UFES - Universidade Federal do Espírito Santo

APLICAÇÃO E ANÁLISE DO MÉTODO FRAMEWEB COM DIFERENTES *FRAMEWORKS WEB*.

Projeto Final de Graduação

Leonardo Araújo Peruch

VITÓRIA, ES
AGOSTO DE 2007

Aplicação e Análise do Método FrameWeb com Diferentes *Frameworks Web*.

Leonardo Araújo Peruch

MONOGRAFIA APRESENTADA COMO EXIGÊNCIA PARCIAL PARA OBTENÇÃO DO TÍTULO DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO À UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO, NA ÁREA DE CONCENTRAÇÃO ENGENHARIA DE SOFTWARE, SOB A ORIENTAÇÃO DO PROF. GIANCARLO GUIZZARDI E CO-ORIENTAÇÃO DO PROF. VÍTOR ESTÊVÃO SILVA SOUZA.

Aprovada por:

Prof. Giancarlo Guizzardi, Ph.D. (Orientador)

Prof. Vítor Estêvão Silva Souza, M.Sc. (Co-orientador)

Prof. Ricardo de Almeida Falbo, D.Sc.

Prof. Davidson Cury, D.Sc.

AGRADECIMENTOS

Agradeço a Deus pelo apoio espiritual em todos os momentos e por permitir a realização deste sonho.

A minha família, o meu muito obrigado por tudo! Sem a sua presença, eu não teria o suporte para chegar até aqui. Esta conquista é nossa!

Agradeço a minha namorada Camila Petarli, pelo amor e compreensão.

Ao Giancarlo e Vítor Souza, meus orientadores, meus profundos agradecimentos por este período de crescimento pessoal e "lapidação" profissional, no qual eu encontrei o meu caminho de realização.

A meus amigos, eu agradeço pela amizade e pelos inesquecíveis momentos de descontração. A meus amigos da UFES, também agradeço pela ajuda técnica.

E a todos aqueles que direta ou indiretamente contribuíram para a realização deste trabalho.

LEONARDO ARAÚJO PERUCH

RESUMO

A acelerada evolução na área da Engenharia *Web* tem motivado propostas de diversos métodos e *frameworks* para o desenvolvimento de aplicações *Web* (*WebApps*). Assim, o método FrameWeb (*Framework-based Design Method for Web Engineering*) propõe uma abordagem baseada em *frameworks* para o desenvolvimento de Sistemas de Informação *Web (Web-based Information Systems – WISs).*

O método propõe uma arquitetura básica para desenvolvimento de WISs, um conjunto de atividades e um perfil UML para quatro tipos de modelos de projeto que trazem conceitos utilizados por algumas categorias de *frameworks*. A idéia é que o uso de FrameWeb favoreça um aumento da produtividade da equipe de desenvolvimento por utilizar uma linguagem de modelagem que permite aos projetistas produzir diagramas que representam conceitos dos *frameworks* e aos desenvolvedores por diretamente traduzir esses diagramas para código.

A proposta de FrameWeb considera algumas categorias de *frameworks*. Dentro dessas categorias existem diversas instâncias de *frameworks* que podem ser utilizadas em determinado projeto de software. Na proposta original de FrameWeb não foram feitos experimentos com diferentes instâncias de *frameworks*. Este trabalho propõe esta experiência, realizando a aplicação e análise de FrameWeb com diferentes *frameworks Web*, sugerindo melhorias nos modelos, quando necessário, de acordo com o *framework Web* utilizado.

ÍNDICE

CAPÍTULO 1 INTRODUÇÃO 9

1.1. OBJETIVOS 10
1.2. METODOLOGIA 10
1.3. ORGANIZAÇÃO DA MONOGRAFIA 11

CAPÍTULO 2 ENGENHARIA *WEB*, *FRAMEWORKS* E FRAMEWEB 13

2.1. ENGENHARIA *WEB* 13
2.2. *FRAMEWORKS* 16
2.2.1. Frameworks MVC (Front Controller) 18
2.2.2. Frameworks RIA (Rich Internet Applications) 20
2.2.3. Frameworks de Decoração 21
2.2.4. Frameworks de Mapeamento Objeto/Relacional 22
2.2.5. Frameworks de Injeção de Dependência 23
2.2.6. Outros Frameworks 24
2.3. FRAMEWEB 25

CAPÍTULO 3 ESPECIFICAÇÃO DE REQUISITOS DO PORTAL DO LABES 30

3.1. DESCRIÇÃO DE ESCOPO 30
3.2. MODELO DE CASOS DE USO 31
3.2.1. Pacote Controle de Usuário 33
3.2.1.1 Caso de Uso Cadastrar Tipo de usuário 33
3.2.1.2 Caso de Uso Cadastrar Funcionalidade 34
3.2.1.3 Caso de Uso Cadastrar Área 35
3.2.1.4 Caso de Uso Cadastrar Usuário 36
3.2.1.5 Caso de Uso Autenticar Usuário 37
3.2.2. Pacote Controle de Itens 38
3.2.2.1 Caso de Uso Cadastrar Projeto 39
3.2.2.2 Caso de Uso Cadastrar Material 39
3.2.2.3 Caso de Uso Efetuar Busca de Itens 40
3.2.2.4 Caso de Uso Consultar Item 41

CAPÍTULO 4 ANÁLISE DO PORTAL DO LABES 42

4.1. MODELAGEM DE CLASSES 42
4.2. PACOTE CONTROLE DE USUÁRIOS 43
4.3. PACOTE CONTROLE DE ITENS 43

CAPÍTULO 5 PROJETO E IMPLEMENTAÇÃO DO PORTAL DO LABES 45

5.1. ARQUITETURA DO SISTEMA 45
5.1.1. Tecnologia 46
5.1.2. Pacotes 47
5.2. UTILITÁRIOS 48
5.2.1. Pacote net.java.dev.esjug.util.persistence e net.java.dev.esjug.util.domain 48
5.2.2. Pacote net.java.dev.esjug.frameworks.crud 50

5.3. MODELO DE DOMÍNIO 52
5.3.1. Pacote br.ufes.inf.labes.portal.controleusuario.dominio *53*
5.3.2. Pacote br.ufes.inf.labes.portal.controleitens.dominio *54*
5.4. MODELO DE PERSISTÊNCIA 55
5.4.1. Pacote br.ufes.inf.labes.portal.controleusuario.persistencia *55*
5.4.2. Pacote br.ufes.inf.labes.portal.controleitens.persistencia *56*
5.5. MODELO DE NAVEGAÇÃO 57
5.5.1. Pacote br.ufes.inf.labes.portal.controleusuario.controlador *58*
5.5.2. Pacote br.ufes.inf.labes.portal.controleitens.controlador *62*
5.6. MODELO DE APLICAÇÃO 63
5.6.1. Pacote br.ufes.inf.labes.portal.controleusuario.aplicacao *64*
5.6.2. Pacote br.ufes.inf.labes.portal.controleitens.aplicacao *65*
5.7. IMPLEMENTAÇÃO 66

CAPÍTULO 6 PROPOSTAS DE MELHORIAS PARA O MÉTODO FRAMEWEB 73

6.1. PROJETO E IMPLEMENTAÇÃO COM *FRAMEWORK* STRUTS[1] 73
6.1.1. Arquitetura do Sistema *74*
6.1.1.1 *Framework* Struts[1] 74
6.1.1.2 JSP 76
6.1.1.3 Pacotes 76
6.1.2. Utilitários *77*
6.1.3. Propostas para o método FrameWeb a partir de uma experiência com o framework Struts[1] *77*
6.1.4. Implementação *80*
6.2. PROJETO E IMPLEMENTAÇÃO COM *FRAMEWORK* ZK 80
6.2.1. Arquitetura do Sistema *80*
6.2.1.1 *Framework* ZK 80
6.2.1.2 Pacotes 84
6.2.2. Utilitários *84*
6.2.3. Propostas para o método FrameWeb a partir de uma experiência com o framework ZK *85*
6.2.4. Implementação *87*

CAPÍTULO 7 CONSIDERAÇÕES FINAIS 96

7.1. CONCLUSÕES 96
7.2. TRABALHOS FUTUROS 98

LISTA DE FIGURAS

FIGURA 1 APLICAÇÃO DESENVOLVIDA UTILIZANDO UM *FRAMEWORK* (TALIGENT, 1995). 18

FIGURA 2 FUNCIONAMENTO DO CONTROLADOR FRONTAL. 20

FIGURA 3 FUNCIONAMENTO DO *FRAMEWORK* DE DECORAÇÃO. 21

FIGURA 4 COMPORTAMENTO DO *FRAMEWORK* DE MAPEAMENTO OBJETO/RELACIONAL. 23

FIGURA 5 FUNCIONAMENTO DO *FRAMEWORK* DE INJEÇÃO DE DEPENDÊNCIA..... 24

FIGURA 6 ARQUITETURA BÁSICA PARA DESENVOLVIMENTO DE *WEBAPPS* (SOUZA ET AL., 2007). 26

FIGURA 7 TIPOS DE USUÁRIOS DO PORTAL LABES (FIORI, 2005). 31

FIGURA 8 DIAGRAMA DE PACOTES (FIORI, 2005)..... 33

FIGURA 9 CASO DE USO DO PACOTE CONTROLE DE USUÁRIO (FIORI, 2005). 33

FIGURA 10 CASO DE USO DO PACOTE CONTROLE DE ITENS..... 38

FIGURA 11 DIAGRAMA DE PACOTES (FIORI, 2005). 42

FIGURA 12 DIAGRAMA DE CLASSES DO PACOTE CONTROLE DE USUÁRIOS (SOUZA, 2007). 43

FIGURA 13 DIAGRAMA DE CLASSES DO PACOTE CONTROLE DE ITENS. 44

FIGURA 14 PACOTES DO PORTAL DO LABES DESENVOLVIDO COM O *FRAMEWORK* STRUTS². 48

FIGURA 15 DIAGRAMA DE CLASSES DO PACOTE UTILITÁRIO DE PERSISTÊNCIA (SOUZA, 2007). 48

FIGURA 16 DIAGRAMA DE CLASSES DO *FRAMEWORK* DE CRUD (SOUZA, 2007). 51

FIGURA 17 MODELO DE DOMÍNIO DO PACOTE CONTROLEUSUARIO (SOUZA, 2007). 53

FIGURA 18 MODELO DE DOMÍNIO DO PACOTE CONTROLEITENS. 54

FIGURA 19 MODELO DE PERSISTÊNCIA DO PACOTE CONTROLEUSUARIO. 55

FIGURA 20 MODELO DE PERSISTÊNCIA DO PACOTE CONTROLEITENS. 57

FIGURA 21 MODELO DE NAVEGAÇÃO PARA O CASO DE USO "AUTENTICAR USUÁRIO"(SOUZA, 2007)..... 59

FIGURA 22 MODELO DE NAVEGAÇÃO PARA O CASO DE USO "CADASTRAR USUÁRIO" (SOUZA, 2007)..... 61

FIGURA 23 MODELO DE NAVEGAÇÃO PARA O CASO DE USO "EFETUAR BUSCA DE ITENS". 63

FIGURA 24 MODELO DE APLICAÇÃO DO PACOTE CONTROLEUSUARIO. 65

FIGURA 25 MODELO DE APLICAÇÃO DO PACOTE CONTROLEITENS. 66

FIGURA 26 TELA INICIAL DO PORTAL DO LABES..... 67

FIGURA 27 TELA DE CADASTRO DE USUÁRIO DO PORTAL DO LABES. 68

FIGURA 28 TELA DE EDIÇÃO DE USUÁRIO DO PORTAL DO LABES. 69

FIGURA 29 TELA DE EXCLUSÃO DE USUÁRIO DO PORTAL DO LABES. 70

FIGURA 30 TELA PARA BUSCA DE ITENS DO PORTAL DO LABES. 71

FIGURA 31 TELA DE RESULTADO DA BUSCA DE ITENS DO PORTAL DO LABES. 72

FIGURA 32 FUNCIONAMENTO DO STRUTS[1] (SPIELMAN, 2003). 75

FIGURA 33 PACOTES DO PORTAL DO LABES DESENVOLVIDO COM O *FRAMEWORK* STRUTS[1]. 77

FIGURA 34 MODELO DE NAVEGAÇÃO PARA O CASO DE USO "AUTENTICAR USUARIO" PARA STRUTS[1]. 79

FIGURA 35 PÁGINA HELLO WORLD CONSTRUÍDA COM O *FRAMEWORK* ZK. 82

FIGURA 36 ARQUITETURA DO *FRAMEWORK* ZK. 82

FIGURA 37 PACOTES DO PORTAL DO LABES DESENVOLVIDO COM O *FRAMEWORK* ZK. 84

FIGURA 38 MODELO DE NAVEGAÇÃO PARA O CASO DE USO "AUTENTICAR USUARIO" PARA ZK 86

FIGURA 39 TELA INICIAL DO PORTAL DO LABES. 88

FIGURA 40 TELA DE CADASTRO DE USUÁRIO DO PORTAL DO LABES. 89

FIGURA 41 TELA DE EDIÇÃO DE USUÁRIO DO PORTAL DO LABES. 90

FIGURA 42 TELA DE EXCLUSÃO DE USUÁRIO DO PORTAL DO LABES. 91

FIGURA 43 TELA PARA BUSCA DE ITENS DO PORTAL DO LABES. 92

FIGURA 44 TELA DE RESULTADO DA BUSCA DE ITENS DO PORTAL DO LABES. 93

FIGURA 45 TELA DE DETALHES DO MATERIAL DO PORTAL DO LABES. 94

FIGURA 46 TELA DE DETALHES DO PROJETO DO PORTAL DO LABES. 95

Capítulo 1

Introdução

A Internet é uma ferramenta poderosa, com linguagens intuitivas e acessíveis, que possibilita vantagens imediatas a quem a adota. Conecta-nos a um mundo virtual de informações, de produtos, de serviços e até de oportunidades para emprego antes nunca vistas. Entre os serviços oferecidos pela Internet destacamos a *World Wide Web*, WWW, ou simplesmente *Web*. Com o serviço *Web* acessamos textos, sons e imagens publicados em outros computadores conectados, além de informações de páginas pessoais e de empresas, sem esquecer alguns serviços populares, entre eles bate-papo (*chat*), *Home-banking* e compras *on-line* (*e-commerce*).

O volume de informação é tão grande que temos, na maioria das vezes, a facilidade de encontrar tudo na hora e no lugar que quisermos por meio da Internet. Esta tecnologia transformou a vida das pessoas e ganha a cada dia ainda mais usuários.

O fato é que a Internet está cada vez mais presente na vida e no trabalho das pessoas. Com isso há uma demanda maior por rapidez e qualidade dos sistemas que executam nesta plataforma. Como conseqüência, surge uma necessidade de abordagens disciplinadas de Engenharia de Software, novos métodos e ferramentas para o desenvolvimento, a implantação e a avaliação de sistema e aplicações baseadas na *Web* (*WebApps* - *Web Applications*). Tais abordagens e técnicas devem levar em conta as características especiais do novo meio, os ambientes e os cenários operacionais e a multiplicidade de perfis de usuários, que adicionam desafios ao desenvolvimento de aplicações *Web*. Nasceria, assim, a Engenharia *Web* (PRESSMAN, 2005).

Com objetivo de contribuir para uma abordagem disciplinada de construção de sistemas para *Web*, foi proposto o método FrameWeb (*Framework-based Design Method for Web Engineering*) (SOUZA, 2007). Essa metodologia de desenvolvimento sugere a utilização de vários *frameworks* para uma rápida e eficiente construção da *WebApp*. Além disso, ela nos fornece recomendações para aumentar a agilidade nas principais fases de

desenvolvimento (requisitos, análise, projeto e outras), uma arquitetura básica para desenvolver esse tipo de aplicação e um perfil UML (extensão da linguagem UML) para a construção de quatro tipos de modelos na fase de projeto. Esses modelos trazem conceitos usados por algumas categorias de *frameworks*.

A proposta de FrameWeb considera algumas categorias de *frameworks*. Dentro dessas categorias, existem diversas instâncias de *frameworks* que podem ser utilizadas em determinado projeto de software. Na proposta original de FrameWeb não foram feitos experimentos com diferentes instâncias de *frameworks*. Este trabalho propõe esta experiência, realizando a aplicação e análise de FrameWeb com diferentes *frameworks Web*, sugerindo melhorias nos modelos, quando necessário, de acordo com o *framework Web* utilizado.

1.1. Objetivos

É objetivo deste trabalho fazer uma aplicação e análise do método FrameWeb com diferentes *frameworks Web*. Podemos subdividir esses objetivos em três sub-objetivos específicos, a saber:

- Seguir o processo proposto pelo método FrameWeb (SOUZA, 2007) no desenvolvimento do Portal do LabES (Portal do Laboratório de Engenharia de Software), utilizando os *frameworks* nos quais o método foi inicialmente testado;
- Desenvolver o Portal do LabES em duas novas versões, utilizando diferentes *frameworks Web*, com o intuito de analisar a eficácia do método em diferentes contextos;
- Sugerir propostas para o método FrameWeb baseadas na experiência adquirida com o uso desses *frameworks* alternativos.

1.2. Metodologia

Este trabalho foi desenvolvido em quatro fases: revisão bibliográfica, pesquisa e estudo sobre *frameworks*, desenvolvimento do Portal do LabES em três versões e sugestões para o método FrameWeb (SOUZA, 2007).

Todo o trabalho foi desenvolvido durante o projeto de graduação. Foi feita uma revisão bibliográfica sobre Engenharia *Web*, seguida de uma pesquisa e estudo sobre *frameworks*, dando ênfase ao estudo dos *frameworks* utilizados no desenvolvimento do Portal do LabES. Os requisitos do Portal do LabES foram inicialmente levantados por (FIORI, 2005).

A fim de estudar os *frameworks* a serem utilizados, foi iniciado junto ao Grupo de Usuários de Java do Estado do Espírito Santo (ESJUG[1]) o desenvolvimento de um ambiente cooperativo de aprendizagem chamado JSchool[2]. Foram conduzidas reuniões de treinamento e o projeto encontra-se em fase de implementação.

Além disso, estudamos também o método FrameWeb (SOUZA, 2007). Seguimos o processo proposto pelo método do Portal no desenvolvimento do LabES com os framework nos quais o método foi inicialmente testado, a saber: Struts[2]; Spring Framework; Hibernate e Sitemesh. Desenvolvemos o Portal do LabES em duas novas versões, utilizando diferentes *frameworks Web* (Struts[1] e ZK). Construímos, então, o Portal em três versões e finalmente sugerimos novas propostas para o método a partir da experiência de desenvolvimento.

1.3. Organização da Monografia

Este documento está dividido em 7 capítulos, incluindo esta instrodução.

No Capítulo 2 é feito um apanhado geral dos temas relevantes a esta monografia: Engenharia *Web*, *Frameworks* e FrameWeb.

O Capítulo 3 descreve a especificação de requisitos de uma aplicação *Web*, denominada Portal do LabES (Portal do Laboratório de Engenharia de Software). Para o desenvolvimento deste Portal aplicaremos o método de projeto de aplicações *Web* baseadas em *frameworks* (FrameWeb).

O Capítulo 4 apresenta a especificação de análise do Portal do LabES dentro do paradigma orientado a objetos.

O Capítulo 5 aplica as diretrizes do método FrameWeb no desenvolvimento do projeto do Portal do LabES. O projeto mostra diversos diagramas definidos pelo método.

[1] http://esjug.dev.java.net
[2] http://jschool.dev.java.net

Este capítulo também ilustra o funcionamento do portal implementado com os *frameworks* nos quais o método foi inicialmente testado.

O Capítulo 6 analisa o desenvolvimento do Portal do LabES em diferentes *frameworks Web* e sugere novas propostas de melhorias para o método a partir da experiência adquirida pelo desenvolvimento.

Finalmente, o Capítulo 7 conclui esta monografia com uma avaliação geral do trabalho e as perspectivas de trabalhos futuros.

Capítulo 2

Engenharia *Web*, *Frameworks* e FrameWeb

Com o surgimento da *World Wide Web*, por meio da qual os usuários de computador podem localizar e visualizar documentos baseados em multimídia sobre uma grande gama de assuntos pela Internet, viu-se um crescimento vertiginoso da rede mundial de computadores, tornando-se um dos principais mecanismos de comunicação do mundo. Conseqüentemente, a *Web* passou a ter uma massiva e permanente influência sobre nossas vidas, pois as mesmas já estão presentes na economia, indústria, educação, saúde, administração pública, entretenimento etc.

As aplicações *desktop*, sistema com interfaces ricas baseadas em janelas, estão se tornando *WebApps* (*Web Applications*), aplicações baseadas na plataforma *Web*, que faz uso de Navegadores de Internet (*browser*), de protocolos como o HTTP e de linguagens como o HTML para exibir interfaces gráficas com o usuário. Existe uma tendência de que as interfaces de sistemas convencionais sejam voltadas para a *Web*, buscando facilitar o acesso desses sistemas pelo usuário e uma convergência digital entre plataformas, mídias e protocolos. Assim, o universo dos sistemas *Web* torna-se bastante vasto.

Este capítulo mostra a adaptação da engenharia de software para englobar e satisfazer necessidades das *WebApps* e mostra a descrição de um método, chamado FrameWeb, para construção dessas aplicações, baseado no uso de *frameworks* que agilizam e facilitam o trabalho. Na seqüência são apresentados diferentes tipos de *frameworks*, ressaltando que existem várias implementações para os diferentes tipos.

2.1. Engenharia *Web*

Recentemente, observamos acontecer uma nova “Crise de *Software*” (GIBBIS, 1994), desta vez ocorrida com o desenvolvimento para a *Web* e apelidada de "Crise da *Web*" (HANSEN et al., 1999). Como resultado, este fato gerou a chamada **Engenharia *Web***

(*Web Engineering* - *WebE*), que é a Engenharia de *Software* aplicada no desenvolvimento *Web* (PRESSMAN, 2005). A Engenharia *Web* trata de abordagens disciplinadas e sistemáticas para o desenvolvimento, a implantação e a manutenção de sistemas e aplicações baseados na *Web* (DESPHANDE, 2001).

Segundo Pandolfi e Melotti (PANDOLFI et al., 2006), existem diferentes propósitos e funcionalidades entre *softwares* convencionais e *Web*: as aplicações convencionais tipicamente oferecem serviços e soluções com ênfase na funcionalidade e aplicabilidade, enquanto *Websites* oferecem conteúdos, que são informações e/ou serviços com ênfase na apresentação, comportamento, aparência, navegação e outras qualidades estéticas. A tecnologia para o software convencional está um pouco mais estável e integrada e o público-alvo é bem definido, ocorrendo o contrário na engenharia *Web*.

À medida que as *WebApps* (*Web Applications* ou Aplicações *Web*) se tornam mais integradas às estratégicas de negócio das empresas (por exemplo, comércio eletrônico – *e-commerce*), a necessidade de construir sistemas confiáveis, usáveis e adaptáveis cresce em importância. É por isso que é necessário uma abordagem disciplinada para o desenvolvimento de *WebApps* (PRESSMAN, 2005).

Como no desenvolvimento de aplicações *desktop*, o desenvolvimento de aplicações *Web* também encontra problemas de engenharia em seus projetos: inexperiência e formação inadequada dos desenvolvedores, falta (de uso) de métricas para estimativas, falta (de uso) de modelos de processo, métodos inadequados e obsoletos, planejamento incorreto, prazos e custos excedidos, falta de documentação, dificuldades de implementação e manutenção, *WebApps* mal definidas e projetadas, *layout* inadequado (comunicação visual), mudança contínua dos requisitos, da tecnologia e da arquitetura, falta de rigor no processo de desenvolvimento, dentre outros.

Sendo assim, existe um conjunto de atributos de qualidade a serem considerados neste caso (OLSINA 2001):

- **Usabilidade**: trata-se de um requisito de qualidade como também um objetivo de aplicações *Web*: permitir a acessibilidade do sistema. Logo, o *site* deve ter uma inteligibilidade global, permitir o *feedback* e ajuda *online*, planejamento da interface/aspectos estéticos e aspectos especiais (acessibilidade por deficientes);

- **Funcionalidade**: o software *Web* deve ter capacidade de busca e recuperação de informações/links/funções, aspectos navegacionais e relacionados ao domínio da aplicação;
- **Eficiência**: o tempo de resposta deve ser inferior a 10s (velocidade na geração de páginas/gráficos);
- **Confiabilidade**: relativa à correção no processamento de *links*, recuperação de erros, validação e recuperação de entradas do usuário;
- **Manutenibilidade**: existe uma rápida evolução tecnológica aliada à necessidade de atualização constante do conteúdo e das informações disponibilizadas na *Web*, logo o *software Web* deve ser fácil de corrigir, adaptar e estender.

Para atingir esses atributos de qualidade, uma *WebApp* deve apresentar simplicidade, consistência, identidade, robustez, navegabilidade e atração visual. Como a *WebE* visa aplicar métodos disciplinados de Engenharia de Software, adaptados para criação de aplicações *Web* com alta qualidade, algum processo deve ser seguido a fim de garantir esses atributos.

Ultimamente, devido à complexidade cada vez maior dos sistemas, os modelos de processo mais utilizados para o desenvolvimento de sistemas complexos são os que usam a abordagem incremental e iterativa. Um processo de desenvolvimento segundo esse modelo divide o desenvolvimento de software em iterações. Em cada iteração são realizadas as atividades de análise, projeto, implementação e testes para uma parte do sistema. Assim no modelo incremental e iterativo, um sistema de software é desenvolvido em vários passos similares (iterativo). A cada passo, o sistema é estendido com mais funcionalidades (incremental).

O modelo de processo da *WebE* começa com a análise de requisitos, que é tudo aquilo que é levantado como necessário ao *site* ou aplicação. A análise de requisitos contém atividades de formulação do problema a ser resolvido pela *WebApp*, coleta de requisitos e modelagem de análise. O objetivo dessas atividades é: descrever a motivação básica (metas) e os objetivos da *WebApp*; definir as categorias de usuários; observar o conteúdo e requisitos funcionais da *WebApp*; e estabelecer um entendimento básico de porquê a

WebApp deve ser construída, quem vai usá-la e que problema(s) ela vai resolver para seus usuários (BEZERRA, 2004).

Para a criação de um modelo completo de análise, sua modelagem enfoca quatro tópicos principais: a análise de conteúdo identifica o aspecto completo de conteúdo a ser fornecido para a *WebApp*; a análise de interação descreve o modo pelo qual o usuário interage com a *WebApp*; a análise funcional define as operações que serão aplicadas ao conteúdo da *WebApp* e descreve outras funções de processamento independentes de conteúdo, mas necessários para o usuário final; e a análise de configuração descreve o ambiente e a infra-estrutura nos quais a *WebApp* reside (PRESSMAN, 2005).

Após a modelagem de análise, o processo da Engenharia *Web* segue com a fase de projeto da *WebApp*. Guiado pela informação obtida durante a modelagem de análise, o projeto da *WebApp* enfoca seis tópicos principais: projeto de interface, projeto de estética, projeto de conteúdo, projeto de navegação, projeto arquitetural e projeto de componentes. Existem alguns métodos que podem ser aplicados no desenvolvimento de aplicações *Web*, como WAE (CONALLEN, 2002), OOWS (FONS, 2003) e OOHDM (SCHWABE, 1998).

Seguindo o processo da *WebE*, a fase de implementação relaciona-se à tradução do projeto a uma linguagem de programação, escolhida de acordo com outras variáveis relacionadas à condução do projeto.

Para garantir alta qualidade, a fase de teste deve verificar se o programa construído está de acordo com o projeto e com os requisitos levantados e documentados. Porém, devido ao curto prazo de entrega das *WebApps*, a fase de teste acaba acontecendo de forma incorreta. Para evitar, Wallace e seus colegas (WALLACE et al., 2003) afirmam:

“Teste não deveria esperar até que o projeto esteja terminado. Comece a testar antes que você escreva uma linha de código. Teste constante e efetivamente, e você vai desenvolver um *site Web* muito mais duradouro”.

O teste de *WebApps* enfoca conteúdo, função, estrutura, usabilidade, navegabilidade, desempenho, compatibilidade, interoperabilidade, capacidade e segurança.

2.2. *Frameworks*

O desenvolvimento das máquinas, compiladores, programação OO e Internet alterou

tanto a arquitetura final quanto o modo de se desenvolver software. Hoje o programador tem inúmeras ferramentas que possibilitam um rápido desenvolvimento de aplicativos, linguagens diversas apropriadas a fins diversos, modelos de arquitetura etc. É neste contexto que a palavra ***framework*** ganha sentido amplo e é citada cada vez mais.

Um *framework* é uma aplicação reutilizável e semicompleta que pode ser especializada para produzir aplicações personalizadas (HUSTED et al., 2004). Um *framework* pode reutilizar análise, projeto e código. Reutiliza análise porque descreve os tipos de objetos importantes e como um problema maior pode ser dividido em problemas menores. Reutiliza projeto porque contém algoritmos abstratos que descrevem a interface que o programador deve implementar e as restrições a serem satisfeitas pela implementação. Finalmente, reutiliza código porque torna mais fácil desenvolver uma biblioteca de componentes compatíveis e porque a implementação de novos componentes pode herdar grande parte de seu código das superclasses abstratas (JOHNSON, 1991), além de incorporar diversos padrões de projeto (ALUR et al., 2003).

