

**Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
Curso de Ciências da Computação**

**Projeto e Implantação da Reformulação do Portal
Museu da Pessoa**

RAFAEL DE SOUZA COELHO

Florianópolis – SC
Ano 2007 / 2

RAFAEL DE SOUZA COELHO

**Projeto e Implantação da Reformulação do Portal
Museu da Pessoa**

Trabalho de conclusão de curso apresentado como parte dos requisitos para
obtenção do grau de Bacharel em Ciências da Computação

Orientador(a): Profa. Maria Marta Leite

Banca Examinadora
Elizabeth Sueli Especialski
João Candido Dovicchi

À minha família...

AGRADECIMENTOS

Agradeço à minha família por ter me dado todas as condições necessárias para completar meus estudos e possibilitar meu crescimento profissional. À minha namorada pelo carinho e dedicação nos momentos difíceis durante esse período.

Um agradecimento especial à empresa JExperts pela oportunidade oferecida para participar do projeto, pela confiança e por acreditar no meu trabalho. Aos integrantes da equipe por todo o esforço e eficiência para realizar o projeto com sucesso.

Agradeço também à professora Maria Marte Leite pela paciência e atenção ao orientar o meu trabalho. Aos membros da banca e a todos os envolvidos no projeto.

Resumo

Esse trabalho tem como objetivo desenvolver um sistema Web para o Instituto Museu da Pessoa. O projeto é uma reformulação do portal já existente para atender às necessidades do novo contexto de arquitetura de conteúdo. Este portal tem por finalidade ser um espaço aberto para que qualquer pessoa através da internet possa divulgar sua história de vida ou de terceiros. Essas histórias são relatadas por meio de depoimentos, podendo ser complementadas com imagens, áudios e vídeos, organizados de acordo com os métodos determinados pelo Museu. A área administrativa do sistema gerencia todo o acervo histórico e o expõe na internet para consulta. Para tanto, foi feito o levantamento das necessidades solicitadas visando após a análise dos requisitos, a implementação das melhorias e mudanças verificadas. Essa nova forma de registrar a história mostra-se inovadora no sentido de estabelecer uma rede mundial entre pessoas e o tempo.

Palavras-chave: Museu da Pessoa, portal, história de vida, pessoa

Abstract

The objective of this research is the development of a Web system for the Institute Museum of the Person. The project is a new formulation of the existent portal in order to meet the demands of the new architecture of contents. The main goal of the portal is to be an open environment where people can show the history of his or others life in the internet. Those histories are shown with testimonies, and can be complemented with images, sound and videos, that are organized in accordance with the methods determined by the Museum. The administrative area of the system manages the historical inventory and put it on the internet, open to the public information. A survey of all the necessities was made and after all the analysis, the implementation of enhancements and changes has been verified. This new way to record the history is an innovation in the sense that it establishes a global web between people and time.

Keywords: Museum of the People, portal, histories of his or other life, people.

Sumário

1	INTRODUÇÃO	11
2	OBJETIVOS	12
2.1	OBJETIVOS GERAL	12
2.2	OBJETIVOS ESPECÍFICOS	12
2.3	METODOLOGIA	13
2.4	A PESQUISA QUALITATIVA	14
2.5	HISTÓRIA DE VIDA	15
3	O INSTITUTO MUSEU DA PESSOA	16
3.1	MÉTODOS DE PESQUISA	16
3.2	PROJETOS REALIZADOS	17
3.2.1	<i>programa conte sua história</i>	18
3.2.2	<i>programa memória institucional</i>	19
3.2.3	<i>programa de formação</i>	22
3.3	NÚCLEOS	23
4	O PORTAL	25
4.1	ARQUITETURA DE CONTEÚDO DO PORTAL – 2003	25
4.2	MOTIVOS PARA A REFORMA DO PORTAL	26
4.3	A NOVA ARQUITETURA DE CONTEÚDO	27
4.3.1	<i>indexação dos depoimentos por xml</i>	28
4.4	REFORMA NA ÁREA ADMINISTRATIVA	29
5	REVISÃO BIBLIOGRÁFICA DAS TECNOLOGIAS E FERRAMENTAS UTILIZADAS	31
5.1	MODEL-VIEW-CONTROLLER (MVC)	31
5.2	DEFINIÇÃO DE UM FRAMEWORK	32
5.3	SPRING	33
5.4	HIBERNATE	34
5.5	STRUTS	34
5.6	AJAX	35
5.7	CVS	35
5.8	JSP	36
5.9	LUCENE	37
5.10	TOMCAT	37
5.11	ANT	37
5.12	TAGLIBS	38
5.13	POSTGRESQL	38
6	O FRAMEWORK	39
7	DESENVOLVIMENTO	42
7.1	LEVANTAMENTO DE ANÁLISE DE REQUISITOS	42
7.2	ATORES DO SISTEMA	43
7.3	FUNCIONALIDADES	44
7.3.1	<i>módulo administrativo</i>	44
7.3.2	<i>módulo acervo</i>	46

7.3.3	<i>módulo de busca</i>	49
7.3.4	<i>módulo publicador</i>	49
7.3.5	<i>módulo internauta</i>	49
7.4	PROTÓTIPO	50
7.5	DIAGRAMA DE CLASSES	52
7.6	IMPLEMENTAÇÃO DAS FUNCIONALIDADES	54
7.6.1	<i>cadastro de perfis</i>	55
7.6.2	<i>cadastro de usuários</i>	57
7.6.3	<i>autenticação</i>	59
7.6.4	<i>cadastros auxiliares</i>	60
7.6.5	<i>cadastro de palavras-chave</i>	61
7.6.6	<i>xsd indexação de depoimentos</i>	62
7.6.7	<i>indexadores do acervo</i>	62
7.6.8	<i>cadastro de pessoas</i>	63
7.6.9	<i>cadastro de histórias</i>	65
7.6.10	<i>relacionamentos do depoente</i>	68
7.6.11	<i>cadastro de projetos</i>	69
7.6.12	<i>relacionamentos do projeto</i>	71
7.6.13	<i>equipe do projeto</i>	71
7.6.14	<i>campos específicos</i>	72
7.6.15	<i>glossário</i>	73
7.6.16	<i>templates do projeto</i>	73
7.6.17	<i>modelo padrão do acervo</i>	77
7.6.18	<i>cadastro do acervo</i>	77
7.6.19	<i>buscas avançadas</i>	79
7.6.20	<i>cadastro de templates do publicador</i>	79
7.6.21	<i>cadastro de páginas do publicador</i>	81
7.7	AJUSTES E CORREÇÕES DE BUGS	83
7.8	MIGRAÇÃO DO BANCO DE DADOS	83
7.9	MIGRAÇÃO DAS SALAS	85
7.10	ETAPA FINAL	90
8	PROJETOS FUTUROS	91
9	CONSIDERAÇÕES FINAIS	92
10	REFERÊNCIAS	94
11	ANEXOS	97
11.1	CÓDIGO FONTE	97
11.1.1	<i>genérico.jar</i>	97
11.1.2	<i>guarda.jar</i>	105
11.1.3	<i>bean</i>	114
11.1.4	<i>business</i>	117
11.1.5	<i>dao</i>	118
11.1.6	<i>tags</i>	120
11.1.7	<i>xml</i>	130
11.1.8	<i>lucene</i>	138
11.1.9	<i>xsd</i>	144
12	ARTIGO	148

Índice de Figuras

Figura 1 – Ordem hierárquica do acervo do portal de 2003	25
Figura 2 – Exemplo de Projeto	26
Figura 3 – Diagrama MVC	32
Figura 4 – Diagrama de fluxo de inserção de uma história	47
Figura 5 – Exemplo da lista do protótipo	51
Figura 6 – Exemplo de cadastro do protótipo	51
Figura 7 – Diagrama de classes do módulo publicador	53
Figura 8 – Exemplo do menu em XML	56
Figura 9 – Cadastro de perfil	57
Figura 10 – Diagrama de classes do usuário	58
Figura 11 – Lista de usuários	59
Figura 12 – Tela de login da área administrativa	59
Figura 13 – Exemplo de cadastro auxiliar	61
Figura 14 – Resultado do exemplo do cadastro auxiliar	61
Figura 15 – Ficha de cadastro de pessoa	64
Figura 16 – Lista de pessoas	65
Figura 17 – Ficha de cadastro de história	67
Figura 18 – Relacionamentos do depoente	68
Figura 19 – Tela para associar projetos ao depoente	69
Figura 20 – Ficha de cadastro do projeto	70
Figura 21 – Lista de projetos	71
Figura 22 – Diagrama de classes do campo específico	72
Figura 23 – Campos específicos	73
Figura 24 – Exemplo de um template	76
Figura 25 – Cadastro de template do projeto	77
Figura 26 – Lista do acervo	78
Figura 27 – Busca avançada por pessoa	79
Figura 28 – Lista de chamadas do template	80
Figura 29 – Código HTML do template	81
Figura 30 – Cadastro de página	82
Figura 31 – Preview da página	82
Figura 32 – Lista de pessoas da Home Geral	89
Figura 33 – Página da pessoa da Home Geral	90

Lista de Abreviaturas e Siglas

CD-ROM – Compact Disk Read Only Memory
EJB – Enterprise JavaBeans
XML – Extensible Markup Language
HTML – HyperText Markup Language
MVC – Model-view-controller
JSP – JavaServer Pages
SQL – Structured Query Language
CVS – Concurrent Version System
API –Application Programming Interface
DHTML – Dynamic HTML
UML – Unified Modeling Language
SGBD – Sistema Gerenciador de Banco de Dados
DAO – Data Access Object
JDK – Java Development Kit
JAR – Java Archive
JUDE – Java and UML Developers' Environment
SESC – Serviço Social do Comércio
UFSC – Universidade Federal de SantaCatarina
AJAX – Asynchronous Javascript And XML
XSD – XML Schema Definition
DWR – Direct Web Remoting
CMP – Container-Managed Persistence

1 INTRODUÇÃO

A história é uma peça fundamental em todo o tipo de cultura social. Através dela podemos conhecer a evolução das tradições e valores intelectuais, morais e espirituais da humanidade ao longo de seu passado e presente. A cultura de uma civilização é preservada ao longo dos anos pela perpetuação de sua história. O homem deixa marcas ao passar pela vida, uma foto, um desenho, uma canção, todos são traços de sua passagem.

A história de vida de cada pessoa tem seu valor, cada indivíduo contribui para a construção da história de sua sociedade. A partir desses princípios, no início de 2003, o Instituto do Museu da Pessoa trouxe à empresa JExperts Tecnologia a missão de criar um sistema integrado de interface única para o armazenamento e gerenciamento de documentos históricos relacionados à história de vida de diversas pessoas do mundo. Esse acervo inclui: imagens, áudios, vídeos, documentos e histórias de pessoas, organizadas segundo as metodologias do Museu.

Após a implantação do sistema, surgiram novos conceitos para a organização desses dados e necessidades não previstas na concepção inicial do projeto. Sob essas circunstâncias, surgiram problemas que acabaram tornando essa estrutura inadequada. Com isso, no segundo semestre de 2006, o Museu da Pessoa retornou à JExperts com o intuito de reformar o portal já existente. Porém, como haveria um grande esforço para reestruturar o banco de dados e as regras do sistema atual tornou-se necessária uma reforma tecnológica, que resultou em um novo sistema.

A equipe inicial do projeto continha quatro integrantes da empresa JExperts: um gerente de projeto e três desenvolvedores. Com a saída de um dos integrantes no início de 2007, o projeto finalizou com dois programadores. Sendo o autor desse trabalho um dos desenvolvedores do sistema ativo desde o início do mesmo.

2 OBJETIVOS

2.1 OBJETIVOS GERAL

Desenvolver um sistema Web, utilizando a linguagem de programação Java, que atenda às necessidades solicitadas pelo Instituto Museu da Pessoa. Esse sistema visa fornecer meios para o registro e consulta de histórias de vida através da internet.

2.2 OBJETIVOS ESPECÍFICOS

- Buscar as últimas tecnologias disponíveis referente à linguagem de programação Java para Web.
- Utilizar somente softwares livres, ou seja, programas de computador que podem ser usados, copiados, estudados, modificados e redistribuídos sem restrições.
- Disponibilizar todo o código fonte do sistema para o Museu da Pessoa.
- Desenvolver o sistema de modo que diminua a dependência do Museu da Pessoa com a JExperts na criação e manutenção de seus projetos.
- Desenvolver o novo contexto para a estrutura de conteúdo dos projetos.
- Indexar os depoimentos através de tags XML.
- Melhorar a ergonomia site para uma melhor navegação dos usuários.

2.3 METODOLOGIA

Foi definido que seria aplicada para a especificação e modelagem do sistema a linguagem visual UML (Unified Modeling Language ou Linguagem de Modelagem Unificada), adotada internacionalmente pela indústria de Engenharia de Software e utilizada para modelar sistemas computacionais por meio do paradigma de Orientação a Objetos (GUEDES, 2004). Este mesmo autor (2004) afirma que:

“A UML não é uma linguagem de programação e sim uma linguagem de modelagem, cujo objetivo é auxiliar os engenheiros de software a definir as características do software, tais como seus requisitos, seu comportamento, sua estrutura lógica, a dinâmica de seus processos e até mesmo suas necessidades físicas em relação ao equipamento sobre o qual o sistema deverá ser implantado”.

Após uma série de conversas com o cliente e análise de todos os documentos disponibilizados para entender os requisitos que deverão ser implementados no novo sistema, foi definido que seria desenvolvido primeiramente um wireframe do projeto. A função do wireframe é estruturar o conteúdo de cada página, indicando o peso e relevância de cada elemento do layout e sua relação com os demais elementos. Ele também indica possibilidades de interação e o caminho percorrido pelo usuário.

Com a aprovação do wireframe pelo cliente, planejou-se criar um protótipo do sistema. O protótipo simula, da maneira mais fiel possível, o sistema a ser desenvolvido, desde a sua interface a usabilidade. Após a aprovação é possível iniciar a construção do diagrama de classes, que descreve os vários tipos de objetos no sistema e os seus relacionamentos.

O passo seguinte, com o diagrama de classes definido, será a implementação do projeto, utilizando um framework desenvolvido para ele. Após a implementação, começará a fase de ajustes e correções de erros.

A etapa final do projeto ocorre com a implantação do sistema em alguns servidores para testes e a migração dos dados e projetos do atual portal. Por último, será realizado o desenvolvimento da parte escrita do projeto.

2.4 A PESQUISA QUALITATIVA

A pesquisa qualitativa é mais comum nas áreas de Ciências Humanas como Psicologia, Sociologia e Comunicação Social. Ao contrário da quantitativa, que procura entender um fenômeno através de estatísticas, generalizações e regras, a qualitativa busca a compreensão de comportamentos e estados subjetivos, utilizando comparações e interpretações (MINAYO E SANCHES, 1983).

Para medir a variabilidade do comportamento e dos estados subjetivos – pensamentos, atitudes, sentimentos – o professor e psicólogo Harmut Günther (2006) cita Kish (1987):

“Sob a ótica das ciências sociais empíricas existem três aproximações para compreender o comportamento e os estados subjetivos: a) observar o comportamento que ocorre naturalmente no âmbito geral; b) criar situações artificiais e observar o comportamento diante das tarefas definidas para essas situações; c) perguntar às pessoas sobre o seu comportamento, o que fazem e fizeram e sobre seus estados subjetivos, o que, por exemplo, pensam e pensaram.”

A pesquisa qualitativa tem por objetivo traduzir e expressar o sentido dos fenômenos do mundo social; trata-se de reduzir a distância entre indicador e indicado, entre teoria e dados, entre contexto e ação (MAANEM, 1979, p.520). Ela é realizada por meio de métodos de coleta de dados, como entrevistas e relatos. A pesquisa é adaptada ao caso específico, não existe um padrão único de método.

2.5 HISTÓRIA DE VIDA

A história de vida é uma ferramenta da pesquisa qualitativa, ela inclui depoimentos, entrevistas, biografias, autobiografias. Com ela é possível analisar a intersecção entre a vida individual de uma pessoa e o contexto social. Maria Ângela Silveira Paulino (2007), Doutora em Serviço Social pela PUC-SP, considera que:

“A história de vida pode ser, desta forma, um instrumento privilegiado para análise e interpretação, na medida em que incorpora experiências subjetivas mescladas a contextos sociais. Ela fornece, portanto, base consistente para o entendimento do componente histórico dos fenômenos individuais, assim como para a compreensão do componente individual dos fenômenos históricos.”

Ela possibilita aprender a cultura de forma aprofundada, é um instrumento valioso pois se coloca entre as relações externas e internas do indivíduo. São sempre relatos de práticas sociais: das formas com que o sujeito se insere e atua no mundo e no grupo do qual ele faz parte.

Existem diversas maneiras para realizar o estudo de vida de uma pessoa. As entrevistas são os meios mais comuns para coletar essas informações, o entrevistador respeita a opinião do indivíduo e acredita no que ele diz. Normalmente o entrevistado relata apenas aquilo que considera mais importante e pode descartar momentos que ache desnecessários. Com isso, caso seja necessário, pode-se abranger os métodos de coleta de dados e incluir qualquer tipo de documento que possa colaborar no desenvolvimento da pesquisa, como: fotos, desenhos, vídeos, áudios, textos.

3 O INSTITUTO MUSEU DA PESSOA

O Instituto Museu da Pessoa foi fundado em 1991 em São Paulo. Seu objetivo é de construir uma rede internacional de histórias de vida capaz de contribuir para a mudança social. Eles acreditam que a memória social pode ajudar a criar diferentes perspectivas da nossa sociedade (MUSEU DA PESSOA, 2007). As histórias de vida são informações únicas, com elas é possível conhecer as diferentes realidades que cada pessoa cria em sua vida.

O Museu da Pessoa tem o objetivo de registrar, preservar e transformar em informação histórias de vida de toda e qualquer pessoa da sociedade, seja ela célebre ou anônima. “Promovendo mudanças sociais por meio da reflexão sobre a identidade e valorização de indivíduos e comunidades” (ASHOKA EMPREENDEDORES SOCIAIS, 2007).

Mesmo não existindo internet na época, o Museu se define desde o início como um museu virtual, pois todos os seus dados são armazenados digitalmente. Seu acervo é constituído de fotos, imagens depoimentos, áudios e vídeos.

3.1 MÉTODOS DE PESQUISA

O museu desenvolveu sua própria metodologia para registrar os depoimentos e formar um acervo inédito de histórias de vida. Iniciou seu trabalho em 1991 como prestador de serviços, realizando projetos de memória em empresas, sindicatos, associações, escolas e comunidades. Procurou sempre capturar diferentes vozes, buscando pessoas que poderiam dar diferentes perspectivas sobre a história.

Com o programa museu itinerante, foram espalhadas diversas cabines de gravação em locais públicos como praças, metrô e ruas da cidade. Assim qualquer pessoa com acesso a essas cabines tinha a chance de contar sua história. Com isso puderam organizar todas essas narrativas em diversos CD-ROMS e colocá-las em jukeboxes para serem escutadas por todos.

Em 1997 - com o avanço da internet no Brasil - foi lançado o primeiro site do Museu da Pessoa, iniciando assim a captação de histórias pela internet, além de disponibilizar ao público parte de seu acervo. Porém, somente uma pequena parte da população do Brasil tem acesso à internet, então, o Museu resolveu disseminar suas metodologias e práticas de pesquisas com alguns programas como os “Agentes da História” e “Memória Local”. O primeiro tem o objetivo de treinar idosos para entrevistar outros idosos e o segundo programa é focado na alfabetização e inclusão digital nas escolas públicas no Brasil e treinamento de multiplicadores em comunidades e organizações da sociedade civil (MUSEU DA PESSOA, 2007).

Com todos os dados coletados durante os anos de existência do Museu, surgiu a necessidade de utilizar essas informações para promover uma mudança social entre as pessoas que contaram suas histórias. Para isso, foi desenvolvido em 2003, pela JExperts Tecnologia, o primeiro Portal Museu da Pessoa.

O Portal tem o objetivo de conectar as pessoas que narram as histórias com as pessoas que as escutam, criando uma rede mundial de histórias de vida. “Essas conexões definitivamente ajudariam a estabelecer novos parâmetros educacionais para entender a História. Então, utilizar o poder das histórias de vida para conectar gerações, comunidades e diferentes níveis de poder na sociedade é um passo essencial” (MUSEU DA PESSOA, 2007), além de disponibilizar todo o seu acervo pela internet.

3.2 PROJETOS REALIZADOS

O Museu da Pessoa já desenvolveu mais de 50 projetos em todo o Brasil, com diferentes empresas, sindicatos, associações, comunidades e escolas. Atua por meio de pesquisa histórica, ações de formação e, especialmente, registro de histórias de vida. Transforma a memória em estratégia de valorização das pessoas, fortalecimento institucional, desenvolvimento comunitário e pedagógico.

3.2.1 PROGRAMA CONTE SUA HISTÓRIA

“Iniciado em 2004, o Programa Conte Sua História é responsável pela política de captação de acervo do Museu da Pessoa nos diferentes meios, ampliando e qualificando a coleta de histórias e integrando-as à publicação no portal” (MUSEU DA PESSOA, 2007). O objetivo desse programa é permitir que qualquer pessoa possa contar sua história, através de campanhas de captação de histórias em locais públicos, em estúdio, pela internet e pelas cabines itinerantes. Além disso, se responsabiliza pela divulgação de todo o acervo coletado. A seguir a lista dos projetos realizados por esse programa:

- Ponto de Cultura Museu da Pessoa – “espaço aberto à comunidade que dispõe de um estúdio de gravação para o registro das histórias de vida, sala de consulta ao acervo com computadores, além de biblioteca e videoteca” (MUSEU DA PESSOA, 2007). Em parceria com o Programa Cultura Viva do Ministério da Cultura, 2006 – 2008.
- Um Milhão de Histórias de Jovens – “tem como objetivo valorizar a expressão da juventude, especialmente no nordeste, por meio da articulação de jovens e da divulgação de suas histórias de vida” (MUSEU DA PESSOA, 2007). Realização em parceria com a Aracati - Agência de Mobilização Social, 2006.
- Catadores de Material Reciclável – em dezembro de 2004, o Movimento Nacional de Catadores, a Fundação Avina e o Museu da Pessoa se reuniram para registrar a história dos catadores de materiais recicláveis com o objetivo de valorizar a profissão e o seu papel na sociedade (MUSEU DA PESSOA, 2007).
- Vou te Contar - Maranhão (2005)
- Vou te Contar – Rio Grande Sul (2004)
- São Paulo, Meu! (2004)

- O Mundo Cabe Numa Cadeira de Barbeiro (Synapse e Superfilmes, 2001/2002)

3.2.2 PROGRAMA MEMÓRIA INSTITUCIONAL

O programa tem por finalidade preservar e divulgar a memória das instituições brasileiras. “Atua para que o conceito de memória coletiva seja compreendido e utilizado para ampliar reflexões, perceber formas diferentes de articular e vivenciar a realidade” (MUSEU DA PESSOA, 2007). Os recursos gerados são direcionados à atividades em organizações sociais, escolas e comunidades. São projetos realizados por esse programa:

- Memória dos Trabalhadores Petrobras (Petrobras, 2007) - exposição virtual com depoimentos de trabalhadores da Petrobras em 27 países.
- Projeto 50 anos da Fundação Bradesco (Fundação Bradesco, 2006) - pesquisa, memória oral, gravação de 60 depoimentos e publicação de livro.
- Memória da ABEVD (Associação Brasileira de Venda Direta, 2006)
- Memória da Convenção da Diversidade Biológica e da Convenção Sobre Mudança do Clima e Protocolo de Kyoto (Em parceria com a Fábio Feldman Consultores, 2006)
- Projeto Fundação Banco do Brasil (Fundação Banco do Brasil, 2006)
- Projeto Museu Nacional da Indústria Têxtil (Companhia Industrial Cataguases e Instituto Francisca Souza Peixoto, 2006)
- Projeto Memória Redecard (Redecard, 2006)
- Memória da Educação em São Paulo – Escola Pueri Domus – 40 anos (Escola Pueri Domus, 2006)
- Memória do Dieese – 50 anos (Departamento Intersindical de Estatística e Estudos Socioeconômicos, 2006)

- Memória da Unimed Brasil – 40 anos (Unimed Brasil, 2006)
- Unimed-Rio (Unimed-Rio, 2004-2006)
- Memória profissional Nizan Guanaes (Agência África, 2005)
- Memória Oral dos Pacientes (Doutores da Alegria, 2004)
- Museu Clube da Esquina (Incentivado pelo Ministério da Cultura com o Patrocínio da Petrobras, 2004/2005)
- Memória Pão de Açúcar (Grupo Pão de Açúcar, 2003-2004)
- Memória das Comunidades Natura (Natura, 2003-2006)
- Memória Aracruz (Aracruz, 2003-2004)
- Memória dos Trabalhadores da Petrobras (Petrobras/Sindipetro, 2002-2006)
- Samarco 25 anos (Samarco Mineração S.A., 2002-2004)
- Centro de Referência Ambev (Companhia de Bebidas das Américas, 2001-2006)
- Memórias do Comércio na Cidade do Rio de Janeiro (SESC/RJ, 2002-2003)
- Memória Votorantim (Grupo Votorantim, 2002-2006)
- CTBC, Companhia Telefônica do Brasil Central – 50 anos (CTBC, 1999/2003/2004)
- Vale Memória (Companhia Vale do Rio Doce, 1999-2003)
- BNDES 50 anos (Banco Nacional de Desenvolvimento Econômico e Social, 2002)
- Memorial do Trabalhador (Ecomuseu/Itaipu Binacional, 2002)
- Museu do Flamengo (Clube de Regatas Flamengo, 2000-2001)
- Laboratório Aché - 35 anos: História e Responsabilidade Social (Laboratório Aché, 2001-2002)
- Memórias do Comércio (SESC/SP, 1994-2004)
- Imigrantes: Crônicas de Vida (Shopping West Plaza, 2001)

- Memória nos Bairros – São Paulo nos Trilhos do Tempo: Brás, Liberdade e Itaquera (Metrô de São Paulo/Eletropaulo, 2000)
- Espaço Abril (Editora Abril, 1999)
- Museu da Agricultura Brasileira (Embrapa, 2000)
- Memória Globo (Organizações Globo, 1996-2000)
- História em Multimídia do São Paulo Futebol Clube (São Paulo Futebol Clube, 1994-1999)
- Serviço de Psicologia - Instituto do Coração - 25 Anos de História (InCor, 1999)
- Museu do Santos Futebol Clube (Santos Futebol Clube, 1998-1999)
- Memória da Rhodia Farma: 80 anos de Brasil (Rhodia Farma, 1998-1999)
- ABC de Luta - Memória dos Trabalhadores (Sindicato dos Metalúrgicos do ABC, 1998-1999)
- Odebrecht (Construtora Norberto Odebrecht, 1998-1999)
- História das Profissões em Extinção (Confederação Nacional dos Metalúrgicos de São Paulo e Companhia Brasileira de Mineração e Metalurgia, 1996-1999)
- Abifarma 50 anos: Indústria Farmacêutica e Cidadania (Associação Brasileira da Indústria Farmacêutica, 1997)
- História em Multimídia do SENAC-SP (SENAC-SP, 1995-1996)
- Farma: Uma Constante Construção – História de Janssen-Cilag Brasil (Janssen-Cilag Brasil, divisão farmacêutica da Johnson & Johnson, 1995-1996)
- Memória & Migração (Metal Leve, Banco Safra e Casa de Cultura de Israel, 1991)

3.2.3 PROGRAMA DE FORMAÇÃO

O objetivo desse programa é preparar as pessoas para desenvolverem os projetos do Museu da Pessoa. Capacitando educadores e mediadores de diferentes instituições para preservarem e divulgarem a memória da sociedade, contribuindo para uma redução da desigualdade social (MUSEU DA PESSOA, 2007).

