UNIVERSIDADE ESTADUAL PAULISTA
JÚLIO DE MESQUITA FILHO
Campus de Presidente Prudente

Pedro Teixeira Fogaça da Silva

CEDEM - Centro de Documentação e Memória da UNESP

2011

Pedro Teixeira Fogaça da Silva

CEDEM - Centro de documentação e Memória da UNESP

Trabalho de conclusão de curso apresentada ao Departamento de Matemática, Estatística e Computação (DMEC), da Universidade Estadual Paulista "Júlio de Mesquita Filho", sob o título **CEDEM - Centro de Documentação e Memória da UNESP**.

Orientador: Prof. Raphael Garcia

Supervisor: Prof. Dr. Ronaldo Celso Messias Correa

Presidente Prudente

2011

TERMO DE APROVAÇÃO

PEDRO TEIXEIRA FOGAÇA DA SILVA

CEDEM - CENTRO DE DOCUMENTAÇÃO E MEMÓRIA DA UNESP

Relatório final do estágio apresentado por Pedro Teixeira Fogaça da Silva e aprovado em 19 de dezembro de 2011, em Presidente Prudente, estado de São Paulo, pela banca examinadora constituída dos seguintes membros:

Orientador: Prof. Raphael Garcia
Universidade Estadual Paulista - UNESP

Prof. Dr. Ronado Celso Messias Correa
Universidade Estadual Paulista - UNESP

Prof. Dr. Messias Meneguetti Junior
Universidade Estadual Paulista - UNESP

Presidente Prudente, 30 de janeiro de 2012

Agradecimentos

"A gratidão é um fruto de grande cultura; não se encontra entre gente vulgar". (Samuel Johnson)

É uma lista grande, mas que não pode ser evitada. Gostaria de agradecer primeiramente a Deus, pai de amor e de luz, pela caminhada ao meu lado desde meu nascimento até o dia que desencarnarei. Em segundo lugar aos dois pais que tive a sorte de ter: Meu pai biológico, que conseguiu prover para sua familia mesmo após a sua morte, e meu atual padrasto, por tolerar minhas imperfeições e se esforçar o máximo pela minha mãe. Não tenho palavras nem para começar a agradecer minha mãe, que nunca cobrou, ou forçou nada em minha vida, e sempre me deu a liberdade de tudo, acompanhada de valores para a distinção entre o certo e o errado... Espero que esteja aproveitando bem essa enorme oportunidade que me foi presenteada, e se me tornar metade do homem que meus dois pais são/foram, já considero a minha vida como bem vivida. Agradeço a minha vó pelo exemplo de pureza que tornaria a humanidade melhor caso fosse seguido pela maioria.

Agradeço os amigos que me acompanharam até a porta da universidade. Marcelo Beghelli Nogueira, por ser o irmão que não tive na infância. José Geraldo Alves, por ter sido um grande amigo. Hermano Lúcio Gomes de Assis Filho, Paulo Eduardo Lucisano Bin, João Franscisco Pedrazzi, Gustavo Garrido, Régis Felipe Cônsulo Belizário, Bruno da Costa Ancheschi, Wilian Carlos Siena, Marcelo Sembeneli, Luciano Bertipaglia Fiori, Mayara Mendes Gasparotto, Mônica Hayashida e Bruna Maria de Freitas Coelho pela companhia, memórias intermináveis, momentos alegres e a grande parte de vocês que permaneceu em mim durante a minha formação pessoal. Agradeço aos amigos que me acompanharam durante a faculdade, aos que sairam prematuramente: Fábio Castello Novo, Rafael Rodrigues Bertola, Bruno Cesar Ratti e André Luiz Bernardino Nucci. Agradeço também ao Fernando Genghini, pelos grandes favores e com-

panhia na kitnet/cortiço onde moramos. Meus agradecimentos também a formação clássica da república butantan (Diogo, Bolinha, Birigui, Everton, Allan, Moller e Ademar) que sempre foi um lugar onde me senti em casa e sempre me proporcionou bons momentos. Sentirei falta também do Paulão da vira-latas. Boa sorte também para Layla Rodrigues Ferreira, permanecendo em Prudente ou transferindo. Fernanda Assen, muito obrigado pela paciência comigo, principalmente no fim de 2010, e muita sorte na tua trajetória. República Miojo, Carolina e Mariana, sentirei saudade de vocês duas também.

Agradeço a república Hangar, o ultimo lar que tive durante a vida de faculdade. Agradeço ao Tiozinho por nunca deixar a internet me distrair, usando a banda toda dela para ele. A vivência que tive nessa casa vai ficar em mim até o dia de minha morte...As risadas, os almoços, os seriados, filmes, as vezes que recebemos amigos, as festanças de arromba, as noites lá fora no quintal por causa do calor...Nunca fui bom em demonstrar gratidão, ou demonstrar importância de amigos em minha vida, e é com uma grande tristeza que vejo que estamos nos ultimos dias de nossa convivência. Dizem sempre que a melhor época da faculdade é começo, pois não há tamanha responsabilidade... Agradeço a vocês, Felca, Dida e Caio, por não deixarem isso ser verdade, já que os melhores anos da minha faculdade foram os dois ultimos, e quero tudo de melhor para a vida de vocês.

Natalina, vai dormir e obrigado por tudo também... Raphael Garcia, pela orientação, pela ajuda que tive, pela tolerância e compreensão, e pelo exemplo de profissional ético e correto a ser seguido. Bom e velho Chô, Fabrício, e você também, Max, pelo ambiente de trabalho descontraído, informal, e muito mas muito agradável. Saibam que levarei as amizades de vocês por onde estiver e sou muito grato por elas.

E, por fim, a Mayara Garcia Trevisani... Pela paciência, força, carinho e compreensão nessa tão dificil etapa de minha vida. Você com certeza tornou isso tudo mais fácil de suportar estando ao meu lado.

Não Agradeço a minha irmã (hehe) nem ao Aurélio, nem ao apelido de lavador de carros do Klaus e nem aos votos que tive para ser juramentista da colação de grau. Golpe Baixo!

Sumário

Lista de Figuras

Capítulo 1

INTRODUÇÃO

O Centro de Documentação e Memória da UNESP - CEDEM é um órgão pertencente a Universidade Estadual Paulista “Júlio de Mesquita Filho” e o responsável pela guarda de acervos documentais conectados à história política e aos movimentos sociais contemporâneos brasileiros além de ser especializado em fornecer apoio informativo à pesquisa de caráter social.

O CEDEM iniciou suas operações com uma linha de pesquisa baseada no projeto Memória da Universidade, de onde proveio o núcleo inicial de documentos que constituem a linha de acervo da história do ensino superior do Estado de São Paulo. Recebendo importantes acervos sobre a história contemporânea do Brasil, o CEDEM assume a condição de centro aglutinador de acervos documentais, focado em movimentos sociais. (REITORIA. . . , 2011)

Figura 1: Centro de Documentação e Memória da UNESP.

O CEDEM realiza diversos eventos com o intuito de que o acesso social de suas informações seja estendido, sendo o objetivo principal de suas atividades a contribuição para a transformação de conhecimento científico em conhecimento público. Assim, espera-se que essa transferência alimente a produção de obras de referência, temáticas, e que ofereça fundamentos para a criação e melhoria de políticas públicas. O CEDEM conta com instalações contidas em três andares do prédio em que se situa, possuindo salas de acervo, de tratamento técnico, de pesquisa, auditório para eventos, dentre outros.

Os itens de acervos são categorizados de acordo com um rigoroso critério chamado plano de classificação. O plano de classificação de um item indica se ele pertence a alguma instituição

acumuladora, a algum fundo ou coleção, a algum subfundo, a um grupo, um subgrupo, uma série e uma sub-série de itens. Os itens podem ser recuperados em uma consulta pelos dados de suas áreas temáticas, chamados descritores, que indicam como o termo fornecido deve ser pesquisado. Por exemplo, pesquisar pelo termo “diretas” em títulos de itens.

Os principais acervos sob a custódia do CEDEM são:

- ASMOB - Archivio Storico del Movimento Operaio Brasiliano
- CEDESP - Centro de Documentação e Estudos da Cidade de São Paulo
- CEMAP - Centro de Documentação do Movimento Operário Mário Pedrosa
- PCB - Fundado em março de 1922 como Partido Comunista do Brasil
- IRM - Instituto Cultural Roberto Morena
- MST - Movimento dos Trabalhadores Rurais Sem Terra
- Editora Oboré
- Santo Dias
- Clube de Mães da Zona Sul
- Clóvis Moura
- Davino Francisco dos Santos
- POLOP - Organização Revolucionária Marxista Política Operária
- UPA - University Publications of America

Figura 2: Edifício que abriga o CEDEM.

O novo sistema em desenvolvimento será muito vantajoso para o funcionamento do CEDEM, uma vez que estará a par com as tecnologias atuais, e poderá ser integrado com outros sistemas da instituição. Com isso um grande volume de informações poderá ser reaproveitado, citando como exemplo as informações de usuário, que poderão ser cadastradas uma única vez e então disponibilizadas para todos os sistemas institucionais aos quais o usuário tem permissão de acesso.

1.1 Objetivos do Trabalho

O objetivo deste trabalho é descrever as etapas do processo de desenvolvimento de um novo sistema para o gerenciamentos dos acervos do CEDEM, discorrer sobre a tecnologia utilizada, a sequência na implementação do sistema e eventos ocorridos.

1.2 Organização do Trabalho

No capítulo dois, será apresentada uma rápida visão da situação atual, em termos de tecnologia da informação, do CEDEM e da UNESP como um todo. Também serão mencionados benefícios que serão obtidos com a atualização do sistema do CEDEM.

No capítulo três será feita uma análise do *framework* desenvolvido pela UNESP, o Core, adotado como padrão das novas aplicações institucionais. Serão analisadas as tecnologias e a arquitetura do sistema, e os principais *frameworks* integrados em sua composição.

O capítulo quatro consiste em descrever a fase de implementação do novo sistema, das reuniões com representantes do CEDEM, das mudanças e correções sofridas durante o curso da implementação.

No capítulo cinco, será feita uma análise sobre o processo de migração de dados do sistema em uso para o novo sistema do CEDEM.

No capítulo seis, a situação atual do sistema será esclarecida.

Por fim, no capítulo sete, será sintetizada uma breve conclusão da experiência proporcionada pelo estágio supervisionado.

Capítulo 2

SITUAÇÃO ATUAL

2.1 O sistema de gerenciamento atual do CEDEM

Com a finalidade de manter as informações e os itens de seus acervos acessíveis ao público, o CEDEM precisa de um sistema que proporcione um modo ágil de localizar seus materiais. Atualmente, o CEDEM conta com um sistema baseado no sistema operacional *Microsoft Windows*. Porém, o sistema utiliza tecnologias defasadas que são incompatíveis com sistemas operacionais modernos, cujo Sistema Gerenciador de Banco de Dados (SGBD) é proprietário. Além disso, o SGBD é de baixa portabilidade, ou seja, não funciona em todos sistemas operacionais.

O sistema atual do CEDEM tornou-se obsoleto, e não se encontra em conformidade com as necessidades atuais do orgão. Com isso em vista, foi feita a requisição ao Comitê Superior de Tecnologia da Informação (CSTI) da UNESP de que um novo sistema seja criado para suprir a demanda do CEDEM. O CSTI então encaminhou a requisição para o Serviço Técnico de Informática (STI) da Faculdade de Ciência e Tecnologia (FCT) , em Presidente Prudente.

2.2 A tecnologia da informação na UNESP

A UNESP possui mais de 35 unidades espalhadas em 23 cidades no estado. Em vista da magnitude da instituição, é natural que haja a necessidade em adotar um padrão de desenvolvimento, uma vez que sistemas desenvolvidos por uma mesma instituição não só podem como devem ser capazes de realizar a comunicação entre si. A padronização possibilita um amplo desenvolvimento das aplicações institucionais, já que aproveita recursos implementados de outras unidades. Um exemplo claro é o cadastro de cidades. Não há a necessidade de cadastrar todas

as cidades em uma aplicação, se uma outra já possuir um registro de cidades cadastradas, economizando tempo e recursos. Com a padronização como meta,o STI da Faculdade de Ciências (FC), situada em Bauru, mplementou um framework já existente e o utilizou como base para as aplicações institucionais, iniciando o seu desenvolvimento. O Core já foi adotado em diversos projetos. Só no STI da FCT, o core está sendo utilizado em cinco projetos, sendo um deles o CEDEM, abordado neste documento.

Os outros projetos que utilizam o Core em Presidente Prudente são:

- PROEX, da Pró-Reitoria de Extensão, cuja finalidade é gerenciar bolsas, projetos e outros recursos.
- CEAFIR, para a clínica de fisioterapia da FCT, para gerenciar suas funções e recursos.
- FCT/STA, para gerência de colegiados e portarias da FCT.
- Descentralização, para aumentar a transparência nos gastos de funcionários da FCT.