Um *framework* é definido por Coad (COAD, 1992) como um esqueleto de classes, objetos e relacionamentos agrupados para construir aplicações específicas e, por Johnson (JOHNSON, 1988) como um conjunto de classes abstratas e concretas que provê uma infra-estrutura genérica de soluções para um conjunto de problemas. Essas classes podem fazer parte de uma biblioteca de classes ou podem ser específicas da aplicação. Os *frameworks* possibilitam reutilizar não só componentes isolados como também toda a arquitetura de um domínio específico (MALDONADO, 2002). As aplicações devem ter algo razoavelmente grande em comum e pertencer a um mesmo domínio de problema, assim o *framework* captura a funcionalidade comum a essas aplicações. Da maneira como é apresentado em (GUIZZARDI, 2000), este domínio do problema pode ser tanto um domínio horizontal (ex. interface com usuário, distribuição, entre outros) quanto vertical (o próprio domínio ao qual a aplicação pertence). Em particular, este trabalho só aborda domínios horizontais.

Além de permitir a reutilização, um *framework* minimiza o esforço de desenvolvimento de aplicações, por portar a definição da arquitetura das aplicações geradas a partir dele, como também por ter predefinido o fluxo de controle dessas aplicações (TALIGENT, 1995). A Figura 1 ilustra uma aplicação desenvolvida a partir de um *framework*. A parte sombreada corresponde ao *framework* – uma estrutura de classes que é

reutilizada – e o restante é o que é produzido pelo usuário do *framework* ao desenvolver uma aplicação específica.

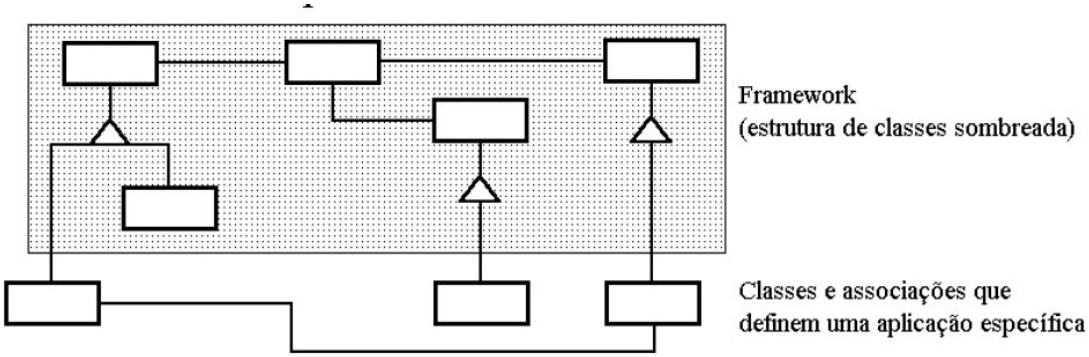

Figura 1 Aplicação desenvolvida utilizando um *framework* (TALIGENT, 1995).

Nos dias atuais é de extrema importância as empresas de TI utilizarem e iniciarem, em seus processos de desenvolvimento de *software*, o uso de *frameworks*, aumentando, assim, a produtividade, qualidade, manutenibilidade e satisfação tanto por parte dos clientes como da empresa em si, e reduzindo tempo.

A seguir, são apresentados diferentes tipos de *frameworks* voltados para a plataforma *Web*, assim como exemplos de alguns *frameworks* existentes para desenvolvimento *Web* na plataforma Java. Os diferentes tipos de *frameworks* foram categorizados por objetivos: MVC (*Front Controller*), RIA (*Rich Internet Applications*), Decoração, Mapeamento Objeto-Relacional, Injeção de Dependência e outros.

2.2.1. *Frameworks* MVC (*Front Controller*)

Os *frameworks Front Controller*, também chamados de Controlador Frontal (ALUR et al., 2003), são baseados na arquitetura MVC (*Model View Controller*) (GAMMA, 1995). *Model View Controller* ou Modelo Visão Controle é um padrão de arquitetura de aplicações que visa separar a lógica da aplicação (*Model*) da interface do usuário (*View*) e do fluxo da aplicação (*Controller*), permitindo que a mesma lógica de negócios possa ser acessada e visualizada por várias interfaces.

Em um projeto de *software* baseado no padrão MVC, define-se uma arquitetura básica com três camadas possivelmente abstratas:

- ***Model***: define a lógica do negócio (MACORATTI, 2000). O modelo representa os

dados da aplicação e as regras do negócio que governa o acesso e a modificação dos dados. O modelo mantém o estado persistente do negócio e fornece ao controlador a capacidade de acessar as funcionalidades da aplicação encapsuladas pelo próprio modelo.

- ***Controller***: implementa a camada responsável pelo gerenciamento de eventos no projeto, tais como cliques do usuário, chamando a camada *Model* para processar os eventos. Também pode manter informações de estado do usuário na aplicação;
- ***View***: gera a interface com o usuário de modo que esta somente requisite o processamento de eventos pelo *Controller*. Não está preocupada com onde e como a informação foi obtida, apenas exibe a informação (MACORATTI, 2000).

Portanto, apesar de MVC ser um nome muito difundido, o nome correto para esse padrão arquitetônico, quando aplicado à *Web*, seria "Controlador Frontal" (*Front Controller*) (ALUR et al., 2003, apud SOUZA, 2007). Neste trabalho, usaremos ambos os nomes indistintamente.

A Figura 2 ilustra o funcionamento de um controlador frontal. O navegador renderiza as páginas na camada *View*, gerando estímulos para o *Controller*, no servidor, que gerencia as requisições e aciona o *Model*, responsável pela execução de lógica de negócio. As respostas retornam nesta mesma cadeia, até sua apresentação na camada *View*, pelo navegador.

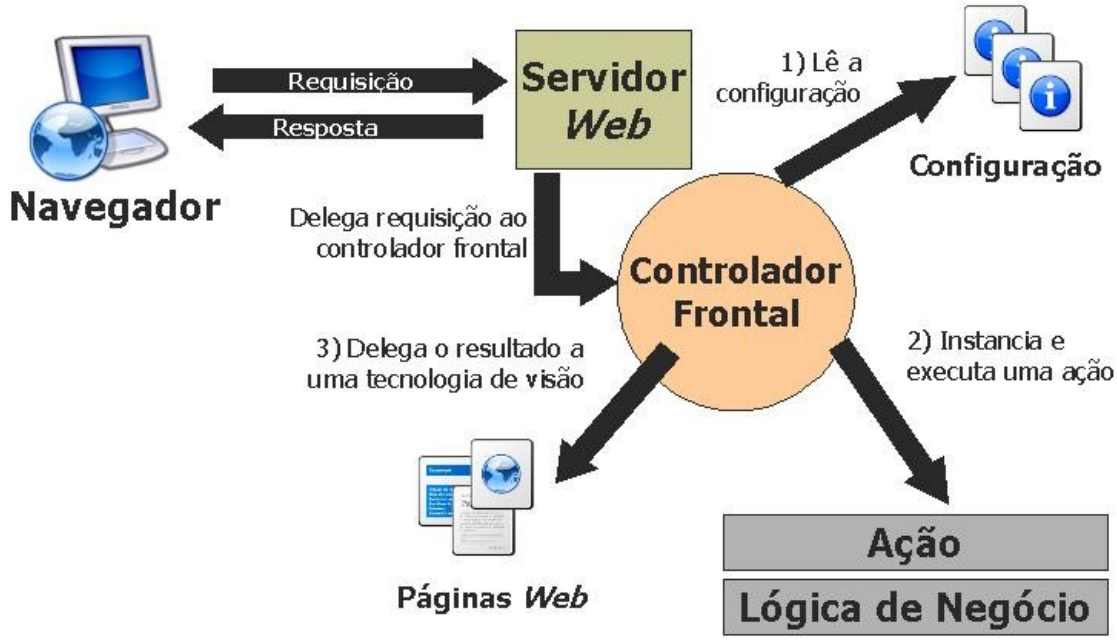

Figura 2 Funcionamento do Controlador Frontal.

Há mais de 50 *frameworks* MVC para a plataforma Java. Os mais populares são o Struts (http://struts.apache.org), WebWork (http://www.opensymphony.com/webwork) e Spring MVC (http://www.springframework.org) (SOUZA, 2007).

2.2.2. *Frameworks* RIA (*Rich Internet Applications*)

RIA é a abreviação de *Rich Internet Applications* ou Aplicações Ricas para Internet. É uma Aplicação *Web* que contém características e funcionalidades de uma aplicação desktop tradicional (roda no cliente e são baseadas em eventos) (LOOSLEY, 2006).

Em uma aplicação *Web* tradicional, todo processamento é feito no servidor. Uma nova página *Web* é carregada a cada clique do usuário (DUHL, 2003).

Tipicamente, uma aplicação RIA transfere a necessidade de processamento do cliente (numa arquitetura cliente-servidor) para o navegador, mas mantém o processamento mais pesado no servidor de aplicação. O processamento no navegador é feito por um *Client Engine* e cada framework o implementa da sua maneira.

Os *frameworks Rich Internet Applications* fornecem aplicações mais ágeis que o comum, por realizar comunicação assíncrona com o servidor (AJAX). AJAX é o acrônimo em língua inglesa de *Asyncronous Javascript And XML*. AJAX não é uma tecnologia, mas várias tecnologias trabalhando juntas, cada uma fazendo sua parte, oferecendo novas

funcionalidades. AJAX utiliza requisições feitas de maneira assíncrona via chamadas de JavaScript e trafegando arquivos XML, para tornar as páginas HTML mais interativas e / ou com interfaces mais agradáveis. Com AJAX, a tela não é somente recarregada toda a vez que o usuário clicar em alguma coisa, possibilitando que apenas a área que precisa ser alterada pela ação realizada seja recarregada em alguns casos. O funcionamento de um *framework* RIA é exibido na seção 6.2.1.1 (*Framework* ZK).

Alguns exemplos de *frameworks Rich Internet Applications* são o ZK (http://zkoss.org/), Echo2 (http://www.nextapp.com/platform/echo2/echo/), OpenLaszlo: (http://www.openlaszlo.org/), Flex (http://www.flex.org/), e o Google Web Toolkit (http://code.google.com/webtoolkit/) (SMITH, 2007).

2.2.3. *Frameworks* de Decoração

Os *frameworks* de decoração são baseados no padrão de projeto *Decorator* (GAMMA, 1995). Eles provêem uma classe que intercepta as requisições e antes de enviar a resposta aos clientes, eles modificam as páginas originais com o *layout* apropriado a fim de gerar uma página final decorada. Decoradores formam modelos de páginas que são aplicados às páginas finais, dando ao *site* um visual uniforme. A Figura 3 ilustra o funcionamento do *framework* de decoração.

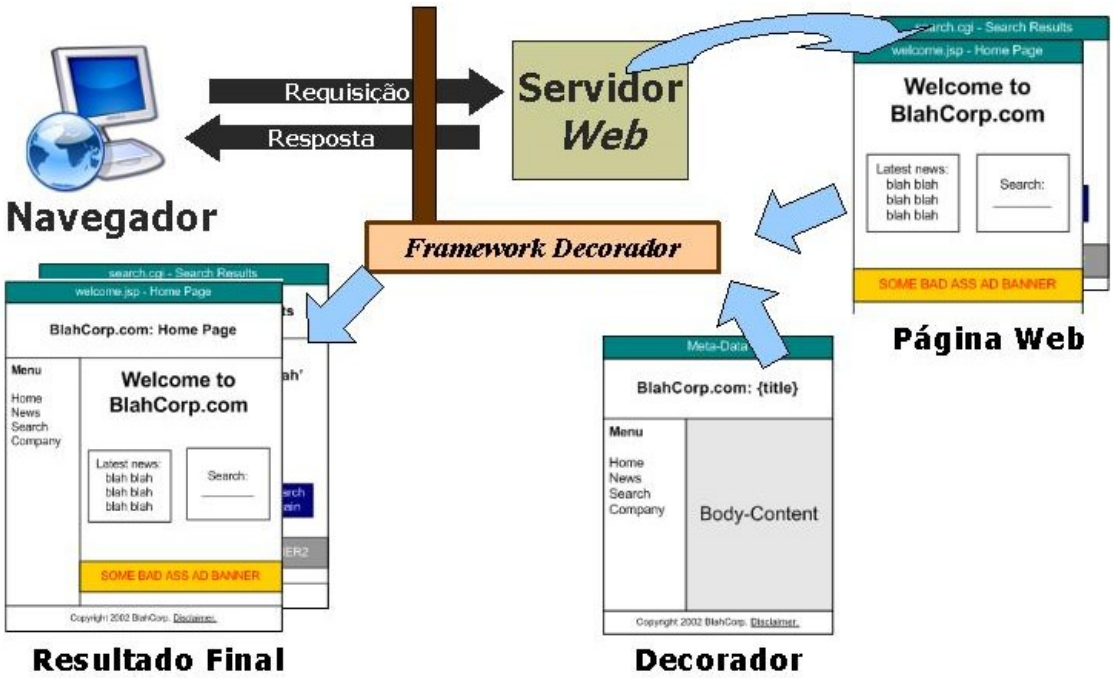

Figura 3 Funcionamento do *framework* de Decoração.

Alguns exemplos de *frameworks* de decoração são o Tiles (http://struts.apache.org/struts-tiles) e o SiteMesh (http://www.opensymphony.com/sitemesh).

2.2.4. *Frameworks* de Mapeamento Objeto/Relacional

Quase todo tipo de aplicação requer que seus dados sejam persistidos de alguma forma. Um sistema de gerenciamento de banco de dados (SGBD) relacional não é específico para Java, tampouco destinado para uma aplicação específica. A tecnologia relacional provê uma forma de compartilhar dados entre aplicações diferentes e entre tecnologias diferentes usadas em uma mesma aplicação, por exemplo, um sistema gerador de relatórios e um sistema de gerenciamento de cadastro. Devido a essas diferentes aplicações que utilizam a mesma base de dados, os SGBDs são o meio de persistir dados das entidades de negócios mais utilizado em aplicações (BAUER et al., 2005).

Quando a tecnologia envolvida é Java, a forma mais baixo nível que esta plataforma oferece para enviar comandos ao SGBD é através da API Java DataBase Connectivity (JDBC, 2005). Através do JDBC uma aplicação Java é capaz de enviar comandos SQL para um SGBD e, assim, realizar operações de consulta, atualização, inserção e remoção de dados. O código JDBC fica responsável por realizar a conversão dos dados provenientes do SGBD em classes orientadas a objeto escritas em Java.

Escrever código de acesso a banco de dados, realizando as conversões necessárias do paradigma relacional para o paradigma OO, é uma tarefa onerosa, repetitiva e algumas vezes propensa a erros devido à diferença entre esses paradigmas. Para facilitar o desenvolvimento de quaisquer tipos de aplicações e evitar os problemas mencionados acima, utilizam-se *frameworks* de mapeamento objeto/relacional.

O termo *Object-Relational Mapping* (ORM) se refere à técnica de mapear uma representação de dados para um modelo relacional dos dados baseado em *Structured Query Language* (SQL), ou seja, mapeamento objeto/relacional é o nome dado para a persistência automática e transparente de classes de uma aplicação em tabelas de bancos de dados relacionais, transformando a representação de dados de um paradigma para o outro (BAUER et al., 2005). Um *framework* ORM é um *framework* de infra-estrutura que fornece serviços de ORM (KING et. al., 2005). A Figura 4 ilustra o comportamento de um

framework ORM.

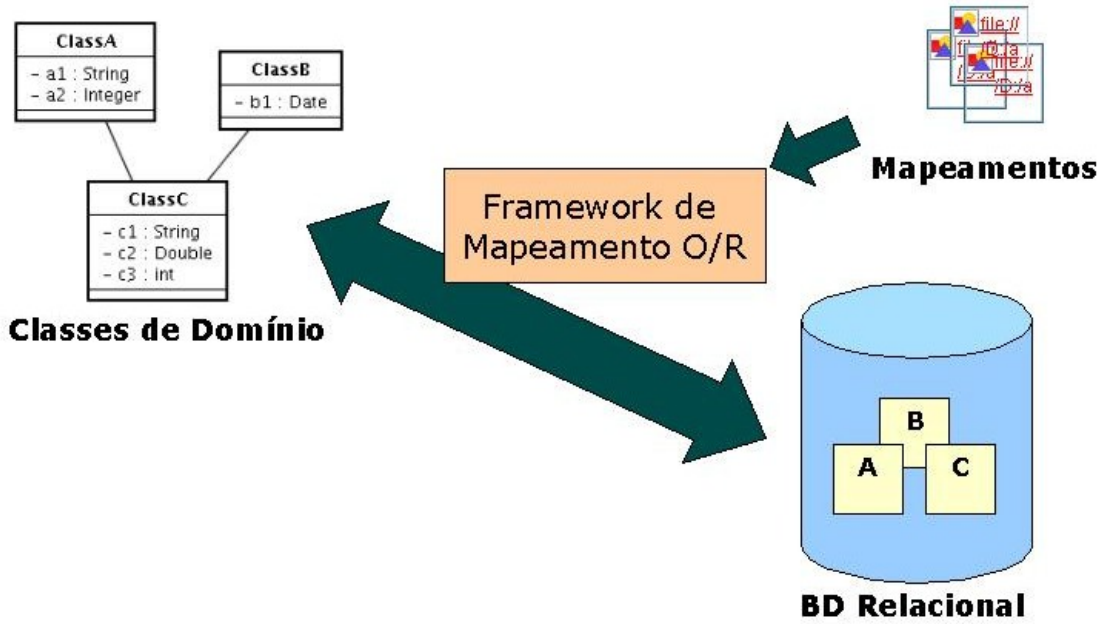

Figura 4 Comportamento do *framework* de Mapeamento Objeto/Relacional.

O mais popular *framework* ORM é o Hibernate (http://www.hibernate.org). Outros conhecidos são: Java Data Object (http://java.sun.com/products/jdo) e Apache Object Relational Bridge (http://db.apache.org/ojb).

2.2.5. *Frameworks* de Injeção de Dependência

O termo Injeção de Dependência (DI - *Dependency Injection)* (FOWLER, 2004) é usado para classificar um tipo específico de inversão de controle onde a responsabilidade de configurar dependências entre classes é assumida por um terceiro objeto. Nesta solução as dependências entre os módulos não são definidas pela programação, mas sim pela configuração de uma infra-estrutura de *software* (*container*) onde eswas dependências são "injetadas". Desta forma, atinge-se um acoplamento mais fraco entre as classes e, por conseqüência, maior reúso. A Figura 5 ilustra o funcionamento de um *framework* de Injeção de Dependência.

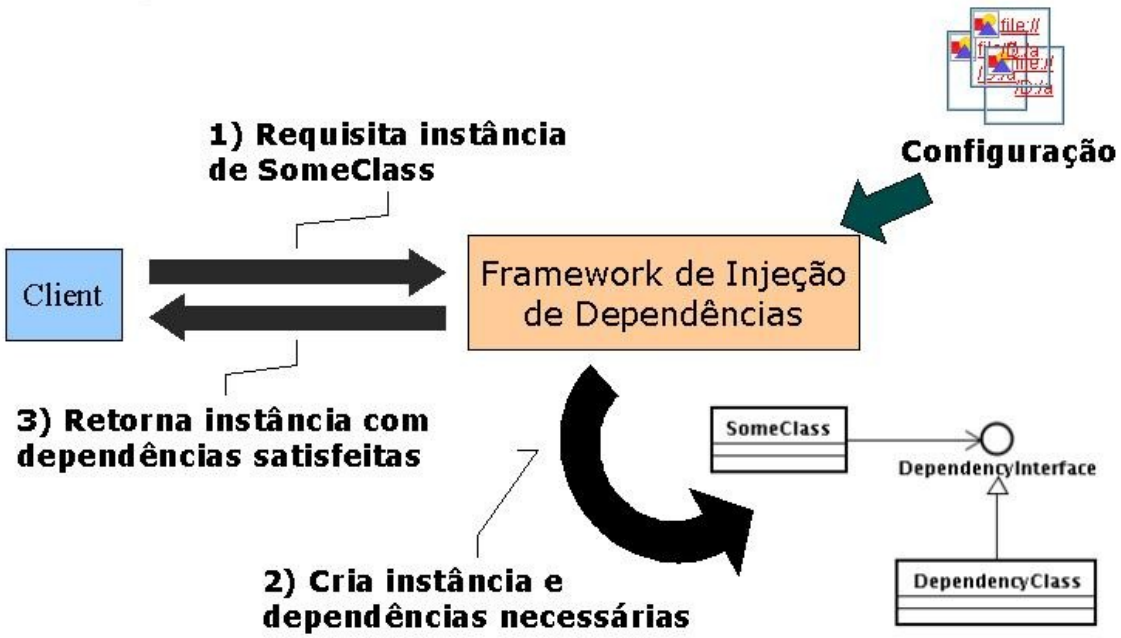

Figura 5 Funcionamento do *framework* de Injeção de Dependência.

Muitos *frameworks* promovem esse serviço. Temos como exemplos o Spring (http://www.springframework.org/), PicoContainer (http://www.picocontainer.org/) e Guice (http://code.google.com/p/google-guice/).

2.2.6. Outros *Frameworks*

Outros tipos de *frameworks* fazem parte do desenvolvimento de *WebApps*, incluindo *frameworks* de Programação Orientada a Aspectos (AOP - *Aspect Oriented Programming frameworks*) e *frameworks* de Autenticação/Autorização os quais provêem autenticação e autorização de usuários (ex.: *login* e senha) para garantir a segurança da aplicação.

Apesar da massiva utilização dos *frameworks*, não há um método da Engenharia *Web* que explore como os mesmos são utilizados na fase de projeto de uma *WebApp*. Para preencher esta abertura, na próxima seção discutiremos um método chamado *FrameWeb, a Framework-based Design Method for Web Engineering*, um método para a *WebE* que foi desenvolvido por (SOUZA, 2007) no Laboratório de Engenharia de Software (LabES) do Departamento de Informática da Universidade Federal do Espírito Santo (DI / UFES).

2.3. FrameWeb

A todo tempo tentamos otimizar os processos da *WebE*. FrameWeb surge com este propósito. Resumidamente, existem duas contribuições principais no método FrameWeb (SOUZA, 2007):

- Uma arquitetura básica para desenvolvimento de *WebApps*;
- Um perfil UML para um conjunto de modelos de projeto que trazem conceitos usados por algumas categorias de frameworks.

FrameWeb é um método para modelagem de *WebApps* que assume a utilização de certas categorias de frameworks durante o processo de software. A idéia é aumentar a agilidade nas principais fases de desenvolvimento, sobretudo nas fases de requisitos, análise, projeto e desenvolvimento.

As fases de levantamento de requisitos e análise podem ser conduzidas da forma que a organização achar mais adequada. FrameWeb apenas sugere que sejam utilizados casos de uso para representação dos requisitos e diagramas de classes UML para modelagem conceitual.

O método dá uma maior importância à fase de projeto (SOUZA, 2007) e, considerando os diferentes tipos de *frameworks* existentes, FrameWeb propõe uma arquitetura básica para desenvolvimento de WebApps. Essa arquitetura divide o sistema em três camadas, como mostra a Figura 6. Temos a lógica de apresentação, lógica de negócio e lógica de acesso a dados.

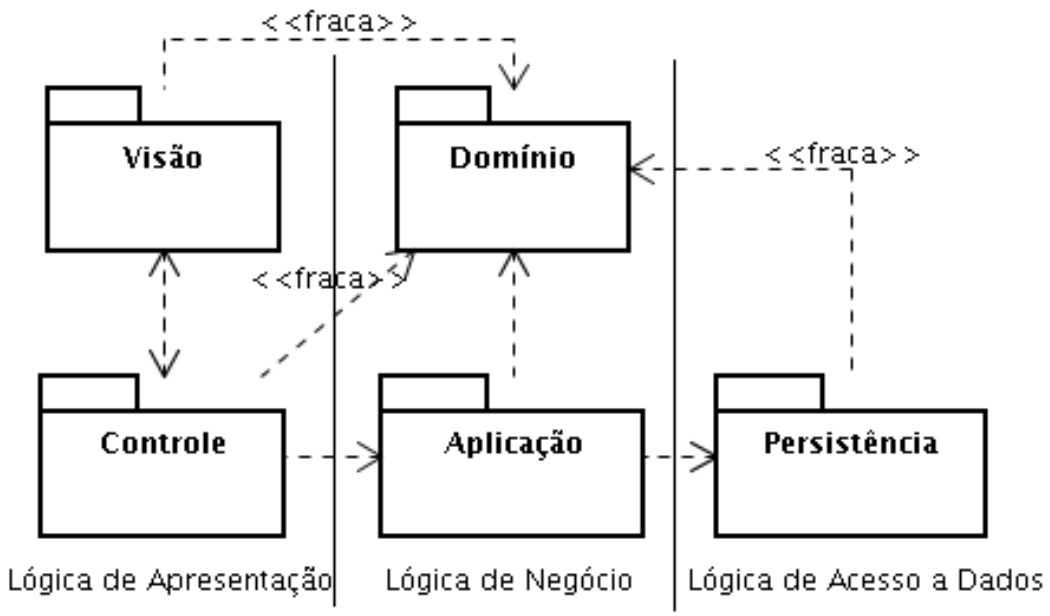

Figura 6 Arquitetura básica para desenvolvimento de *WebApps* (SOUZA et al., 2007).

A primeira camada, lógica de apresentação, tem por objetivo prover interfaces gráficas com o usuário. O pacote **Visão** contém páginas *Web*, folhas de estilo, imagens, *scripts* que executam do lado do cliente, modelos de documentos e outros arquivos relacionados exclusivamente com a exibição de informações ao usuário. O pacote **Controle** envolve classes de ação e outros arquivos relacionados ao *framework* Controlador Frontal (seção 2.2.1). A lógica de negócio encontra-se implementada na segunda camada, dividida em dois pacotes: **Domínio** e **Aplicação**. O primeiro contém classes que representam conceitos do domínio do problema identificados e modelados pelos diagramas de classes na fase de Análise e refinados durante o Projeto. O último tem por responsabilidade implementar os casos de uso definidos na especificação de requisitos, provendo uma camada de serviços independente da interface com o usuário. A terceira e última camada, acesso a dados, contém apenas o pacote **Persistência**, que é responsável pelo armazenamento dos objetos persistentes em mídia de longa duração, como bancos de dados, arquivos etc (SOUZA, 2007).

Os pacotes de Visão e Controle possuem uma dependência mútua, visto que elementos da Visão enviam estímulos do usuário para classes do Controle, que, por sua vez, processam as respostas utilizando páginas, modelos e outros componentes da Visão. O

pacote Controle, na camada de apresentação, depende do pacote de Aplicação, que provê ao usuário acesso às funcionalidades do sistema. O pacote Aplicação possui um relacionamento com Domínio, porque manipula objetos desse pacote. A Aplicação também depende do pacote Persistência para recuperar, gravar e excluir objetos de domínio como resultado da execução dos casos de uso. Os pacotes Controle**,** Visão e Persistência possuem relacionamento <<fraco>> de dependência com o pacote Domínio, representando um baixo acoplamento. O estereótipo indica que não são feitas alterações em entidades de domínio: objetos de Domínio são utilizados na camada de apresentação apenas para exibição de seus dados ou como parâmetros nas evocações de métodos entre um pacote e outro; e objetos de Domínio são utilizados na camada de Persistência apenas para mapeá-los ao banco de dados relacional (SOUZA, 2007).

Segundo o método, o pacote Controle envolve classes de ação e outros arquivos relacionados ao *framework* Controlador Frontal (seção 2.2.1). No pacote de Persistência um *framework* de mapeamento objeto/relacional (ORM, seção 2.2.4) deve ser utilizado por meio do padrão de projeto *Data Access Object* (DAO) (ALUR et al., 2003). O padrão DAO adiciona um pacote a mais, separando a lógica de acesso a dados da tecnologia de persistência de maneira que a camada de aplicação não conheça qual *framework* ORM está sendo utilizado, permitindo que o mesmo seja trocado, se necessário. (SOUZA, 2007). As classes DAO são responsáveis pela lógica de persistência de objetos de uma classe de domínio específico.

Seguindo a mesma abordagem de linguagens de modelagem, como WAE (CONALLEN, 2002) e UWE (KOCH, 2000), FrameWeb define extensões leves para o meta-modelo da UML a fim de representar componentes típicos da *Web* e os componentes relacionados aos *frameworks*, criando uma linguagem de modelagem, baseada em UML, para quatro diagramas relativos às diferentes camadas da arquitetura (SOUZA, 2007):

- O modelo de domínio representa os objetos do domínio do problema e seu mapeamento para a persistência;
- O modelo de persistência apresenta os objetos responsáveis pela persistência dos dados;
- O modelo de navegação mostra páginas *Web* e demais artefatos que compõem a camada de visão e interagem com a camada de controle para enviar ao sistema os

estímulos do usuário;

- O modelo de aplicação reúne as classes de serviço que implementam a lógica de negócio descrita nos casos de uso e suas dependências em relação aos demais modelos do sistema.

Com os diagramas prontos, os desenvolvedores terão maior facilidade na fase de codificação ao traduzirem esses diagramas para o código, pois estes já apresentam conceitos diretamente ligados com a tecnologia de implementação usando *frameworks*.

O método FrameWeb (SOUZA, 2007) estabelece uma correspondência entre os modelos e as categorias de *frameworks*:

- Existe uma correspondência entre os *frameworks* Controladores Frontais (seção 2.2.1) e o Modelo de Navegação, porque o modelo de navegação é um diagrama de classe da UML que representa os diferentes componentes que formam a camada de Lógica de Apresentação, como páginas *Web*, formulários HTML e classes de ação do *framework* Controlador Frontal.
- Existe uma correspondência entre os *frameworks* de Mapeamento O/R (seção 2.2.4) e os Modelos de Persistência e Domínio. O modelo de persistência é um diagrama de classes da UML que representa as classes DAO existentes e são implementados de acordo com o *framework* de Mapeamento O/R utilizado (ex.: *framework* Hibernate, ou *framework* OJB etc.). O modelo de domínio também representa alguns conceitos utilizados nos *frameworks* de Mapeamento O/R permitindo que estes transformem objetos que estão na memória em linhas de tabelas no banco de dados relacional.
- Por fim, existe uma correspondência entre os *frameworks* de Injeção de Dependência (seção 2.2.5) e o Modelo de Aplicação, porque o modelo de aplicação é utilizado para guiar a configuração das dependências entre os entre os pacotes Controle, Aplicação e Persistência, ou seja, quais classes de ação dependem de quais classes de serviço e quais DAOs são necessários para que as classes de serviço alcancem seus objetivos. Essas configurações são realizadas nos *frameworks* de Injeção de Dependência.

