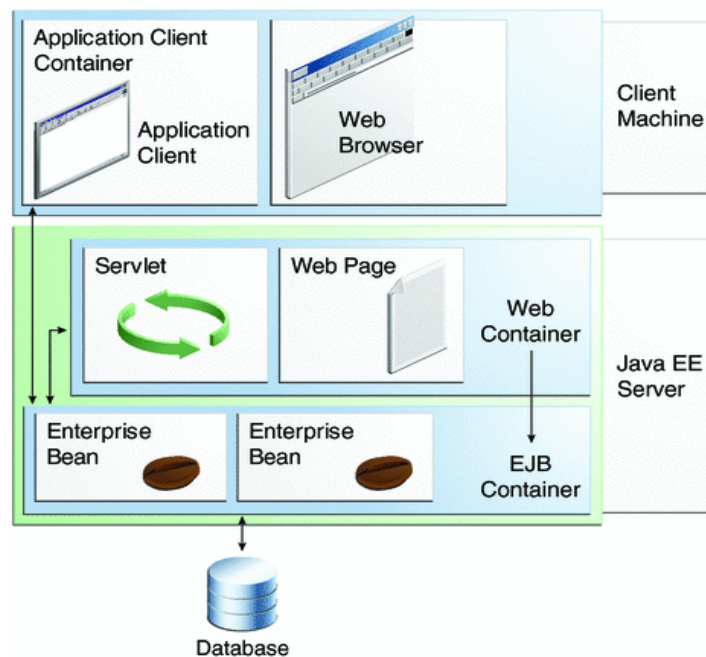


Fig. 2: Java EE Server, Componentes e *Containers*

Fonte: <http://download.oracle.com/javaee/6/tutorial/doc/figures/overview-serverAndContainers.gif>.

Neste trabalho será abordado o *Web Container*, pois é nele que o sistema proposto pelos autores está inserido.

Um *Web Container* tem a responsabilidade de instanciar, inicializar e invocar componentes como páginas JSP e *Servlets*, é ele o responsável pelo gerenciamento do ciclo de vida destes, os quais são abordados nas próximas sessões.

O *Web Container* faz parte de servidores web ou servidores de aplicações, como exemplo pode-se citar o Tomcat ou JBoss, que fornecem serviços de rede através de requisições e respostas que são enviadas aos mesmos.

A seguir será falado sobre duas importantes tecnologias contidas na plataforma Java EE, são elas *Servlets* e JSP (*Java Server Pages*).

3.1.1.1 Servlet

Servlet é uma tecnologia usada na construção de componentes do lado servidor no desenvolvimento de aplicações web em Java e que está incluída na plataforma Java EE. De acordo com a ORACLE (2010) *Servlets* fornecem aos desenvolvedores web um

mecanismo simples e consistente para estender a funcionalidade de um servidor web e para acessar as lógicas de negócios existentes em uma aplicação. Ela explica também que um *Servlet* pode quase ser considerado como um *applet* que é executado no lado do servidor.

Em suma um *Servlet* consiste em uma classe Java que deve ser executada em um *Container Web*. Eles atendem a requisições, as processam e retornam uma resposta, a qual poderá ser em forma de uma página HTML ou de dados quaisquer solicitados. Assim é possível construir páginas cujo conteúdo será montado no momento em que uma requisição HTTP for tratada. A figura 3 ilustra a interação de um *Servlet* com os demais componentes que podem conter em uma solicitação.

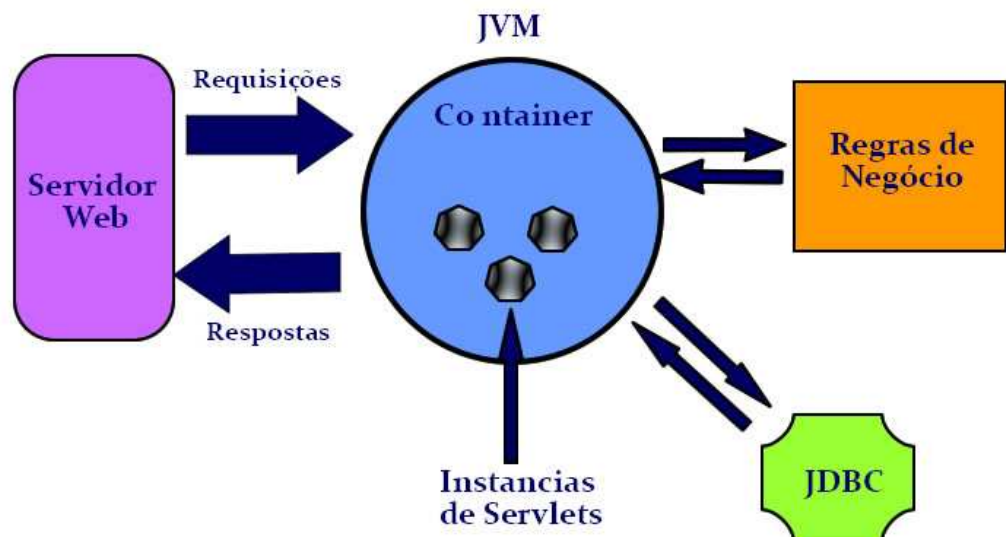


Fig. 3: Interação entre Servlets, Servidor, *Container* e acesso a Regras de Negócio
Fonte: Elaborado pelos autores.

Conforme explica a SUN MICROSYSTEMS (2010d) quando houver a implementação de um serviço genérico, um *Servlet* deverá estender da classe *HttpServlet*, subclasse da *GenericServlet* e implementar ao menos os métodos *doGet()* e *doPost()* para a manipulação de serviços HTTP específicos. A seguir será listado um exemplo de *Servlet*:

```

public class MyServlet extends HttpServlet{

    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        /*implement code*/
    }

    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        /*implement code*/
    }

}

```

Servlets são carregados por servidores Web que suportam Java como Apache Tomcat, JBoss ou GlassFish. Eles são mapeados através de URL's em um arquivo chamado web.xml e quando requisitadas são associadas a um determinado *Servlet*. Abaixo segue um exemplo de mapeamento de um *Servlet*.

```

<servlet>
  <description></description>
  <display-name>MyServlet</display-name>
  <servlet-name>MyServlet</servlet-name>
  <servlet-class>mypackage.MyServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>MyServlet</servlet-name>
  <url-pattern>/myServlet</url-pattern>
</servlet-mapping>

```

No código acima mapeamos através do elemento `<servlet>` a classe `MyServlet` indicando o pacote onde ela se encontra. E utilizando o elemento `<servlet-mapping>` é feita a associação entre o *Servlet* e a URL que corresponderá ao mesmo.

Ao utilizar *Servlet*, se o código HTML for inserido dentro da classe Java, isso pode deixar a lógica do código não muito clara e também dificultar a busca dentro do código, causando dificuldade de manutenção, além de não ser recomendado pelas boas práticas de programação. Tentando solucionar este problema, surge o JSP que será apresentado na próxima sessão.

3.1.1.2 Java Server Pages - JSP

Uma das tecnologias utilizadas na plataforma Java EE é o JSP (Java Server Pages), que se trata de uma forma simples de criar conteúdo dinâmico para web, explica SUN MICROSYSTEMS (2010). Uma página JSP permite que código Java seja intercalado com conteúdo estático de marcação (comumente código HTML ou XML).

Para escrever código Java nas páginas HTML são utilizados os chamados *scriptlets* que consistem em código Java escrito dentro das *tags* `<%` (menor e por cento) e `%>` (por cento e menor). O trecho de código a seguir ilustra o uso de *scriptlets*.

```
<html>
<title>MyJSP</title>
</head>
<body>

    <%
    String message = "My JSP Page!!";
    %>

</body>
</html>
```

Quando uma página JSP é carregada no *container* pela primeira vez e o seu código Java é compilado, um *Servlet* correspondente a esta página é gerado. BATES *et al* (2004) explica que o *container* irá traduzir uma `mypage.jsp` por exemplo, para um arquivo `mypage_jsp.java`. A partir daí ele será compilado para um arquivo `mypage_jsp.class`. O *container* irá carregar o *Servlet* `.class`, instanciá-lo e inicializá-lo, fará uma *thread* separada para cada requisição e chamará o método `service()` do *Servlet*. O processo de compilação de um JSP é ilustrado na Figura 4.

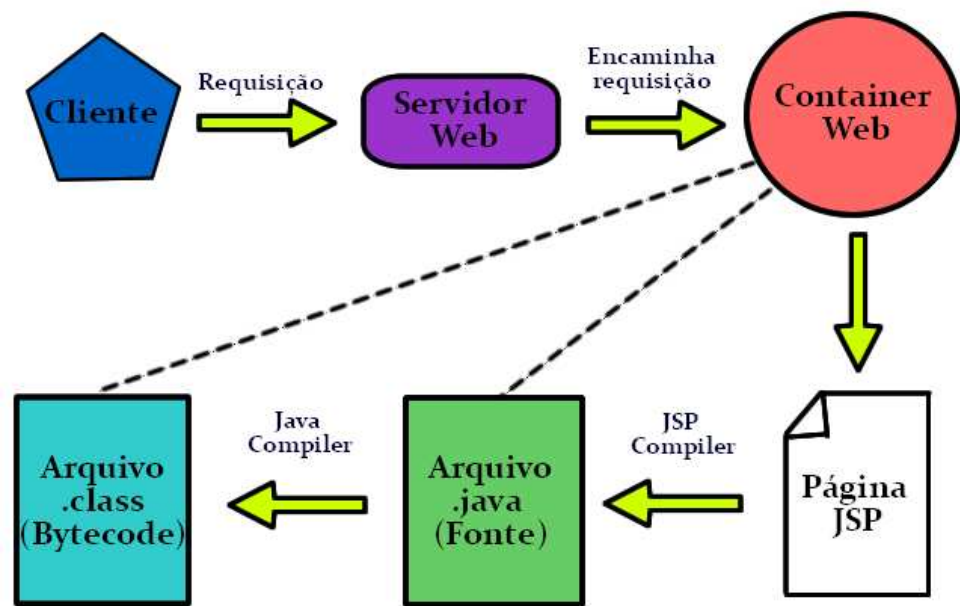


Fig. 4: Processo da primeira execução de uma página JSP
Fonte: Elaborado pelos autores.

O uso de *scriptlet* em páginas JSP é uma solução para a criação de conteúdo dinâmico, porém de acordo com BATES *et al* (2004), o uso de código Java dentro de páginas HTML pré supõe que um web design que cuidará do design da página deverá saber a linguagem Java, o que não é verdade. Ele completa também que o código Java em uma página JSP faz com que a execução de alterações e manutenção se tornem difíceis.

Para que estes problemas fossem amenizados foi criado a *Expression Language*¹⁰ ou Linguagem de Expressão que se tornou parte do JSP a partir da especificação 2.0. BATES *et al* (2004) dizem que o EL quase sempre permite de forma simples fazer coisas que normalmente você apenas faria com *scriptlets*. A proposta do EL é oferecer uma maneira fácil de invocar código Java. Abaixo segue a comparação entre uma expressão EL e uma expressão Java usada em uma página JSP.

```

/* Usando EL */
Nome: ${applicationScope.name}

/* Usando scriptlet */
Nome: <%= applicationContext.getAttribute("name"); %>

```

¹⁰ O termo *Expression Language* será, doravante, designado por EL.

Quando se trabalha com aplicações mais complexas, são necessárias alternativas para simplificar a estrutura do projeto. Alguns frameworks surgiram com a finalidade de suprir essa necessidade, na sessão seguinte será abordado o Spring, o qual foi utilizado no desenvolvimento deste projeto.

3. 2 Spring 3.0

O Spring é um poderoso framework de código aberto escrito em linguagem Java. Ele foi criado por Rod Johnson no ano de 2002 com a finalidade de diminuir a complexidade do desenvolvimento de aplicativos corporativos. “...a utilidade do Spring não é limitada ao desenvolvimento do lado servidor. Qualquer aplicativo em Java pode se beneficiar do Spring em termos de simplicidade, testabilidade e baixo acoplamento” WALLS (2008, p.5).

“O Framework Spring vem com uma série de módulos construídos a partir da base de injeção de dependência ou Inversão de Controle - IoC¹¹ e POA, e cria uma plataforma cheia de recursos, sobre a qual se constroem os aplicativos” WALLS (2008, p.5).

Este framework foi projetado para ser não-intrusivo. Isto significa que seu código de lógica de domínio geralmente não possuirá dependências do framework em si. Em sua camada de integração, como a camada de acesso a dados por exemplo, algumas dependências entre a tecnologia de acesso aos dados e as bibliotecas do Spring existirão, contudo, elas devem ser fáceis de ser isoladas do resto do seu código, explica SPRING SOURCE (2010).

Algumas das vantagens que um desenvolvedor pode obter com o Spring são citadas a seguir, conforme SPRING SOURCE (2010):

- Usar transações de banco de dados em um método sem ter que lidar com API's de transação.
- Transformar um método Java local em um procedimento remoto sem ter que lidar com API's remotas.

¹¹ O termo Inversão de Controle será, doravante, designado IoC.

- Fazer um método Java local se transformar em uma operação de gestão sem ter que lidar com API's JMX¹² - *Java Management Extensions*.
- Transformar um método local Java em um manipulador de mensagem sem ter que lidar com API's JMS¹³ - *Java Message Service*.

A versão do Spring utilizada neste projeto será a 3.x., que é a última lançada até a data do feitiço deste trabalho. Nesta versão o Spring é baseado em Java 5 e com suporte completo ao Java 6, ele também é compatível com Java EE 1.4 e Java EE 5, enquanto ao mesmo tempo é introduzido suporte inicial para o Java EE 6.

A versão 3.x conta com diversos novos recursos dos quais se podem destacar:

- *Spring Expression Language* - SpEL¹⁴: é uma linguagem de expressão semelhante a EL Unificada em sua sintaxe, porém com alguns recursos adicionais. Ela pode ser usada em definições de XML e em anotações. Ela também serve como base para suporte de EL em todos os produtos do portfólio Spring.
- Melhorias no IoC: alguns recursos do projeto JavaConfig¹⁵ foram adicionados ao Spring Framework, trazendo assim, suporte para as seguintes anotações: *@Configuration*, *@Bean*, *@DependsOn*, *@Primary*, *@Lazy*, *@Import*, *@ImportResource*, *@Value*.
- Sistema de conversão de tipo (*Converter SPI*) e sistema de formatação de campo (*Formatter SPI*): o *Converter SPI* é usado pelo SpEL para conversão de tipos Java e pode ser usado pelo *container* Spring. O *Formatter SPI* foi criado para formatar valores de campo, fornecendo uma alternativa mais simples e mais robusta para *PropertyEditors JavaBean* para uso em ambientes cliente, tais como Spring MVC.
- Módulo *Object/XML Mapping*¹⁶ foi incorporado ao Spring: a funcionalidade de mapeamento de objeto para XML - OXM do projeto *Spring Web Services* foi incorporado ao Spring Framework.

¹² JMX é uma API que permite o monitoramento e o gerenciamento de dispositivos, aplicações e serviços de rede.

¹³ JMS é uma API padrão para envio de mensagens que permite que componentes baseados na plataforma Java EE envie, receba e leia mensagens.

¹⁴ O termo *Spring Expression Language* será, doravante, designado SpEL.

¹⁵ JavaConfig (Spring Java Configuration Project) é um projeto da SpringSource para acesso de propriedades de configuração.

¹⁶ *Object/XML Mapping* Mapeamento Objeto/XML

- Suporte ao REST: suporte *server-side*¹⁷ e *client-side*¹⁸ para funcionalidades REST fazendo uso de *HttpConverters* para facilitar a comunicação entre objetos e suas representações em requisições e respostas HTTP.
- Adição do @MVC: o *namespace* MVC foi adicionado para simplificar as configurações do Spring MVC e novas anotações também foram adicionadas como *@CookieValue* e *@RequestHeaders*.
- Validação de modelo declarativo: várias melhorias de validação incluindo suporte *Java Specification Requests - JSR*¹⁹ 303 que utiliza o Hibernate Validator como o provedor padrão.
- Suporte inicial ao Java 6: chamadas de métodos assíncronos através do uso da anotação *@Async* (ou EJB 3.1, anotação *@Asynchronous*) JSR 303, JSF 2.0, JPA 2.0, etc.
- Suporte para bancos de dados incorporados: suporte embutido para mecanismos de banco de dados Java, incluindo HSQL, H2 e Derby.

O Spring possui inúmeros recursos, porém, ele se destaca como sendo um leve *framework* e um *container* orientado a aspecto e com injeção de dependência. A seguir são mais bem detalhadas as características deste *framework*.

WALLS (2008) destaca que o Spring possui um *container* leve em termos de tamanho e sobrecarga, enfatizando que a sobrecarga de processamento exigida pelo *framework* pode ser considerada desprezível.

Outra importante característica que deve ser citada é a promoção de baixo acoplamento através da técnica denominada injeção de dependência. O rico suporte à POA também deve ser destacado. Isto possibilita um desenvolvimento coeso, separando a lógica de negócio dos serviços de sistema.

A sessão seguinte tratará da técnica de injeção de dependência.

¹⁷ *Server-side* ou lado do servidor é o termo designado a operações executados no servidor em uma arquitetura cliente-servidor.

¹⁸ *Client-side* ou lado do cliente é o termo designado a operações executadas no computador do usuário e não em um servidor.

¹⁹ *Java Specification Requests* são documentações formais que descrevem especificações e tecnologias adicionadas na plataforma Java. Este termo será, doravante, designado JSR.

3.2.1 Injeção de dependência

Para promover o baixo acoplamento o Spring utiliza a técnica conhecida como Injeção de Dependência - DI²⁰, que será explicada a seguir.

Todo aplicativo não-trivial (qualquer coisa mais complexa que HelloWorld.java) é composto de duas ou mais classes que colaboram uma com a outra para executar alguma lógica de negócios. Tradicionalmente cada objeto é responsável por obter suas próprias referências para os objetos com os quais colabora (suas dependências). Isso pode levar a códigos com alto acoplamento e difícil de testar (WALLS, 2008, p.12).

Utilizando injeção de dependência, os objetos obtêm suas dependências em tempo de criação, ou seja, suas dependências são injetadas por alguma entidade externa no momento em que os objetos são criados.

De acordo com WALLS (2008), inicialmente a injeção de dependência era referenciada por inversão de controle – IoC. Porém, em um artigo, Martin Fowler questionou em qual aspecto o controle era invertido, chegou-se então a conclusão que o que invertia era a responsabilidade pela aquisição da dependência. A partir daí, Fowler propôs que fosse utilizado o termo injeção de dependência, por acreditar que este melhor explicasse o princípio de DI.

Antes de explicar como funciona a DI no Spring é importante explicar o conceito de um termo que comumente aparecerá neste trabalho: a definição de um *bean*. A SPRING SOURCE (2010) afirma que os objetos que formam a “espinha dorsal” de uma aplicação e que são gerenciados pelo *container* IoC do Spring são chamados *beans*. Um *bean* é um objeto instanciado, montado e gerenciado pelo *container* IoC do Spring.

Beans e suas dependências são definidos em configurações de metadados usados pelo container, eles podem ser representados via XML, anotações ou mesmo código Java nativo. Estas configurações permitem declarar os objetos que compõem um aplicativo e todas as dependências existentes entre eles.

A interface `org.springframework.context.ApplicationContext` representa o *container* IoC do Spring que é o responsável por instanciar e configurar os *beans*. A figura 5 ilustra como trabalha o Spring na inicialização do contexto da aplicação, no

²⁰ O termo Injeção de Dependência será, doravante, designado DI.

qual são combinadas as classes com os metadados configurados e, então, o *ApplicationContext* é criado e inicializado.

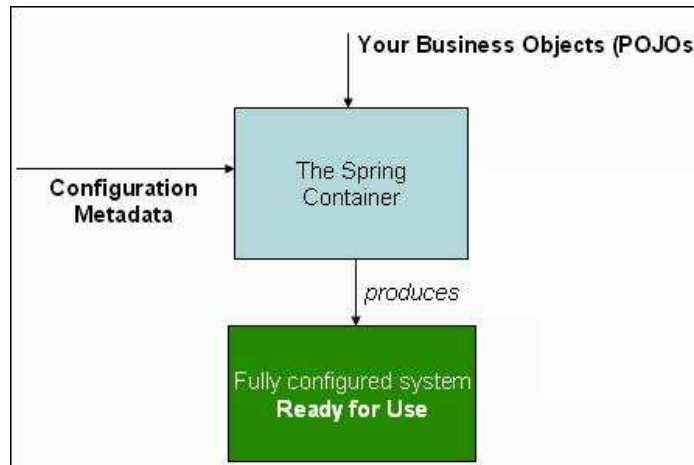


Fig. 5: Funcionamento do Container IoC do Spring

Fonte: <http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/images/container-magic.png>

Como já mencionado, a associação entre os *beans* pode ser feita através de XML ou anotações, podendo também utilizar ambos em conjunto, neste caso, associações feitas com anotações são realizadas antes de associações feitas com XML. Desta forma, configurações feitas com anotações podem ser subscritas por configurações provenientes de XML.

Neste trabalho a associação de *beans* foi feita, em sua maioria, através de anotações. Para que se possa habilitar a configuração dos *beans* por anotações, será utilizado no arquivo de configuração do Spring o elemento `<context:annotation-config/>`. A listagem a seguir ilustra o uso deste elemento.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-
3.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-
3.0.xsd">

  <context:annotation-config/>

</beans>
```

Este elemento irá procurar por anotações nos *beans* presentes dentro do contexto de aplicação. A seguir será apresentado um pequeno exemplo utilizando a DI no Spring. A listagem seguinte ilustra a classe *Message*, que possui alguns atributos pré-definidos no código e seus respectivos métodos de acesso. A anotação *@Component* é um estereótipo genérico usado para mapear qualquer componente gerenciado pelo Spring.

```
@Component
public class Message {

    @Value("Somebody")
    private String author;

    @Value("It's DI with Spring's annotation")
    private String message;

    public String getAuthor() {
        return author;
    }
    public void setAuthor(String author) {
        this.author = author;
    }
    public String getMessage() {
        return message;
    }
    public void setMessage(String message) {
        this.message = message;
    }
}
```

Prosseguindo com o exemplo, foi criada uma *interface* que contém a assinatura do método *writeMessage*, utilizado para escrever uma mensagem qualquer.

```
public interface Messenger {
    public void writeMessage(Message message);
}
```

A classe *MessengerImpl* implementa a *interface Messenger* criada anteriormente. A anotação *@Service* é uma especialização da anotação *@Component*, ela é usada especificamente para indicar um serviço.