São promovidos encontros de curta duração para que as pessoas possam conhecer os conceitos básicos sobre memória e história. Também são realizados projetos de longa duração, para aprofundar a formação de profissionais que possam colaborar para o registro de histórias. Além do Memória Local, que é uma ação junto às escolas públicas do ensino fundamental de diferentes estados do país. São projetos relacionados ao Programa de Formação:

- Projeto Memória Local na Escola (Votorantim, 2006) – “promove a inserção do tema memória oral no cotidiano escolar, a partir da formação contínua de professores e alunos” (MUSEU DA PESSOA, 2007).
- Projeto Histórias da Nossa Terra (CTBC, Instituto Algar e Hedging Griffo, desde 2001) – “é um projeto que resgata a história de municípios através da técnica de memória oral, refletindo a lembrança dos moradores sobre a própria cidade, suas famílias, infância, trabalho, credo, tradições e origens” (MUSEU DA PESSOA, 2007).
- Trilheiros, uma aventura em Minas (Fundação Vale do Rio Doce e Canal Futura, 2006)
- Encontro de Fellows no Rio de Janeiro (Ashoka Empreendedores Sociais, 2006)
- Projeto Memória da Humanização da Saúde (Ministério da Saúde, 2006)
- Oficina de Memória Cipó (Cipó Comunicação Interativa, 2006)
- Celebrações (Aberje, Associação Brasileira de Comunicação Empresarial, 2005-2006)

- Programa Formação de Multiplicadores (Votorantim, 2005)
- Projeto Memória Votorantim (Votorantim, 2004)
- Realizações - Memória Local (em parceria com o Instituto Avisa Lá desde 2001)
- Projeto Pão de Açúcar faz História (Instituto Pão de Açúcar, 2001-2004)
- Projeto Fundo Desafio (em parceria com a Ashoka Empreendedores Sociais do Brasil, 2004-2005)
- Programa de Formação de Representantes do Projeto Memória Petrobras (em parceria com a Universidade Corporativa Petrobras, 2005)
- Programa de Formação de Jovens - Heliópolis dos Sonhos (em parceria com Aracati, Agência de Mobilização Social, e Instituto Esporte e Educação. Patrocínio: Instituto Credicard, 2003-2005)

3.3 NÚCLEOS

Além do Instituto Museu da Pessoa sediado em São Paulo, o Museu da Pessoa hoje é formado por mais quatro núcleos localizados em diferentes países. Foi em 1999, na conferência internacional Museums and Web, nos Estados Unidos, o momento em que o Museu começou a ampliar suas fronteiras. A partir dessa conferência foi criado o Núcleo Museu da Pessoa de Portugal.

Em 2001 em uma nova edição da conferência, o Museu buscou expandir seu trabalho. Durante a conferência surgiu a idéia de criar um novo núcleo – Museum of the Person, Indiana - ligado à Universidade de Indiana nos Estados Unidos. A convite do Colégio Loyola, o Museu da Pessoa inaugurou uma Unidade de Memória na capital mineira, em 2003. Outro núcleo foi fundado no ano de 2004

em parceria com o Centro de Histórias de Montreal no Canadá, chamado Musée de La Personne – Montreal.

A seguir os endereços na internet de alguns núcleos:

- Núcleo Português do Museu da Pessoa – Braga, Portugal (<http://www.museu-da-pessoa.net>).
- Núcleo Museum of the Person – Indiana, Estados Unidos (<http://www.bloomington.us/~mop-j>).
- Núcleo Musée de la Personne – Montreal, Canadá (<http://www.museedelapersonne.ca>).

4 O PORTAL

O Portal hoje é fundamental para o Museu da Pessoa. O sistema desenvolvido em 2003 permite armazenar todo o acervo coletado onde, através de uma interface Web, os pesquisadores e coordenadores envolvidos nos projetos possam inserir e editar imagens, depoimentos, áudios, vídeos, etc. Todos esses dados são organizados e classificados de acordo com a metodologia definida pelo Museu.

4.1 ARQUITETURA DE CONTEÚDO DO PORTAL – 2003

A arquitetura de conteúdo planejada para o gerenciamento do acervo do primeiro portal desenvolvido em 2003 seguia algumas regras de classificação e relacionamento entre os diferentes objetos do acervo. O conteúdo do acervo segue uma ordem hierárquica, conforme pode ser visualizado na figura 1 a seguir.

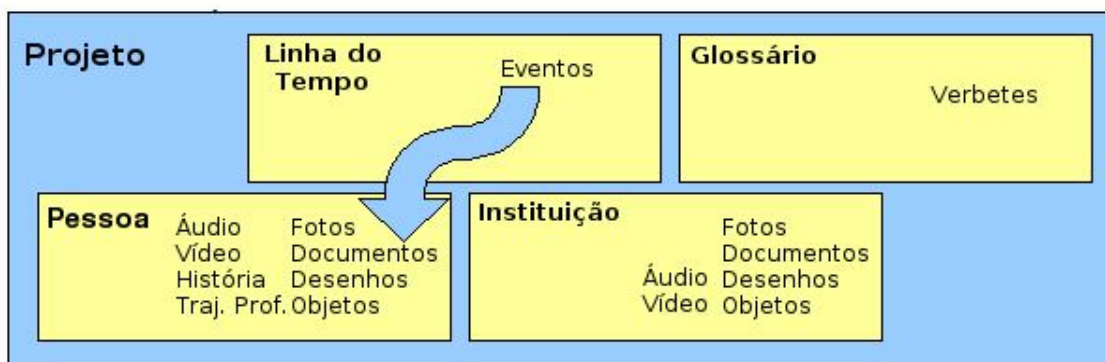


Figura 1 – Ordem hierárquica do acervo do portal de 2003
Fonte: (CARDOSO, 2006)

A hierarquia adotada para a classificação do conteúdo do acervo tem como principal o projeto, que é a célula matriz, onde todo o acervo está inserido, de acordo com a figura 2.

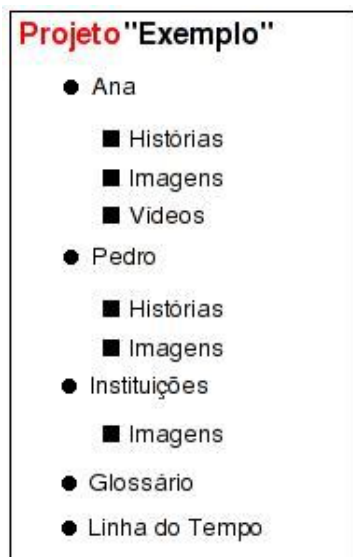


Figura 2 – Exemplo de Projeto
Fonte: (CARDOSO, 2006)

4.2 MOTIVOS PARA A REFORMA DO PORTAL

Após alguns anos utilizando o portal, a equipe do Museu da Pessoa constatou que existiam alguns problemas na estrutura de conteúdo dos projetos, tornando-a inadequada. Dentre elas podem ser citadas:

- Uma pessoa ou instituição não podia se relacionar com projetos diferentes;
- Não podia existir uma pessoa sem relacionamento à um projeto;
- Uma pessoa cadastrada em um projeto por um pesquisador não podia se tornar um internauta, podendo assim editar sua própria história.

Outras necessidades não previstas na concepção inicial surgiram:

- Permitir relacionar diferentes histórias de vida;
- Existir a possibilidade do internauta poder comentar, traduzir ou colaborar com uma história;
- Possibilitar um usuário da área administrativa a acessar funcionalidades do perfil internauta, entre outras.

Também foi necessário uma melhoria nos aspectos ergonômicos do portal, melhorando a usabilidade do sistema e reduzindo os caminhos para execução de ações administrativas. Com isso, um novo contexto foi idealizado com o objetivo de eliminar os problemas existentes e implantar as novas necessidades ao sistema. Assim surgiu o projeto para reformular o portal.

Como a reestruturação das regras do sistema e do banco de dados seria muito trabalhosa, mantendo a tecnologia baseada em EJB-CMP e Struts, foi necessária uma reforma tecnológica. Essa reforma proporcionou mais flexibilidade ao sistema, permitindo assim ao Museu mais autonomia para desenvolver seus projetos virtuais.

A estrutura atual é muito dependente da JExperts, mesmo para pequenas mudanças, o que gera atraso nos projetos. Como o Museu é uma entidade sem fins lucrativos, fica inviável a contratação de serviços terceirizados.

4.3 A NOVA ARQUITETURA DE CONTEÚDO

A análise dos novos requisitos trouxe muitas mudanças para o sistema. O projeto deixa de ser a célula matriz de referência ao acervo e a pessoa se torna o foco principal de todo conteúdo. Os resultados das mudanças são:

- Todo o acervo relacionado à pessoa não tem relação com o projeto, somente com ela mesma;
- Uma pessoa pode ser relacionada à outra pessoa;
- Uma pessoa pode ter uma conta para acesso ao sistema com perfil de internauta que lhe permite desempenhar algumas funcionalidades no portal;
- Possibilidade dos internautas de criar e coordenar comunidades semelhantes ao conceito do Orkut;
- Permitir o relacionamento entre uma pessoa com nenhum ou vários projetos.

4.3.1 INDEXAÇÃO DOS DEPOIMENTOS POR XML

Através de uma linguagem especial de marcação em XML, configurável pela área administrativa do portal, pode-se classificar e contextualizar os depoimentos inseridos no sistema. Essa funcionalidade permite delimitar temas, citações, palavras-chaves de acordo com a metodologia do Museu.

Com o uso do XML nos depoimentos é possível criar metadados - dados capazes de descrever outros dados – sobre o texto, permitindo definir marcações de acordo com um vocabulário controlado, determinado por um esquema em XML. Cada projeto tem um arquivo de configuração com seu vocabulário controlado e marcações específicas. Existe um arquivo de configuração geral para o sistema, com definições essenciais para todos os projetos.

Depois do depoimento ser cadastrado com as marcações, ele pode ser visualizado através de um XML Stylesheet – folha de estilos – que transforma o conteúdo do documento XML em um formato de saída HTML. A indexação do conteúdo do depoimento estruturado por metadados a partir das marcações XML é realizado pelo sistema por uma ferramenta escrita em Java chamada Lucene. O Lucene é um recurso que oferece dois tipos principais de serviços: indexação e pesquisa de texto. Ele tem habilidade para criar e armazenar informações em um

índice. Com ele é possível realizar buscas temáticas baseadas no vocabulário controlado do projeto e também transforma o conteúdo do documento em um formato de saída HTML, formatando os títulos e armazenando o depoimento já processado no banco de dados.

4.4 REFORMA NA ÁREA ADMINISTRATIVA

A área administrativa do portal foi reformulada para melhorar sua ergonomia e implementar novas funcionalidades. Essa reformulação dá ao Museu da Pessoa mais autonomia para criação de novos projetos e melhora também a navegabilidade do sistema. A seguir os principais requisitos implementados da área administrativa:

- Um menu central e hierárquico que permite o acesso rápido e direto à todas as funcionalidades do sistema.
- Possibilidade de criar perfis e configura-los de acordo com as permissões de acesso desejadas.
- Permite associar os usuários com os perfis cadastrados no sistema e também personalizar as permissões de cada usuário.
- Os usuários cadastrados só poderão ser removidos por um usuário com perfil de administrador.
- Permite bloquear e desbloquear o acesso ao sistema de qualquer usuário;
- Todos os campos de seleção dos formulários são configuráveis, suas opções são cadastradas pela interface administrativa.
- Existem ferramentas para criar templates que permite a visualização e consulta pública do acervo do portal, esses templates podem facilmente ser alterados e customizados.

- Permite enviar um arquivo compactado em formato ZIP contendo a estrutura de um template para associar à um projeto.
- Todo conteúdo do acervo pode ser moderado com três diferentes estados: aprovado, reprovado e pendente.
- Permite a configuração de campos específicos para os formulários de conteúdo de cada projeto (ficha de pessoa e as fichas dos acervos relacionados), além dos campos já previstos nos formulários padrão.
- A cessão de direitos é feita somente na ficha da pessoa estendendo-se pra todo o conteúdo referente a ela.
- Todas as listas de conteúdo da área administrativa contém opções para busca com filtros.
- Existe uma seção de busca avançada que permite filtrar o conteúdo com mais precisão.
- Os formulários são preenchidos com um único passo.

5 REVISÃO BIBLIOGRÁFICA DAS TECNOLOGIAS E FERRAMENTAS UTILIZADAS

Um dos requisitos do projeto é que o código fonte do sistema seja liberado para a equipe do Museu da Pessoa para que ele possa ser estudado e, se necessário, alterado. Com essa premissa foi desenvolvido um framework composto somente de softwares livres, ou seja, programas de computador que podem ser usados, copiados, estudados, modificados e redistribuídos sem nenhuma restrição.

A linguagem de programação escolhida para desenvolver o portal foi Java, devido à sua portabilidade, segurança e seu vasto conjunto de bibliotecas. Isso permitiu arquitetar um framework com diversas tecnologias e conceitos em Java o que agilizou o andamento do projeto.

A seguir será revisada todas as ferramentas e tecnologias que compõem o framework.

5.1 MODEL-VIEW-CONTROLLER (MVC)

O Model-View-Controller ou MVC 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) (WIKIPÉDIA, 2007). O uso do MVC oferece muitos benefícios ao projeto de aplicações. Por separar os conceitos de apresentação, controle de fluxo e lógica de negócios, ele reduz a duplicação de código, centraliza o controle e torna a aplicação mais robusta, portátil e de fácil manutenção.

- View ou Visualização - constrói a interface do sistema com o usuário através de navegadores.
- Model ou Persistência de Informações - camada responsável pela persistência das informações em um banco de dados.

- Controller ou Lógica de Negócios - controla o processamento das ações requeridas pelo usuário, aplica segurança, validação de dados e aciona operações no banco de dados, retornando o conteúdo que será enviado ao navegador utilizado pelo cliente.

O MVC é usado em padrões de projeto de software, mas ele abrange mais da arquitetura de uma aplicação do que é típico para um padrão de projeto, é útil para aplicações grandes onde os mesmos dados são requeridos e manipulados de formas diferentes (WIKIPÉDIA, 2007). É bastante adequado para empresas que desenvolvem de maneira modular e concorrente com muitos desenvolvedores.

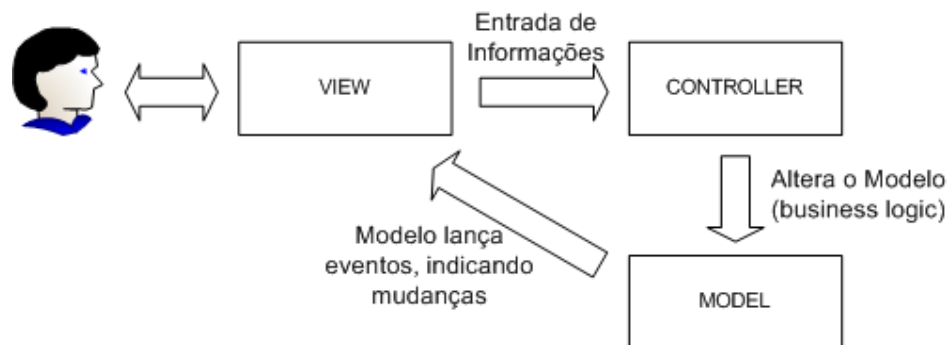


Figura 3 – Diagrama MVC
Fonte: (ALMEIDA, 2007)

5.2 DEFINIÇÃO DE UM FRAMEWORK

Segundo Heinz Züllighoven um framework é uma arquitetura de classes que oferece soluções genéricas para problemas similares em um contexto específico (ZÜLLIGHOVEN, 2005), incluindo programas de suporte, bibliotecas de códigos e até mesmo outros frameworks. Um framework fornece a estrutura base para uma

aplicação e seu fluxo de controle. Seu propósito é eliminar uma quantidade significativa de código que é essencial para o desenvolvimento de um sistema, implementando mecanismos e rotinas que são usados com frequência. Isso possibilita aos programadores gastarem menos tempo com a criação de códigos repetitivos de baixo nível e se concentrarem mais nas exigências do software.

5.3 SPRING

Spring é um framework de código aberto, ele foi criado para direcionar a complexidade de desenvolvimento de aplicações corporativas, com ele é possível utilizar o JavaBeans para alcançar objetivos que somente eram possíveis com o Enterprise JavaBeans (EJBs) (WALLS; BREIDENBACH, 2005). Qualquer aplicação em Java pode ser beneficiada em termos de simplicidade, testabilidade e acoplamento frouxo pelo uso do Spring .

Ele é um framework leve de inversão de controle e container orientado à aspecto.

- Leve – o Spring é leve em termos de tamanho e de processamento. O framework inteiro pode ser distribuído em um único arquivo JAR com menos de 1 megabyte.
- Inversão de Controle – o Spring promove um acoplamento frouxo através da técnica conhecida como “inversão de controle”. Quando ela é aplicada, os objetos dão passivamente suas dependências em vez de criar ou procurar por objetos dependentes deles mesmos. Ao invés do objeto procurar pelas suas dependências em um container, o container dá as dependências para o objeto na sua instanciação sem esperar ser chamado (WALLS; BREIDENBACH, 2005).
- Orientado à Aspecto – o Spring tem suporte para programação orientada à aspecto que permite um desenvolvimento coeso, separando a lógica de negócio dos serviços do sistema (WALLS; BREIDENBACH, 2005).

- Container – é considerado como container, pois contém e gerencia o ciclo de vida e a configuração dos objetos da aplicação (WALLS; BREIDENBACH, 2005).

As funcionalidades do Spring permitem escrever um código mais limpo, melhor de gerenciar e testar.

5.4 HIBERNATE

O Hibernate é um framework que automatiza a persistência com objetos Java para um banco de dados relacional. Ela transforma os dados tabulares de um banco de dados em um grafo de objetos definido pelo desenvolvedor, o problema entre a programação orientada a objetos e o banco de dados relacional é que necessita que se escreva o código que mapeia um para outro (BAUER; KING, 2005). Esse código é complexo e gasta muito tempo do programador, o Hibernate faz esse mapeamento para o usuário. Ele fica em uma camada entre o aplicativo e o banco de dados, carregando e salvando os objetos (BAUER; KING, 2005). Usando o Hibernate, o desenvolvedor se livra de escrever muito do código de acesso a banco de dados e de SQL que ele escreveria sem usar a ferramenta, acelerando a velocidade do seu desenvolvimento.

5.5 STRUTS

Struts é um framework de código aberto para desenvolver aplicações Java/Web. Ele implementa o controlador da aplicação, sendo este controlador responsável por toda a logística e também pela integração das camadas de visualização e regra de negócio, o Struts fornece o controlador e facilita a escrita dos templates para a camada de visualização ou de apresentação (geralmente em JSP) (HUSTED et al., 2002). O programador é responsável pela implementação do

modelo de código e a criação de um arquivo de configuração central chamado `struts-config.xml` o qual se liga junto à camada `model`, `view` e `controller`.

As requisições do cliente são enviadas ao controlador em forma de “Ações” definidas no arquivo de configuração. Se o controlador recebe alguma requisição ele chama a classe de ação correspondente o qual interage com o modelo de código específico da aplicação. O modelo de código retorna um `ActionForward`, que é uma string dizendo ao controlador qual página de saída será enviada ao cliente. A informação é passada pelo `model` e pelo `view` na forma especial de `JavaBeans`. Uma poderosa biblioteca permite ler e escrever o conteúdo desses beans para a camada de apresentação sem a necessidade de um código Java embutido.

5.6 AJAX

Utilizando uma combinação de tecnologias, o AJAX ou `Asynchronous JavaScript and XML` possibilita a criação de aplicações web mais interativas. Quando um formulário é preenchido ao apertar o botão para submeter os dados uma função JavaScript vai obter os valores das caixas de texto, validar os dados e chamar seu mecanismo AJAX, que é um encapsulamento para o método `XMLHttpRequest` (WIKIPÉDIA, 2007). O `XMLHttpRequest` - solicitações assíncronas de informações cliente/servidor sem necessidade de atualizar a página - envia os dados fornecidos em tempo real para a linguagem de programação escolhida - Java - do lado do servidor, que irá processá-los e retornará outros dados. Uma outra função em JavaScript insere os dados na página sem ter que atualizá-la (WIKIPÉDIA, 2007). Existem diversos frameworks de AJAX, o utilizado será o DWR.

5.7 CVS

O CVS – `Concurrent Versions System` – é uma ferramenta de código aberto para ajudar o desenvolvimento de aplicações, cuja principal função é controlar as modificações realizadas nos arquivos de um projeto (WIKIPÉDIA, 2007).

O CVS permite que sejam realizadas modificações paralelas de forma coerente e padronizada.

Os arquivos modificados são armazenados em um servidor, permitindo que os outros integrantes do projeto possam copiá-los pela rede, mantendo o sistema sempre atualizado (WIKIPÉDIA, 2007). Isso permite a uma equipe de programadores trabalharem em um projeto ao mesmo tempo de forma dinâmica e segura.

5.8 JSP

A tecnologia Java Server Page (JSP) utiliza o conceito de container, responsável por gerar o conteúdo dinâmico via protocolo http, é uma extensão da API Servlet, atuando na camada de apresentação, as páginas JSP utilizam a tecnologia Java no formato de servlets no lado do servidor para a criação do conteúdo dinâmico junto ao HTML para o conteúdo estático (BOND et al., 2003). Para executar as páginas JSP é utilizado o servidor Tomcat. O JSP possui as seguintes características:

- Separação do conteúdo estático do dinâmico: a parte lógica é encapsulada em componentes Java do tipo JavaBeans, que são utilizados pelas páginas JSP através de tags especiais, chamadas taglibs. A geração do conteúdo dinâmico é mantida separada das tags HTML responsáveis pela interface para o usuário (BOND et al., 2003).
- Independência de plataforma: como a tecnologia JSP é uma extensão da plataforma Java, ela não depende de uma plataforma específica para rodar;
- Diversos formatos: possibilidade de implementação em diversas linguagens (HTML, XML, DHTML, etc).
- Utilização de código Java: é possível utilizar códigos Java inseridos dentro das páginas JSP.

5.9 LUCENE

O Lucene é uma biblioteca de alta performance para recuperar informações em dados. Permite adicionar capacidades de buscas e indexações nas aplicações com um mínimo de conhecimento de indexação de textos e buscas mais complexas (GOSPODNETIC; HATCHER, 2005). Ele é uma ferramenta de código aberto implementada em Java.

5.10 TOMCAT

O Apache Tomcat é um servidor de aplicações Java para web (D'ÁVILA, 2007). Ele é um servlet container e é usado como referência para implementação de tecnologias Java Servlet e JavaServer Pages (JSP). Seu desenvolvimento é aberto e permite a participação e colaboração de programadores de todo o globo (THE APACHE SOFTWARE FOUNDATION, 2007). O Tomcat é um servidor robusto, seguro e eficiente mesmo para aplicações de grande porte.

5.11 ANT

O Apache Ant é uma ferramenta Java que serve para automatizar a construção de software. Ele é parecido com o Make – programa para compilar automaticamente o código fonte de um programa – porém é escrito em Java e foi inicialmente desenvolvido para ser utilizado em projetos desta linguagem (WIKIPÉDIA, 2007). O Ant é estendido usando classes Java e é multiplataforma. Ao invés de escrever comandos shell, seus arquivos de configurações são baseados em XML, onde suas ações são executadas (WIKIPÉDIA, 2007).

5.12 TAGLIBS

As bibliotecas de tags do JavaServer Pages definem declarativamente as funcionalidades modulares que podem ser reusadas em qualquer página JSP. As TagLibs reduzem a necessidade de encaixar uma grande quantidade de código Java nas páginas JSP, movendo a funcionalidade das tags para dentro das classes tags implementadas (SUN DEVELOPER NETWORK, 2007).

5.13 POSTGRESQL

O PostgreSQL é um poderoso sistema de banco de dados objeto-relacional de código aberto. “Ele é considerado objeto-relacional por implementar, além das características de um SGBD relacional, algumas características de orientação a objetos, como herança e tipos personalizados” (POSTGRESQLBR, 2007). Com mais de 15 anos de desenvolvimento, ganhou uma forte reputação de confiabilidade, integridade de dados e exatidão. Ele é multiplataforma e funciona na maioria dos sistemas operacionais. Possui suporte completo para chaves estrangeiras, joins, views, triggers e stored procedures. Existem sistemas PostgreSQL ativos em ambientes de produção que gerenciam mais de 4 terabytes de dados (POSTGRESQL, 2007).

6 O FRAMEWORK

A escolha das ferramentas e tecnologias que compõem o framework do projeto foi de fundamental importância para um desenvolvimento ágil e seguro do mesmo. Ele é baseado no padrão de projeto de software MVC que separa as tarefas de acesso aos dados e lógica do negócio da apresentação e interface com o usuário. Isso permitiu estruturar os arquivos de código do sistema de forma organizada, separando-os por suas funcionalidades. A seguir uma breve descrição dessa estrutura:

- beans – esse módulo contém todas as classes JavaBeans, ou seja, todos os objetos que fazem parte do sistema. Essas classes seguem certas convenções sobre nomes de métodos, construção e comportamento. Essas convenções permitem que ferramentas possam usá-las e reusá-las.
- action – nesse pacote encontram-se todas as classes que contém os métodos da “lógica de negócios”, onde são executadas ações de consulta ao banco de dados, validações e segurança. Fica entre a camada de persistência e de aplicação.
- DAO ou Data Access Object – nesse local encontram-se as classes com métodos para persistência com banco de dados. Onde são implementadas ações que utilizam instruções SQL. O DAO deve buscar os dados do banco e converter em objetos para ser usado pela aplicação, e também pegar os objetos e transformar em instruções SQL para ser salvo no banco de dados.
- business – esse pacote contém classes que agem como uma abstração para os métodos que fazem as regras de negócio no DAO. Elas escondem a complexidade que essas classes DAO possuem. Nas classes business podemos criar diferentes assinaturas de métodos que utilizam as funções DAO para consultar o banco de dados. Podendo incluir filtros, ordenações, exceções e paginação dos dados. Isso elimina a necessidade de criar códigos excessivos com instruções SQL e reutilizar de diferentes maneiras os já existentes. A camada business fica entre a action e o DAO.

- jsp - módulo que contém todos os arquivos JSPs do portal, onde estão todos os códigos da interface.
- museu_arquivos – diretório onde estão todos os arquivos do acervo e arquivos XML contendo informações sobre o menu principal, permissões e perfis do portal.

Com o uso do Hibernate não é preciso se preocupar a persistência dos dados, já que ele transforma uma consulta em um banco de dados relacional em um objeto e vice-versa. Além de criar todas as tabelas necessárias com seus atributos e relações à outros objetos. Seu arquivo de configuração permite ter controle total do PostgreSQL, o banco de dados utilizado.

O Spring une-se ao Hibernate e permite que as consultas ao banco de dados sejam mais poderosas. Ele gerencia a sessão do sistema retornando quando preciso todas as relações de um objeto persistente. Evitando a criação de excessivos códigos de consulta ao banco.

O Struts implementa o controlador da aplicação, que é responsável por toda a logística e pela integração das camadas de visualização e regra de negócio do sistema. Ele recebe e envia valores das páginas em JSP através das chamadas feitas pelas funções contidas no módulo action do framework e também executa consultas ao banco de dados.

A versão JDK 1.6 – Java Development Kit – possui uma extensão de suporte a programação genérica (generics). Os tipos genéricos evitam exceções de conversão (ClassCastException) ao percorrer uma coleção de objetos. Também possibilita a criação de classes tipadas. Os objetos são parametrizados de acordo com o tipo desejado. Assim não é necessário convertê-los para seus tipos originais para poder resgatar seus valores ou inserir novos.