Uma descrição mais completa do método FrameWeb, um método para projeto de sistemas de informação *Web* com arquiteturas baseadas no uso de determinados tipos de *frameworks*, pode ser lida em (SOUZA, 2007).

Alguns estudos de caso preliminares já foram feitos com *frameworks* específicos das categorias listadas na seção 2.2. O Portal do Laboratório de Engenharia de Software (LabES) do Departamento de Informática da Universidade Federal do Espírito Santo (DI / UFES), utilizado durante todo o capítulo para aplicação e análise do método, foi o projeto piloto de FrameWeb, desenvolvido inicialmente por uma equipe de estudantes de computação que passaram por um treinamento sobre Engenharia *Web*, FrameWeb e um conjunto de *frameworks* (Struts², Spring *Framework*, Hibernate e Sitemesh).

Atualmente encontram-se em desenvolvimento outras iniciativas de uso de FrameWeb na prática. O Grupo de Usuários de Java (JUG) do Estado do Espírito Santo[3] organiza grupos de desenvolvimento de software em Java e atualmente conduz o projeto JSchool[4], no qual uma equipe de 10 membros do JUG, da qual eu também participo, desenvolve um ambiente colaborativo de aprendizagem modelado com FrameWeb. A equipe recebeu treinamento sobre o método e os *frameworks* utilizados e o projeto encontra-se atualmente no estágio de implementação.

Nos próximos capítulos apresentamos a Especificação de Requisitos, Análise, Projeto e Implementação para o Portal do LabES, seguindo os moldes do FrameWeb e utilizando o mesmo conjunto de *frameworks* nos quais o método foi inicialmente baseado. Em seguida desenvolvemos duas novas versões do sistema, utilizando *frameworks Web* diferentes, com o intuito de analisar a eficácia do método em diferentes contextos e sugerir alterações nos modelos para uma melhor adaptação a diferentes *frameworks Web*.

De acordo com a arquitetura básica para desenvolvimento de WebApps proposta por (SOUZA, 2007), os *frameworks Web* são os *frameworks* utilizados na camada de apresentação.

[3] http://esjug.dev.java.net
[4] http://jschool.dev.java.net

Capítulo 3

Especificação de Requisitos do Portal do LabES

Neste capítulo descreveremos a especificação de requisitos funcionais, baseado no trabalho do (FIORI, 2005), de uma aplicação *Web* que será implementada em três versões a fim de analisar a adaptabilidade de FrameWeb em diferentes instâncias de *frameworks Web*.

O principal objetivo da fase de especificação de requisitos é entender como o sistema será usado, preocupando-se com o que foi solicitado pelo usuário. Ela deve produzir uma modelagem das funcionalidades solicitadas.

O Laboratório de Engenharia de Software (LabES) da Universidade Federal do Espírito Santo deseja desenvolver um portal para melhor interagir com o público interessado na área de pesquisa de Engenharia de Software.

Dessa forma, descreveremos os requisitos de uma etapa inicial desse portal, com serviços para controlar usuários e o nível de interação dos mesmos com as funcionalidades providas pelo sistema, bem como um conjunto básico de serviços provendo informações sobre os projetos em curso no LabES e materiais disponíveis.

Esta atividade foi desenvolvida usando a técnica de modelagem de casos de uso e, portanto, este capítulo apresenta, na seção 3.2, os diagramas de caso de uso gerados, bem como as descrições dos atores e dos casos de usos identificados nesses diagramas. Antes disso, porém, o escopo do sistema é descrito na seção 3.1.

3.1. Descrição de Escopo

Para prover uma maior interação com a comunidade, o LabES resolveu desenvolver o seu próprio Portal de Engenharia de Software, o Portal LabES, para disponibilizar informações e materiais desenvolvidos nessa área de estudos.

Nesse portal estarão disponíveis informações sobre itens. Itens podem ser projetos ou materiais publicados pelos usuários do sistema. Os itens estão ligados a áreas de interesse

de pesquisa, como por exemplo, a área de interesse chamada Informática. Áreas de interesse de pesquisa podem possuir sub-áreas.

Um projeto no sistema pode não ter materiais, porém quando for criado algum material, este deve pertencer a algum projeto. Um projeto também pode possuir sub-projetos.

3.2. Modelo de Casos de Uso

O gerenciamento e o acesso ao Portal do LabES serão feitos de acordo com o tipo de usuário. Para tal, os usuários são agrupados nas seguintes categorias: Internauta, Usuário Padrão, Membro LabES, Colaborador, Professor e Administrador, como mostrado na Figura 7.

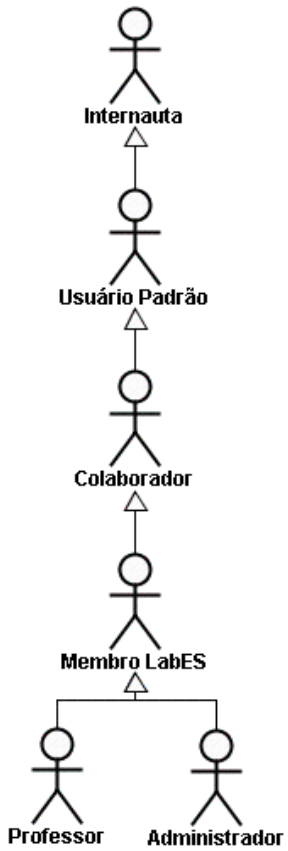

Figura 7 Tipos de Usuários do Portal LabES (FIORI, 2005).

Os tipos de usuários estão dispostos segundo as suas funcionalidades da seguinte

forma:

Internauta: representa qualquer pessoa que esteja navegando na Internet. Esse tipo de usuário pode se cadastrar como um Usuário Padrão, tem acesso às funcionalidades de visualização de qualquer material ou informação disponível, pode efetuar *download* de materiais e pode utilizar o mecanismo de busca do portal;

Usuário Padrão: representa usuários cadastrados no LabES. Esse tipo de usuário poderá alterar seus dados e sua senha e terá de se autenticar no sistema para ter acesso às funcionalidades específicas dessa classe de usuário. Apenas usuários padrão poderão futuramente fazer o download do ambiente ODE. Além disso, poderão relatar falhas do ambiente e propor novas funcionalidades para o mesmo;

Colaborador: representa um Usuário Padrão que se dispôs a colaborar com o Portal, corrigindo alguma falha ou desenvolvendo uma nova funcionalidade para o ambiente;

Membro LabES: agrupa todos os usuários que atuam dentro do laboratório. Esse tipo de usuário pode consultar os dados de qualquer outro membro do LabES e pode registrar materiais a serem disponibilizados no portal;

Professor: representa os professores associados ao LabES. Esse tipo de usuário pode disponibilizar projetos, cadastrar áreas de pesquisa, além de poder consultar os dados dos membros do LabES e registrar materiais, como qualquer membro do LabES;

Administrador: representa os administradores do sistema, que têm permissão para cadastrar novas funcionalidades, tipo de usuário, membros do LabES, professores e outros administradores. De fato, um administrador tem acesso a todas as funcionalidades do sistema.

Deve-se observar que essa descrição é uma contextualização do sistema e, portanto, os dados de cada um dos tipos de usuários a serem cadastrados são apresentados nas descrições dos casos de uso apresentados na próxima seção.

A Figura 8 mostra o diagrama de pacotes do sistema, subdividindo-o em dois subsistemas, a saber:

Controle de Usuário: envolve toda a funcionalidade relacionada com o controle de usuários do Portal LabES, abrangendo controle de funcionalidades, tipos de usuários, usuários e áreas.

Controle de Itens: disponibiliza todas as funcionalidades do sistema aos seus

usuários, abrangendo todo o controle de material e projeto.

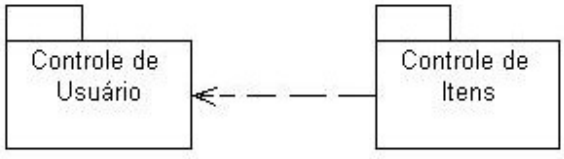

Figura 8 Diagrama de pacotes (FIORI, 2005).

3.2.1. Pacote Controle de Usuário

A Figura 9 mostra o diagrama de caso de uso referente ao controle de usuários. Na seqüência, os casos de usos identificados são descritos.

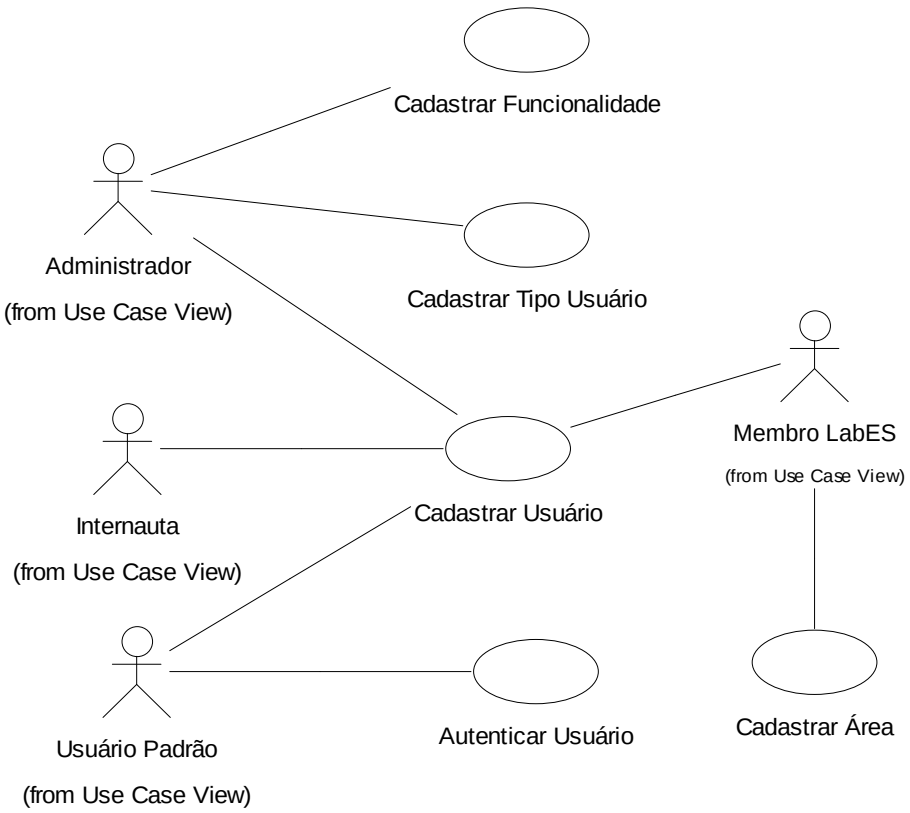

Figura 9 Caso de Uso do Pacote Controle de Usuário (FIORI, 2005).

3.2.1.1 Caso de Uso Cadastrar Tipo de usuário

Este caso de uso é responsável pelo controle de tipos de usuário, abrangendo a criação

de um novo tipo de usuário, alteração, consulta e exclusão de tipos de usuário existentes. Os eventos *Consultar Tipo de Usuário, Criar Novo Tipo de Usuário, Alterar Dados do Tipo de Usuário e Excluir Tipo de Usuário* compõem este caso de uso.

Curso Normal

Consultar Tipo de Usuário

São listados o nome e a descrição de todos os tipos de usuário cadastrados no sistema. Este cenário somente é habilitado para o usuário do tipo administrador.

Criar Novo Tipo de Usuário

O administrador solicita a criação de um novo tipo de usuário e informa o nome e descrição. O tipo de usuário, então, é criado e o cenário *Consultar Tipo de Usuário* é executado automaticamente para exibição da listagens dos tipos de usuários cadastrados no sistema.

Alterar Dados do Tipo de Usuário

O administrador informa qual tipo de usuário deseja alterar as características e essas são exibidas. O administrador pode alterar as informações listadas no cenário *Criar Novo Tipo de Usuário*.

Excluir Tipo de Usuário

O administrador informa quais tipos de usuário deseja excluir e uma mensagem é exibida, pedindo a confirmação da exclusão. Caso seja confirmada, os tipos de usuário serão excluídos. Não é permitida a exclusão de um tipo de usuário que tenha funcionalidades e usuários associados a ele.

3.2.1.2 Caso de Uso Cadastrar Funcionalidade

Este caso de uso é responsável pelo controle de funcionalidades, abrangendo a criação de uma nova funcionalidade, alteração, consulta e exclusão de funcionalidades existentes. Os eventos *Consultar Funcionalidade, Criar Nova Funcionalidade, Alterar Dados da Funcionalidade e Excluir Funcionalidade* compõem este caso de uso.

Curso Normal

Consultar Funcionalidade

São listados o nome e a descrição de todas as funcionalidades cadastradas no sistema. Este cenário somente é habilitado para o usuário do tipo administrador.

Criar Nova Funcionalidade

O administrador solicita a criação de uma nova funcionalidade e informa o nome, descrição e tipos de usuários que podem ter acesso. A funcionalidade, então, é criada e o cenário *Consultar Funcionalidade* é executado automaticamente.

Alterar Dados da Funcionalidade

O administrador informa qual funcionalidade deseja alterar as características e essas são exibidas. O administrador pode alterar as informações listadas no cenário *Criar Nova Funcionalidade*.

Excluir Funcionalidade

O administrador informa quais funcionalidades deseja excluir e uma mensagem é exibida, pedindo a confirmação da exclusão. Caso seja confirmada, as funcionalidades serão excluídas.

3.2.1.3 Caso de Uso Cadastrar Área

Este caso de uso é responsável pelo controle de áreas de interesse, abrangendo a criação de uma nova área, alteração, consulta e exclusão de áreas existentes. Os eventos *Consultar Área, Criar Nova Área, Alterar Dados da Área e Excluir Área* compõem este caso de uso.

Curso Normal

Consultar Área

São listados o nome e a descrição de todas as áreas cadastradas no sistema. Este cenário somente é habilitado para o usuário do tipo membro LabES.

Criar Nova Área

O membro do LabES solicita a criação de uma nova área e informa o nome, descrição

e área da qual faz parte, quando pertinente. A área, então, é criada e o cenário *Consultar Área* é executado automaticamente.

Alterar Dados da Área

O membro do LabES informa qual área deseja alterar as características e essas são exibidas. O membro do LabES pode alterar as informações listadas no cenário *Criar Nova Área*.

Excluir Área

O membro do LabES informa quais áreas deseja excluir e uma mensagem é exibida, pedindo a confirmação da exclusão. Caso seja confirmada, as áreas serão excluídas.

Um usuário está associado a uma área se está associado a algum de suas sub-áreas. Uma área não pode ser excluída se tiver usuários associados ou itens associados. Ao excluir uma área, suas sub-áreas devem ser excluídas.

3.2.1.4 Caso de Uso Cadastrar Usuário

Este caso de uso é responsável pelo controle de usuário, abrangendo a criação de um novo membro do LabES pelo Administrador ou de um usuário padrão pelo Internauta, consulta, alteração e exclusão de usuários existentes. Os eventos *Consultar Usuário, Criar Novo Usuário, Alterar Dados do Usuário e Excluir Usuário* compõem este caso de uso.

Curso Normal

Consultar Usuário

São listados o login e o nome de todos os usuários cadastradas no sistema. Este cenário somente é habilitado para o usuário do tipo membro LabES.

Criar Novo Usuário

O internauta ou o administrador solicita a criação de um novo usuário e informa o nome, data de nascimento, sexo, escolaridade, instituição à qual está vinculado, áreas de interesse, atuação profissional, e-mail, login, senha, endereço e telefone para contato. Se ação estiver sendo realizada pelo administrador, deve-se informar o tipo do usuário, senão o usuário cadastrado será um usuário padrão. O usuário, então, é criado e se a ação estiver

sendo realizada pelo administrador o cenário *Consultar Usuário* é executado automaticamente.

Alterar Dados do Usuário

O usuário padrão informa que deseja alterar suas características e essas são exibidas. O usuário padrão pode alterar as informações listadas no cenário *Criar Novo Usuário*.

Excluir Usuário

O administrador informa quais usuários deseja excluir e uma mensagem é exibida, pedindo a confirmação da exclusão. Caso seja confirmada, os usuários serão excluídos. Um usuário padrão também pode excluir a si mesmo.

Cursos Alternativos / de Exceção

Criar Nova / Alterar Dados de Usuário

Um usuário deve possuir um login e não deve haver dois usuários com o mesmo login. Caso não cumpra estes requisitos, uma mensagem de erro é mostrada e a ação não é efetivada.

3.2.1.5 Caso de Uso Autenticar Usuário

Este caso de uso é responsável pela autenticação do usuário no sistema, abrangendo a efetuação de login, logout e envio de senha.

Curso Normal

Efetuar *Login*

O usuário padrão informa seu *login* e sua senha. Verifica-se se o *login* existe. Caso exista, verifica-se se a senha corresponde ao *login* informado. Caso corresponda, o *login* é efetuado e o usuário tem acesso às funcionalidades do sistema que são específicas para seu tipo de usuário.

Efetuar *Logout*

O usuário padrão efetua o *logout* não tendo mais acesso às funcionalidades do sistema que são específicas para seu tipo de usuário.

Enviar Senha

O usuário padrão informa seu *login* e solicita o envio de senha. Caso o *login* exista, a senha é enviada para o e-mail do usuário que possui o *login* informado.

Cursos Alternativos / de Exceção

Efetuar *Login*

Caso o *login* não exista no sistema ou a senha não corresponda ao *login* informado, uma mensagem de erro é exibida.

Enviar Senha

Caso o *login* não exista, uma mensagem de erro é exibida.

3.2.2. Pacote Controle de Itens

A Figura 10 mostra o diagrama de casos de uso referente ao pacote Controle de Itens. Na seqüência, os casos de uso identificados são descritos.

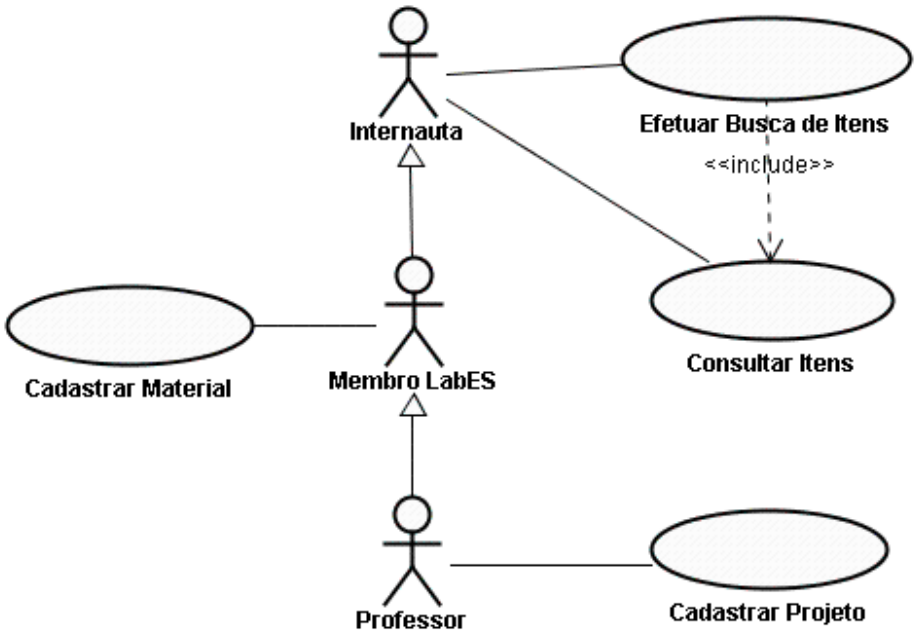

Figura 10 Caso de Uso do Pacote Controle de Itens.

3.2.2.1 Caso de Uso Cadastrar Projeto

Este caso de uso é responsável pelo controle de projetos, abrangendo a criação de um novo projeto, alteração, consulta e exclusão de projetos existentes. Os eventos *Consultar Projeto, Criar Novo Projeto, Alterar Dados do Projeto e Excluir Projeto* compõem este caso de uso.

Curso Normal

Consultar Projeto

São listados o título, resumo e data de disponibilização de todos os projetos cadastrados no sistema. Este cenário é habilitado para qualquer tipo de usuário.

Criar Novo Projeto

O professor solicita a criação de um novo projeto e informa o título, resumo, data em que foi disponibilizado, áreas de interesse, professor responsável, alunos envolvidos e projeto do qual faz parte, quando for o caso. O projeto, então, é criado e o cenário *Consultar Projeto* é executado automaticamente.

Alterar Dados do Projeto

O professor informa qual projeto deseja alterar as características e essas são exibidas. O professor pode alterar as informações listadas no cenário *Criar Novo Projeto*.

Excluir Projeto

O professor informa quais projetos sob sua responsabilidade deseja excluir e uma mensagem é exibida, pedindo a confirmação da exclusão. Caso seja confirmada, os projetos são excluídos. Ao excluir um projeto, seus sub-projetos são excluídos.

Um material está associado a um projeto se está associado a algum de seus sub-projetos. Um projeto não pode ser excluído se tiver materiais associados.

3.2.2.2 Caso de Uso Cadastrar Material

Este caso de uso é responsável pelo controle de materiais, abrangendo a criação de um novo material, alteração, consulta e exclusão de materiais existentes. Os eventos *Consultar Material, Criar Novo Material, Alterar Dados do Material e Excluir Material* compõem

este caso de uso.

Curso Normal

Consultar Material

São listados o título, resumo e data de disponibilização de todos os materiais cadastrados no sistema. Este cenário é habilitado para qualquer tipo de usuário.

Criar Novo Material

O membro do LabES solicita a criação de um novo material e informa o título, resumo, autores, data em que foi disponibilizado, áreas de interesse, responsável pelo material, arquivo e projeto, se pertinente. O material, então, é criado e o cenário *Consultar Material* é executado automaticamente.

Alterar Dados do Material

O responsável pelo material ou o administrador informa qual material deseja alterar as características e essas são exibidas. O usuário pode alterar as informações listadas no cenário *Criar Novo Material*.

Excluir Material

O responsável pelo material ou o administrador informa quais materiais deseja excluir e uma mensagem é exibida, pedindo a confirmação da exclusão. Caso seja confirmada, os materiais serão excluídos.

3.2.2.3 Caso de Uso Efetuar Busca de Itens

Este caso de uso é responsável pela busca de itens (materiais e projetos) no portal.

Curso Normal

O Internauta informa áreas e/ou um conjunto de palavras para busca de itens e os tipos de item (projeto, material, todos) que deseja consultar. É apresentada uma lista com os títulos dos itens encontrados que estão classificados nas áreas informadas e que têm alguma das palavras informadas em seu resumo. Os itens são apresentados de forma agrupada por tipo de item. Caso deseje, o internauta poderá consultar algum item, realizando o caso de

uso *Consultar Item*.

3.2.2.4 Caso de Uso Consultar Item

Este caso de uso disponibiliza itens (projetos e materiais) para consulta. Os eventos *Consultar Projeto e Consultar Material* compõem este caso de uso.

Curso Normal

Consultar Projeto

O internauta seleciona um projeto e são mostrados os seguintes dados do mesmo: título, resumo, áreas, data em que foi disponibilizado, situação atual, professor responsável, alunos envolvidos e sub-projetos. Caso deseje, o internauta pode consultar sub-projetos e materiais do projeto selecionado. Para consultar os materiais disponíveis para esse projeto, é realizada a ação *Consultar Material*, contudo, considerando apenas os itens relativos ao projeto em questão.

Consultar Material

O internauta seleciona um material e são mostrados os seguintes dados do mesmo: título, resumo, autores, data em que foi disponibilizado, áreas de interesse e um *link* para o arquivo do material. Caso deseje, ele pode abrir ou salvar o arquivo do material em seu computador.

Capítulo 4

Análise do Portal do LabES

A análise dentro do paradigma da orientação a objetos tem o propósito de definir todas as classes (atributos, as associações e comportamentos associados a ela), que são relevantes para o problema que o sistema se propõe resolver. Neste capítulo, apresenta-se a especificação de análise para o projeto de desenvolvimento do Portal do Laboratório de Engenharia de Software (Portal do LabES) proposto no capítulo anterior. A seguir, são apresentados os modelos de classes como resultado desta fase.

4.1. Modelagem de Classes

Ao abstrair os conceitos do mundo real em classes de objetos, seus atributos e relacionamentos entre eles que são relevantes para o problema a ser resolvido, são formados diagramas de classes, que mostram as propriedades e as associações entre as classes do sistema.

Para efeito de modularização, classes são separadas em pacotes. Os diagramas de pacotes dão uma visão geral da divisão das classes e colaboração entre pacotes. O sistema foi dividido pacotes mostrados na Figura 11.

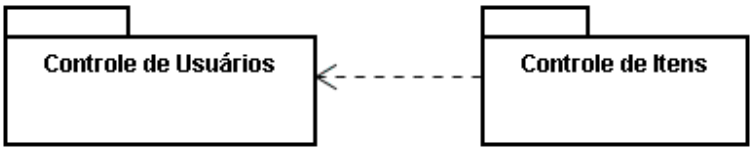

Figura 11 Diagrama de pacotes (FIORI, 2005).

O pacote Controle de Usuários reúne as classes relacionadas com o controle de usuários do Portal LabES, abrangendo controle de funcionalidades, tipos de usuários, usuários e áreas. O pacote Controle de Itens contém classes relacionadas com o controle de materiais e projetos.

A seguir, são mostrados e explicados os diagramas de classe de cada pacote.

4.2. Pacote Controle de Usuários

Este pacote contém classes relacionadas com o processo de controle de usuários. O diagrama de classes da Figura 12 exibe as classes que compõem o pacote.

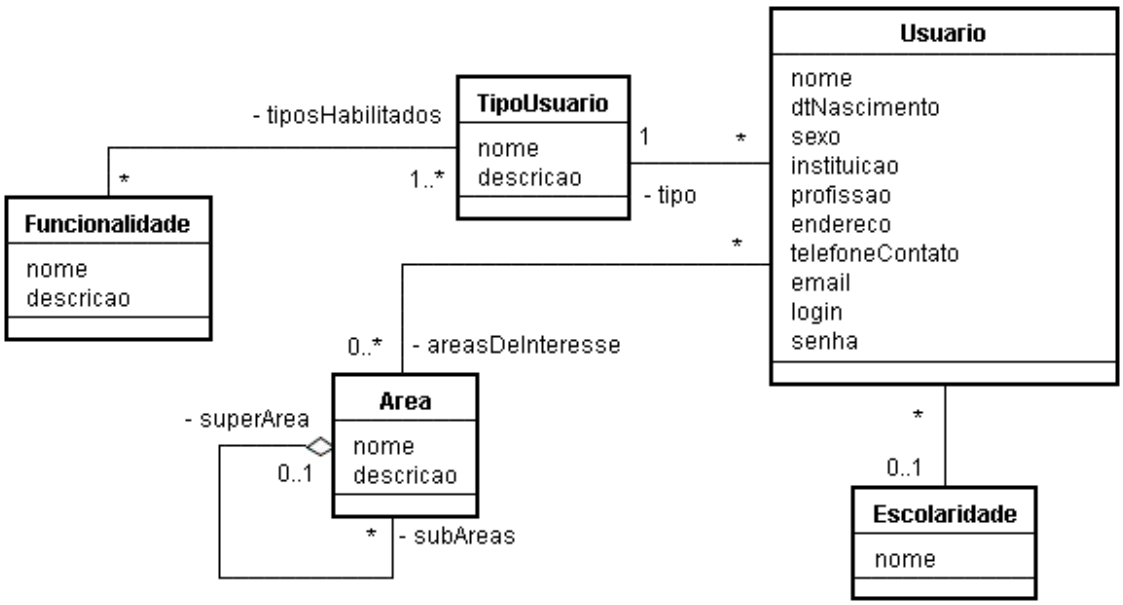

Figura 12 Diagrama de Classes do Pacote Controle de Usuários (SOUZA, 2007).

A classe `Usuario` representa os usuários cadastrados no sistema. Um `Usuario`, que pode ter alguma `Escolaridade` (Ensino Médio, Superior Incompleto, entre outros), está vinculado ao tipo de usuário e associado a zero ou mais áreas de interesse. A `Área` pode conter sub áreas e assim sucessivamente. A classe `TipoUsuario` representa os tipos de usuários cadastrados. Por fim, a classe `Funcionalidade` representa funcionalidades disponíveis para os tipos de usuários.

4.3. Pacote Controle de Itens

Este pacote contém classes relacionadas com o processo de controle de itens. O diagrama de classes da Figura 13 exibe as classes que compõem o pacote.