```
@Service("messeger")
public class MessengerImpl implements Messenger {

    @Override
    public void writeMessage(Message message) {
        System.out.println("Author: " + message.getAuthor());
        System.out.println("Message: " + message.getMessage());
    }
}
```

A próxima classe é a *WelcomeController*, nesta classe de controle existem dois atributos: uma mensagem e um mensageiro. Estes atributos são anotados com o *@Autowired*, isto fará com que o Spring injete uma instância da classe correspondente no atributo. Pode-se notar então que os atributos não são instanciados, eles são usados diretamente.

```
@Controller
public class WelcomeController {

    @Autowired
    private Message message;

    @Autowired
    private Messenger messenger;

    @RequestMapping("/showMessage")
    public void showMessage(){
        messenger.sendMessage(message);
    }
}
```

A execução do método *showMessage* da classe anterior irá imprimir no console a mensagem apresentada na listagem abaixo.

```
Author: Somebody
Message: It's DI with Spring's annotation
```

No contexto de DI, além das anotações mostradas no exemplo apresentado, existem outras que são importantes serem abordadas como *@Inject* que possui a mesma função da anotação *@Autowired*, para que esta anotação seja usada é preciso adicionar ao *classpath* da aplicação o JAR do JRS-303. Porém, com a anotação *@Inject* não é possível usar o atributo *required* que indica se a injeção de algum parâmetro é obrigatório ou não, o valor *default* deste atributo é *true*. Abaixo segue um exemplo do uso da anotação *@Autowired* com o atributo *required* utilizado em um método *setter*.

```
public class SimpleMovieLister {

    private MovieFinder movieFinder;