2.3 Vantagens do novo sistema

A reformulação do sistema gerenciador do CEDEM em uma plataforma atual servirá para que auxiliar as atividades do órgão da UNESP. Como mencionado anteriormente, o sistema atual possui instâncias instaladas nos computadores dos usuários, ao contrário de sistemas web, que dependem somente de um web browser, implementado na plataforma SQLServer da Microsoft em uma versão também obsoleta.Assim, apenas um computador pode acessar o sistema, e caso ele passe por manutenção ou dificuldades técnicas, o sistema deve ser instalado em um segundo computador e o banco de dados replicado para essa máquina. Somente após esse procedimento o sistema voltaria a ser usado normalmente.

No novo padrão, o sistema do CEDEM será um sistema na plataforma web. Estando hospedado em um servidor juntamente com sua base de dados, a forma de acesso será por navegadores de internet, podendo ser acessado de qualquer computador conectado à internet eliminando a necessidade de instalação. Basta ao usuário acessar o servidor pelo navegador e se autenticar, não importando qual computador ele estará usando.

Visualmente, as aplicações são mais intuitivas para o usuário, uma vez que será compatível com maiores resoluções. Apresentará o padrão visual da UNESP, já visto nos websites da

universidade e de suas unidades.E o sistema atuará como um módulo do Core, já integrado com outros sistemas institucionais. Outra vantagem importante é que a aplicação se baseia no padrão *Model View Controller* (MVC), facilitando manutenção e organização entre equipes de desenvolvimento.

Capítulo 3

FRAMEWORK CORE

Este capítulo tratará do *framework* Core, desenvolvido como uma resposta a necessidade de padronização na UNESP. As principais mentes envolvidas na criação do Core são Alessandro Morais e André Penteado, da Faculdade de Ciências, um dos Campi da UNESP em Bauru. Dadas as proporções da UNESP, a demanda por sistemas informatizados naturalmente deve ser grande. Ainda mais no contexto de crescente modernização nos dias atuais.O Core então, além de um padrão comun de desenvolvimento, fornece ferramentas e recursos prontos aos desenvolvedores, a fim de aumentar a velocidade da implementação. Baseado na plataforma Java, utilizando o Sistema Gerenciador de Banco de Dados (SGBD) PostgreSQL versão 8.4 e o servidor *Web* Apache Tomcat versão 6.0, o Core consiste em um amálgama de *frameworks* e utilitários já configurados e funcionando juntos. A configuração de um ambiente de desenvolvimento é trabalhosa e exige muito tempo, e a integração entre *frameworks* apresenta uma dificuldade ainda maior(por exemplo, integrar Hibernate com Mentawai). Uma grande parte desse processo é eliminada com o uso do Core. O Core agrega inúmeros *frameworks*, sendo eles:

Tabela 1: *Frameworks* integrados e suas funções.

Framework	**Função**
JPA	Acesso ao banco de dados
Hibernate	Acesso ao banco de dados
Mentawai	Ações, autenticação e autorização
Log4J	Auditoria
Apache commons-configurator	Gerência de configurações
Sitemesh	Layout
JasperReports	Relatórios
iReport	Relatórios
DisplayTable	Paginação web

3.1 O padrão MVC

O Core é baseado na arquitetura *Model-View-Controller* - MVC, que divide a aplicação em três camadas. A primeira camada é a camada de modelo, onde são modeladas as tabelas do banco de dados para que sejam representadas apropriadamente, é uma ponte entre a base de dados e a lógica de negócio do programa. Ela é geralmente constituída de classes de persistência e classes que fazem requisições diretas ao banco de dados. A segunda camada é a camada controladora, a camada portadora das regras de negócio da aplicação. A camada controladora é responsável por requisitar dados da camada de modelo, efetuar os processamentos necessários e enviar a resposta a camada de exibição. Ou seja, é a camada em converter os dados de entrada em uma saída adequada, "solucionar o problema". A terceira camada é a camada de exibição, cujo propósito é gerar a saída para o usuário, de maneira compreensiva. A Figura 3 é um diagrama que ilustra a organização das camadas da arquitetura MVC.

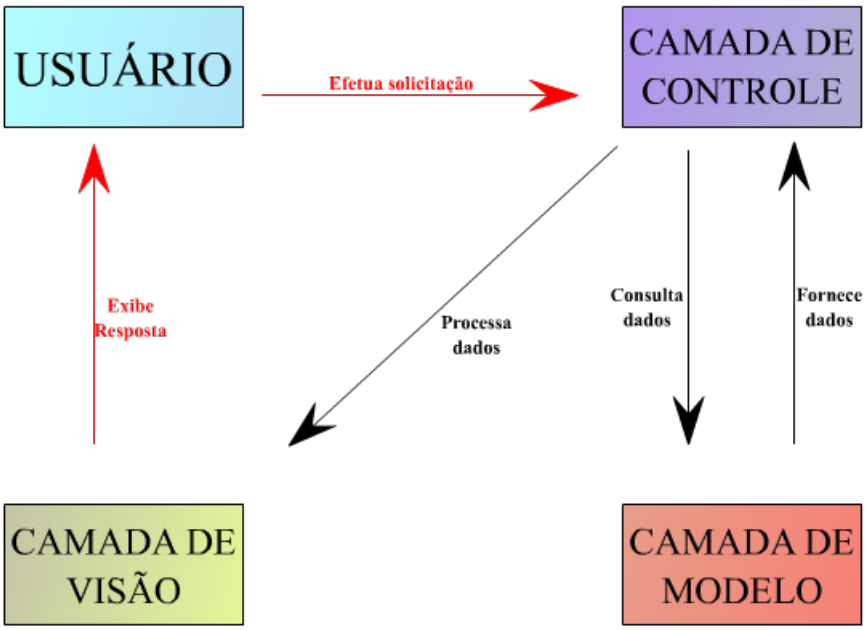

Figura 3: O padrão MVC.

Com essa divisão em camadas, a manutenção e a alteração em um sistema que use o padrão MVC é favorecida, uma vez que as camadas trocam mensagens entre si e não são dependentes uma da outra. O código-fonte é mais legível já que não há mistura entre elementos de visualização, de processamento e de persistência. O trabalho em equipe também é favorecido, pois as três camadas podem ser desenvolvidas simultaneamente.

O Core representa a camada de modelo com classes do tipo bean, que são classes de persistência sem métodos e classes do tipo service, que são classes com métodos de acesso ao banco de dados. A camada controladora é representada por classes do tipo action, que normalmente contém muitos métodos e classes do tipo service, instanciadas para recuperar os dados a serem processados da camada de modelo. E por fim, a camada de exibição são representadas por páginas de internet *Java Server Pages* - JSP, por onde os usuários interagem com o sistema.De fato, é uma forma organizada de se trabalhar, pois uma página JSP contém apenas elementos voltados a exibição de conteúdo, enquanto que classes action contém apenas conteúdo voltado ao processamento e classes service conteúdo voltado a interface com o SGBD. Assim, o paralelismo no desenvolvimento é possível e encorajado, e os sistemas que usam esse padrão podem ser implementados com maior agilidade. A Figura 4 demonstra a distribuição dos elementos constituintes do Core nas camadas da arquitetura MVC.

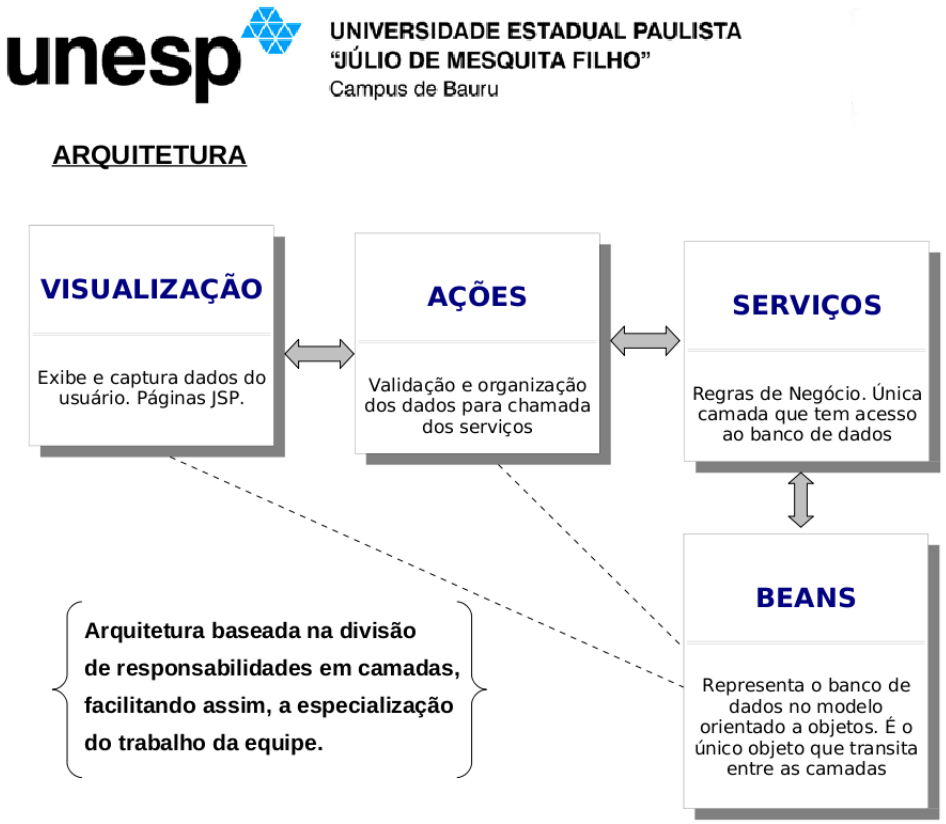

Figura 4: O padrão MVC representado no Core.

Uma aplicação MVC pode ser vista como uma coleção de componentes de dados, de lógica e de exibição, sendo que cada trinca é responsável pela gestão de um elemento da interface com o usuário e pode ser vista como um micro-sistema MVC embutido. A interface gráfica *Swing* apresenta este comportamento, onde a maioria de seus componentes de interface são sistemas

MVC completos. Apesar de ser frequentemente associado com *frameworks*, o modelo MVC é uma arquitetura, podendo ser implementado inclusive em linguagens não orientadas a objeto. (MVC..., 2011)

3.2 Hibernate

Hibernate é um *framework* open-source de Java desenvolvido pela empresa JBOSS. Sua função é permitir que bases de dados relacionais sejam manipuladas com a abordagem orientada a objetos. Isso é possível através do mapeamento das tabelas de um banco de dados em classes especiais denominadas *beans*, e usam comandos especiais chamados *Annotations* para sincronizar cada campo da tabela com seu respectivo atributo representante na classe. Sendo assim, a programação com o Hibernate é mais natural ao paradigma de orientação a objetos, uma vez que permite que os dados na linguagem Java, como herança, coleções, polimorfismo, entre outros, sejam gravados no banco de dados exatamente como estão em memória.

O Hibernate possui suporte a *linguagem Structured Query Language* - SQL, mas a linguagem mais usada nele é o *Hibernate Query Language* - HQL. O HQL, diferentemente do SQL, pode pesquisar por diretamente por objetos ao banco, uma vez que faz a conversão entre beans e tabelas. Tomando como exemplo uma tabela fictícia carro que contém uma chave estrangeira de uma outra tabela garagem, após a conversão, os dados da tabela carro seriam manipulados no ambiente do Hibernate como uma classe carro, cujo atributo garagem contem os dados da tabela garagem a qual a chave estrageira relacionou a tabela carro.

Outras vantagens do Hibernate são sua estabilidade e a sua persistência transparente, pois como as beans são as classes que são armazenadas no banco de dados, não há necessidade de geração de bytecode, aumentando a performance de execução.A ação *dirty checking* evita que o banco de dados sofra operações de escrita desnecessárias, permitindo a mudança somente em tabelas cujas classes de persistência representantes sofram alteração. No Core, o Hibernate é usado para o acesso ao banco de dados. (HIBERNATE..., 2011)

3.3 Mentawai

Mentawai é um *framework* baseado no padrão MVC com o intuito de ser simples, evitando que o desenvolvedor precise configurar arquivos do tipo *Extensible Markup Language* - XML, e utilizando código Java puro. Com a crescente complexidade das aplicações Java, muito tempo é

perdido com a configuração de XML, em detrimento do desenvolvimento de código Java em si. O excesso de XML aumenta a curva de aprendizado, uma vez que cada *framework* possui uma codificação XML própria, que o programador deve aprender. No Core, a configuração XML é feita apenas em arquivos do servidor, e não em arquivos de projeto.

A simplicidade do Mentawai propicia a resolução dos problemas mais comuns usando métodos mais fáceis, e por isso é um *framework* bem aceito entre os desenvolvedores Java. Com pouco tempo de prática, os programadores já conseguem desenvolver aplicações complexas, aumentando a produtividade. e para iniciar o uso, basta que o arquivo Jar do Mentawai seja referenciado no projeto que o utiliza.