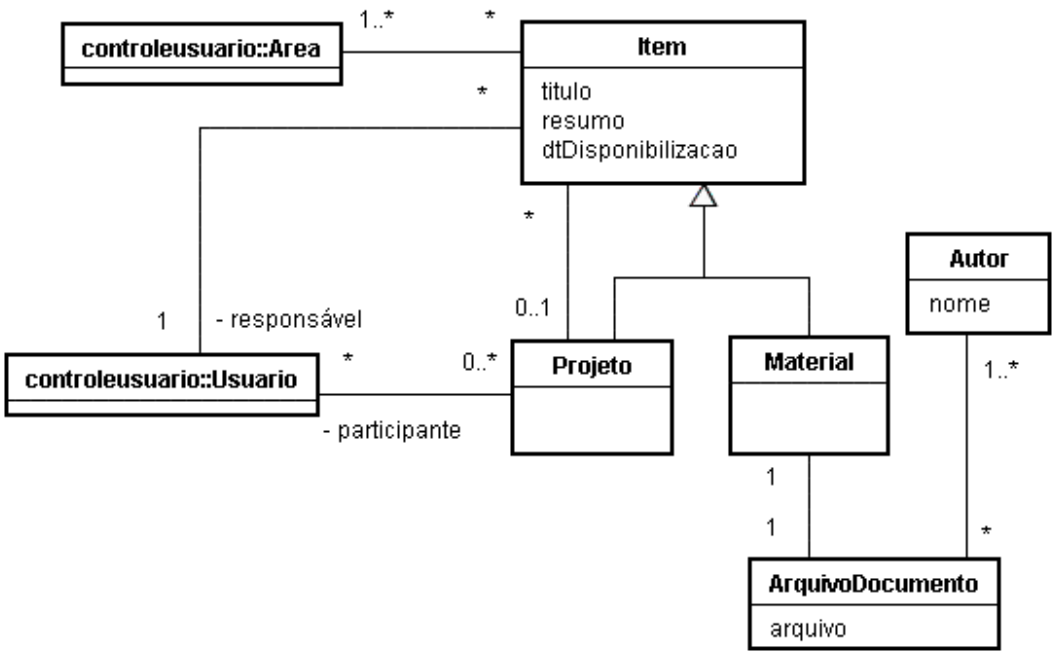

Figura 13 Diagrama de Classes do Pacote Controle de Itens.

`Projeto` e `Material` são Itens. Todo `Item` possui algum usuário responsável e está associado a pelo menos uma `área de interesse`. `Item` pode ou não ter um `Projeto` no âmbito do qual foi produzido. Um `Projeto` contém os usuários participantes. `Material` está associado a um `ArquivoDocumento` que contém o arquivo do material e está associado aos `Autores` do material.

Capítulo 5

Projeto e Implementação do Portal do LabES

O produto da fase de projeto é o modelo de projeto, que é obtido a partir do modelo de análise, e que é a base da construção do software. O projeto, ao contrário da análise, não supõe que há uma tecnologia "perfeita" disponível. Ele traz a análise do sistema para a plataforma computacional na qual o software será implementado. No caso do projeto para o Portal do Laboratório de Engenharia de Software (Portal do LabES), tal plataforma é composta pela linguagem de programação *Java* e um banco de dados relacional, com vistas à implantação do sistema na plataforma *Web* utilizando um conjunto de *frameworks* para sua construção.

O desenvolvimento deste ambiente segue a aplicação do processo de software para sistemas Orientados a Objeto para a *Web* baseados em *frameworks* segundo o método FrameWeb (seção 2.3).

A seção 5.1 detalha a Arquitetura do Sistema, listando as tecnologias escolhidas para implementação do sistema em código e os diferentes pacotes criados para modularização do software. A seção Figura 14 descreve as classes utilitárias necessárias no desenvolvimento do sistema. As seções 5.3, 5.4, 5.5 e 5.6 apresentam os modelos definidos pelo método FrameWeb, respectivamente: modelo de domínio, de persistência, de navegação e de aplicação. A seção 5.7 procura ilustrar o funcionamento do Portal do LabES implementado.

5.1. Arquitetura do Sistema

A fase de Projeto do Sistema começa com a definição da Arquitetura do Sistema, que consiste em mapear os conceitos modelados na fase de análise para a plataforma tecnológica de desenvolvimento, adicionando aspectos de implementação e diminuindo o nível de abstração para chegar mais próximo do nível de implementação em uma linguagem de programação.

5.1.1. Tecnologia

O Portal do LabES foi implementado utilizando a linguagem de programação Java, especificamente a plataforma Java EE (Enterprise Edition) em conjunto com diversos *frameworks* gratuitos e de código aberto que auxiliam na formação de um arquitetura em 3 camadas para a *Web*. Além disso, foi utilizado um banco de dados relacional para persistência, por ainda ser considerada a melhor tecnologia para armazenamento de dados. A tabela 5.1 descreve as tecnologias escolhidas.

Tabela 5.1 – Tabela com as tecnologias do Portal do LabES.

Tecnologia	***Descrição***	***Propósito***
Java SE	Base da linguagem de programação orientada a objetos e plataforma de desenvolvimento Java.	Linguagem de programação utilizada.
Java EE	Especificação da plataforma Java para aplicações distribuídas, incluindo a API Servlet para desenvolvimento *Web*.	Implementação de páginas *Web* para desenvolvimento de sistemas cliente-servidor.
Tomcat Servlet Container	Servidor que implementa a API Servlets e permite a criação de aplicações *Web* em Java.	Prover o servidor para as páginas *Web* e componentes de negócio.
Banco de Dados Relacional	Banco de dados relacional HSQLDB que suporta o padrão SQL-92.	Armazenar dados de objetos persistentes.
Spring *Framework*	*Framework* gratuito que provê uma fábrica de *beans*. Qualquer objeto que forma sua aplicação e que está sob controle do Spring é considerado um *bean*. Enfim, um *bean* trata-se apenas de um objeto de sua aplicação e nada mais. Esses *beans*, provavelmente possuem dependências entre si. Essas dependências são definidas através de metadados. Neste contexto, o Spring, via injeção automática de dependências, gerencia a dependência entre os *beans*.	*Framework* de Injeção de Dependência com o propósito de integrar as diferentes camadas da arquitetura e prover serviços de transação. Para maiores informações veja a seção 2.2.5.
Hibernate ORM	*Framework* gratuito que realiza mapeamento objeto/relacional, ou seja, faz persistência de objetos em banco de dados relacional.	*Framework* de Mapeamento Objeto/Relacional com o propósito de realizar a persistência dos objetos. Para maiores informações veja a seção 2.2.4.

Tecnologia	*Descrição*	*Propósito*
Struts²	Framework gratuito que auxilia na adequação de aplicações Web ao padrão de projeto MVC, provendo, além disso, uma série de funcionalidades úteis como conversão de dados, validação, i18n (desenvolver páginas internacionalizadas, ou seja, em vários idiomas) etc.	*Framework* Controlador Frontal com o propósito de facilitar o desenvolvimento da aplicação *Web*. Para maiores informações veja a seção 2.2.1.
FreeMarker	*Framework* gratuito que provê uma linguagem de construção de modelos de documento (*templates*) que são mesclados com modelos de dados para montar documentos *Web*. Integra-se com o Struts².	Montar as páginas *Web* como modelos e mesclar com os dados vindos do Struts² para gerar o HTML final.
SiteMesh	*Framework* gratuito para decoração de páginas *Web*. Decoradores formam modelos de páginas que são aplicados às páginas finais, dando ao site um visual uniforme. Integra-se com o Struts².	*Framework* de Decoração utilizado para montar os modelos visuais das páginas e dar um tratamento visual uniforme a todo o sistema. Para maiores informações veja a seção 2.2.3.

O *framework* Struts², descrito na tabela acima, é um dos *framework Web* nos quais o método FrameWeb foi baseado. Na proposta original de FrameWeb não foram feitos experimentos com diferentes instâncias de *frameworks*. Por isso a necessidade, neste trabalho, da experiência do método FrameWeb com diferentes *frameworks Web* (Capítulo 6).

5.1.2. Pacotes

O diagrama da Figura 14 mostra a divisão do sistema em módulos (pacotes Java). A divisão foi feita, primeiramente, correspondendo à divisão em subsistemas da Especificação de Requisitos e à divisão em pacotes da Especificação de Análise. Em segundo lugar, foi feita uma nova divisão referente à arquitetura de camadas definida pelo FrameWeb.

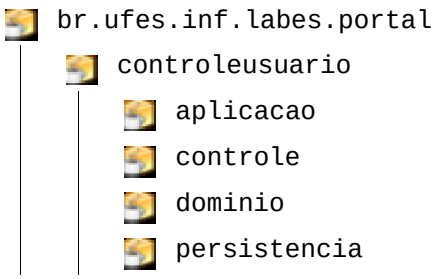

Figura 14 Pacotes do Portal do LabES desenvolvido com o *framework* Struts[2].

5.2. Utilitários

O pacote `util` contém classes utilitárias que são usadas por classes de diversos outros pacotes.

5.2.1. Pacote net.java.dev.esjug.util.persistence e net.java.dev.esjug.util.domain

Estes pacotes contêm classes e interfaces utilitárias relacionadas à persistência dos objetos e aos objetos de domínio. A Figura 15 ilustra as classes e interfaces utilitárias com seus relacionamentos.

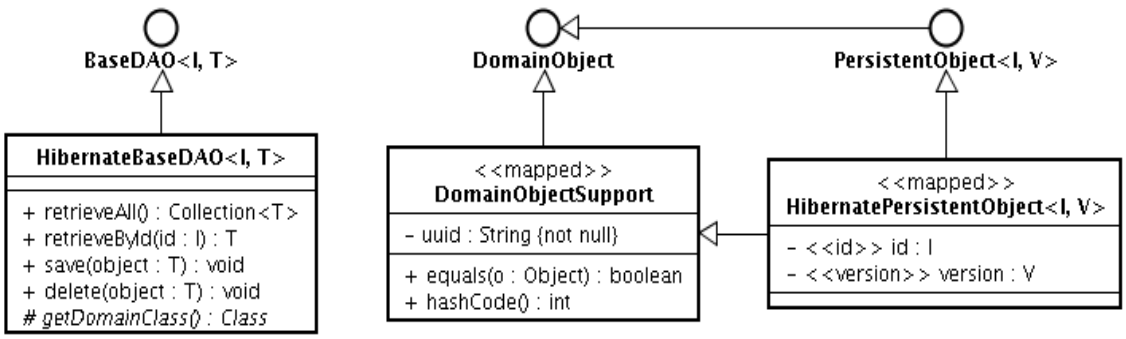

Figura 15 Diagrama de classes do pacote utilitário de persistência (SOUZA, 2007).

Tanto a interface **BaseDAO**, quanto a sua implementação **HibernateDAOBase** presentes no pacote **`net.java.dev.esjug.util.persistence`** são declaradas usando tipos genéricos, deixando a cargo de suas sub-interfaces e sub-classes a especificação da classe gerenciada por cada DAO (padrão de projeto *Data Access Object*).

O DAO base define métodos para:

- Recuperar todos os objetos de uma determinada classe (método retrieveAll);
- Recuperar um objeto dado seu identificador (método retrieveById);
- Salvar um objeto (método save);
- Excluir um objeto (método delete).

A interface **`DomainObject`** e sua implementação **`DomainObjectSupport`** presentes no pacote **`net.java.dev.esjug.util.domain`** definem um identificador universal único (UUID – *Universal Unique Identifier*) e uma implementação padrão dos métodos equals() e hashCode() que utilizam esta identificador universal, que é criado automaticamente quando o objeto é construído.

O métodos hashCode() e equals(), definidos na classe `Object` da API do Java, estão presentes em todos os objetos Java (todas as classes Java automaticamente herdam de `Object`). Tais métodos são utilizados por coleções baseadas em tabela *hash* (como HashSet) e em diversas outras situações. É boa prática que hashCode() retorne o mesmo código *hash* para o mesmo objeto e que equals() retorne true quando comparamos um objeto com ele mesmo (esta comparação é feita através do UUID do objeto). A noção de "mesmo objeto" em sistemas de informação não é aquele que encontra-se na mesma posição de memória, portanto a implementação padrão destes métodos não atende. A utilização de um UUID[5] que é gerado na criação do objeto e é armazenado no banco de dados sem nunca ser alterado garante que os métodos equals() e hashCode() funcionarão de acordo.

A interface **`PersistentObject`** e sua implementação **`HibernatePersistentObject`** presentes no pacote **`net.java.dev.esjug.util.persistence`** definem os atributos de identidade e de versão, além de herdar um identificador (id) único universal. Mais informações sobre identificadores podem ser encontradas na seção 5.1 da documentação do Hibernate disponível no endereço http://www.hibernate.org/hib_docs/v3/reference/en/html/, enquanto colunas de versão são abordadas nas seções 5.1 e 11.3 do mesmo documento.

FrameWeb adicional na UML um estereótipo de classe chamado <<mapeada>>. Este

[5] Uma discussão sobre a utilização de UUID pode ser vista em http://jroller.com/page/jcarreira?entry=overcoming_the_hashcode_object_identity.

estereótipo significa que a classe é mapeada, ou seja, ela não é persistente, mas suas propriedades serão. As classes `DomainObject` e `HibernatePersistentObject` foram declaradas como mapeadas, portanto elas não são entidades persistente em si, mas suas sub-classes herdarão seus atributos juntamente com seus mapeamentos de persistência.

Por questões de brevidade, as classes e interfaces desses pacotes não são exibidas em nenhum outro diagrama, porém assume-se que:

- Todas as classes de domínio exibidas nos modelos de domínio (seção 5.3) que não possuem alguma superclasse definida no diagrama herdam automaticamente da classe `HibernatePersistentObject<Long, Long>`;
- Todas as interfaces DAO exibidas nos modelos de persistência (seção 5.3) herdam automaticamente da interface `BaseDAO`;
- Todas as classes exibidas nos modelos de persistência (seção 5.3) que implementam as interfaces DAO para o *framework* ORM Hibernate (seção 2.2.4) herdam automaticamente da classe `HibernateBaseDAO<I,T>`, especificando `Long` no lugar do tipo genérico I e no lugar do tipo genérico T, a classe de domínio cuja persistência é gerenciada por aquele DAO.

Ex.:

- public class Usuario extends HibernatePersistentObject<Long, Long>;
- public interface UsuarioDAO extends BaseDAO<Long, Usuario>;
- public class UsuarioDAOHibernate extends HibernateBaseDAO<Long, Usuario> implements UsuarioDAO.

5.2.2. Pacote net.java.dev.esjug.frameworks.crud

Para facilitar a construção de cadastros simples (também conhecidos por CRUD, abreviação de *Create, Retrieve, Update and Delete*), foi construído um *framework* de CRUD que se integra com o *framework* Controlador Frontal Struts[2] e o *framework* de Injeção de Dependência Spring. A Figura 16 mostra duas classes e uma interface que fazem parte desse *framework*.

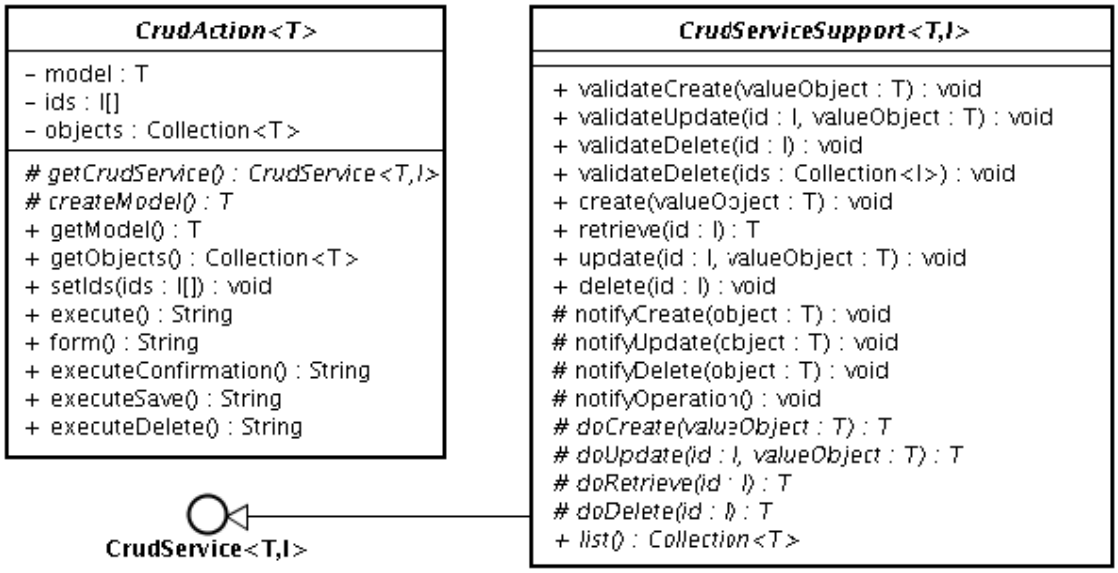

Figura 16 Diagrama de classes do *framework* de CRUD (SOUZA, 2007).

A classe abstrata **`CrudAction`** deve ser estendida por classes de ação do Struts[2] que controlam o cadastro de objetos de domínio. A classe do objeto de domínio cujo cadastro será controlado deve ser especificada no lugar do tipo genérico T e os métodos abstratos getCrudService() e createModel() devem ser implementados retornando, respectivamente, a aplicação de cadastro e um objeto de domínio vazio.

Para exemplificar, seguem abaixo algumas linhas da implementação da classe AcaoCadastrarUsuario, responsável por repassar à aplicação correspondente os estímulos do usuário com relação ao caso de uso "Cadastrar Usuário" (seção 3.2.1.4).

```
public class AcaoCadastrarUsuario extends CrudAction<Usuario, Long> {
    /** Aplicação responsável pelo caso de uso "Cadastrar Usuário". */
    private AplCadastrarUsuario aplCadastrarUsuario;
    ...
    protected Usuario createModel() {
            return new Usuario();
    }
    protected CrudService<Usuario, Long> getCrudService() {
            return aplCadastrarUsuario;
```

```
    }
    ...
}
```

A interface **`CrudService`** e sua implementação padrão **`CrudServiceSupport`** servem de base para implementação da classe de aplicação responsável pela execução de casos de uso de cadastro. Utilizando do padrão de projeto *Template Method* (GAMMA et al., 1994), provê os métodos CRUD, delegando a métodos abstratos à parte específica ao objeto de domínio em questão. Métodos de validação também são providos e devem ser sobrescritos se desejado, lançando exceções (*runtime*) no caso de erros de validação.

Por questões de brevidade, as classes e interfaces desses pacotes não são exibidas em nenhum outro diagrama, porém assume-se que: Todas as interfaces e classes de aplicação nos modelos de aplicação (seção 5.6) automaticamente herdam da classe `CrudService <T, I>` e `CrudServiceSupport<T, I>`, respectivamente, especificando Long no lugar do tipo genérico I e no lugar do tipo genérico T, a classe de domínio.

Ex.:

- public interface AplCadastrarUsuario extends CrudService<Usuario, Long>;
- public class AplCadastrarUsuarioImpl extends CrudServiceSupport<Usuario, Long> implements AplCadastrarUsuario;

As seções a seguir descrevem classes que pertencem às camadas de domínio, persistência, controle e aplicação, respectivamente. Todos os diagramas seguem a linguagem de modelagem definida pelo método adotado, FrameWeb.

5.3. Modelo de Domínio

O modelo de domínio é um diagrama de classes da UML que representa os objetos de domínio do problema e seu mapeamento para a persistência em banco de dados relacional.

Por meio de um perfil UML definido pelo método FrameWeb, o mapeamento de persistência é adicionado ao modelo conceitual, indicando como converter classes para tabelas, atributos para colunas etc. A maioria dos meta-dados para configuração do

mapeamento é opcional (SOUZA, 2007).

Nenhuma das classes possui um atributo identificador (chave-primária), pois o mesmo é herdado de uma classe (`HibernatePersistentObject`) do pacote utilitário (seção 5.2.1).

5.3.1. Pacote br.ufes.inf.labes.portal.controleusuario.dominio

O modelo de domínio para esse pacote é mostrado na Figura 17.

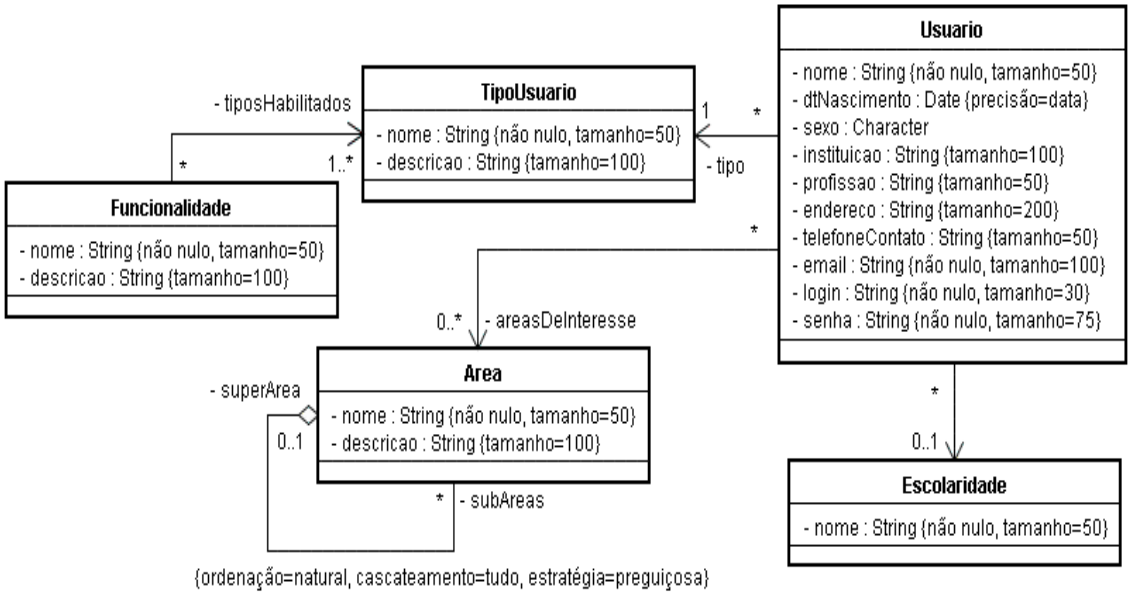

Figura 17 Modelo de domínio do pacote controleusuario (SOUZA, 2007).

A partir desse diagrama construímos o código para as classes da camada de domínio e seus mapeamentos objeto/relacionais. Em primeiro lugar, note que os tipos de dado definidos pela API Java foram adicionados aos atributos, as navegabilidades foram determinadas e nomes foram dados para as várias extremidades navegáveis. A associação entre `Usuario` e `Escolaridade`, que não possui nome na extremidade, recebe o nome padrão (o nome da classe com 1ª letra minúscula): escolaridade.

Vários atributos receberam configurações de mapeamento O/R. Aqueles que receberam a restrição {não nulo} não podem ter valores vazios inseridos no banco de dados, enquanto os demais ficam com o padrão, que permite valores nulos. O atributo dtNascimento é do tipo Date, que armazena data e hora, porém só estamos interessados em parte da data e, por isso, FrameWeb define a restrição {precisão=data} (eventuais

informações de hora, minuto e segundo serão desconsideradas).

A associação um-para-muitos entre `Area` e ela mesma recebeu configurações de mapeamento, indicando que o conjunto deve ser ordenado naturalmente (ordem definida pelo método compareTo() do objeto `Area`), todas as operações (inserção, alteração, atualização e exclusão) devem ser cascateadas para as sub-áreas e que a estratégia de recuperação da associação será preguiçosa (*lazy*), ou seja, o *framework* de persistência não carrega os relacionamentos automaticamente. Os demais relacionamentos de multiplicidade "n" ficam com os valores *default*: o tipo é conjunto, a ordenação é nenhuma, o cascateamento é também nenhum e a estratégia é normal (carregando os relacionamentos).

Não foi definido nenhum atributo como chave primária (estereótipo <<id>>) nas classes de domínio, porque todas as classes herdam automaticamente de uma classe base `HibernatePersistentObject`, conforme discutido na seção 5.2.1.

5.3.2. Pacote br.ufes.inf.labes.portal.controleitens.dominio

O modelo de domínio para esse pacote é mostrado na Figura 18.

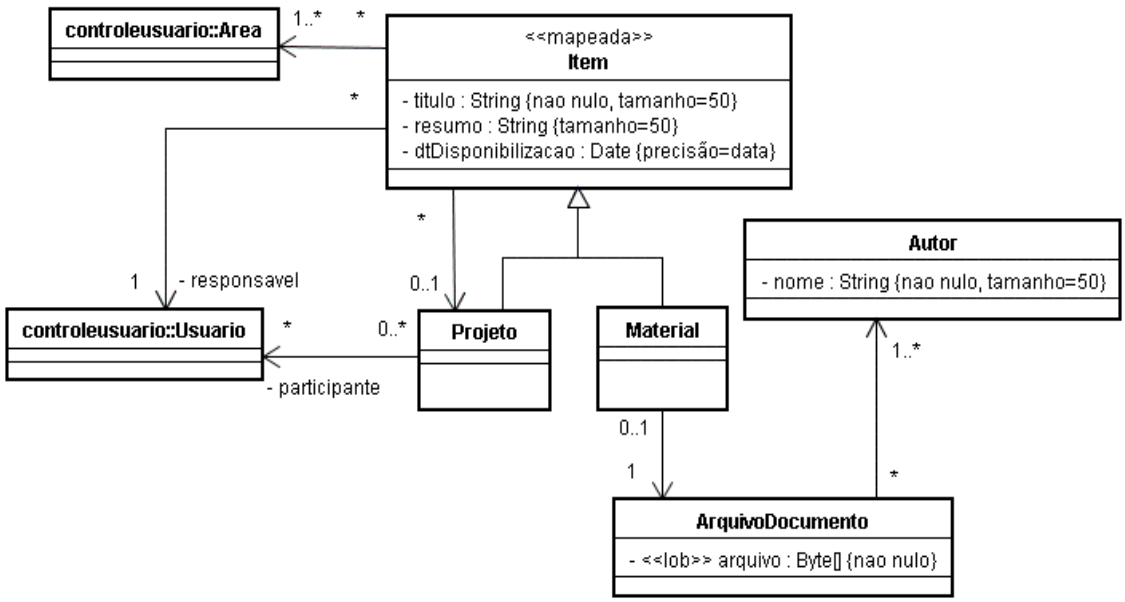

Figura 18 Modelo de domínio do pacote controleitens.

FrameWeb define que o estereótipo de classe chamado <<mapeada>> na classe `Item` significa que a classe Item é mapeada, ou seja, ela não é persistente, mas suas propriedades devem ser persistidas nas subclasses. O estereótipo de atributo chamado <<lob>> define

que o atributo deve ser armazenado em um campo grande (CLOB, BLOB e afins).

A leitura do restante desse diagrama segue os mesmos princípios da descrição do modelo de domínio do pacote `controleusuario` (seção 5.3.1).

5.4. Modelo de Persistência

Segundo o método, o padrão de projeto DAO (*Data Access Object*) deve ser usado para criar uma camada entre os objetos de domínio e a tecnologia de persistência (geralmente um *framework* de mapeamento O/R). As classes DAO são responsáveis pela lógica de persistência de objetos de uma classe de domínio específica.

O modelo de persistência é um diagrama de classes da UML que apresenta as classes DAO que compõem a camada de persistência de um determinado pacote do sistema. O diagrama representa, para cada classe de domínio que necessita de lógica de acesso a dados, uma interface e uma classe concreta DAO que implementa essa interface.

Como descrito na seção 5.2.1, por questões de brevidade, subtende-se que todas as classes herdam automaticamente de uma classe base (`HibernateBaseDAO`) e todas as interfaces herdam de uma interface base (`BaseDAO`).

5.4.1. Pacote br.ufes.inf.labes.portal.controleusuario.persistencia

O modelo de persistência para o pacote é mostrado na Figura 19.

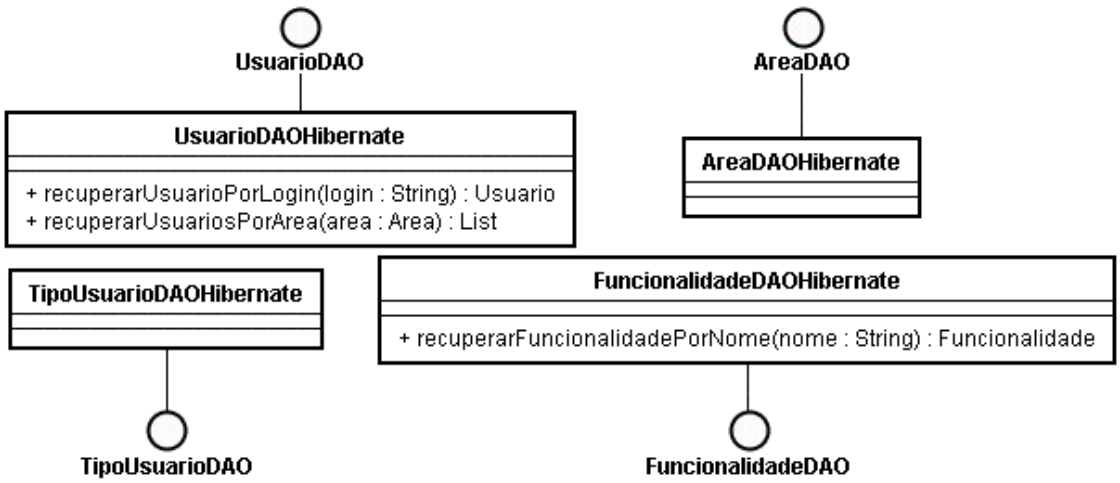

Figura 19 Modelo de persistência do pacote controleusuario.

O método recuperarUsuarioPorLogin() é uma consulta utilizada pelo caso de uso “Autenticar Usuário” descrito na seção 3.2.1.5. Dado um *login*, recupera o objeto Usuário que possui aquele *login*, se existe algum.

O método recuperarUsuariosPorArea(), dada uma área, recupera todos os usuários que escolheram aquela área como área de interesse. Esta consulta é requerida pelo cenário “Excluir Área” do caso de uso “Cadastrar Área”, que impede que uma área seja excluída se ela já foi escolhida por algum usuário.

É importante notar que o método de pesquisa recuperarUsuariosPorArea() só é necessário porque no modelo de domínio não há navegabilidade na associação entre `Usuario` e `Area` na extremidade de `Usuario`. Esta decisão de projeto foi tomada para não onerar o modelo de domínio com a manutenção de uma coleção (`Area` teria uma coleção de `Usuarios` que a escolheram) que serve somente ao propósito de verificar uma condição de integridade para exclusão de uma área.