    @Autowired(required=false)
    public void setMovieFinder(MovieFinder movieFinder) {
        this.movieFinder = movieFinder;
    }
}
```

O Spring também possui uma anotação *@Required*, ela obriga que a injeção de dependência seja feita em tempo de configuração, evitando assim a possibilidade de um

NullPointerException. Se no momento de configuração não for encontrado um *bean* que satisfaça a dependência, uma exceção é lançada.

A próxima sessão abordará os módulos contidos no Spring.

3.2.2 Módulos

Os recursos do Spring estão distribuídos em cerca de vinte módulos organizados em cinco grupos, são eles, Core Container, Integração/Acesso a Dados, Web, Programação Orientada a Aspecto - POA²¹, Instrumentação e Teste como mostrado na figura 6.

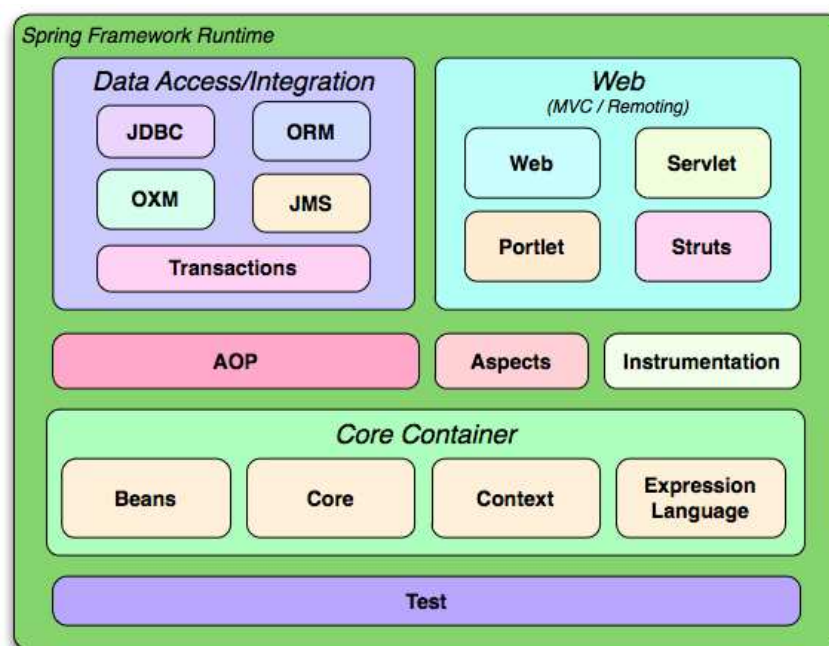


Fig. 6: Estrutura dos módulos do Spring Framework.

Fonte: <http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/overview.html>

O Spring é um *framework* modular, permitindo que você use somente aquelas partes que você realmente necessita, sem ter de utilizar o restante explica SPRING SOURCE (2010), ele completa que você pode usar o *container* IoC com o Struts por exemplo. Mas você também pode usar somente o código de integração do Hibernate ou a camada de abstração do JDBC.

²¹ O termo Programação Orientada a Aspecto será, doravante, designado por POA.

3.2.3 Spring MVC

O Spring MVC é o *framework* web nativo do Spring que se baseia no padrão *Model-View-Controller* (MVC²²).

Conforme explica (Spring) a documentação do *framework*, o módulo *web* do Spring contém inúmeros recursos, tais como:

- Separação clara de papéis.
- Adaptabilidade e flexibilidade.
- Código de negócio reutilizável sem a necessidade de duplicação.
- Customização de mapeamento de manipuladores e de resolução de visualização.
- Transferência flexível do *Model*.
- Uma biblioteca de *tags* JSP, introduzida no Spring 2.0, que torna a escrita de formulários em páginas JSP muito mais fácil.

Como a maioria dos *frameworks* MVC em Java, o Spring MVC é projetado em torno de um *DispatcherServlet*. Walls(2008) explica que o *DispatcherServlet* tem o papel de um *front controller*, que consiste em um modelo de aplicativo comum onde todas as requisições são passadas para um único *servlet* e este é responsável por delegar a requisição a *servlets* ou componentes de um outro aplicativo.

De acordo com a documentação de referência do Spring o *DispatcherServlet* recebe todas as requisições e as despacha para manipuladores que geralmente são classes baseadas nas anotações *@Controller* e *@RequestMapping* que oferecem uma grande gama de flexíveis métodos que são responsáveis pelo tratamento da requisição.

Para que seja possível entender o funcionamento deste *framework* vamos falar do caminho que uma requisição percorre desde o momento em que ela deixa o navegador até o momento em que ela retorna com uma resposta ao usuário conforme representado na figura 7.

²² O termo *Model-View-Controller* será, doravante, designado por MVC.

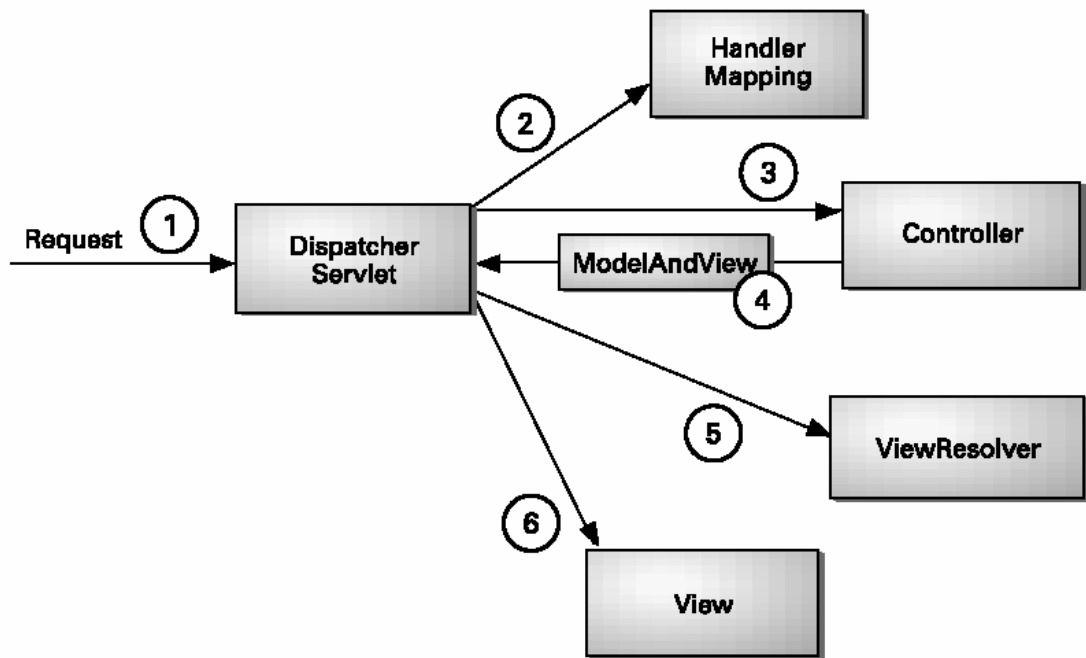


Fig. 7: O ciclo de vida de uma requisição no Spring MVC

Fonte: <http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/mvc.html>

Todas as requisições devem passar pelo *DispatcherServlet*, este, por sua vez determinará através da URL da requisição o *controller* apropriado para tal solicitação. Após escolhido o *controller*, o mesmo irá tratar a requisição e fornecer os dados que serão mostrados ao usuário. Estes dados são chamados de modelo (*model*) e serão enviados para uma *view*, que é a responsável por exibir de forma amigável as informações ao usuário. O *controller* utilizará um objeto *ModelAndView* para enviar ao *DispatcherServlet* os dados do *model* e o nome da *view*. Por fim, os dados do *model* serão entregues a *view* correspondente que geralmente será uma JSP (Java Servers Pages). Assim termina o ciclo de vida de uma requisição no Spring.

3.3 Java Persistence API

Na camada de persistência da aplicação foi utilizado o Java Persistence API - JPA²³, que é a especificação Java para o gerenciamento da camada de persistência e

²³ O termo *Java Persistence API* será, doravante, designado por JPA.

mapeamento objeto/relacional, facilitando desta forma este tipo de mapeamento em aplicações Java. Neste projeto esta API terá seu funcionamento integrado ao Spring.

De acordo com ORACLE (2006) o JPA lida com a forma como os dados relacionais são mapeados para objetos Java e como esses objetos são armazenados em um banco de dados relacional, para que estes possam ser posteriormente acessados. Lida também com a continuação da existência do estado de uma entidade mesmo após o término de um aplicativo. Além de simplificar o modelo de persistência de uma entidade o JPA padroniza o mapeamento objeto-relacional. A ORACLE (2006) ressalta também que O JPA é baseado nas idéias dos principais frameworks de persistência e API's como Hibernate, TopLink e o Java Data Objects - JDO²⁴.

Esta especificação consiste de quatro áreas segundo a SUN (2010):

- A API Java de Persistência.
- A linguagem *query*.
- A API Critéria Java de Persistência.
- Os metadados para mapeamentos objeto/relacional.

O JPA está contido no pacote *javax.persistece* e utiliza como linguagem para suas sentenças a *Java Persistence query language - JPQL*²⁵. Neste trabalho foi utilizada sua versão 2.0 que é descrita pela JSR 317. Entre as propostas desta versão estão: a expansão do mapeamento objeto/relacional; alguns adicionais para o JPQL; contrato adicional para entidades e extensão dos contextos de persistência; suporte para validação com funcionamento do JSR 303²⁶.

A figura 8 mostra como é feito o acesso aos dados utilizando o JPA. Pode-se notar que é necessário escolher um provedor que irá implementar a especificação JPA, como Hibernate ou TopLink por exemplo. O provedor irá lidar com o *Java Database Connection -JDBC*²⁷ que através de um *driver* irá acessar um determinado Sistema Gerenciador de Base de Dados - SGBD²⁸. Como exemplo de SGBD's, pode-se citar o PostgreSQL, MySQL, DB2, etc.

²⁴ O termo *Java Data Objects* será, doravante, designado por JDO.

²⁵ O termo *Java Persistence query language* será, doravante, designado por JPQL.

²⁶ JSR 303 define a API para o *Java Bean Validation* baseado em anotações.

²⁷ O termo *Java Database Connection* será, doravante, designado por JDBC.

²⁸ O termo determinado Sistema Gerenciador de Base de Dados será, doravante, designado por SGBD.

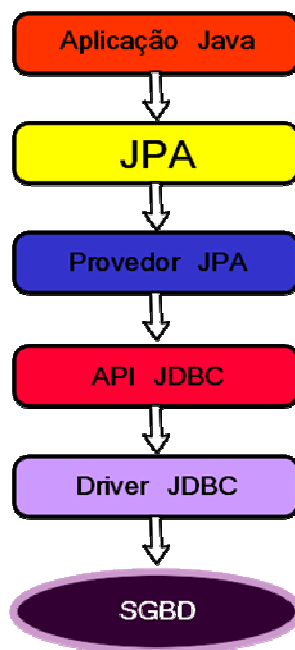


Fig. 8: Aplicação utilizando JPA
Fonte: Elaborado pelos autores.

A sessão seguinte tratará da implementação de referência do JPA 2.0: o EclipseLink.

3.3.1 EclipseLink

O *Eclipse Persistence Services Project*, mais comumente conhecido como EclipseLink, iniciou-se com a doação total do código fonte do produto TopLink da empresa Oracle. Este projeto trouxe a experiência de doze anos de utilização comercial e de desenvolvimento de recursos por toda a comunidade Java. Com essa doação o projeto evoluiu para um *software* livre, facilitando assim, sua distribuição e seu acesso aos desenvolvedores.

O EclipseLink é a implementação de referência do JPA 2.0. De acordo com ECLIPSE (2010) o EclipseLink pode ser definido como uma solução em código aberto para a camada de persistência Java focada em padrões com avançados recursos, performance e escalabilidade para *softwares* corporativos.

4 METODOLOGIA DE DESENVOLVIMENTO

Uma metodologia se consiste em um roteiro, um conjunto de passos pré-estabelecidos para se atingir um determinado objetivo. Trata-se de um processo dinâmico e estruturado para auxiliar o desenvolvimento de projetos, focando na qualidade, eficiência e produtividade.

A metodologia de desenvolvimento que foi adotada para este trabalho é o ICONIX. ROSENBERG (2005) a define como sendo um guia desde os casos de uso até a codificação e também afirma que sua missão é tirar a ambiguidade dos requisitos e propiciar a implementação de um projeto limpo.

“O ICONIX é um processo simplificado que unifica conjuntos de métodos de orientação a objetos em uma abordagem completa, com o objetivo de dar cobertura ao ciclo de vida” (BONA e COSTA, 2003, p.221). Seu processo é iterativo, incremental e utiliza técnicas de *Unified Modeling Language* – UML²⁹.

De acordo com GUEDES (2006), a UML é uma linguagem visual utilizada para a modelagem de sistemas seguindo o paradigma de orientação a objetos. Ela se tornou um padrão internacional para a modelagem de *software*.

A UML tem como objetivo auxiliar engenheiros de *software* a definir os requisitos do sistema, suas características, seu comportamento, sua estrutura lógica, a dinâmica de seus processos e até mesmo requisitos de hardware para a implantação do sistema, explica GUEDES (2006). Algumas das técnicas de UML utilizadas pelo ICONIX são: casos de uso, diagrama de classes, análise de robustez, diagrama de sequência e de colaboração.

O ICONIX é considerado uma metodologia de desenvolvimento prática e classificada como intermediária em relação à complexidade do *Rational Unified Process* - RUP³⁰ e à simplicidade do *Extreme Programming* – XP³¹.

O RUP é um processo proprietário que foi criado pela *Rational Software Corporation*, posteriormente adquirida pela IBM. Ele é considerado um processo pesado e, preferencialmente, deve ser aplicado a grandes equipes de desenvolvimento.

²⁹ O termo *Unified Modeling Language* também pode ser tratado por sua tradução para o português: Linguagem de Modelagem Unificada. Este termo será, doravante, designado por UML.

³⁰ O termo *Rational Unified Process* será, doravante, designado por RUP.

³¹ O termo *Extreme Programming* será, doravante, designado por XP.

Porém, conforme explica IBM (2010), o RUP é ajustável e permite implementar apenas os componentes de processo que são necessários para cada estágio do projeto. Isso torna possível que ele seja adaptável a projetos de qualquer escala.

O XP é uma metodologia ágil, adotada para equipes pequenas e médias. Ele prioriza a simplicidade, prioriza o funcionamento do software em relação à documentação. Com essa metodologia o software é desenvolvido com requisitos vagos, o que torna necessário a realização de constantes ajustes durante o desenvolvimento.

A adoção do ICONIX como metodologia para o desenvolvimento deste projeto se deve a sua abordagem completa em cada etapa do projeto, acelerando o processo de implementação do *software* e tendo uma visão documentada de todos os requisitos do SAS.

4.1 Origem do ICONIX e questões fundamentais

O ICONIX foi criado por Doug Rosenberg e Kendall Scott na década de 90, a partir da síntese do processo unificado, uma teoria dos “*three amigos*”, expressão pela qual Grady Booch, Jim Rumbaugh e Ivar Jacobson são conhecidos, explicam ROSENBERG e SCOTT (1999).

Na figura 9 são apresentados os elementos chaves da metodologia ICONIX.

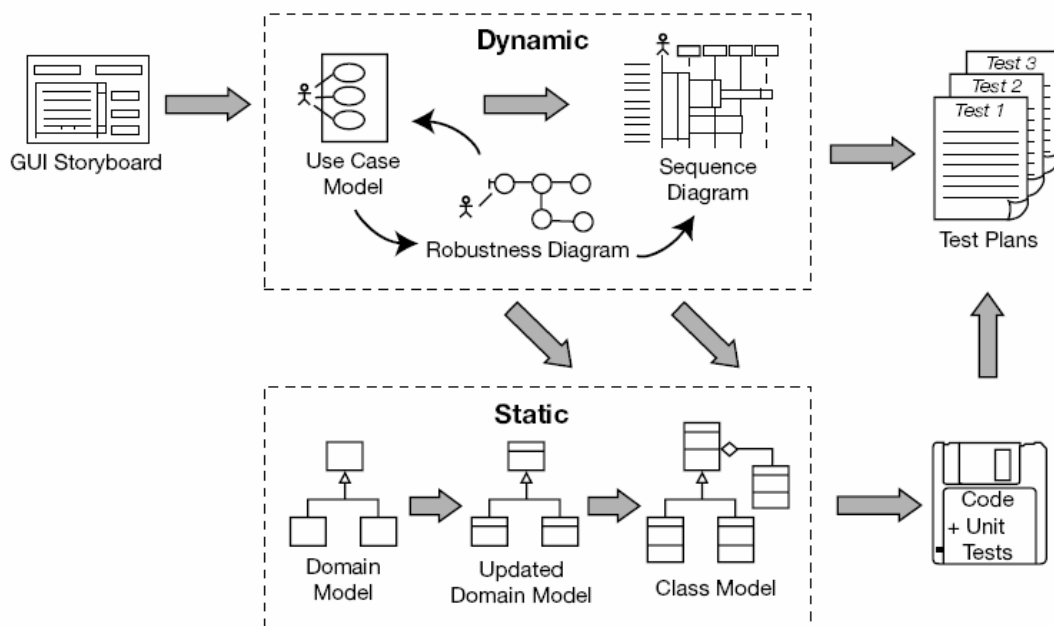


Fig. 9: Elementos Chaves da Abordagem do ICONIX

Fonte: ROSENBERG e SCOTT. Livro *Use Case Driven Object Modeling with UML*

De acordo com ROSENBERG e SCOTT (1999), o ICONIX tem o papel de responder a algumas questões fundamentais sobre o software através de técnicas de UML, as questões e as técnicas de UML utilizadas para auxiliar a encontrar as melhores respostas são as seguintes:

- Quem são os usuários do sistema, e o que eles estão tentando fazer? A técnica de UML utilizada para ajudar a obter essas respostas são os casos de uso;
- O que são, no "mundo real" (chamado domínio de problema), os objetos e associações entre eles? Utiliza-se diagrama de classe de alto nível;
- Quais objetos são necessários para cada caso de uso? A análise de robustez é a técnica utilizada para responder essa pergunta;
- Como está a interação entre os objetos dentro de cada caso de uso? Utiliza-se diagrama de seqüência e de colaboração;
- Como será manipulado em tempo-real aspectos de controle? Utiliza-se diagrama de estado;
- Como realmente será construído o sistema em um nível prático? Utiliza-se diagrama de classe de baixo nível.

4.2 Fases e marcos do ICONIX

Esta sessão apresentará as várias fases do desenvolvimento utilizando o ICONIX, são elas: análise de requisitos, análise e projeto preliminar, projeto e, por fim, implementação. A fase de análise de requisitos será abordada a seguir.

4.2.1 Análise de requisitos

A primeira atividade da análise de requisitos compreende a identificação dos objetos do “mundo real” e as relações de agregação e generalização entre eles. Neste caso utiliza-se um diagrama de alto nível chamado modelo de domínio. Após isso, são feitos esboços das telas do sistema, ou diagramas de navegação, para melhor compreensão do programa.

Em sequência, são identificados os casos de uso, mostrando os atores envolvidos e uma idéia geral de como o sistema irá se comportar. Este diagrama apresenta uma linguagem simples e de fácil compreensão, e serve de base para a elaboração de outros diagramas. Segundo GUEDES (2006) os atores podem ser usuários, outros sistemas ou até mesmo algum *hardware* especial que utilização o programa de alguma maneira.

Por fim, é necessário associar requisitos funcionais aos casos de uso e aos objetos de domínio. É quando acontece o primeiro marco do ICONIX: a revisão dos requisitos.

4.2.2 Análise e projeto preliminar

Na fase de análise e projeto preliminar, a primeira tarefa a ser cumprida é a descrição dos casos de uso através de fluxo de eventos, que contem o fluxo principal de ações de cada caso de uso e pode conter fluxo alternativo e fluxo de exceção.

A segunda tarefa é a apresentação da análise de robustez, explica BONA (2002). Para cada caso de uso identificam-se os objetos relacionados e o fluxo de comunicação entre os mesmos. Terminada a análise de robustez, o diagrama de classes do modelo de domínio deve ser atualizado. Nesta etapa acontece o segundo marco, chamado de revisão do projeto preliminar.