A partir desses princípios foi desenvolvido uma biblioteca Java que implementa um mecanismo genérico para os métodos comuns de banco de dados, como: listar, editar, salvar e apagar. Essas funções são usadas por todos os objetos do sistema. A reutilização de código e o emprego da técnica de herança são

características muito importantes no framework, são usados sempre que possíveis para eliminar códigos repetitivos.

Além de todos esses conceitos para diminuir o código, foram criados rotinas em ANT para automatizar alguns processos. A seguir uma descrição dos scripts mais importantes:

- generate-beans - gera todas as classes beans a partir da interface dos objetos exportados pelo diagrama de classes do Jude.
- generate-hibernate - cria todos os arquivos do Hibernate necessários para mapear os relacionamentos entre os objetos e o banco de dados.
- generate-dao-interface - cria todas as classes referentes ao pacote DAO de todos os objetos.
- generate-business-interface – cria todas as classes do pacote business.
- build-war – gera o arquivo JAR necessário para servidor Tomcat executar o sistema.

Para cada objeto novo incorporado ao sistema é necessário haver uma estrutura de classes e arquivos de configurações para que ele possa se integrar ao resto dos componentes do projeto. São arquivos repetitivos e indispensáveis para o funcionamento da aplicação. Os scripts em ANT diminuem o processo penoso de replicar esses códigos.

Como é possível perceber, o framework desenvolvido pela equipe usa tecnologia de ponta, somente com softwares livres relacionados à linguagem de programação Java. Foram aplicados os mais conceituados padrões de projeto para um desenvolvimento sólido e organizado do projeto. As técnicas genéricas, de herança e os scripts ANT que eliminam excessos de códigos e processos repetitivos. Tudo isso dá ao programador mais segurança e rapidez para implementar um sistema de grande porte.

7 DESENVOLVIMENTO

Foram utilizados conceitos renomados de Engenharia de Software para desenvolver o sistema. Segundo Fedelli, Polloni e Peres (2003, p. 171):

“A Engenharia de Software ressalta a utilização de sólidos conhecimentos na área de engenharia, visando a obtenção de um software de padrão e qualidade elevados, proporcionando maior eficiência quando utilizado nas máquinas adequadas, e resolvendo de forma mais criteriosa os eventuais problemas que possam advir durante a sua implantação”.

A Engenharia de Software oferece mecanismos para planejar e gerenciar o processo de desenvolvimento, que permite ao engenheiro especificar, projetar, implementar e manter sistemas de software de qualidade. A metodologia usada é a linguagem padrão de modelagem UML. O processo deu-se por várias etapas: levantamento e análise de requisitos, definição dos atores e das funcionalidades do sistema, construção do protótipo e criação do diagrama de classes. Com o projeto modelado é possível iniciar a implementação do código fonte através do framework preparado. Todas essas etapas devidamente gerenciadas pelo gerente de projetos.

O trabalho é feito de forma concorrente entre os participantes, para isso, utiliza-se o CVS para controlar as modificações dos arquivos que pertencem ao projeto. Com o CVS a equipe mantém os arquivos sempre atualizados em um servidor onde todos possuem acesso.

7.1 LEVANTAMENTO DE ANÁLISE DE REQUISITOS

De acordo com Guedes (2004, p.19), uma das primeiras fases da Engenharia de Software consiste no Levantamento de Requisitos. “Nesta etapa, o engenheiro de software busca compreender as necessidades do usuário e o que ele deseja que o sistema a ser desenvolvido realize” (GUEDES, 2004). Isso é feito

através de entrevistas, onde o engenheiro tenta entender quais são os serviços que o cliente precisa que o software forneça. Foram realizadas entrevistas suficientes para a compreensão de todas as idéias e requisitos levantados pelo Museu da Pessoa para o novo portal.

Logo após o levantamento, os requisitos foram analisados verificando se estes foram especificados corretamente e se foram realmente bem compreendidos. “A partir da Análise de Requisitos são determinadas as reais necessidades do sistema de informação” (GUEDES, 2004).

7.2 ATORES DO SISTEMA

“Os atores representam os papéis desempenhados pelos diversos usuários que poderão utilizar de alguma maneira os serviços e funções do sistema” (GUEDES, 2004). O portal possui os seguintes atores:

- Administrador – super-usuário do sistema.
- Coordenador e Gestor – responsável pela coordenação, moderação, organização dos projetos e iniciativas do Museu. Pode atuar como gerente de projeto.
- Pesquisador – responsável pela inserção, edição e manutenção do conteúdo de projetos e iniciativas do Museu.
- Internauta – personagem que não é um usuário administrativo, ou colaborador direto do Museu da Pessoa. Pode contar a história de sua vida, a história de vida de outras pessoas, cadastrar acervos, criar e participar de comunidades, construir coleções sobre suas comunidades, ler e contribuir com histórias de vida de outras pessoas do acervo.

7.3 FUNCIONALIDADES

Foi feito um documento que apresenta uma descrição detalhada de cada funcionalidade do portal, inclusive protótipos não funcionais. Esses protótipos propõem apenas questões de natureza funcional, estando fora da função do wireframe apresentar projetos de layout e design. Esse documento é importante pois delimita o escopo do projeto.

7.3.1 MÓDULO ADMINISTRATIVO

CADASTRO DE PERFIL

A função do perfil é determinar permissões de acesso, visando facilitar o cadastramento de novos usuários no sistema. Durante o cadastramento de um novo usuário, é possível selecionar manualmente as permissões de acesso para cada funcionalidade, ou selecionar um perfil, que já seleciona automaticamente estas permissões, permitindo ainda que o usuário altere permissões específicas diferentes do perfil selecionado.

CADASTRO DE USUÁRIO

O cadastro de usuários é permitido tanto para usuários administrativos, como para usuários da parte pública do portal (internauta), permitindo uma unificação dos cadastros. Na versão atual do portal, o usuário internauta utiliza um cadastro diferenciado do administrativo, o que gerou um problema para pessoas que trabalham no Museu e gostariam de contar suas histórias no portal. Por isso, cada usuário administrativo receberá permissões para ser um internauta automaticamente. O contrário também será possível (um usuário internauta receber permissões administrativas posteriormente ao seu auto-cadastro).

AUTENTICAÇÃO

O sistema de autenticação (login) da área administrativa verifica as permissões do usuário que está entrando no sistema, e constrói o menu de funcionalidades dinamicamente, apenas com funções que ele possui permissão.

CONFIGURAÇÃO DO SISTEMA

O arquivo XML Schema que determina as regras para a estrutura dos depoimentos em XML é gerido através de uma interface administrativa, com controle de versões. Quando um novo depoimento indexado em XML é cadastrado no sistema, a validação do mesmo é realizada a partir do XML schema em sua versão mais atual.

O template padrão da home de pessoa é a formatação padrão utilizada na navegação e visualização do acervo do portal. Também é utilizado quando um projeto não possui identidade visual própria.

Os classificadores são palavras-chave que são apresentadas ao internauta que estiver consultando o acervo de uma pessoa, para que o mesmo possa classificar o conteúdo. Existem cadastros auxiliares de todos os combo-boxes de todas as fichas, permitindo o controle total do museu sobre esse conteúdo.

O esquema de indexação de conteúdo segue os seguintes parâmetros:

- um arquivo XSD com as regras da formatação do XML dos depoimentos.
- um arquivo XML com a lista de indexadores gerais do portal, cadastrados na área de configuração do sistema.
- um arquivo XML pra cada projeto com a lista dos indexadores específicos do mesmo.
- em cada depoimento, o sistema verifica com quais projetos o depoente está associado, para então validar as referências aos indexadores inseridos no meio do conteúdo.

7.3.2 MÓDULO ACERVO

FICHA PESSOA

A nova ficha de pessoa contém todas as informações de uma pessoa. Ela apresenta uma mudança quanto ao seu modelo de relacionamentos com os projetos. Na nova versão, a pessoa não está obrigatoriamente associada à um projeto, podendo então, relacionar-se com diversos projetos simultaneamente, ou com nenhum. No novo portal, o projeto assume um papel de organizar uma navegação temática pelas pessoas do portal. Outra novidade é a possibilidade de relacionamento entre diversas pessoas. Cada pessoa possui um acervo com imagens, áudios, vídeos e depoimentos.

FICHA IMAGEM

A ficha imagem possui todas as informações necessárias para armazenar uma imagem no sistema. Na nova arquitetura, pode se relacionar com projetos e pessoas.

FICHA AUDIOVISUAL

A ficha de audiovisual permite o armazenamento de áudios e vídeos. Ela pode ser relacionada tanto com projetos quanto com pessoas.

FICHA HISTÓRIA

A ficha de história contém informações de um depoimento. Contempla agora a tecnologia XML para classificar e contextualizar as informações do depoimento. O objetivo do uso do XML no cadastramento de histórias é permitir a

criação de metadados sobre o texto do depoimento, definindo marcações de acordo com um vocabulário controlado, determinado pelo XML Schema cadastrado previamente na área de configuração do sistema. Existe também arquivos XML auxiliares para definir os vocabulários controlados, determinando dicionários genéricos e específicos de cada projeto.

Cada projeto terá um arquivo de configuração em XML, onde serão alimentados os vocabulários controlados e as marcações específicas. Existe um arquivo de configuração geral para o portal com definições de palavras-chave elementares a todos os projetos. Ao inserir uma nova história, o sistema considera as regras do XML geral e do XML específico do projeto. Abaixo a imagem do fluxo de inserção de uma história:

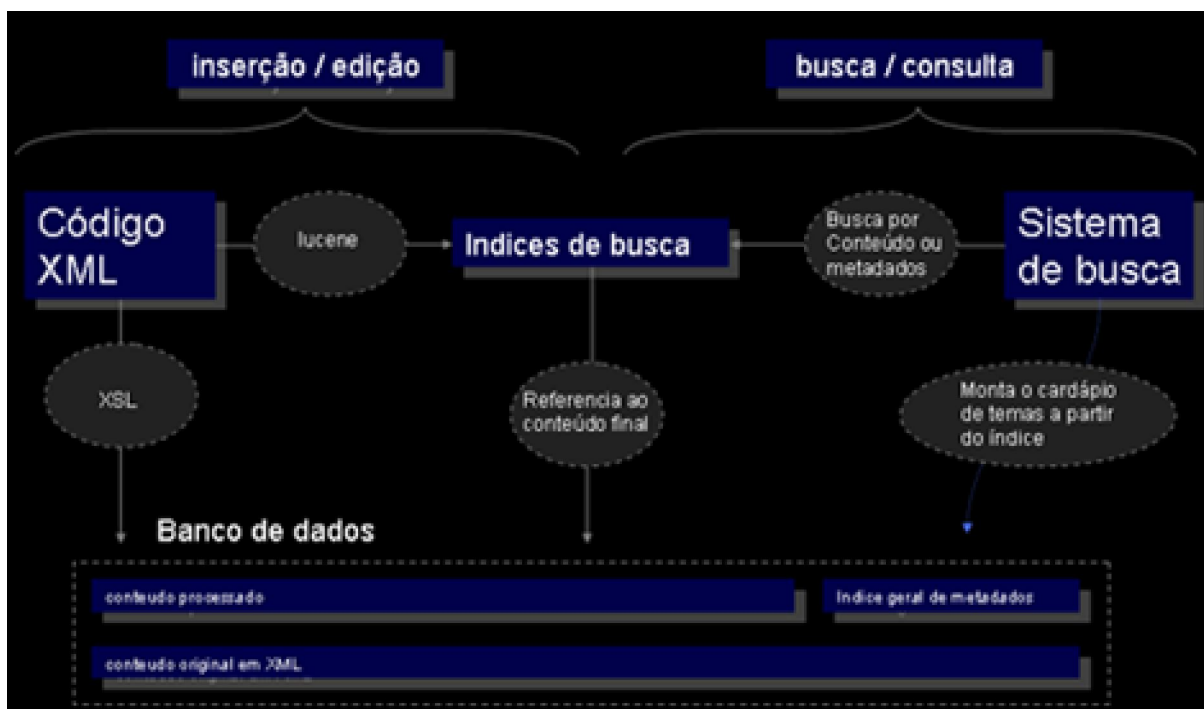


Figura 4 – Diagrama de fluxo de inserção de uma história

FICHA PROJETO

A ficha de projeto contém os dados de um projeto. As principais mudanças no novo portal são: a possibilidade de atrelar acervos diretamente ao projeto e a mudança em alguns campos do cadastro. Outra novidade é a possibilidade de anexar os kits com os templates da área pública (home de pessoas, listagens dinâmicas, glossário, formulários do internauta).

No momento do cadastro são associados dois arquivos XML. Um contendo o glossário do projeto (verbetes e descrição) e outro com o mapa dos indexadores específicos do projeto, que servirão para a realização de buscas temáticas no projeto.

Ao contrário da versão atual, em que as pessoas estão atreladas obrigatoriamente à um projeto, agora o projeto apresenta apenas ligações com pessoas já existentes no acervo, sendo que uma pessoa pode ter relacionamentos com N projetos.

HOME DE PESSOA

A home de pessoa é a vitrine pública do acervo de uma pessoa. Ela pode ser acessada através do portal ou através dos projetos e salas do portal. A nova versão de home de pessoa possibilita customizações mais flexíveis, que podem ser facilmente feitas pelos webdesigners do Museu da Pessoa. Para cada projeto do sistema pode-se cadastrar uma skin. Esses templates são compactados em um arquivo ZIP, que contém todos os JSPs, imagens e estilos de um determinado tema visual. Dentro desses arquivos JSPs poderá ser usado as TAGLIBS especialmente implementadas para realizar funções específicas no sistema.

7.3.3 MÓDULO DE BUSCA

O sistema contém buscas simples em todas as listas de dados. Também possui uma busca avançada de pessoas, acervo e histórias, onde é possível realizar buscas mais complexas. As buscas são efetuadas com a poderosa ferramenta Lucene, e podem ser implementadas por TAGLIBS para serem usadas nas skins dos projetos para acesso público.

7.3.4 MÓDULO PUBLICADOR

TEMPLATES

A idéia é ser uma funcionalidade que disponibilize kits de templates, para que o internauta possa construir exposições sobre o conteúdo do portal, sobre suas próprias histórias de vida e acervos. Permite inserir facilmente templates customizados e associá-los à projetos.

PÁGINAS

Com os templates disponibilizados pode-se criar páginas e inserir conteúdo de texto e imagens dinamicamente. Essas páginas serão salvas em arquivos HTML e terão seus endereços próprios de acesso. Essa funcionalidade gerencia todas elas, permitindo editá-las, publicá-las e removê-las.

7.3.5 MÓDULO INTERNAUTA

Possibilita ao internauta cadastrar-se no portal para contar sua história ou de uma outra pessoa. O internauta possui uma conta de usuário para entrar no sistema e permitir que gerencie seu conteúdo de acervo. Essas funcionalidades

podem ser facilmente implementadas nas páginas HTML de acesso público dos projetos pelas TAGLIBS desenvolvidas. Isso dá mais autonomia para a equipe do Museu criar projetos que utilizem o módulo internauta. Sendo assim, cada projeto pode ter sua área de login, cadastro de pessoas e histórias em suas páginas HTML.

7.4 PROTÓTIPO

“Um protótipo normalmente apresenta pouco mais que a interface do software a ser desenvolvido, ilustrando como as informações seriam inseridas e recuperadas no sistema apresentando alguns exemplos com dados fictícios de quais seriam os resultados apresentados pelo software” [15]. A prototipação é uma técnica que consiste em desenvolver rapidamente um modelo de como seria o sistema quando finalizado. Com ele é possível observar detalhes da ergonomia da aplicação, como cores, tipos de letras, estilos, bem como a disposição do seu layout. Um protótipo fiel ao portal foi desenvolvido, simulando algumas funcionalidades com dados fictícios. Ele mostra dois cenários básicos de layout: as listas e os formulários. Nele podemos observar a disposição do menu principal no canto superior da tela, esse menu dá acesso rápido a todas as funcionalidades do portal. As páginas seguem um estilo específico para as cores, fontes, botões e ícones. O protótipo possibilita testar a usabilidade do sistema com a ergonomia escolhida. Sua aprovação é fundamental, pois um dos requisitos do Museu era melhorar a ergonomia do portal e sua usabilidade. Logo abaixo exemplos de imagens do protótipo de uma lista e de um formulário do sistema:

MUSEU da PESSOA Usuário: Administrador

Home Administração Configuração Acervo Busca Publicador Trocar Senha Sair do Sistema

Administração >> Cadastro de Usuários >> Lista de Usuários Cadastrar Usuário

Lista de Usuários

Nome do Usuário Login Perfil Situação bloqueado liberado todos Buscar
Limpar

NOME / LOGIN	BLOQ.	ÚLTIMO ACESSO	PERFIL	AÇÕES
<input type="checkbox"/> Administrador (admin)	<input checked="" type="checkbox"/>		Administrador	<input type="text"/> <input type="text"/>

Página 1 de 1

Editar Apagar

Figura 5 – Exemplo da lista do protótipo

MUSEU da PESSOA Usuário: Administrador

Home Administração Configuração Acervo Busca Publicador Trocar Senha Sair do Sistema

Administração >> Cadastro de Usuários >> Incluir Usuário

Incluir Usuário

Campos marcados com * são obrigatórios

Nome *

E-mail

Telefone **Ramal** **Celular**

Login * administrador do sistema usar e-mail como login

Perfil * >> [Clique aqui para atribuir permissões específicas à este usuário](#)

Senha * **Confirmação *** **Lembrete ***

Figura 6 – Exemplo de cadastro do protótipo

7.5 DIAGRAMA DE CLASSES

O diagrama de classes é um dos documentos mais importantes para o programador. “Seu principal enfoque está em permitir a visualização das classes que comporão o sistema com seus respectivos atributos e métodos, bem como em demonstrar como as classes do diagrama se relacionam, complementam e transmitem informações entre si” [15]. Esse diagrama se preocupa em definir a estrutura lógica das classes, mostrando como elas estão organizadas. Ele basicamente é composto por suas classes e associações existentes entre elas. Com o escopo do projeto fechado após a aprovação do documento de funcionalidades (wireframe) e do protótipo pelo cliente. A equipe definiu todas as classes da aplicação, seus atributos e relacionamentos. Cada entidade tem seu papel no sistema e necessita que seus dados sejam armazenados e seus relacionamentos preservados. Para desenvolver o diagrama de classes foi usado o programa para modelagem de dados JUDE – Java and UML Developers Environment. Ele é um software multiplataforma escrito em Java e existe uma versão gratuita para download. Possui uma interface gráfica clara, bem intuitiva e dispõe de oito tipos de diagramas diferentes para trabalhar. Logo abaixo, uma imagem do diagrama de classes do módulo do publicador do projeto, exportada pelo JUDE:

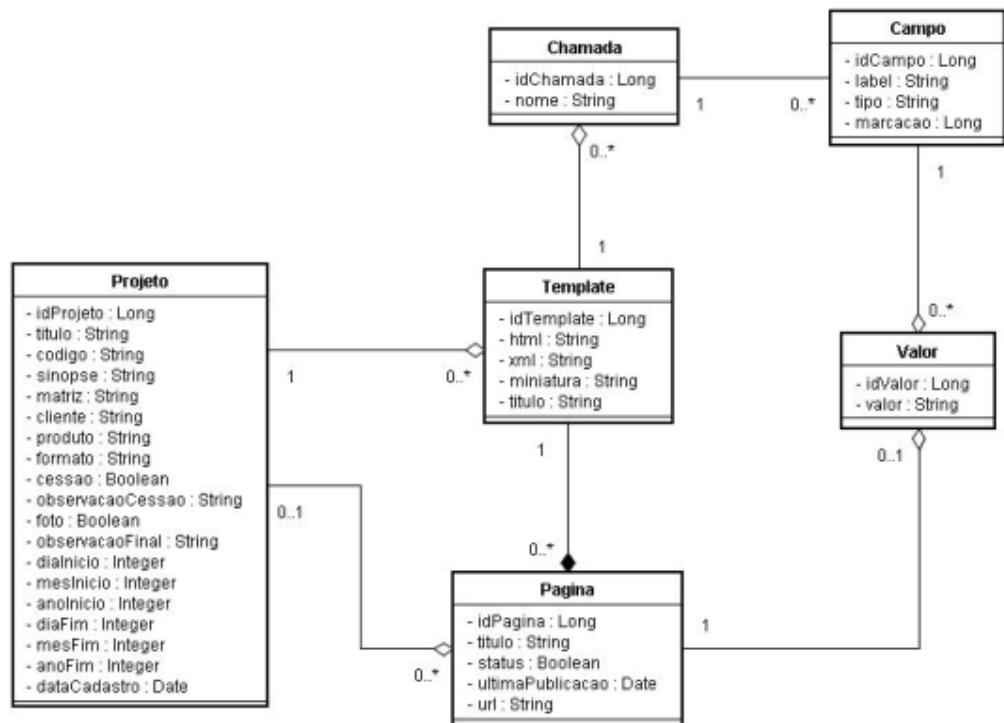


Figura 7 – Diagrama de classes do módulo do publicador

7.6 IMPLEMENTAÇÃO DAS FUNCIONALIDADES

A estrutura básica de arquivos do sistema é gerada automaticamente pelos scripts em ANT do framework. O JUDE possui uma ferramenta para exportar o diagrama de classes para arquivos de interface Java. Foi criado a partir dessas interfaces utilizando o script generate-beans, todas as classes beans e seus arquivos de Hibernate. Todos os arquivos dos pacotes actions, business e DAO também são gerados, inclusive os arquivos de configurações do Struts e do Spring. Com tudo isso pronto, foi possível iniciar a implementação das funcionalidades do portal.

Cada membro da equipe é responsável pela implementação de determinadas partes do sistema, o código implementado deve ser limpo e sempre testado. O desenvolvimento é de forma simultânea, e o projeto é sempre atualizado no servidor CVS quando uma parte é finalizada.

Como já mostrou o protótipo, existem dois tipos básicos de cenários no portal: os formulários e as listas. Todos os formulários passam por testes de validações para garantir a integridade dos conteúdos armazenados. As páginas de listas possuem buscas simples de conteúdo e seus dados são paginados. Essa paginação é feita por uma biblioteca criada em conjunto com TAGLIBS, que permite paginar os dados diretamente pelo banco de dados e facilmente implementá-las nas páginas JSP.

As ações padrões do sistema são: listar, editar, salvar e apagar. Todas as classes actions estendem uma classe chamada DefaultPagedAction que implementa essas ações de forma genérica, utilizando a biblioteca com a tecnologia Generic DAO. Com isso, só é necessário implementar essas funções em casos de exceções, onde a mudança do método é indispensável.

7.6.1 CADASTRO DE PERFIS

Essa funcionalidade permite gerenciar os perfis de acesso ao sistema. Apresenta a página com o formulário de cadastro e a página com a lista dos perfis inseridos. Cada perfil é composto por permissões, que determinarão o nível de acesso às funcionalidades do sistema de cada usuário. Para isso, foi implementada uma biblioteca – Guarda.jar - para controlar as permissões de acesso dos usuários no portal e criar o menu principal de acordo com essas permissões. Essa biblioteca trabalha junto com três arquivos XML:

- permissões.guarda.xml – esse arquivo contém um mapa das funcionalidades do sistema. Todas as ações que o usuário acessa diretamente devem estar contidas nesse arquivo. A estrutura dos dados são organizadas basicamente pelos módulos que contém suas funcionalidades.

```
<modulo id="ADMINISTRACAO" titulo="Administração" diretorio="admin">
  <funcionalidade id="PERFIL_LISTAR" titulo="Listar Perfis" action="perfil.do"
actionParam="listar" />
```

- menu.guarda.xml – o menu principal é gerado a partir desse XML. Nele contém todos os dados que formam os menus e sub-menus do sistema e suas respectivas ações. Essa hierarquia pode ser organizada de forma recursiva, possibilitando a inserção de diversos sub-menus. Abaixo um exemplo:

```
<menu id="ADMINISTRACAO">
  <titulo>Administração</titulo>
  <submenus>
    <menu id="USUARIO">
      <titulo>Cadastro de Usuários</titulo>
      <link>/admin/usuario.do?acao=listar</link>
      <submenus>
        <menu id="LISTAR">
```

```

        <titulo>Lista de Usuários</titulo>
        <link>/admin/usuario.do?acao=listar</link>
    </menu>
    <menu id="INCLUIR">
        <titulo>Cadastrar Usuário</titulo>
        <link>/admin/usuario.do?acao=cadastrar</link>
    </menu>
</submenus>
</menu>
</menu>

```

Abaixo a figura 8 mostra o resultado:



Figura 8 – Exemplo do menu em XML

- config.guarda.xml – esse arquivo contém todos os perfis e suas permissões. Além das funcionalidades que são liberadas para acesso público.

```

<paginasLiberadas>/indexProjeto.do </paginasLiberadas>
<papeis>
  <papel>
    <id>ADMINISTRADOR</id>
    <titulo>Administrador</titulo>
    <visivel>>false</visivel>
    <permissoes id="GLOBAL_ADMINISTRADOR" />
  </papel>
</papeis>

```

A inserção de um perfil é feita por um formulário de cadastro que possui uma lista com todas as funcionalidades que foram mapeadas pela biblioteca Guarda através do arquivo XML “permissões.guarda”. A partir dessa lista pode-se escolher as permissões do perfil, conforme a imagem:

Cadastro de Perfil Campos marcados com * são obrigatórios

Nome do Perfil Id do Perfil

Visível

PERMISSÕES

↳ Acervo	<input type="checkbox"/>
↳ Listar Acervos	<input type="checkbox"/>
↳ Apagar Áudiovisual	<input type="checkbox"/>
↳ Cadastrar Áudiovisual	<input type="checkbox"/>

Figura 9 – Cadastro de perfil

7.6.2 CADASTRO DE USUÁRIOS

Nessa área é possível inserir, editar e listar os usuários do sistema. Todos os usuários são associados à um perfil com determinadas permissões, também possuem um login e senha que garantem o acesso ao portal. O sistema não permite o cadastro de um login duplicado.

O perfil de cada usuário pode ser customizado. As permissões de cada um podem ser definidas pelo administrador. Esse foi um dos requisitos do projeto. Porém essa flexibilidade trouxe algumas adversidades. Quando as permissões de um perfil cadastrado são alteradas, temos a opção de atualizar os perfis de todos os usuários perdendo as suas customizações, ou não atualizar. Outra questão é em relação ao banco de dados, veja abaixo o diagrama de classes entre o usuário e o perfil:

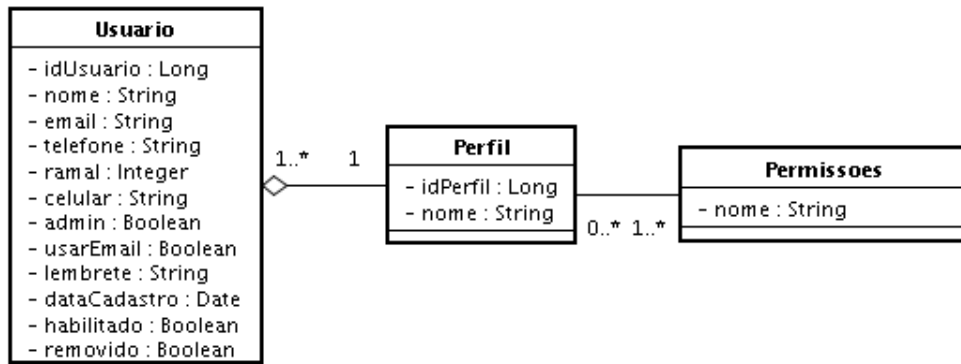


Figura 10 – Diagrama de classes do usuário

A relação é dada da seguinte maneira, para cada usuário é criado um novo perfil com N permissões. Essa combinação possibilita identificar as permissões de cada usuário, porém aumenta consideravelmente a quantidade de registros na tabela permissões.

Quando um usuário é removido, ele é apagado logicamente do sistema, ou seja, é modificado o atributo removido para true e seus dados permanecem armazenados no banco de dados. A lista de usuários retorna somente os usuários com o atributo removido igual a false.