O método recuperarFuncionalidadePorNome() recupera o objeto `Funcionalidade` que possui um dado nome, se existe algum. Esta consulta é requerida por todas as funcionalidades, que devem verificar quais usuários possuem permissão para utilizá-la antes de executar.

5.4.2. Pacote br.ufes.inf.labes.portal.controleitens.persistencia

O modelo de persistência para esse pacote é mostrado na Figura 20.

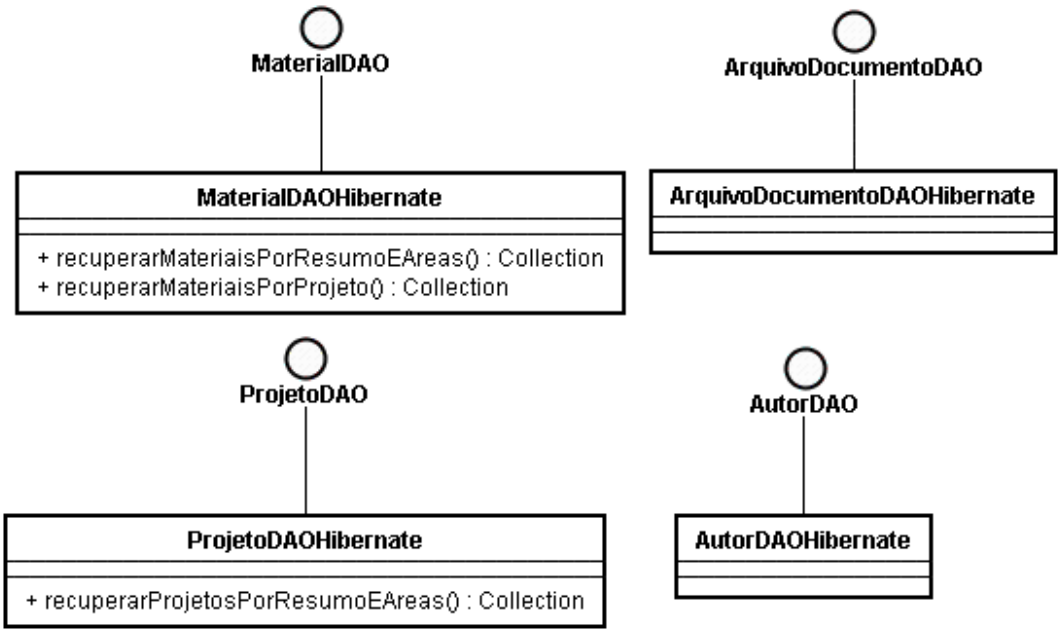

Figura 20 Modelo de persistência do pacote controleitens.

O método de pesquisa recuperarMateriaisPorProjeto() retorna todos os materiais de um determinado projeto. No modelo de domínio não há uma associação direta entre `Projeto` e `Material`. Essa associação aparece na associação entre a classe pai `Item` e a classe `Projeto`.

Já os métodos recuperarMateriaisPorResumoEAreas() e recuperarProjetosPorResumoEAreas () recuperam os `Materiais` e `Projetos` respectivamente, dado as áreas de interesse e/ou uma descrição do resumo. Essas consultas são requeridas para listar os itens oriundos do caso de uso “Efetuar Busca de Itens”, descrito na seção 3.2.2.3.

5.5. Modelo de Navegação

O modelo de navegação mostra páginas *Web* e demais artefatos que compõem a camada de visão e interagem com a camada de controle para enviar ao sistema os estímulos do usuário.

FrameWeb define na Tabela 5.2 os estereótipos UML utilizados pelos diferentes elementos que podem ser representados no Modelo de Navegação.

Tabela 5.2 – Possíveis mapeamentos objeto/relacionais para o Modelo de Domínio

Estereótipo	***O que representa***
(nenhum)	Uma classe de ação, para a qual o *framework* Controlador Frontal delega a execução da ação.
<<página>>	Uma página *Web* estática ou dinâmica.
<<modelo>>	Um modelo (*template*) de uma página *Web*, processado por *engine* que retorna um documento HTML como resultado.
<<form>>	Um formulário HTML.

5.5.1. Pacote br.ufes.inf.labes.portal.controleusuario.controlador

O pacote controleusuario é o que possui a maior quantidade de casos de uso de cadastro. Apenas um de seus cinco casos de uso (Autenticar Usuário) não é um caso de uso no padrão CRUD (*Create, Retrieve, Update & Delete* – Criar, Consultar, Alterar e Excluir).

Para implementação dos casos de uso de cadastro foi elaborado um pequeno *framework* para reúso das lógicas de apresentação e negócio comuns em casos de cadastro. Este *framework* é mostrado no diagrama de aplicação do pacote utilitário na seção 5.2.2, e o modelo de navegação do caso de uso "Cadastrar Usuário" é exibido nesta seção (Figura 22) como representante de todos os casos de uso CRUD. Vale lembrar que todas as ações de cadastro herdam de CrudAction, do pacote utilitário.

O modelo de navegação para o caso de uso "Autenticar Usuário" é mostrado na Figura 21.

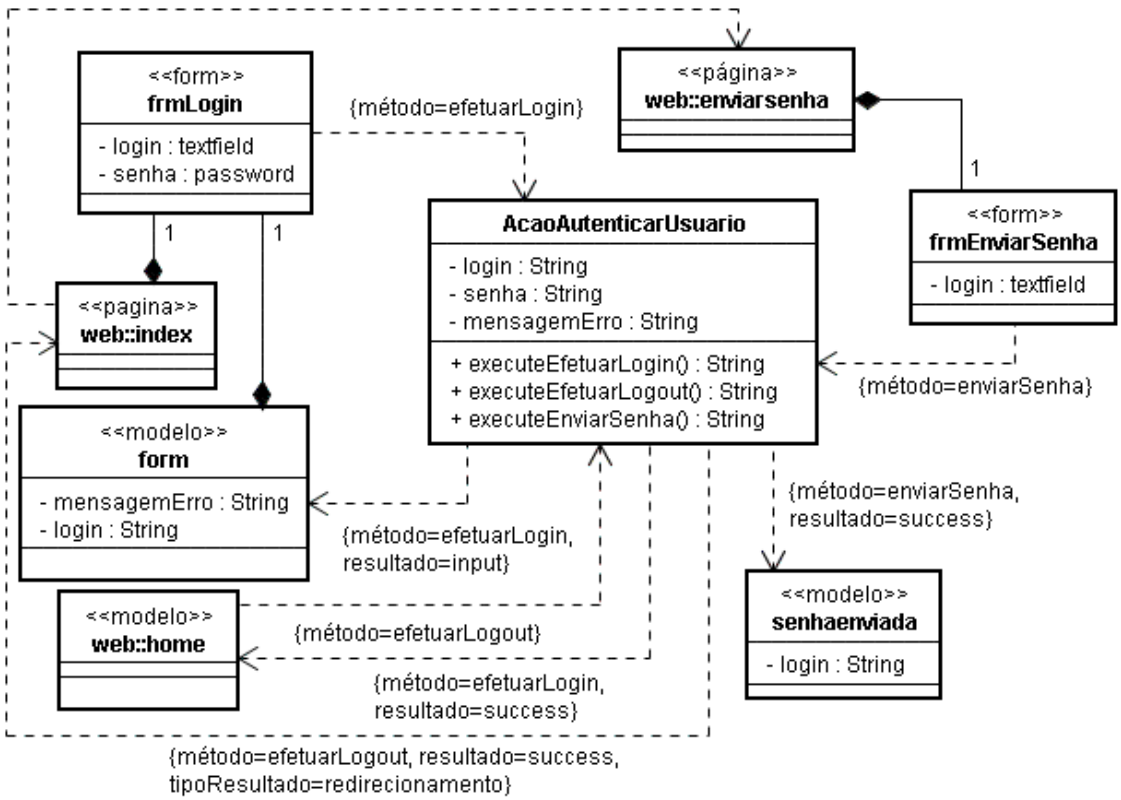

Figura 21 Modelo de navegação para o caso de uso "Autenticar Usuário"(SOUZA, 2007).

O diagrama acima mostra os elementos relevantes para a execução dos três cenários do caso de uso "Autenticar Usuário", descrito na seção 3.2.1.5, no que diz respeito às camadas de visão e controle. Descrevemos, a seguir, cada um dos cenários, colocando os elementos em ordem cronológica de participação.

- **Efetuar Login**: o visitante, antes de identificar-se, encontra-se na página inicial do portal, indicada por web::index. Essa página possui um formulário que oferece um campo de login e um campo senha que, quando preenchidos, são enviados para AcaoAutenticarUsuario que executará o método executeEfetuarLogin(), como especificado na restrição da dependência. Esse método deve chamar funcionalidades de negócio em classes de aplicação e retornar "input" ou "success", como visto nas duas relações de dependência que indicam efetuarLogin como método. O primeiro resultado indica um erro em um dos campos preenchidos, o que remete ao processamento de um modelo que apresentará novamente o mesmo formulário de login com uma mensagem de erro e o campo login preenchido (vide seus atributos).

O segundo resultado remete ao despacho do controle a uma página dinâmica web::home, que representa a página inicial para usuários corretamente identificados no sistema;

- **Efetuar Logout**: não existe uma página específica que chama esta funcionalidade. Ela pode ser chamada por um link que existe no menu de navegação e, portanto, encontra-se em várias páginas do site. Por isso não há um relacionamento de dependência apontando para a classe de ação e definindo o método como efetuarLogout. No entanto, existe a dependência saindo da classe de ação e indicando que no caso de resultado positivo ("success", único resultado possível), a classe de ação deve redirecionar para a página inicial do portal após executar a lógica de negócio referente ao cenário;
- **Enviar Senha**: da página inicial do site, há um *link* para uma página web::enviarsenha que pede ao usuário que preencha o seu login e envie a informação ao site. Essa informação vai parar na AcaoAutenticarUsuario, que executará o método executeEnviarSenha(), que deve providenciar o envio de senha e retornar "success". O *framework*, então, processa o resultado senhaenviada, que é um modelo que gera uma página HTML para informar que a senha foi enviada para o login especificado.

É interessante notar que, para o cenário de envio de senha, a implementação deve contemplar um curso alternativo que é quando o sistema não consegue enviar o e-mail com a senha do usuário. Cursos alternativos não são modelados e, portanto, a aparência é que só há um curso possível para o cenário "Enviar Senha", o que na implementação de fato não é o caso.

O modelo de navegação para o caso de uso "Cadastrar Usuário" é mostrado na Figura 22.

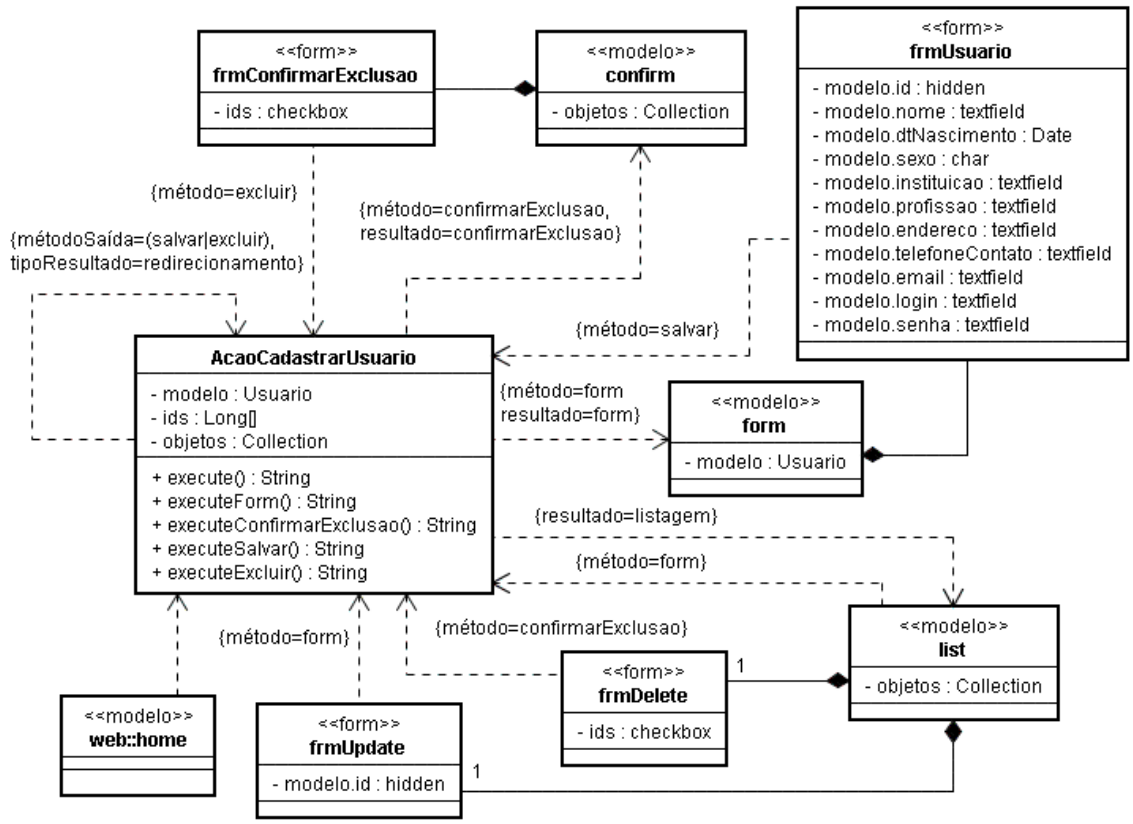

Figura 22 Modelo de navegação para o caso de uso "Cadastrar Usuário" (SOUZA, 2007).

O diagrama acima mostra os elementos relevantes para a execução dos quatro cenários do caso de uso "Cadastrar Usuário", descrito na seção 3.2.1.4, no que diz respeito às camadas de visão e controle. Descrevemos, a seguir, cada um dos cenários.

- **Consultar Usuário**: Como descrito no modelo de navegação para o caso de uso "Autenticar Usuário", o modelo web::home representa uma página que somente usuários registrados e identificados têm acesso. Nesta página há um *link* para executar o método *default* execute() da ação AcaoCadastrarUsuario. A requisição resulta na listagem dos usuários cadastrados na página do modelo list.
- **Criar Novo Usuário**: A partir do modelo list, a requisição para criação de um novo usuário (método=form) exibirá o modelo form. Esta página possui um formulário que oferece alguns campos que, quando preenchidos, são enviados para AcaoCadastrarUsuario que executará o método executeSalvar(), como especificado na restrição da dependência. O usuário será criado e o internauta será redirecionado para a listagem dos usuários cadastrados na página do modelo list.

- **Alterar Dados do Usuário**: A partir da listagem dos usuários cadastrados na página do modelo list, o internauta, após seleção de algum usuário, requisita a alteração dos dados desse usuário selecionado (método=form). A requisição exibirá o modelo form. Esta página possui um formulário que oferece alguns campos que, quando preenchidos, são enviados para AcaoCadastrarUsuario que executará o método executeSalvar(), como especificado na restrição da dependência. O usuário será editado e o internauta será redirecionado para a listagem dos usuários cadastrados na página do modelo list.
- **Excluir Usuário**: A partir da listagem dos usuários cadastrados na página do modelo list, o internauta, após seleção dos usuários a serem excluídos, requisita a exclusão desses usuários selecionados (método=confirmarExclusao). A requisição exibirá o modelo confirm. Esta página possui um formulário que lista os usuários a serem excluídos que, quando confirmados, são enviados para AcaoCadastrarUsuario que executará o método executeExcluir(), como especificado na restrição da dependência. Os usuários selecionados serão excluídos e o internauta será redirecionado para a listagem dos usuários cadastrados na página do modelo list.

5.5.2. Pacote br.ufes.inf.labes.portal.controleitens.controlador

O modelo de navegação para o caso de uso “Efetuar Busca de Itens” é mostrado na Figura 23.

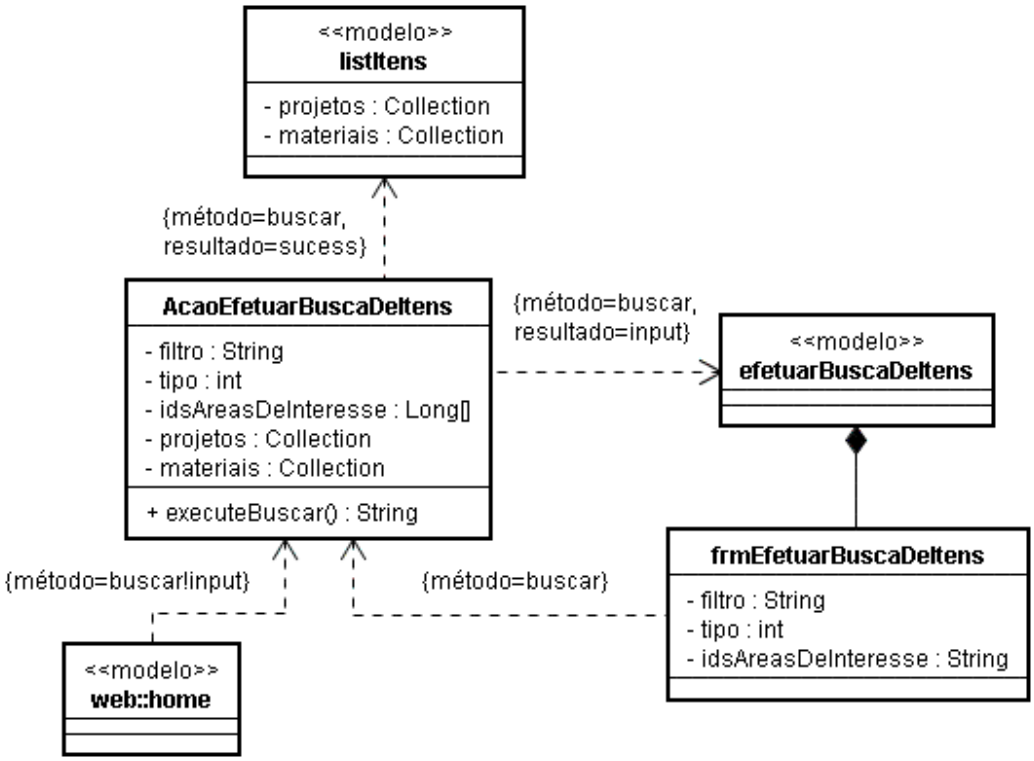

Figura 23 Modelo de navegação para o caso de uso "Efetuar Busca de Itens".

O diagrama mostra os elementos relevantes para a execução do caso de uso "Efetuar Busca de Itens", descrito na seção 3.2.2.3, no que diz respeito às camadas de visão e controle.

A requisição buscar!input mostra o modelo efetuarBuscaDeItens que contém o formulário frmEfetuarBuscaDeItens oferecendo campos para seleção das áreas de interesse, um campo para filtro de pesquisa nos resumos dos itens e um campo para seleção dos tipos de item (projeto, material, todos) a serem pesquisados. Quando o usuário requisita a pesquisa, os campos são enviados para AcaoEfetuarBuscaDeItens que executará o método executeBuscar(). O resultado é o despacho do controle a uma página dinâmica listItens, que representa a página onde serão listados os projetos e materiais encontrados, se existirem.

5.6. Modelo de Aplicação

O modelo de aplicação reúne as classes que implementam a lógica de negócio descrita nos casos de uso e suas dependências em relação aos demais modelos do sistema.

Basicamente ele mostra as classes da camada de aplicação, que são as responsáveis pela implementação da lógica dos casos de uso (lógica do negócio). Mostra também seus relacionamentos com as classes de controle (classes de ação que dependem delas) e persistência (de quais DAOs elas dependem).

Segundo o método FrameWeb, tais dependências entre diferentes camadas da arquitetura são satisfeitas por um *framework* de Injeção de Dependências e o modelo de aplicação pode servir de base para a configuração deste *framework*.

5.6.1. Pacote br.ufes.inf.labes.portal.controleusuario.aplicacao

O modelo de aplicação para uma parte do pacote `controleusuario` é mostrado na Figura 24. A classe de aplicação `AplCadastrarUsuario` é a única classe de aplicação de cadastro modelada e representa todas as outras, que funcionam da mesma forma. Todas as interfaces de aplicação de cadastro estendem `CrudService` (seção 5.2.2) e todas as implementações estendem `CrudServiceSupport`.

Como descrito na seção 5.2.2, por questões de brevidade, subtende-se que todas as classes de aplicação herdam automaticamente de uma classe base (`CrudServiceSupport`) e todas as interfaces herdam de uma interface base (`CrudService`).

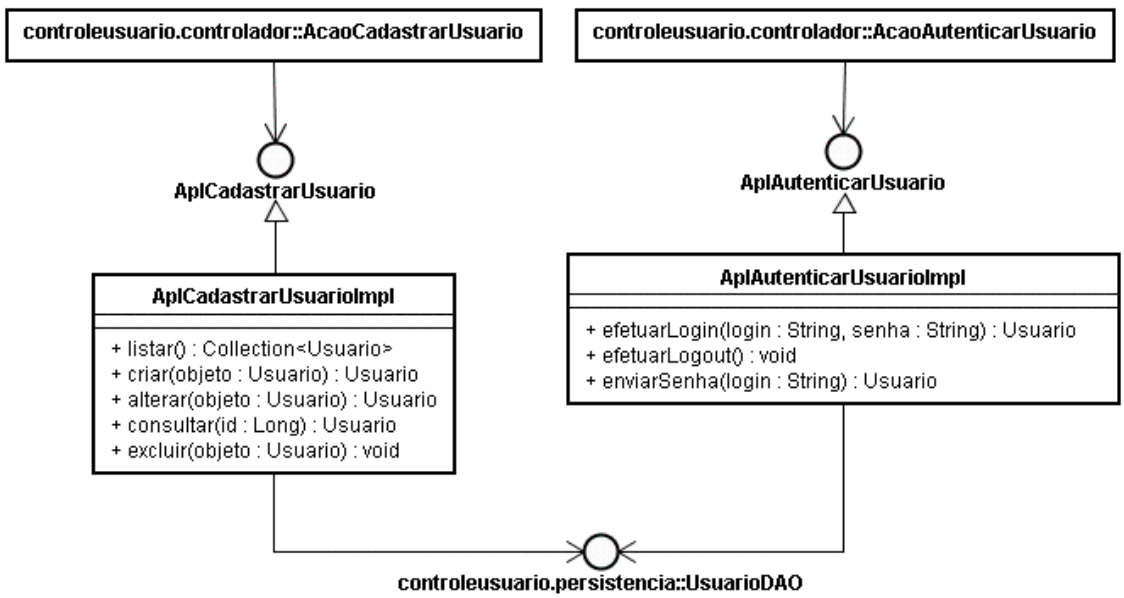

Figura 24 Modelo de aplicação do pacote controleusuario.

O diagrama acima mostra as classes de aplicação para os casos de uso “Cadastrar Usuário” e “Autenticar Usuário”. Como pode ser visto, a granularidade escolhida para as classes de aplicação foi uma classe para cada caso de uso, com um método para cada cenário. Um cenário pode, claro, abranger mais de um método, como é o caso do cenário “Excluir Usuário” que efetua primeiro o método consultar() para mostrar os dados dos objetos que devem ser excluídos, para depois, se confirmado pelo usuário, chamar o método excluir().

O diagrama mostra, também, as dependências. A classe de ação de cadastro de usuários depende da aplicação quase homônima e o mesmo ocorre com a ação de autenticação e sua respectiva classe de aplicação. Com relação aos DAOs, a aplicação de cadastro de usuário requer, obviamente, o DAO responsável pela persistência de objetos `Usuario`. A aplicação de autenticação de usuários precisa do mesmo DAO de `Usuario` para buscar os usuários pelo *login* e conferir se as senhas persistidas correspondentes a estes logins são iguais às senhas utilizadas pelo autor deste caso de uso.

5.6.2. Pacote br.ufes.inf.labes.portal.controleitens.aplicacao

Assim como ocorre no pacote `controleusuario`, por questões de brevidade, todas as interfaces de aplicação de cadastro no pacote `controleitens` estendem `CrudService` (apresentada na seção 5.2.2) e todas as implementações estendem `CrudServiceSupport`. O modelo de aplicação para o pacote `controleitens` é mostrado na Figura 25.

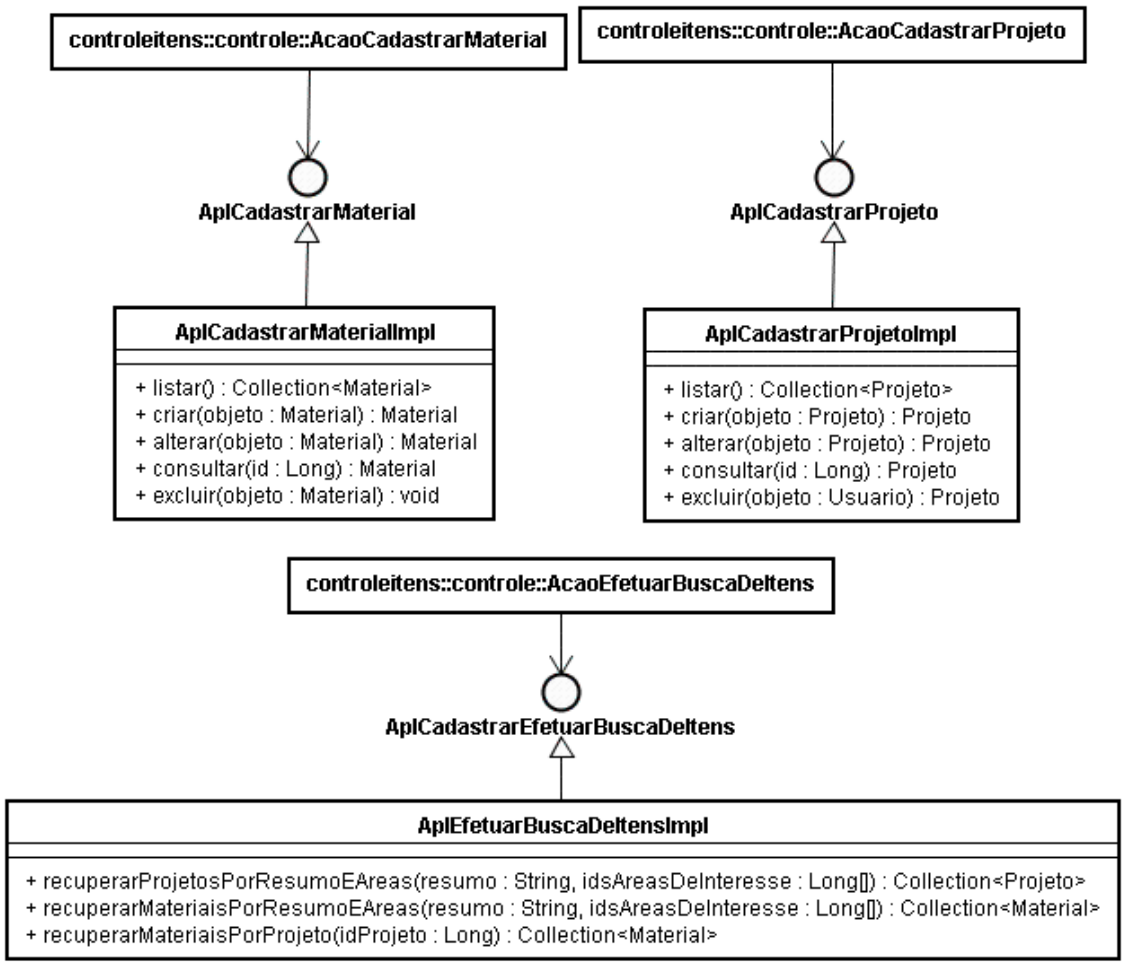

Figura 25 Modelo de aplicação do pacote controleitens.

5.7. Implementação

Esta seção procura ilustrar o funcionamento do Portal do LabES implementado, exibindo suas principais funcionalidades.

Ao acessarmos o endereço do Portal do LabES, a página principal do sistema é exibida (Figura 26). Podemos observar que há um menu lateral para acesso às funcionalidades do pacote Controle de Usuários e Controle de Itens.

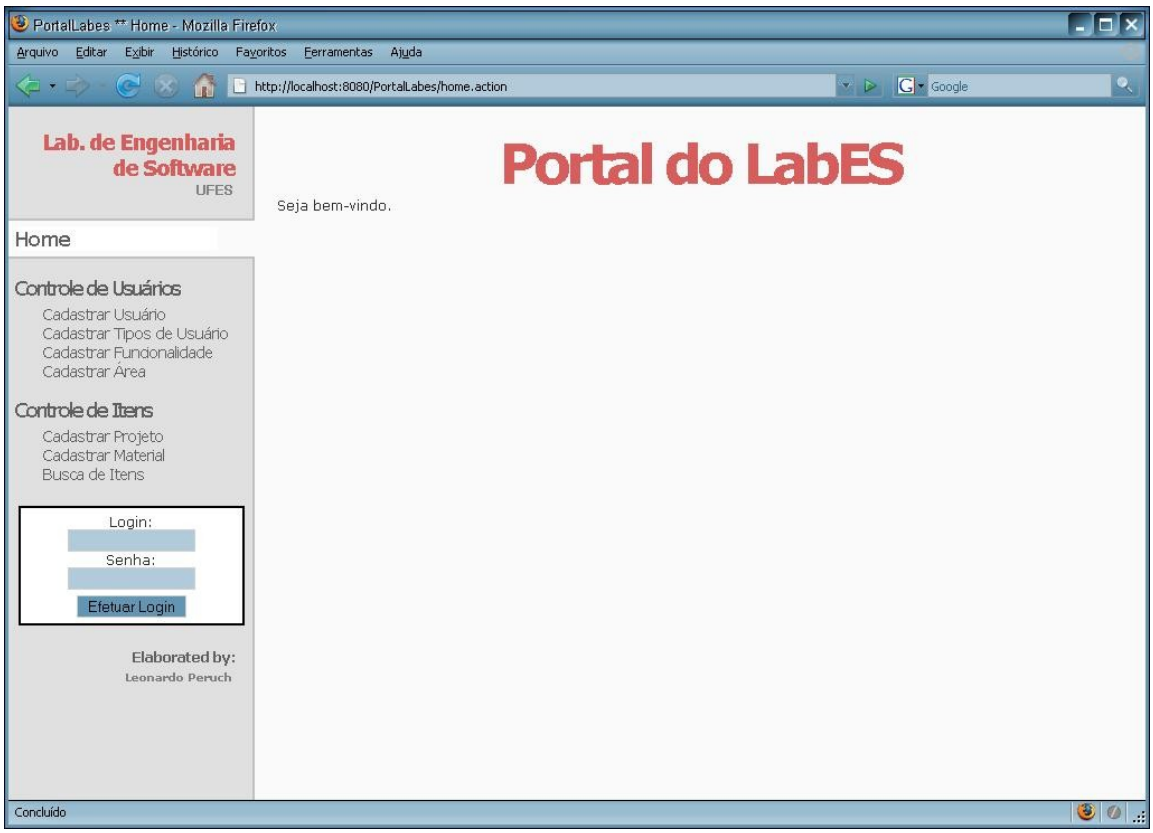

Figura 26 Tela inicial do Portal do LabES.