4.2.3 Projeto

Na fase de projeto é necessário especificar o comportamento de cada caso de uso através de diagramas de sequência, identificando as mensagens entre os diferentes objetos envolvidos. De acordo com BONA (2002), pode-se também usar diagrama de colaboração para identificar as transações mais importantes e diagramas de estado para complementar, mostrando o comportamento em tempo real.

Logo após, é necessário terminar o modelo estático, adicionando detalhes do projeto no diagrama de classes e, então, verificar se o projeto satisfaz todos os requisitos

identificados. Neste ponto há o marco conhecido como revisão detalhada e crítica do projeto.

4.2.4 Implementação

Segundo BONA (2002), dentre as atividades que dão apoio à etapa de implementação do projeto estão: a utilização de diagrama de componente para apoiar o desenvolvimento, se for necessário; a realização de testes de unidade e de integração, e; a realização de testes de aceitação pelo usuário. Nesta etapa acontece o último marco do ICONIX que se trata da entrega da aplicação.

4.3 Técnicas usadas no ICONIX

Esta sessão abordará um pouco mais sobre as técnicas de UML utilizadas em cada fase da modelagem com o ICONIX. O primeiro tema a ser tratado será o modelo de domínio.

4.3.1 Modelo de domínio

O modelo de domínio se consiste em um diagrama de classes com um enfoque específico, simplificado, para uma melhor visão das classes que servirão de base para o projeto. De acordo com GUEDES (2007), o diagrama de classes é o mais importante e o mais usado da UML, permite a visualização das classes que irão compor o *software* com seus respectivos métodos e atributos.

ROSENBERG e STEPHENS (2007) explicam que, na prática, o modelo de domínio se consiste em um diagrama de classes simplificado, que possui linhas desenhadas entre cada classe para mostrar como elas se relacionam. O modelo de domínio exhibe relações de agregação e de generalização existentes entre as classes de domínio.

A figura 10 mostra um exemplo deste diagrama.

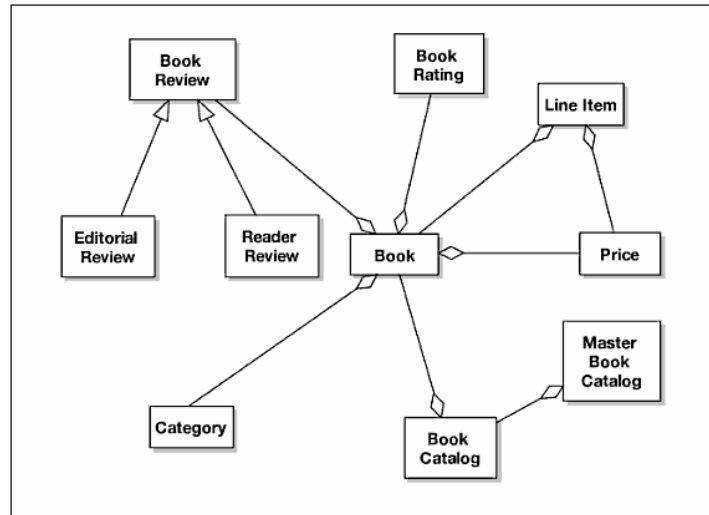


Fig. 10: Exemplo de Modelo de Domínio

Fonte: ROSENBERG e SCOTT. Livro Use Case Driven Object Modeling with UML

O modelo de domínio é o fundamento para a parte estática da modelagem, enquanto que os casos de uso são o fundamento para a parte dinâmica. A parte estática define a estrutura enquanto que a parte dinâmica define o comportamento do sistema.

4.3.2 Casos de uso

O diagrama de casos de uso é o diagrama mais informal da UML, possui uma abordagem mais geral do sistema. Ele é usado normalmente na fase de análise de requisitos, mas é consultado durante todo o processo de modelagem e serve de base para outros diagramas, explica GUEDES (2007).

O objetivo dos casos de uso é oferecer uma visão externa geral das funcionalidades que o *software* deve possuir, não se preocupando com a maneira como essas funcionalidades devem ser implementadas.

Segundo GUEDES (2007), este diagrama ajuda a especificar, visualizar e documentar as características e funções desejadas pelo usuário. É um artefato interessante para se mostrar aos clientes em reuniões iniciais de um projeto, pois o mesmo é de fácil compreensão ao usuário, e assim, eles podem identificar falhas na especificação do sistema.

Posteriormente, os casos de uso devem ser detalhados, através da descrição dos fluxos de eventos. Este documento deve conter, em linhas gerais, quais os atores interagem com o sistema, quais etapas devem ser executada pelo ator e quais devem ser executadas pelo sistema para que o caso de uso cumpra seu objetivo. Segundo os padrões de UML, não existe um formato específico a ser seguido, mas é recomendado o uso de uma linguagem simples que até mesmo leigos possam entender.

4.3.3 Diagrama de sequência

Este diagrama tem como base o diagrama de caso de uso e normalmente cada diagrama de sequência se refere a um caso de uso específico. Isso se deve ao fato de que cada caso de uso se relaciona a um processo realizado pelo usuário, sendo que o diagrama de sequência tem por finalidade determinar a sequência de eventos que ocorrem em um determinado processo, explica GUEDES (2006).

O diagrama de sequência deve apresentar os objetos envolvidos e quais métodos eles invocam, esses eventos também devem aparecer na ordem em que a interação acontece.

4.3.4 Análise de robustez

A análise de robustez é baseada no diagrama de atividades e no diagrama de classes, tem por objetivo identificar os objetos que estão envolvidos e o fluxo de suas interações durante a execução de um determinado caso de uso.

Este diagrama tem algumas informações em comum com o diagrama de sequência e juntos se complementam. Ambos são considerados diagramas de interação. A análise de robustez se encontra em um passo intermediário entre a fase de análise e a fase de projeto no ICONIX, conforme mostra figura 11.

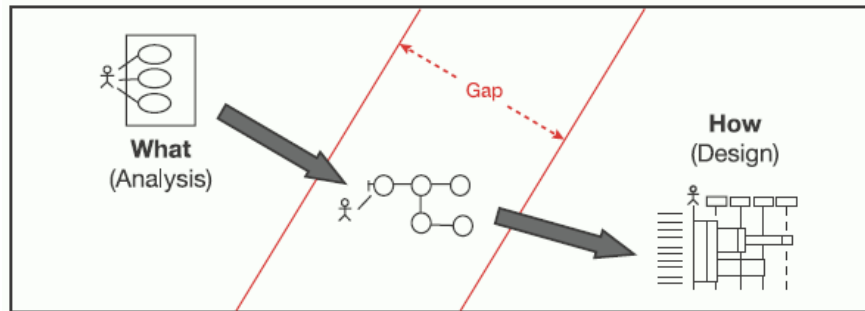


Fig. 11: Análise de Robustez

Fonte: <http://iconixprocess.files.wordpress.com/2007/01/bridgingthechasm.png>

Segundo ICONIXPROCESS (2007), com a elaboração desse diagrama se deve começar a pensar na arquitetura técnica e nas várias estratégias de projeto possíveis.

5 DESENVOLVIMENTO DO PROJETO UTILIZANDO ICONIX

Neste capítulo são abordadas as várias fases do desenvolvimento do projeto usando a metodologia ICONIX. As etapas de cada fase são descritas e exemplificadas com parte da documentação do sistema, que poderá ser encontrada completa nos apêndices do trabalho. A primeira fase do desenvolvimento foi a análise de requisitos que será apresentada na primeira sessão deste capítulo.

5.1 Análise de requisitos

De acordo com BONA e COSTA (2003), os requisitos e os casos de uso são conceitos distintos no ICONIX. ROSENBERG *et al* (2005) apresentam quatro pontos que permitem distinguir esses conceitos:

- Um caso de uso descreve uma unidade de comportamento;
- Um requisito descreve uma regra que dita o comportamento;
- Um caso de uso pode satisfazer um ou mais requisitos;
- Um requisito funcional por ser satisfeito por um ou mais casos de uso.

Sempre que abordado algum requisito desta etapa do processo, utiliza-se o termo requisito funcional, pois num sistema há outros tipos de requisitos como, por exemplo, desempenho e usabilidade. Mas na etapa de análise de requisitos apenas requisitos funcionais são analisados.

Desta forma, foi elaborada uma lista de requisitos funcionais do SAS. A tabela 1 apresenta esta lista.

RF-001 O sistema deverá permitir o cadastros de questões por disciplina em seu curso correspondente.	RF-007 Um simulado/avaliação poderá conter várias questões de disciplinas diversas.
RF-002 Cada questão será composta de cinco alternativas e apenas uma será correta.	RF-008 Para cada simulado/avaliação será gerado um gabarito.
RF-003 O sistema permitirá o cadastro de professores e alunos.	RF-009 O sistema deverá gerar relatórios de rendimento por questão.
RF-004 O sistemas permitirá o cadastro de cursos e suas respectivas disciplinas	RF-010 O sistema deverá gerar relatórios de desempenho por aluno.
RF-005 Cada questão será classificada em um nível de dificuldade. Podendo ser fácil, média ou avançada.	RF-011 O sistema deverá gerar relatórios de desempenho por simulados.
RF-006 O sistema deverá permitir a geração de simulados/avaliações impressos ou <i>online</i> a partir das questões cadastradas no banco.	RF-012 O sistema deverá gerar relatórios comparativos de desempenho do aluno em relação à disciplina.

Tabela 1: Lista de requisitos funcionais.

Fonte: Elaborado pelos autores.

Na lista foram identificados 12 requisitos funcionais, que são os norteadores para o desenvolvimento do sistema. O objetivo do *software* desenvolvido foi atender a aos requisitos levantados por esta lista. A seguir são apresentadas as etapas cumpridas na análise de requisitos.

5.1.1 Prototipação

Esta etapa do desenvolvimento permite que se tenha uma idéia visual do sistema antes de sua implementação. Assim, através de *storyboards*, os primeiros esboços das principais telas da interface gráfica foram projetadas, permitindo uma análise de usabilidade e detalhes da implementação de suas funcionalidades.

As primeiras funcionalidades que a equipe desenhou foram as telas de cadastro de professores e a tela de realização de simulado. A seguir, a figura 12 ilustra a tela de cadastro de professores no sistema.

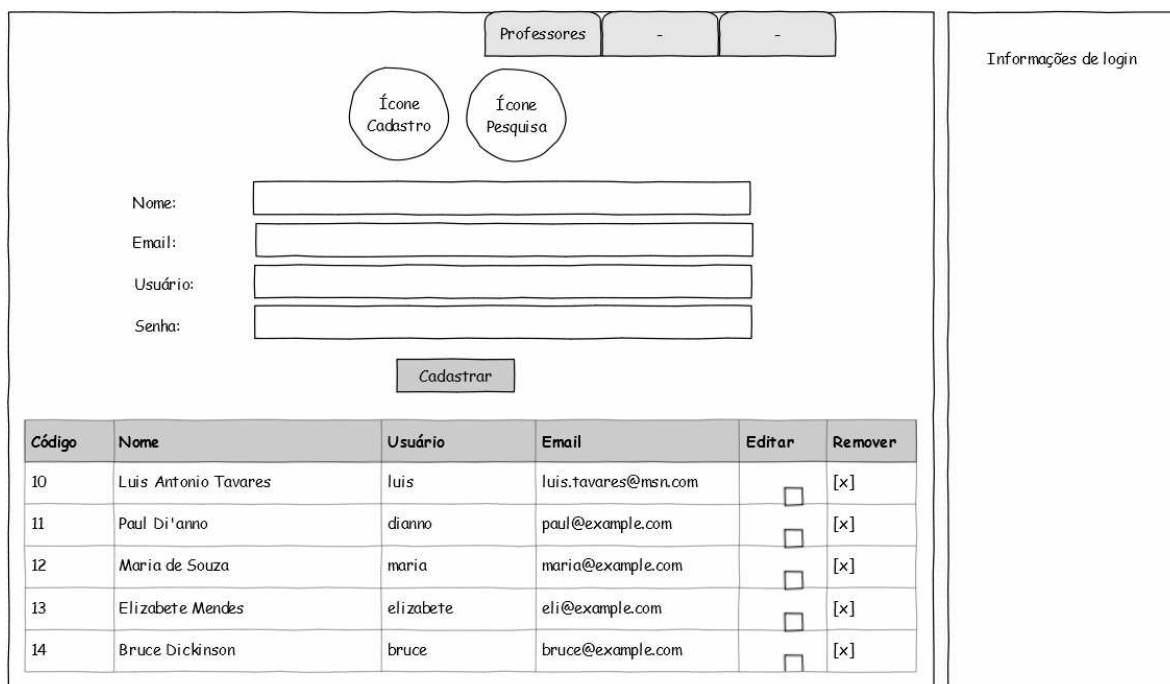


Fig. 12: Tela de Cadastro de Professor
Fonte: Elaborado pelos autores.

No esboço exibido na figura 12 percebe-se que a tela de cadastro de professor prevê funcionalidades de inserção, edição, remoção e pesquisa de professores. Desta forma, nota-se que os *storyboards* são importantes meios para se identificar todas as funcionalidades que são necessárias a cada tela do sistema. A figura 13 ilustra a tela de realização de simulados e avaliações.

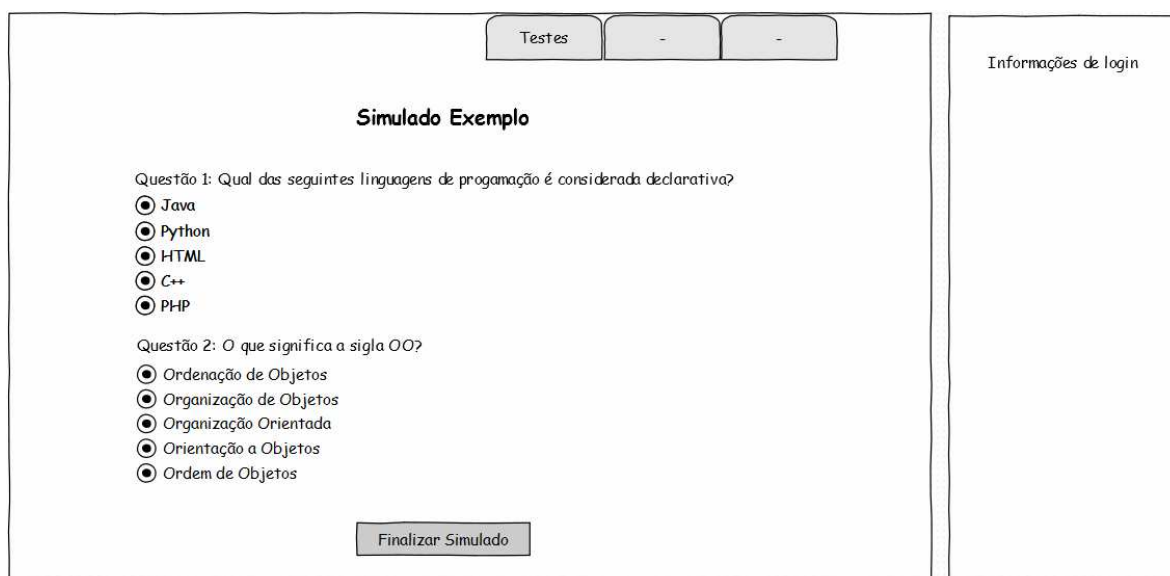


Fig. 13: Tela para realização de Simulados
Fonte: Elaborado pelos autores.

Na tela de realização de simulados, após o aluno responder as questões e finalizar o simulado, uma tela apresentando o resultado do seu teste será exibida.

Após o esboço das telas previstas no sistema, a equipe elaborou o fluxo de navegação do sistema que é a segunda atividade a ser realizada na tarefa de análise de requisitos. O fluxo de navegação é ilustrado pela figura 14.

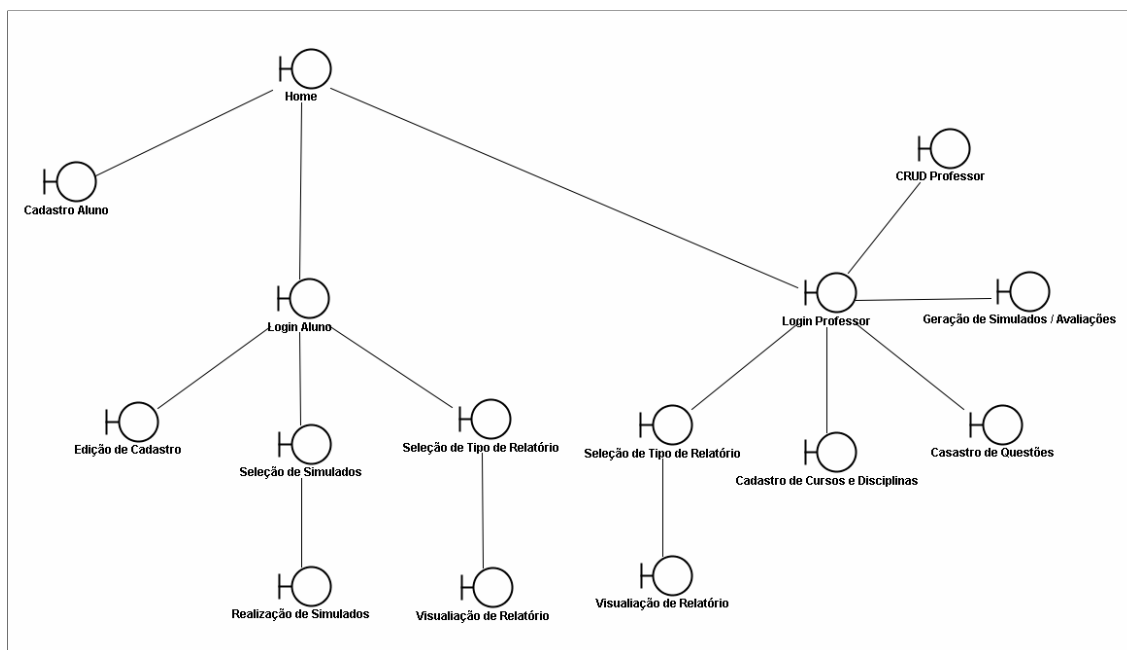


Fig. 14: Diagrama de Fluxo de Navegação
Fonte: Elaborado pelos autores.

De acordo com SILVA (2004), esta atividade é importante para que o usuário tenha uma visão clara de como será o sistema proposto, através dela é possível identificar as principais telas presentes no sistema. Na próxima sessão será apresentada a etapa de modelo de domínio.

5.1.2 Modelo de domínio

Esta tarefa teve como finalidade identificar as classes que podem representar os objetos do mundo real. Para sua realização, foi necessário uma análise de alguns simulados e testes, entre eles foram analisados exames do ENADE já ocorridos para se identificar os dados necessários ao sistema. Também foram levantados dados junto aos

professores da Universidade do Vale do Sapucaí sobre o exame e sobre as funcionalidades que eles julgam ser importantes para estarem presentes no SAS. O modelo de domínio do SAS é apresentado na figura 15.

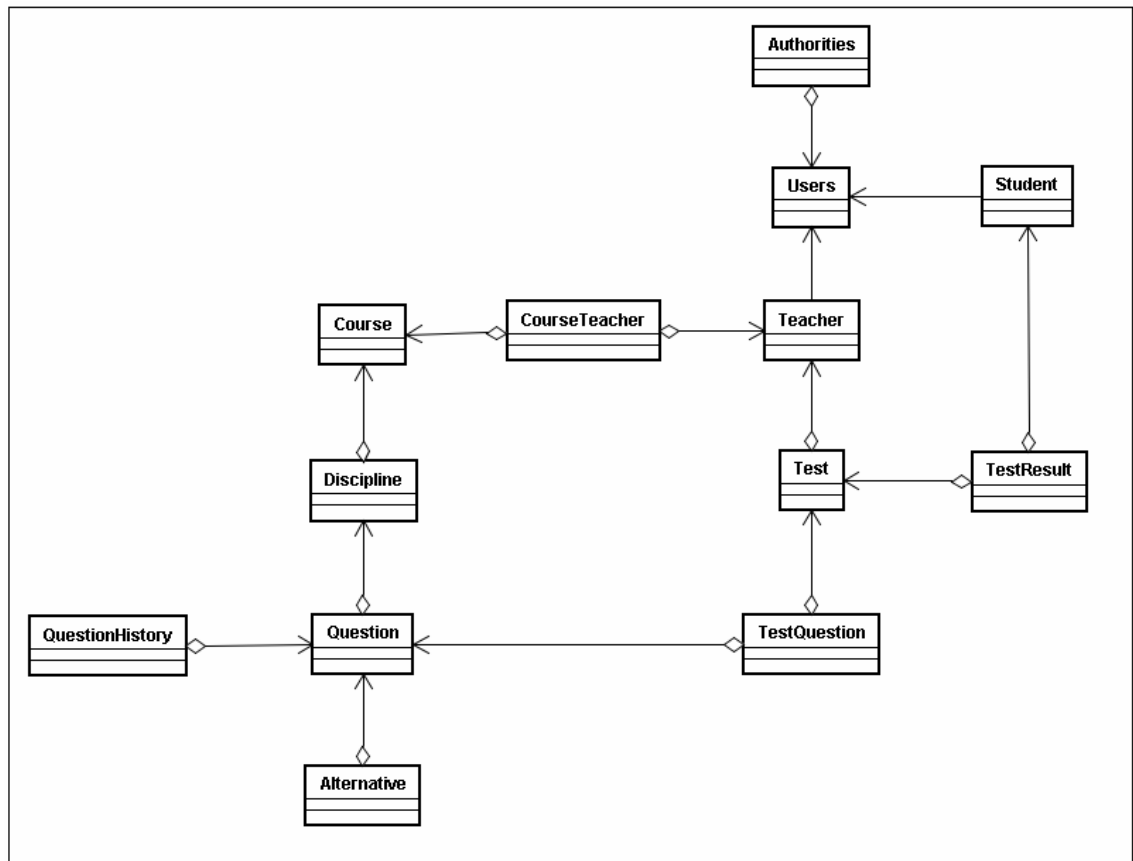


Fig. 15: Modelo de domínio
Fonte: Elaborado pelos autores.

Além de identificar os objetos do mundo real, o modelo de domínio deve mostrar todas as relações de generalização, associação e agregação. O próximo passo realizado foi identificação dos casos de uso envolvidos no sistema.

5.1.3 Identificação dos casos de uso

Segundo BONA e COSTA (2003), esta atividade consiste na identificação das funcionalidades do sistema a partir da atuação dos atores envolvidos. É através dos

casos de uso que podemos identificar os limites do sistema e sua interação com o ambiente.

De acordo com ICONIXPROCESS (2007), os casos de uso ajudam a responder algumas perguntas fundamentais:

- O que os usuários do sistema estão tentando fazer?
- Qual a experiência do usuário?

A quantidade de funcionalidades do sistema pode ser ditada pela maneira como os usuários devem interagir com ele.

O diagrama é composto pelos seguintes componentes:

- Ator: especifica um papel executado por um usuário ou outro sistema que interage com a aplicação.
- Casos de uso: descreve as funcionalidades disponibilizadas pelo sistema aos atores envolvidos. O conjunto de casos de uso de um sistema representa todas as formas através das quais o mesmo poderá ser utilizado.

A figura 16 apresenta os estereótipos que são utilizados no diagrama de casos de uso.

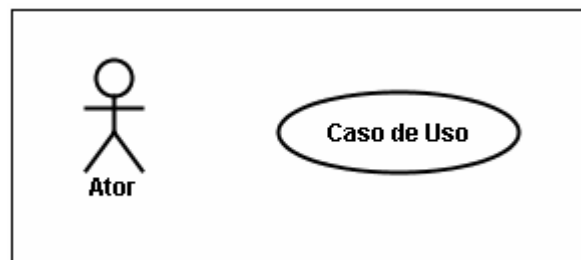


Fig. 16: Componentes do Caso de Uso.

Fonte: Elaborado pelos autores.

Os casos de uso mapeados no SAS são exibidos na figura 17. Neste diagrama são identificados os dois principais atores do sistema: aluno e professor. Também pode-se visualizar os sete casos de uso e qual(is) ator(es) interagem com os mesmos.

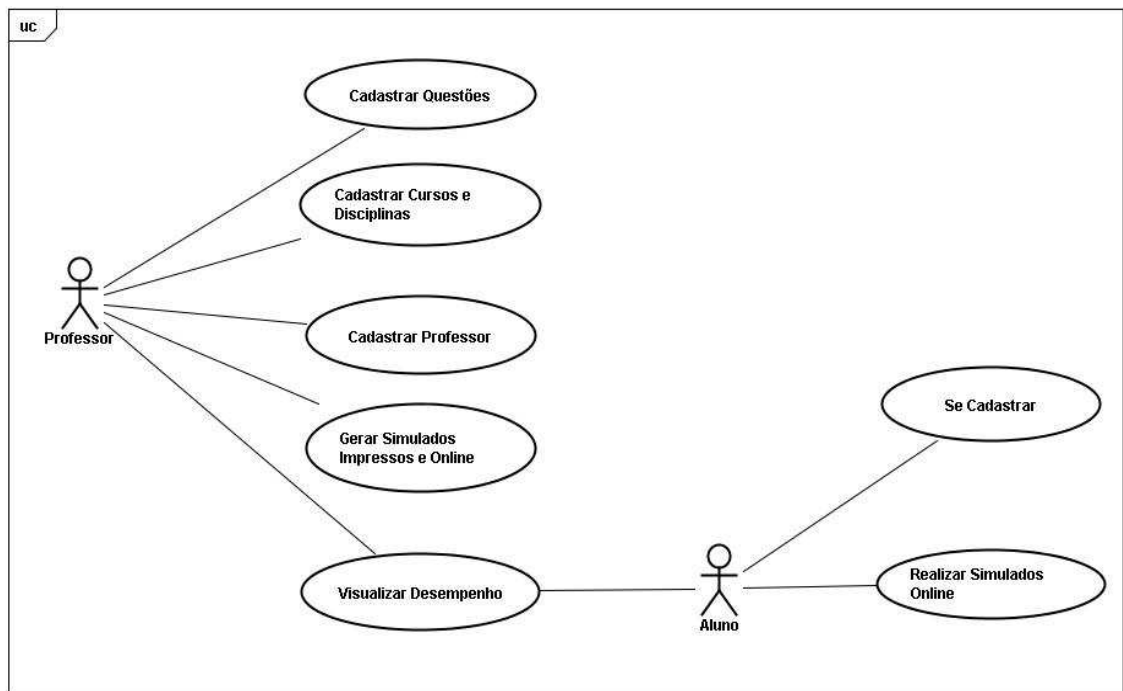


Fig. 17: Casos de Uso.
Fonte: Elaborado pelos autores.

A identificação dos casos de uso é a última etapa da análise de requisitos e é fundamental para a próxima fase, que irá trabalhar diretamente com os dados modelados no mapeamento de casos de uso.

5.2 Análise e projeto preliminar

Nesta fase, como afirma BONA e COSTA (2003), os casos de uso são descritos com fluxo principal das ações, podendo conter o fluxo alternativo e o fluxo de exceção. Também deve ser apresentada a análise de robustez. Sendo que, para cada caso de uso deve-se identificar um conjunto de objetos e atualizar o diagrama de classes do modelo de domínio.

5.2.1 Descrição dos casos de uso

A descrição dos casos de uso é a primeira atividade da etapa de análise e projeto preliminar. Esta atividade consiste em detalhar os casos de uso que foram identificados

anteriormente. A tabela 2 apresenta o fluxo de eventos para o caso de uso Cadastrar Professor.

Nome do caso de uso	Cadastrar Professor
Ator principal	Professor
Pré-condições	Estar logado no sistema.
Fluxo principal	
Ações do ator	Respostas do sistema
1. Clica em Professores.	2. Busca os professores já cadastrados. 3. Apresenta uma tela com a listagem de professores e opções para cadastro, pesquisa, remoção e edição.
4. Seleciona a opção de cadastro. 5. Entra com os dados do professor a ser cadastrado. 6. Clica em Cadastrar.	7. Salva os dados do professor no banco de dados. 8. Apresenta uma mensagem na tela informando o sucesso da inserção.
9. Se necessário volta ao item 4 para cadastrar outros professores.	
Fluxo alternativo 1	
1. No item 4 do fluxo principal, seleciona a opção pesquisar. 2. Entra com o nome parcial do professor.	3. Será apresentada a listagem de professores que têm o nome iniciado com o dado informado.