Um usuário só pode acessar o sistema se estiver habilitado, a página da lista de usuários permite ao administrador bloquear ou desbloquear o acesso dos usuários, que são facilmente identificados nessa lista por um ícone especial. A página contém busca por: nome, login, perfil e situação (bloqueado ou desbloqueado).

Lista de Usuários

Nome do Usuário Login Perfil Situação bloqueado liberado todos

NOME / LOGIN	BLOQ.	ÚLTIMO ACESSO	PERFIL	AÇÕES
<input type="checkbox"/> Administrador (admin)		24/09/2007 20:54	Administrador	
<input type="checkbox"/> Rafael (rafael)		nunca acessou	Internauta	

Página 1 de 1

Editar Apagar

Figura 11 – Lista de usuários

7.6.3 AUTENTICAÇÃO

Para realizar uma ação no portal, como listar os projetos, é preciso que o usuário tenha o perfil com a permissão necessária e que sua sessão não tenha expirado. Cabe a biblioteca Guarda monitorar os acessos, se for negado ou se a sessão expirar, ele será redirecionado para a página de login da área administrativa do sistema. Somente os usuários desbloqueados e com logins e senhas válidas poderão entrar no sistema. A seguir a imagem da tela de login do portal:

MUSEU DA PESSOA

Acesso Restrito - Área Administrativa

login senha

[Esqueci minha senha!](#)

Figura 12 – Tela de login da área administrativa.

7.6.4 CADASTROS AUXILIARES

Nessa área encontram-se todos os cadastros que auxiliam o funcionamento do portal, pertence ao módulo de Configurações do Sistema. São dados pré-definidos que aparecem nos campos de formulários principais para o usuário escolher. Uma característica dessas funcionalidades é uma única página para inserir e listar os dados.

1. **Campos Comuns** – são os campos que são comuns a muitos formulários (Estado, Idioma, Período).
2. **Campos da Ficha Pessoa** – são os campos: Cor/Raça, Religião, Escolaridade, País, Profissão.
3. **Campos da Ficha História** – campos para auxiliar a inserção de uma história (Suporte da Entrevista, Tipo de Entrevista, Formato da Entrevista, Tratamento).
4. **Campos da Ficha Projeto** – auxiliam o cadastro de um Projeto (Perfil, Produtos, Formato dos Depoimentos).
5. **Campos da Ficha Imagem** – possui somente o campo Tipo de Imagem.
6. **Campos da Ficha Audiovisual** – somente o Formato Original.

A seguir uma imagem de um formulário simples para cadastro auxiliar:

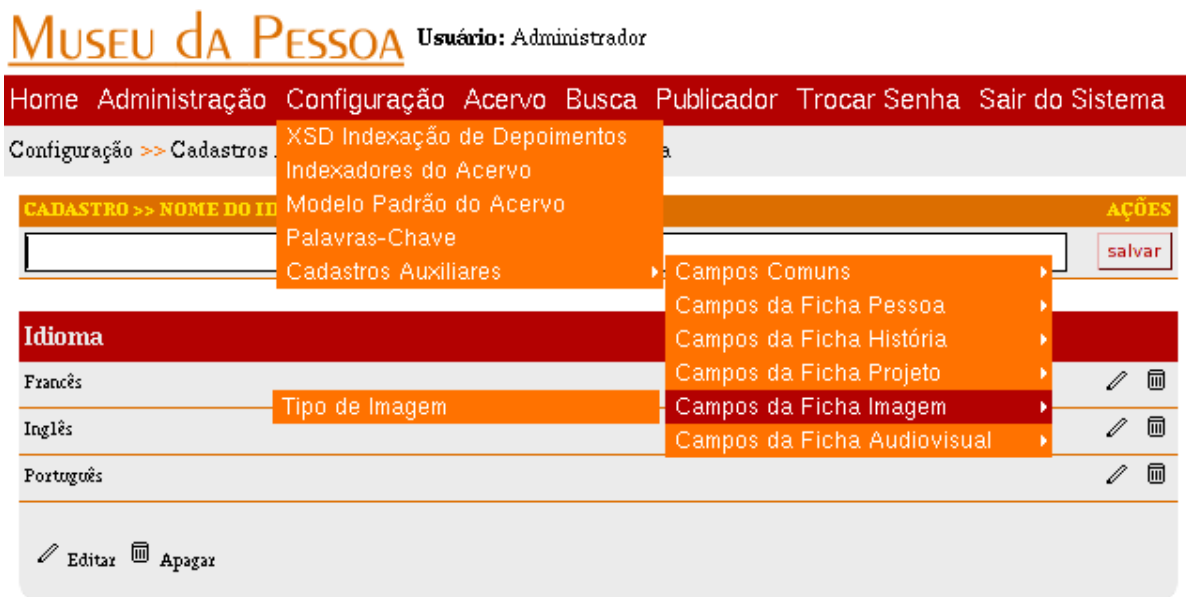


Figura 13 – Exemplo de cadastro auxiliar

A figura 14 mostra o resultado de um campo em um formulário de cadastro:

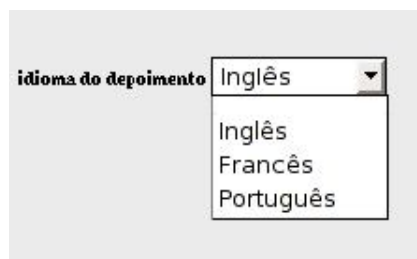


Figura 14 – Resultado do exemplo do cadastro auxiliar

7.6.5 CADASTRO DE PALAVRAS-CHAVE

O cadastro de palavras-chave apresenta a interface para inserir as palavras que servirão de classificação do conteúdo. Essas palavras auxiliam na organização e na busca do acervo.

7.6.6 XSD INDEXAÇÃO DE DEPOIMENTOS

A idéia é que se cadastre um arquivo XML Schema pelo sistema, onde serão determinadas todas as regras genéricas para a indexação dos depoimentos (nomes de tags, tipos de atributos, etc). O cadastro do Schema prevê um controle de revisões, para que não tenha incoerências nos depoimentos já relacionados quando qualquer especificação no Schema for modificado. A indexação é feita na forma de atributos conforme o exemplo:

```
<indice nome="familia">  
  o texto vem aqui  
</indice>
```

Dessa forma é possível validar os nomes dos índices conforme um vocabulário controlado a partir dos projetos em que a pessoa está inserida, e também com os indexadores gerais, independente das regras do XML Schema. Isso permite ter diversos critérios de classificação de um trecho indexado, como no exemplo:

```
<indice nome="familia" idioma="pt_BR" link="http://"(...+atributos...)>  
  o texto vem aqui  
</indice>
```

Todos os depoimentos inseridos no sistema com formato XML serão validados a partir da versão mais atual do XML Schema.

7.6.7 INDEXADORES DO ACERVO

Parte onde permite cadastrar os indexadores gerais do sistema para o Lucene indexar os depoimentos XML a partir do seu Schema.

7.6.8 CADASTRO DE PESSOAS

A página de cadastro de uma pessoa é uma das mais complexas do portal. O formulário é extenso e possui campos organizados por blocos: Dados Pessoais, Dados Profissionais, Dados do Pai, Dados da Mãe e Outras Informações. Nesse mesmo cadastro existe a possibilidade de ativar uma conta de usuário com perfil internauta para a pessoa, com um login e senha, que darão acesso ao portal. Os dados do usuário são preenchidos com as informações do depoente. Pode-se acrescentar campos específicos relacionados à um projeto para suprir a necessidade de mais informações na inserção de um indivíduo, isso dá mais flexibilidade pois muitos projetos necessitam de mais dados no cadastro além dos existentes.

Cada pessoa possui um acervo com histórias, imagens e audiovisuais. Pode também se relacionar com outras pessoas e projetos. Pensando em uma melhor usabilidade, foi colocada abas no cabeçalho do formulário possibilitando uma rápida navegação entre essas funcionalidades. Existe uma opção na ficha para assinar ou não a cessão de direitos da pessoa, essa opção vale para todo o seu acervo. Quando uma pessoa é inserida no banco de dados, ela recebe o status pendente. Assim um administrador pode facilmente reconhecer os novos cadastros e analisar seu conteúdo para decidir se irá aprová-lo ou reprová-lo.

A página da lista de pessoas tem linhas com cores diferentes para diferenciar a situação de cada um. Também permite mudar o status de diversos depoentes ou removê-los facilmente. Somente os aprovados serão visualizados nos projetos virtuais do Museu. Nela pode-se realizar buscas para filtrar os resultados por: período da última alteração, nome, autor, projeto, situação, somente internautas e com cessão de direitos assinada. Existe um botão especial que abre um menu para ir direto às listas do acervo de uma pessoa ou seus relacionamentos.

Assim como nos usuários, a remoção de uma pessoa é lógica e não física, para manter a integridade do banco de dados. Abaixo uma imagem de um

pedaço da página de cadastro, observando as abas e os campos com dados dos cadastros auxiliares:

The image shows a web interface for a person's registration. At the top, there are four tabs: 'cadastro de pessoa' (selected), 'histórias', 'acervo', and 'relacionamentos'. Below the tabs is a red header bar with the text 'Dados Pessoais' on the left and 'Campos marcados com * são obrigatórios' on the right. The form contains the following fields:

- Nome:** Text input field.
- Cod. Depoente:** Text input field.
- Sexo:** Radio buttons for 'fem' and 'masc'.
- Endereço:** Large text input field.
- Bairro:** Text input field.
- Cep:** Text input field.
- Cidade:** Text input field.
- Estado/província:** Dropdown menu.
- E-mail:** Text input field.
- País:** Dropdown menu.
- Telefone:** Text input field.
- País de nascimento:** Dropdown menu.
- Data de nascimento:** Text input field.
- Cidade de nasc.:** Text input field.
- Estado/prov. nasc.:** Dropdown menu.
- Biografia:** Large text input field.
- Cor/raça:** Dropdown menu.
- Religião:** Dropdown menu.

At the bottom of the form, there is another red header bar with the text 'Dados Profissionais' on the left and 'Campos marcados com * são obrigatórios' on the right.

Figura 15 – Ficha de cadastro de pessoa

Nessa outra imagem vemos as diferentes cores identificando o status, o menu especial, as buscas e os botões para alterar a situação:

MUSEU da PESSOA Usuário: Administrador

Home Administração Configuração Acervo Busca Publicador Trocar Senha Sair do Sistema

Acervo >> Pessoas >> Lista de Pessoas Cadastrar Pessoa

Lista de Pessoas Cadastradas

Período da última alteração

 a

Nome da Pessoa

Nome do Autor

Projeto

Filtro

somente internautas

com cessão de direitos assinada

Situação

aprovados

reprovados

pendentes

Buscar Limpar

ID / NOME	STATUS	AUTOR	AÇÕES
<input type="checkbox"/>  [9595] [2ACII - D - EMEB Prof. Florestan Fernandes - Vania Cristina Vieira]	? Pendente	vania cristina vieira	  
<input type="checkbox"/>  [9587] [2º ano do ciclo II H - Emeb Maria Rosa Barbosa - Mezle Amazilia Navarro]	Reprovado	Mezle Amazilia Navarro	  
<input type="checkbox"/>  [7619] [2º ano C da EMEF Pe Nildo do Amaral Jr. docente Andréa Ma Lima]	Aprovado	andrea maria de lima	  

remover aprovar reprovar

Página 1 de 936 1 2 3 4 5 6 7 8 9 10

Atalhos  Editar  Apagar

Figura 16 – Lista de pessoas

7.6.9 CADASTRO DE HISTÓRIAS

Uma história é inserida de duas maneiras, com conteúdo em XML ou sem. A ficha de cadastro possui diversos campos dos cadastros auxiliares, pode-se também classificar a história com as palavras-chave do sistema. Existe um botão para listar todos os indexadores disponíveis e uma caixa de texto para conteúdo sem XML com opções como negrito, itálico e sublinhado. Do lado esquerdo da caixa consta a lista de todas as tags disponíveis no XML Schema para estruturar o texto.

Após a inserção do depoimento em XML com as marcações e os indexadores, o sistema valida a formatação XML e verifica se os indexadores

escolhidos existem. Em seguida, processa o conteúdo XML a partir do XML Schema para depois o Lucene gerar os índices de busca da história e das marcações. Então o conteúdo processado é armazenado assim como o texto em XML. As tags disponibilizadas pelo Schema são:

- mp – indica o início e o fim do texto.
- identification – texto de identificação.
- p – transforma o texto em parágrafo.
- glossary – caso exista um glossário para a palavra ela será mostrada por uma pequena caixa de texto.
- citation – tranforma o texto para uma citação.
- bold – muda o texto para negrito.
- italic – muda o texto para itálico.
- link – cria um link para o texto com o endereço escolhido.
- page – define o conteúdo de uma página.
- index – indexa o conteúdo do texto a partir da lista de indexadores existentes.

A estrutura do texto deve seguir a seguinte regra:

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<mp>
  <identification>
    <p>
      Texto de identificação
    </p>
  </identification>
  <page>
    <index name="indexador" title="Nome do Indexador">
      <p>
        Texto da página
      </p>
    </index>
  </page>
</mp>
```

O resultado é o conteúdo da história indexado, formatado e dividido entre as páginas definidas, pronto para ser visualizado. Quando a home de pessoa for mostrada, o sistema trará o conteúdo já processado pelo StyleSheet, em forma HTML comum. O internauta ao pesquisar pelos metadados ou pelo conteúdo do depoimento, não irá pesquisar diretamente sobre o arquivo XML, mas sobre os índices do Lucene, que estarão referenciando diretamente o conteúdo processado, no banco de dados. O desempenho das buscas será melhor. O XML não é consultado diretamente, ele é um meio de transformação dos dados. Logo abaixo uma imagem de uma parte da página de cadastro de história:

cadastro de pessoa | **histórias** | acervo | relacionamentos

História Campos marcados com * são obrigatórios

conteúdo XML?

idioma do depoimento ▼

Lista de Indexadores

↳	mp	[+]
↳	identification	[+]
↳	p	[+]
↳	glossary	[+]
↳	citation	[+]
↳	bold	[+]
↳	italic	[+]
↳	link	[+]
↳	page	[+]
↳	index	[+]

```

<?xml version='1.0' encoding='ISO-8859-1'?>
<mp>
  <identification>
    <p>
      Texto de identificação
    </p>
  </identification>
  <page>
    <index name="indexador" title="Nome do Indexador">
      <p>
        Texto dp index
      </p>
    </index>
  </page>
</mp>

```

Sobre a História Campos marcados com * são obrigatórios

Título da história: *

Figura 17 – Ficha de cadastro de história

7.6.10 RELACIONAMENTOS DO DEPOENTE

Essa área gerencia os relacionamentos de uma pessoa. Ela pode estar ligada a diversas pessoas e projetos. A página da lista é simples e divide as listas dos dois tipos de relações. Pode-se editar a ficha de uma pessoa ou projeto a partir do botão ao lado de seu nome que quando clicado abre uma caixa de informações extras.

The screenshot shows a web interface for managing relationships. At the top, there are navigation tabs: 'cadastro de pessoa', 'histórias', 'acervo', and 'relacionamentos'. The main title is 'Relacionamentos do Depoente'. Below this, there are two columns: 'PESSOAS RELACIONADAS' and 'PROJETOS RELACIONADOS'. Each column has an 'Associar' button and a trash icon. The 'PESSOAS RELACIONADAS' column lists '[Adriana Batista]' and '[Edmar dos Santos Viana]'. The 'PROJETOS RELACIONADOS' column lists '[Histórias de Internet]'. At the bottom left, there are 'Editar' and 'Apagar' buttons. At the bottom right, there is a yellow box containing details: 'data de início: 01/01/2003', 'data de fim: //', and 'cliente/patrocinador: Instituto Museu da Pessoa.Net', along with a 'Cancelar' button.

Figura 18 – Relacionamentos do depoente

A página para associar contém buscas para filtrar os dados e chegar ao resultado esperado. Ela é igual para projeto e pessoa. A lista não retorna os dados já associados e possibilita diversas associações de uma vez. Também possui a caixa com informações clicando no nome. Conforme a imagem abaixo:

Relacionar projetos ao cadastro de Francisco Ferreira Rafael

Nome do Projeto

Código do Projeto

Filtro

com cessão de direitos assinada

Situação

publicados

retizados

não publicados

PROJETOS

Museu.Clube da Esquina	<input type="checkbox"/>
Santos Futebol Clube	<input type="checkbox"/>
São data de início: 01/01/1995 data de fim: // cliente/patrocinador: Santos Futebol Clube	<input type="checkbox"/>

Figura 19 – Tela para associar projetos ao depoente

7.6.11 CADASTRO DE PROJETOS

Nessa área o Museu administrará os seus projetos virtuais. Todos possuem um acervo de imagens e audiovisuais, não existe a possibilidade de inserir uma história. O cadastro também tem a opção de assinar ou não a cessão de direitos.

Um projeto pode ter um projeto matriz associado. Isso permite criar uma árvore de relacionamentos. Pode-se também associar pessoas, definir uma equipe para administrar o projeto com usuários e coordenadores, inserir campos específicos, criar um glossário e enviar um template. Todas essas funcionalidades são organizada em abas e em um menu especial de fácil acesso. Todas as funcionalidades serão explicadas posteriormente.

Existem três situações possíveis para um projeto. Depois de inserido no sistema ele é pendente, podendo ser publicado ou não. Na página da lista as linhas são de cores diferentes para diferenciar as situações dos projetos. No seu canto inferior esquerdo encontram-se botões para alterar facilmente o status. Na parte superior consta uma área para realizar buscas por: nome do projeto, código, pessoa

relacionada, situação e com cessão de direitos assinada. Somente os projetos publicados podem ser visualizados nas páginas externas de acesso público. Na figura abaixo o exemplo do cadastro de projeto:

cadastro de projeto equipe pessoas arquivo templates campos específicos glossário

Dados do Projeto Campos marcados com * são obrigatórios

Título do projeto: **Cod. Projeto:** **Idioma:**

Sinopse:

Projeto Matriz: **Perfil:**

Cliente/Patrocinador: **Parceiros:**

URL: **Data de início:** **Data de fim:**

Outras informações Campos marcados com * são obrigatórios

Cessão de direitos assinada?
 sim não

Figura 20 – Ficha de cadastro do projeto

Na página da lista pode-se clicar no nome do projeto e aparecerá uma pequena caixa amarela com mais informações além das que são retornadas em cada linha, veja o exemplo:

MUSEU DA PESSOA Usuário: Administrador

Home Administração Configuração Acervo Busca Publicador Trocar Senha Sair do Sistema

museu >> Cadastro de Projeto >> Lista de Projeto Cadastrar Projeto

Lista de Projetos Cadastrados

Nome do Projeto

Código do Projeto

Buscar Pessoa Associada ao Projeto

Filtro
 com cessão de direitos assinada
Situação
 publicados
 retirados
 não publicados

COD / PROJETO	STATUS	ID BASE	AÇÕES
<input type="checkbox"/> [Bial - 01] [5 ^o Bial da União Nacional dos Estudantes]	Publicado	10169	
<input type="checkbox"/> [SMUS] [5 ^a Semana Nacional de Museus]	data de início: 08/02/2007	10173	<div style="background-color: #f4a460; padding: 5px;"> Equipe Pessoas Acervo do Projeto Templates Campos Específicos </div>
<input type="checkbox"/> [ABC] [ABC de Luta - Preservação da Memória d	data de fim: //	10128	
<input type="checkbox"/> [ATD] [Abifazma 50 Anos]	cliente/patrocinador: Secretaria Estadual da Cultura e outros	10077	

Página 1 de 16 1 2 3 4 5 6 7 8 9 10 ▶

Atalhos Editar Apagar

Figura 21 – Lista de projetos

7.6.12 RELACIONAMENTOS DO PROJETO

Nesse local podemos relacionar pessoas ao projeto. O procedimento de associação é o mesmo que a funcionalidade de Relacionamentos dos Depoentes. Porém a página da lista contém somente pessoas. Todas elas são importantes, pois isso significa que essas pessoas tiveram alguma relevância histórica para o projeto. O relacionamento entre pessoa e projeto é de muitos para muitos, uma pessoa pode ter diversos projetos e um projeto pode ter diversas pessoas.

7.6.13 EQUIPE DO PROJETO

Um projeto pode ser administrado por uma equipe com usuários e coordenadores. O processo de associação é o mesmo da funcionalidade de

Relacionamentos do Depoente. A página da lista é dividida ao meio também. Uma com a lista dos usuários e a outra de coordenadores.

7.6.14 CAMPOS ESPECÍFICOS

Alguns projetos necessitam de mais campos na ficha de cadastro de uma pessoa. Para eliminar esse problema foi desenvolvida a funcionalidade para criar campos específicos de um projeto. Cada campo possui um nome, um identificador e um tipo, que pode ser: texto, número ou data. Quando os dados de uma pessoa são preenchidos, podem-se escolher os campos específicos selecionando o projeto desejado. Para implementar essa função foi preciso relacionar as classes da seguinte forma:

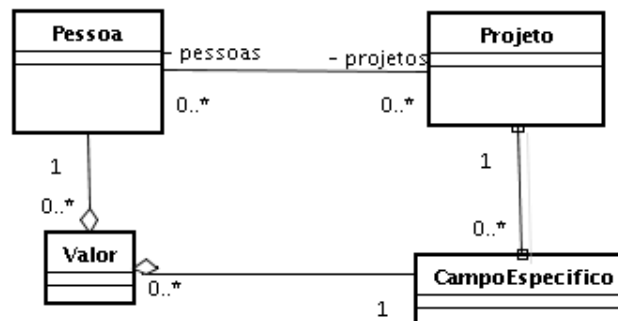


Figura 22 – Diagrama de classes do campo específico

Um projeto contém nenhum ou alguns campos específicos. Esses campos podem ter ou não valores relacionados à eles, e os valores são associados à uma pessoa que pode estar em diversos projetos onde encontram-se os campos. A página é simples, o cadastro e a lista de dados estão juntos na interface:

Campos Específicos do Projeto

CADASTRAR CAMPO **IDENTIFICADOR** **TIPO**

CAMPOS CADASTRADOS	IDENTIFICADOR	TIPO	
Anos de trabalho	ANOSTRABALHO	Numero	
Data de admissão	DATA	Data	
Nome do setor	NOME	Texto	

Editar Apagar

Figura 23 – Campos específicos

7.6.15 GLOSSÁRIO

Existe a possibilidade de criar um glossário para o projeto, através dele pode-se incluir palavras e suas descrições. Quando a história de uma pessoa for visualizada em alguma página externa relacionada ao projeto, ela será processada antes, o código varre o conteúdo da história em busca de palavras que estejam no glossário do projeto. Com isso, são acrescentados links às palavras que abrem uma pequena caixa de informação quando clicados contendo a descrição das mesmas. O conteúdo permanece inalterado no banco de dados, sendo modificado em tempo real quando selecionado.

7.6.16 TEMPLATES DO PROJETO

A forma para visualizar o conteúdo do acervo de um projeto é feita a partir de seu template, também chamado de home de pessoa. O template é um pacote contendo páginas JSP, imagens, arquivos de estilos, javascripts e tudo que for necessário para construir uma página. O conteúdo das páginas JSP são montadas dinamicamente através de variáveis guardadas na sessão e na utilização das taglibs especialmente implementadas para a ocasião.

Essa é a ferramenta desenvolvida para diminuir a dependência do Museu da Pessoa ao criar novos portais para seus projetos virtuais. A lógica utilizada dá

mais flexibilidade aos webdesigners do Museu, pois não é mais preciso alterar o sistema para publicar uma nova sala para disponibilizar o acervo de um projeto. Basta seguir a estrutura e regras necessárias para seu funcionamento.

As taglibs foram projetadas para executar diversas funções no sistema, como consultas e escritas ao banco de dados. Cada uma é associada à uma classe Java que executa o código necessário retornando a informação desejada. De forma dinâmica podemos retornar a lista de pessoas relacionadas à um projeto, ou a lista do acervo de uma pessoa, entre outros. Além de poder inserir dados no sistema e alterar o layout da página à partir dessas classes Java. Foram criadas todas as funções necessárias para gerenciar o conteúdo do portal Museu da Pessoa. A seguir algumas funções taglibs:

- `iterate` – itera os dados de acordo com as variáveis passadas. Pode-se retornar uma lista de pessoas, projetos e do acervo. Algumas variáveis importantes são: identificador do objeto, nome da variável e paginar. O exemplo abaixo retorna a lista de pessoas do projeto com identificador igual à 123, o nome da variável é “pessoa” e através de seu atributo obtemos seu nome.

```
<museu:iteratePessoa var="pessoa" idProjeto="123" paginar="true">
    ${pessoa.nome}
</museu:iteratePessoa>
```

- `paginador` – essa tag retorna a estrutura da paginação de uma lista, cria os números e os links de acordo com as variáveis. Foi feito para ser utilizado junto com o `iterate`.

```
<museu:paginadorPessoa idProjeto="123" styleClass="txtLista" />
```

- `objeto` – obtém os dados de um objeto, podendo ser da pessoa, do projeto ou do acervo. Através da tag `print` é impresso na página o atributo do objeto em que estiver dentro.

```
<museu:pessoa idObjeto="7899">
  <museu:print campo="nome"/>
</museu:pessoa>
```

- select – essas tags montam caixas de seleções nas páginas com os dados dos cadastros auxiliares, possibilitando criar formulários dinâmicos. O exemplo abaixo monta um select com a lista dos estados cadastrados no sistema.

```
<museu:selectEstado styleClass="dropDownclass"/>
```

- busca – existe um mecanismo para realizar buscas simples no acervo. Especificando os parâmetros necessários, o código abaixo executa a busca de uma pessoa no projeto 123 e vai para a página “busca.jsp”.