O Mentawai adota o paradigma de *Actions*. As *Actions* possuem uma entrada denominada de *input*, de onde dados de uma requisição web são adquiridos, e uma saída nomeada *output*, por onde os resultados das ações efetuadas na *Action* podem ser recuperados. O resultado é gerado após a execução, sendo geralmente sucesso (*Success*) ou erro (*Error*) e novos resultados podem ser criados pelo desenvolvedor. Cada resultado contém uma conseqüência, cuja invocação depende do resultado obtido. As *Actions* também possuem contextos, que usualmente são contexto de sessões ou de aplicações, mas novamente o usuário tem a liberdade de incluir seu próprio contexto.

Toda a configuração do Mentawai, apesar de ser em Java, não se mistura a lógica da aplicação. A configuração é acumulada em uma classe chamada *ApplicationManager*, que contém o mapeamento das ações presentes no sistema e suas respectivas conseqüências. As funcionalidades encontradas com maior frequência nas aplicações Java, como filtros, auntenticação, validação, entre outras, já estão implementadas no Mentawai, não sendo necessário reescrever seus respectivos códigos e adapta-los ao projeto. No Core, o Hibernate é usado principalmente para o controle de ações, autenticação e autorização. (MENTAWAI, 2011)

3.4 JPA

Java Persistence API - JPA é um *framework* para Java cuja designação é gerenciar dados relacionais na linguagem Java. A principal funcionalidade são as entidades de persistência, que são classes em que seus estados são persistidos para uma base de dados. Entidades relacionam-se umas com as outras, representadas no *framework* por metadados entre objetos e dados relacionais. JPA usa uma linguagem inspirada em SQL chama *Java Persistence Query Language*

- JPQL, que opera em entidades de persistência ao contrário de operar diretamente com dados das tabelas do banco de dados. O Hibernate é uma implementação do JPA. No Core, o JPA é usado para acessar o banco de dados. (JPA, 2011)

3.5 Log4J

Log4J é um *framework* para Java com a finalidade de registrar dados decorrentes da execução de um sistema. esse registro é denominado *logging*. A maioria dos Ambientes Integrados de Desenvolvimento (IDEs) possuem depurações eficientes. *Logging* é uma forma de depuração, exibindo mensagens de status e de erro na tela, ou então armazenando-as em um arquivo.

Existem neste *framework* níveis diferentes de logging, com diferentes graus de importância. As mensagens de nível *OFF* tem o maior grau de importância, e desabilitam o *logging*. Mensagens de nível *FATAL* são erros graves, que causam encerramento prematuro da execução, e são visíveis no console de status. as mensagens de nível *ERROR* são para erros de execução ou resultados inesperados, sendo imediatamente visíveis no console. Mensagens de nível *WARN* são para uso indevido de APIs, uso de APIs obsoletas, e situações onde o resultado é inesperado, mas não constitui um erro de execução, e são visíveis de imediato no console. Mensagens de nível *INFO* registram pontos de interesse do desenvolvedor na execução, e por serem imediatamente visíveis no console, devem ser usadas o mínimo possível. Mensagens do tipo *DEBUG* relatam o fluxo de execução do sistema, e costumam ser escritas em um arquivo de *logging* apenas.Por fim, mensagens do tipo *TRACE* oferecem informações detalhadas sobre algum ponto de interesse na execução e também costumam ser escritas apenas em arquivos de *logging*. É usado para *logging* e depuração no Core. (GUJ. . . , 2011)

3.6 Apache commons-configurator

Apache commons-configurator é uma biblioteca que providencia uma interface genérica de configuração, permitindo a uma aplicação Java a funcionalidade de recuperar configurações de uma variedade de fontes, como arquivos de propriedades, arquivos XML, dentre outros. No Core, o Apache *commons-configurator* tem como função gerenciar as configurações internas de seus *frameworks* integrantes. (APACHE. . . , 2011)

3.7 Sitemesh

Sitemesh é um *framework* com a finalidade de decorar e enfeitar aplicações *Web*, sem interferir no desenvolvimento da lógica, para que estas aplicações apresentem aparência, organização e navegação consistentes e amigáveis. Páginas *Hyper Text Markup Language* - HTML enviadas como Resposta a requisições feitas ao servidor são interceptadas pelo Sitemesh, que extrai e processa o conteúdo relevante e o mescla com um ou mais modelos conhecidos individualmente por *decorator*, para construir o resultado a ser enviado para o cliente. Estruturas de organização maiores e mais complexas podem ser obtidas a partir de estruturas menores. É extensível, apresenta alto desempenho e é compatível com aplicações *Web* baseadas na linguagem Java e aplicações de conteúdo *offline*. Sitemesh tem o papel de gerenciar a aparência nas aplicações usuárias do Core. (SITEMESH3. . . , 2011)

3.8 JasperReports e iReports

JasperReports e iReports constituem um ambiente para a produção e desenvolvimento de relatórios em aplicações voltadas a internet e aplicações voltadas a desktop. Este ambiente é designado para formatar documentos gerados, possibilitando um maior controle sobre o que será impresso. Páginas *Web*, por exemplo, não permitem o controle do que será impresso no rodapé de cada página.

JasperReports é um *framework open-source* escrito em Java para gerar relatórios dinâmicamente em inúmeros formatos, sendo os principais formatos usados o formato PDF, o formato HTML, o formato XML, dentre outros. O design do relatório é criado em um arquivo XML, onde são definidos os campos para que possam ser mapeados em uma próxima etapa. Utilizando o XML é possível definir elementos estáticos, como textos, imagens e formas geométricas, e elementos dinâmicos, como campos a serem preenchindos com dados provenientes de um banco de dados. O framework pode ser alimentado com informações de dois modos: Com consultas SQL dentro de seus arquivos XML ou através do resultado de uma consulta realizada por um outro objeto, sendo repassado ao JasperReports com a forma de um objeto *ResultSet*. É comum se referir ao relatório obtido como resultado da alimentação de um arquivo Jasper, utilizado pelo JasperReports, com uma fonte de dados por *print* que então pode ser convertido para formatos mais tradicionais e difundidos.

O iReport é uma ferramenta que habilita ao usuário criar o design do relatório, utilizando um ambiente gráfico, com as características de um relatório gerado pelo JasperReports. Porém,

ao contrário do processo de criação de um relatório através de arquivos XML do JasperReports, o arquivo XML é gerado de maneira automática pelo iReport, aumentando a produtividade da equipe de desenvolvimento. No Core, os relatórios são gerados via JasperReports e organizados usando iReport. (RELATORIOS. . . , 2011)

3.9 DisplayTable

DisplayTable é uma forma de usar a *Display Tag Library*, que por sua vez é um conjunto *open-source* de *Tags* que proporcionam padrões de apresentação *Web* de alto nível, integraveis em aplicações que utilizam a arquitetura MVC. Contém um alto número de funcionalidades, apesar de manter a simplicidade de uso. A *Tag* de *Display* recebe uma lista de objetos como parâmetro de entrada e realiza operações de ordenação, posicionamento, decoração, agrupamento, definições de *link*. ajuste de tamanho e exportação por si própria em um estilo XHTML que pode ser customizado. A *Display Tag* é utilizada no Core para gerenciar tabelas e exibir de maneira rápida e eficaz resultados de pesquisas. (DISPLAY. . . , 2011)

3.10 Outras tecnologias

Outras duas tecnologias utilizadas em grande escala e mencionadas nos tópicos anteriores, mas não esclarecidas até então por não serem *frameworks*, são XML e AJAX.

XML (*Extensible Markup Language*) é atualmente um padrão para gerar linguagens de marcação de acordo com recomendação do W3C, que é um consórcio internacional que almeja o desenvolvimento de padrões para a concepção e entedimento de conteúdo *Web*. As principais metas de um arquivo XML são manter a simplicidade e a legibilidade tanto para pessoas quanto para *softwares*, foco na maneira como a informação é estruturada e não como é apresentada, e separação entre conteúdo e formatação. Nos projetos que utilizam o Core o uso de XML é ocultado pelo Mentawai, sendo apenas manuseado na configuração do servidor Apache Tomcat. (XML. . . , 2011)

Assynchronous Javascript And XML - AJAX não é um modelo novo de desenvolvimento,porém só veio se popularizar recentemente, e é um dos responsáveis pelo advento do paradigma da *Web* 2.0 vivido atualmente. O funcionamento do Ajax consiste em fazer com que elementos diferentes de uma página sejam atualizados sem sincronia entre sí, somente quando necessário. Assim uma página não precisa ser gerada por inteira novamente devido a um envio

de dados do servidor, adicionando um grau de interatividade similar ao de aplicações *desktop*. Gmail, Google Earth e Google Maps fazem uso de AJAX. No Core, códigos AJAX são criados para que as aplicações respondam requisições do servidor em tempo real sem que haja a recarga da página inteira. Com isso não se perde os dados já entrados em formulários de cadastro, a aplicação se torna mais simples e intuitiva ao usuário e são evitados estados com potencial para causar erros. (AJAX. . . , 2011)

3.11 O projeto Sentinela

O projeto Sentinela é um projeto que é executado junto com os demais projetos no servidor *Web*. A função do Sentinela é providenciar uma interface para o acesso de usuários que não usa do artifício de sessão dos navegadores *Web*, sendo assim mais seguro, e efetuar o *logging* das ações dos usuários nos projetos sob a "tutela" do Sentinela. Para que um sistema feito no Core seja acessado, é necessário efetuar a autenticação no sentinela. Qualquer página acessada sem autenticação é redirecionada para o *login* do Sentinela.

Capítulo 4

O PROCESSO DE DESENVOLVIMENTO

Neste capítulo será tratado o processo de desenvolvimento do novo sistema. Serão vistos em detalhes o desenrolar do desenvolvimento, em ordem cronológica dos acontecimentos, e sendo usado como "percurso" de desenvolvimento as modelagens do banco de dados. A rotina de estágio, e as experiências de trabalho serão discorridas, assim como os principais desafios e soluções encontradas com o embasamento em conceitos adquiridos nas disciplinas pertencentes ao curso de Bacharelado em Ciência da Computação.

4.1 Condições de Desenvolvimento

Para desenvolver o novo sistema do CEDEM, foram disponibilizados dois computadores *Itautec*, da UNESP de Presidente Prudente. Os computadores possuem processador AMD *athlon* x2, com memória RAM de 2 Gb. O sistema operacional usado é o Linux, distribuição Ubuntu, versão 10.10 - 64 bits. Para servidor local de testes, foi usado o servidor *Web* Apache Tomcat versão 6.0 (o servidor usado pelo Core) e o próprio *framework* Core. Como Sistema Gerenciador de Banco de Dados, foi usado o PostgreSQL versão 8.4. A codificação foi feita na IDE Eclipse Galileo for Java EE (Figura 5), gratuíta.

O controle de versão e o gerenciamento de versão são feitos pelo *Subversion*, também conhecido como SVN. O SVN foi desenvolvido para ser um substituto moderno para o modelo CVS de gerência de configuração. (SUBVERSION. . . , 2011)

A carga horária semanal é de doze horas, mas eram cumpridas quinze horas semanais, com essa margem de seguraça visando a ocorrência de imprevistos, ou a liberação de um turno de estágio para fins acadêmicos.

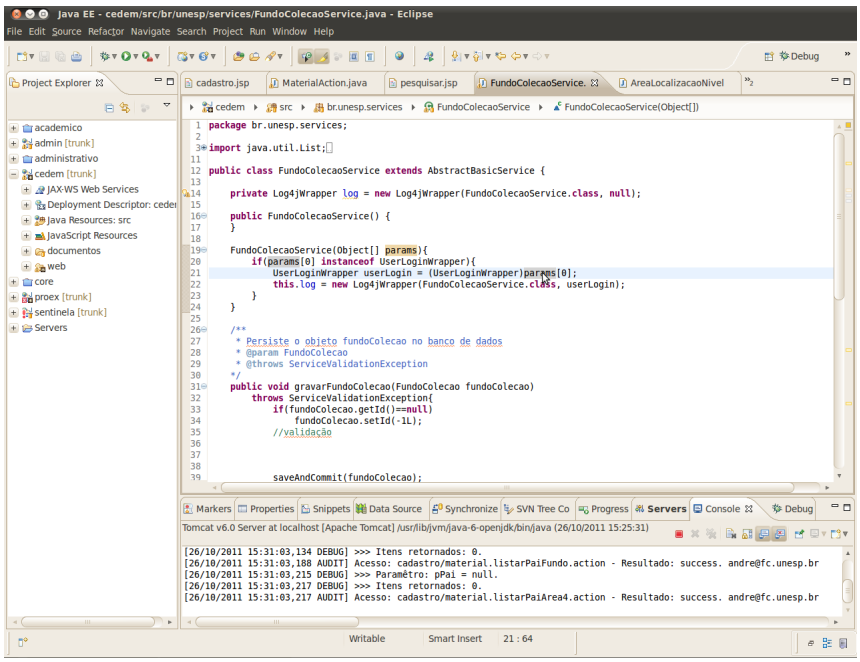

Figura 5: O ambiente de desenvolvimento *Eclipse Galileo for Java EE*.