Fluxo alternativo 2	
1. No item 4 ou 9 do fluxo principal, seleciona a opção de edição de um cadastro específico na listagem apresentada. 2. Entra com os novos dados para o professor. 3. Clica em Atualizar.	4. Será apresentada uma mensagem de sucesso na edição do cadastro e a listagem de professores com as informações atualizadas.
Fluxo alternativo 3	
1. No item 4 ou 9, seleciona a opção de exclusão de um cadastro específico na listagem apresentada.	2. Será apresentada uma mensagem de sucesso na exclusão do cadastro e a listagem de professores com as informações atualizadas.

Tabela 2: Fluxo de eventos para caso de uso Cadastrar Professor

Fonte: Elaborado pelos autores.

No fluxo de eventos para cadastro de professor, exibido na tabela 2, pode-se identificar um fluxo principal e três fluxos de eventos alternativos. Para cada ação ou

conjunto de ações do ator há uma ação ou conjunto de ações do sistema. A tabela 3 exibe o fluxo de eventos para o caso de uso Realizar Simulados *Online*.

Nome do caso de uso	Realizar Simulados <i>Online</i>
Ator principal	Aluno
Pré-condições	Estar logado no sistema.
Fluxo principal	
Ações do ator	Respostas do sistema
1. Clica em Testes.	2. Uma tela requerendo dados sobre o teste e uma opção para pesquisa será apresentada.
3. Informa o código e a senha do teste. 4. Clica em Realizar Simulado.	5. Consulta no banco o teste com o código e senha informado. 6. Apresenta o nome do simulado, suas questões e alternativas em uma tela para o aluno responder.
7. Informa para cada questão a alternativa que julgue estar correta. 8. Clica em Finalizar Simulado.	9. Salva no banco de dados o desempenho do aluno. 10. Exibe na tela o número de questões que o aluno acertou e o número de questões que errou.
Fluxo alternativo 1	
1. No item 3 do fluxo principal, seleciona a opção pesquisar. 2. Entra com o nome parcial do teste.	3. Será apresentada a listagem com dados sobre os simulados e avaliações que têm o nome iniciado com o dado informado.
4. Seleciona opção Realizar Simulados. 5. Volta ao item 3 do fluxo principal.	

Tabela 3: Fluxo de eventos para caso de uso Realizar Simulado Online

Fonte: Elaborado pelos autores.

Terminada a atividade de descrever os casos de uso, deu-se início a atividade de análise de robustez, apresentada na próxima sessão.

5.2.2 Análise de robustez

Segundo ICONIXPROCESS (2007), a análise de robustez é um híbrido entre diagrama de classe e diagrama de atividade. É uma representação gráfica de um

comportamento descrito por um caso de uso mostrando tanto as classes participantes quanto o comportamento do software. Cada classe é representada por um estereótipo gráfico conforme mostrado na figura 18.

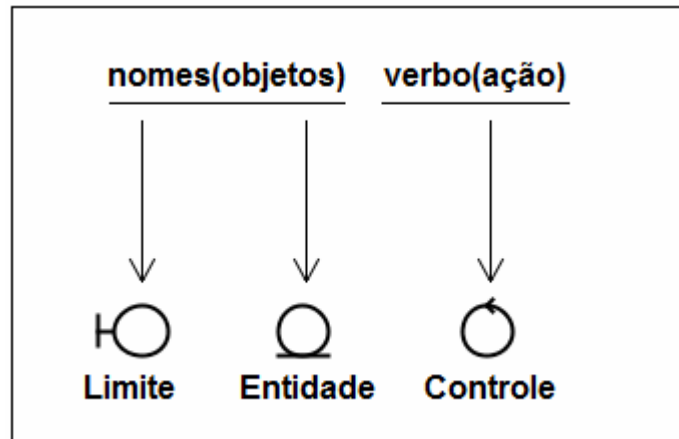


Fig. 18: Estereótipos da análise de robustez.
Fonte: Elaborado pelos autores.

Os estereótipos apresentados na figura 18 são:

- **Limite:** São as classes de interface entre o sistema e o ambiente, normalmente são as telas ou as páginas *web*.
- **Entidade:** São as classes do modelo de domínio.
- **Controle:** São as classes de ligação entre os objetos de limite e os objetos de entidade.

É útil pensar em objetos de limite e objetos de entidade como sendo nomes e em controles como sendo verbos se seguirmos as seguintes regras ao desenhar um diagrama de robustez:

- Nomes podem se comunicar apenas com verbos (e vice versa);
- Nomes não podem se comunicar com outros nomes;
- Verbos podem falar com outros verbos.

As figuras apresentadas a seguir são parte dos diagramas de robustez do projeto. Elas ilustram a forma como os objetos se interagem em cada caso de uso. A figura 19 representa o diagrama de robustez para o caso de uso gerar simulados impressos e *online*.

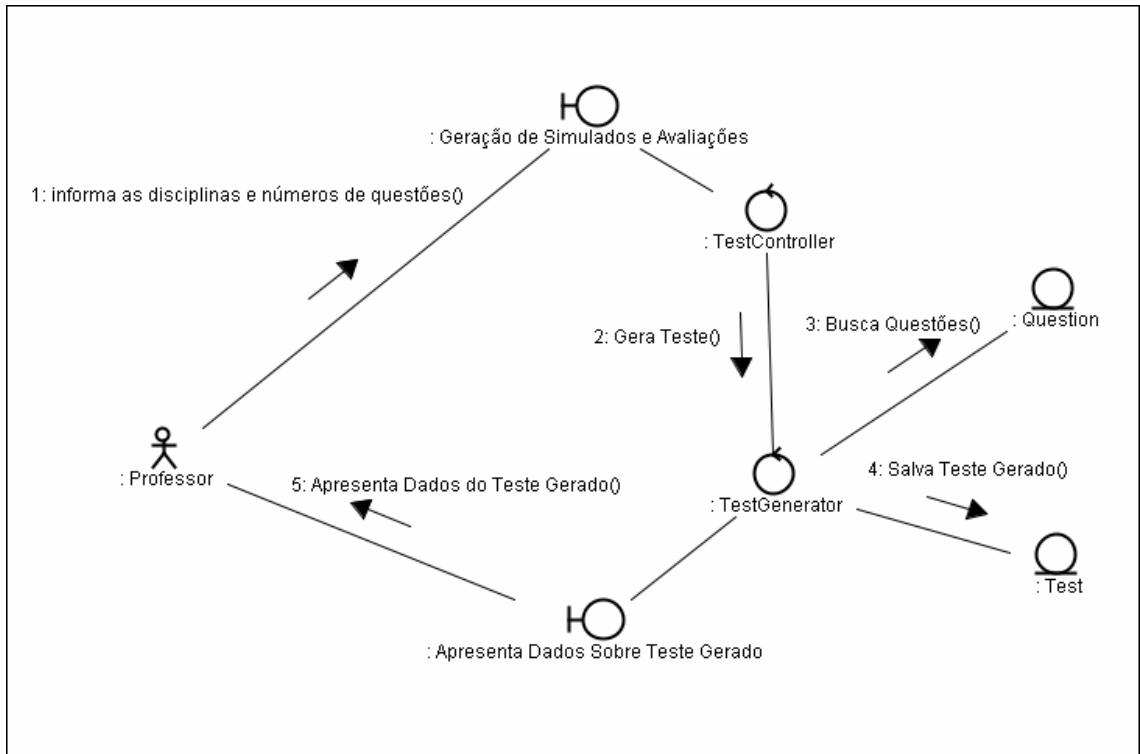


Fig. 19: Diagrama de robustez do caso de uso Gerar Simulados Impressos e Online
Fonte: Elaborado pelos autores.

Na figura 19 nota-se o fluxo de dados entre os vários objetos relacionados ao caso de uso, permitindo também que sejam identificados os objetos de limite, os objetos de controle e os objetos de entidade envolvidos. A figura 20 ilustra o diagrama de robustez para o caso de uso realizar simulado *online*.

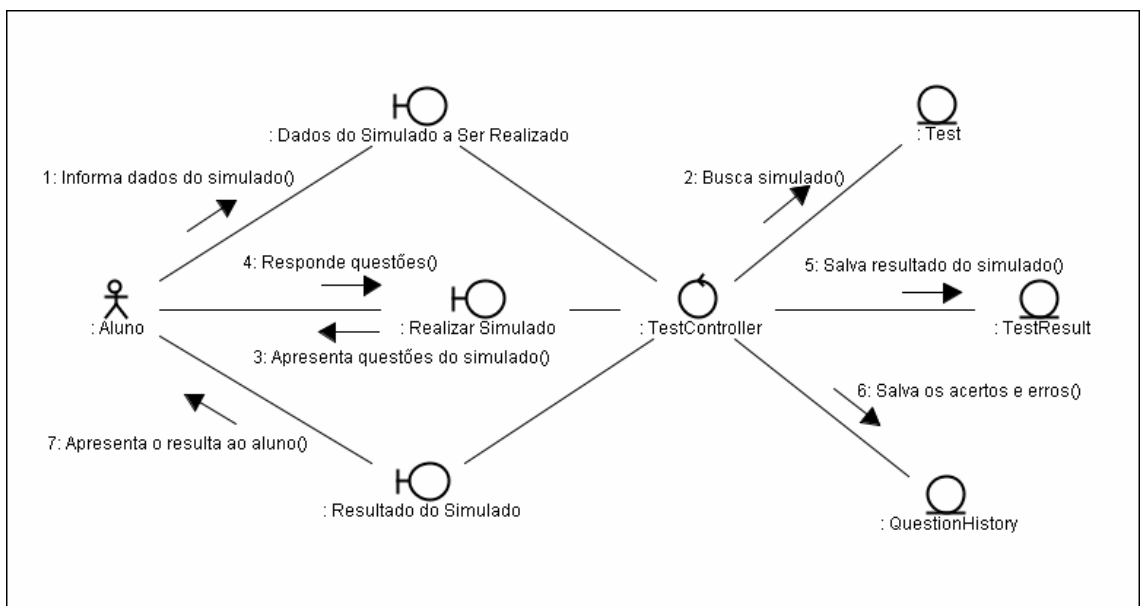


Fig. 20: Diagrama de robustez do caso de uso Realizar Simulado Online
Fonte: Elaborado pelos autores.

Uma vez completada a fase de análise de robustez, é necessário fazer uma revisão para verificar se a análise de robustez, o modelo de domínio e os casos de uso estão condizentes, explica ICONIXPROCESS (2007). Só após esta revisão que se deve avançar para a etapa apresentada na próxima sessão.

5.2.3 Atualização do modelo de domínio

O modelo de domínio deve ser atualizado ao mesmo tempo em que os casos de uso são descritos e a análise de robustez é realizada. Neste passo devem ser adicionados os atributos e métodos identificados. A Figura 21 mostra o modelo de domínio atualizado, neste foram omitidos os métodos de acesso aos atributos (*getters and setters*) por motivos de simplificação do diagrama, os autores não constataram a necessidade da descrição dos mesmos.

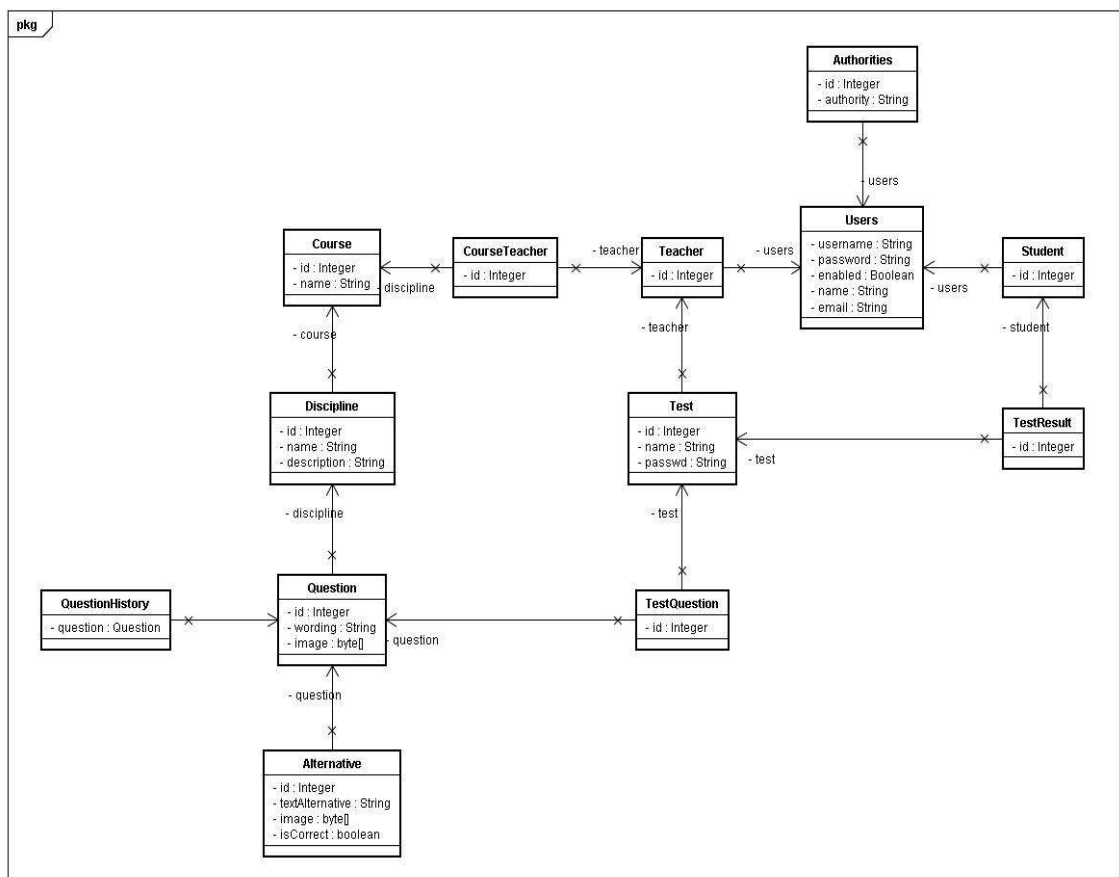


Fig. 21: Atualização do Modelo de Domínio adicionando os atributos das Entidades
Fonte: Elaborado pelos autores.

De acordo ROSENBERG (2005), o modelo de domínio também é um glossário do projeto, que permite que todos os artefatos subsequentes como, por exemplo, os diagramas de sequência, sejam escritos usando nomes consistentes.

5.3 Projeto

Esta fase tem a função de dar apoio à fase de desenvolvimento, analisando o comportamento do sistema. Ela consiste em detalhar o comportamento de cada caso de uso, identificando os objetos, as mensagens trocadas entre objetos e os métodos que são invocados e, após isso, terminar o modelo estático adicionando informações detalhadas sobre o projeto. Também é necessário analisar se os requisitos funcionais são atendidos pelo projeto. Na próxima sessão são apresentados os diagramas de sequência, a primeira tarefa desta fase.

5.3.1 Diagramas de sequência

Nesta etapa, para cada caso de uso ou diagrama de robustez existente é elaborado um diagrama de sequência. Serão exibidos a seguir, diagramas de sequência referentes a dois casos de uso do sistema. A figura 22 ilustra o detalhamento do comportamento do caso de uso cadastrar questões.

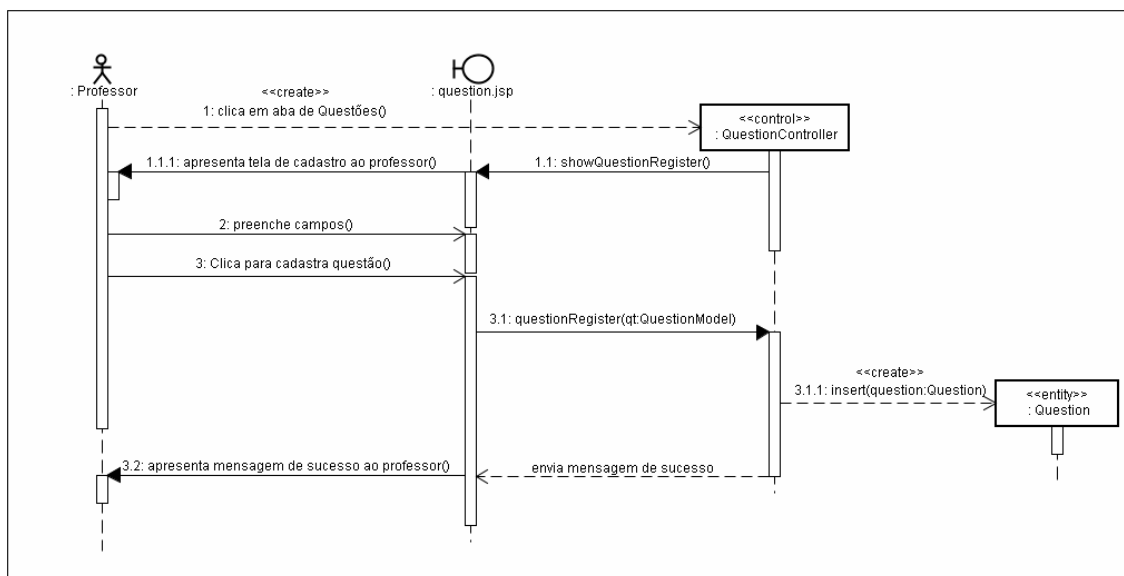


Fig. 22: Diagrama de Sequência Cadastrar Questões
Fonte: Elaborado pelos autores.

O objetivo dessa etapa é evidenciar o fluxo de mensagens entre os objetos envolvidos em cada caso de uso e os atores. A figura 23 é mais um exemplo de diagrama de sequência do trabalho, ela ilustra os detalhes do caso de uso gerar simulados impressos e *online*.

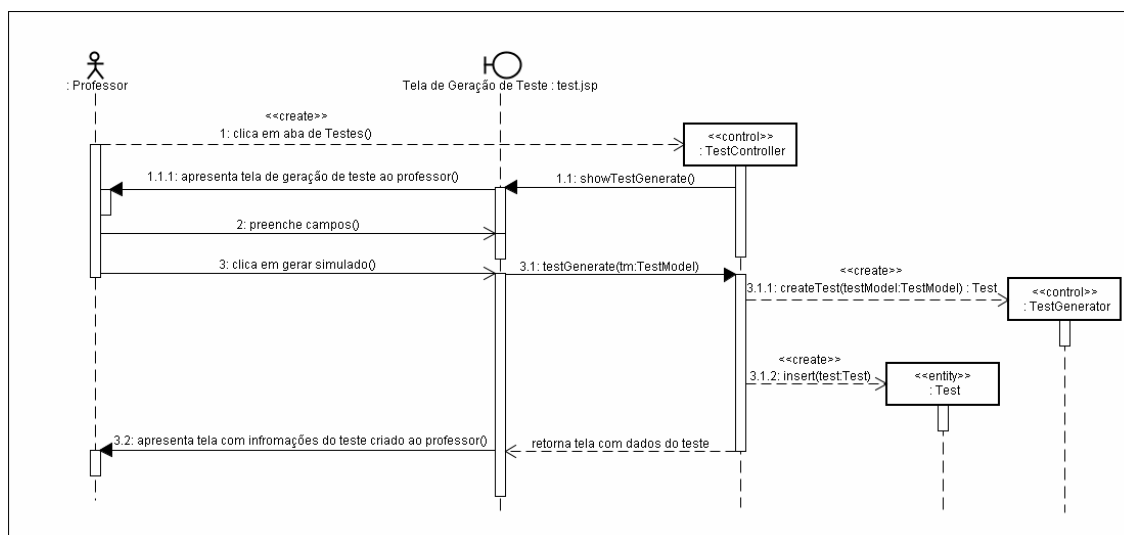


Fig. 23: Diagrama de Sequência Gerar Simulados Impressos e *Online*
Fonte: Elaborado pelos autores.

Na figura 23 pode-se perceber a interação do professor com a página de geração de testes e os outros objetos e interações necessários a essa atividade. Conforme explica

BORILLO (2010), nos diagramas de sequência são identificados os seguintes tipos de elementos:

- Objetos, os mesmos do diagrama de robustez. Eles são representados com o nome do objeto e opcionalmente com o nome da classe a qual ele pertence (objeto:classe).
- Mensagens, que são representadas por setas entre os objetos.
- Métodos (ou operações), os quais são exibidos como retângulos no decorrer das linhas tracejadas do objeto ao qual o método pertence.
- Texto sobre o curso de ações do caso de uso. Este item normalmente se localiza no lado esquerdo superior do diagrama.

Após encerrada a etapa da modelagem dos diagramas de sequência, passou-se à etapa de conclusão do modelo estático. Esta etapa que conclui a fase de projeto e será apresentada na próxima sessão.

5.3.2 Conclusão do modelo estático

Nesta etapa o diagrama de classe é atualizado baseando-se nos diagramas de sequências que foram desenvolvidos. A figura 24 apresenta o modelo de domínio depois da atualização.

5.4 Desenvolvimento do projeto

A fase de desenvolvimento do projeto é composta pelas etapas de codificação e teste. Esta fase está fora do foco de interesse do ICONIX, entretanto BONA(2002) explica que o ICONIX fornece um conjunto de sugestões, são elas:

- Pessoas com experiência em desenho de interface ou técnicos com experiência na produção de manuais de usuário devem escrever os casos de uso;
- É desejável que os modelos de domínio e diagramas de classe sejam construídos por pessoas com experiência em projeto de banco de dados;
- Programadores devem pensar em aspectos como desempenho, segurança, e devem ser responsáveis pelos diagramas de estado e colaboração, caso forem utilizados.
- O diagrama de sequência deve ser realizado, ou pelo menos supervisionado, por pessoas com experiência em modelação OO.

Nas próximas sessões são apresentadas as etapas de codificação e testes.

5.4.1 Codificação

A codificação do projeto foi realizada no ambiente de desenvolvimento do Eclipse IDE, uma plataforma largamente usada que possui suporte às mais diversas linguagens de programação. Dentre os critérios para escolha das tecnologias buscou-se o que há de mais recente e que atendesse da melhor forma as necessidades do projeto.

A linguagem Java foi escolhida por inúmeras características, muitas delas já citadas no trabalho, como por exemplo, ser multiplataforma e possuir um grande número de bibliotecas e APIs disponíveis para apoiar o desenvolvimento. O Spring é um *framework* com inúmeros recursos, o seu módulo MVC foi o que mais foi abordado pelo trabalho. Este módulo possibilitou que a aplicação fosse desenvolvida seguindo as boas práticas de desenvolvimento, mantendo uma determinada separação entre as classes de persistência, controle e apresentação.

O Apache Tomcat foi escolhido para a execução da aplicação. Ele foi utilizado por ser um *web container* muito conhecido pela equipe, tanto durante a experiência acadêmica como durante a profissional. Instalando a aplicação no Tomcat, a mesma pôde ser testada através de qualquer navegador de internet com acesso à rede onde o *software* se encontra instalado.

5.4.2 Testes

Nesta etapa houve uma análise do sistema, verificando se o mesmo atende a todos os requisitos funcionais estabelecidos no início da modelagem. Isso foi feito através de testes por toda a interface do sistema, navegando pelas telas, inserindo registros, criando simulados e visualizando relatórios.

Os testes apresentaram resultados satisfatórios, as funcionalidades propostas pelo trabalho apresentaram um comportamento adequado, necessitando apenas alguns ajustes na disposição dos componentes gráficos e o incremento de recursos em algumas funcionalidades.

Para a execução dos testes foram elaborados alguns casos de teste que serviram como um roteiro durante a realização dos mesmos. Nestes casos de teste são descritas as várias ações durante a operação do sistema, o resultado esperado e o resultado obtido durante a navegação pelo sistema. Também foram simuladas algumas situações de erro.

A tabela 4 apresenta um exemplo de caso de teste elaborado para o SAS.

Objetivo do teste: Realização de Simulados e Avaliações			
Tipo	Descrição	Resultado esperado	Resultado obtido
Funcional	Logado como aluno, clica na aba Testes do menu.	A tela de realização de simulados e avaliações será apresentada na tela.	Sucesso.
Funcional	Entra com código e senha da avaliação e clica em realizar teste.	As questões e alternativas da avaliação são apresentadas na tela.	Sucesso.
Funcional	Escolhe as alternativas e clica em finalizar teste.	O resultado da avaliação é exibido na tela. Também são exibidos o aproveitamento e os resultados de outros testes feitos pelo aluno.	Sucesso.

Tabela 4: Caso de teste para Realização de Simulados ou Avaliações

Fonte: Elaborado pelos autores.

Os demais casos de testes criados para o sistema podem ser encontrados nos anexos deste trabalho.

6 DISCUSSÃO DOS RESULTADOS

São abordados neste capítulo, os resultados alcançados a partir dos objetivos propostos por este trabalho. Durante o mesmo, foram estudados conceitos relacionados a avaliações e alguns simulados, como por exemplo, os simulados preparatórios para o ENADE e para exames pré-vestibulares, e a avaliação institucional.

Entender esses conceitos serviu de base para a definição das funcionalidades da aplicação, pois através desse estudo, procurou-se buscar os dados necessários para se levantar os requisitos do sistema a ser desenvolvido.

Também foram abordadas as várias tecnologias utilizadas na implementação do *software*, aprofundando-se um pouco mais no *framework* Spring, cumprindo assim, um dos objetivos do trabalho, que era o de demonstrar alguns de seus recursos.

O *framework* Spring foi muito importante para o projeto. O seu estudo e sua utilização na codificação do *software* proporcionaram grandes vantagens, como a simplificação da estrutura e do desenvolvimento da ferramenta. Os conceitos de injeção de dependência e orientação a aspecto que o Spring incorpora, possibilitam um código mais limpo e menos dependência entre as classes da aplicação, o que torna o código mais legível, facilitando sua manutenção.

O ambiente *web* foi uma escolha que trouxe um diferencial a mais para o sistema, tornando o mesmo diferenciado em relação aos demais *softwares* do mercado que possuem características semelhantes ao SAS. A maior vantagem deste ambiente é o programa não depender de ser instalado em todo computador que irá utilizá-lo, uma vez que, uma aplicação *web* somente depende de um navegador com acesso a rede, a qual disponibilizará o acesso ao sistema.

O uso da linguagem de programação Java propiciou, além das várias vantagens listadas no trabalho, maior produtividade ao processo de codificação. Pois a equipe já conhecia a linguagem, tanto através do curso, onde esteve em contato com a linguagem desde o início do terceiro período, como através da experiência profissional da equipe.