```
<form name="formBusca" method="post"
action="indexProjeto.do?pagina=busca.jsp&idProjeto=123">
  <input type="text" name="buscarPessoa"/>
</form>
```

- link – prepara o link com o endereço necessário para acessar a página escolhida sem perder as variáveis da sessão.

A figura abaixo mostra o exemplo de um template, com a lista das pessoas, a paginação e um campo para busca:

The screenshot shows the website interface for 'Museu da Pessoa Brasil'. At the top, there is a logo with three interlocking circles and the text 'Museu da Pessoa Brasil'. Navigation links include 'O QUE É O MUSEU', 'HISTÓRIAS', and 'FAÇA PARTE'. A search bar is present with the text 'busca' and an 'ok' button. In the top right corner, there are links for 'contato', 'mapa', and 'english'.

The main content area is titled 'Depoentes' and includes a breadcrumb trail: 'Home > Histórias > Depoentes'. On the left, a red sidebar contains a menu with the following items: 'DEPOENTES', 'GRANDES TEMAS', 'ESCOLAS, INSTITUIÇÕES E COMUNIDADES', 'MEMÓRIA DOS BRASILEIROS', 'FOTOGRAFIAS', 'RÁDIO MUSEU DA PESSOA', 'ONTEM E HOJE', and 'BUSCAS'.

The search results section is titled 'CONSULTA POR LETRA:' and lists the alphabet 'A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Todos'. Below this is a 'BUSCA POR NOME:' section with a search input field, an 'ok' button, and a link '» Mais opções de Busca'. The search results are displayed in a list with a pagination bar at the top showing '1 2 3 4 5 >'. The names listed are: Abadio Alves de Lima Junior, Abdalla Abel Mohamed Ali Osman, Abdenio Santos Rufino, Abdon Baptista Filho, Abdu Kexfe, Abilio da Silva Guimaraes, Abilio Gonçalves Junior, Abimael Coelho da Cruz, Abrahão Joseph Epelboim, and Abram Abe Szajman.

Figura 24 – Exemplo de um template

Com o template pronto, ele deve ser compactado em um arquivo ZIP e enviado pelo sistema. O endereço de todas as páginas são iguais, somente alterando o identificador do projeto. Ele abre o “index.jsp” do diretório do template. O endereço padrão é: <http://www.museudapessoa.net/museu/indexProjeto.do?idProjeto=123>.

A página de cadastro é simples e ela permite cadastrar indexadores do acervo específicos do projeto. Pode-se também fazer o download do arquivo ZIP, como mostra a figura 25:

TEMPLATES DA HOME DE PESSOAS

Arquivo ZIP *

Arquivo...

Comentário da revisão *

Limpar Formulário Cadastrar Revisão

Versão Atual

DATA	COMENTÁRIOS	VERSÃO	AÇÕES
27/09/2007	1a versão	1	

Versões Anteriores

DATA	COMENTÁRIOS	VERSÃO	AÇÕES
Nenhum registro encontrado.			

INDEXADORES DO ACERVO

CADASTRO >> NOME

salvar

Nome

título 1		
título 2		

Cancelar

Figura 25 – Cadastro de template do projeto

7.6.17 MODELO PADRÃO DO ACERVO

Essa função define o template padrão do portal, onde todo o conteúdo do sistema é disponibilizado. Nenhum projeto é relacionado e o endereço é: <http://www.museudapessoa.net/museu/indexProjeto.do>.

7.6.18 CADASTRO DO ACERVO

Tanto a pessoa quanto o projeto possuem imagens e audiovisuais. A lista dos dois estão disponíveis em uma única página para melhorar a usabilidade, nela pode-se realizar buscas por: última alteração, tipo, autor e qualquer palavra. Já a ficha de cadastro é diferente com os campos necessários de cada. Só é permitido o envio de arquivos com extensão compatíveis e de tamanho menor a 3 megas. Todos eles são armazenados no computador onde está rodando o sistema, sendo organizados por diretórios com o identificador do proprietário como nome.

- imagem – é do tipo foto, documento, desenho ou objeto. Ela pode ser definida como retrato da pessoa, caso seja inserida por uma. Pode-se classificar uma imagem com diversas palavras-chaves cadastradas previamente no sistema.
- audiovisual – é um áudio ou vídeo e também pode ser classificada.

A imagem abaixo mostra a página da lista do acervo de uma pessoa:

cadastro de pessoa | histórias | **acervo** | relacionamentos

Acervo do Projeto

última alteração

qualquer palavra



autor

tipo

foto documento objeto

desenho vídeo áudio

Imagens

MINIATURA	ID / TÍTULO	TIPO	CÓDIGO	ÚLTIMA ALTERAÇÃO	AÇÕES
	[16636] [Eu bebe fazendo 2 anos]	foto		25/05/2007	<input type="button" value="✎"/> <input type="button" value="🗑"/>
	[16633] [Lindo]	foto		25/05/2007	<input type="button" value="✎"/> <input type="button" value="🗑"/>

Audiovisual

ID / TÍTULO	TIPO	CÓDIGO	ÚLTIMA ALTERAÇÃO	AÇÕES
[1374] [19 de abril dia do Índio.]	video		25/05/2007	<input type="button" value="✎"/> <input type="button" value="🗑"/>

Figura 26 – Lista do acervo

7.6.19 BUSCAS AVANÇADAS

Caso as buscas nas páginas não sejam suficientes para chegar ao resultado esperado, foi desenvolvida uma seção de buscas avançadas com mais opções de filtros. A busca por acervo utiliza os classificadores e os indexadores para ajudar na consulta. Logo em seguida a imagem da tela da busca avançada por pessoa:

Busca por Pessoa

Nome

expressão exata qualquer palavra

Buscar somente pessoas que contenham
 histórias imagens audiovisuais

Data/Ano Nascimento Período Nascimento Projeto

Cor / Raça Sexo Masculino Feminino Ambos

Religião Situação Migratória Migrante Imigrante

Cidade de Nascimento UF País de Nascimento

Cidade de Residência UF País de Residência

Escolaridade País de Nascimento dos Pais

Figura 27 – Busca avançada por pessoa

7.6.20 CADASTRO DE TEMPLATES DO PUBLICADOR

Para complementar as páginas dinâmicas das homes de pessoa dos projetos foi implementado uma ferramenta para criar templates para páginas de

conteúdo estático. A finalidade do template é servir como molde para a inserção de conteúdo que não pertence ao acervo em páginas HTML.

Um template pode ou não ter como referência um projeto. Sua estrutura é dividida em dois blocos: as chamadas que contêm os campos e o seu código HTML. O primeiro passo é cadastrar as chamadas com seus campos, que podem ser do tipo texto grande, texto pequeno e upload de imagem. Cada campo possui um nome e uma marcação, que é numérica. A página para realizar essas funções é a mesma, o AJAX é utilizado para ela não ser atualizada a cada inserção, melhorando a usabilidade. A seguir a tela da lista de chamadas de um template:

The screenshot displays a web interface for managing template calls. At the top, there are three tabs: 'cadastro de template', 'chamadas', and 'html'. Below the tabs is a red header with the text 'Cadastro de Chamada'. Underneath, there is a yellow bar with 'CADASTRO >> NOME DA CHAMADA' on the left and 'AÇÕES' on the right. A white text input field is present, followed by a 'salvar' button. Below this is a table with the following data:

Chamada			AÇÕES	
Chamada 01				
↳ Texto Pequeno	1			
↳ Foto	2			
↳ Texto Grande	3			

At the bottom of the interface, there are three buttons: 'Adicionar Campos', 'Editar', and 'Apagar'.

Figura 28 – Lista de chamadas do template

O segundo passo depois que todas as chamadas e campos forem cadastrados é criar um código HTML onde esses campos são distribuídos através de suas marcações. Os valores das marcações precisam estar dentro de `{}` para serem reconhecidas. Abaixo a figura da página para inserir o código HTML com suas marcações listadas à esquerda:

The screenshot shows a web application interface for managing templates. At the top, there are three tabs: 'cadastro de template', 'chamadas', and 'html', with 'html' being the active tab. Below the tabs is a red header bar with the text 'Cadastro de Template'. The main content area is divided into two sections: 'Chamadas' on the left and 'Código HTML da página' on the right.

Chamadas

Chamada 01	
↳ Texto Pequeno	[1][+]
↳ Texto Grande	[3][+]
↳ Foto	[2][+]

Código HTML da página

```
<html>
<head>
<title>Template Exemplo</title>
</head>
<table align="center" border="0" width="100%">
  <tr>
    <td>Texto Pequeno:</td><td><b> ${1} </b></td>
  </tr>
  <tr>
    <td>Foto:</td><td>  </td>
  </tr>
  <tr>
    <td>Texto Grande: </td><td> ${3}</td>
  </tr>
</table>
</html>
```

At the bottom right of the interface, there are two buttons: 'Cancelar' and 'Salvar'.

Figura 29 – Código HTML do template

7.6.21 CADASTRO DE PÁGINAS DO PUBLICADOR

A partir dos templates do publicador pode-se criar páginas de conteúdo estático relacionadas ou não à um projeto. Nessas páginas são inseridos os valores nos campos das chamadas do template. Depois de salvar o formulário é necessário publicar a página para gerar um arquivo HTML com o código do template trocando as marcações pelos valores da página. O endereço é padrão mudando somente o identificador de cada: www.museudapessoa.net/museu/museu_arquivos/publicador/valorPublicador/3/index.html.

Somente as páginas publicadas podem ser visualizadas, quando elas são retiradas seus arquivos HTML são apagados. Essa ferramenta facilita a administração das páginas estáticas do portal, permitindo rapidamente alterar seus conteúdos. A vantagem de usar o template é eliminar a necessidade de reescrever o

código HTML para cada página criada. A figura 30 mostra um exemplo de uma página cadastrada:

Figura 30 – Cadastro de página

O resultado dos valores acima com o template escolhido é a página:

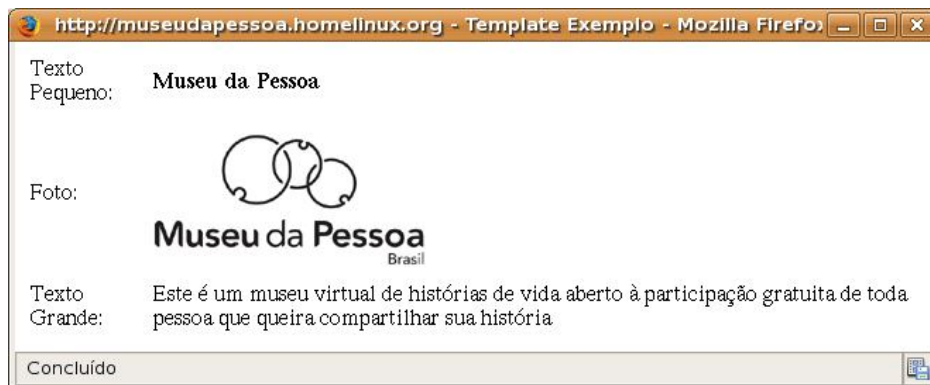


Figura 31 – Preview da página

7.7 AJUSTES E CORREÇÕES DE BUGS

Depois da fase de implementação das funcionalidades do portal, iniciou-se os ajustes e correções de bugs. Mesmo com os testes dos códigos escritos sempre permanecem alguns erros. Para localizar os erros e corrigi-los, foi organizado pela equipe uma força tarefa, para testar exaustivamente o funcionamento do sistema, inserindo dados fictícios para simular o comportamento em produção. O sistema foi instalado em um servidor, permitindo assim que o cliente participasse dessa etapa de testes. Com isso, o Museu avaliou o sistema e requisitou alguns ajustes e também reportou erros.

A fase de correção e ajustes é lenta e muito trabalhosa, pois há problemas que são difíceis de resolver. Todas as funções devem ser testadas de todas as maneiras possíveis, é fundamental ter uma visão mais aberta do sistema e prever todos os cenários existentes. Quanto menor a quantidade de bugs maior será robustez e a integridade dos dados do portal. Os erros encontrados foram listados e delegados entre os programadores dando prioridade aos mais importantes. Todos os ajustes que não estavam fora do escopo inicial do projeto foram implementados para atender as necessidades do Museu.

7.8 MIGRAÇÃO DO BANCO DE DADOS

A migração do conteúdo do banco de dados do portal antigo para o novo é extremamente delicado e é essencial para não perder todo o trabalho já realizado pelo Museu da Pessoa. O maior problema foi com a diferença da arquitetura dos dois sistemas, que inviabilizava a transferência direta do conteúdo entre os bancos de dados. A solução foi estudar a estrutura do banco de dados do portal antigo e analisar todas as tabelas e seus conteúdos para verificar o que poderia ser aproveitado. Foi feito uma cópia do banco de dados do portal antigo, que é ainda utilizado, e seu conteúdo foi mapeado para possibilitar a migração dos dados para a nova arquitetura. O que dificultou o reconhecimento dos campos foi a falta da

documentação do sistema antigo. A importância de um estudo detalhado garante maior integridade dos dados migrados.

O objetivo da migração foi transferir o máximo de conteúdo do portal antigo para o novo, o processo de transferência teve que ser implementado manualmente. Ela é realizada por uma sequência de etapas, que devem ser seguidas em uma ordem determinada para inserir os dados da maneira correta.

Desenvolver o migrador seria menos trabalhoso se fosse possível utilizar as funções implementadas do sistema para fazer consultas e escritas no banco de dados. Porém os testes demonstraram que o tempo para executá-las é alto, tornando inviável a transferência de tabelas com muitos dados. Levando em conta que foram aproximadamente mais de 16 mil registros do banco de dados migradas só de informações de pessoas, acervos e projetos, foi preciso uma outra estratégia para aumentar a performance.

A solução foi usar uma API Java para acessar e processar dados armazenados em bancos de dados relacionais usando a linguagem SQL Query. A vantagem desse método é a velocidade, mas a desvantagem é sua implementação, que é muito mais trabalhosa. Cada passo da migração segue uma lógica básica. Primeiro uma consulta é feita ao banco de dados antigo para retornar a lista de dados da tabela escolhida, depois a lista é percorrida para guardar em variáveis os valores das colunas de cada linha. Por último, as variáveis passam por validações e são estruturadas para serem inseridas no novo banco de dados.

Depois de algumas funções prontas, viu-se necessário a criação de métodos para automatizar as consultas e escritas usando a linguagem SQL. A idéia foi criar um objeto Campo com atributos de nome, valor e tipo. Assim, todos os procedimentos SQL foram montados baseados nessa estrutura. Na inserção de dados, para cada coluna de uma linha da lista percorrida dos dados de uma tabela, será instanciado um objeto Campo com o nome da coluna, valor e o tipo (inteiro, texto, booleano). Os campos são armazenados em uma lista e a partir dela são montadas os códigos para inserir-lhes em uma única linha na tabela determinada. Houve a preocupação em não alterar os valores das chaves primárias para manter a integridade dos relacionamentos das tabelas. Todos os arquivos do acervo foram

copiados e adequados ao novo sistema. Segue abaixo a parte em que o método de inserção monta a SQL Query através da lista de campos e do nome da tabela a ser inserida.

```
public boolean insert(ArrayList<Campo> listaCampos, String tabela) {

    String meio = "(";
    String fim = "(";

    Campo campoAux = listaCampos.remove(0);

    meio = meio + campoAux.getChave();
    fim = fim + campoAux.getValor();

    for (Campo campo : listaCampos) {
        meio = meio + "," + campo.getChave();
        fim = fim + "," + campo.getValor();
    }

    meio = meio + ")";
    fim = fim + ")";

    String sql = "INSERT INTO "+ tabela + " " + meio + " VALUES " + fim;
```

A migração é feita em 24 passos com mais de 2240 linhas de código. Sua duração é aproximadamente 14 horas e foram transferidos 7219 usuários, 8300 pessoas, 140 projetos, 10130 acervos entre outros. O resultado foi o esperado, o novo portal funcionando com todo o conteúdo já coletado pelo Museu da Pessoa.

A possibilidade de testar o sistema com o conteúdo atual do Museu é única. Simula da maneira mais real o ambiente de produção que o portal teria em alguns anos de funcionamento. Foram realizados testes de performance e usabilidade, além de uma nova varredura por erros.

7.9 MIGRAÇÃO DAS SALAS

As salas do Museu é um termo usado para se referir as home de pessoas de alguns projetos. Elas servem para criar uma identidade visual própria para o projeto, onde ele pode disponibilizar o seu acervo para o público. Atualmente

existem algumas em funcionamento e suas criações dependeram de alterações no código fonte do sistema. As salas e seus endereços são:

- Home Geral – local onde encontra-se a lista completa de todos os depoentes que têm suas histórias publicadas no portal Museu da Pessoa (www.museudapessoa.net/MuseuVirtual/hmdepoente/pessoas.jsp).
- Ashoka – espaço onde as organizações Ashoka Empreendedores Sociais fazem sua campanha Todo Mundo Pode Mudar o Mundo, em parceria com o Museu da Pessoa (www.museudapessoa.net/ashoka).
- Musée de La Personne – o portal do núcleo canadense usa o sistema do Brasil para gerenciar o seu acervo (www.museudapessoa.net/mdlp).
- Memórias do Comércio na Cidade do Rio de Janeiro – o projeto é uma iniciativa do SESC Rio desenvolvida pelo Museu da Pessoa para contar relatos de comerciantes e comerciários sobre como chegaram ao comércio, ou como convivem com clientes, patrões e empregados (www.museudapessoa.net/sescrj).
- Clube da Esquina – surgiu para ser o espaço de reunião da história viva desse movimento contada por seus protagonistas e pelo público que vivenciou ou vivencia o sucesso e o legado cultural do Clube (www.museudapessoa.net/clube).
- Um Milhão de Histórias de Vida de Jovens – espaço criado para valorizar a juventude como sujeito na construção da história do país (www.museudapessoa.net/ummilhao).
- Histórias da Nossa Terra - é um projeto que resgata a história de municípios através da técnica de memória oral, refletindo a lembrança dos moradores sobre a própria cidade, suas famílias, infância, trabalho, credo, tradições e origens (www.museudapessoa.net/ctbc).

O novo portal permite enviar pacotes de templates para criar páginas com acesso ao acervo do sistema e ferramentas para criar páginas com conteúdos estáticos, sem precisar mexer no código fonte.


Essa etapa deu-se por migrar as salas existentes, criando um pacote de template para cada. O objetivo é manter o layout das páginas iguais e nas partes onde os dados são dinâmicos, utilizar as taglibs desenvolvidas para acessar o banco de dados do portal. A migração foi trabalhosa, mas útil para testar o funcionamento das tags. Durante o processo verificou-se que elas não atendiam à todos os cenários que as salas apresentam, o que é um problema, pois as tags devem fornecer todos os métodos necessários para gerenciar o acervo do portal. Essa eventualidade atrasou o projeto, foi preciso implementar muito mais código de taglibs para satisfazer todos os requisitos, além dos pacotes de templates. Algumas mudanças são:

- iterate pessoa - só permitia listar as pessoas de um projeto, agora ele lista todas as pessoas de todos os projetos relacionados à uma matriz, e também lista todas as pessoas que um usuário é responsável.
- iterate projeto - podem ser listados os projetos de uma matriz.
- iterate acervo – antes o acervo era listado somente pelo seu proprietário, agora é possível listar o acervo de um projeto e todas as pessoas relacionadas, e também retornar o acervo de um projeto matriz, seus projetos e pessoas relacionadas. Pode-se escolher o tipo do acervo a ser retornado.
- paginador – teve que ser adequado para paginar as novas possibilidades de listas.
- recomendar – envia um link do endereço do portal para os e-mails escolhidos pelo internauta.
- comentário – permite adicionar comentários à um acervo.
- fale conosco – envia um e-mail para o Museu com alguma mensagem do internauta.

- acesso do internauta – foi implementado algumas ações que o internauta pode fazer nas salas do Museu. Cadastrar uma nova pessoa e gerenciar seu acervo.

Os pacotes de templates contêm o necessário para estruturar uma página na internet, basicamente arquivos de JSP, javascripts, css e as imagens. As salas seguem um padrão básico de distribuição do conteúdo. Normalmente elas apresentam a lista das pessoas pertencentes ao projeto, que sucessivamente leva para a página da pessoa onde pode ser visualizado o seu acervo. Existe a opção de realizar buscas pelas pessoas, projetos ou acervo. As salas possuem uma área onde o internauta pode preencher uma ficha de cadastro de pessoa com um login e senha para acessar algumas funcionalidades do portal. O acesso permite ao usuário gerenciar através da interface definida pelo template todas as pessoas que ele é responsável.

O resultado da migração das salas para os templates foi um sucesso, o que comprovou a eficiência das taglibs desenvolvidas. Com certeza após um bom treinamento, a equipe do Museu poderá fazer facilmente seus próprios pacotes de templates. Abaixo imagens da Home Geral após a migração usando o conteúdo também migrado do banco de dados.



[contato](#) • [mapa](#) • [english](#)

Museu da Pessoa
Brasil

O QUE É O MUSEU | HISTÓRIAS | FAÇA PARTE

busca

Home > Histórias > Depoentes

Depoentes

DEPOENTES

LISTA DE SITES

ESCOLAS, INSTITUIÇÕES E COMUNIDADES

MEMÓRIA DOS BRASILEIROS

FOTOGRAFIAS

RÁDIO MUSEU DA PESSOA

ONTEM E HOJE

BUSCAS

CONSULTA POR LETRA:
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

BUSCA POR NOME:

» Mais opções de Busca

Página 1 de 25 1 2 3 4 5 6 7 8 ▶▶

A 3ª série da Professora Giselle Freitas Xavier da EMEIEF. "Helena Pereira de Moraes"

Abdalla Abel Mohamed Ali Osman

Abdon Baptista Filho

abelha

Abrahão Joseph Epelboim

Abram Abe Szajman

Abram Szajman

Acácio Tavares Ferreira Marques

Adair Antônio de Freitas Meira

Adair Emanuel Caon Pieruccini

Adair Felizardo

Adalton Caldeira Fonseca

Adam Edward Drozdowicz

Adauto Borsatti Felix, o Coxinha

Adauto Tadeu Basílio

Página 1 de 25 1 2 3 4 5 6 7 8 ▶▶

Figura 32 – Lista de pessoas da Home Geral

Quando o nome é clicado a página da pessoa é aberta disponibilizando sua história paginada e indexada além de outras informações:



ABRAHÃO JOSEPH EPELBOIM

Nascimento: 17/06/1935,
Rio de Janeiro

Profissão: Comerciante

Projeto: Memórias do Comércio na
Cidade do Rio de Janeiro

Abrahão Joseph Epelboim, também conhecido como seu Adolfo, nasceu no Rio de Janeiro, em 1935. Filho de imigrantes judeus, foi criado no Engenho de Dentro, onde morou até se casar. Começou sua vida profissional ajudando o pai, vendedor prestamista. Trabalhou na extinta Mesbla, em várias lojas de móveis e está há 34 anos na Icarai Móveis, na Rua do Catete.

FOTOS •

Índice da história:

Selecionar

Identificação

Meu nome é Abrahão Joseph Epelboim, nasci em 17 de junho de 1935, na cidade do Rio de Janeiro. Meus pais se chamam Saul Epelboim e Hana (ou Ana) Epelboim. Tenho uma irmã e seu nome aporuguesado – apesar de não se traduzir nomes – é Rosa Epelboim.

Meus pais chegaram ao Brasil por volta de 1927, fugindo do país de origem. Papai nasceu numa aldeia próxima a Bucareste, na Bessarábia (que significa "alem da Arábia"), e mamãe próximo a um lugar sob domínio romeno, também na Bessarábia.

Infelizmente não conheci meus avós porque, sendo de origem israelita, meus pais eram imigrantes fugitivos. Apenas tenho conhecimento dos nomes deles.

Refúgio no Brasil

Meus pais vieram morar no Rio porque receberam uma "carta de chamado" de um tio por parte de mãe, do qual eu levo o nome como homenagem. Nós, de origem judaica, sempre damos o nome a um novo membro da família em homenagem a um ente falecido. Joseph nada mais é do que um nome de origem russa.

Esse tio faleceu por causas desconhecidas logo depois. Não tinham conhecimento disso, pois ele se ausentou do Brasil, refugiando-se em Montevidéu, e perderam o contato com ele. As famílias de meus pais eram grandes, e durante a Segunda Guerra Mundial foram dizimadas, tanto por parte de papai como de mamãe.

Página 1 de 5 1 2 3 4 5 ▶▶

✉ Índice de Depoentes

✉ Enviar por e-mail

📄 Versões para impressão:

desta página

do depoimento na íntegra

Figura 33 – Página da pessoa da Home Geral

7.10 ETAPA FINAL

A etapa final consiste em fazer mais uma varredura no sistema para certificar seu bom funcionamento, e somente depois instalá-lo no servidor de produção. Depois, migrar o conteúdo do banco de dados e enviar os templates das salas. Iniciando assim as operações do novo portal e desligando o antigo.

8 PROJETOS FUTUROS

A atual arquitetura de conteúdo do Portal Museu da Pessoa.Net, que estabelece o núcleo do acervo sobre a pessoa, permitindo relacionamentos com projetos institucionais, não atende ao modelo de metodologia utilizado nos projetos de memória desenvolvidos em escolas pelo Museu. Nesse novo paradigma, o centro do acervo é a escola, e suas turmas, surgindo ainda a figura do professor e do aluno, que desempenham papéis diferentes dos perfis atualmente atendidos no portal Museu da Pessoa. Com isso, surgiu o projeto Memória Local, que pretende gerenciar o acervo dos projetos comunitários e educativos desenvolvidos pelo Museu em escolas e comunidades, usando o mesmo sistema implementado pro portal. Sua interface será simples e amigável para as crianças. A meta é finalizar o portal Museu da Pessoa e iniciar o Memória Local.

Uma boa idéia para outro projeto é utilizar o portal Museu da Pessoa para contar a vida das pessoas que participaram de alguma maneira da história da UFSC. O curso de Jornalismo poderia ser responsável pela coleta e organização dos dados, e alunos ligados à webdesign ficariam encarregados da manutenção das páginas do projeto e seu template. Com certeza o resultado desse projeto será um presente à Universidade.

9 CONSIDERAÇÕES FINAIS

Após aproximadamente um ano de trabalho, é com muita satisfação que a equipe chega ao final do projeto alcançando todos os objetivos propostos na sua concepção. O estudo das tecnologias, o gerenciamento adequado das atividades, a ajuda e seriedade da equipe foram responsáveis pelo bom resultado.

O framework utilizado foi fundamental para o desenvolvimento do projeto. As tecnologias envolvidas para a sua construção são as melhores em softwares livres, relacionados a linguagem de programação Java. Os conceitos de funções genéricas, reutilização de código e scripts para automatizar a criação de arquivos do sistema foram fundamentais para aumentar a produtividade.

A nova ergonomia do portal melhorou consideravelmente a sua usabilidade, o menu principal dá acesso rápido à todas as funcionalidades disponíveis. O cadastro de perfil é flexível e possibilita dar permissões personalizadas aos usuários.

O novo contexto para a estrutura de conteúdo foi implementado de acordo com os requisitos. O projeto deixou de ser a célula matriz de todo o acervo, e a pessoa se tornou o foco principal do sistema. A sessão de direito é assinada uma única vez e vale para todo o acervo. As histórias podem ser escritas usando as marcações em XML que transformam seus conteúdos em formato HTML. Existem vários tipos de buscas que permitem encontrar de tudo no banco de dados.

A facilidade de gerenciar as páginas estáticas e dinâmicas com as ferramentas disponibilizadas garante a autonomia que o Museu da Pessoa precisava para a criação e manutenção das salas. As taglibs desenvolvidas são poderosas, pois permitem aos webdesigners do Museu efetuarem consultas e escritas no banco de dados do sistema diretamente pelas tags inseridas nos arquivos JSP dos templates. As páginas estáticas têm seus conteúdos facilmente alterados por uma interface simples de edição.

Um projeto de grande porte como esse não escapa de erros. Após a fase de implementação foi feito buscas por bugs em todo o portal. A etapa de correções de bugs e ajustes é perigosa, pois o tempo para corrigí-los variam indefinidamente.

Com isso, o projeto sofreu um pequeno atraso não previsto, todos os erros reportados foram arrumados. Os ajustes foram mínimos, a maioria de layout, nenhum fora do escopo.

Com o sistema estável foi possível iniciar o processo de migração do conteúdo do portal antigo para a nova estrutura. O bom resultado da migração garantiu que todo o trabalho já realizado pelo Museu fosse aproveitado no novo portal. Além disso, permitiu testar o sistema com o conteúdo real, que provou ter uma boa performance para banco de dados com muitos registros.

A migração das salas do portal foi uma fase importante para mostrar que as taglibs feitas não eram suficientes para atender à todos os cenários apresentados por elas. O resultado da migração foi a implementação de um número maior de tags que atendem as necessidades das salas. Isso garante que os templates possam ser desenvolvidos utilizando a tecnologia das tags para acesso ao banco de dados.

O Instituto Museu da Pessoa é uma organização sem fins lucrativos, que há 16 anos coleta histórias de vida das pessoas. Sua missão é realizar a construção de uma identidade cultural da sociedade. A substituição do portal trará muitos benefícios para a instituição, as novas ferramentas aumentarão a produtividade do Museu e darão total autonomia para o desenvolvimento de seus projetos.

São muitas as possibilidades de projetos possíveis de desenvolver utilizando as novas ferramentas. O projeto de Memória Local irá complementar sistema para atender as escolas e comunidades. O portal servirá a população brasileira, possibilitando a todas as pessoas o direito de contar a sua história, sem exclusão de raça ou classe social.

A finalidade do portal Museu da Pessoa é educativa. Existe a necessidade de um apoio geral das instituições educacionais do Brasil. A criação de novos agentes formadores para continuar a ideologia do Instituto e maior divulgação que é fundamental para seu crescimento.

10 REFERÊNCIAS

ALMEIDA, Rodrigo Rebouças de. **Model View Controller**. Disponível em: <<http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/arqu/mvc/mvc.htm>>. Acesso em: 12 set. 2007.

ASHOKA EMPREENDEDORES SOCIAIS. **Fique por Dentro de Quem está Mudando o Mundo**. Disponível em: <<http://www.ashoka.org.br/fellow.php?acao=visualizar&id=854>>. Acesso em: 30 ago. 2007.

BAUER, Christian; KING, Gavin. **Hibernate em Ação**. Rio de Janeiro: Ciência Moderna, 2005.

BOND, Martin et al. **Aprenda J2EE: Com EJB, JSP, Servlets, JNDI, JDBC e XML**. São Paulo: Pearson Education do Brasil, 2003.

CARDOSO, Michael. **Portal Museu da Pessoa: Visão Geral**. Florianópolis: 2006.

D'ÁVILA, Márcio. **Tutorial Tomcat: Instalação e Configuração Básica**. Disponível em: <<http://www.mhavila.com.br/topicos/java/tomcat.html#intro>>. Acesso em: 22 set. 2007.

FEDELI, Ricardo; POLLONI, Enrico; PERES, Fernando. **Introdução à Ciência da Computação**. São Paulo: Pioneira Thomsom Learning, 2003.

GOSPODNETIC, Otis; HATCHER, Eric. **Lucene in Action**. Connecticut: Manning, 2005.

GUEDES, Gilleanes. **UML: Uma Abordagem Prática**. São Paulo: Novatec, 2004.

HUSTED, Ted N. et al. **Struts in Action**. Connecticut: Manning, 2002.

MAANEN, John, Van. **Reclaiming qualitative methods for organizational research: a preface**, In *Administrative Science Quarterly*, vol. 24, no. 4, Dezembro 1979 a, p 520-526.

MINAYO, M.C.S. e SANCHES, O. (1983) Quantitativo-Qualitativo: oposição ou complementaridade. *Cadernos de Saúde Pública*. Rio de Janeiro, v.9, n.3, pp.239-262.

MUSEU DA PESSOA. **Conte Sua História.** Disponível em:

<http://www.museudapessoa.net/oquee/oque_programas_conte.shtml>. Acesso em: 30 ago. 2007.

MUSEU DA PESSOA. **Conte Sua História: Projetos.** Disponível em:

<http://www.museudapessoa.net/oquee/oque_programas_conte_projetos.shtml>. Acesso em: 04 set. 2007.

MUSEU DA PESSOA. **Formação.** Disponível em:

<http://www.museudapessoa.net/oquee/oque_programas_formacao.shtml>. Acesso em: 06 set. 2007.

MUSEU DA PESSOA. **Formação: Projetos.** Disponível em:

<http://www.museudapessoa.net/oquee/oque_programas_formacao_projetos.shtml>. Acesso em: 06 set. 2007.

MUSEU DA PESSOA. **Memória Institucional.** Disponível em:

<http://www.museudapessoa.net/oquee/oque_programas_memoria.shtml>. Acesso em: 04 set. 2007.

MUSEU DA PESSOA. **Nossa História.** Disponível em:

<http://www.museudapessoa.net/oquee/oque_nossahistoria.shtml>. Acesso em: 30 ago. 2007.

PAULILO, Maria Ângela Silveira. **A Pesquisa Qualitativa e a História de Vida.**

Disponível em: <http://www.ssrevista.uel.br/c_v2n1_pesquisa.htm>. Acesso em: 20 ago. 2007.

POSTGRESQL. **About.** Disponível em: <<http://www.postgresql.org/about/>>. Acesso em: 25 set. 2007.

POSTGRESQLBR. **Introdução e Histórico.** Disponível em:

<<http://www.mhavila.com.br/topicos/java/tomcat.html#intro>>. Acesso em: 25 set. 2007.

THE APACHE SOFTWARE FOUNDATION. **Apache Tomcat.** Disponível em:

<<http://tomcat.apache.org/>>. Acesso em: 22 set. 2007.

UN DEVELOPER NETWORK. **Tag Libraries**. Disponível em:

<<http://java.sun.com/products/jsp/taglibraries/index.jsp#tutorials>>. Acesso em: 25 set. 2007.

WIKIPÉDIA. **Ajax**. Disponível em: <[http://en.wikipedia.org/wiki/Ajax_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming))>. Acesso em: 15 set. 2007.

WIKIPÉDIA. **ANT**. Disponível em: <http://en.wikipedia.org/wiki/Apache_Ant>. Acesso em: 15 set. 2007.

WIKIPÉDIA. **CVS**. Disponível em: <<http://en.wikipedia.org/wiki/ CVS>>. Acesso em: 15 set. 2007.

WIKIPÉDIA. **MVC**. Disponível em: <<http://pt.wikipedia.org/wiki/MVC>>. Acesso em: 10 set. 2007.

11 ANEXOS

11.1 CÓDIGO FONTE

11.1.1 GENÉRICO.JAR

GenericBusinessImpl.java