4.2 O Servidor

Até o momento do início da implementação, era esperado que a implantação seria efetuada em um servidor *Dell Power Edge* (DETALHES. . . , 2011), com múltiplos processadores *Intel Xeon*. A replicação de dados usa a tecnologia Slony-l, que consiste em um sistema com um servidor mestre e seus escravos. Quando uma transação é confirmada, isto é, sofre um *commit*, não é garantido que a informação esteja disponível para os servidores escravos. Slony também é considerada uma tecnologia "em cascata", pois réplicas podem ser criadas e atualizadas através de servidores escravos, não necessitando de contato com o servidor mestre.(SLONY-L. . . , 2011)

4.3 Etapa Inicial

As atividades tiveram início em agosto de 2010. A equipe desenvolvedora, que era formada por um Analista e dois bolsistas, tinha acesso a alguns documentos que foram usados como base para a primeira modelagem do banco de dados e estudo dos requisítos. Os documentos eram um impresso que contia algumas imagens de telas do sistema atual e alguns de seus formulários, e uma tabela com a descrição do que era desejado para os campos dos formulários e a relação entre os tipos de itens contidos nos acervos sob a custódia do CEDEM e os campos dos formulários. Através de reuniões entre a equipe de desenvolvimento e o estudo do documentos foi feita uma primeira modelagem e uma primeira estimativa dos requisitos.

Como o novo sistema do CEDEM seria desenvolvido usando o Core, e o sistema gestor da Pró-Reitoria de Extensão (PROEX) também utilizaria o *framework* proprietário da UNESP, o STI de Presidente Prudente organizou um treinamento com duração de uma semana sobre Java, Java para *Web* e Core. O treinamento foi aplicado por Alessandro Morais, um dos desenvolvedores do Core. O treinamento foi prático e foi um exercício acompanhado montar o ambiente de programação e depois desenvolver um sistema de cadastro usando as principais ferramentas do Core e adequando a codificação na arquitetura MVC.

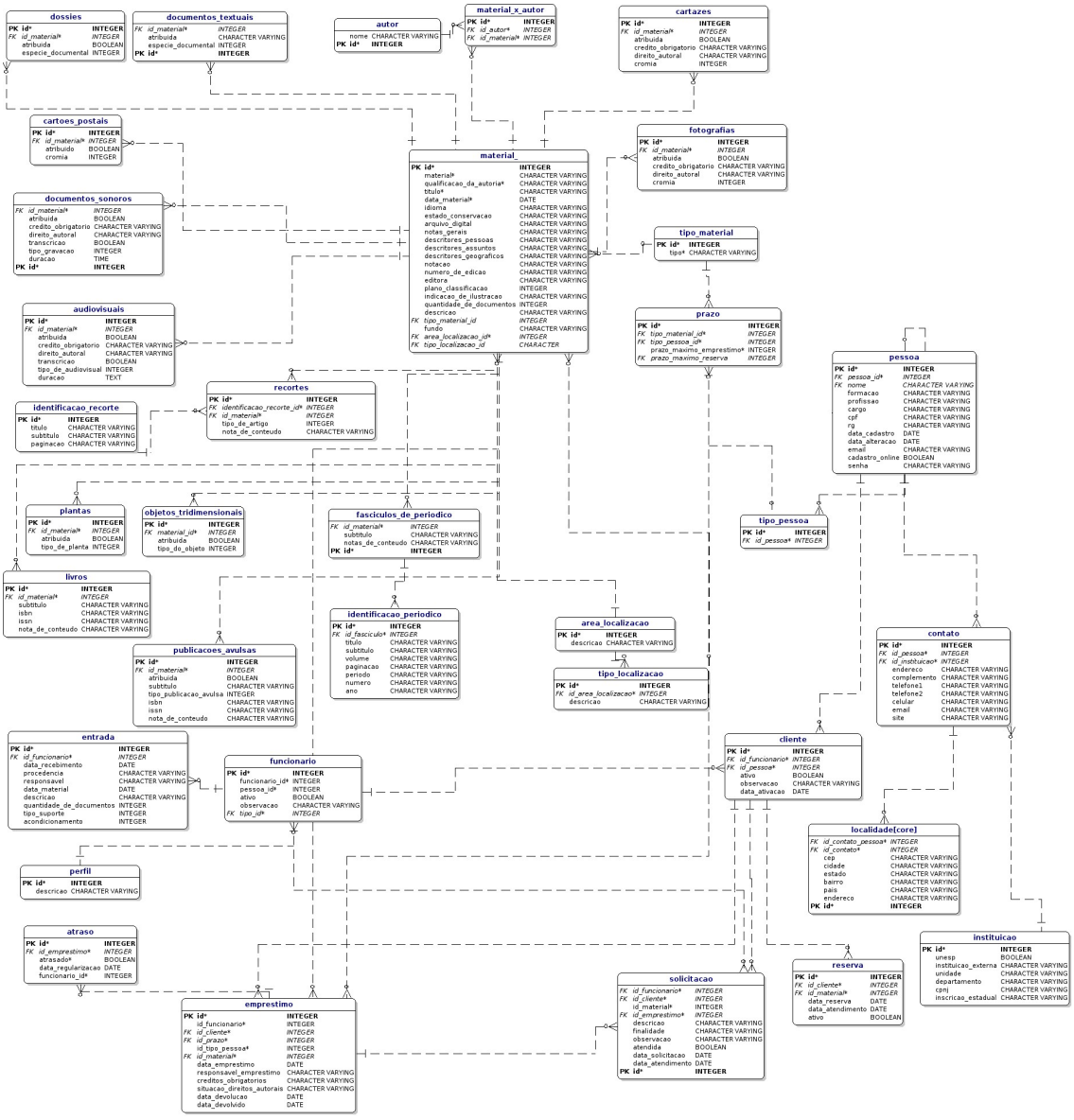

Figura 6: A primeira candidata à modelagem definitiva.

Paralelamente ao desenvolvimento, foi então disponibilizada uma sala e computadores para que a equipe do CEDEM, e posteriormente as outras equipes desenvolvedoras do STI de Presi-

dente Prudente, utilizasse como ambiente de trabalho. As equipes se alternavam no uso da sala devido ao seu limitado espaço.

A medida em que as reuniões entre a equipe desenvolvedora do CEDEM (referenciada a partir deste ponto como equipe do CEDEM apenas) iam acontecendo, havia um maior refinamento e uma maior congruência na modelagem do banco de dados. As modelagens eram verificadas e validadas via testes de mesa, que simulavam casos de uso para o sistema e o testavam contra o esquema atual apontando pontos positivos e negativos, e simulações de inserções nas tabelas, que poderiam identificar problemas com a modelagem em um estágio anterior ao estágio de implementação. Uma versão candidata a definitiva foi produzida, que tratava diversas situações, tais quais empréstimo de materiais juntamente com seus prazos e atrasos, e informações administrativas de usuários, como endereço, localidade e dados pessoais (Figura 6).

Uma reunião foi marcada com duas representantes do CEDEM, Solange de Souza e Sandra Moraes, para o dia 6 de dezembro de 2010. A reunião se extendeu por toda a tarde, e diversos pontos foram levantados. As representantes do CEDEM esclareceram alguns dos protocolos de funcionamento do orgão, o que gerou mudanças na estrutura das tabelas do banco de dados apresentada na reunião. Após o fim da tarde, foi decidido que a reunião precisaria continuar, e a reunião continuou no dia 7 de dezembro de 2010, na parte da manhã. O objetivo da reunião era efetuar testes na modelagem, considerando as correções sofridas, para que as representantes do CEDEM e a equipe de desenvolvimento do sistema do CEDEM atingissem um consenso (EQUIPE..., 2010). O consenso foi de fato alcançado mas o desenvolvimento não se iniciou imediatamente em seguida. A modelagem sofreu então otimizações, para que o desenvolvimento não fosse afetado por correções tardias e que por consequência são as que possuem o curso mais alto para serem efetuadas (Figura 7). Assim se deu o encerramento da etapa inicial de desenvolvimento, no dia 17 de dezembro de 2010.

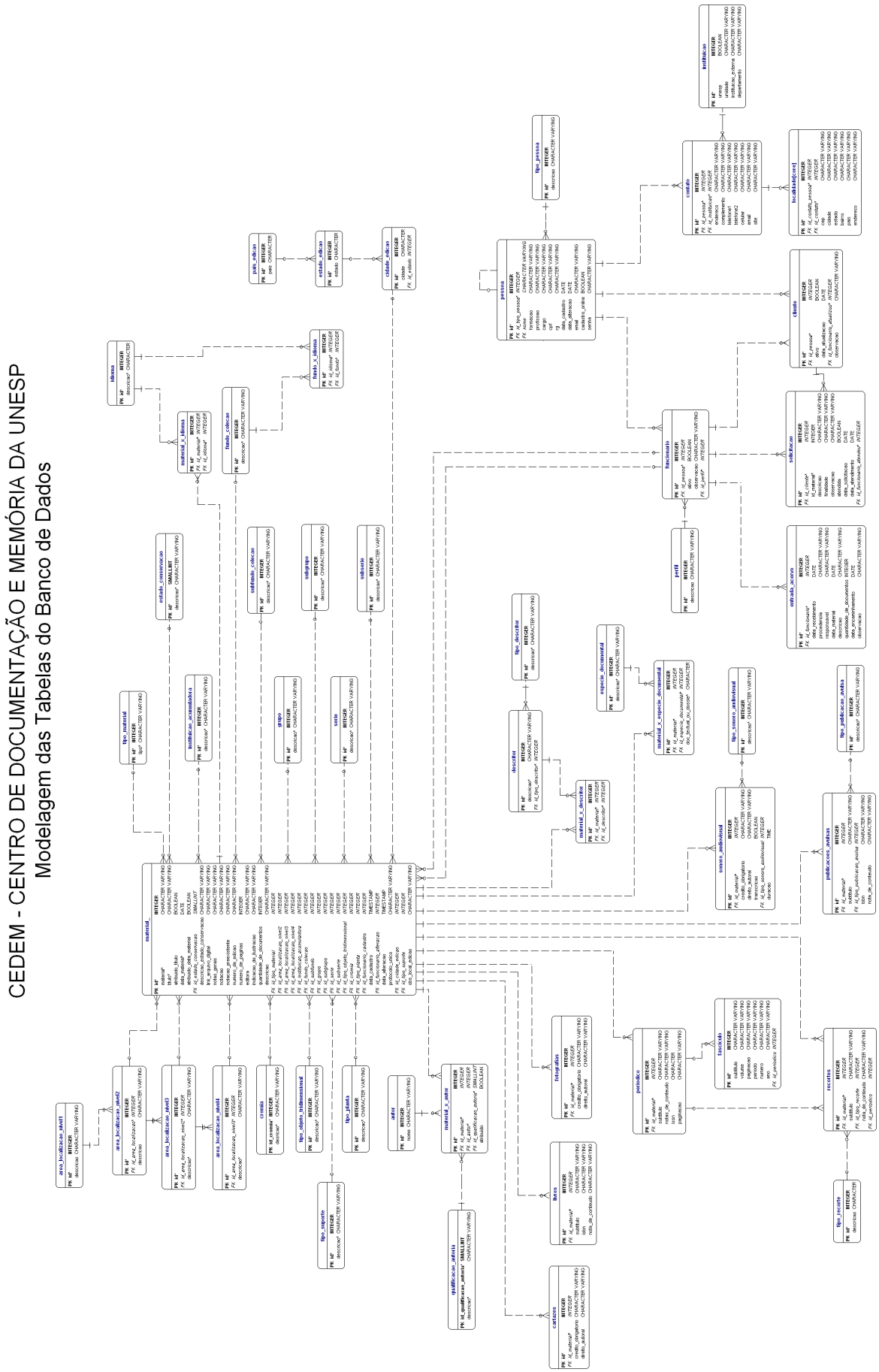

Figura 7: Modelagem resultante da reunião com representantes do CEDEM.

As principais alterações sofridas na modelagem com a reunião foram a eliminação das tabelas encarregadas de lidar com empréstimos de materiais e prazo, e a exclusão de tabelas que representam subtipos de materiais, que por apresentarem poucos campos exclusivos, sofreram normalizaçao e foram incorporados à tabela **material** ou em outras tabelas menores. São exemplos de normalização as tabelas **Documentos Sonoros** e **Audiovisuais** que passaram a ser representados em uma tabela chamada **sonoro_audiovisual**.A tabela **planta** foi eliminada, e um campo chamado **tipo_planta** foi adicionado a tabela **material**.

4.4 Codificação

4.4.1 Janeiro de 2011 a Julho de 2011

As atividades foram retomadas no primeiro dia de fevereiro de 2011. As primeiras ações a serem tomadas foram a realização de testes de mesa e simulação de inserções na modelagem otimizada (Figura 7). Após a realização da simulação e dos testes de mesa, foi disponibilizada uma conta de acesso ao repositório de projetos do Core em Bauru. Foi gerada uma versão de modelo de um projeto do Core nessa conta, acessivel aos desenvolvedores via SVN. O projeto modelo é apenas um "esqueleto", isto é, contém apenas o essencial para ser reconhecido pelo projeto Sentinela.