Estudou-se ainda a metodologia de desenvolvimento usada para apoiar o processo de construção da aplicação. A partir do entendimento dos artefatos e etapas da metodologia ICONIX, foi elaborada a documentação do sistema, a qual seguiu cada

passo proposto pela metodologia. Com isso, a modelagem e a documentação foram concluídas, cumprindo mais um dos objetivos.

A implementação das funcionalidades propostas pelo projeto também foi concluída. Foram desenvolvidos os cadastros, os relatórios, as telas de geração e realização de provas. Cumprindo assim, todos os objetivos propostos pelo trabalho.

A equipe pretende agora, com a finalização do trabalho, dar continuidade na codificação do software, através da adição e incremento de funcionalidades, melhorando-o constantemente, visando assim, transformá-lo em um produto para atender as diversas instituições de ensino que podem se beneficiar desta ferramenta em seus processos avaliativos.

Os resultados obtidos com este trabalho foram bastante satisfatórios, uma vez que todos os objetivos propostos foram atingidos, possibilitando desta forma, a construção de uma ferramenta, a qual proporcionará a otimização dos processos avaliativos de instituições educacionais. Enfatizando também que a necessidade de uma ferramenta com as características do SAS, foi relatada por diversos professores da universidade e de outras instituições.

CONCLUSÃO

Concluimos com o trabalho realizado que a ferramenta proposta mostrou possuir uma grande aplicabilidade no ambiente educacional, onde irá possibilitar uma maior agilidade e diversidade nos processos de avaliação e de preparação do aluno.

A ferramenta também proporcionará uma visão, através de relatórios, do rendimento da avaliação e do desempenho do aluno, além de apontar quais questões ou disciplinas tiveram maior índice de erros e acertos. Desta forma é possível saber quais conteúdos necessitam de maior atenção dentro processo de aprendizagem.

Os conceitos relacionados a avaliações e simulados foram importantes para a definição do escopo do *software* desenvolvido, além de propiciar à equipe uma melhor visão sobre a abrangência da aplicação.

A metodologia de desenvolvimento ICONIX supriu de forma satisfatória todas as necessidades do projeto durante o processo de desenvolvimento. Pode-se concluir que esta metodologia é menos burocrática que o RUP, possibilitando uma maior agilidade durante o desenvolvimento, mas ainda gerando uma documentação importante e necessária para apoiar a codificação e auxiliar durante todo o processo produtivo.

As tecnologias utilizadas se mostraram condizentes com a realidade do mercado e atenderam as expectativas, agregando também conhecimento aos participantes. O Spring foi decisivo para a estruturação do projeto, pois a simplificação é uma vantagem marcante deste *framework*.

A equipe considerou a realização deste trabalho muito satisfatória, pois além de todos os conceitos e tecnologias abordados, a ferramenta se mostrou com potencial para ser aplicada nas instituições de ensino.

REFERÊNCIAS

BATES, B; SIERRA, K; BASHAM B. **Head First Servlets and JSP**. Sebastopol, US: Editora O'Reilly, 2004.

BARROS, A. de J. P. de; LEHFELD, N. A. de S. **Projeto de pesquisa: propostas metodológicas**. Petrópolis: Editora Vozes, 2002.

BONA, C. **Avaliação de Processos de Software**: Um estudo de caso em XP e Iconix. Dissertação de Pós-Graduação. Florianópolis: Universidade Federal de Santa Catarina, 2002.

BONA, C; COSTA, M. T. C. da. **Processo de software**: Um estudo de caso em Iconix. III Congresso Brasileiro de Computação, 2003. Disponível em:<<http://200.169.53.89/download/CD%20congressos/2003/3%20CBCComp/html/artigos/cbcomp/engenharia%20de%20software/eng123.pdf>>. Acesso em: 14 de agosto de 2010 às 20:35.

BORILLO, D. **The iconix approach**, 2000. Disponível em:<<http://pst.web.cern.ch/PST/HandBookWorkBook/Handbook/engineeringHB.html>>. Acesso em: 11 de setembro de 2010 às 11:45.

BRASIL, Ministério da Educação. **Lei de Diretrizes e Bases da Educação Nacional**. Lei n.º 9394/96. Brasília, 1996.

_____. Ministério da Educação. SINAES. **Sistema Nacional de Avaliação da Educação Superior**: Da concepção à regulamentação. 2. ed. Brasília: Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira, 2004.

_____. **Sistema Nacional de Avaliação da Educação Superior**: Da concepção à regulamentação. 2. ed. Brasília: Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira, 2004. p.94.

_____. **Sistema Nacional de Avaliação da Educação Superior**: Da concepção à regulamentação. 2. ed. Brasília: Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira, 2004. p.97.

_____. **Sistema Nacional de Avaliação da Educação Superior**: Da concepção à regulamentação. 2. ed. Brasília: Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira, 2004. p.98.

_____. Presidência da República. **Lei n.º 10.861**. Brasília, 2004. Art.1 §1.

_____. Presidência da República. **Lei n.º 10.861**. Brasília, 2004. Art.4 §2.

_____. Presidência da República. **Lei n.º 10.861**. Brasília, 2004. Art.5 §2.

DORNELLES, C. **Conhece-te a ti mesmo, ensina-te a ti mesmo**. 2004. Disponível em: < http://www.ufrgs.br/tramse/med/textos/2004_07_16_tex.htm >. Acessado em 12 de abril de 2010 às 20:35.

DEITEL, H. M. **Java: Como Programar**. Tradução de Edson Furmankiewicz. 6. ed. São Paulo: Pearson Prentice Hall, 2005.

ECLIPSE, The Eclipse Foundation. **EclipseLink**. 2010. Disponível em: < <http://www.eclipse.org/eclipselink> >. Acessado em 27 de abril de 2010 às 20:23.

FERREIRA, A. B. de H. **Novo Dicionário da Língua Portuguesa**. 2. ed. Rio de Janeiro: Nova Fronteira, 1999.

GUEDES, G. T. A. **UML: Uma abordagem prática**. 2. ed. São Paulo: Novatec, 2006.

IBM. **Rational Unified Process**. 2010. Disponível em: < <http://www-01.ibm.com/software/br/rational/rup.shtml> >. Acessado em 16 de setembro de 2010 às 23:50.

LEVY, P. **Ciberultura**. Tradução Carlos Irineu da Costa. São Paulo: Editora 34, 1999.

MÁTTAR NETO, J. A. **Metodologia Científica na Era da Informática**. São Paulo: Saraiva, 2003.

MAXIMO, L. F; BARONE, D. A. C; CARVALHO M. J. S. **Informática e Avaliação na Educação a Distância na UFRGS: Um Panorama de 1998 A 2008**. Porto Alegre, 2008.

ORACLE. **Java Servlet Technology**. 2010. Disponível em: < <http://www.oracle.com/technetwork/java/index-jsp-135475.html> >. Acessado em 4 de setembro de 2010 às 11:45.

_____. **Java Servlet Technology Overview**. 2010. Disponível em: < <http://www.oracle.com/technetwork/java/index-jsp-135475.html> >. Acessado em 4 de setembro de 2010 às 12:00.

PERRENOUD, P. **Avaliação: Da Excelência à Regulação das Aprendizagens Entre Duas Lógicas**. São Paulo: Artmed Editora, 1999.

ROSENBERG, D; STEPHENS, M; COLLINS-COPE, M. **Agile Development with ICONIX Process: People, Process, and Pragmatism**. Berkeley, US: Apress, 2005.

ROSENBERG, D; SCOTT, K. **Use Case Driven Object Modeling with UML: A Practical Approach**. Boston, US: Addison-Wesley, 1999.

ROSENBERG, D; STEPHENS, M. **Use Case Driven Object Modeling with UML: Theory and practice**. Berkeley, US: Apress, 2007.

SILVA, M. R; **Processos de Desenvolvimento de Software: RUP e ICONIX**. Monografia. Londrina: Universidade Estadual de Londrina, 2004.

SUN MICROSYSTEMS. **Sobre a SUN**. Disponível em: < <http://br.sun.com/aboutsun/historia.jsp#1995> >. Acesso em: 11 de abril de 2010 às 20:40.

_____. **About Java Technology**. Disponível em: < <http://www.sun.com/java/about> >. Acesso em: 11 de abril de 2010 às 20:40.

_____. **The Java EE 6 Tutorial**, Volume I, 2010. Disponível em: < <http://java.sun.com/javae/6/docs/tutorial/doc/bnaaw.html> >. Acesso em: 11 de abril de 2010 às 21:19.

_____. **The J2EE Tutorial: What Is a Servlet?** Disponível em: < http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets2.html#75087 >. Acesso em: 4 de setembro de 2010 às 09:10.

WALLS, C. **Spring em Ação**, Tradução de Priscila Reiz Franz e Leandro Chu. 2. ed. Rio de Janeiro: Alta Book, 2008.

APÊNDICES

APÊNDICE I – STORYBOARDS

Professores - -

Ícone Cadastro Ícone Pesquisa

Nome:

Email:

Usuário:

Senha:

Cadastrar

Código	Nome	Usuário	Email	Editar	Remover
10	Luis Antonio Tavares	luis	luis.tavares@msn.com	<input type="checkbox"/>	[x]
11	Paul Di'anno	dianno	paul@example.com	<input type="checkbox"/>	[x]
12	Maria de Souza	maria	maria@example.com	<input type="checkbox"/>	[x]
13	Elizabete Mendes	elizabete	eli@example.com	<input type="checkbox"/>	[x]
14	Bruce Dickinson	bruce	bruce@example.com	<input type="checkbox"/>	[x]

Informações de login

Tela de Cadastro de Professor

Testes - -

Simulado Exemplo

Questão 1: Qual das seguintes linguagens de programação é considerada declarativa?

- Java
- Python
- HTML
- C++
- PHP

Questão 2: O que significa a sigla OO?

- Ordenação de Objetos
- Organização de Objetos
- Organização Orientada
- Orientação a Objetos
- Ordem de Objetos

Finalizar Simulado

Informações de login

Tela de Realização de Simulado

Testes			-	-
Informações de login				
Nome do simulado: <input type="text"/>				
Senha: <input type="text"/>				
Selecione as disciplinas e quantidade de questões				
<input type="checkbox"/>	Matemática			
<input type="checkbox"/>	Engenharia de Produção			
<input type="checkbox"/>	Sistemas de Informação			
		Disciplinas	Nº Questões	Fácil
				Médio
				Difícil
		Banco de dados	<input type="text"/>	<input type="text"/>
		Estrutura de Dados	<input type="text"/>	<input type="text"/>
		Java	<input type="text"/>	<input type="text"/>
		Linguagem C	<input type="text"/>	<input type="text"/>
<input type="checkbox"/>	Administração			
<input type="checkbox"/>	Biologia			
<input type="checkbox"/>	História			
<input type="checkbox"/>	Publicidade			
<input type="button" value="Gerar Simulado"/>				

Tela de Geração de Simulado

Questões			-	-
Informações de login				
<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; border-radius: 50%; padding: 5px; text-align: center;">Ícone Cadastro</div> <div style="border: 1px solid black; border-radius: 50%; padding: 5px; text-align: center;">Ícone Pesquisa</div> </div>				
Curso: <input type="text" value="Selecione"/>				
Disciplina: <input type="text" value="Selecione"/>				
Enunciado: <input type="text"/>				
Alternativa A: <input type="text"/>				
Alternativa B: <input type="text"/>				
Alternativa C: <input type="text"/>				
Alternativa D: <input type="text"/>				
Alternativa E: <input type="text"/>				
Alternativa correta: <input type="text" value="Selecione"/>				
Grau de dificuldade: <input type="text" value="Selecione"/>				
<input type="button" value="Cadastrar"/>				

Tela de Cadastro de Questões

Cursos e Disciplinas		-	-
Cursos (+)			
▷	Matemática	<input checked="" type="checkbox"/>	
▷	Engenharia de Produção	<input checked="" type="checkbox"/>	
▷	Sistemas de Informação	<input checked="" type="checkbox"/>	
Disciplinas (+)			
	Banco de dados	<input checked="" type="checkbox"/>	
	Estrutura de Dados	<input checked="" type="checkbox"/>	
	Java	<input checked="" type="checkbox"/>	
	Linguagem C	<input checked="" type="checkbox"/>	
▷	Administração	<input checked="" type="checkbox"/>	
Disciplinas (+)			
	Teorias Administrativas	<input checked="" type="checkbox"/>	
	Contabilidade	<input checked="" type="checkbox"/>	
	Cálculo Aplicado	<input checked="" type="checkbox"/>	
▷	Publicidade	<input checked="" type="checkbox"/>	
Informações de login			

Tela de Cursos e Disciplinas

Cadastro		-	-
Cadastro de Aluno			
Nome:	<input type="text"/>	<div style="border: 1px solid blue; background-color: #e6f2ff; padding: 5px; width: fit-content;"> Esta tela aparece se não houver aluno logado no sistema. </div>	
Email:	<input type="text"/>		
Usuário:	<input type="text"/>		
Senha:	<input type="password"/>		
<input type="button" value="Cadastrar"/>			
Informações de login			

Cadastro		-	-
Dados Conta			
Nome:	<input type="text" value="Adrian Smith"/>	<div style="border: 1px solid blue; background-color: #e6f2ff; padding: 5px; width: fit-content;"> Esta tela aparece se já houver aluno logado no sistema. </div>	
Email:	<input type="text" value="adrian@example.com"/>		
Usuário:	<input type="text" value="adrian"/>		
Senha:	<input type="password" value="*****"/>		
<input type="button" value="Editar"/> <input type="button" value="Excluir"/>			
Informações de login			

Tela de Cadastro de Alunos

Relatórios - -

Relatório por Aluno Relatório por Questões Relatório por Simulado

Aluno:

Teste	Número de Questões	Número de Acertos	Aproveitamento
Simulado ENADE	59	34	54%
Simulado ENEM	70	35	59%
Simulado Vestibular Univas	79	44	58%
Simulado UFMG	65	23	62%
Simulado USP	70	45	59%

Informações de login

Tela de Relatório por Aluno

Relatórios - -

Relatório por Aluno Relatório por Questões Relatório por Simulado

Simulado/Avaliação:

Aluno	Número de Acertos	Número de Questões	Desempenho Aluno	Desempenho Geral
Fernando da Silva	17	34	54%	60%
Felipe Albuquerque	14	35	59%	40%
João Simões	25	44	58%	62%
Paul Di'anno	12	23	62%	54%
Ana Ferreira	25	45	59%	61%

Informações de login

Tela de Relatório por Teste

Relatórios - -

Relatório por Aluno
Relatório por Questões
Relatório por Simulado

Selecione o simulado: ▼

Questões Mais Erradas	Número Erros	Ver	Questões Mais Acertadas	Número Acertos	Ver
Qual ferramenta ...	120	<input type="checkbox"/>	Qual ferramenta ...	120	<input type="checkbox"/>
Qual linguagem ...	34	<input type="checkbox"/>	Qual linguagem ...	34	<input type="checkbox"/>
Que framework ...	45	<input type="checkbox"/>	Que framework ...	45	<input type="checkbox"/>
Qual o método ...	23	<input type="checkbox"/>	Qual o método ...	23	<input type="checkbox"/>
O que faz ...	56	<input type="checkbox"/>	O que faz ...	56	<input type="checkbox"/>

Informações de login

Tela de Relatório por Questão

APÊNDICE II – FLUXOS DE EVENTOS