Todas as telas de cadastro estão no padrão CRUD (*Create, Retrieve, Update & Delete* – Criar, Consultar, Alterar e Excluir) e seguem a mesma aparência da tela de Cadastro de Usuário, exibida na Figura 27 como representante de todas telas de cadastro.

Para criar um novo usuário o internauta clica no ícone, que simboliza a criação de um novo usuário. Uma nova tela para edição de usuário(Figura 28) é exibida com todos os campos vazios.

Para editar algum usuário, o internauta clica no ícone que simboliza a edição do usuário correspondente. Uma nova tela para edição de usuário(Figura 28) será exibida com todos os campos preenchidos de acordo os dados do usuário a ser editado.

Através de *checkboxs*, o usuário pode selecionar um ou mais usuários para exclusão. Após clicar no ícone, que simboliza a exclusão, o internauta será redirecionado para a tela da Figura 29, onde, ao clicar no botão *Excluir*, será confirmada a exclusão.

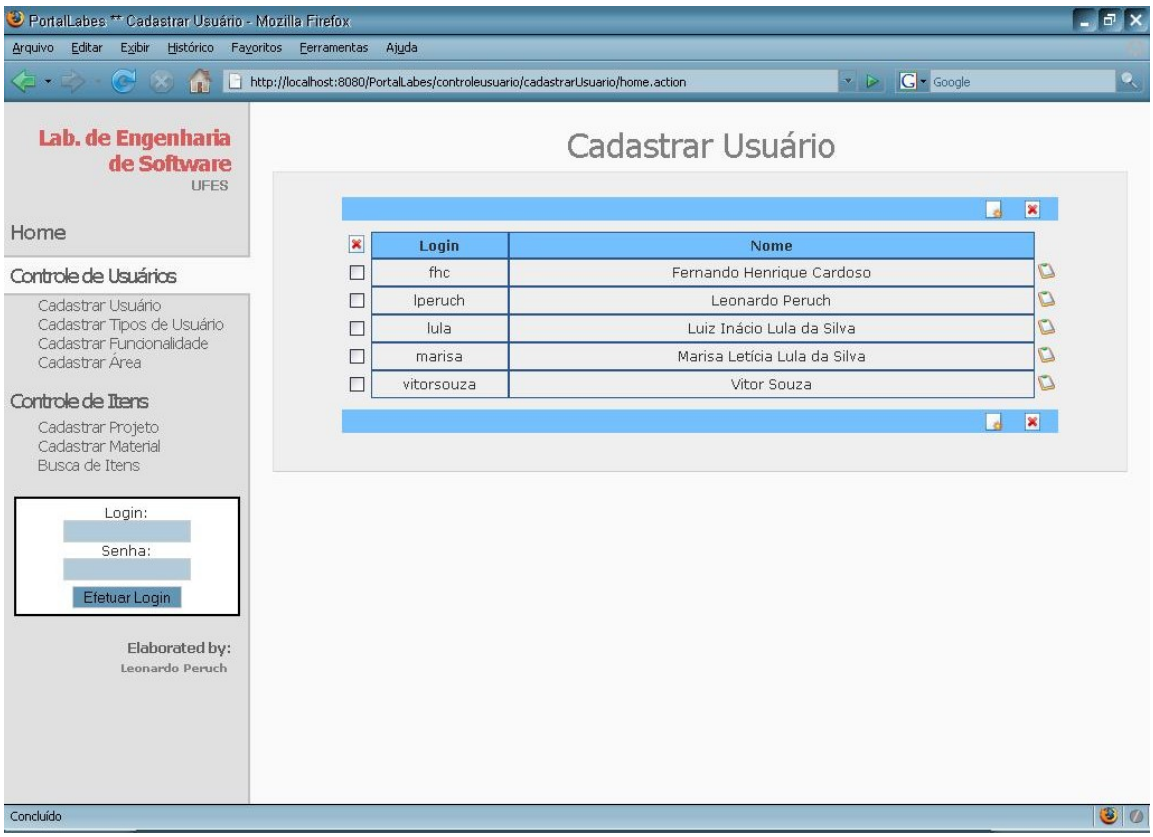

Figura 27 Tela de cadastro de usuário do Portal do LabES.

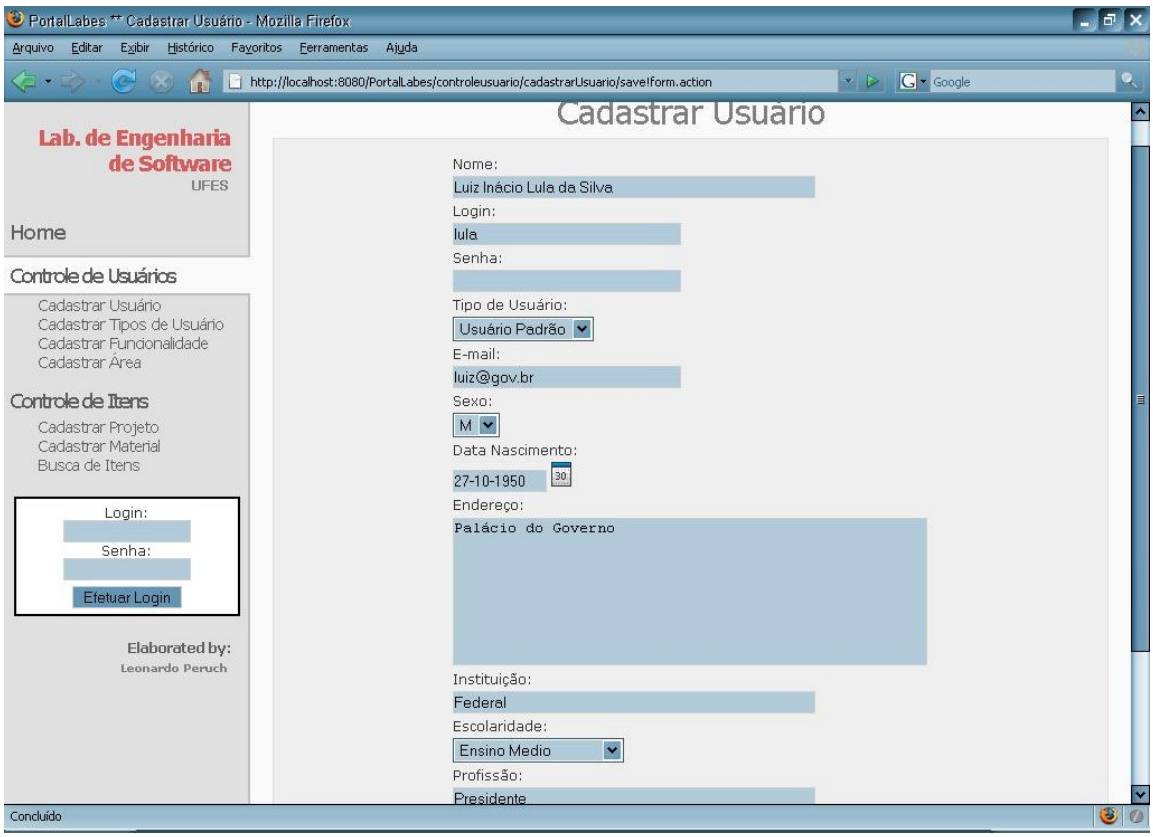

Figura 28 Tela de edição de usuário do Portal do LabES.

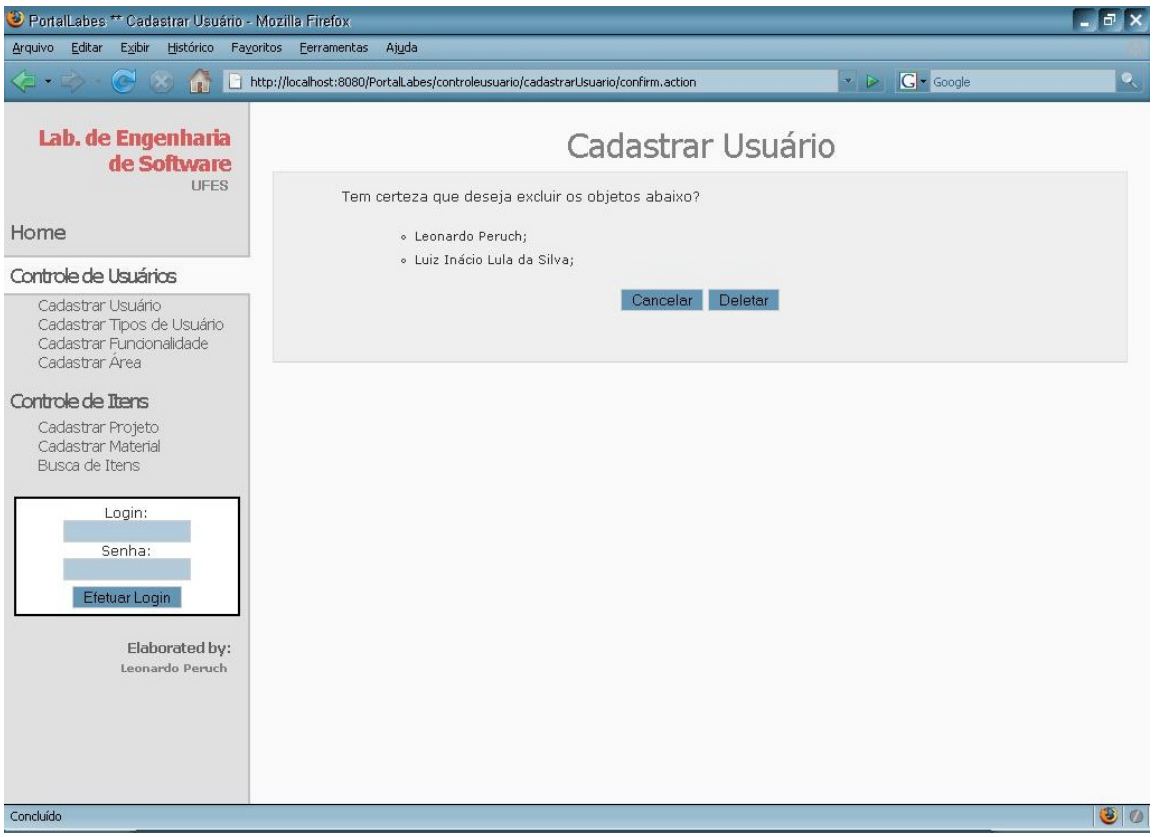

Figura 29 Tela de exclusão de usuário do Portal do LabES.

A Figura 30 exibe a tela responsável pelo caso de uso “Efetuar Busca de Itens” (seção 3.2.2.3). Ao clicar no botão *Pesquisar*, o sistema pesquisará os itens de acordo com os filtros. Após a pesquisa, o internauta será redirecionado para a tela de listagem dos itens encontrados (Figura 31).

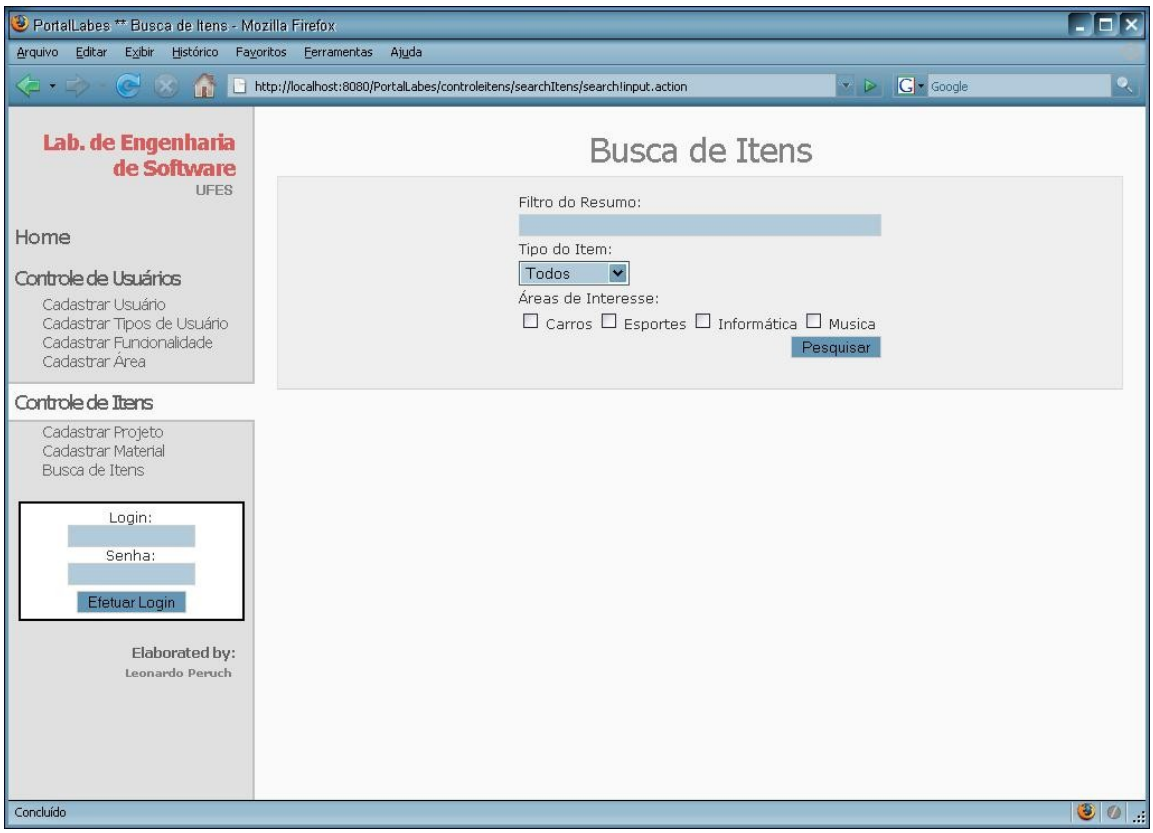

Figura 30 Tela para busca de itens do Portal do LabES.

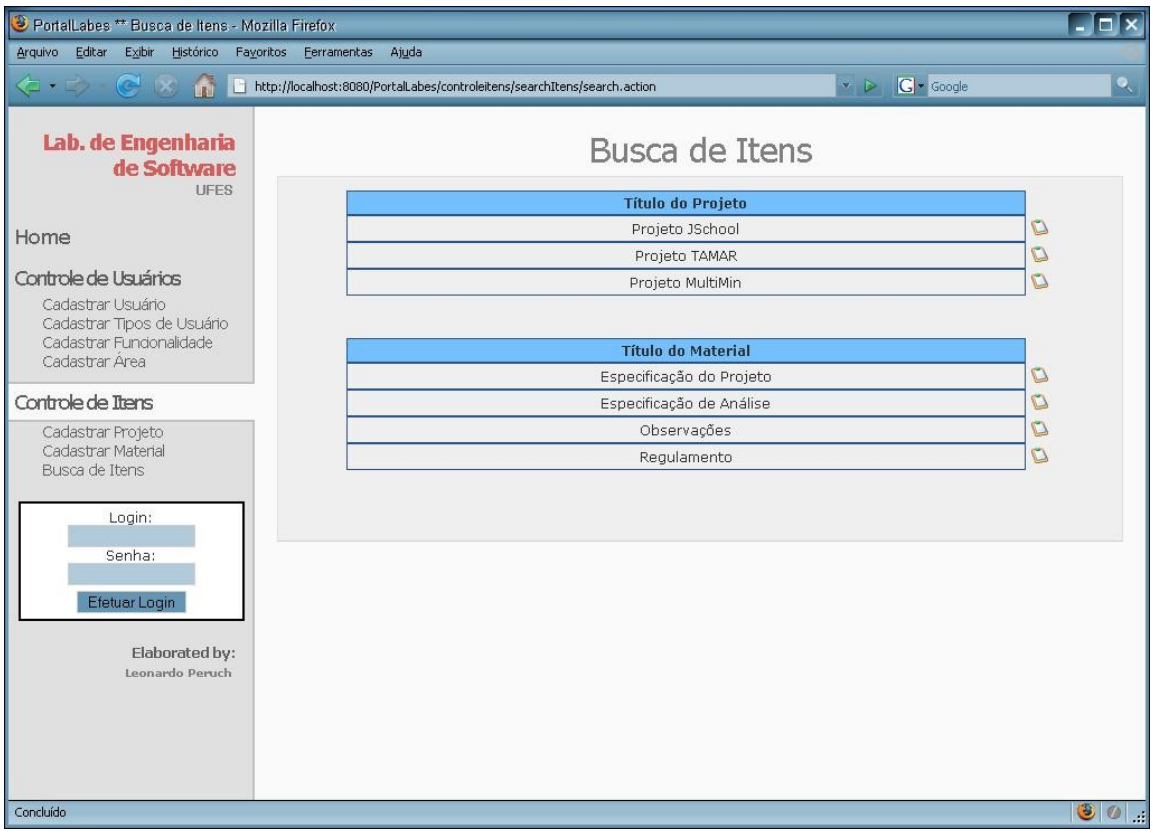

Figura 31 Tela de resultado da busca de itens do Portal do LabES.

Capítulo 6

Propostas de Melhorias para o Método FrameWeb

Após aplicação do método FrameWeb no desenvolvimento do Portal do Laboratório de Engenharia de Software (Portal do LabES), desenvolvemos o sistema em duas novas versões, utilizando diferentes *frameworks Web*, a saber Struts¹([6]) e ZK[7]. Struts¹ é um *framework* Controlador Frontal (seção 2.2.1) e o ZK é um *framework* RIA (*Rich Internet Application*, seção 2.2.2).

Na proposta original de FrameWeb, não foram feitos experimentos com diferentes instâncias de *frameworks*. Este capítulo sugere essa experiência, com o intuito de analisar a eficácia do método em diferentes contextos e sugerir alterações nos modelos para uma melhor adaptação a diferentes *frameworks*.

A seção 6.1 descreve o projeto do Portal do LabES desenvolvido com o *framework* Struts¹ e a seção 6.2 descreve o projeto desenvolvido o *framework* ZK.

As tecnologias utilizadas são as mesmas descritas na seção 5.1.1, porém, na seção 6.1, alteramos os *frameworks Web* de Struts² para Struts¹ e de Fremarker para JSP. Na seção 6.2 alteramos os *frameworks Web* de Struts², Freemarker e Sitemesh para o ZK.

6.1. Projeto e Implementação com *Framework* Struts¹

Esta seção descreve o projeto do Portal do LabES desenvolvido com o *framework Web* Struts¹.

A seção 6.1.1 detalha a Arquitetura do Sistema, listando as tecnologias escolhidas para implementação do sistema em código e os diferentes pacotes criados para modularização do software. A seção Figura 33 descreve as classes utilitárias necessárias no desenvolvimento do sistema. A seção 6.1.3 procura sugerir alterações nos modelos do

[6] http://struts.apache.org/1.x/
[7] http://www.zkoss.org/

método FrameWeb. A seção 6.1.4 procura ilustrar o funcionamento do Portal do LabES implementado.

6.1.1. Arquitetura do Sistema

As tecnologias utilizadas são as mesmas descritas na seção 5.1.1, porém alteramos os *frameworks Web* de Struts[2] para Struts[1] e de Fremarker para JSP a fim de analisar a adaptabilidade dos modelos do método FrameWeb a diferentes *frameworks*.

Nas próximas seções descreveremos um pouco sobre o Struts[1] e JSP.

6.1.1.1 *Framework* Struts[1]

O Struts[1] é um dos *frameworks* de maior sucesso da história da computação. É um dos principais responsáveis pela popularização do conceito de *framework* (seção 2.2), sendo um exemplo clássico de sucesso do modelo *open source* e certamente um dos mais populares dentre os *frameworks* para desenvolvimento *Web* (HUSTED et al, 2004).

O Struts[1] favorece o desenvolvimento de aplicações *Web* seguindo o padrão MVC, portanto, o incluímos na categoria dos *frameworks* MVC (seção 2.2.1).

Struts[1] provê seu próprio componente de controle e o integra com outras tecnologias para possibilitar o desenvolvimento dos componentes de Modelo e Visão. Com relação ao Modelo, Struts[1] pode interagir com qualquer tecnologia padrão de acesso a dados. Quanto à Visão, o modelo Struts[1] integra-se com JSP, descrito na próxima seção (HUSTED et al, 2004). A Figura 32 ilustra o funcionamento do Struts[1].

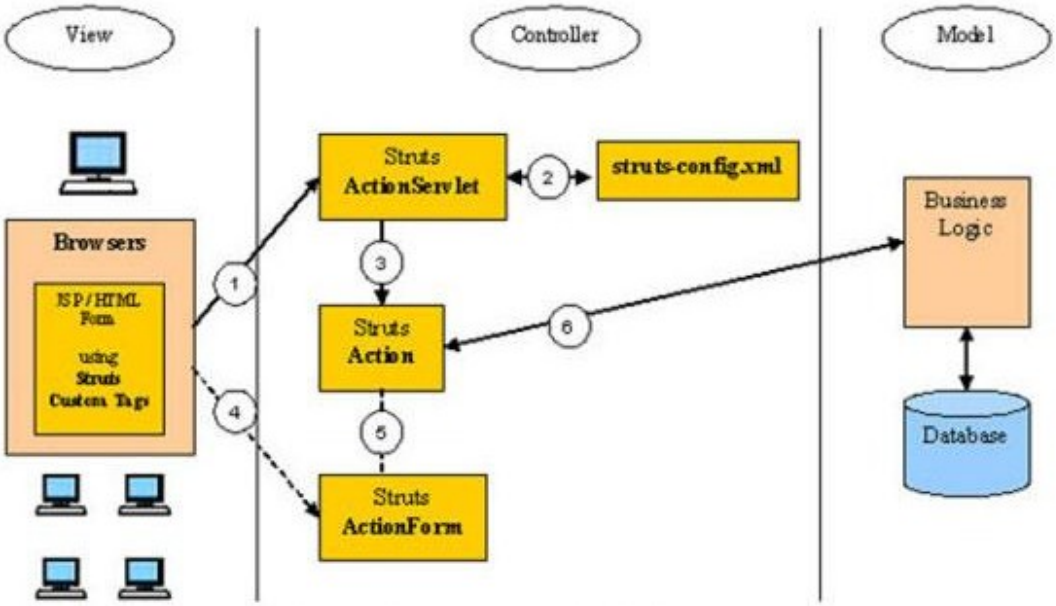

Figura 32 Funcionamento do Struts[1] (SPIELMAN, 2003).

As etapas estão descritas a seguir, de acordo com a numeração indicada na Figura 32:

1. Uma solicitação HTTP é enviada ao servidor. Essa solicitação normalmente é definida como “requisicao.do”, que é um nome lógico para a requisição do usuário.
2. A solicitação “requisicao.do” está mapeada no arquivo struts-config.xml. Nesse arquivo estão todas as definições do controlador do *framework*. O arquivo é então lido por um `ActionServlet` (que fará efetivamente o papel do controlador frontal) na inicialização da aplicação, criando, então, um banco de objetos com o arquivo de configuração. No arquivo de configuração são definidos os `Actions` (requisições dos usuários ou ações) para cada solicitação.
3. O `ActionServlet` (que faz o papel do controlador frontal) define o `Action` correspondente para a solicitação. Um `Action` pode validar a entrada de dados e acessar a camada de negócios para recuperar as informações nos bancos de dados e outros serviços de dados.
4. A requisição HTTP pode ser feita também através de um formulário HTML. Em vez de fazer com que cada `Action` retire os valores do campo da solicitação, o `ActionServlet` coloca a entrada em um `JavaBean` (objeto). Esses

`JavaBeans` são definidos como `FormBeans` no Struts[1] e estendem a classe `ActionForm` (*org.apache.action.ActionForm*) que é responsável por tratar os formulários. As instâncias dos `FormBeans` são preenchidas com os dados do formulário e validados (se necessário). Esse componente auxilia a classe de ação na captura dos dados digitados pelo usuário para que os mesmos sejam inseridos, geralmente, em um banco de dados na camada modelo.

5. O `Action` pode acessar o `FormBean` para efetuar qualquer operação.
6. O `Action` interage com a camada de negócio onde uma base de dados poderá ser atualizada.

Em geral, o Struts[1] não apresenta a resposta em si, mas envia a solicitação para outro recurso, como uma página JSP, descrito na próxima seção.

Para maiores informações sobre o *framework* Struts[1], acesse o endereço http://struts.apache.org/1.x/.

6.1.1.2 JSP

Páginas **JSP (*JavaServer Pages*)** são documentos que mesclam código Java e HTML, permitindo mostrar conteúdos dinâmicos e mais facilmente conteúdos estáticos.

Com JSP podemos criar aplicações *Web* que executam em vários servidores *Web*, de múltiplas plataformas, já que Java é, em essência, uma linguagem multiplataforma. As páginas JSP estão compostas de código HTML/XML misturado com etiquetas especiais para programar scripts de servidor em sintaxe Java. Portanto, poderemos escrever as páginas JSP em editores HTML/XML comuns.

6.1.1.3 Pacotes

O diagrama da Figura 33 mostra a divisão do sistema em módulos (pacotes Java). A divisão foi feita da mesma forma que na seção 5.1.2, porém adicionamos o pacote *`form`* contendo os formulários do Struts[1] (`FormBeans`).

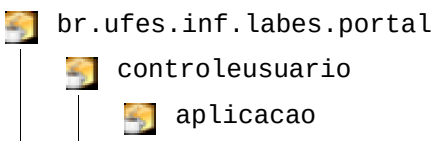

Figura 33 Pacotes do Portal do LabES desenvolvido com o *framework* Struts¹.

6.1.2. Utilitários

O pacote `util` contém classes utilitárias que são usadas por classes de diversos outros pacotes.

Os pacotes **net.java.dev.esjug.util.persistence** e **net.java.dev.esjug.util.domain** contêm classes utilitárias relacionadas à persistência dos objetos e aos objetos de domínio. Como só alteramos a camada de apresentação, substituição do *framework Web*, esses pacotes não sofreram alteração em relação à versão do Portal do LabES desenvolvida com o *framework Web* Struts² (seção Figura 14).

Já o pacote **net.java.dev.esjug.frameworks.crud** sofreu alteração para se adaptar ao Struts¹. A classe abstrata `CrudAction` foi adaptada para ser estendida por classes de ação do Struts¹ e não do Struts², como era na versão inicial (seção 5.2.2). O pacote contém classes do *framework* de CRUD que se integra com o **Struts¹** e o Spring *Framework* a fim de facilitar a construção de cadastros simples (também conhecidos por CRUD, abreviação de *Create, Retrieve, Update and Delete*).

6.1.3. Propostas para o método FrameWeb a partir de uma experiência com o *framework* Struts¹

Na fase de projeto o método que estamos analisando sugere alguns modelos, a saber:

Modelo de Domínio, Modelo de Persistência, Modelo de Navegação e Modelo de Aplicação. Mostramos alguns destes modelos no capítulo anterior quando abordamos a fase de projeto do Portal do LabES desenvolvido com o *framework Web* Struts[2].

Após experimento do método FrameWeb no desenvolvimento do Portal utilizando o *framework Web* Struts[1], concluímos que somente o Modelo de Navegação receberá alterações a fim de melhorar sua adaptação a diferentes *frameworks Web*. Segundo o método FrameWeb (SOUZA, 2007), existe uma correspondência entre os *frameworks* Controlador Frontal (seção 2.2.1) e o Modelo de Navegação. Então este modelo é o que deve mesmo sofrer alterações, já que o Struts[1] também está na categoria dos *frameworks* Controladores Frontais (*Front Controller*).

Adicionamos um novo estereótipo UML, representado por **<<formBean>>**, para representar a classe `FormBean` utilizada pelo Struts[1]. Essa classe é responsável por tratar os formulários HTML (seção 6.1.1.1) e suas instâncias são preenchidas com os dados do formulário em questão.

Os atributos do formBean representam parâmetros de entrada e saída relevantes à ação ao qual está associada. Se um atributo do formBean possui o mesmo nome de um atributo de um formulário HTML que está sendo submetido a ela, significa que os dados do formulário são injetados pelo *framework* no formBean e ficam disponíveis para o seu processamento na classe de ação. Analogamente, quando o formBean e a página que apresenta seu resultado possuem atributos homônimos, indica que a página exibe essa informação, obtendo-a do formBean.

Como exemplo temos a Figura 34 que mostra em um único Modelo de Navegação ações para os três cenários do caso de uso “Autenticar Usuário” (seção 3.2.1.5) do subsistema *ControleUsuario* do Portal do LabES.