A codificação foi iniciada pelas *beans*. A ordem de implementação das *beans* foi determinada pelo grau de complexidade, das mais simples com poucos campos até as mais complexas com muitos campos e representações de relações entre tabelas no banco de dados. Com esse critério, as primeiras *beans* a serem implementadas foram as *beans* do plano de classificação, e as *beans* que representam áreas de localização física e seus subníveis, como salas, armários, gavetas e pastas. As próximas *beans* foram as que definiam e descreviam tipos, por exemplo, tipo de material e tipo de planta. Foram então implementadas as textitbeans da área temática (*beans* que representam os descritores), da área administrativa (representando usuários, clientes e funcionários), e das *beans* representantes das entradas de itens nos acervos do CEDEM e das solicitações oriundas de pesquisa. Depois foi criada a *bean* material, suas relações com outras beans, e as *beans* que derivam de material.

A fase seguinte foi a fase de implementação das *services* ,*actions* e páginas JSP respeitando a ordem acima. Porém, em incrementos de sistemas MVC encapsulados, isto é, implementando as *services*, *actions*, e páginas JSP de cadastro e pesquisa respectivas a uma tabela de cada vez e efetuando os testes necessários para confirmar o funcionamento. No mês de julho de 2011

as informações adicionais, que incluem o plano de classificação, áreas de localização física e tabelas auxiliares de tipo, já estavam com seus ”micro-sistemas em funcionamento“. Metade da área administrativa, que rege funcionários, entrada de itens, dentre outros encontrava-se em funcionamento.

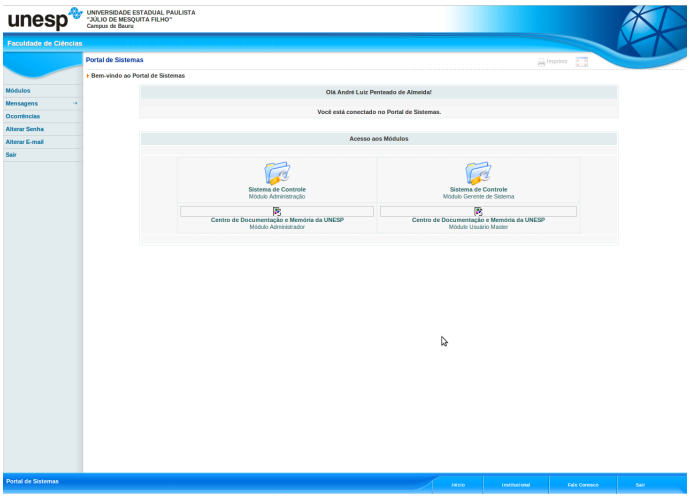

Figura 8: Portal de Sistemas Institucionais.

4.4.2 As Mudanças no padrão de desenvolvimento

Decorrente de um congresso de tecnologia em julho de 2011, foram apresentadas mudanças no padrão de desenvolvimento de projetos que utilizam o Core. Tabelas auxiliares que definem tipos foram substituídas por *enuns*, que são tipos enumeradores, que, no caso do Core, retornam descrições quando alimentados com índices numéricos de identificação. O sistema do CEDEM transformou três tabelas em *enuns*.

As principais mudanças estruturais para o desenvolvimento do sistema do CEDEM no entanto foram a inclusão de meios de autenticação, que foi realizada com a inclusão do projeto Admin a ser executado com o Sentinela, e um maior foco para a classe KGlobal. No KGlobal são inseridos os perfis de usuário dos sistemas, e as funcionalidades as quais estes perfis de usuário tem acesso. O projeto Admin permite controlar minunciosamente o acesso dos usuários, permitindo e restringindo funcionalidades especificadas no KGlobal e disponibilizando ou ocultando o acesso de perfis a sistemas. Sistemas que um dado perfil de usuário não tem acesso não é visível no portal de sistemas. Sofreu alteração também a nomenclatura básica de classes, tornando-se simplificada. (Figura 8).

A modelagem do sistema do CEDEM foi discutida e sofreu mudanças. Os itens do plano de classificação, antes independentes entre si, passaram a ter relações. Instituições acumuladoras

passaram a conter fundos e coleções, que por sua vez passaram a conter subfundos. Subfundos passaram a conter grupos, estes que passaram a conter subgrupos. Subgrupos passaram a conter séries e séries passaram a conter subséries. A seção administrativa sofreu uma redução, já que a responsabilidade pela autenticação de usuários é do projeto Sentinela. Com o novo ajuste na modelagem (Figura 9), o esquema de dados passou a ter 46 tabelas.

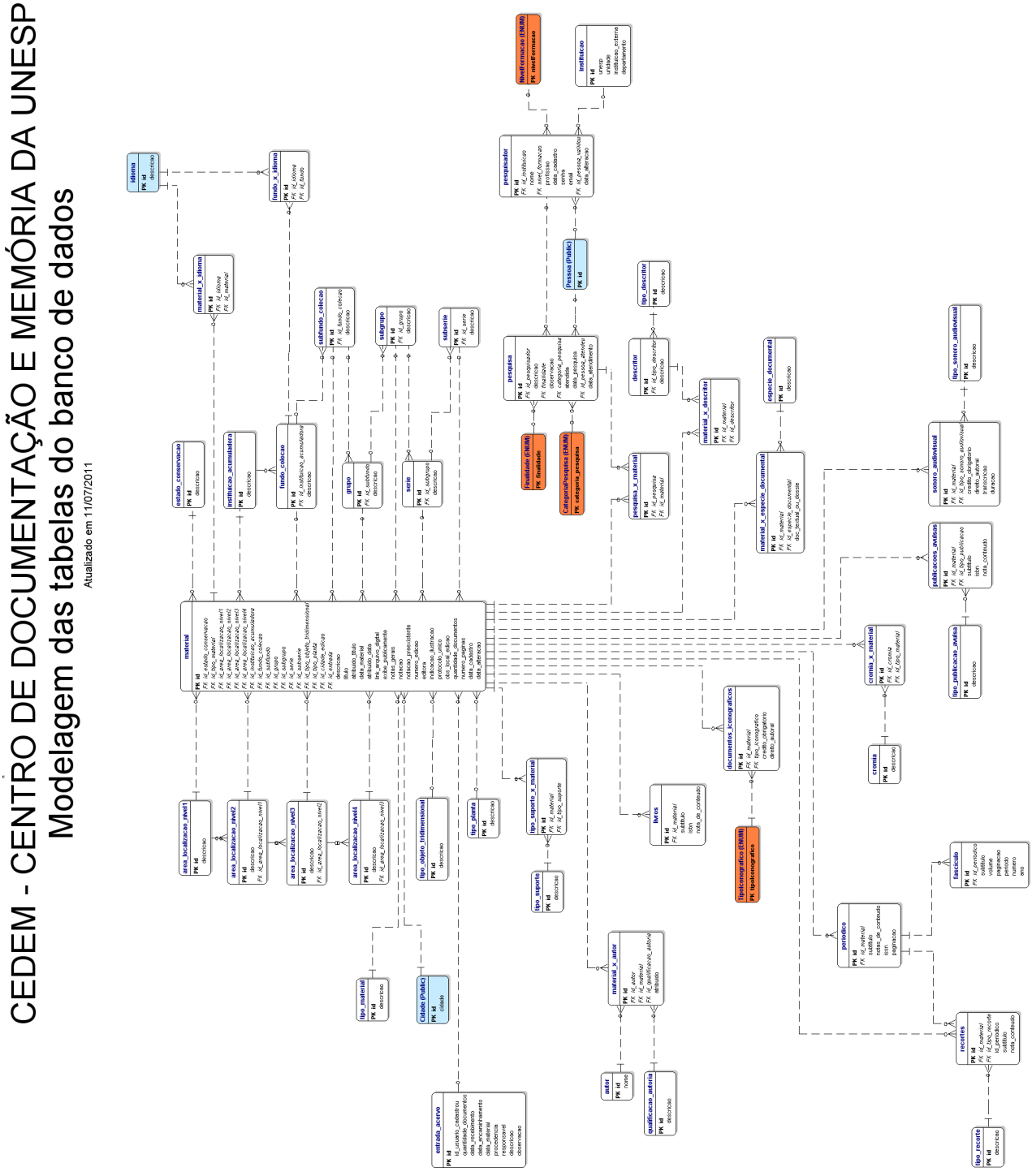

Figura 9: Ultima correção da modelagem, que será utilizada até o fim da primeira versão estável do sistema.

Como essas mudanças ocorreram durante a fase de codificação, a alteração dos componentes que estavam funcionando necessitou de mais trabalho. Houve alteração, verificação e validação das partes alteradas, e estas correções foram mais trabalhosas. O processo de correção de páginas JSP, e suas *actions*, *services* e *beans*, exigiu tempo.

Implementadas as seções de informações adicionais e as informações administrativas não dependentes de material, o desenvolvimento foi focado na tabela material e suas tabelas ”satélite“. Diversas abordagens foram cogitadas, mas só duas foram iniciadas: Tratar material apenas pelos seus subtipos (Livros, Fotografias, dentre outros), abordagem esta que foi abandonada por diferir da inteface cujos funcionários do CEDEM estão habituados, e tratar material como um todo e atrelar as informações correspondentes aos subtipos ao processo, sendo mais próximo da interface do sistema atual do CEDEM. Seguindo essa abordagem, o formulário de cadastro de materiais tem um tamanho considerável (Figura 10) e para evitar erros e estados inconsistentes, foram muito utilizados códigos *ajax*, *javascript* e *jquery*.

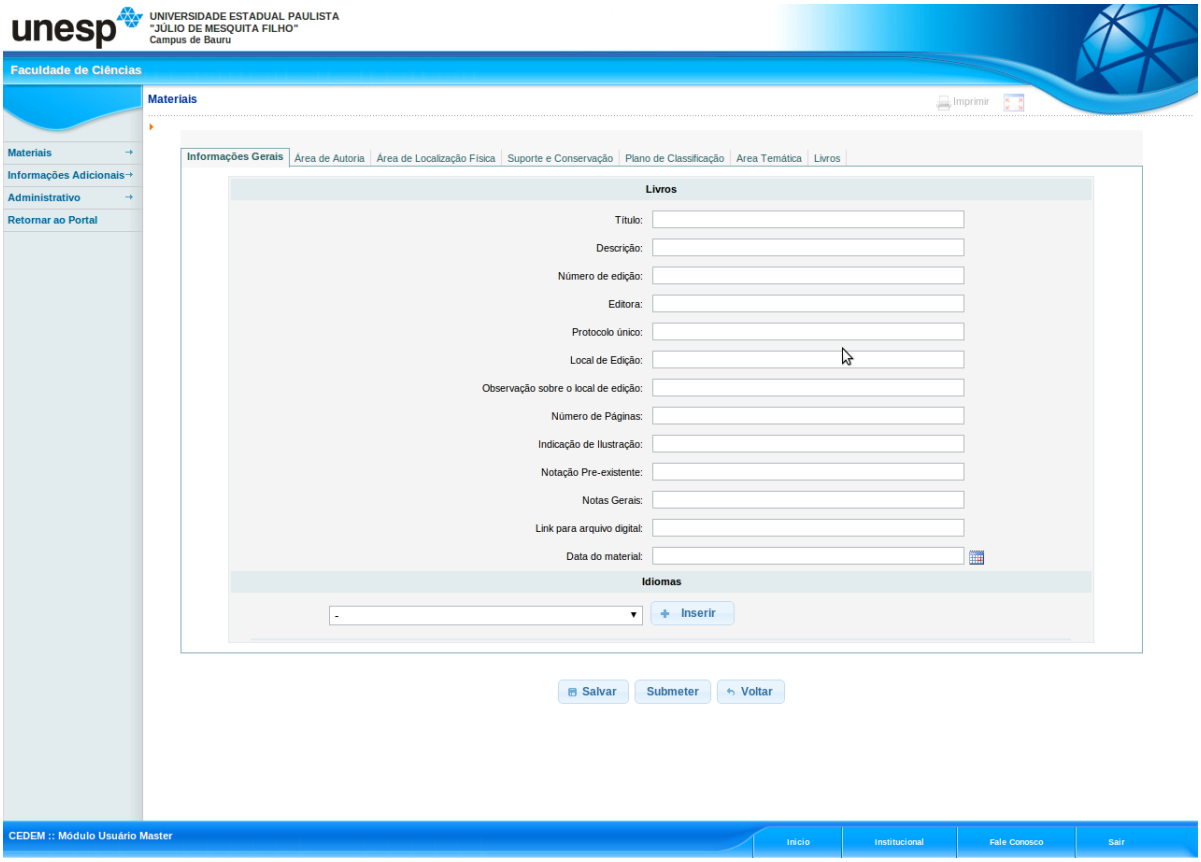

Figura 10: Formulário de cadastro de materiais

4.5 Reunião por video-conferência em 9 de novembro de 2011

Foi agendada uma reunião por vídeo-conferência entre a equipe de desenvolvimento do sistema do CEDEM e as duas representantes que visitaram presidente prudente em 2010, Solange e Sandra. A reunião teve como foco o andamento do novo sistema. As representantes do CEDEM constataram erros de terminologia, isto é, textos a serem alterados. Foi requisitada a equipe de desenvolvimento permitir mais informações a serem cadastradas nas tabelas do plano de classificação, requisito este não introduzido ao sistema previamente. Foi tratada a hospedagem de arquivos no futuro servidor do CEDEM, e a migração de dados para o banco de dados do novo sistema. A migração, por sua vez, deverá ser realizada quando a primeira versão estável do sistema estiver finalizada, para estar apta a armazenar os dados da base de dados antiga.