Nome do caso de uso	Cadastrar Professor
Ator principal	Professor
Pré-condições	Estar logado no sistema.
Fluxo principal	
Ações do ator	Respostas do sistema
1. Clica em Professores.	2. Busca os professores já cadastrados. 3. Apresenta uma tela com a listagem de professores e opções para cadastro, pesquisa, remoção e edição.
4. Seleciona a opção de cadastro. 5. Entra com os dados do professor a ser cadastrado. 6. Clica em Cadastrar.	7. Salva os dados do professor no banco de dados. 8. Apresenta uma mensagem na tela informando o sucesso da inserção.
9. Se necessário volta ao item 4 para cadastrar outros professores.	
Fluxo alternativo 1	
1. No item 4 do fluxo principal, seleciona a opção pesquisar. 2. Entra com o nome parcial do professor.	3. Será apresentada a listagem de professores que têm o nome iniciado com o dado informado.
Fluxo alternativo 2	
1. No item 4 ou 9 do fluxo principal, seleciona a opção de edição de um cadastro específico na listagem apresentada. 2. Entra com os novos dados para o professor. 3. Clica em Atualizar.	4. Será apresentada uma mensagem de sucesso na edição do cadastro e a listagem de professores com as informações atualizadas.
Fluxo alternativo 3	
1. No item 4 ou 9 do fluxo principal, seleciona a opção de exclusão de um cadastro específico na listagem apresentada.	2. Será apresentada uma mensagem de sucesso na exclusão do cadastro e a listagem de professores com as informações atualizadas.

Fluxo de Eventos Para Cadastro de Professor

Nome do caso de uso	Realizar Simulados <i>Online</i>
Ator principal	Aluno
Pré-condições	Estar logado no sistema.
Fluxo principal	
Ações do ator	Respostas do sistema
1. Clica em Testes.	2. Uma tela requerendo dados sobre o teste e uma opção para pesquisa será apresentada.
3. Informa o código e a senha do teste. 4. Clica em Realizar Simulado.	5. Consulta no banco o teste com o código e senha informado. 6. Apresenta o nome do simulado, suas questões e alternativas em uma tela para o aluno responder.
7. Informa para cada questão a alternativa que julgue estar correta. 8. Clica em Finalizar Simulado.	9. Salva no banco de dados o desempenho do aluno. 10. Exibe na tela o número de questões que o aluno acertou e o número de questões que errou.
Fluxo alternativo 1	
1. No item 3 do fluxo principal, seleciona a opção pesquisar. 2. Entra com o nome parcial do teste.	3. Será apresentada a listagem com dados sobre os simulados e avaliações que têm o nome iniciado com o dado informado.
4. Seleciona opção Realizar Simulados. 5. Volta ao item 3 do fluxo principal.	

Fluxo de Eventos Para Realizar Simulado *Online*

Nome do caso de uso	Cadastrar Questões
Ator principal	Professor
Pré-condições	Estar logado no sistema.
Fluxo principal	
Ações do ator	Respostas do sistema
1. Clica em Questões.	2. A tela de pesquisa e cadastro de questões será aberta.
3. Clica em Cadastrar Questões. 4. Seleciona o curso e disciplina para a questão. 5. Entra com os dados da questão. 6. Clica em Cadastrar	7. Salva questão no banco de dados. 8. Apresenta a tela de pesquisa e cadastro de questões novamente.
Fluxo alternativo 1	
1. No item 3 do fluxo principal, seleciona a opção pesquisar. 2. Entra com o enunciado parcial da questão.	3. Será apresentada a listagem com dados sobre as questões que têm o enunciado iniciado com o dado informado.
4. Seleciona opção Excluir Questão.	5. Exclui questão do banco de dados.

Fluxo de Eventos Para Cadastrar Questões

Nome do caso de uso	Gerar Simulados Impressos e <i>Online</i>
Ator principal	Professor
Pré-condições	Estar logado no sistema.
Fluxo principal	
Ações do ator	Respostas do sistema
1. Clica em Testes.	2. Uma tela com uma lista de cursos e disciplinas será apresentada.
3. Informa o nome e senha do teste. 4. Seleciona as disciplinas e números de questões para cada uma delas. 5. Clica em Gerar Teste.	6. Gera teste com de acordo com as disciplinas e números de questões informadas e salva no banco. 7. Apresenta na tela o nome do teste, seu código e senha.

Fluxo de Eventos Para Gerar Simulados Impressos e *Online*

Nome do caso de uso	Auto Cadastro do Aluno
Ator principal	Aluno
Pré-condições	Nenhuma
Fluxo principal	
Ações do ator	Respostas do sistema
1. Clica em Cadastro.	2. A tela de cadastro de aluno será aberta.
3. Informa seus dados. 4. Informa nome de usuário e senha para sistema. 5. Clica em Cadastrar.	6. Salva dados do aluno no banco de dados. 7. Apresenta tela confirmando o cadastro.
Fluxo alternativo 1	
1. Após o item 7 do fluxo principal clica em editar. 2. Entra com os novos dados. 3. Clica em Atualizar.	4. Salva novos dados do aluno no banco de dados. 5. Apresenta tela confirmando a atualização dos dados.
Fluxo alternativo 2	
1. Após o item 7 do fluxo principal clica em excluir conta.	2. Exclui registro do banco de dados. 3. Apresenta tela confirmando exclusão.

Fluxo de Eventos Para Cadastrar Aluno

Nome do caso de uso	Cadastrar Curso e Disciplina
Ator principal	Professor
Pré-condições	Estar logado no sistema.
Fluxo principal	
Ações do ator	Respostas do sistema
1. Clica em Cursos e Disciplinas.	2. A tela de cadastro de cursos e disciplinas será aberta apresentando uma listagem de cursos e disciplinas já apresentadas.
3. Clica em Adicionar Curso. 4. Informa dados sobre o curso a ser inserido. 5. Clica em Cadastrar.	6. Salva dados do curso no banco de dados. 7. Apresenta tela de cursos e disciplinas com a inclusão do novo curso na listagem.
Fluxo alternativo 1	
1. No item 3 do fluxo principal clica em Adicionar Disciplina no curso correspondente. 2. Entra com o nome da disciplina. 3. Clica em Cadastrar.	4. Salva os dados da disciplina no banco de dados. 5. Apresenta tela de cursos e disciplinas com a inclusão da nova disciplina na listagem.
Fluxo alternativo 2	
1. No item 3 do fluxo principal clica em Excluir Curso.	2. Exclui registro do banco de dados. 3. Apresenta tela de cursos e disciplinas com a exclusão do curso da listagem.
Fluxo alternativo 3	
1. No item 3 do fluxo principal clica em Excluir Disciplina.	2. Exclui registro do banco de dados. 3. Apresenta tela de cursos e disciplinas com a exclusão da disciplina da listagem.

Fluxo de Eventos Para Cadastro de Curso e Disciplina

Nome do caso de uso	Visualizar Relatórios
Ator principal	Professor / Aluno
Pré-condições	Estar logado no sistema.
Fluxo principal	
Ações do ator	Respostas do sistema
1. Clica em Relatórios.	2. A tela para escolha do relatório será apresentada.
3. Seleciona o relatório. 4. Entra com dados requeridos sobre o relatório escolhido. 5. Clica em Visualizar.	6. Gera o relatório através dos dados presentes no banco de dados. 7. Apresenta em o relatório gerado.
Fluxo alternativo 1	
1. Após o item 7 do fluxo principal, clica em imprimir relatório.	2. Faz a impressão do relatório.

Fluxo de Eventos Para Visualizar Relatório

APÊNDICE III - DIAGRAMAS DE ROBUSTEZ

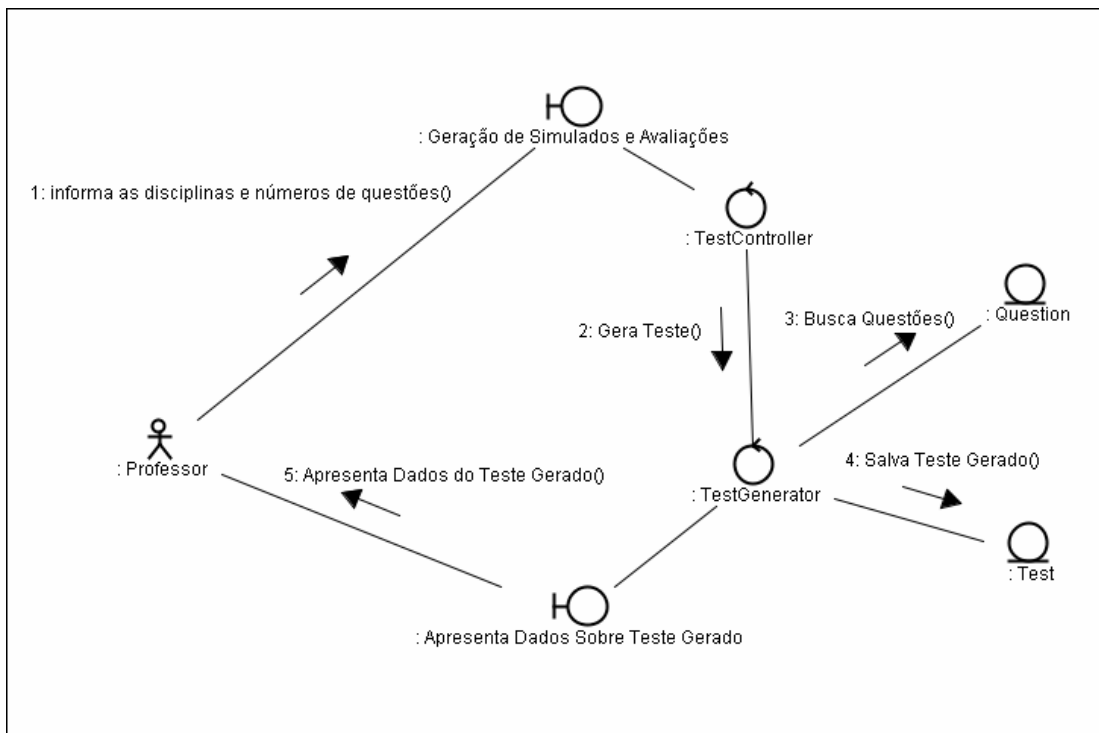


Diagrama de Robustez Para Gerar Simulados Impressos e Online

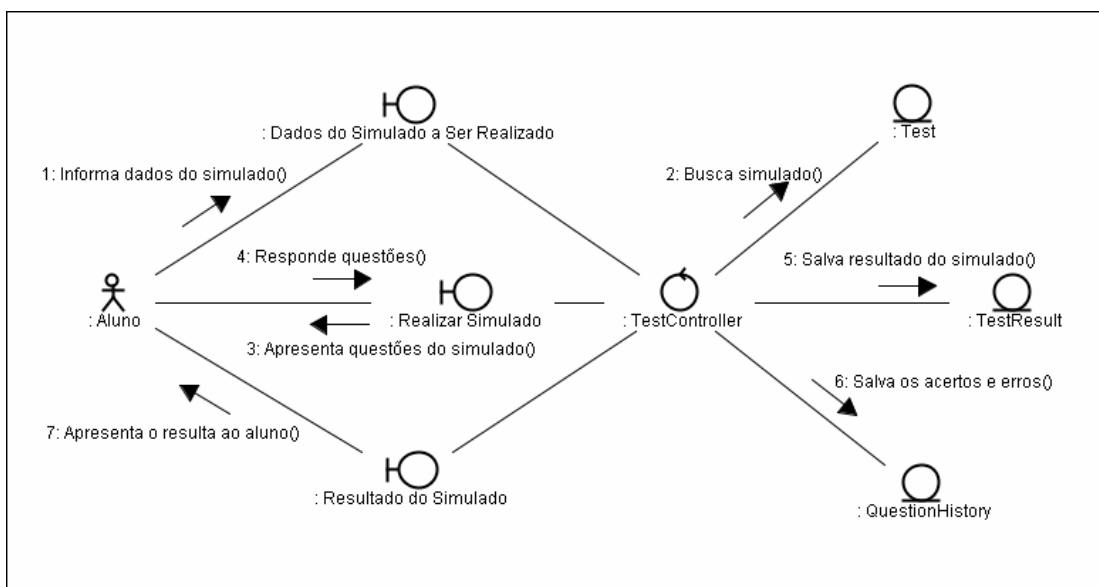


Diagrama de Robustez Para Realizar Simulado Online

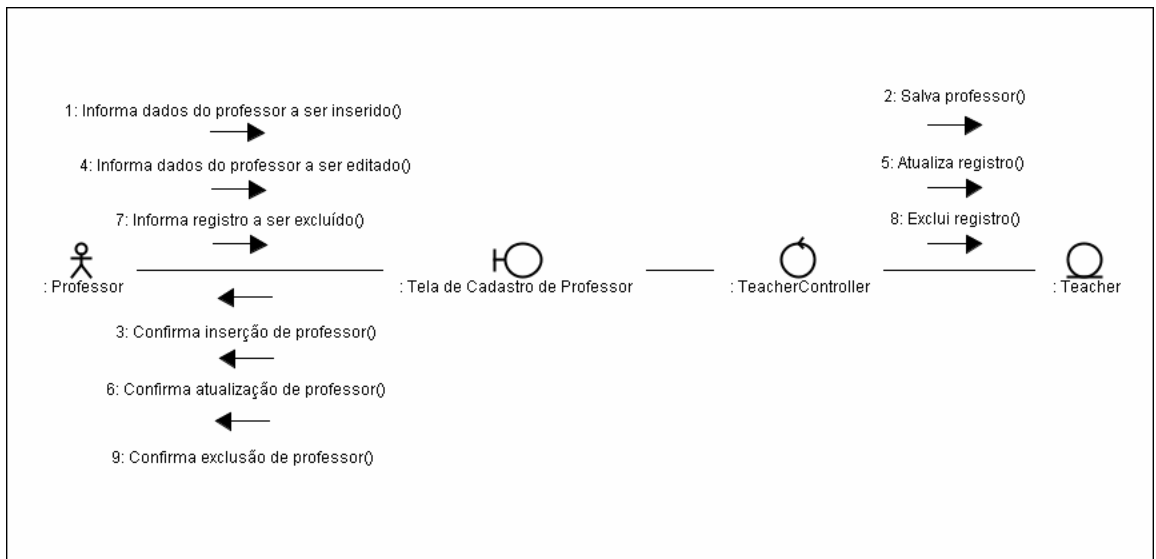


Diagrama de Robustez Para Cadastrar Professor

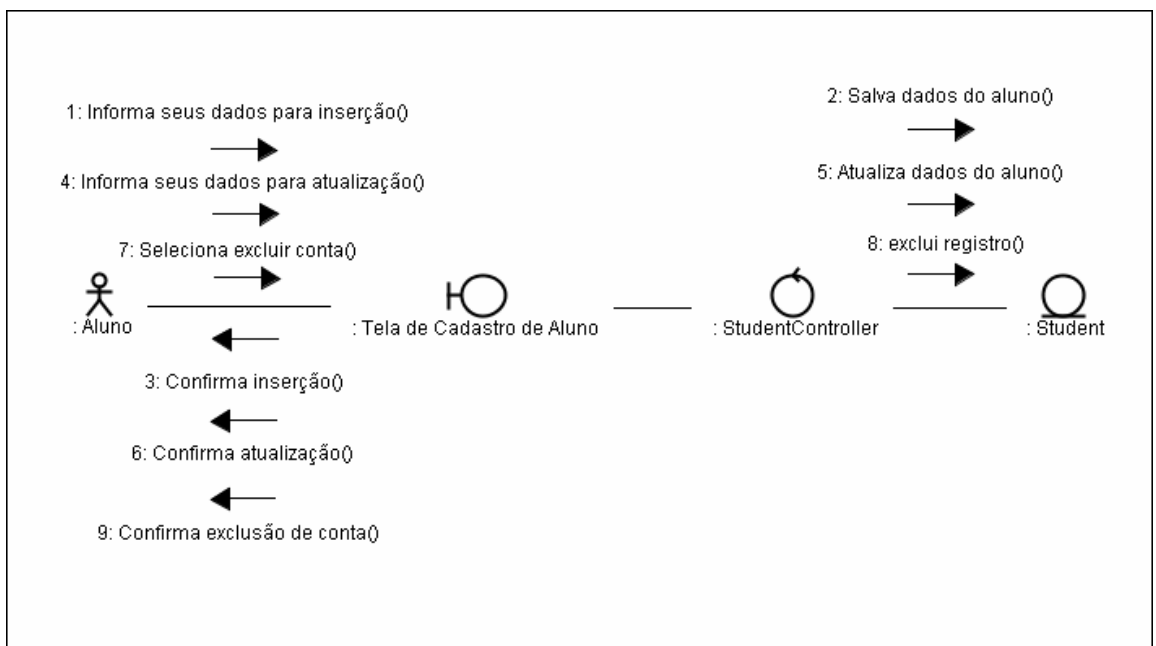


Diagrama de Robustez Para Cadastrar Aluno

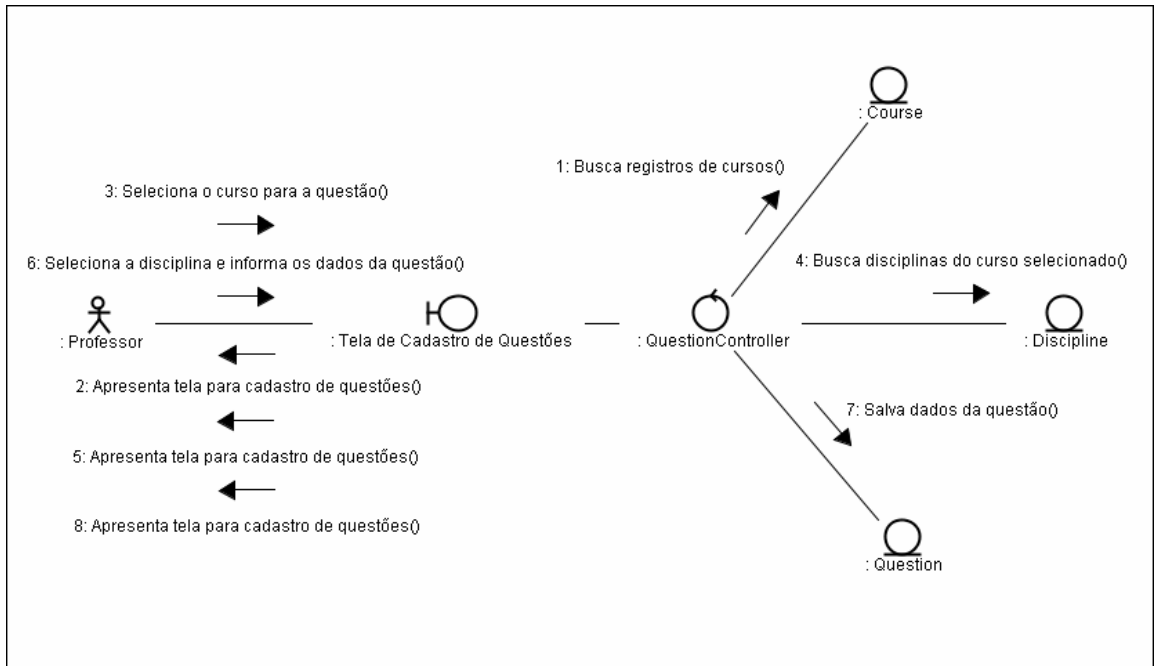


Diagrama de Robustez Para Cadastrar Questões

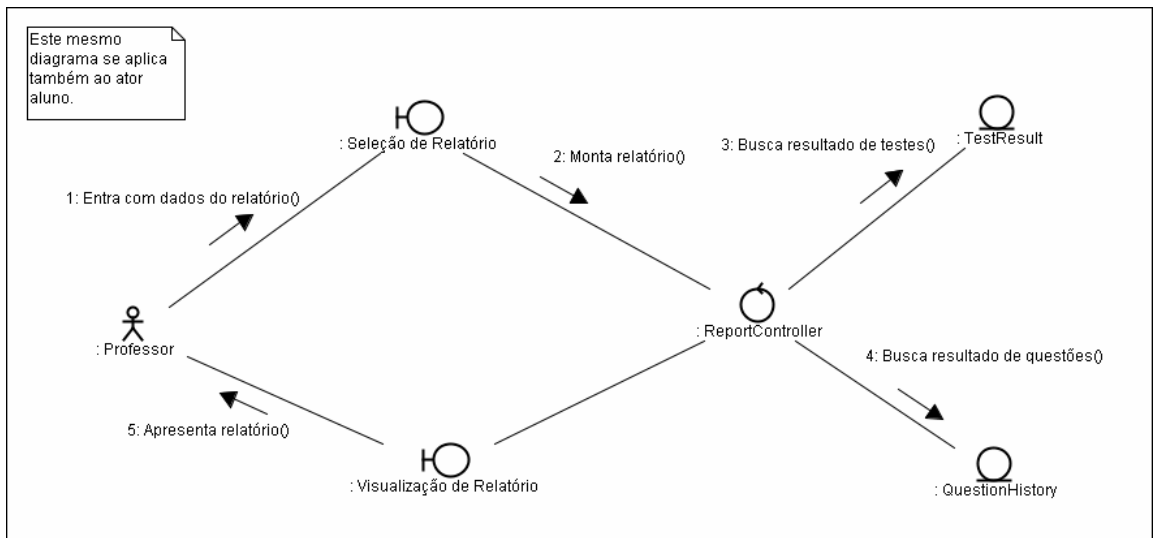


Diagrama de Robustez Para Visualizar Relatório

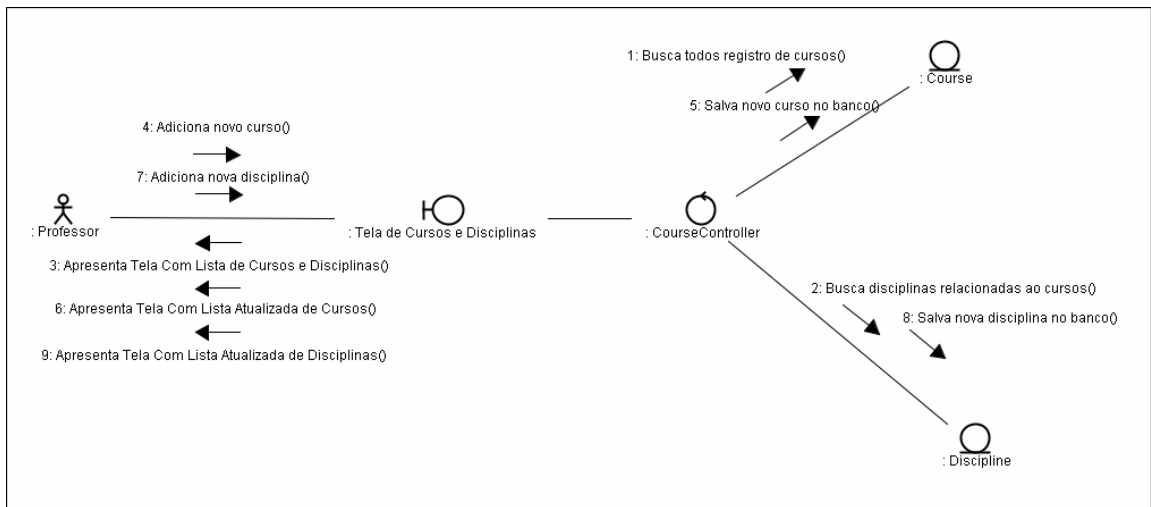


Diagrama de Robustez Para Cadastrar Cursos e Disciplinas

APÊNDICE IV - DIAGRAMAS DE SEQUÊNCIA

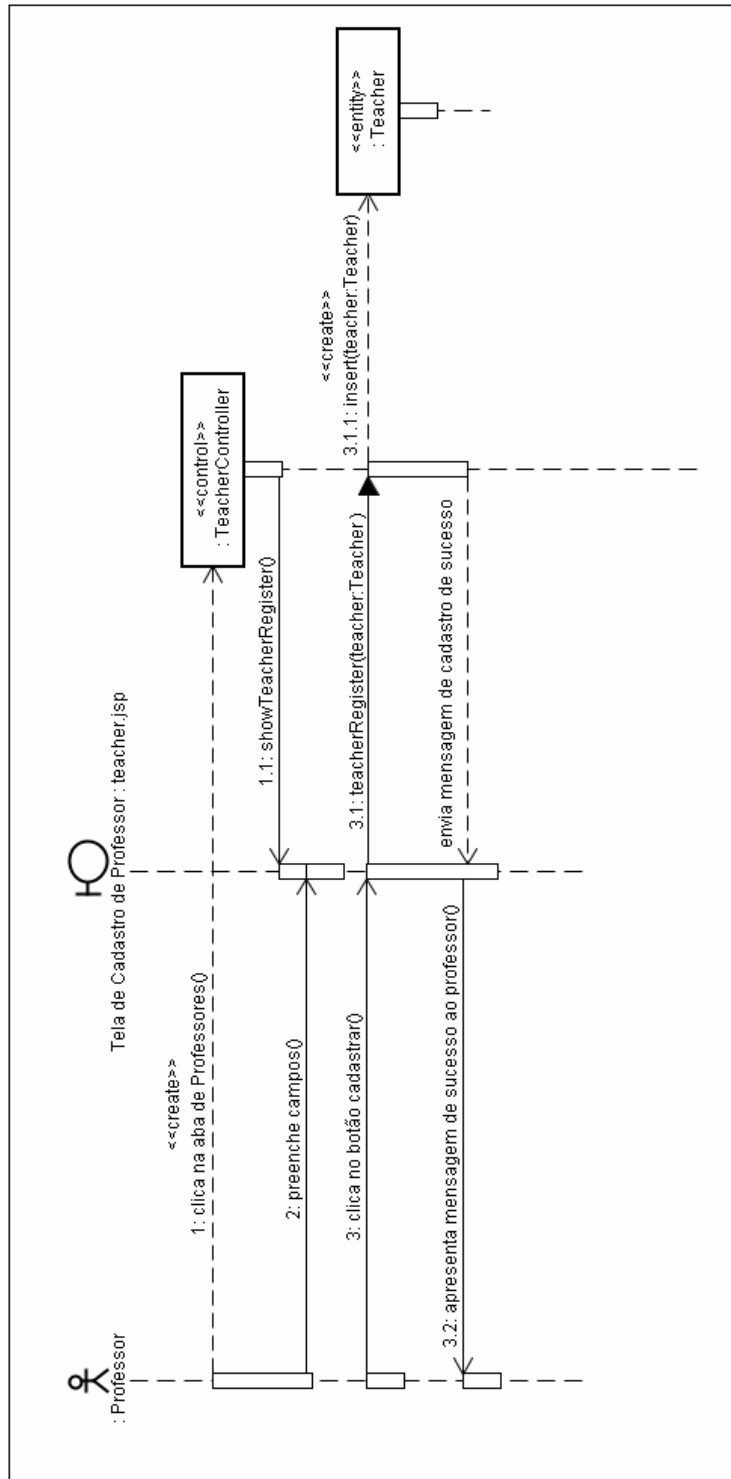


Diagrama de Sequência do Cadastro de Professor

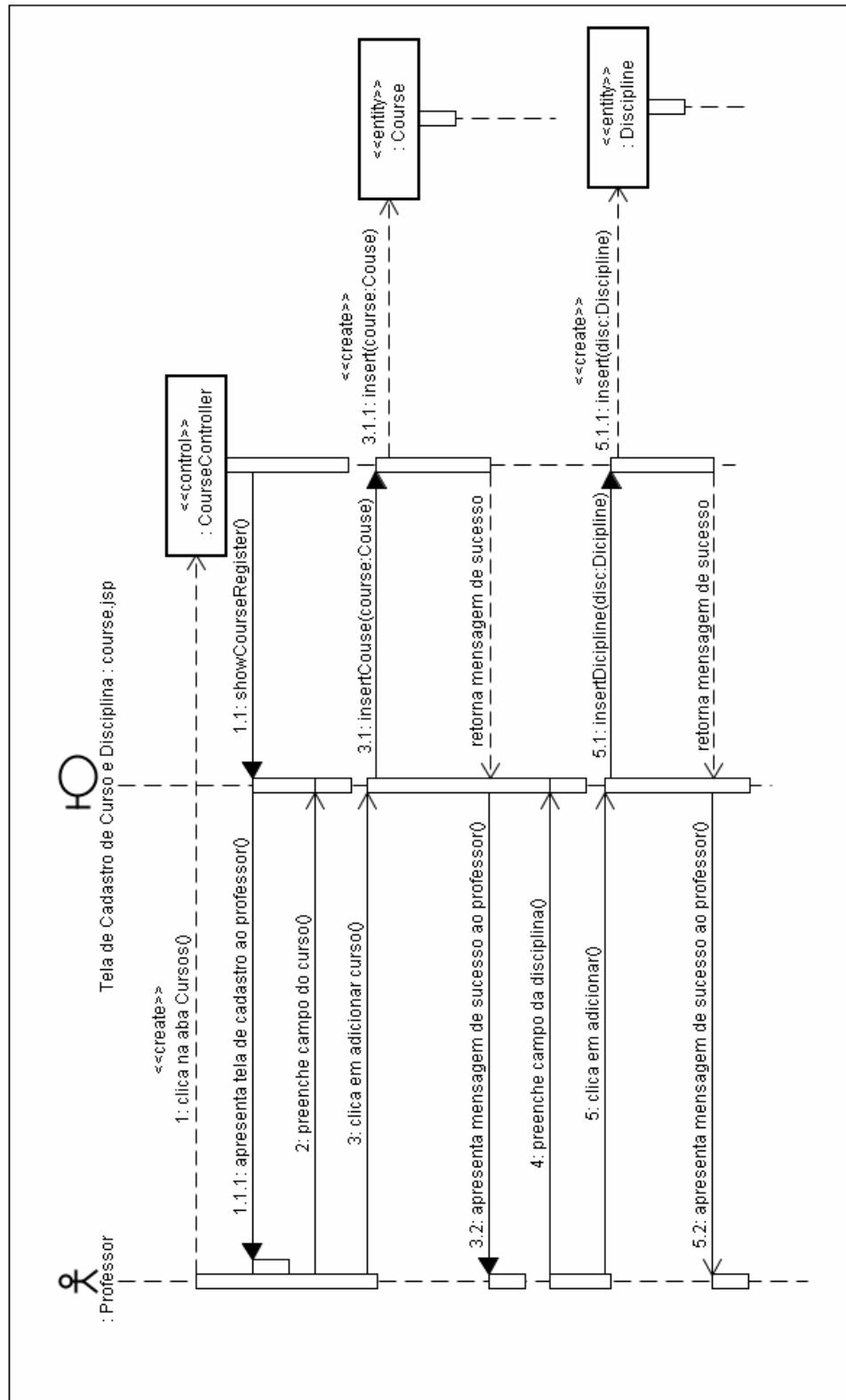


Diagrama de Sequência do Cadastro de Curso e Disciplina

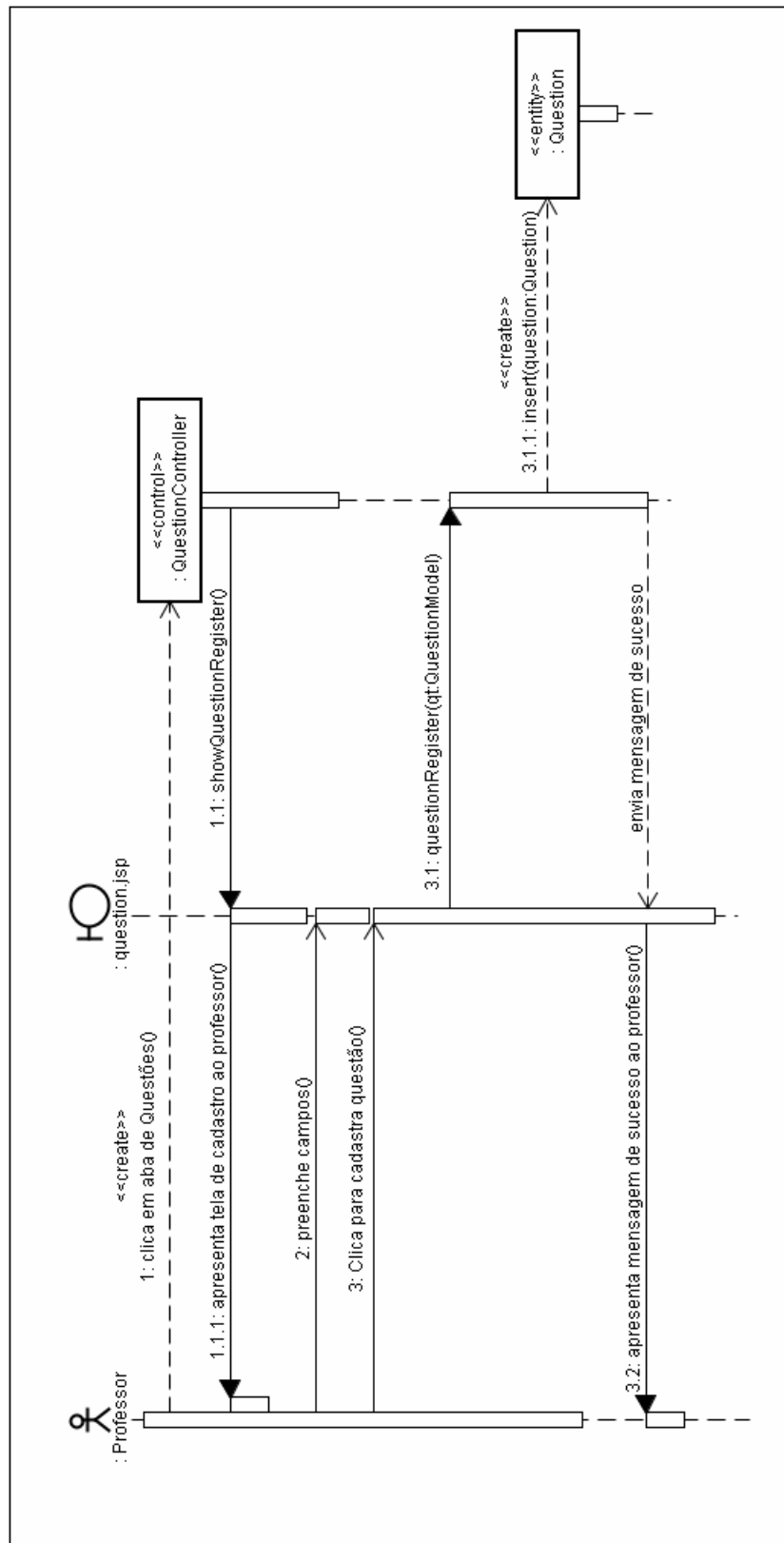


Diagrama de Sequência do Cadastro de Questões

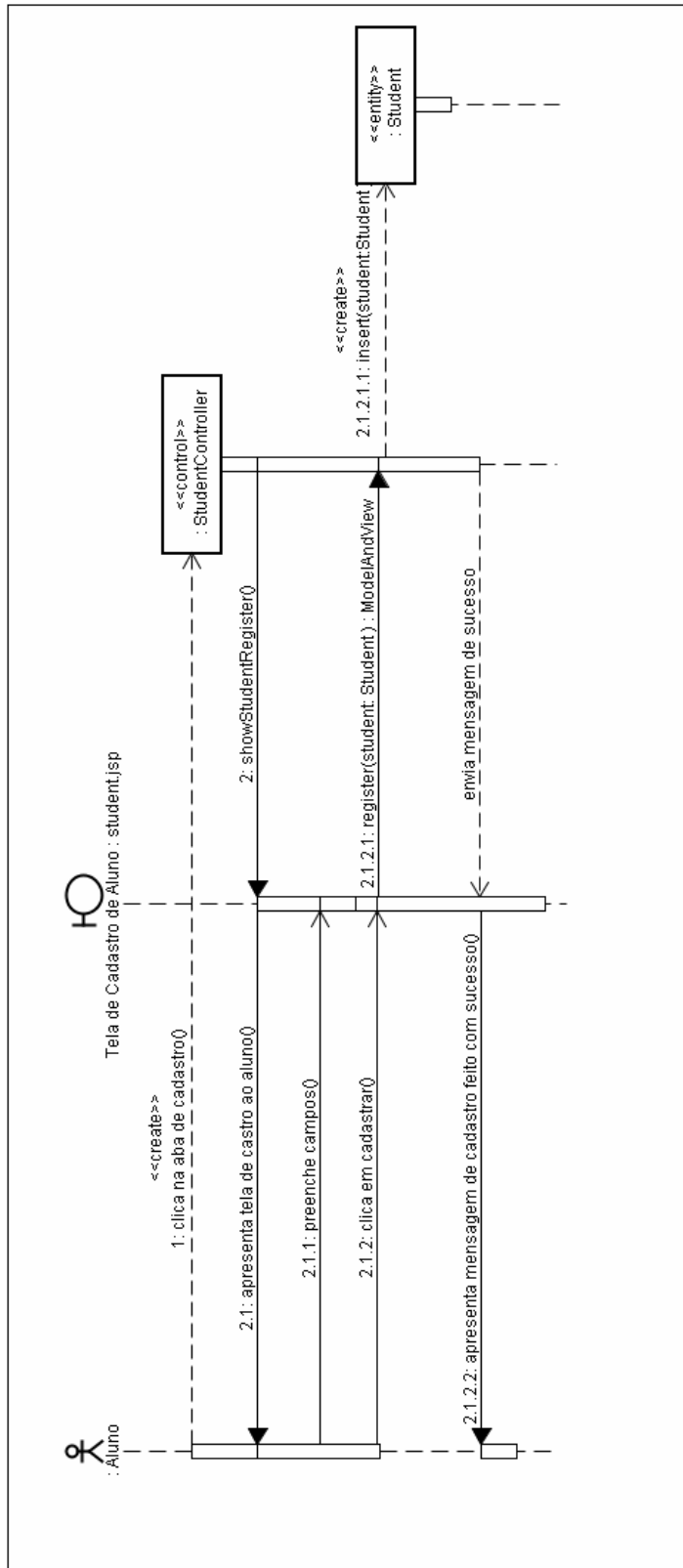


Diagrama de Sequência do Cadastro de Aluno

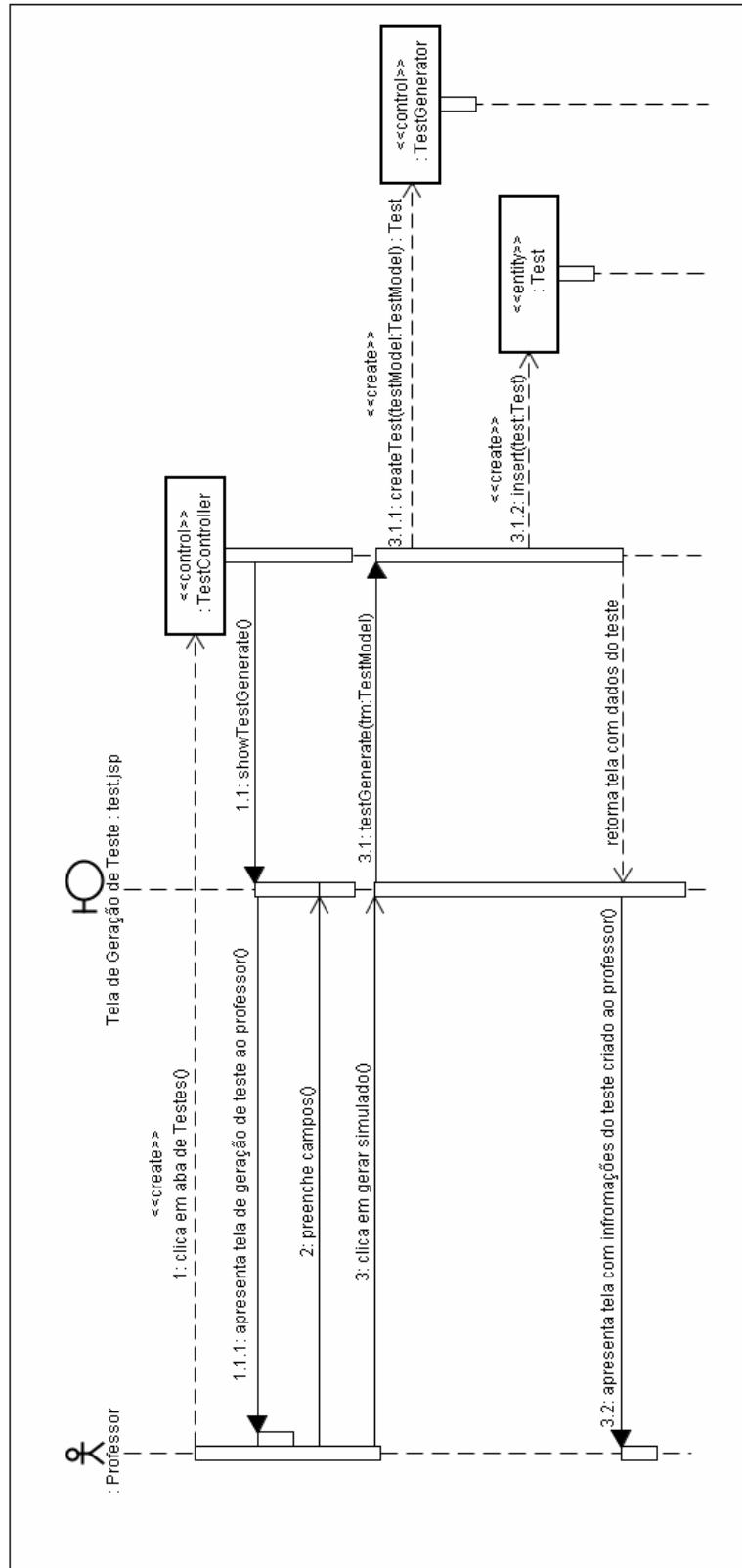


Diagrama de Sequência de Geração de Testes

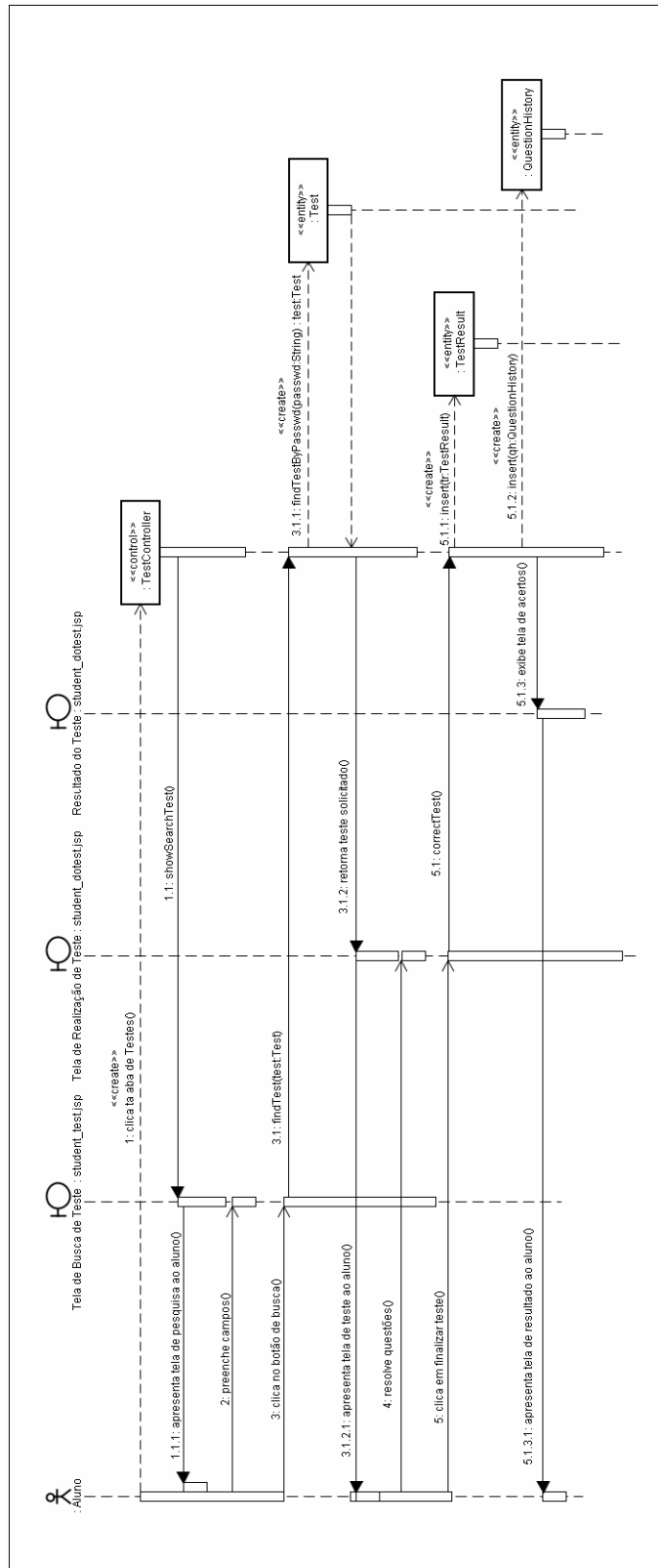


Diagrama de Sequência de Realizar Testes

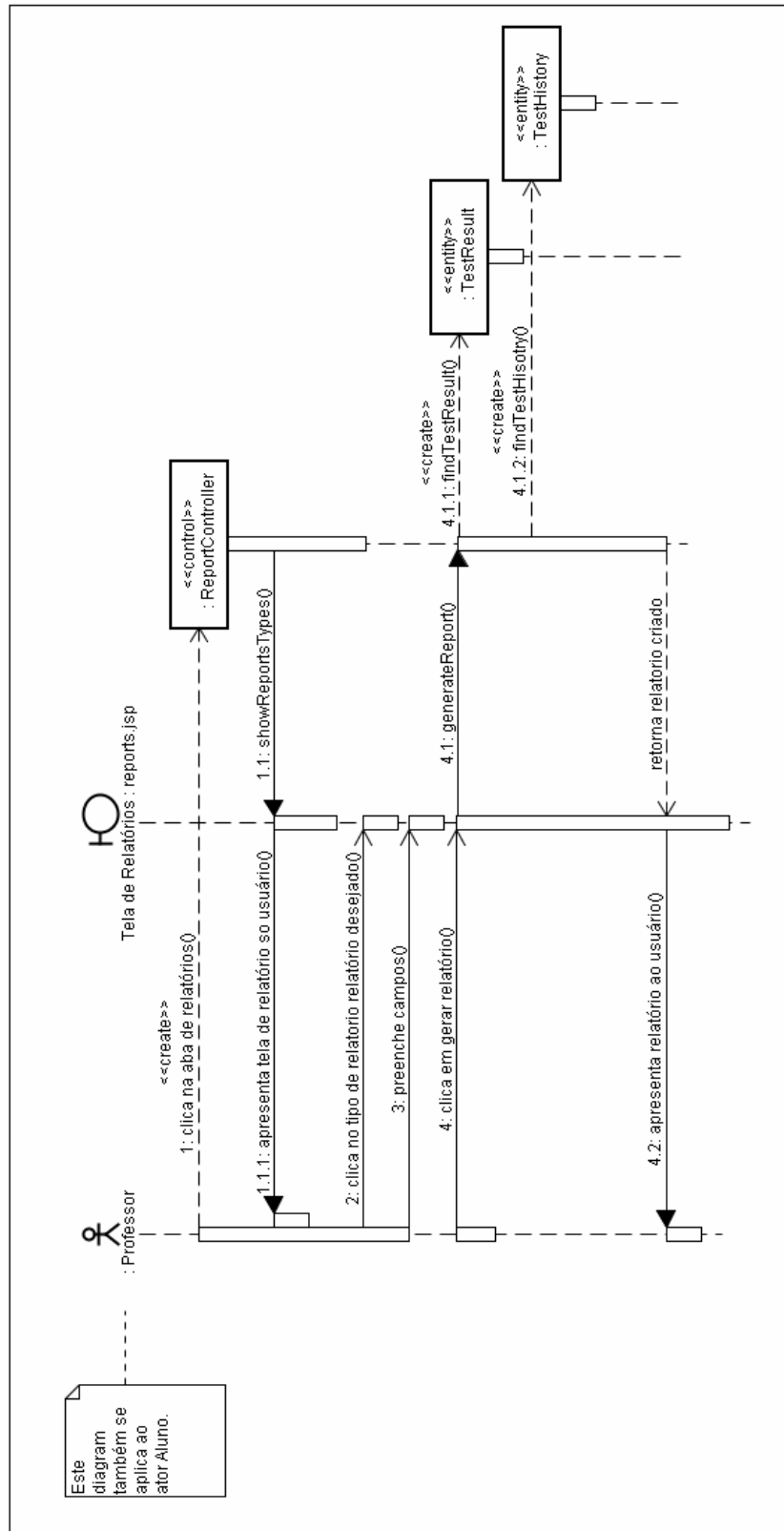


Diagrama de Sequência de Visualização de Relatório

APÊNDICE V – CASOS DE TESTE

Casos de Teste Específicos				
Objetivo do teste: Efetuar o <i>login</i> no Sistema				
Id	Tipo de teste	Descrição	Resultado esperado	Resultado obtido
E1	Funcional	Logar com usuário e senha válidos.	Mensagem de sucesso na tela.	Sucesso.
E2	Funcional	Logar um usuário e senha inválidos.	Uma mensagem de erro foi apresentada na tela.	Sucesso.

Casos de Teste Específicos				
Objetivo do teste: Realização de Simulados e Avaliações				
Id	Tipo de teste	Descrição	Resultado esperado	Resultado obtido
E1	Funcional	Logado como aluno, clica na aba Testes do menu.	A tela de realização de simulados e avaliações será apresentada na tela.	Sucesso.
E2	Funcional	Entra com código e senha da avaliação e clica em realizar teste.	As questões e alternativas da avaliação são apresentadas na tela.	Sucesso.
E3	Funcional	Escolhe as alternativas e clica em finalizar teste.	O resultado da avaliação é exibido na tela. Também são exibidos o aproveitamento e os resultados de outros testes feitos pelo aluno.	Sucesso.

Casos de Teste Específicos				
Objetivo do teste: Cadastrar Professor				
Id	Tipo de teste	Descrição	Resultado esperado	Resultado obtido
E1	Funcional	Clicar na aba Professores do menu.	Abre tela de cadastro de professores.	Sucesso.
E2	Funcional	Preencher dados do novo professor e clicar no botão cadastrar.	Mensagem de sucesso no cadastro e tabela atualizada com o novo registro.	Sucesso.
E3	Funcional	Clicar no ícone Editar de um dos registros da tabela.	Os dados do registro são carregados nos campos para atualização.	Sucesso.
E4	Funcional	Preencher os campos com os dados a serem atualizados e clicar em atualizar.	Os dados são atualizados na tela e uma mensagem de sucesso é apresentada.	Sucesso.
E5	Funcional	Clica no ícone Deletar de um dos registros da tabela.	O registro é removido da tabela e uma mensagem é apresentada na tela.	Sucesso.

Casos de Teste Específicos				
Objetivo do teste: Gerar Simulados e Avaliações				
Id	Tipo de teste	Descrição	Resultado esperado	Resultado obtido
E1	Funcional	Logado como professor, clica na aba Testes do menu principal.	A tela de geração de simulados e avaliações será apresentada na tela.	Sucesso.
E2	Funcional	Preenche os campos de nome do teste, senha. Seleciona o número de questões de cada disciplina e níveis de dificuldade. Clica em gerar.	O teste é gerado e as informações são apresentadas na tela.	Sucesso.
E3	Funcional	Clica no ícone Imprimir.	O teste em formato para impressão é exibido na tela.	Sucesso.
E4	Funcional	Clica em Imprimir.	A tela para seleção da impressora será exibida.	Sucesso.

Casos de Teste Específicos				
Objetivo do teste: Visualização de Relatórios				
Id	Tipo de teste	Descrição	Resultado esperado	Resultado obtido
E1	Funcional	Clica na aba Relatórios do menu.	A tela de seleção do tipo de relatório será apresentada.	Sucesso.
E2	Funcional	Clica em relatório por aluno.	A página de pesquisa do aluno será apresentada.	Sucesso.
E3	Funcional	Seleciona o aluno.	O relatório dos simulados e avaliações do aluno será apresentado.	Sucesso.
E4	Funcional	Na página de seleção de relatório, clica em relatório por teste.	A página de pesquisa de testes será apresentada.	Sucesso.
E5	Funcional	Seleciona o teste.	O relatório do teste é apresentado.	Sucesso.
E6	Funcional	Na página de seleção de relatório, clica em relatório de questões.	A página de pesquisa de testes será apresentada.	Sucesso.
E7	Funcional	Seleciona o teste.	Um relatório com as cinco questões mais acertadas e as cinco mais erradas é apresentado na tela.	Sucesso.

ANEXOS

ANEXO I - LEI Nº 10.861, DE 14 DE ABRIL DE 2004

Institui o Sistema Nacional de Avaliação da Educação Superior -SINAES e dá outras providências

O PRESIDENTE DA REPÚBLICA

Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:

Art. 1º Fica instituído o Sistema Nacional de Avaliação da Educação Superior SINAES, com o objetivo de assegurar processo nacional de avaliação das instituições de educação superior, dos cursos de graduação e do desempenho acadêmico de seus estudantes, nos termos do art. 9º , VI, VIII e IX, da Lei nº 9.394, de 20 de dezembro de 1996.

§ 1º O SINAES tem por finalidades a melhoria da qualidade da educação superior, a orientação da expansão da sua oferta, o aumento permanente da sua eficácia institucional e efetividade acadêmica e social e, especialmente, a promoção do aprofundamento dos compromissos e responsabilidades sociais das instituições de educação superior, por meio da valorização de sua missão pública, da promoção dos valores democráticos, do respeito à diferença e à diversidade, da afirmação da autonomia e da identidade institucional.

§ 2º O SINAES será desenvolvido em cooperação com os sistemas de ensino dos Estados e do Distrito Federal.

Art. 2º O SINAES, ao promover a avaliação de instituições, de cursos e de desempenho dos estudantes, deverá assegurar:

I - avaliação institucional, interna e externa, contemplando a análise global e integrada das dimensões, estruturas, relações, compromisso social, atividades, finalidades e responsabilidades sociais das instituições de educação superior e de seus cursos;

II - o caráter público de todos os procedimentos, dados e resultados dos processos avaliativos;

III - o respeito à identidade e à diversidade de instituições e de cursos;

IV - a participação do corpo discente, docente e técnico administrativo das instituições de educação superior, e da sociedade civil, por meio de suas representações.

Parágrafo único. Os resultados da avaliação referida no caput deste artigo constituirão referencial básico dos processos de regulação e supervisão da educação superior, neles compreendidos o credenciamento e a renovação de credenciamento de instituições de educação superior, a autorização, o reconhecimento e a renovação de reconhecimento de cursos de graduação.