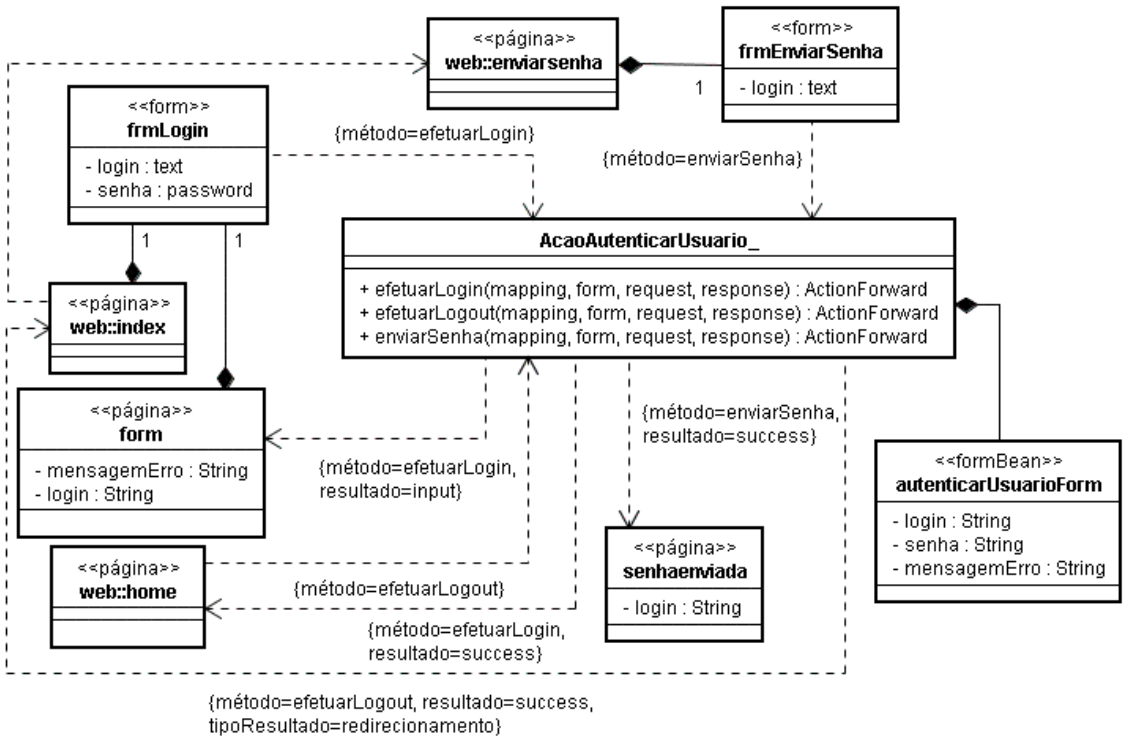

Figura 34 Modelo de Navegação para o caso de uso "Autenticar Usuario" para Struts[1].

O Modelo de Navegação do caso de uso "Autenticar Usuário" mostra-nos que na página inicial do sistema (representada pela página **web::index**, convenção estabelecida para o projeto do Portal do LabES) possui um formulário com os campos **login** e **senha**. Esses dados são submetidos para a classe de ação para evocação do método **efetuarLogin()** que deverá acessar a camada de Lógica de Negócio para execução do caso de uso.

Como os campos submetidos à classe de ação e os atributos do formBean **autenticarUsuarioForm** são homônimos, indica que os dados serão salvos no formBean. No caso das informações estarem corretas (**result = success**), o usuário é enviado para a página **web::home**, que representa a área de acesso restrito dos usuários autenticados. Caso contrário, o resultado retornará o resultado **"input"** (que indica problemas nos dados fornecidos) e o cliente será direcionado a uma página que exibirá novamente o formulário de *login* juntamente com a mensagem do erro ocorrido.

No caso de esquecimento da senha, o usuário pode acionar um *link* na página principal para uma página que contém um formulário que pede seu login e o envia à ação **enviarSenha**. A classe de ação solicita à camada de Lógica de Negócio o envio da senha para o usuário por e-mail. Como o campo submetido à classe de ação e o atributo do

formBean **autenticarUsuarioForm** são homônimos, indica que o dado será salvo no formBean. O sistema exibirá como resultado, uma página informando que a senha foi submetida. Para efetuar *logout*, basta que o usuário acione **efetuarLogout** que o sistema redirecionar-lhe-á novamente à página inicial, como se fosse um visitante comum.

6.1.4. Implementação

Esta seção procuraria ilustrar o funcionamento do Portal do LabES implementado com o *framework Web* Struts¹, porém a aparência da versão do sistema desenvolvido com Struts¹ é a mesma da versão do sistema desenvolvido com Struts² (seção 5.7).

6.2. Projeto e Implementação com *Framework* ZK

Esta seção descreve o projeto do Portal do LabES desenvolvido com o *framework Web* ZK[8], um *framework Web* AJAX Open Source.

A seção 6.2.1 detalha a Arquitetura do Sistema, listando as tecnologias escolhidas para implementação do sistema em código e os diferentes pacotes criados para modularização do software. A seção Figura 37 descreve as classes utilitárias necessárias no desenvolvimento do sistema. A seção 6.2.3 procura sugerir alterações nos modelos do método FrameWeb.

6.2.1. Arquitetura do Sistema

As tecnologias utilizadas são as mesmas descritas na seção 5.1.1, porém alteramos os *frameworks Web* de Struts², Freemarker e Sitemesh para ZK a fim de analisar a adaptabilidade dos modelos do método FrameWeb a diferentes *frameworks*.

A seção 6.2.1.1 descreve o *framework Web* ZK

6.2.1.1 *Framework* ZK

ZK é um *framework* que utiliza um sistema de eventos baseado em AJAX. É um *framework Web* AJAX *Open Source*, que permite o desenvolvimento de interfaces ricas para aplicações *Web* (RIA - *Rich Internet Applications*, seção 2.2.2) com pouca

[8] http://www.zkoss.org/

programação e um custo de desenvolvimento reduzido. O desenvolvimento de aplicações *Web* utilizando o *framework* ZK simplifica bastante a programação da interface com o usuário, devido aos vários recursos existentes, além de aprimorar o visual gráfico das aplicações.

Simplicidade é um dos valores de base do ZK. Outra das facilidades que encontramos, é a disponibilidade com que a sua equipe de desenvolvedores estão dispostos a ajudar. ZK também oferece uma documentação bastante detalhada e atualizada constantemente.

ZK inclui um mecanismo baseado em AJAX para automação interativa, um conjunto rico de componentes baseados em XUL (*XML User Interface Language*) para enriquecer a usabilidade e uma linguagem de marcação para simplificar o desenvolvimento.

O *framework* inclui vários componentes XUL e XHTML (*eXtensible Hypertext Markup Language*) para facilitar a criação de aplicações dinâmicas para a Internet sem a necessidade de programação em Java Script.

Os arquivos XUL criam componentes UI (*User Interface*) como listas, janelas, diálogos, modais, botões e outros componentes comuns, tomando as aplicações para a Internet interativa, similar em muitos aspectos à programação para desktop (baseado em eventos).

O mecanismo baseado em AJAX é constituído de três partes: ZK Loader, ZK AU Engine[9] e ZK Cliente Engine. A Figura 36 ilustra a arquitetura do *framework* ZK.

Baseado na requisição do usuário, o ZK Loader carrega um ZK Page (arquivo .ZUL), a intepreta, e renderiza o resultado para uma página HTML em resposta à URL requisitada. Uma ZK Page é escrita em uma linguagem de marcação chamada ZUML (ZK User Interface Markup Language). ZUML é como um HTML, ele é usado para descrever quais componentes serão criados e como eles são apresentados na visualização. Essa representação também pode ser feita em classes Java.

Para exemplificar uma ZK Page (arquivo .ZUL), segue abaixo o código contido em uma página HelloWorld.zul.

```
<window title="Hello" border="normal">
        Hello World!
</window>
```

[9] Asynchronous Update (Atualização Assíncrona)

Uma requisição, utilizando a URL http://localhost/minhaaplicacao/HelloWorld.zul, exibirá a página da Figura 35 no navegador de Internet *(browser)*.

Figura 35 Página Hello World construída com o *framework* ZK.

Em uma página ZUML, elementos em XML descrevem quais componentes serão criados. No exemplo, existe o componente `Window` (`org.zkoss.zul.Window`). Os atributos do XML são utilizados para configurar valores das propriedades do componente `Window`. No exemplo, criamos um componente `Window` configurando as propriedades *title* (título) e *border* (borda) para "Hello" e "normal", respectivamente.

Normalmente, uma ZK Page utiliza componentes `Windows` customizados para captura dos eventos realizados pelo usuário, a fim de que haja uma comunicação com a base de dados, geralmente, um banco de dados. Este componente deve estender a classe base `Window` (`org.zkoss.zul.Window`) do *framework*.

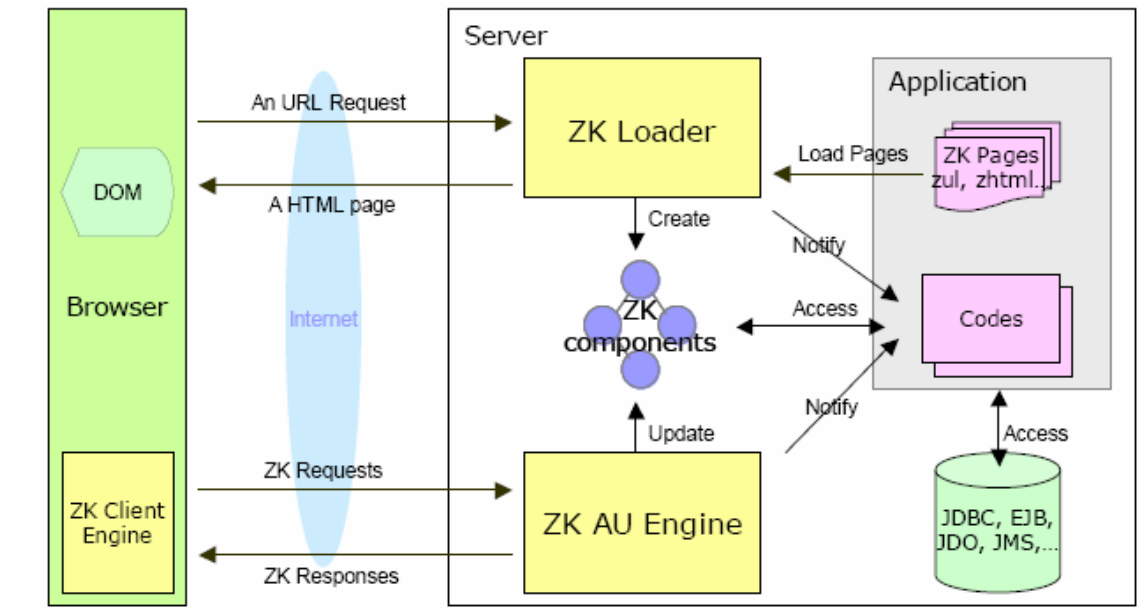

Figura 36 Arquitetura do *framework* ZK[10].

[10] Imagem disponível em http://www.zkoss.org/doc/ZK-devguide.pdf

O ZK AU Engine e o ZK Client Engine trabalham como arremessador (*picher*) e receptor (*catcher*), ou seja, enviando e recebendo informações. Eles entregam os eventos que aconteceram no *browser* para a aplicação que está rodando no servidor, e atualizam a árvore DOM no *browser* com um componente que é manipulado por esta aplicação. O DOM (*HTML Document Object Model*) representa o documento como uma estrutura em árvore, ou seja, toda ramificada, com nós representando os elementos do HTML, seus atributos e o texto.

Segue abaixo o fluxo de execução:

1. Quando o usuário requisita uma URL do sistema ou clica em algum *hiperlink*, uma requisição é enviada para o servidor *Web*. ZK Loader é, então, invocado para servir essa requisição;
2. ZK Loader carrega a página específica e a interpreta para criar os componentes apropriados;
3. Após interpretar a página toda, ZK Loader renderiza o resultado para uma página HTML. A página HTML é enviada para o browser "acompanhada" pelo ZK Client Engine[11];
4. ZK Client Engine é responsável por detectar todos eventos realizados pelo usuário, tais como arrasto do *mouse* ou algum *click*. Uma vez detectado, ele notifica o ZK AU Engine enviando um ZK requests;
5. Através do ZK Resquest recebido do ZK Cliente Engine, AU Engine atualiza o componente correspondente, se necessário;
6. Se a aplicação escolher mudar o conteúdo de componentes, adicionar ou mover componentes, AU Engine envia o novo conteúdo para alteração do componente pelo Cliente Engine através do envio de um ZK Response;
7. O ZK Response é um comando para instruir o Cliente Engine de como atualizar a árvore DOM apropriadamente.

Para maiores informações sobre o *framework* ZK, acesse o endereço http://www.zkoss.org/.

[11] ZK Client Engine é escrito em JavaScript. Browsers podem colocar o ZK Client Engine no cache, então o engine só é enviado na primeira vez.

6.2.1.2 Pacotes

O diagrama da Figura 37 mostra a divisão do sistema em módulos (pacotes Java). A divisão foi feita da mesma forma que na seção 5.1.2, porém adicionamos o pacote br.ufes.inf.labes.portal.*controlador*.`componentes` contendo todos os componentes utilizados no desenvolvimento do sistema.

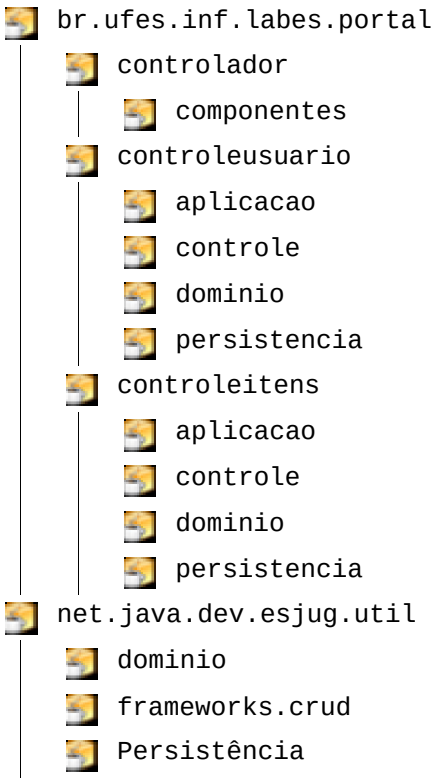

Figura 37 Pacotes do Portal do LabES desenvolvido com o *framework* ZK.

6.2.2. Utilitários

O pacote `util` contém classes utilitárias que são usadas por classes de diversos outros pacotes.

Os pacotes **net.java.dev.esjug.util.persistence** e **net.java.dev.esjug.util.domain** contêm classes utilitárias relacionadas à persistência dos objetos e aos objetos de domínio. Como só alteramos a camada de apresentação (substituição do *framework Web*), esses pacotes não sofreram alteração em relação a versão do Portal do LabES desenvolvida com o *framework Web* Struts² (seção Figura 14).

Já o pacote **net.java.dev.esjug.frameworks.crud** sofreu alteração para se adaptar ao

framework ZK. A classe abstrata `CrudAction` foi removida porque a arquitetura do *framework* ZK não utiliza o conceito de `Actions` (Ações). Uma nova classe abstrata (`CrudWindow`) foi criada a fim de representar uma `Window` utilizada com recursos a serem utilizados nas telas de cadastro.

O pacote contém classes do *framework* de CRUD que se integra com o **ZK** e o Spring *Framework*, a fim de facilitar a construção de cadastros simples (também conhecidos por CRUD, abreviação de *Create, Retrieve, Update and Delete*).

6.2.3. Propostas para o método FrameWeb a partir de uma experiência com o *framework* ZK

Após experimento do método FrameWeb no desenvolvimento do Portal utilizando o *framework* ZK, concluímos que somente o Modelo de Navegação receberá alterações a fim de melhorar sua adaptação a diferentes *frameworks*.

Segundo o método FrameWeb (SOUZA, 2007), existe uma correspondência entre os *frameworks* Controlador Frontal (seção 2.2.1) e o Modelo de Navegação. Como substituímos o *framework* Controlador Frontal por um *framework* que não se encaixa nesta categoria, porém é aplicado à mesma camada (camada de apresentação), esse modelo é o que sofrerá alterações.

Segue abaixo as melhorias no Modelo de Navegação:

1. O estereótipo UML <<página>> passa a representar um ZK Page (arquivo .zul) utilizado pelo *framework* para representar páginas *Web* a serem exibidas ao usuário;
2. Substituímos o estereótipo UML <<form>> por <<Window>> para representar o elemento XML `Window`, presente nas páginas (arquivos .zul). Dentro de uma `Window` podemos encontrar componentes do framework ZK, como listas (*listbox*), input (*textbox*), botões (*button*), menu (*menu*), datas (*datebox*), entre outros. Os componentes são representados como atributos da classe que utiliza o estereótipo <<Window>>.
3. Quando nenhum estereótipo for definido, estaremos representando uma classe Java herdada da classe `Window`, para o qual o *framework* ZK delega a execução dos eventos. Todos os componentes também estarão representados como atributos dessa classe.

Como exemplo temos a Figura 38 que mostra em um único Modelo de Navegação o fluxo para os três cenários do caso de uso "Autenticar Usuário" (seção 3.2.1.5) do subsistema *ControleUsuario* do Portal do LabES.

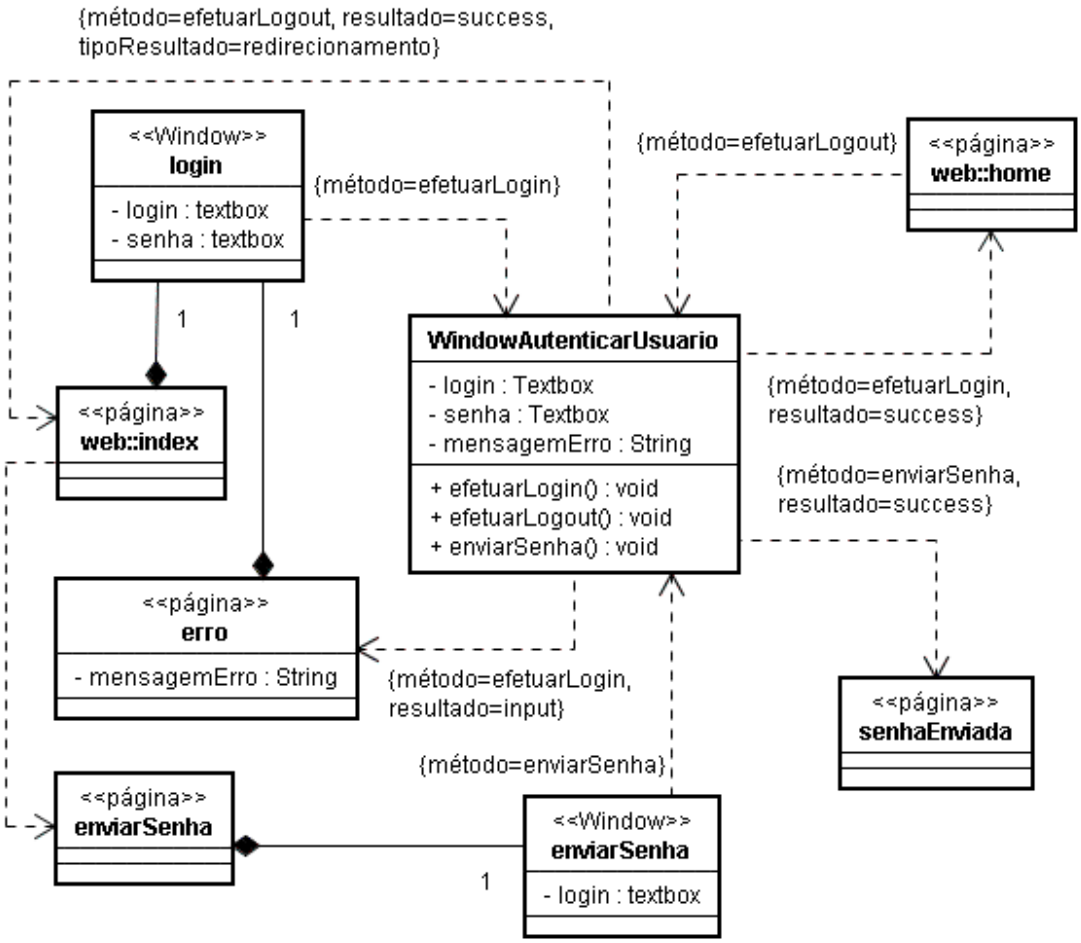

Figura 38 Modelo de Navegação para o caso de uso "Autenticar Usuario" para ZK

O Modelo de Navegação do caso de uso "Autenticar Usuário" mostra-nos que a página inicial do sistema (representada pela página **web::index**, convenção estabelecida para o projeto do Portal do LabES) possui uma `Window` com os campos **login** e **senha**. Esses dados são submetidos para a classe **WindowAutenticarUsuario** para evocação do método **efetuarLogin()** que deverá acessar a camada de Lógica de Negócio para execução do caso de uso. No caso das informações estarem corretas (**result = success**), o usuário é enviado para a página **web::home**, que representa a área de acesso restrito dos usuários autenticados. Caso contrário, o resultado retornará o resultado **"input"** (que indica problemas nos dados fornecidos) e o cliente será direcionado a uma página que exibirá

novamente o `Window` de *login* juntamente com a mensagem do erro ocorrido.

No caso de esquecimento da senha, o usuário pode acionar um *link* na página principal para uma página que contém um `Window` que pede seu login e o envia à ação **enviarSenha**. A classe **WindowAutenticarUsuario** solicita à camada de Lógica de Negócio o envio da senha para o usuário por email, exibindo como resultado uma página informando que a senha foi submetida. Para efetuar *logout*, basta que o usuário acione **efetuarLogout** que o sistema redirecionar-lhe-á novamente à página inicial **web::index**, como se fosse um visitante comum.

6.2.4. Implementação

Esta seção procura ilustrar o funcionamento do Portal do LabES implementado com o *framework Web* ZK.

Ao acessarmos o endereço do Portal do LabES, a página principal do sistema é exibida (Figura 39). Há um menu superior para acesso às funcionalidades de cadastros, relacionadas aos pacotes Controle de Usuário e Controle de Itens, juntamente com um sub-menu para Pesquisa e Ajuda.

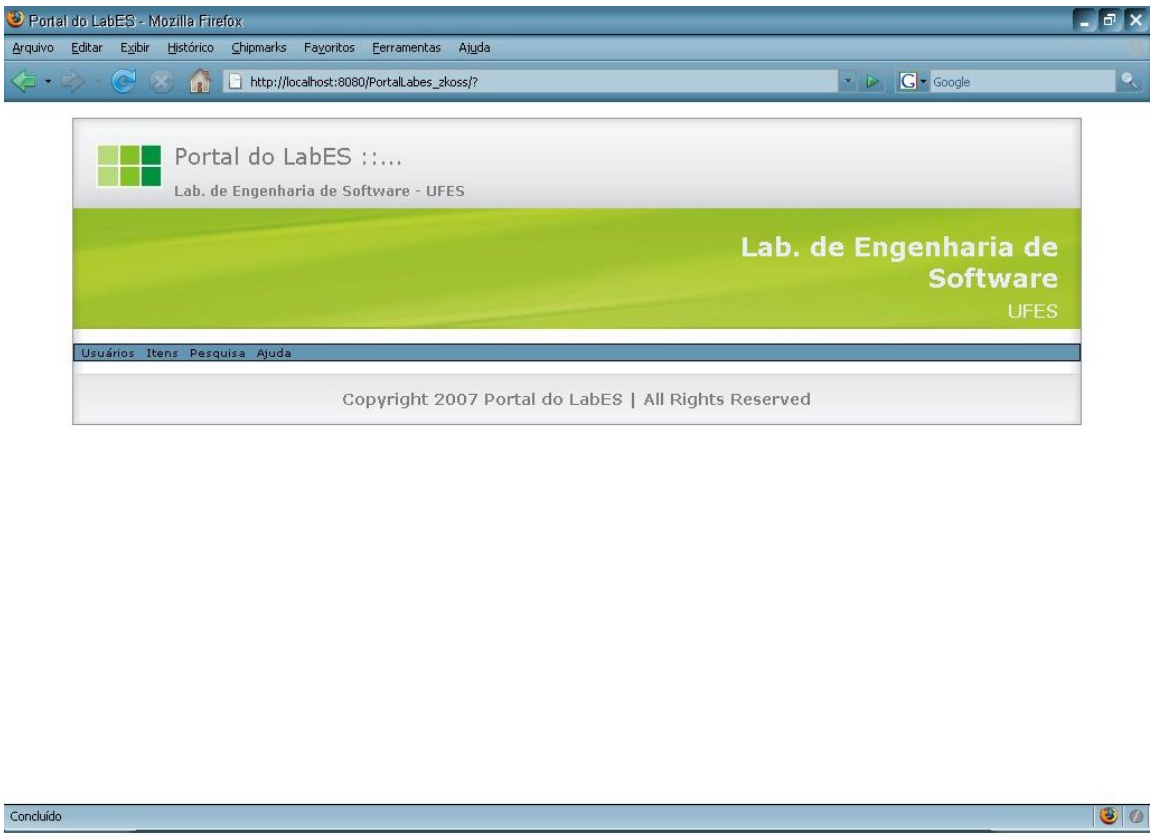

Figura 39 Tela inicial do Portal do LabES.

Todas as telas de cadastro estão no padrão CRUD (*Create, Retrieve, Update & Delete* – Criar, Consultar, Alterar e Excluir) e seguem o mesmo leiaute da tela de Cadastro de Usuário, exibido na Figura 40 como representante de todas telas de cadastro.

Ao clicar no botão *Cadastrar*, o internauta será redirecionado para a tela de edição de usuário (Figura 41) com todos os campos vazios.

Ao clicar no botão *Editar*, o internauta será redirecionado para a tela de edição de usuário (Figura 41) com todos os campos preenchidos de acordo os dados do usuário a ser editado.

Através da listagem de usuários, o usuário poderá selecionar um ou mais usuários para exclusão. Após clicar no botão *Excluir*, o internauta será redirecionado para a tela da Figura 42, onde, ao clicar no botão *Sim*, será confirmada a exclusão.

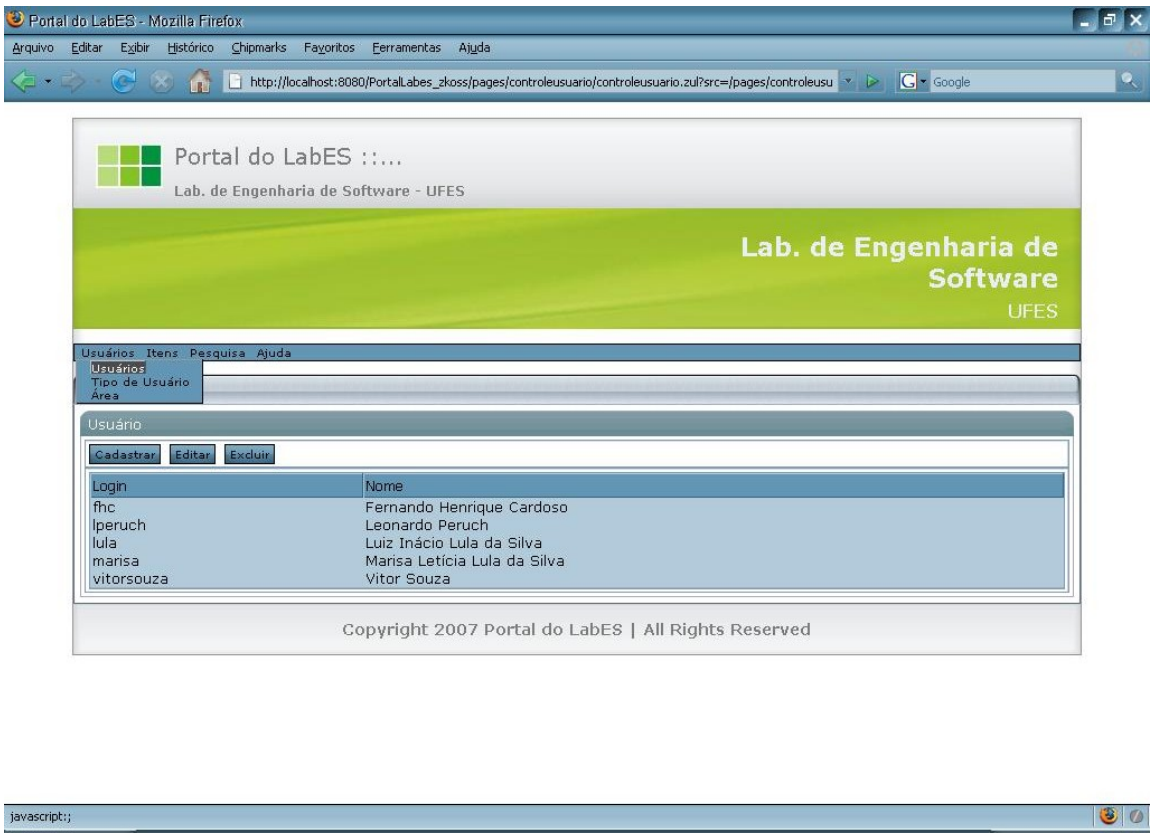

Figura 40 Tela de cadastro de usuário do Portal do LabES.