A avaliação do sistema em desenvolvimento foi muito positiva, e gerou incentivos para que o CEDEM institucionalize seu novo sistema, fazendo com que o Comitê Superior de Tecnologia da Informação, visando garantir a continuidade do desenvolvimento do novo sistema, disponibilize novos recursos à equipe. Com isso, foi encerrada a reunião.

Capítulo 5

MIGRANDO OS DADOS PARA O NOVO SISTEMA

Conforme a última reunião efetuada com representantes do CEDEM via vídeo-conferência no dia nove de novembro de 2011, foi reforçada a importância da migração dos dados entre os sistemas, pois nos dados armazenados no banco de dados atual estão contidos 20 anos de atividades e trabalhos do orgão.Tendo isso em vista, foi feito um estudo para verificar a maneira mais produtiva em termos de tempo, recursos e reaproveitamento para esse fim.

5.1 Pesquisa

Uma pesquisa foi efetuada para verificar procedimentos tomados em outros projetos quando deparados com a situação da migração. Todos os casos que apresentaram uma documentação clara e precisa seguiram basicamente os mesmos passos, com poucas diferenças. As etapas básicas são a criação de um arquivo de restauração no banco de dados de origem, a criação de um esquema de dados análogo no banco de dados de destino e o uso de um algoritmo ou uma ferramenta para efetuar a migração.(MIGRAçãO. . . , 2011)

No caso do sistema do CEDEM, onde o novo esquema de dados foi desenvolvido para uma arquitetura diferente e com foco diferente, os esquemas de dados não são equivalentes. Com base nisso foi feito o estudo de diversas ferramentas para avaliar se, dentre as disponíveis, existia uma alternativa já implementada. As ferramentas que mais se aproximaram deste objetivo foram DBTools e Data Transformation Services, ou *DTS*.

A ferramenta DBTools, open-source, efetua a migração entre dados. Porém, além de sua limitação em trabalhar com esquemas simétricos entre os bancos de dados diferentes, não recupera tabelas com quantidades elevadas de dados (na ordem de 1.000.000 de registros).

A ferramenta *DTS* é um utilitário do banco de dados SQLServer, que consegue lidar eficientemente com situações de migração de dados. Entretanto, devido a necessidade de bases de dados similares, não será utilizado.

5.2 Resolução Provável

Como não foi encontrada uma ferramenta completa e open-source para efetuar a migração, a UNESP desenvolverá uma ferramenta para isso, provavelmente em *PHP*. A ferramenta realizará o seguinte processamento:

1. Estabelecer conexão com o banco de dados de origem.
2. Estabelecer conexão com o banco de dados de destino.
3. Efetuar uma consulta por todos os registros de uma ou mais tabelas no banco de dados de origem.
4. Armazenar os resultados da consulta anterior em coleções de classes de persistência do banco de dados de origem.
5. Transferir os dados das classes de persistência do banco de dados de origem para classes de persistência do banco de dados de destino.
6. Inserir dados restantes e efetuar ajustes nos dados das classes de persistência do banco de dados de destino.
7. Inserir as classes de persistência do banco de dados de destino no banco de dados de destino.
8. Repetir a partir da terceira etapa para todas as tabelas do banco de dados de origem.

O primeiro passo pode ser substituído, se for conveniente, pela leitura de uma cópia de segurança do banco de dados de origem. A migração de dados será efetuada assim que a primeira versão estável do novo sistema do CEDEM for verificada e validada, pois será a versão que receberá os dados do sistema atual.

Capítulo 6

SITUAÇÃO ATUAL DO NOVO SISTEMA DO CEDEM

Este capítulo discorrerá sobre o estado atual do novo sistema do CEDEM. Será feita uma breve análise do que foi feito, do que deverá ser alterado e do que não pode ser feito até então. O objetivo é expor a situação deixada para a retomada do desenvolvimento no início do próximo ano.

6.1 Etapas concluídas

Uma ampla parcela do sistema foi implementada. Toda a parte correspondente a estrutura, representada no sistema como informações adicionais (plano de classificação, áreas de localização fisica, autoria, área temática) , encontra-se implementada e apenas aguardando correções de terminologia a serem enviadas por representantes do CEDEM. A parte administrativa encontra-se quase completa, com os aspectos de pesquisadores, entradas de itens nos acervos e instituições já funcionais. A parte de materiais, o núcleo do novo sistema do CEDEM, está avançando, e deve tornar-se funcional em pouco tempo.

6.2 Etapas em andamento

O cadastro e relatório de materiais, o núcleo do sistema, está em andamento, porém deve ser finalizada em um curto espaço de tempo. Com a finalização da seção de materiais e subtipos de material, tambem será finalizada a ultima pendência da área administrativa, que são as solicitações para disponibilizar um item do acervo em formato digital, dependente da tabela de materiais. Com isso, a primeira versão estável do sistema será concluída. As correções serão

efetuadas na medida em que serão recebidas.

6.3 Etapas futuras

As etapas futuras do desenvolvimento são divididas em três grandes grupos. O primeiro deles é o refinamento do sistema atual, que será contínuo. O segundo deles envolve a migração de dados do banco de dados do sistema atual do CEDEM, o SQLServer para o novo sistema do CEDEM, que utiliza o PostgreSQL. Será desenvolvida uma ferramenta que acessa os dados armazenados no sistema antigo, os processa e converte para o novo formato de dados e os grava no sistema em desenvolvimento. E o último grande grupo envolve a hospedagem dos itens dos acervos do CEDEM em formatos digitais, mas este grupo, assim como a migração de dados, depende da finalização da primeira versão estável do sistema.

Capítulo 7

Conclusão

Duas das principais preocupações de recém-formados são a inserção e a adaptação ao mercado de trabalho. A experiência prática adquirida durante o estágio supervisionado foi extremamente positiva, proporcionando experiência em diversos campos, tais quais o cumprimento de horário e de prazos, o trabalho em equipe, a boa comunicação, o processo de elaborar soluções e traçar metas a serem buscadas na implementação. E, Mais importante, como aplicar o conhecimento teórico adquirido durante as aulas para solucionar problemas práticos. Com isso, além de se habituar com a vida no mercado de trabalho, permitiu experiência, que garante que o formando tenha mais facilidade de se adequar em um trabalho futuro.

Referências Bibliográficas

AJAX: A New Approach to Web Applications. 2011. Disponível em: <http://adaptivepath.com/ideas/ajax-new-approach-web-applications>.

APACHE Commons. 2011. Disponível em: <http://commons.apache.org/>.

DETALHES do produto Servidor em torre PowerEdge T410 11G. 2011. Disponível em: <http://www.dell.com/br/empresa/p/poweredge-t410/pd>.

DISPLAY tag library - Overview. 2011. Disponível em: <http://www.displaytag.org/1.2/>.

EQUIPE da FCT desenvolverá sistema para um dos maiores centros de documentação e memória do país. 2010. Disponível em: <http://www.fct.unesp.br/index.php?CodDivulgacao=99>.

GUJ - Introdução ao Log4J. 2011. Disponível em: <http://www.guj.com.br/articles/130>.

HIBERNATE - JBoss comunity. 2011. Disponível em: <http://www.hibernate.org>.

JPA. 2011. Disponível em: <http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html>.

MENTAWAI. 2011. Disponível em: <http://www.vivaolinux.com.br/artigo/Introducao-ao-framework-Mentawai>.

MIGRAçãO de Banco de Dados Oracle para PostgreSQL. 2011. Disponível em: <http://www.cnptia.embrapa.br/files/doc71.pdf>.

MVC - Wikipédia, a enciclopédia livre. 2011. Disponível em: <http://pt.wikipedia.org/wiki/MVC>.

REITORIA - Centro de Documentação e Memória da UNESP. 2011. Disponível em: <http://www.cedem.unesp.br/>.

RELATORIOS em Java. 2011. Disponível em: <http://www.dsc.ufcg.edu.br/ jacques/cursos/daca/html/documentviews/relatorios.htm>.

SITEMESH3 Overview. 2011. Disponível em: <http://www.sitemesh.org/overview.html>.

SLONY-L - Wikipedia, the free encyclopedia. 2011. Disponível em: <http://en.wikipedia.org/wiki/Slony-I>.

SUBVERSION - Wikipédia, a enciclopédia livre. 2011. Disponível em: <http://pt.wikipedia.org/wiki/Subversion>.

WHY use? - www.dspace.org. 2011. Disponível em: <http://www.dspace.org/why-use>.

XML - Wikipédia, a enciclopédia livre. 2011. Disponível em: <http://pt.wikipedia.org/wiki/XML>.

Apêndice A

APÊNDICE A: DSPACE E O NOVO SISTEMA DO CEDEM

A.1 Dspace

Dspace é um sistema de repositório digital open source com a licença BSD, que é uma licensa que permite a utilização gratuíta do software. Concebido e implementado pelo *Massachussetts Institute of Technology* (MIT), a plataforma de repositórios Dspace é altamente configurável, podendo ser adaptada para inúmeros casos e cenários diferentes, razão pela sua crescente popularidade. Atualmente mais de 1000 organizações utilizam a plataforma. Uma de suas características mais elogiadas e responsável pelo favoritismo é a capacidade de organização de repositórios.

A capacidade de personalização do Dspace é alta, pondendo ser personalizados seus temas visuais, suas buscas e navegações em conteúdos, metadados de itens de acervos, autenticação, base de dados, critério de organização de itens e acervos e idiomas.

O Dspace consegue reconhecer e gerenciar os tipos mais comuns de arquivos, sendo que tipos de arquivos novos podem ser adicionados em sua base de dados, para que sejam gerenciados pelo sistema. (WHY..., 2011)

A.2 Opção pelo desenvolvimento

Conforme a descrição resumida acima, o Dspace foi considerado como uma possível solução para o novo sistema do CEDEM. O propósito do sistema, a maneira como o Dspace permite

o manuseio e categorização de acervos e itens. e a autenticação apresentam uma similaridade com o projeto do novo sistema do CEDEM.

A opção pelo desenvolvimento de um novo sistema foi feita por uma série de motivos. O primeiro deles, é que o novo sistema estaria em conformidade com os requisitos do CEDEM. O segundo motivo é a padronização, o ideal é que os sistemas institucionais sigam o mesmo padrão.O terceiro motivo é a integração com o esquema público do banco de dados do Core, que é o esquema comum. Através dessa integração, é possível recuperar dados comuns no contexto da universidade e do CEDEM, como nome de cidades e estados nacionais, ou informações de usuários dos sistemas institucionais. O quarto motivo é a integração do sistema do CEDEM com outros sistemas institucionais em um momento futuro.

Concluindo, apesar de o Dspace apresentar uma ótima solução, não é a solução ideal.E portanto, uma universidade do porte da UNESP tem plena estrutura, e profissionais de qualidade para construir sua solução ideal.

Apêndice B

APÊNDICE B: EXEMPLOS DE CÓDIGO-FONTE

Neste capítulo são apresentados exemplos de código-fonte. As classes escolhidas como exemplo formam um micro-sistema na arquitetura MVC, representando a tabela *fundo_colecao* do esquema de dados do sistema do CEDEM. As classes FundoColecao, do tipo *bean* e FundoColecaoService, do tipo service representam a camada de modelo. A classe FundoColecaoAction, do tipo *action* representa a camada de controle, e as páginas JSP cadastro e pesquisar representam a camada de visualização.