Art. 3º A avaliação das instituições de educação superior terá por objetivo identificar o seu perfil e o significado de sua atuação, por meio de suas atividades, cursos, programas, projetos e setores, considerando as diferentes dimensões institucionais, dentre elas obrigatoriamente as seguintes:

I - a missão e o plano de desenvolvimento institucional;

II - a política para o ensino, a pesquisa, a pós-graduação, a extensão e as respectivas formas de operacionalização, incluídos os procedimentos para estímulo à produção acadêmica, as bolsas de pesquisa, de monitoria e demais modalidades;

III - a responsabilidade social da instituição, considerada especialmente no que se refere à sua contribuição em relação à inclusão social, ao desenvolvimento econômico e social, à defesa do meio ambiente, da memória cultural, da produção artística e do patrimônio cultural;

IV - a comunicação com a sociedade;

V - as políticas de pessoal, as carreiras do corpo docente e do corpo técnico-administrativo, seu aperfeiçoamento, desenvolvimento profissional e suas condições de trabalho;

VI - organização e gestão da instituição, especialmente o funcionamento e representatividade dos colegiados, sua independência e autonomia na relação com a mantenedora, e a participação dos segmentos da comunidade universitária nos processos decisórios;

VII - infra-estrutura física, especialmente a de ensino e de pesquisa, biblioteca, recursos de informação e comunicação;

VIII - planejamento e avaliação, especialmente os processos, resultados e eficácia da auto-avaliação institucional;

IX - políticas de atendimento aos estudantes;

X - sustentabilidade financeira, tendo em vista o significado social da continuidade dos compromissos na oferta da educação superior.

§ 1º Na avaliação das instituições, as dimensões listadas no caput deste artigo serão consideradas de modo a respeitar a diversidade e as especificidades das diferentes organizações acadêmicas, devendo ser contemplada, no caso das universidades, de acordo com critérios estabelecidos em regulamento, pontuação específica pela existência de programas de pós-graduação e por seu desempenho, conforme a avaliação mantida pela Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES.

§ 2º Para a avaliação das instituições, serão utilizados procedimentos e instrumentos diversificados, dentre os quais a autoavaliação e a avaliação externa in loco.

§ 3º A avaliação das instituições de educação superior resultará na aplicação de conceitos, ordenados em uma escala com 5 (cinco) níveis, a cada uma das dimensões e ao conjunto das dimensões avaliadas.

Art. 4º A avaliação dos cursos de graduação tem por objetivo identificar as condições de ensino oferecidas aos estudantes, em especial as relativas ao perfil do corpo docente, às instalações físicas e à organização didático-pedagógica.

§ 1º A avaliação dos cursos de graduação utilizará procedimentos e instrumentos diversificados, dentre os quais obrigatoriamente as visitas por comissões de especialistas das respectivas áreas do conhecimento.

§ 2º A avaliação dos cursos de graduação resultará na atribuição de conceitos, ordenados em uma escala com 5 (cinco) níveis, a cada uma das dimensões e ao conjunto das dimensões avaliadas.

Art. 5º A avaliação do desempenho dos estudantes dos cursos de graduação será realizada mediante aplicação do Exame Nacional de Desempenho dos Estudantes - ENADE.

§ 1º O ENADE aferirá o desempenho dos estudantes em relação aos conteúdos programáticos previstos nas diretrizes curriculares do respectivo curso de graduação, suas habilidades para ajustamento às exigências decorrentes da evolução do conhecimento e suas competências para compreender temas exteriores ao âmbito específico de sua profissão, ligados à realidade brasileira e mundial e a outras áreas do conhecimento.

§ 2º O ENADE será aplicado periodicamente, admitida a utilização de procedimentos amostrais, aos alunos de todos os cursos de graduação, ao final do primeiro e do último ano de curso.

§ 3º A periodicidade máxima de aplicação do ENADE aos estudantes de cada curso de graduação será trienal.

§ 4º A aplicação do ENADE será acompanhada de instrumento destinado a levantar o perfil dos estudantes, relevante para a compreensão de seus resultados.

§ 5º O ENADE é componente curricular obrigatório dos cursos de graduação, sendo inscrita no histórico escolar do estudante somente a sua situação regular com relação a essa obrigação, atestada pela sua efetiva participação ou, quando for o caso, dispensa oficial pelo Ministério da Educação, na forma estabelecida em regulamento.

§ 6º Será responsabilidade do dirigente da instituição de educação superior a inscrição junto ao Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira - INEP de todos os alunos habilitados à participação no ENADE.

§ 7º A não-inscrição de alunos habilitados para participação no ENADE, nos prazos estipulados pelo INEP, sujeitará a instituição à aplicação das sanções previstas no § 2º do art. 10, sem prejuízo do disposto no art. 12 desta Lei.

§ 8º A avaliação do desempenho dos alunos de cada curso no ENADE será expressa por meio de conceitos, ordenados em uma escala com 5 (cinco) níveis, tomando por base padrões mínimos estabelecidos por especialistas das diferentes áreas do conhecimento.

§ 9º Na divulgação dos resultados da avaliação é vedada a identificação nominal do resultado individual obtido pelo aluno examinado, que será a ele exclusivamente fornecido em documento específico, emitido pelo INEP.

§ 10º Aos estudantes de melhor desempenho no ENADE o Ministério da Educação concederá estímulo, na forma de bolsa de estudos, ou auxílio específico, ou ainda alguma outra forma de distinção com objetivo similar, destinado a favorecer a excelência e a continuidade dos estudos, em nível de graduação ou de pós-graduação, conforme estabelecido em regulamento.

§ 11º A introdução do ENADE, como um dos procedimentos de avaliação do SINAES, será efetuada gradativamente, cabendo ao Ministro de Estado da Educação determinar anualmente os cursos de graduação a cujos estudantes será aplicado.

Art. 6º Fica instituída, no âmbito do Ministério da Educação e vinculada ao Gabinete do Ministro de Estado, a Comissão Nacional de Avaliação da Educação Superior - CONAES, órgão colegiado de coordenação e supervisão do SINAES, com as atribuições de:

I - propor e avaliar as dinâmicas, procedimentos e mecanismos da avaliação institucional, de cursos e de desempenho dos estudantes;

II - estabelecer diretrizes para organização e designação de comissões de avaliação, analisar relatórios, elaborar pareceres e encaminhar recomendações às instâncias competentes;

III - formular propostas para o desenvolvimento das instituições de educação superior, com base nas análises e recomendações produzidas nos processos de avaliação;

IV - articular-se com os sistemas estaduais de ensino, visando a estabelecer ações e critérios comuns de avaliação e supervisão da educação superior;

V - submeter anualmente à aprovação do Ministro de Estado da Educação a relação dos cursos a cujos estudantes será aplicado o Exame Nacional de Desempenho dos Estudantes ENADE;

VI - elaborar o seu regimento, a ser aprovado em ato do Ministro de Estado da Educação;

VII - realizar reuniões ordinárias mensais e extraordinárias, sempre que convocadas pelo Ministro de Estado da Educação.

Art. 7º A CONAES terá a seguinte composição:

I - 1 (um) representante do INEP;

II - 1 (um) representante da Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES;

III - 3 (três) representantes do Ministério da Educação, sendo 1 (um) obrigatoriamente do órgão responsável pela regulação e supervisão da educação superior;

IV - 1 (um) representante do corpo discente das instituições de educação superior;

V - 1 (um) representante do corpo docente das instituições de educação superior;

VI - 1 (um) representante do corpo técnico-administrativo das instituições de educação superior;

VII - 5 (cinco) membros, indicados pelo Ministro de Estado da Educação, escolhidos entre cidadãos com notório saber científico, filosófico e artístico, e reconhecida competência em avaliação ou gestão da educação superior.

§ 1º Os membros referidos nos incisos I e II do caput deste artigo serão designados pelos titulares dos órgãos por eles representados e aqueles referidos no inciso III do caput deste artigo, pelo Ministro de Estado da Educação.

§ 2º O membro referido no inciso IV do caput deste artigo será nomeado pelo Presidente da República para mandato de 2 (dois) anos, vedada a recondução.

§ 3º Os membros referidos nos incisos V a VII do caput deste artigo serão nomeados pelo Presidente da República para mandato de 3 (três) anos, admitida 1 (uma) recondução, observado o disposto no parágrafo único do art. 13 desta Lei.

§ 4º A CONAES será presidida por 1 (um) dos membros referidos no inciso VII do caput deste artigo, eleito pelo colegiado, para mandato de 1 (um) ano, permitida 1 (uma) recondução.

§ 5º As instituições de educação superior deverão abonar as faltas do estudante que, em decorrência da designação de que trata o inciso IV do caput deste artigo, tenha participado de reuniões da CONAES em horário coincidente com as atividades acadêmicas.

§ 6º Os membros da CONAES exercem função não remunerada de interesse público relevante, com precedência sobre quaisquer outros cargos públicos de que sejam titulares e, quando convocados, farão jus a transporte e diárias.

Art. 8º A realização da avaliação das instituições, dos cursos e do desempenho dos estudantes será responsabilidade do INEP.

Art. 9º O Ministério da Educação tornará público e disponível o resultado da avaliação das instituições de ensino superior e de seus cursos.

Art. 10. Os resultados considerados insatisfatórios ensejarão a celebração de protocolo de compromisso, a ser firmado entre a instituição de educação superior e o Ministério da Educação, que deverá conter:

I - o diagnóstico objetivo das condições da instituição;

II - os encaminhamentos, processos e ações a serem adotados pela instituição de educação superior com vistas na superação das dificuldades detectadas;

III - a indicação de prazos e metas para o cumprimento de ações, expressamente definidas, e a caracterização das respectivas responsabilidades dos dirigentes;

IV - a criação, por parte da instituição de educação superior, de comissão de acompanhamento do protocolo de compromisso.

§ 1º O protocolo a que se refere o caput deste artigo será público e estará disponível a todos os interessados.

§ 2º O descumprimento do protocolo de compromisso, no todo ou em parte, poderá ensejar a aplicação das seguintes penalidades:

I - suspensão temporária da abertura de processo seletivo de cursos de graduação;

II - cassação da autorização de funcionamento da instituição de educação superior ou do reconhecimento de cursos por ela oferecidos;

III - advertência, suspensão ou perda de mandato do dirigente responsável pela ação não executada, no caso de instituições públicas de ensino superior.

§ 3º As penalidades previstas neste artigo serão aplicadas pelo órgão do Ministério da Educação responsável pela regulação e supervisão da educação superior, ouvida a Câmara de Educação Superior, do Conselho Nacional de Educação, em processo administrativo próprio, ficando assegurado o direito de ampla defesa e do contraditório.

§ 4º Da decisão referida no § 2º deste artigo caberá recurso dirigido ao Ministro de Estado da Educação.

§ 5º O prazo de suspensão da abertura de processo seletivo de cursos será definido em ato próprio do órgão do Ministério da Educação referido no § 3º deste artigo.

Art. 11. Cada instituição de ensino superior, pública ou privada, constituirá Comissão Própria de Avaliação - CPA, no prazo de 60 (sessenta) dias, a contar da publicação desta Lei, com as atribuições de condução dos processos de avaliação internos da instituição, de sistematização e de prestação das informações solicitadas pelo INEP, obedecidas as seguintes diretrizes:

I - constituição por ato do dirigente máximo da instituição de ensino superior, ou por previsão no seu próprio estatuto ou regimento, assegurada a participação de todos os segmentos da comunidade universitária e da sociedade civil organizada, e vedada a composição que privilegie a maioria absoluta de um dos segmentos;

II - atuação autônoma em relação a conselhos e demais órgãos colegiados existentes na instituição de educação superior.

Art. 12. Os responsáveis pela prestação de informações falsas ou pelo preenchimento de formulários e relatórios de avaliação que impliquem omissão ou distorção de dados a serem fornecidos ao SINAES responderão civil, penal e administrativamente por essas condutas.

Art. 13. A CONAES será instalada no prazo de 60 (sessenta) dias a contar da publicação desta Lei.

Parágrafo único. Quando da constituição da CONAES, 2 (dois) dos membros referidos no inciso VII do caput do art. 7º desta Lei serão nomeados para mandato de 2 (dois) anos.

Art. 14. O Ministro de Estado da Educação regulamentará os procedimentos de avaliação do SINAES.

Art. 15. Esta Lei entra em vigor na data de sua publicação.

Art. 16. Revogam-se a alínea a do § 2º do art. 9º da Lei nº 4.024, de 20 de dezembro de 1961, e os arts. 3º e 4º da Lei nº 9.131, de 24 de novembro de 1995.

Brasília, 14 de abril de 2004; 183º da Independência e 116º da República.

LUIZ INÁCIO LULA DA SILVA

Tarso Genro

(DOU de 15/04/2004 - Seção - p.3)

TAVARES, Luis Antonio. TRIPOLI, Crislaine da Silva.

SAS / Sistema de Avaliações e Simulados – Pouso Alegre:
UNIVÁS, 2010. 76f.

Trabalho de conclusão de curso – Universidade do Vale do
Sapucaí, UNIVÁS, Curso de Sistemas de Informação, 2010.

1. Avaliação.
 2. Simulado.
 3. Ferramenta.
 4. Spring.
 5. Java.
 6. ICONIX.
- I. SAS – Sistema de Avaliações e Simulados.