```
import java.beans.Introspector;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import br.com.aragao.resources.Result;

public class GenericBusinessImpl<T> implements GenericBusiness<T> {

    protected GenericDAO<T> dao = null;

    public GenericDAO<T> getDao() {
        return dao;
    }

    public void setDao(GenericDAO<T> dao) {
        this.dao = dao;
    }

    public List<T> listar() {
        return listar(null, 0, 0);
    }

    public List<T> listar(String filtro) {
        return listar(filtro, 0, 0);
    }

    public List<T> listar(String filtro, int inicio, int qtd, String...campos) {
        ArrayList<Filtro> filtros = new ArrayList<Filtro>();
        String[] ordem = null;
        if (campos != null && campos.length > 1) {
            ordem = new String[campos.length];
            setupFiltros(filtro, campos, ordem, filtros);
        }
    }
}
```

```

        return getDao().listar(filtros, inicio, qtd, ordem);
    }

    public Result<T> filtrar(HashMap<String, String> filtro, int inicio, int qtd,
String...ordem) {
        ArrayList<Filtro> filtros = new ArrayList<Filtro>();

        if (filtro != null) {
            for (String campo : filtro.keySet()) {
                Filtro f = getFiltro(campo, filtro.get(campo));
                f.setCondicao("and");
                filtros.add(f);
            }
        }

        List<T> list = getDao().listar(filtros, inicio, qtd, ordem);
        long total = getDao().getTotalRegistro(filtros);

        Result<T> res = new Result<T>(list, total);

        return res;
    }

    public Result<T> filtrar(ArrayList<Filtro> filtro, int inicio, int qtd, String...ordem)
{
        ArrayList<Filtro> filtros = new ArrayList<Filtro>();

        if (filtro != null) {
            for (Filtro f : filtro) {
                Filtro tmp = getFiltro(f.getCampo(), f.getOperacao(),
(String) f.getValor());
                tmp.setCondicao(f.getCondicao());
                filtros.add(tmp);
            }
        }

        List<T> list = getDao().listar(filtros, inicio, qtd, ordem);
        long total = getDao().getTotalRegistro(filtros);

        Result<T> res = new Result<T>(list, total);

        return res;
    }

    public int getTotalRegistro(String filtro, String...campos) {
        ArrayList<Filtro> filtros = new ArrayList<Filtro>();

```

```

        String[] ordem = null;
        if (campos != null && campos.length > 1) {
            ordem = new String[campos.length];
            setupFiltros(filtro, campos, ordem, filtros);
        }
        return getDao().getTotalRegistro(filtros);
    }

    public T editar(Long id) {
        return (T) getDao().editar(id);
    }

    public void salvar(T obj) {
        getDao().salvar(obj);
    }

    public void apagar(Long id) {
        apagar(editar(id));
    }

    public void apagar(T obj) {
        getDao().apagar(obj);
    }

    protected void setupFiltros(String filtro, String[] campos, String[] ordem,
        ArrayList<Filtro> filtros) {
        String[] fs = new String[1];
        fs[0] = filtro;
        setupFiltros(fs, campos, ordem, filtros, "or");
    }

    protected void setupFiltros(String[] filtro, String[] campos, String[] ordem,
        ArrayList<Filtro> filtros, String condicao) {
        ordem[0] = campos[0];
        int j = 0;
        for (int i = 1; i < campos.length; i++) {

            Filtro f = getFiltro(campos[i], filtro[j]);
            if (f != null) {
                if (filtro[j] != null && !filtro[j].equals("")) {
                    f.setCondicao(condicao);
                    filtros.add(f);
                }
                ordem[i] = f.getCampo();
            }
        }

        if (filtro.length > i) {
            j++;
        }
    }

```

```

    }
}

protected Filtro getFiltro(String campo, String valor) {
    return getFiltro(campo, "=", valor);
}

protected Filtro getFiltro(String campo, String operacao, String valor) {
    Filtro filtro = null;
    if (campo != null && campo.length() > 0) {
        char tipo = campo.charAt(0);
        String c = Introspector.decapitalize(campo.substring(1));
        if (c.indexOf('#') > 0) {
            c = c.substring(0, c.indexOf('#'));
        }

        if(tipo == 's') {
            filtro = new Filtro(c, "like", valor);
        } else if(tipo == 'd') {
            SimpleDateFormat fmt = null;
            if (valor != null && valor.matches("[0-3][0-9]/[01][0-2]/[12][0-9][0-9][0-9] [0-2][0-9]:[0-6][0-9]")) {
                fmt = new SimpleDateFormat("dd/MM/yyyy
                hh:mm");
            } else if (valor != null && valor.matches("[0-3][0-9]/[01][0-2]/[12][0-9][0-9][0-9]")) {
                fmt = new SimpleDateFormat("dd/MM/yyyy");
            } else if (valor != null && valor.matches("[0-2][0-9]:[0-6][0-9]")) {
                fmt = new SimpleDateFormat("hh:mm");
            }
            if (fmt != null) {
                try {
                    filtro = new Filtro(c, operacao,
                    fmt.parse(valor));
                } catch (ParseException e) {}
            }
        } else if(tipo == 'i') {
            Integer v = null;
            if (valor != null && valor.matches("[0-9]+")) {
                v = Integer.valueOf(valor);
            }
            filtro = new Filtro(c, operacao, v);
        } else if(tipo == 'l') {
            Long v = null;
            if (valor != null && valor.matches("[0-9]+")) {
                v = Long.valueOf(valor);
            }
        }
    }
}

```

```

        }
        filtro = new Filtro(c, operacao, v);
    } else if(tipo == 'r') {
        Double v = null;
        if (valor != null && valor.matches("[0-9]+")) {
            v = Double.valueOf(valor);
        }
        filtro = new Filtro(c, operacao, v);
    } else if(tipo == 'b') {
        if (valor != null && (valor.equals("true") ||
valor.equals("false"))) {
            filtro = new Filtro(c, operacao,
Boolean.valueOf(valor));
        }
    } else {
        filtro = new Filtro(c, operacao, valor);
    }
}
return filtro;
}
}

```

GenericDaoImpl.java

```

package br.com.aragao.generico;
import java.util.ArrayList;
import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Conjunction;
import org.hibernate.criterion.Criterion;
import org.hibernate.criterion.Disjunction;
import org.hibernate.criterion.Expression;
import org.hibernate.criterion.Junction;
import org.hibernate.criterion.Projections;
import org.hibernate.criterion.Property;
import org.hibernate.criterion.Restrictions;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;

import br.com.aragao.resources.NaoRemovivel;

public abstract class GenericDAOImpl<T> extends HibernateDaoSupport implements
GenericDAO<T> {

    protected abstract Class getClazz();

    public T editar(Long id) {

```

```

        Session session = getSession();
        return (T) session.get(getClazz(), id);
    }

    public int getTotalRegistro(ArrayList<Filtro> filtros) {
        Criteria crit = getCriteria(filtros, 0,
0).setProjection(Projections.rowCount());
        return ((Integer) crit.uniqueResult()).intValue();
    }

    public List<T> listar(ArrayList<Filtro> filtros, int inicio, int qtd,
String...camposOrdem) {
        Criteria crit = getCriteria(filtros, inicio, qtd, camposOrdem);
        return crit.list();
    }

    public void salvar(T obj) {
        Session session = getSession();
        session.saveOrUpdate(obj);
        session.flush();
    }

    public void apagar(Long id) {
        apagar(editar(id));
    }

    public void apagar(T obj) {
        Session session = getSession();
        session.delete(obj);
        session.flush();
    }

    public void evict(T obj) {
        Session session = getSession();
        session.evict(obj);
    }

    protected Criteria getCriteria(ArrayList<Filtro> filtros, int inicio, int qtd) {
        return getCriteria(filtros, inicio, qtd, null);
    }

    protected Criteria getCriteria(ArrayList<Filtro> filtros, int inicio, int qtd,
String...camposOrdem) {
        Session session = getSession();
        Criteria crit = session.createCriteria(getClazz());

        Class[] interfaces = getClazz().getInterfaces();
        for (Class i : interfaces) {

```

```

        if (i.equals(NaoRemovivel.class)) {
            filtros.add(new Filtro("removido", "=", false));
            break;
        }
    }

    crit.setFirstResult(inicio);
    if (qtd > 0) {
        crit.setMaxResults(qtd);
    }

    if (camposOrdem != null && camposOrdem.length > 0) {
        setupOrdem(crit, camposOrdem);
    }

    if (filtros.size() > 0) {
        Conjunction conj = Restrictions.conjunction();
        Disjunction dis = Restrictions.disjunction();
        String alias = "";
        for (Filtro filtro : filtros) {
            String campo = "";
            String[] campos = filtro.getCampo().split("\\.");
            if (campos.length == 2 && !campos[1].matches("id[A-Z]\\w+") && !campos[1].equals("id")) {
                if (!alias.equals(campos[0] + "0")) {
                    alias = campos[0] + "0";
                    crit = crit.createAlias(campos[0], alias);
                }
                campo = alias + "." + campos[1];
            } else if (campos.length > 2) {
                for (int i = 0; i < (campos.length - 1); i++) {
                    crit = crit.createCriteria(campos[i]);
                }
                campo = campos[(campos.length - 1)];
            } else {
                campo = filtro.getCampo();
            }

            Junction j = null;
            if (filtro.getCondicao().equals("or")) {
                j = dis;
            } else {
                j = conj;
            }

            Criterion c = null;
            if (filtro.getOperacao().equals("like") && filtro.getValor() !=
null && !filtro.getValor().equals("")) {

```


11.1.2 GUARDA.JAR

Sistema.java

```

package br.com.aragao.guarda;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import java.util.Collections;
import java.util.HashMap;
import java.util.Set;

import br.com.aragao.guarda.menu.Menu;
import br.com.aragao.guarda.menu.MenuBar;

public class Sistema implements Serializable {
    private static Sistema sistema;

    private String usuarioLogado = "USUARIO_LOGADO";
    private String paginaRedirecionamento = "index.jsp";
    private String paginaLogin = "login.jsp";
    private String[] paginasLiberadas;
    private String actionParam = "acao";
    private String loginAdmin = "admin@jexperts.com.br";
    private String senhaAdmin = "admin";
    private String papelGlobal = "GLOBAL_ADMINISTRADOR";
    private HashMap<String, Modulo> modulos = new HashMap<String,
Modulo>();
    private HashMap<String, Papel> papeis = new HashMap<String, Papel>();
    private MenuBar menuBar;

    private static final long serialVersionUID = 2L;

    private Sistema() {
    }

    public static Sistema getInstance() {
        if (sistema == null) {
            sistema = new Sistema();
        }
        return sistema;
    }

    public String getSenhaAdmin() {
        return senhaAdmin;
    }
}

```

```
public void setSenhaAdmin(String senhaAdmin) {
    this.senhaAdmin = senhaAdmin;
}

public String getUsuarioLogado() {
    return usuarioLogado;
}

public void setUsuarioLogado(String usuarioLogado) {
    this.usuarioLogado = usuarioLogado;
}

public String getPaginaRedirecionamento() {
    return paginaRedirecionamento;
}

public void setPaginaRedirecionamento(String paginaRedirecionamento) {
    this.paginaRedirecionamento = paginaRedirecionamento;
}

public String[] getPaginasLiberadas() {
    return paginasLiberadas;
}

public void setPaginasLiberadas(String[] paginasLiberadas) {
    this.paginasLiberadas = paginasLiberadas;
}

public String getActionParam() {
    return actionParam;
}

public void setActionParam(String actionParam) {
    this.actionParam = actionParam;
}

public String getPaginaLogin() {
    return paginaLogin;
}

public void setPaginaLogin(String paginaLogin) {
    this.paginaLogin = paginaLogin;
}

public String getLoginAdmin() {
    return loginAdmin;
}
```

```
public void setLoginAdmin(String loginAdmin) {
    this.loginAdmin = loginAdmin;
}

public String getPapelGlobal() {
    return papelGlobal;
}

public void setPapelGlobal(String papelGlobal) {
    this.papelGlobal = papelGlobal;
}

public HashMap<String, Papel> getPapeis() {
    return papeis;
}

public ArrayList<Papel> getPapeisVisiveis() {
    ArrayList<Papel> papeisVisiveis = new ArrayList<Papel>();

    for (Papel p : papeis.values()) {
        if (p.getVisivel()) {
            papeisVisiveis.add(p);
        }
    }
    Collections.sort(papeisVisiveis);

    return papeisVisiveis;
}

public Papel getPapel(String sis) {
    return (Papel) papeis.get(sis);
}

public void putPapel(String chave, Papel papel) {
    papeis.put(chave, papel);
}

public void putModulo(String key, Modulo value) {
    this.modulos.put(key, value);
}

public Modulo getModulo(String key) {
    return this.modulos.get(key);
}

public Modulo getModuloPorDiretorio(String dir) {
    Collection<Modulo> ms = modulos.values();
```

```

        for (Modulo m : ms) {
            if (m.getDiretorio().equals(dir)) {
                return m;
            }
        }
        return null;
    }

    public ArrayList<Modulo> getModulos() {
        ArrayList<Modulo> tmp = new ArrayList<Modulo>(modulos.values());
        Collections.sort(tmp);
        return tmp;
    }

    public ArrayList<Permissao> getPermissoes() {
        ArrayList<Permissao> perms = new ArrayList<Permissao>();
        for (Modulo tmp : modulos.values()) {
            perms.addAll(tmp.getPermissoes());
        }
        Collections.sort(perms);
        return perms;
    }

    public Permissao getPermissao(String modulo, String perm) {
        Permissao p = null;
        ArrayList<Permissao> lista = modulos.get(modulo).getPermissoes();
        if (lista != null) {
            for (Permissao tmp : lista) {
                if (tmp.getId().equals(perm)) {
                    p = tmp;
                    break;
                }
            }
        }
        return p;
    }

    public Permissao getPermissaoPorPapel(String papel) {
        String[] tmp = papel.split("_");

        return getPermissao(tmp[0], tmp[1] + (tmp.length > 2 ? "_" + tmp[2] :
""));
    }

    public boolean podeAcessar(Set<String> permissoes, String url) {
        String[] parte = (url.indexOf('/') == 0 ? url.substring(1) : url).split("/");

        String diretorio = null;

```

```

String action = null;
String actionParam = null;

int indice = 0;

if (parte.length == 2) {
    diretorio = parte[0];
    indice = 1;
}

int questionMark = parte[indice].indexOf('?');
if (questionMark > 0) {
    action = parte[indice].substring(0, questionMark);
} else {
    action = parte[indice];
}

int equals = parte[indice].indexOf('=');
if (equals > 0) {
    actionParam = parte[indice].substring(equals + 1);
} else {
    actionParam = "listar";
}

int ampersand = actionParam.indexOf('&');
if (ampersand >= 0) {
    actionParam = actionParam.substring(0, ampersand);
}

if (action.equals(paginaLogin) || action.equals(paginaRedirecionamento)
|| (paginasLiberadas != null && Arrays.binarySearch(paginasLiberadas, "/" +
(diretorio != null ? diretorio + "/" : "") + action) >= 0)) {
    return true;
}

return podeAcessar(permissoes, diretorio, action, actionParam);
}

public boolean podeAcessar(Set<String> permissoes, String diretorio, String
action, String actionParam) {
    for (String p : permissoes) {
        String[] tmp = p.split("_");
        Modulo m = getModulo(tmp[0]);
        Permissao permissao = getPermissao(tmp[0], tmp[1] +
(tmp.length > 2 ? "_" + tmp[2] : ""));
        if (m != null && permissao != null && ((m.getDiretorio().equals("**")
&& permissao.getAction().equals("**") && permissao.getActionParam().equals("**")) ||

```

```

(m.getDiretorio().equals(diretorio) && permissao.getAction().equals(action) &&
permissao.getActionParam().equals(actionParam))) {
    return true;
}
}
return false;
}

public MenuBar getMenuBar() {
    return menuBar;
}

public void setMenuBar(MenuBar menuBar) {
    this.menuBar = menuBar;
}

public MenuBar getMenuBarPorPermissoes(Set<String> perms) {
    MenuBar menuBar = new MenuBar();
    for (Menu menu : getMenuBar().getMenus()) {
        Menu menuUsuario = getMenu(perms, menu);
        if (menuUsuario != null) {
            menuBar.add(menuUsuario);
        }
    }
    return menuBar;
}

private Menu getMenu(Set<String> perms, Menu menuSistema) {
    Menu menuUsuario = null;
    if (menuSistema.getSubmenus().size() > 0) {
        for (Menu submenuSistema : menuSistema.getSubmenus()) {
            Menu submenuUsuario = getMenu(perms,
submenuSistema);
            if (submenuUsuario != null) {
                // menuUsuario = submenuUsuario.getParent();
                if (menuUsuario == null) {
                    menuUsuario = new Menu();
                    menuUsuario.setId(menuSistema.getId());

menuUsuario.setLink(menuSistema.getLink());

menuUsuario.setTitulo(menuSistema.getTitulo());

menuUsuario.setDropdown(menuSistema.isDropdown());

menuUsuario.setProtegido(menuSistema.isProtegido());
                }
            }
        }
    }
}

```

```

        if
(!menuUsuario.getSubmenus().contains(submenuUsuario)) {
            menuUsuario.addMenu(submenuUsuario);
        }
    }
} else {
    if (!menuSistema.isProtegido() ||
menuSistema.getSubmenus().size() > 0 || (menuSistema.getLink() != null &&
podeAcessar(perms, menuSistema.getLink()))) {
        menuUsuario = new Menu();
        menuUsuario.setId(menuSistema.getId());
        menuUsuario.setLink(menuSistema.getLink());
        menuUsuario.setTitulo(menuSistema.getTitulo());
        menuUsuario.setDropdown(menuSistema.isDropdown());
        menuUsuario.setProtegido(menuSistema.isProtegido());
    }
}
return menuUsuario;
}
}
}

```

ChecaPermissaoFiltro.java

```

package br.com.aragao.guarda;

import java.io.IOException;

import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.fileupload.FileUpload;

import com.oreilly.servlet.MultipartRequest;

public class ChecaPermissaoFiltro implements javax.servlet.Filter {

    FilterConfig filterConfig = null;

    public void init( FilterConfig filterConfig) throws ServletException {
        this.filterConfig = filterConfig;
    }
}

```

```

    }

    public void destroy() {
        this.filterConfig = null;
    }

    public void doFilter( ServletRequest request, ServletResponse response,
        FilterChain chain ) throws IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse resp = (HttpServletResponse) response;
        HttpSession sess = req.getSession();

        Sistema sis = Sistema.getInstance();

        String contexto = req.getContextPath();

        String url = req.getRequestURI().substring(contexto.length() + 1) +
        (req.getQueryString() != null ? "?" + req.getQueryString() : "");

        System.out.println("URL: " + url);

        if (url.indexOf "?" + sis.getActionParam() < 0) {
            String param = req.getParameter(sis.getActionParam());
            if (param == null && FileUpload.isMultipartContent(req)) {
                MultipartRequest mp = new MultipartRequest(req, "/tmp");
                param = mp.getParameter(sis.getActionParam());
            }
            url = url + "?" + sis.getActionParam() + "=" + param;
        }

        Guarda u = (Guarda) sess.getAttribute(sis.getUsuarioLogado());
        if (!sis.podeAcessar(u != null ? u.getPermissoes() : null, url)) {
            resp.sendRedirect(contexto + "/" + sis.getPaginaRedirecionamento());
            return;
        }
        chain.doFilter(request,response);
    }
}

```

MontaMenuTag.java

```

package br.com.aragao.guarda.menu;

import java.io.IOException;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.jsp.JspException;

```



```

import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.TagSupport;

import br.com.aragao.guarda.Guarda;
import br.com.aragao.guarda.Sistema;

public class MontaMenuTag extends TagSupport {
    private static final long serialVersionUID = 1L;

    public int doStartTag() throws JspException {
        HttpServletRequest request = (HttpServletRequest) pageContext.getRequest();

        MenuBar menuBar = (MenuBar)
request.getSession().getAttribute("MENU_SISTEMA");
        if (menuBar == null) {
            Sistema sis = Sistema.getInstance();
            Guarda u = (Guarda)
request.getSession().getAttribute(sis.getUsuarioLogado());
            menuBar = sis.getMenuBarPorPermissoes(u.getPermissoes());
        }

        if (menuBar != null) {
            JspWriter out = pageContext.getOut();
            try {
                out.println("<script type='text/javascript'>");
                out.println();
                out.println("webfxMenuImagePath = contexto +
\"/include/imagens/");
                out.println();
                for (Menu menu : menuBar.getMenus()) {
                    if (menu.isDropdown()) {
                        out.println("var " + menu.getIdCompleto() + "
= new WebFXMenu;");
                        if (menu.getSubmenus().size() > 0) {
                            for (Menu submenu :
menu.getSubmenus()) {
                                out.println();
                                write(submenu, " ");
                            }
                        }
                        out.println();
                    }
                }
                out.println("var myBar = new WebFXMenuBar;");
                for (Menu menu : menuBar.getMenus()) {
                    out.println("myBar.add(new WebFXMenuButton(\""
+ menu.getTitulo() + "\", " + (menu.getLink() != null ? "contexto + \"" + menu.getLink()

```

```

+ "\" : "null") + ", null" + (menu.isDropdown() ? ", " + menu.getIdCompleto() : "") +
");");

        }
        out.println();
        out.println("</script>");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

return SKIP_BODY;
}

private void write(Menu menu, String tab) {
    JspWriter out = pageContext.getOut();
    try {
        if (menu.getSubmenus().size() > 0) {
            out.println(tab + " var " + menu.getIdCompleto() + " =
new WebFXMenu;");
            for (Menu submenu : menu.getSubmenus()) {
                write(submenu, tab + " ");
            }
            out.println();
            out.println(tab + (menu.getParent() != null ?
menu.getParent().getIdCompleto() : "") + ".add(new WebFXMenuItem(\"" +
menu.getTitulo() + "\", " + (menu.getLink() != null ? "contexto + \"" + menu.getLink() +
"\" : \"null\") + ", null, " + menu.getIdCompleto() + "));");
        } else {
            out.println(tab + (menu.getParent() != null ?
menu.getParent().getIdCompleto() : "") + ".add(new WebFXMenuItem(\"" +
menu.getTitulo() + "\", " + (menu.getLink() != null ? "contexto + \"" + menu.getLink() +
"\" : \"null\") + ", null));");
        }

        out.println();

    } catch (IOException e) {
        e.printStackTrace();
    }
}
}
}

```

11.1.3 BEAN

Perfil.java

```

package br.com.jexperts.admin.beans;

import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;

/**
 * @hibernate.class table="admin_perfil" proxy="br.com.jexperts.admin.beans.Perfil"
 lazy="true"
 */
public class Perfil implements Serializable {
    private Long idPerfil;

    private String nome;

    private String titulo;

    private Set<String> permissoes = new HashSet<String>();

    private static final long serialVersionUID = 1L;