B.1 A classe FundoColecao

A classe FundoColecao, do tipo *bean* da camada de modelo da arquitetura MVC, é a responsável por mapear os atributos da tabela *fundo_colecao* em objeto. Abaixo segue o código-fonte:

```
1  package br.unesp.cedem.beans;
2
3  import java.io.Serializable;
4
5  import javax.persistence.Column;
6  import javax.persistence.Entity;
7  import javax.persistence.GeneratedValue;
8  import javax.persistence.GenerationType;
9  import javax.persistence.Id;
10 import javax.persistence.JoinColumn;
11 import javax.persistence.ManyToOne;
```

```
12 import javax.persistence.SequenceGenerator;
13 import javax.persistence.Table;
14
15
16 import br.unesp.core.GenericDAO;
17
18 @Entity
19 @Table(name="cedem.fundo_colecao")
20
21 public class FundoColecao extends GenericDAO<FundoColecao> implements
       Serializable{
22
23
24         /**
25          *
26          */
27         private static final long serialVersionUID = 5346555117916002909L;
28
29         @Id
30         @SequenceGenerator
31                         (name = "SEQ", sequenceName = "cedem.
                               seq_fundo_colecao",allocationSize =1)
32         @GeneratedValue
33                         (strategy = GenerationType.SEQUENCE, generator = "
                               SEQ")
34         @Column (name = "id", nullable = false)
35         private Long id;
36
37         @Column (name = "descricao", nullable = false)
38         private String descricao;
39
40         @ManyToOne
41         @JoinColumn (name = "id_instituicao_acumuladora",nullable = true)
42         private InstituicaoAcumuladora instituicaoAcumuladora;
43
44         public static long getSerialversionuid() {
45                 return serialVersionUID;
46         }
47
48         @Override
49         public Long getId() {
50                 return this.id;
51         }
```

```
52
53          @Override
54          public void setId(Long id) {
55                  this.id = id;
56          }
57
58          public void setDescricao(String descricao) {
59                  this.descricao = descricao;
60          }
61
62          public String getDescricao() {
63                  return descricao;
64          }
65
66          public void setInstituicaoAcumuladora(InstituicaoAcumuladora
                instituicaoAcumuladora) {
67                  this.instituicaoAcumuladora = instituicaoAcumuladora;
68          }
69
70          public InstituicaoAcumuladora getInstituicaoAcumuladora() {
71                  return instituicaoAcumuladora;
72          }
73
74  }
```

Os comandos precedidos pelo caracter arroba são *Annotations* do Hibernate, responsáveis por mapear os campos da tabela na classe.As *Annotations* **ManyToOne** e **JoinColumn** nas linhas 40 e 41, respectivamente, indicam que o objeto do tipo *InstituicaoAcumuladora* representa, no banco de dados, a chave estrangeira da tabela *instituicao_acumuladora* e a relação de um para muitos entre as duas tabelas. Pelo código fonte, nota-se também que não há o uso de tipos primitos do Java no Core. A *Annotation* **Column** sincroniza o campo da tabela do banco de dados com o atributo da clase. O grupo de comandos que se inicia com a *Annotation* **Id** na linha 29 e termina na linha 33 é responsável por efetuar a função de auto-incremento do campo id, função ausente no PostgreSQL.

B.2 A classe FundoColecaoService

A classe FundoColecaoService, do tipo *service* e pertencente a camada de modelo do *Core*, tem a função de utilizar a *bean* para efetuar consultas e atualizações na tabela *fundo_colecao*.

Abaixo encontra-se o código-fonte:

```
1  package br.unesp.services;
2
3  import java.util.List;
4
5  import br.unesp.beans.UserLoginWrapper;
6  import br.unesp.core.AbstractBasicService;
7  import br.unesp.core.Log4jWrapper;
8  import br.unesp.exception.ServiceValidationException;
9  import br.unesp.cedem.beans.FundoColecao;
10 import br.unesp.cedem.beans.InstituicaoAcumuladora;
11
12 public class FundoColecaoService extends AbstractBasicService {
13
14         private Log4jWrapper log = new Log4jWrapper(FundoColecaoService.
               class, null);
15
16         public FundoColecaoService() {
17         }
18
19         FundoColecaoService(Object[] params){
20                 if(params[0] instanceof UserLoginWrapper){
21                         UserLoginWrapper userLogin = (UserLoginWrapper)
                            params[0];
22                         this.log = new Log4jWrapper(FundoColecaoService.
                            class, userLogin);
23                 }
24         }
25
26         /**
27          * Persiste o objeto fundoColecao no banco de dados
28          * @param FundoColecao
29          * @throws ServiceValidationException
30         */
31         public void gravarFundoColecao(FundoColecao fundoColecao)
32                 throws ServiceValidationException{
33                         if(fundoColecao.getId()==null)
34                                 fundoColecao.setId(-1L);
35                         //valida   o
36
37
38
39                         saveAndCommit(fundoColecao);
```

```
40          }
41
42          public List<FundoColecao> filtrarFundoColecaoPorDescricao(String
                descricao) {
43                  StringBuilder hql = new StringBuilder();
44                  Object[][] param = new Object[][] {{"pDescricao","%".concat
                        (descricao).concat("%")}};
45
46                  hql.append("SELECT c FROM br.unesp.cedem.beans.FundoColecao
                         c WHERE UPPER(c.descricao) LIKE UPPER(:pDescricao) ");
47                  hql.append("ORDER BY c.descricao");
48
49                  return (List<FundoColecao>)executeQuery(hql.toString(),
                        param, -1);
50          }
51
52          public List<FundoColecao> filtrarFundoColecaoPorInstituicao(
                InstituicaoAcumuladora instituicao) {
53                  StringBuilder hql = new StringBuilder();
54                  Object[][] param = new Object[][] {{"pPai",instituicao}};
55
56                  hql.append("SELECT c FROM br.unesp.cedem.beans.FundoColecao
                         c WHERE (c.instituicaoAcumuladora) = (:pPai) ");
57                  hql.append("ORDER BY c.id");
58
59                  return (List<FundoColecao>)executeQuery(hql.toString(),
                        param, -1);
60          }
61
62          public FundoColecao buscarFundoColecaoPorId(int id) {
63                  return (FundoColecao)findByPK(FundoColecao.class, id);
64          }
65
66          public void excluirFundoColecao(int id){
67                  deleteAndCommit(buscarFundoColecaoPorId(id));
68          }
69
70  }
```

Na linha catorze é criada uma instância de um objeto de *logging* da biblioteca *Log4j*, para depuração das operações da classe. No método gravar (linha 31) é salvo o objeto *fundoCole-*

cao proveniente da classe FundoColecaoAction, onde este recebe os valores adequados para inserção na tabela. O método filtrarFundoColecaoPorInstituicao (linha 52) retorna uma lista com os objetos da classe FundoColecao em que o atributo instituicaoAcumuladora é o objeto do tipo instituicaoAcumuladora passado como parâmetro, usando HQL para comparar objetos ao invés de dados na base de dados.

B.3 A classe FundoColecaoAction

A classe FundoColecaoAction, do tipo *action* e pertencente a camada de modelo na arquitetura MVC, tem a função de recuperar dados da entrada, tratá-los, e redirecioná-los para as outras camadas.

```
1  package br.unesp.cedem.actions;
2
3  import br.unesp.exception.ServiceValidationException;
4
5  import org.mentawai.authorization.Authorizable;
6  import org.mentawai.core.BaseAction;
7  import org.mentawai.list.ListManager;
8  import org.mentawai.list.SimpleListData;
9
10 import java.lang.reflect.Array;
11 import java.util.List;
12 import java.util.ArrayList;
13 import java.util.Arrays;
14 import br.unesp.annotations.ActionClass;
15 import br.unesp.annotations.ConsequenceOutput;
16 import br.unesp.annotations.Consequences;
17 import br.unesp.beans.UserLoginWrapper;
18 import br.unesp.core.ConfigHelper;
19 import br.unesp.core.Log4jWrapper;
20 import br.unesp.cedem.ActionsManager;
21 import br.unesp.cedem.beans.AreaLocalizacaoNivel1;
22 import br.unesp.cedem.beans.FundoColecao;
23 import br.unesp.cedem.beans.Idioma;
24 import br.unesp.cedem.beans.FundoXIdioma;
25 import br.unesp.cedem.beans.InstituicaoAcumuladora;
26 import br.unesp.services.ServicesFactory;
27 import br.unesp.services.FundoColecaoService;
28 import br.unesp.services.FundoIdiomaService;
```

```
29 import br.unesp.services.IdiomaService;
30 import br.unesp.services.InstituicaoAcumuladoraService;
31
32 @ActionClass (prefix="/cadastro/fundoColecao")
33 public class FundoColecaoAction
34         extends BaseAction implements Authorizable
35 {
36         private Log4jWrapper log =
37                 new Log4jWrapper(FundoColecaoAction.class,null);
38
39         private FundoColecaoService FundoColecaoService = null;
40         private IdiomaService idiomaService = null;
41         private FundoIdiomaService fundoXIdiomaService = null;
42         private InstituicaoAcumuladoraService instituicaoAcumuladoraService
               = null;
43
44         public void instanciarServicos(UserLoginWrapper userLogin){
45                 try {
46                         FundoColecaoService = (FundoColecaoService)
                            ServicesFactory.
47                         getInstance(FundoColecaoService.class, new Object[]
                             {userLogin});
48
49                         idiomaService = (IdiomaService)ServicesFactory.
50                         getInstance(IdiomaService.class, new Object[] {
                            userLogin});
51
52                         fundoXIdiomaService = (FundoIdiomaService)
                            ServicesFactory.
53                         getInstance(FundoIdiomaService.class, new Object[]
                            {userLogin});
54
55                         instituicaoAcumuladoraService = (
                            InstituicaoAcumuladoraService)ServicesFactory.
56                         getInstance(InstituicaoAcumuladoraService.class,
                            new Object[] {userLogin});
57
58                         log = new Log4jWrapper(FundoColecaoAction.class,
                            userLogin);
59                 }
60                 catch(Exception ex){
61                         log.fatal("A   ES  N  O  INSTANCIADAS:".
62                         concat(FundoColecaoAction.class.getName()), ex);
```

```
63              }
64      }
65
66      @Override
67      public boolean authorize(String innerAction, Object user, List
            groups)
68      {
69              if(user == null || groups == null)
70                      return false;
71              instanciarServicos((UserLoginWrapper)user);
72              return true;
73
74      }
75
76      @Consequences(outputs = @ConsequenceOutput(page="/cadastro/
            fundoColecao/cadastro.jsp"))
77              public String cadastrar()
78              {
79                      try{
80                              FundoColecao FundoColecao = null;
81
82                              if(input.getInt("txt_id") > 0)
83                                      FundoColecao = FundoColecaoService.
                                            buscarFundoColecaoPorId(input.
                                            getInt("txt_id"));
84
85                              if(FundoColecao == null)
86                                      FundoColecao = new FundoColecao();
87
88                              SimpleListData sldInstituicaoAcumuladora
                                        = new SimpleListData();
89
90                              for (InstituicaoAcumuladora item :
                                    instituicaoAcumuladoraService.
                                    filtrarInstituicaoAcumuladoraPorDescricao
                                    (""))
91                                      sldInstituicaoAcumuladora.add(item.
                                            getId().toString(),item.
                                            getDescricao()); //sobrescrever
                                            metodo toString()?
92
93                              ListManager.addList(
                                    sldInstituicaoAcumuladora);
```

```
94
95                              output.setValue("
                                    lista_instituicao_acumuladora",
                                    sldInstituicaoAcumuladora);
96
97                              output.setValue("FundoColecao",FundoColecao
                                    );
98                              output.setValue("listaIdioma",ListManager.
                                    convert("listaIdioma",idiomaService.
                                    filtrarIdiomaPorDescricao(""),"id","
                                    descricao"));
99
100
101                         }
102                         catch(Exception ex){
103                                 addError(ConfigHelper.get().getString("
                                        error.general"));
104                                 log.error(ConfigHelper.get().getString("
                                        error.general"),ex);
105                                 return ERROR;
106                         }
107                         return SUCCESS;
108                 }
109
110         @Consequences(outputs =
111                 { @ConsequenceOutput
112                         (result = SUCCESS,page = ActionsManager.
                                JSP_SHOW_MESSAGE),
113                 @ConsequenceOutput
114                         (result = ERROR,page = "/cadastro/
                                fundoColecao/cadastro.jsp")
115
116                 })
117
118         public String gravar(){
119
120                 String txtIdioma = input.getString("txt_idioma");
121                 String idIdioma = input.getString("id_idioma");
122                 String aux = "";
123                 FundoXIdioma fundoXIdioma = new FundoXIdioma();
124
125                 List<String> listaId = new ArrayList();
126
```

```
127             for(int i = 0; i < idIdioma.length(); i++) {
128
129                     if(idIdioma.charAt(i) != ',') {
130                             aux = aux + idIdioma.charAt(i);
131                     }
132
133                     else {
134                             if(!aux.equals("")) listaId.add(aux);
135                             aux = "";
136                     }
137             }
138             if(!aux.equals("")) listaId.add(aux);
139
140                     try{
141                             FundoColecao fundoColecao = new
                                    FundoColecao();
142
143                             InstituicaoAcumuladora instituicao = null;
144
145                             instituicao = instituicaoAcumuladoraService
                                    .buscarInstituicaoAcumuladoraPorId(input
                                    .getInt("sel_instituicao_acumuladora"));
146
147
148                             if(input.getInt("txt_id")> 0)
149                                     fundoColecao = FundoColecaoService.
                                            buscarFundoColecaoPorId(input.
                                            getInt("txt_id"));
150
151                             if (fundoColecao == null)
152                                     fundoColecao = new FundoColecao();
153
154                             fundoColecao.setDescricao(input.getString("
                                    txt_descricao").trim());
155                             fundoColecao.setInstituicaoAcumuladora(
                                    instituicao);
156
157                             FundoColecaoService.gravarFundoColecao(
                                    fundoColecao);
158
159
160                             for(int i = 0;i < listaId.size();i++) {
161                                     //fundoXIdioma.setId(-1L);
```

```
162                                     fundoXIdioma.setIdioma(
                                            idiomaService.buscarIdiomaPorId(
                                            Integer.parseInt(listaId.get(i)
                                            ) ) );
163                                     fundoXIdioma.setFundo(fundoColecao)
                                            ;
164                                     fundoXIdiomaService.
                                            gravarFundoXIdioma(fundoXIdioma)
                                            ;
165                             }
166                     }
167                     catch(ServiceValidationException unespex)
168                     {
169                             addError(unespex.getMessage());
170                             return ERROR;
171                     }
172                     catch(Exception ex)
173                     {
174                             addError(ConfigHelper.get().getString("
                                    error.general"));
175                             log.error
176                                     (ConfigHelper.get().getString("
                                        error.general"),ex);
177                             return ERROR;
178                     }
179
180                     output.setValue("titulo", "Cadastro de Fundos e
                            Cole  es");
181                     output.setValue("mensagem", ConfigHelper.
                            getProperty("info.saveOk","Fundos e Cole  es")
                            );
182                     output.setValue("url", input.getProperty("
                            contextPath") + "/cadastro/fundoColecao.
                            pesquisar.action");
183                     return SUCCESS;
184             }
185
186             @Consequences(outputs =
187                     @ConsequenceOutput(page = "/cadastro/fundoColecao/pesquisar
                            .jsp"))
188                     public String pesquisar() {
189
190                     try {
```

```
191                     String descricao = "";
192                     if(input.getString("txt_descricao") != null)
193                             descricao = input.getString("txt_descricao"
                                );
194                     output.setValue("listaFundoColecao",
                            FundoColecaoService.
                            filtrarFundoColecaoPorDescricao(descricao));
195                     output.setValue("listaIdioma",idiomaService.
                            filtrarIdiomaPorDescricao(descricao));
196             }
197             catch (Exception ex) {
198                     addError(ConfigHelper.get().getString("error.
                            general"));
199                     log.error(ConfigHelper.get().getString("error.
                            general"), ex);
200             }
201             return SUCCESS;
202     }
203
204             @Consequences(outputs = {
205             @ConsequenceOutput (result = SUCCESS, page = ActionsManager
                    .JSP_SHOW_MESSAGE),
206             @ConsequenceOutput(result = ERROR,page = "/cadastro/
                    fundoColecao/pesquisar.jsp")})
207             public String excluir() {
208
209                     try {
210                             FundoColecaoService.excluirFundoColecao(
                                    input.getInt("txt_id"));
211                     }
212                     catch (Exception ex) {
213                             addError(ConfigHelper.get().getString("
                                    error.general"));
214                             log.error(ConfigHelper.get().getString("
                                    error.general"), ex);
215                             return ERROR;
216                     }
217
218                     output.setValue("titulo", "Exclus o de Fundos e
                            Cole  es");
219                     output.setValue("mensagem", ConfigHelper.
                            getProperty("info.deleteOk", "Fundos e
                            Cole  es"));
```

```
220             output.setValue("url", input.getProperty("
                    contextPath") + "/cadastro/fundoColecao.
                    pesquisar.action");
221
222             return SUCCESS;
223             }
224         //
            ////////////////////////////////////////////////////////////////

225         @Consequences(outputs =
226             @ConsequenceOutput(page = "/cadastro/idioma/
                    pesquisar.jsp"))
227             public String pesquisarIdioma() {
228
229             try {
230                 String descricao = "";
231                 if(input.getString("txt_descricao") != null
                        )
232                     descricao = input.getString("
                            txt_descricao");
233                 output.setValue("listaIdioma",idiomaService
                        .filtrarIdiomaPorDescricao(descricao));
234             }
235             catch (Exception ex) {
236                 addError(ConfigHelper.get().getString("
                        error.general"));
237                 log.error(ConfigHelper.get().getString("
                        error.general"), ex);
238             }
239             return SUCCESS;
240         }
241
242         @Consequences(outputs =
243         { @ConsequenceOutput
244             (result = SUCCESS, page = ActionsManager.
                    JSP_SHOW_MESSAGE),
245         @ConsequenceOutput
246                 (result = ERROR, page = "/cadastro/
                        fundoColecao/cadastro.jsp")
247
248         })
249         public void incluirIdioma() {
250             //List<Idioma> listaIdiomasSelecionados =
```

```
251                         System.out.println("
                                ************************************************
                                ");
252                     }
253
254 }
```