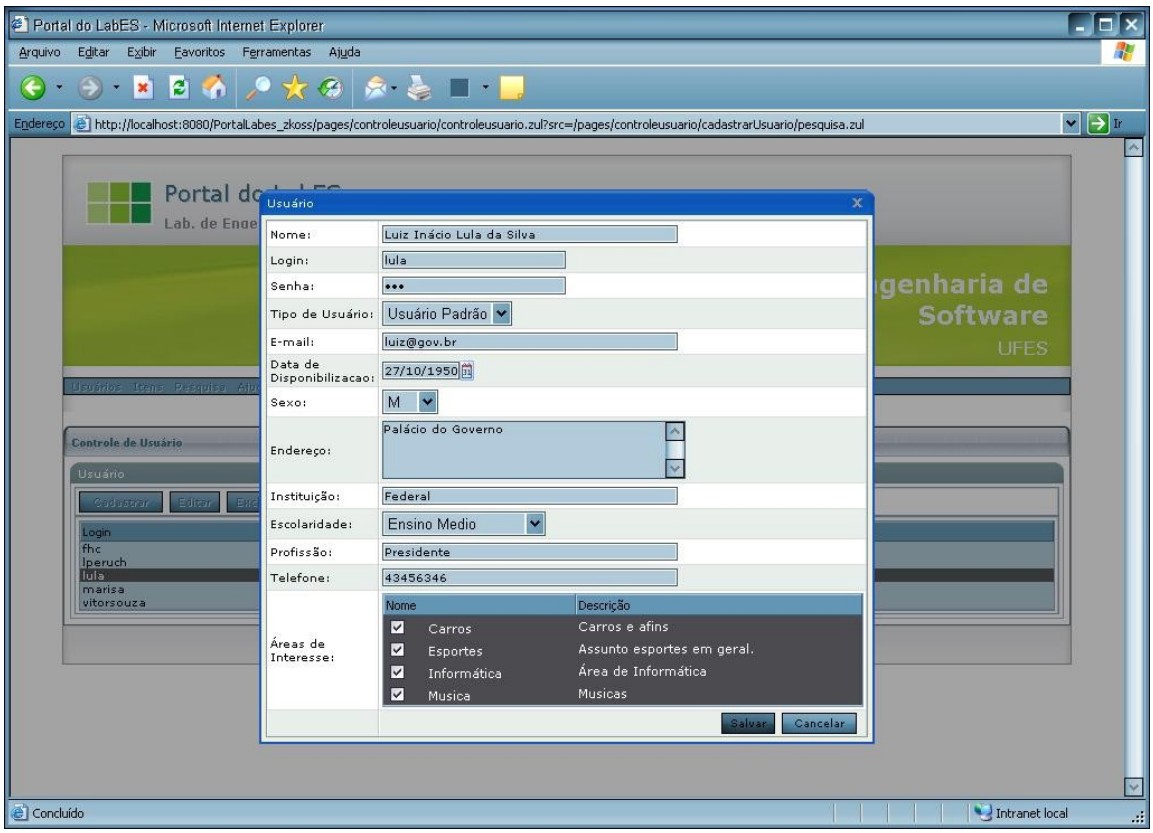

Figura 41 Tela de edição de usuário do Portal do LabES.

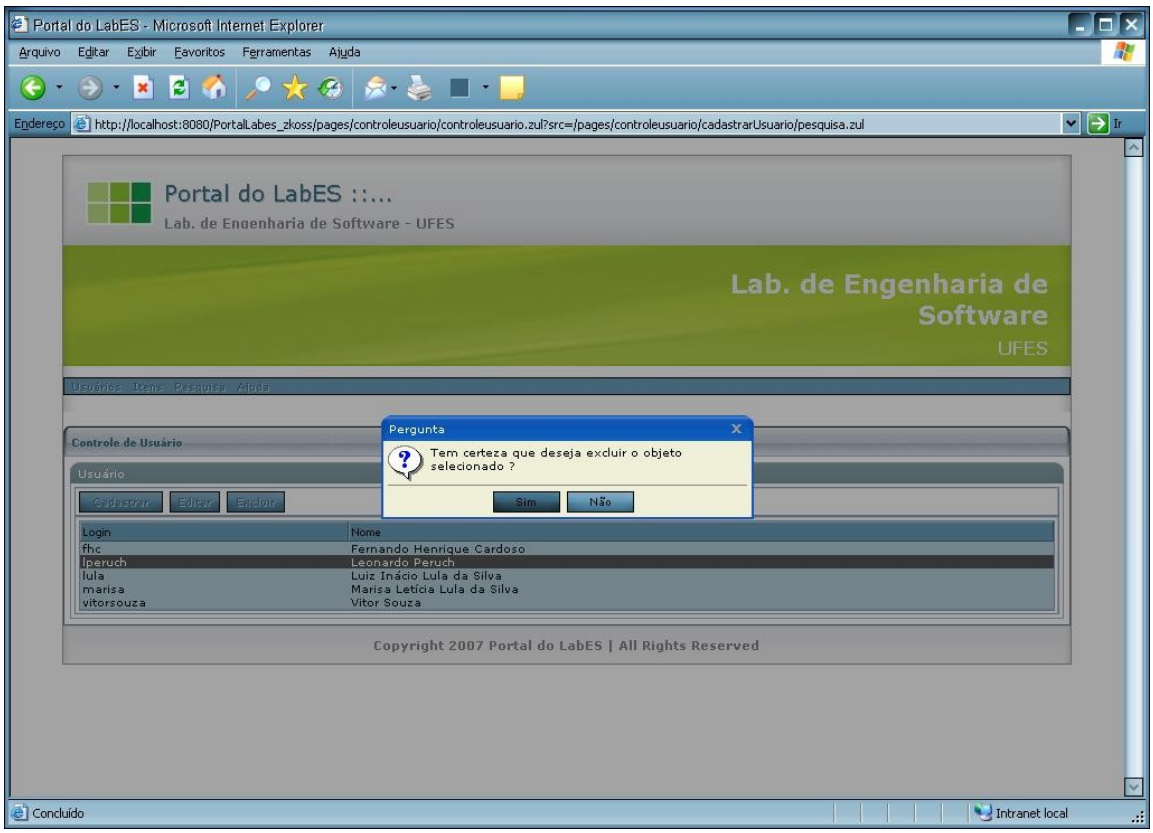

Figura 42 Tela de exclusão de usuário do Portal do LabES.

A Figura 43 exibe a tela responsável pelo caso de uso “Efetuar Busca de Itens” (seção 3.2.2.3). Ao clicar no botão *Pesquisar* o sistema pesquisará os itens de acordo com os filtros. Após a pesquisa, o internauta será redirecionado para a tela de listagem dos itens encontrados (Figura 44). Caso queira visualizar algum item, basta selecioná-lo e clicar no botão *Abrir Item Selecionado*. Para itens do tipo Material, será exibida a tela da Figura 45 e para itens do tipo Projeto será exibida a tela da Figura 46.

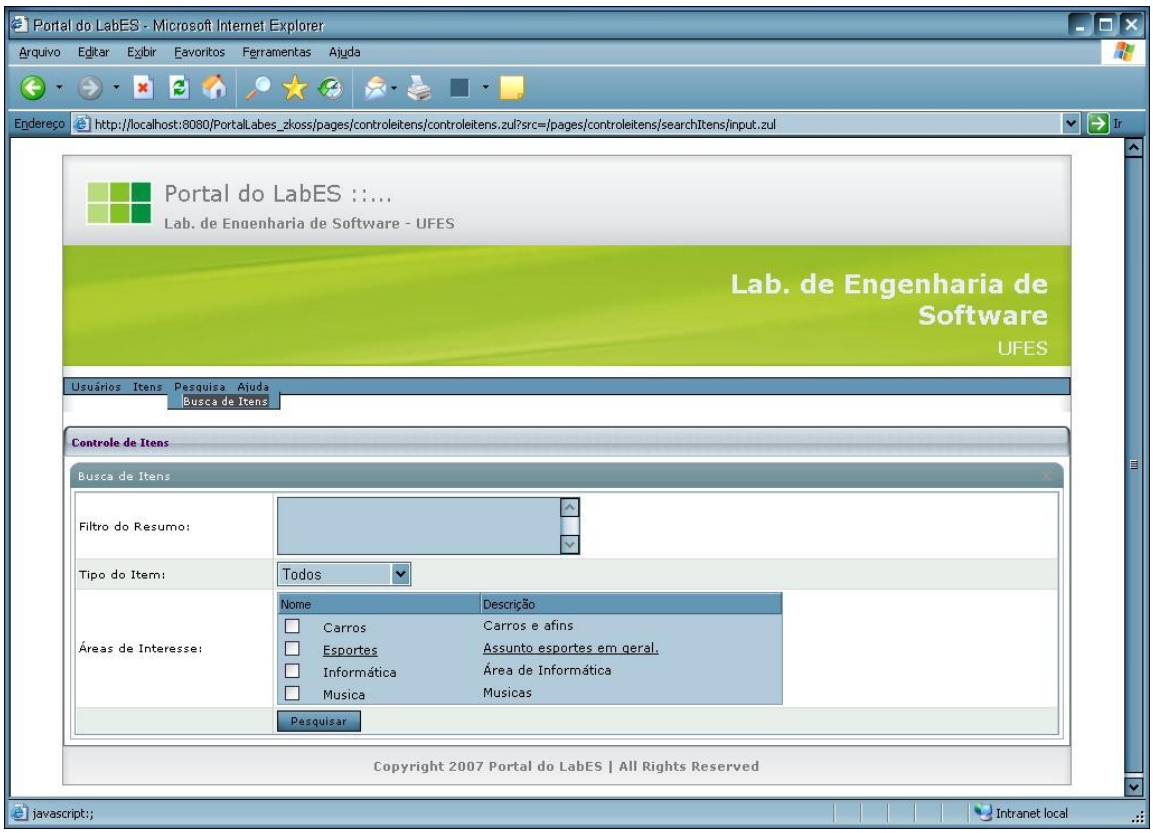

Figura 43 Tela para busca de itens do Portal do LabES.

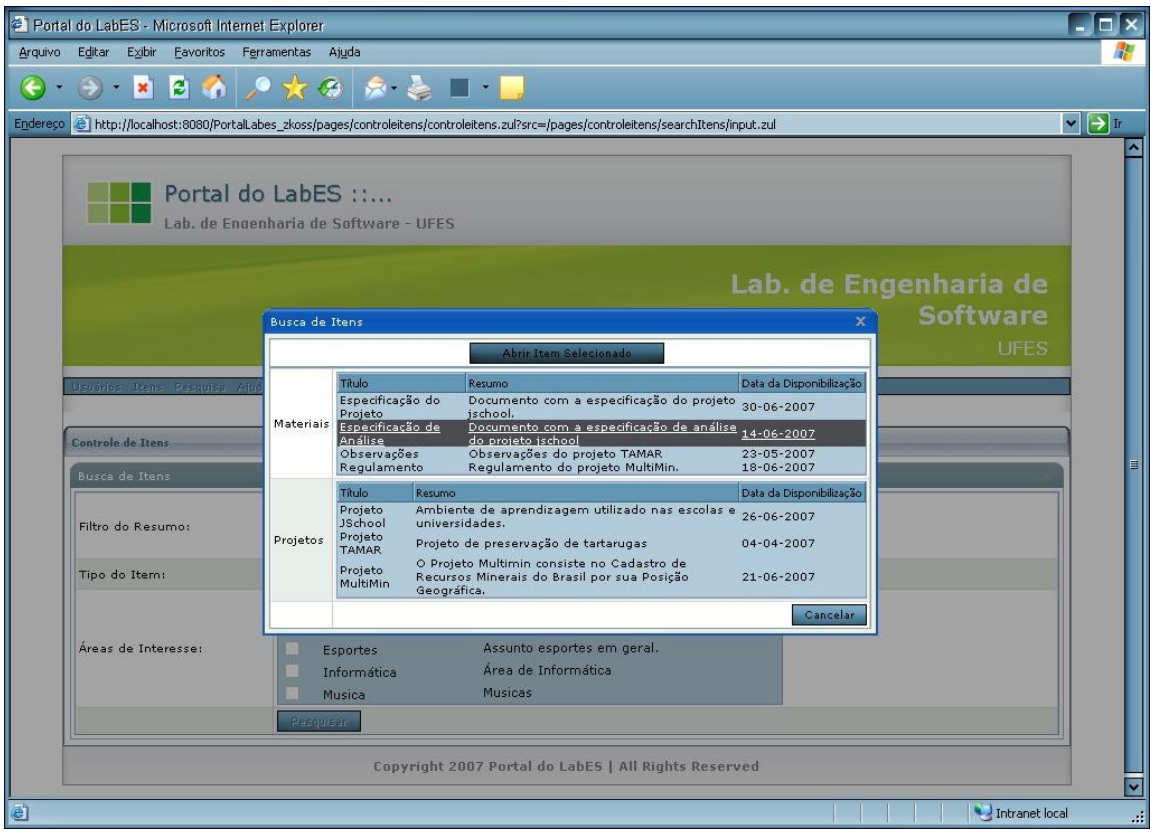

Figura 44 Tela de resultado da busca de itens do Portal do LabES.

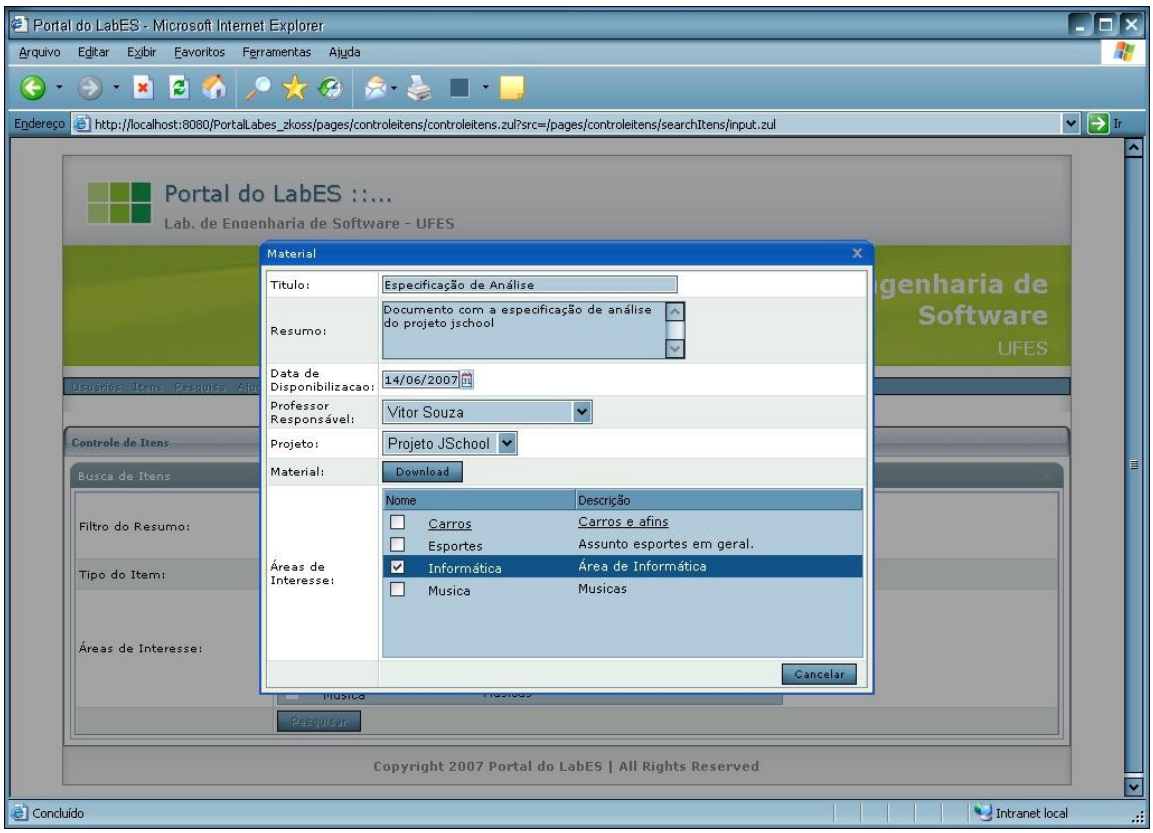

Figura 45 Tela de detalhes do Material do Portal do LabES.

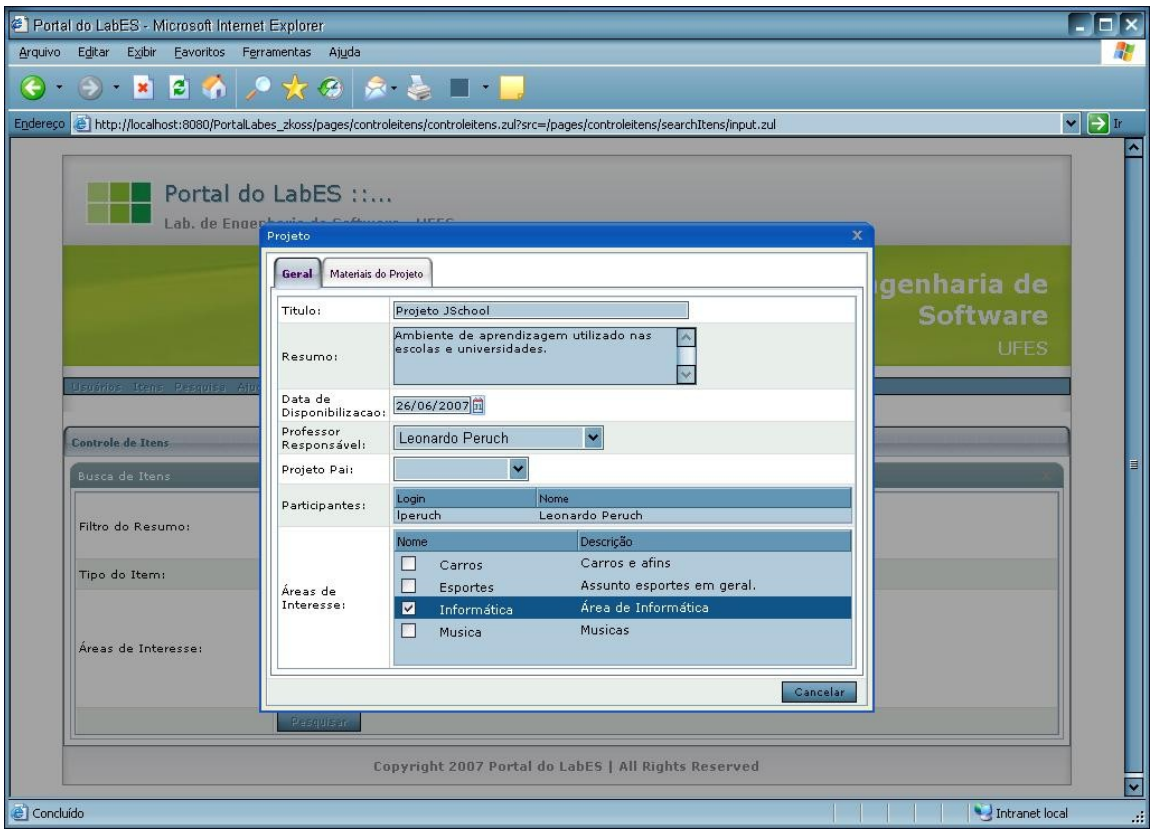

Figura 46 Tela de detalhes do Projeto do Portal do LabES.

Capítulo 7

Considerações Finais

Neste capítulo são apresentadas as conclusões a respeito do projeto desenvolvido ao longo deste trabalho (seção 7.1) e as perspectivas de trabalhos futuros (seção 7.2).

7.1. Conclusões

Destaca-se o surgimento de diversos *frameworks* que provêem uma sólida infra-estrutura para o desenvolvimento de Sistema de Informação *Web* (*Web-based Information Systems* - WISs). O uso desses *frameworks*, com o passar do tempo, tornou-se estado-da-prática e até mesmo influenciou a definição de padrões para desenvolvimento de aplicações distribuídas. Seu uso (ou reúso) auxilia a equipe de desenvolvimento a construir software mais rapidamente (vários componentes já estão prontos) e com maior qualidade (os *frameworks* já foram extensivamente testados por seus criadores) (SOUZA, 2007).

Contudo, a fim de aproveitar a consolidação de algumas categorias de *frameworks* e aumentar a produtividade no desenvolvimento de *WebApps* (abreviação de *Web Applications*), SOUZA, (2007) propõe o método FrameWeb (*Framework-based Design Method for Web Engineering*). O método assume que determinados tipos de *frameworks* serão utilizados durante a construção da aplicação, define uma arquitetura básica para o WIS e propõe modelos de projeto que se aproximam da implementação do sistema usando esses *frameworks*.

Neste trabalho aplicou-se o processo proposto pelo método FrameWeb no desenvolvimento do Portal do LabES, utilizando *frameworks* nos quais o método inicialmente foi baseado (Struts[2], Spring *Framework*, Hibernate e Sitemesh). Continuou-se o estudo do método, desenvolvendo o sistema em outras duas versões que utilizam diferentes *frameworks Web* (Struts[1] e ZK).

Na segunda versão do sistema, foi substituído o *framework* Controlador Frontal

(seção 2.2.1) original, por outro de mesma categoria. Na terceira versão do sistema, foi substituído o mesmo *framework* por outro de categoria diferente, a categoria de *frameworks* RIA (*Rich Internet Application*, seção 2.2.2). As duas categorias de *frameworks* são aplicadas à mesma camada, a camada de apresentação.

A fim de estudar os *frameworks* a serem utilizados no desenvolvimento da primeira versão do Portal do LabES (versão com o *framework* Struts[2]), foi iniciado junto ao Grupo de Usuários de Java do Estado do Espírito Santo (ESJUG[12]) o desenvolvimento de um ambiente cooperativo de aprendizagem chamado JSchool[13]. Foram conduzidas reuniões de treinamento e o projeto encontra-se em fase de implementação.

Para estudos dos outros *frameworks Web* (Struts[1] e ZK) utilizados no desenvolvimento das outras duas versões do portal, realizou-se pesquisas, leitura da documentação dos *frameworks*, leitura de artigos, entre outras atividades a fim de ganhar habilidade na utilização dos mesmos.

A partir da implementação do projeto e da experiência adquirida, é possível realizar uma avaliação do método FrameWeb e das dificuldades encontradas na utilização do aparato tecnológico para desenvolvimento de aplicações *Web*. Os estudos, os questionamentos e a implementação deste trabalho conferiram ao autor um crescimento profissional, acadêmico e pessoal imensuráveis por consolidar conceitos aprendidos ao longo do curso de Ciência da Computação, bem como a absorção de assuntos relacionados à área de interesse (Engenharia *Web*).

Após essa experiência, sugerimos propostas de melhorias para o modelo de navegação, proposto pelo método, para que ele seja adaptável aos *frameworks Web*. Quando utilizamos os *frameworks Web* analisados, as propostas de melhorias para o método são necessárias. Quando utilizamos outros *frameworks Web* no desenvolvimento de *WebApps*, talvez será necessário uma nova análise do modelo de navegação a fim de verificar sua adaptabilidade ao novo *framework Web*.

O modelo de navegação é, desta forma, dependente do *framework Web* utilizado. Como existem dezenas de instâncias de *frameworks Web* para cada uma das categorias Controladores Frontais e RIA, então o método é, de fato, viável quando se utiliza os

[12] http://esjug.dev.java.net
[13] http://jschool.dev.java.net

frameworks Web analisados nesse trabalho.

Na tentativa de generalizar o método, tentamos criar elementos abstratos que possam ser mapeados para elementos concretos dos *frameworks Web* particulares. Como não conseguimos fazer uma proposta de investigação neste sentido, concluir-se que o método FrameWeb é dependente de instâncias de *frameworks Web* e não somente da categoria de *framework* Controlador Frontal.

7.2. Trabalhos Futuros

Como trabalhos futuros relacionados ao tema exposto, podem ser destacados:

- Incrementar o projeto de Portal do LabES adicionando todas funcionalidades necessárias pelo laboratório. Utilizar como base a monografia do (FIORI, 2005) porque nela há a descrição de todos os requisitos do Portal do LabES.
- Analisar o método FrameWeb com diferentes *frameworks* da camada de persistência, ou seja, alteração no *framework* de mapeamento objeto / relacional (seção 2.2.4) a fim de analisar o Modelo de Persistência.

REFERÊNCIAS BIBLIOGRÁFICAS

ALUR, D., Crupi, J., Malks, D.: *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall / Sun Microsystems Press, 2003.

BEZERRA, E.; Desenvolvimento Incremental e Iterativo, 2004; Artigo disponível em: http://www.mundooo.com.br/php/mooartigos.php?pa=showpage&pid=20. Data de acesso: 02/04/2007.

COAD, P.; *Object-Oriented Patterns*. Communications of the ACM. 1992.

CONALLEN, J.: *Building Web Applications with UML*. 2nd edn. AddisonWesley, 2002.

DESHPANDE, Y.; Murugesan, S.; Hansen, S.; Ginige, A.; *Web Engineering: A New Discipline for Development of Web-Based Systems*, 2001.

DUHL, J.; White Paper: Rich Internet Applications; Artigo disponível em http://www.adobe.com/platform/whitepapers/idc_impact_of_rias.pdf . Data de acesso: 20/08/2007.

FIORI, G. C.; Desenvolvimento de um Portal para o Laboratório de Engenharia de Software. Monografia (Ciência da Computação) – Departamento de Informática, Universidade Federal do Espírito Santo, Vitória, 2005.

FONS, J., Valderas, P., Ruiz, M., Rojas, G., Pastor, O: OOWS: *A Method to Develop Web Applications from WebOriented Conceptual Models*. 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI), Orlando, USA, 2003.

FOWLER, M.; *Inversion of Control Containers and the Dependency Injection pattern*. 2004; Artigo disponível em: http://martinfowler.com/articles/injection.html. Data de acesso: 01/05/2007.

GAMMA, E., Helm, R., Johnson, R., Vlissides, J.; *Design Patterns: Elements of Reusable Object-Oriented Software*. EUA. AddisonWesley, 1995.

GIBBIS, W.; *Software's chronic crisis*. Scientific American, September, 1994.

GUIZZARDI, G.; Uma abordagem metodológica de desenvolvimento para e com reuso baseada em Ontologias formais de Domínio. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, Vitória, 2000.

HANSEN, S., Deshpande, Y., Murugusan, S., Ginige A.; *Web Engineering: A New Discipline For Development of Web-based Systems*. First Workshop on Web Engineering in International Conference on Software Engineering, 1999.

HUSTED, T., Dumolin, C., Franciscus, G, Interfeldt, D.; Struts em ação. Rio de Janeiro. Editora Ciência Moderna Ltda, 2004.

JDBC; Documentação da tecnologia JDBC; Disponível em: http://java.sun.com/products/jdbc/index.jsp. Data de acesso: 23/04/2007.

JOHNSON, R.E.; Foote B.; *Designing Reusable Classes*. Journal of Object Oriented Programming JOOP, 1988.

JOHNSON, R.E.; Russo, V.; *Reusing Object-Oriented Designs*. Relatório Técnico da Universidade de Illinois, UIUCDCS 91-1696, 1991.

KING, G. & Bauer, C. (2005): *Hibernate in Action. Manning Publishing* Co., 2005.

KOCH, N., Baumeister, H., Hennicker, R., Mandel, L., "*Extending UML to Model Navigation and Presentation in Web Applications".* Modelling Web Applications in the UML Workshop, October 2000.

LOOSLEY, C.; *Rich Internet Applications*: *Design, Measurement, and Management Challenges*. 2006. Data de Acesso: 20/08/2007.
Disponível em http://www.keynote.com/docs/whitepapers/RichInternet_5.pdf.

MACORATTI, J. C.; Padrões de Projeto : O modelo MVC - *Model View Controller*, 2000; Disponível em: http://www.macoratti.net/vbn_mvc.htm. Data de acesso: 23/04/2007.

MALDONADO, José Carlos; Vaccare, R. T. B.; Germando, F. S. R.; Masiero, P.C.; Padrões e *Frameworks* de software, 2002. São Paulo: Universidade de São Paulo. Disponível em: http://www.icmc.sc.usp.br/~rtvb/apostila.pdf. Data de acesso: 23/04/2007.

OLSINA, L.; *Specifying Quality Characteristics and Attributes for WebSites*, 2001.

PANDOLFI, D. A. C.; Melotti, E.; Ciclo de Desenvolvimento de Projetos Web, 2006

PRESSMAN, R.S. *Software Engineering: A Practitioner's Approach*. 5th edition. New York, USA, McGraw Hill, 2001.

PRESSMAN, R.S. *Software Engineering: A Practitioner's Approach*. 6th edition. McGraw Hill, 2005.

SCHWABE, D., Rossi, G.: OOHDM, *An Object Oriented Approach to WebBased Application Design. Theory and Practice of Object Systems 4*. Wiley and Sons, 1998.

SITEMESH; Disponível em: http://www.opensymphony.com/sitemesh/. Data de acesso: 15/05/2007.

SMITH, C.; *ZK Rich Client Framework And Agile Development*. Disponível em: http://www.theserverside.com/tt/articles/article.tss?l=ZKandAgile. Data de acesso: 20/07/2007.

SOUZA, V. E. S. FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web. 2007. Dissertação (Mestrado em Informática), Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, Vitória, 2007.

SOUZA, V. E. S., Falbo, R. A.: FrameWeb*: A Framework-based Design Method for Web Engineering*. Euro American Conference on Telematics and Information Systems, EATIS, 2007. Faro, Portugal, 2007.

SOUZA, V.; Falbo, R.A. *An Agile Approach for Web Systems Engineering*, XI Simpósio Brasileiro de Sistemas Multimídia e Web(WebMedia), MG, Brasil, 2005.

SPIELMAN, S. The Struts Framework - Pratical Guid for Java Programmers. Morgan Kaufmann Publishers, 2003.

TALIGENT. *Leveraging object-oriented frameworks*. Taligent Inc. White paper. 1995.

WALLACE, D.; Ragget L.; Aufgang J.; *Extreme Programming for Web Projects*. Addison-Wesley, 2003.