    /**
     * @hibernate.id generator-class="native" column="id_perfil" unsaved-
value="null"
     * @hibernate.generator-param name="sequence" value="seq_perfil"
     */
    public Long getIdPerfil() {
        return idPerfil;
    }

    public void setIdPerfil(Long idPerfil) {
        this.idPerfil = idPerfil;
    }

    /**
     * @hibernate.property
     */
    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    /**
     * @hibernate.property
     */

```

```

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    /**
     * @hibernate.set table="admin_permissoes" lazy="false" cascade="all-delete-
orphan" inverse="false"
     * @hibernate.key column="id_perfil"
     * @hibernate.element column="permissoes" type="string"
     */
    public Set<String> getPermissoes() {
        return permissoes;
    }

    public void setPermissoes(Set<String> permissao) {
        this.permissoes = permissao;
    }
}

```

Perfil.hbm.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
  <class proxy="br.com.jexperts.admin.beans.Perfil" table="admin_perfil" lazy="true"
name="br.com.jexperts.admin.beans.Perfil">
    <id column="id_perfil" unsaved-value="null" access="property" name="idPerfil">
      <generator class="native">
        <param name="sequence">seq_perfil</param>
      </generator>
    </id>
    <property name="nome" access="property"/>
    <property name="titulo" access="property"/>
    <set table="admin_permissoes" access="property" lazy="false" inverse="false"
cascade="all-delete-orphan" name="permissoes">
      <key column="id_perfil"/>
      <element type="string" column="permissoes"/>
    </set>
  </class>
</hibernate-mapping>

```

11.1.4 BUSINESS

ProjetoBusinessImpl.java

```
package br.com.jexperts.museu.business.impl;

import java.util.ArrayList;
import java.util.List;
import java.util.Set;

import br.com.aragao.resources.generico.Filtro;
import br.com.aragao.resources.generico.GenericBusinessImpl;
import br.com.jexperts.museu.DAO.ProjetoDAO;
import br.com.jexperts.museu.beans.Acervo;
import br.com.jexperts.museu.beans.Pessoa;
import br.com.jexperts.museu.beans.Projeto;
import br.com.jexperts.museu.business.AcervoBusiness;
import br.com.jexperts.museu.business.ProjetoBusiness;

public class ProjetoBusinessImpl extends GenericBusinessImpl<Projeto>
implements ProjetoBusiness {
    private AcervoBusiness acervoBusiness;

    public AcervoBusiness getAcervoBusiness() {
        return acervoBusiness;
    }

    public void setAcervoBusiness(AcervoBusiness acervoBusiness) {
        this.acervoBusiness = acervoBusiness;
    }

    public List<Projeto> listarProjetoMatriz() {
        ArrayList<Filtro> filtros = new ArrayList<Filtro>();
        String[] ordem = {"asc", "nome"};
        filtros.add(new Filtro("matriz.id", "=", null));

        List<Projeto> projetos = getDao().listar(filtros, 0, 0, ordem);

        return projetos;
    }

    public List<Projeto> listarProjetosOrdenados() {
        ArrayList<Filtro> filtros = new ArrayList<Filtro>();
        String[] ordem = {"asc", "nome"};

        List<Projeto> projetos = getDao().listar(filtros, 0, 0, ordem);

        return projetos;
    }
}
```

```

    }

    public List<Projeto> listarProjetoMemoriaLocal() {
        ArrayList<Filtro> filtros = new ArrayList<Filtro>();
        String[] ordem = null;
        filtros.add(new Filtro("isMemoriaLocal", "=", true));

        List<Projeto> projetos = getDao().listar(filtros, 0, 0, ordem);

        return projetos;
    }

    public void apagar(Projeto obj) {
        Set<Pessoa> pessoas = obj.getPessoasRelacionadas();
        for (Pessoa pessoa : pessoas) {
            pessoa.getProjetosRelacionados().remove(obj);
        }

        Set<Acervo> acervos = obj.getAcervos();
        for (Acervo acervo : acervos) {
            getAcervoBusiness().apagarArquivos(acervo);
        }
        getDao().apagar(obj);
    }

    public int getTotalProjetosPorMatriz(Long id) {
        return ((ProjetoDAO)getDao()).getTotalProjetosPorMatriz(id);
    }
    public List<Projeto> listarProjetosPorMatriz(Long idMatriz, String filtro, int
    inicio, int qtd, String... campos) {
        return ((ProjetoDAO)getDao()).listarProjetosPorMatriz(idMatriz, filtro,
    inicio, qtd, campos);
    }
}

```

11.1.5 DAO

ProjetoDaolmpl.java

```

package br.com.jexperts.museu.DAO.impl;

import java.util.List;

import org.hibernate.Query;

import br.com.aragao.resources.generico.GenericDAOImpl;

```

```

import br.com.jexperts.museu.DAO.ProjetoDAO;
import br.com.jexperts.museu.beans.Projeto;

public class ProjetoDAOImpl extends GenericDAOImpl<Projeto> implements
ProjetoDAO {
    protected Class getClazz() {
        return Projeto.class;
    }

    public int getTotalProjetosPorMatriz(Long id) {
        Query q = getSession().createQuery("select count(*) from
br.com.jexperts.museu.beans.Projeto projeto where projeto.matriz.id = :id");
        q.setLong("id", id);
        return ((Long) q.uniqueResult()).intValue();
    }

    public List<Projeto> listarProjetosPorMatriz(Long idMatriz, String filtro, int inicio, int
qtd, String... campos) {

        StringBuilder sb = new StringBuilder();

        if (filtro != null) {
            for (String campo : campos) {
                sb.append("lower(" + campo + ")");
                sb.append(" like ");
                sb.append(":" + campo);
                sb.append(" and ");
            }
        }

        Query q = null;
        if (idMatriz != null) {

            q = getSession().createQuery("select projeto from
br.com.jexperts.museu.beans.Projeto projeto where projeto.matriz.id = :id order by
projeto.nome");
            q.setLong("id", idMatriz);

        }

        q.setFirstResult(inicio);
        if (qtd > 0) {
            q.setMaxResults(qtd);
        }

        List<Projeto> lista = (List<Projeto>) q.list();

        return lista;
    }
}

```

```

    }
}

```

11.1.6 TAGS

IterateDefaultTag.java

```

package br.com.jexperts.museu.tags;

import java.util.Iterator;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.BodyTagSupport;

import br.com.jexperts.museu.beans.Projeto;

public abstract class IterateDefaultTag extends BodyTagSupport {

    private String var;
    private boolean paginar;
    private String busca;
    protected Iterator iterate;
    protected String param;
    private Integer qtdPorPagina;
    private Long idProjetoMatriz;
    private Long idProjetoRelacionado;
    private String tipo;
    private String tipoDiferente;

    private static final long serialVersionUID = 1L;

    public static final int QTDPORPAGINA = 10;

    public String getVar() {
        return var;
    }

    public void setVar(String var) {
        this.var = var;
    }

    public String getBusca() {
        if (param == null) {
            param = "buscar";
        }
    }
}

```



```

        HttpServletRequest request = (HttpServletRequest)
pageContext.getRequest();
        String buscar = request.getParameter(param);
        if (buscar == null) {
            buscar = busca;
        }

        return buscar;
    }

    public void setBusca(String busca) {
        this.busca = busca;
    }

    public int getPagina() {
        HttpServletRequest request = (HttpServletRequest)
pageContext.getRequest();
        int pagina = 0;
        if (paginar) {
            String nome = this.getClass().getSimpleName();
            nome = nome.substring(7, nome.indexOf("Tag"));
            if (request.getParameter("pagina" + nome) != null) {
                pagina = Integer.valueOf(request.getParameter("pagina" +
nome));
            } else {
                pagina = 1;
            }
        }
        return pagina;
    }

    public boolean getPaginar() {
        return paginar;
    }

    public void setPaginar(boolean pagina) {
        this.paginar = pagina;
    }

    public int doStartTag() throws JspException {

        if (var == null) {
            var = getClass().getName();
        }

        iterate = getlterator();
        if (process() == EVAL_BODY_AGAIN) {
            return EVAL_BODY_INCLUDE;
        }
    }

```

```
        } else {
            return SKIP_BODY;
        }
    }

    public int doAfterBody() throws JspException {
        return process();
    }

    public int doEndTag() {
        return EVAL_PAGE;
    }

    public void release() {
        var = null;
        paginar = false;
        busca = null;
        iterate = null;
    }

    protected int getInicio() {
        int inicio = (getPagina() - 1) * getQtdPorPagina();
        return inicio;
    }

    protected Integer getQtdPorPagina() {
        if (this.qtdPorPagina != null && this.qtdPorPagina >= 0) {
            return this.qtdPorPagina;
        }
        return getPagina() > 0 ? QTDPORPAGINA : 0;
    }

    public void setQtdPorPagina(Integer qtdPorPagina) {
        this.qtdPorPagina = qtdPorPagina;
    }

    protected int process() {
        if (iterate != null && iterate.hasNext()) {
            pageContext.setAttribute(var, iterate.next());
            return EVAL_BODY_AGAIN;
        }
        return SKIP_BODY;
    }

    protected Projeto getProjeto() {
        HttpServletRequest request = (HttpServletRequest)
pageContext.getRequest();
        return UtilHelper.getProjeto(request);
    }
}
```

```
    }  
  
    public abstract Iterator getIterator();  
  
    public Long getIdProjetoMatriz() {  
        return idProjetoMatriz;  
    }  
  
    public void setIdProjetoMatriz(Long idProjetoMatriz) {  
        this.idProjetoMatriz = idProjetoMatriz;  
    }  
  
    public Long getIdProjetoRelacionado() {  
        return idProjetoRelacionado;  
    }  
  
    public void setIdProjetoRelacionado(Long idProjetoRelacionado) {  
        this.idProjetoRelacionado = idProjetoRelacionado;  
    }  
  
    public String getTipo() {  
        return tipo;  
    }  
  
    public void setTipo(String tipo) {  
        this.tipo = tipo;  
    }  
  
    public String getTipoDiferente() {  
        return tipoDiferente;  
    }  
  
    public void setTipoDiferente(String tipoDiferente) {  
        this.tipoDiferente = tipoDiferente;  
    }  
}
```

HistóriaTag.java

```
package br.com.jexperts.museu.tags;  
  
import java.util.List;  
  
import javax.servlet.http.HttpServletRequest;  
  
import br.com.jexperts.museu.beans.Historia;  
import br.com.jexperts.museu.beans.Pessoa;  
import br.com.jexperts.museu.beans.Projeto;
```

```

import br.com.jexperts.museu.business.delegate.HistoriaDelegate;
import br.com.jexperts.xml.Parser;

public class HistoriaTag extends ObjectDefaultTag {

    private int pagina;
    private boolean paginar;

    private static final long serialVersionUID = 1L;

    @Override
    protected Historia getObjeto() {
        Historia historia = null;
        IterateHistoriaTag ht = (IterateHistoriaTag) findAncestorWithClass(this,
IterateHistoriaTag.class);
        if (ht != null) {
            historia = (Historia) pageContext.getAttribute(ht.getVar());
        }
        if (historia == null) {
            PessoaTag pt = (PessoaTag) findAncestorWithClass(this,
PessoaTag.class);
            if (pt != null) {
                Pessoa p = pt.getPessoa();
                List<Historia> historias =
HistoriaDelegate.getInstance().listarPorProprietario(p.getId(), null, 0, 1, new
String[]{});
                if (historias != null && historias.size() > 0) {
                    historia = (Historia) historias.get(0);
                }
            }
        }
        if (historia != null) {
            HttpServletRequest request = (HttpServletRequest)
pageContext.getRequest();
            request.setAttribute("idHistoria", historia.getIdAcervo());
        }

        return historia;
    }

    @Override
    protected Historia getObjeto(Long id) {
        return HistoriaDelegate.getInstance().editar(getIdObjeto());
    }

    protected Object getToPrint(String print) {
        Historia historia = (Historia) tmp;
    }

```

```

        Object toPrint = null;
        if (historia != null) {

            }
            return toPrint;
        }

        protected Parser getParser() {
            Historia hist = getHistoria();
            Parser parser = null;
            if (hist != null && hist.getXml()) {
                IterateHistoriaTag ht = (IterateHistoriaTag)
findAncestorWithClass(this, IterateHistoriaTag.class);
                if (ht != null) {
                    parser = ht.getParser();
                }
                Projeto proj = getProjeto();
                parser = parser != null ? parser : hist != null && hist.getXml() ?
new Parser(proj, hist.getConteudo()) : null;
            }
            return parser;
        }

        public Historia getHistoria() {
            return tmp != null ? (Historia) tmp : null;
        }

        public int getPagina() {
            return pagina;
        }

        public void setPagina(int pagina) {
            this.pagina = pagina;
        }

        public boolean getPaginar() {
            return paginar;
        }

        public void setPaginar(boolean paginar) {
            this.paginar = paginar;
        }
    }
}

```

PaginadorDefaultTag.java

```

package br.com.jexperts.museu.tags;

import java.io.IOException;

import javax.servlet.http.HttpServletRequest;

```

```

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.TagSupport;

import br.com.jexperts.museu.beans.Projeto;

public abstract class PaginadorDefaultTag extends TagSupport {

    private String styleClass;

    private Long idProprietarioAcervo;
    private Long idProjetoRelacionado;
    private Long idProjetoMatriz;
    private String tipo;
    private String tipoDiferente;

    private String tituloAnterior;
    private String tituloProxima;
    private Integer qtdPorPagina;

    private Boolean slide;

    private static final long serialVersionUID = 1L;

    public int doStartTag() throws JspException {

        JspWriter out = pageContext.getOut();
        HttpServletRequest request = (HttpServletRequest)
pageContext.getRequest();

        int pagina = 1;
        if (request.getParameter(getName()) != null) {
            pagina = Integer.parseInt(request.getParameter(getName()));
        }
        double totalRegistros = getTotalRegistros();
        request.setAttribute("TOTAL", (int)totalRegistros);
        double qtdPagina = lterateDefaultTag.QTDPORPAGINA;

        if (qtdPorPagina != null && qtdPorPagina > 0) {
            qtdPagina = this.qtdPorPagina;
        }

        int totalPaginas = (int) Math.ceil(((double) totalRegistros / (double)
qtdPagina);
        try {
            if (totalPaginas > 1) {
                String c = styleClass != null ? "class=\"" + styleClass + "\""
: "";

```

```

        if (slide == null || !slide) {
            if (pagina > 1) {
                out.println("<a href=\"indexProjeto.do?\" +
UtilHelper.getParams(request, getName(), String.valueOf(pagina - 1)) + \"\
\" + c + \">\"
+ setupTituloAnterior() + \"</a> \");
            }

            for (int i = 1; i <= totalPaginas; i++) {
                if (i == pagina) {
                    out.println(i + " ");
                } else {
                    out.println("<a href=\"indexProjeto.do?\" +
UtilHelper.getParams(request, getName(), String.valueOf(i)) + \"\
\" + c + \">\" + i +
\"</a> \");
                }
            }

            if (pagina < totalPaginas) {
                out.println("<a href=\"indexProjeto.do?\" +
UtilHelper.getParams(request, getName(), String.valueOf(pagina + 1)) + \"\
\" + c + \">\"
+ setupTituloProxima() + \"</a> \");
            }
            } else {
                if (pagina > 1) {
                    out.println("<a href=\"indexProjeto.do?\" +
UtilHelper.getParams(request, getName(), String.valueOf(pagina - 1)) + \"\
\" + c + \">\"
+ setupTituloAnterior() + \"</a> \" + \" : \");
                } else {
                    out.println("<span "+c+">\"
+setupTituloAnterior()+ \"</span>\"+ \" : \");
                }

                if (pagina < totalPaginas) {
                    out.println("<a href=\"indexProjeto.do?\" +
UtilHelper.getParams(request, getName(), String.valueOf(pagina + 1)) + \"\
\" + c + \">\"
+ setupTituloProxima() + \"</a> \");
                } else {
                    out.println("<span "+c+">\"
+setupTituloProxima()+ \"</span>\"");
                }
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }

    return SKIP_BODY;

```

```
    }

    private String setupTituloAnterior() {
        String s = tituloAnterior;
        if(s != null && !s.equals("")) {
            return s;
        } else {
            return "anterior";
        }
    }

    private String setupTituloProxima() {
        String s = tituloProxima;
        if(s != null && !s.equals("")) {
            return s;
        } else {
            return "próxima";
        }
    }

    private String getName() {
        String name = this.getClass().getSimpleName();
        return "pagina" + name.substring(9, name.indexOf("Tag"));
    }

    public abstract int getTotalRegistros();

    public String getStyleClass() {
        return styleClass;
    }

    public void setStyleClass(String styleClass) {
        this.styleClass = styleClass;
    }

    protected Projeto getProjeto() {
        HttpServletRequest request = (HttpServletRequest)
pageContext.getRequest();
        return UtilHelper.getProjeto(request);
    }

    public Long getIdProprietarioAcervo() {
        return idProprietarioAcervo;
    }

    public void setIdProprietarioAcervo(Long idProprietarioAcervo) {
        this.idProprietarioAcervo = idProprietarioAcervo;
    }
}
```



```
public String getTipo() {
    return tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
}

public String getTipoDiferente() {
    return tipoDiferente;
}

public void setTipoDiferente(String tipoDiferente) {
    this.tipoDiferente = tipoDiferente;
}

public Long getIdProjetoRelacionado() {
    return idProjetoRelacionado;
}

public void setIdProjetoRelacionado(Long idProjetoRelacionado) {
    this.idProjetoRelacionado = idProjetoRelacionado;
}

public String getTituloAnterior() {
    return tituloAnterior;
}

public void setTituloAnterior(String tituloAnterior) {
    this.tituloAnterior = tituloAnterior;
}

public String getTituloProxima() {
    return tituloProxima;
}

public void setTituloProxima(String tituloProxima) {
    this.tituloProxima = tituloProxima;
}

public Boolean isSlide() {
    return slide;
}

public void setSlide(Boolean slide) {
    this.slide = slide;
}
```

```

    public Integer getQtdPorPagina() {
        return qtdPorPagina;
    }

    public void setQtdPorPagina(Integer qtdPorPagina) {
        this.qtdPorPagina = qtdPorPagina;
    }

    public Long getIdProjetoMatriz() {
        return idProjetoMatriz;
    }

    public void setIdProjetoMatriz(Long idProjetoMatriz) {
        this.idProjetoMatriz = idProjetoMatriz;
    }
}

```

11.1.7 XML

Parser.java

```
package br.com.jexperts.xml;
```

```
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
```

```
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.xml.sax.InputSource;
```

```
import br.com.jexperts.admin.beans.Glossario;
import br.com.jexperts.admin.business.delegate.GlossarioDelegate;
import br.com.jexperts.museu.beans.Projeto;
```

```
public class Parser {
```

```
    private Projeto projeto;
    private Document doc;
    private ArrayList<String> citations = new ArrayList<String>();
    private LinkedHashMap<String, String> indexes = new
LinkedHashMap<String, String>();

```

```
private LinkedHashMap<String, String> indexesTitles = new
LinkedHashMap<String, String>();
private HashMap<String, Integer> indexesPagina = new HashMap<String,
Integer>();
private ArrayList<String> paginas = new ArrayList<String>();

public Parser(Projeto projeto, String xml) {
    SAXBuilder parser = new SAXBuilder();
    if(xml == null) {
        xml = "";
    }

    try {
        doc = parser.build(new InputSource(new
ByteArrayInputStream(xml.getBytes())));
    } catch (JDOMException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    this.projeto = projeto;
    this.parse();
}

public String getHtml() {
    StringBuilder html = new StringBuilder();
    for (String pagina : paginas) {
        html.append(pagina);
    }
    return html.toString();
}

public String getHtml(int pagina) {
    if (pagina > 0 && this.paginas.size() >= pagina) {
        return this.paginas.get(pagina - 1);
    }
    return "";
}

public int getTotalPaginas() {
    return this.paginas.size();
}

private void parse() {
    Element root = doc.getRootElement();
    StringBuilder html = new StringBuilder();
```

```

Element identificationEl = root.getChild("identification");
html.append("<b>IdentificacÃ§Ã£o</b>\n<br><br>\n");
Iterator pldents = identificationEl.getChildren("p").iterator();
StringBuilder tmp = new StringBuilder();
while (pldents.hasNext()) {
    Element pEl = (Element) pldents.next();
    tmp.append("<p>");
    tmp.append(paragraphProcessor(pEl));
    tmp.append("</p>\n");
}
this.indexes.put("identification", tmp.toString());
html.append(tmp);

Iterator paginas = root.getChildren("page").iterator();
int paginaNro = 1;
while (paginas.hasNext()) {
    Element pagina = (Element) paginas.next();
    Iterator indexes = pagina.getChildren("index").iterator();
    while (indexes.hasNext()) {
        html.append("<br><br>\n");
        Element indexEl = (Element) indexes.next();
        html.append("<a name=\"" +
indexEl.getAttributeValue("name") + "\"></a><b>" + indexEl.getAttributeValue("title")
+ "</b>\n<br><br>\n");
        Iterator plIndexs = indexEl.getChildren("p").iterator();
        tmp = new StringBuilder();
        while (plIndexs.hasNext()) {
            Element pEl = (Element) plIndexs.next();
            tmp.append("<p>");
            tmp.append(paragraphProcessor(pEl));
            tmp.append("</p>\n");
        }
        this.indexes.put(indexEl.getAttributeValue("name"),
tmp.toString());
        this.indexesTitles.put(indexEl.getAttributeValue("name"),
indexEl.getAttributeValue("title"));
        this.indexesPagina.put(indexEl.getAttributeValue("name"),
paginaNro);
        html.append(tmp);
    }
    this.paginas.add(html.toString());
    html = new StringBuilder();
    paginaNro ++;
}
}

public ArrayList<String> getCitations() {

```

```
        return citations;
    }

    public LinkedHashMap<String, String> getIndexes() {
        return indexes;
    }

    public String getIndex(String key) {
        return this.indexes.get(key);
    }

    public String getIndexTitle(String key) {
        return this.indexesTitles.get(key);
    }

    public int getIndexPagina(String key) {
        return this.indexesPagina.get(key).intValue();
    }

    private String paragraphProcessor(Element p) {

        Iterator glossaries = p.getChildren("glossary").iterator();
        while (glossaries.hasNext()) {
            Element glossary = (Element) glossaries.next();
            glossary.setText(glossaryProcessor(glossary));
        }

        Iterator bolds = p.getChildren("bold").iterator();
        while (bolds.hasNext()) {
            Element bold = (Element) bolds.next();
            bold.setText(boldProcessor(bold));
        }

        Iterator italics = p.getChildren("italic").iterator();
        while (italics.hasNext()) {
            Element italic = (Element) italics.next();
            italic.setText(italicProcessor(italic));
        }

        Iterator links = p.getChildren("link").iterator();
        while (links.hasNext()) {
            Element link = (Element) links.next();
            link.setText(linkProcessor(link));
        }

        Iterator citations = p.getChildren("citation").iterator();
        while (citations.hasNext()) {
            Element citation = (Element) citations.next();
        }
    }
}
```

```

        citation.setText(citationProcessor(citation));
    }

    return p.getValue();
}

private String glossaryProcessor(Element glossary) {
    Long idProjeto = projeto != null ? projeto.getId() : null;
    Glossario glossario =
GlossarioDelegate.getInstance().editarPorNome(idProjeto, glossary.getText());
    StringBuilder sb = new StringBuilder();
    if (glossario != null) {
        sb.append("<a href=\"javascript:void(0)\"
onclick=\"showHelpTip(event, \" + glossario.getDescricao().replaceAll(\"\\n\", \"\") + \"
,true); return false;\">\" + glossario.getNome() + \"</a>\"");
    } else {
        sb.append(glossary.getText());
    }
    return sb.toString();
}

private String boldProcessor(Element bold) {
    return "<b>\" + bold.getText() + \"</b>\"";
}

private String italicProcessor(Element italic) {
    return "<i>\" + italic.getText() + \"</i>\"";
}

private String linkProcessor(Element link) {
    return "<a href=\"\" + link.getAttributeValue(\"url\") + \"\" target=\"_blank\">\"
+ link.getText() + \"</a>\"";
}

private String citationProcessor(Element citation) {
    this.citations.add(citation.getText());
    return citation.getText();
}
}

```

XSDTools.java

```

package br.com.jexperts.xml;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;

```

```
import java.util.ArrayList;

import org.apache.ws.jaxme.xs.XSAttributable;
import org.apache.ws.jaxme.xs.XSAttribute;
import org.apache.ws.jaxme.xs.XSComplexType;
import org.apache.ws.jaxme.xs.XSElement;
import org.apache.ws.jaxme.xs.XSParser;
import org.apache.ws.jaxme.xs.XSParticle;
import org.apache.ws.jaxme.xs.XSSchema;
import org.apache.ws.jaxme.xs.XSType;
import org.apache.xerces.impl.xs.traversers.XSDHandler;
import org.apache.xerces.parsers.SAXParser;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.helpers.DefaultHandler;

import br.com.jexperts.resources.GerenciadorArquivo;

public class XSDTools {

    private String xsdPath;

    private GerenciadorArquivo ga = new GerenciadorArquivo();

    public XSDTools(String xsdPath) {
        this.xsdPath = xsdPath;
    }

    public String validateSchema(String xml) {
        SAXParser parser = new SAXParser();
        try {
            parser.setFeature("http://xml.org/sax/features/validation", true);

            parser.setFeature("http://apache.org/xml/features/validation/schema", true);

            parser.setFeature("http://apache.org/xml/features/validation/schema-full-
checking", true);

            parser.setProperty("http://apache.org/xml/properties/schema/external-
noNamespaceSchemaLocation", ga.getPath() + "/" + xsdPath);

            Validator handler = new Validator();
            parser.setErrorHandler(handler);
```

```

        parser.parse(new InputSource(new
ByteArrayInputStream(xml.getBytes())));
        if (handler.validationError == true) {
            return handler.saxParseException.getMessage();
        }
    } catch (java.io.IOException ioe) {
        return ioe.getMessage();
    } catch (SAXException e) {
        return e.getMessage();
    }
    return null;
}

public ArrayList<Tag> getTags() throws Exception {
    XSParser xsp = new XSParser();
    xsp.setValidating(false);
    File f = new File(ga.getPath() + xsdPath);
    InputSource isource = new InputSource(new FileInputStream(f));

    XSSchema schema = xsp.parse(isource);

    ArrayList<Tag> tags = new ArrayList<Tag>();
    XSElement[] els = schema.getElements();
    for (int i = 0; i < els.length; i++) {
        Tag t = new Tag();
        t.setName(els[i].getName().toString());
        tags.add(t);
        getSubElement(els[i], t);
    }

    printTags(tags, "");

    return tags;
}

private void printTags(ArrayList<Tag> tags, String tab) {
    if (tags.size() > 0) {
        for (Tag t : tags) {
            printTags(t.getSubtags(), tab + "\t");
        }
    }
}

private void getSubElement(XSElement el, Tag tag) throws SAXException {
    XSType type = el.getType();
    if (!type.isSimple()) {
        XSComplexType ct = type.getComplexType();
        if (ct.hasSimpleContent() && ct.getAttributes().length > 0) {

```



```

        this.addAttributes(tag, ct.getAttributes());
    } else if (ct.isExtension()) {
        System.out.println(ct.getExtendedType().getClass());
    } else {
        XSParticle particle = ct.getParticle();
        if (particle != null && particle.isElement()) {
            XSElement subElement = particle.getElement();
            Tag subTag =
tag.addSubTag(subElement.getName().toString());
            getSubElement(subElement, subTag);
        } else if (particle != null && particle.isGroup() &&
particle.getGroup() != null) {
            for (int i = 0; i <
particle.getGroup().getParticles().length; i++) {
                XSElement subElement =
particle.getGroup().getParticles()[i].getElement();
                Tag subTag =
tag.addSubTag(subElement.getName().toString());

                getSubElement(particle.getGroup().getParticles()[i].getElement(), subTag);
            }
        }
    }
}

private void addAttributes(Tag t, XSAttributable[] attributes) {
    for (XSAttributable tmp : attributes) {
        XSAttribute attribute = (XSAttribute) tmp;
        t.addAttribute(attribute.getName().toString());
    }
}

public static void main(String[] args) {
    XSDHandler xsd;
    // XSDTools xml = new
XSDTools("/config/indexacaoDepoimento/2/xml/mp_noglobal.xsd");
    XSDTools xml = new XSDTools("/regras.xsd");
    try {

        xml.validateSchema("<mp><identification><p>rtesa</p></identification><inde
x name=\"teste\"><p>asfaf</p></index></mp>");
        // xml.getTags();
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