Na linha 32, o comando precedido por arroba é uma anotação do Mentawai, que indica onde se localiza o contexto da classe. O método instanciarServicos (linha 44) tem como objetivo instaciar as classes de tipo *service* que fazem a interface com o banco de dados.Nesta classe também é feita a autenticação do usuário (método authorize, linha 67). Na linha 76, o comando precedido de arroba é uma anotação do Mentawai, do tipo *consequence*, que diz para onde serão direcionados os dados do método que possui a anotação (no exemplo é feito o redirecionamento para a página cadastro.jsp).

Os atributos da *bean* são definidos nesta classe, e esta é redirecionada para a service. Devido a responsabilidade das classes da camada controladora, sua extensão é significantemente maior que a extensão das classes que representam os outros modelos.

B.4 A página de cadastro

A página JSP cadastro pertencente à camada de visualização da arquitetura MVC. Seu propósito é disponibilizar um formulário para o cliente inserir ou atualizar um registro na base de dados. Abaixo, encontra-se o código-fonte:

```
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2      pageEncoding="ISO-8859-1"%>
3  <%@taglib uri="http://www.mentaframework.org/tags-mtw/" prefix="mtw" %>
4  <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
5
6  <html>
7  <head>
8  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
9  <title>Fundos e Coleções</title>
10 <script type="text/javascript">
11 function buscaIdioma() {
12                     var ret = new Array();
13                     ret.push(new Array('id_idioma', 'id'));
```

```
14                  ret.push(new Array('txt_idioma', 'idioma'));
15                  //ret.push(new Array('txt_descricao', 'Idiomas'));
16                      pCtrl.search('<%=request.getContextPath()%>/cadastro/
                            idioma.buscarIdioma.action', 600, 320, ret);
17
18                  }
19 </script>
20 </head>
21 <body>
22
23 <!-- caminho correto? fundoColecaos.gravar.action, FundoColecaos.gravar.
       action, fundoColecaos.gravar.action -->
24
25 <form action="<%=request.getContextPath()%>/cadastro/fundoColecao.gravar.
       action" method="post" name="formFundoColecao" id="formFundoColecao">
26 <%-- <form action="<%=request.getContextPath()%>/cadastro/fundoColecao.
       incluirIdioma.action" method="post" name="formFundoColecao" id="
       formFundoColecao"> --%>
27
28          <mtw:input type="hidden" name="txt_id" id="txt_id" value="${
                FundoColecao.id}" />
29          <mtw:input type="hidden" name="txt_idioma" id="txt_idioma" value=""
                />
30          <mtw:input type="hidden" name="id_idioma" id="id_idioma" value=""
                />
31
32          <input type="hidden" name="id" id="id"/>
33
34
35          <table class="form">
36                      <tr>
37                              <th colspan="2">Fundos e Colees</th>
38                      </tr>
39                      <tr>
40                              <td width="40%" class="label">Descri </td
                                    >
41                              <td >
42                                      <mtw:inputText name="txt_descricao"
                                            value="${FundoColecao.descricao
                                            }" size="30" maxlength="40"/>
43                              </td>
44                      </tr>
45
```

```
46                          <tr>
47                                  <td class="label">Contido em: </td>
48                                  <td >
49                                          <mtw:select name="
                                                sel_instituicao_acumuladora"
                                                list="
                                                lista_instituicao_acumuladora"
                                                defValue="${FundoColecao.
                                                instituicaoAcumuladora.id}"
                                                emptyField="true"/>
50                                  </td>
51                          </tr>
52
53                          <tr>
54                                  <td width="40%" class="label">Idiomas: </td
                                        >
55                                  <td >
56                                          <input type="button" name="
                                                Selecionar Idiomas" value="
                                                Selecionar Idiomas" onclick="
                                                buscaIdioma()" class="botao"/>
57                                          <%-- <mtw:select name="sel_idioma"
                                                list="listaIdioma" defValue="${
                                                Idioma.id}" emptyField="true" />
                                                 --%>
58                                  </td>
59                          </tr>
60
61                          <tr>
62                                  <td colspan="2" class="barrabotao">
63                                          <input name="btn_salvar" type="
                                                submit" value="Salvar" class="
                                                botao">
64                                          <input name="btn_voltar" type="
                                                button" value="Voltar" class="
                                                botao" onclick="window.location.
                                                href='<%=request.getContextPath
                                                ()%>/cadastro/fundoColecao.
                                                pesquisar.action';">
65                                  </td>
66                          </tr>
67
68          </table>
```

```
69 </form>
70
71 </body>
72 </html>
```

B.5 A página de pesquisa

A página JSP pesquisar, pertencente a camada de visualização da arquitetura MVC. Seu propósito é disponibilizar dados préviamente armazenados pelo cliente, e a busca pode ser filtrada por critérios. Abaixo encontra-se o código-fonte:

```
1  <%@page contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO
       -8859-1"%>
2  <%@page errorPage="/common.exception.action" %>
3  <%@taglib uri="http://www.mentaframework.org/tags-mtw/" prefix="mtw" %>
4  <%@taglib uri="http://displaytag.sf.net" prefix="display" %>
5  <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
6  <%@taglib uri="http://www.fc.unesp.br/skin" prefix="skin" %>
7  <%@page import="br.unesp.core.ConfigHelper"%>
8
9  <c:set var="linkCadastro">
10         <%=request.getContextPath()%>/cadastro/fundoColecao.cadastrar.
               action
11 </c:set>
12 <c:set var="linkPesquisar">
13         <%=request.getContextPath()%>/cadastro/fundoColecao.pesquisar.
               action
14 </c:set>
15 <c:set var="linkExcluir">
16         <%=request.getContextPath()%>/cadastro/fundoColecao.excluir.action
17 </c:set>
18
19 <html>
20 <head>
21         <meta http-equiv="Content-Type"
22                 content="text/html; charset=iso-8859-1">
23         <meta name="header"
24                 content="Exibe a listagem dos fundoColecaos cadastrados" />
25         <title>Pesquisa de Fundos e Coleções</title>
26
27         <script language="JavaScript" type="text/javascript">
```

```
28                  function excluir(id, descricao) {
29              if (confirm('<%=ConfigHelper.getProperty("warning.delete", "")
                    %>' + descricao)) {
30                  document.location.href = '${linkExcluir}?txt_id=' + id;
31              }
32                  }
33          </script>
34
35  </head>
36  <body>
37
38
39
40          <center>
41          <form name="form_pesquisa_fundoColecaos" method="post" action="${
                linkPesquisar}">
42                  <table class="form">
43                          <tr>
44                                  <th colspan="2">Pesquisa de Fundos e
                                        Coleções</th>
45                          </tr>
46
47                          <tr>
48                                  <td width="20%" class="label">Descricao</td>
49                                  <td >
50                                          <mtw:inputText name="txt_descricao"
                                                id="txt_descricao" size="60"
                                                maxlength="100" klass="setFocus"
                                                />
51                                  </td>
52                          </tr>
53
54                          <tr>
55                                  <td colspan="2" class="barrabotao">
56                                          <input name="btn_pesquisar_dados"
                                                type="submit" value="Pesquisar"
                                                class="botao">
57                                          <input type="button" value="Incluir
                                                " onClick="window.location.href
                                                ='${linkCadastro}'" class="botao
                                                ">
58                                  </td>
59                          </tr>
```

```
60                  </table>
61 </form>
62
63 <c:if test="${listaFundoColecao != null}">
64
65 <display:table name="listaFundoColecao" id="fundoColecao" export="true"
       requestURI="${linkPesquisar}" class="listagem">
66         <display:column property="id" title="ID" class="img" headerClass="
               img"/>
67         <display:column property="descricao" title="Descricao" href="${
               linkCadastro}" paramId="txt_id" paramProperty="id"/>
68         <display:column property="instituicaoAcumuladora.descricao" title="
               Institui  Acumuladora" href="${linkCadastro}" paramId="txt_id"
                paramProperty="id"/>
69         <display:column title="Alterar" class="img" headerClass="img" href=
               "${linkCadastro}" paramId="txt_id" paramProperty="id"><skin:icon
                imagem="SALVAR" alt="Alterar" /></display:column>
70         <display:column title="Excluir" class="img noprint" headerClass="
               img nopr\int"><a href="javascript:excluir(${fundoColecao.id},'${
               fundoColecao.descricao}')"><skin:imgDelete/></a></display:column
               >
71 </display:table>
72
73 </c:if>
74
75 </center>
76 </body>
77 </html>
```

Nota-se que, na linha 65, há o uso de um *Display Table* pertencente à *Display Tag Library* para exibir os resultados encontrados nas buscas efetuadas.