```

private class Validator extends DefaultHandler {
    public boolean validationError = false;

    public SAXParseException saxParseException = null;

    public void error(SAXParseException exception) throws SAXException {
        validationError = true;
        saxParseException = exception;
    }

    public void fatalError(SAXParseException exception) throws
SAXException {
        validationError = true;
        saxParseException = exception;
    }

    public void warning(SAXParseException exception) throws
SAXException {
    }
}

```

11.1.8 LUCENE

Indexador.java

```

package br.com.jexperts.museu.lucene;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import org.apache.lucene.analysis.br.BrazilianAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.Term;
import org.apache.lucene.queryParser.ParseException;
import org.apache.lucene.queryParser.QueryParser;
import org.apache.lucene.search.Hits;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;

```

```

import br.com.aragao.resources.generico.Filtro;

public abstract class Indexador<T> {
    private IndexReader indexReader;
    private IndexWriter indexWriter;
    private IndexSearcher indexSearcher;

    public synchronized void indexar(T object) throws IOException {
        delete(object);
        closeReader();
        if (indexWriter == null) {
            indexWriter = new IndexWriter(getDir(), new BrazilianAnalyzer());
        }
        indexWriter.addDocument(getDocument(object));
    }

    public synchronized void indexar(List<T> objects) throws IOException {
        for (T object : objects) {
            delete(object);
        }
        closeReader();

        if (indexWriter == null) {
            indexWriter = new IndexWriter(getDir(), new BrazilianAnalyzer());
        }
        for (T object : objects) {
            indexWriter.addDocument(getDocument(object));
        }
    }

    public void closeWriter() throws IOException {
        indexWriter.optimize();
        indexWriter.flush();
        indexWriter.close();
        indexWriter = null;
    }

    public void delete(String key, String value) {
        try {
            if (indexReader == null) {
                indexReader = IndexReader.open(getDir());
            }

            if (indexReader != null) {
                indexReader.deleteDocuments(new Term(key, value));
            }
            closeReader();
        }
    }
}

```

```

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void closeReader() throws IOException {
        if (indexReader != null) {
            indexReader.close();
            indexReader = null;
        }
    }

    public ArrayList<Long> buscaIndexada(String palavraBusca, String
chaveBusca) throws IOException, ParseException {
        if (indexReader == null) {
            indexReader = IndexReader.open(getDir());
        }
        indexSearcher = new IndexSearcher(indexReader);

        QueryParser parser = new QueryParser(chaveBusca, new
BrazilianAnalyzer());
        Query query = parser.parse(palavraBusca);
        Hits hits = indexSearcher.search(query);

        Set<Long> ids = new HashSet<Long>();
        for (int i = 0; i < hits.length(); i++) {
            Document d = hits.doc(i);
            Enumeration enumeration = d.fields();
            while (enumeration.hasMoreElements()) {
                Field f = (Field) enumeration.nextElement();
                // System.out.println(f.name() + " - " + f.stringValue());
                if (f.name().equals("id")) {
                    ids.add(Long.valueOf(f.stringValue()));
                }
            }
        }

        indexSearcher.close();
        return new ArrayList<Long>(ids);
    }

    public Hits performSearch(ArrayList<Filtro> filtros) throws IOException,
ParseException {
        if (indexReader == null) {
            indexReader = IndexReader.open(getDir());
        }
        indexSearcher = new IndexSearcher(indexReader);

```

```

        QueryParser parser = new QueryParser("historia", new
BrazilianAnalyzer());
        Query query = parser.parse(getQueryString(filtros));
        Hits hits = indexSearcher.search(query);
        indexSearcher.close();

        return hits;
    }

    protected String getQueryString(ArrayList<Filtro> filtros) {
        StringBuilder sb = new StringBuilder();
        for (Filtro filtro : filtros) {
            sb.append(filtro.getCampo());
            sb.append(": ");
            sb.append(filtro.getValor());
            if (filtro.getCondicao() != null) {
                sb.append(" ");
                sb.append(filtro.getCondicao());
            }
            sb.append(" ");
        }
        return sb.toString();
    }

    protected abstract void delete(T object);

    protected abstract Document getDocument(T historia);

    protected abstract String getDir();
}

```

IndexadorHistoria.java

```

package br.com.jexperts.museu.lucene;

import org.apache.lucene.document.DateTools;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;

import br.com.jexperts.museu.beans.Historia;
import br.com.jexperts.resources.GerenciadorArquivo;

public class IndexadorHistoria extends Indexador<Historia> {

    @Override
    protected Document getDocument(Historia historia) {

```

```

        String texto =
Tools.removerAcento(historia.getConteudo().replaceAll("<.*?>|\n", ""));
        Document doc = new Document();
        doc.add(new Field("id", historia.getIdAcervo().toString(),
Field.Store.YES, Field.Index.UN_TOKENIZED));
        doc.add(new Field("modified",
DateTools.timeToString(historia.getDataUltimaAlteracao().getTime(),
DateTools.Resolution.MINUTE), Field.Store.YES, Field.Index.UN_TOKENIZED));
        doc.add(new Field("sinopse", historia.getSinopse(),
Field.Store.COMPRESS, Field.Index.TOKENIZED));
        doc.add(new Field("historia", texto, Field.Store.NO,
Field.Index.TOKENIZED));
        doc.add(new Field("frase", historia.getFrase(), Field.Store.NO,
Field.Index.TOKENIZED));
        doc.add(new Field("nome", historia.getNome(), Field.Store.NO,
Field.Index.TOKENIZED));

        String h = " " + historia.getNome();
        h += " " + texto;
        h += " " + historia.getFrase();
        h += " " + historia.getSinopse();
        doc.add(new Field("h", h, Field.Store.NO, Field.Index.TOKENIZED));

        return doc;
    }

    @Override
    protected void delete(Historia h) {
        delete("id", h.getIdAcervo().toString());
    }

    @Override
    protected String getDir() {
        GerenciadorArquivo ga = new GerenciadorArquivo();

        String path = ga.getContainerPath() + ga.getPath() +
"/museu_arquivos/index/historia";
        // String path = "c:\\projetos\\museu-da-pessoa-
2007\\WebContent\\museu_arquivos\\index\\historia";

        return path;
    }
}

```

Search.java

```
package br.com.jexperts.museu.lucene;
```

```

import org.apache.lucene.document.DateTools;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;

import br.com.jexperts.museu.beans.Historia;
import br.com.jexperts.resources.GerenciadorArquivo;

public class IndexadorHistoria extends Indexador<Historia> {

    @Override
    protected Document getDocument(Historia historia) {
        String texto =
Tools.removerAcento(historia.getConteudo().replaceAll("<.*?>|\n", ""));
        Document doc = new Document();
        doc.add(new Field("id", historia.getIdAcervo().toString(),
Field.Store.YES, Field.Index.UN_TOKENIZED));
        doc.add(new Field("modified",
DateTools.timeToString(historia.getDataUltimaAlteracao().getTime(),
DateTools.Resolution.MINUTE), Field.Store.YES, Field.Index.UN_TOKENIZED));
        doc.add(new Field("sinopse", historia.getSinopse(),
Field.Store.COMPRESS, Field.Index.TOKENIZED));
        doc.add(new Field("historia", texto, Field.Store.NO,
Field.Index.TOKENIZED));
        doc.add(new Field("frase", historia.getFrase(), Field.Store.NO,
Field.Index.TOKENIZED));
        doc.add(new Field("nome", historia.getNome(), Field.Store.NO,
Field.Index.TOKENIZED));

        String h = " " + historia.getNome();
        h += " " + texto;
        h += " " + historia.getFrase();
        h += " " + historia.getSinopse();
        doc.add(new Field("h", h, Field.Store.NO, Field.Index.TOKENIZED));

        return doc;
    }

    @Override
    protected void delete(Historia h) {
        delete("id", h.getIdAcervo().toString());
    }

    @Override
    protected String getDir() {
        GerenciadorArquivo ga = new GerenciadorArquivo();

```

```

        String path = ga.getContainerPath() + ga.getPath() +
"/museu_arquivos/index/historia";
        // String path = "c:\\projetos\\museu-da-pessoa-
2007\\WebContent\\museu_arquivos\\index\\historia";

        return path;
    }
}

```

11.1.9 XSD

Mp_noglobal.xsd

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2006 sp2 U (http://www.altova.com) by michael
(EMBRACE) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="mp">
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element name="identification">
                    <xs:complexType>
                        <xs:choice>
                            <xs:element name="p">
                                <xs:complexType
mixed="true">
                                    <xs:choice
minOccurs="0" maxOccurs="unbounded">
                                        <xs:element
name="glossary" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                                        <xs:element
name="citation" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                                        <xs:element
name="bold" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                                        <xs:element
name="italic" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                                        <xs:element
name="link" minOccurs="0" maxOccurs="unbounded">
                                            <xs:complexType>
                                                <xs:simpleContent>
                                                    <xs:extension base="xs:string">

```



```

        <xs:extension base="xs:string">
            <xs:attribute name="url" type="xs:string"
use="required"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
                                </xs:element>
                                </xs:choice>
                                <xs:attribute
name="name" type="xs:string" use="required"/>
                                <xs:attribute name="title"
type="xs:string" use="optional"/>
                                </xs:complexType>
                                </xs:element>
                                </xs:sequence>
                                </xs:complexType>
                                </xs:element>
                                </xs:sequence>
                                </xs:complexType>
                                </xs:element>
</xs:schema>

```


12 ARTIGO

Projeto de Reformulação e Implantação do Portal Museu da Pessoa

Rafael de Souza Coelho

Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 - 88040-900 - Santa Catarina - SC - Brasil

socoelho@inf.ufsc.br

Resumo. *Esse trabalho tem como objetivo desenvolver um sistema Web para o Instituto Museu da Pessoa. O projeto é uma reformulação do portal já existente para atender às necessidades do novo contexto de arquitetura de conteúdo. Este portal tem por finalidade ser um espaço aberto para que qualquer pessoa através da internet possa divulgar sua história de vida ou de terceiros. Essas histórias são relatadas por meio de depoimentos, podendo ser complementadas com imagens, áudios e vídeos, organizados de acordo com os métodos determinados pelo Museu. A área administrativa do sistema gerencia todo o acervo histórico e o expõe na internet para consulta. Para tanto, foi feito o levantamento das necessidades solicitadas visando após a análise dos requisitos, a implementação das melhorias e mudanças verificadas. Essa nova forma de registrar a história mostra-se inovadora no sentido de estabelecer uma rede mundial entre pessoas e o tempo.*

Abstract. *The objective of this research is the development of a Web system for the Institute Museum of the Person. The project is a new formulation of the existent portal in order to meet the demands of the new architecture of contents. The main goal of the portal is to be an open environment where people can show the history of his or others life in the internet. Those histories are shown with testimonies, and can be complemented with images, sound and videos, that are organized in accordance with the methods determined by the Museum. The administrative area of the system manages the historical inventory and put it on the internet, open to the public information. A survey of all the necessities was made and after all the analysis, the implementation of enhancements and changes has been verified. This new way to record the history is an innovation in the sense that it establishes a global web between people and time.*

1. Introdução

A história é uma peça fundamental em todo o tipo de cultura social. Através dela podemos conhecer a evolução das tradições e valores intelectuais, morais e espirituais da humanidade ao longo de seu passado e presente. A cultura de uma civilização é preservada ao longo dos anos pela perpetuação de sua história. O homem deixa marcas ao passar pela vida, uma foto, um desenho, uma canção, todos são traços de sua passagem.

A história de vida de cada pessoa tem seu valor, cada indivíduo contribui para a construção da história de sua sociedade. A partir desses princípios, no início de 2003, o Instituto do Museu da Pessoa trouxe à empresa JExperts Tecnologia a missão de criar um sistema integrado de interface única para o armazenamento e gerenciamento de documentos históricos relacionados à história de vida de diversas pessoas do mundo. Esse acervo inclui: imagens, áudios, vídeos, documentos e histórias de pessoas, organizadas segundo as metodologias do Museu.

Após a implantação do sistema, surgiram novos conceitos para a organização desses dados e necessidades não previstas na concepção inicial do projeto. Sob essas circunstâncias, surgiram problemas que acabaram tornando essa estrutura inadequada. Com isso, no segundo semestre de 2006, o Museu da Pessoa retornou à JExperts com o intuito de reformar o portal já existente. Porém, como haveria um grande esforço para reestruturar o banco de dados e as regras do sistema atual tornou-se necessária uma reforma tecnológica, que resultou em um novo sistema.

2. História de Vida

A história de vida é uma ferramenta da pesquisa qualitativa. A pesquisa qualitativa tem por objetivo traduzir e expressar o sentido dos fenômenos do mundo social; trata-se de reduzir a distância entre indicador e indicado, entre teoria e dados, entre contexto e ação (MAANEM, 1979, p.520). Ela é realizada por meio de métodos de coleta de dados, como entrevistas e relatos.

Através das histórias de vida é possível analisar a intersecção entre a vida individual de uma pessoa e o contexto social. Maria Ângela Silveira Paulino (2007), Doutora em Serviço Social pela PUC-SP, considera que:

“A história de vida pode ser, desta forma, um instrumento privilegiado para análise e interpretação, na medida em que incorpora experiências subjetivas mescladas a contextos sociais. Ela fornece, portanto, base consistente para o entendimento do componente histórico dos fenômenos individuais, assim como para a compreensão do componente individual dos fenômenos históricos.”

Ela possibilita aprender a cultura de forma aprofundada, é um instrumento valioso pois se coloca entre as relações externas e internas do indivíduo. São sempre relatos de práticas sociais: das formas com que o sujeito se insere e atua no mundo e no grupo do qual ele faz parte.

Existem diversas maneiras para realizar o estudo de vida de uma pessoa. As entrevistas são os meios mais comuns para coletar essas informações, o entrevistador respeita a opinião do indivíduo e acredita no que ele diz. Normalmente o entrevistado relata apenas

aquilo que considera mais importante e pode descartar momentos que ache desnecessários. Com isso, caso seja necessário, pode-se abranger os métodos de coleta de dados e incluir qualquer tipo de documento que possa colaborar no desenvolvimento da pesquisa, como: fotos, desenhos, vídeos, áudios, textos.

3. O Instituto Museu da Pessoa

O Instituto Museu da Pessoa foi fundado em 1991 em São Paulo. Seu objetivo é de construir uma rede internacional de histórias de vida capaz de contribuir para a mudança social. Eles acreditam que a memória social pode ajudar a criar diferentes perspectivas da nossa sociedade (MUSEU DA PESSOA, 2007). As histórias de vida são informações únicas, com elas é possível conhecer as diferentes realidades que cada pessoa cria em sua vida.

O Museu da Pessoa tem o objetivo de registrar, preservar e transformar em informação histórias de vida de toda e qualquer pessoa da sociedade, seja ela célebre ou anônima. “Promovendo mudanças sociais por meio da reflexão sobre a identidade e valorização de indivíduos e comunidades” (ASHOKA EMPREENDEDORES SOCIAIS, 2007).

Mesmo não existindo internet na época, o Museu se define desde o início como um museu virtual, pois todos os seus dados são armazenados digitalmente. Seu acervo é constituído de fotos, imagens, depoimentos, áudios e vídeos. Ele desenvolveu sua própria metodologia para registrar os depoimentos e formar um acervo inédito de histórias de vida.

O Museu da Pessoa já desenvolveu mais de 50 projetos em todo o Brasil, com diferentes empresas, sindicatos, associações, comunidades e escolas. Atua por meio de pesquisa histórica, ações de formação e, especialmente, registro de histórias de vida. Transforma a memória em estratégia de valorização das pessoas, fortalecimento institucional, desenvolvimento comunitário e pedagógico.

3.1. Núcleos

Além do Instituto Museu da Pessoa sediado em São Paulo, o Museu da Pessoa hoje é formado por mais quatro núcleos localizados em diferentes países:

- Núcleo Brasileiro do Museu da Pessoa – São Paulo, Brasil (<http://www.museudapessoa.net>).
- Núcleo Português do Museu da Pessoa – Braga, Portugal (<http://www.museu-da-pessoa.net>).
- Núcleo Museum of the Person – Indiana, Estados Unidos (<http://www.bloomington.us/~mop-j>).
- Núcleo Musée de la Personne – Montreal, Canadá (<http://www.museedelapersonne.ca>).
- Unidade de Memória – Minas Gerais, Brasil.

4. O Portal

O Portal hoje é fundamental para o Museu da Pessoa. O sistema desenvolvido em 2003 permite armazenar todo o acervo coletado onde, através de uma interface Web, os pesquisadores e coordenadores envolvidos nos projetos possam inserir e editar imagens, depoimentos, áudios, vídeos, etc. Todos esses dados são organizados e classificados de acordo com a metodologia definida pelo Museu.

4.1. Arquitetura de Conteúdo do Portal – 2003

A arquitetura de conteúdo planejada para o gerenciamento do acervo do primeiro portal desenvolvido em 2003 seguia algumas regras de classificação e relacionamento entre os diferentes objetos do acervo. A hierarquia adotada para a classificação do conteúdo do acervo tem como principal o projeto, que é a célula matriz, onde todo o acervo está inserido.

4.2. Motivos para a Reforma do Portal

Após alguns anos utilizando o portal, a equipe do Museu da Pessoa constatou que existiam alguns problemas na estrutura de conteúdo dos projetos, tornando-a inadequada. Dentre elas podem ser citadas:

- Uma pessoa ou instituição não podia se relacionar com projetos diferentes.
- Não podia existir uma pessoa sem relacionamento à um projeto.
- Uma pessoa cadastrada em um projeto por um pesquisador não podia se tornar um internauta, podendo assim editar sua própria história.

Outras necessidades não previstas na concepção inicial surgiram:

- Permitir relacionar diferentes histórias de vida.
- Existir a possibilidade do internauta poder comentar, traduzir ou colaborar com uma história.
- Possibilitar um usuário da área administrativa a acessar funcionalidades do perfil internauta, entre outras.

Também foi necessário uma melhoria nos aspectos ergonômicos do portal, melhorando a usabilidade do sistema e reduzindo os caminhos para execução de ações administrativas. Com isso, um novo contexto foi idealizado com o objetivo de eliminar os problemas existentes e implantar as novas necessidades ao sistema. Assim surgiu o projeto para reformular o portal.

Como a reestruturação das regras do sistema e do banco de dados seria muito trabalhosa, mantendo a tecnologia baseada em EJB-CMP e Struts, foi necessária uma reforma tecnológica. Essa reforma proporcionou mais flexibilidade ao sistema, permitindo assim ao Museu mais autonomia para desenvolver seus projetos virtuais.

A estrutura atual é muito dependente da JExperts, mesmo para pequenas mudanças, o que gera atraso nos projetos. Como o Museu é uma entidade sem fins lucrativos, fica inviável a contratação de serviços terceirizados.

4.3. A Nova Arquitetura de Conteúdo

A análise dos novos requisitos trouxe muitas mudanças para o sistema. O projeto deixa de ser a célula matriz de referência ao acervo e a pessoa se torna o foco principal de todo conteúdo. Os resultados das mudanças são:

- Todo o acervo relacionado à pessoa não tem relação com o projeto, somente com ela mesma.
- Uma pessoa pode ser relacionada à outra pessoa.
- Uma pessoa pode ter uma conta para acesso ao sistema com perfil de internauta que lhe permite desempenhar algumas funcionalidades no portal.
- Possibilidade dos internautas de criar e coordenar comunidades semelhantes ao conceito do Orkut.
- Permitir o relacionamento entre uma pessoa com nenhum ou vários projetos.

4.3.1. Indexação dos Depoimentos por XML

Através de uma linguagem especial de marcação em XML, configurável pela área administrativa do portal, pode-se classificar e contextualizar os depoimentos inseridos no sistema. Essa funcionalidade permite delimitar temas, citações, palavras-chaves de acordo com a metodologia do Museu.

Com o uso do XML nos depoimentos é possível criar metadados – dados capazes de descrever outros dados – sobre o texto, permitindo definir marcações de acordo com um vocabulário controlado, determinado por um esquema em XML. Cada projeto tem um arquivo de configuração com seu vocabulário controlado e marcações específicas. Existe um arquivo de configuração geral para o sistema, com definições essenciais para todos os projetos.

Depois do depoimento ser cadastrado com as marcações, ele pode ser visualizado através de um XML Stylesheet – folha de estilos – que transforma o conteúdo do documento XML em um formato de saída HTML. A indexação do conteúdo do depoimento estruturado por metadados a partir das marcações XML é realizado pelo sistema por uma ferramenta escrita em Java chamada Lucene. O Lucene é um recurso que oferece dois tipos principais de serviços: indexação e pesquisa de texto. Ele tem habilidade para criar e armazenar informações em um índice. Com ele é possível realizar buscas temáticas baseadas no vocabulário controlado do projeto e também transforma o conteúdo do documento em um formato de saída HTML, formatando os títulos e armazenando o depoimento já processado no banco de dados.

4.4. Reforma na Área Administrativa

A área administrativa do portal foi reformulada para melhorar sua ergonomia e implementar novas funcionalidades. Essa reformulação dá ao Museu da Pessoa mais autonomia para

criação de novos projetos e melhora também a navegabilidade do sistema. A seguir os principais requisitos implementados da área administrativa:

- Um menu central e hierárquico que permite o acesso rápido e direto à todas as funcionalidades do sistema.
- Possibilidade de criar perfis e configura-los de acordo com as permissões de acesso desejadas.
- Permite associar os usuários com os perfis cadastrados no sistema e também personalizar as permissões de cada usuário.
- Os usuários cadastrados só poderão ser removidos por um usuário com perfil de administrador.
- Permite bloquear e desbloquear o acesso ao sistema de qualquer usuário.
- Todos os campos de seleção dos formulários são configuráveis, suas opções são cadastradas pela interface administrativa.
- Existem ferramentas para criar templates que permite a visualização e consulta pública do acervo do portal, esses templates podem facilmente ser alterados e customizados.
- Permite enviar um arquivo compactado em formato ZIP contendo a estrutura de um template para associar à um projeto.
- Todo conteúdo do acervo pode ser moderado com três diferentes estados: aprovado, reprovado e pendente.
- Permite a configuração de campos específicos para os formulários de conteúdo de cada projeto (ficha de pessoa e as fichas dos acervos relacionados), além dos campos já previstos nos formulários padrão.
- A cessão de direitos é feita somente na ficha da pessoa estendendo-se pra todo o conteúdo referente a ela.
- Todas as listas de conteúdo da área administrativa contém opções para busca com filtros.
- Existe uma seção de busca avançada que permite filtrar o conteúdo com mais precisão.
- Os formulários são preenchidos com um único passo.

5. Desenvolvimento

O processo de desenvolvimento utilizou a linguagem padrão de modelagem UML, que permite ao engenheiro especificar, projetar, implementar e manter sistemas de software de qualidade. O processo deu-se por várias etapas: levantamento e análise de requisitos, definição dos atores e das funcionalidades do sistema, construção do protótipo e criação do diagrama de classes. Com o projeto modelado foi possível iniciar a implementação do código fonte através do framework preparado.

5.1. Tecnologias e Ferramentas Utilizadas

Um dos requisitos do projeto é que o código fonte do sistema seja liberado para a equipe do Museu da Pessoa para que ele possa estudado e, se necessário, alterado. Com essa premissa foi desenvolvido um framework composto somente de softwares livres, ou seja, programas de computador que podem ser usados, copiados, estudados, modificados e redistribuídos sem nenhuma restrição.

A linguagem de programação escolhida para desenvolver o portal foi Java, devido à sua portabilidade, segurança e seu vasto conjunto de bibliotecas. Isso permitiu arquitetar um framework com diversas tecnologias e conceitos em Java o que agilizou o andamento do projeto. A seguir serão citadas as ferramentas e tecnologias que compõem o framework:

- Model-View-Controller (MVC)
- Spring, Hibernate, Struts
- Ajax, CVS, JSP
- Lucene, Tomcat, Ant
- Taglibs, PostgreSQL

6. Considerações Finais

Após aproximadamente um ano de trabalho, é com muita satisfação que a equipe chega ao final do projeto alcançando todos os objetivos propostos na sua concepção. O estudo das tecnologias, o gerenciamento adequado das atividades, a ajuda e seriedade da equipe foram responsáveis pelo bom resultado.

O framework utilizado foi fundamental para o desenvolvimento do projeto. Os conceitos de funções genéricas, reutilização de código e scripts para automatizar a criação de arquivos do sistema foram fundamentais para aumentar a produtividade. A nova ergonomia do portal melhorou consideravelmente a sua usabilidade, o menu principal dá acesso rápido à todas as funcionalidades disponíveis. O cadastro de perfil é flexível e possibilita dar permissões personalizadas aos usuários.

O novo contexto para a estrutura de conteúdo foi implementado de acordo com os requisitos. O projeto deixou de ser a célula matriz de todo o acervo, e a pessoa se tornou o foco principal do sistema. A sessão de direito é assinada uma única vez e vale para todo o acervo. As histórias podem ser escritas usando as marcações em XML que transformam seus conteúdos em formato HTML. Existem vários tipos de buscas que permitem encontrar de tudo no banco de dados.

A facilidade de gerenciar as páginas estáticas e dinâmicas com as ferramentas disponibilizadas garante a autonomia que o Museu da Pessoa precisava para a criação e manutenção das salas. As taglibs desenvolvidas são poderosas, pois permitem aos webdesigners do Museu efetuarem consultas e escritas no banco de dados do sistema diretamente pelas tags inseridas nos arquivos JSP dos templates. As páginas estáticas têm seus conteúdos facilmente alterados por uma interface simples de edição.

O Instituto Museu da Pessoa é uma organização sem fins lucrativos, que há 16 anos coleta histórias de vida das pessoas. Sua missão é realizar a construção de uma identidade cultural da sociedade. A substituição do portal trará muitos benefícios para a instituição, as

novas ferramentas aumentarão a produtividade do Museu e darão total autonomia para o desenvolvimento de seus projetos. São muitas as possibilidades de projetos possíveis de desenvolver utilizando as novas ferramentas.

A finalidade do portal Museu da Pessoa é educativa. Existe a necessidade de um apoio geral das instituições educacionais do Brasil. A criação de novos agentes formadores para continuar a ideologia do Instituto e maior divulgação que é fundamental para seu crescimento.

Referências Bibliográficas

ASHOKA EMPREENDEDORES SOCIAIS. **Fique por Dentro de Quem está Mudando o Mundo**. Disponível em: <<http://www.ashoka.org.br/fellow.php?acao=visualizar&id=854>>. Acesso em: 30 ago. 2007.

MAANEN, John, Van. **Reclaiming qualitative methods for organizational research: a preface**, In *Administrative Science Quarterly*, vol. 24, no. 4, Dezembro 1979 a, p 520-526.

MUSEU DA PESSOA. **Nossa História**. Disponível em: <http://www.museudapessoa.net/oquee/oque_nossahistoria.shtml>. Acesso em: 30 ago. 2007.

PAULINO, Maria Ângela Silveira. **A Pesquisa Qualitativa e a História de Vida**. Disponível em: <http://www.ssrevista.uel.br/c_v2n1_pesquisa.htm>. Acesso em: 20 ago. 2007.