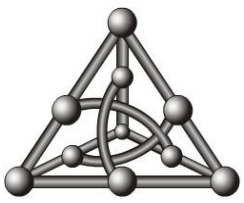
UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL

PANTANEIRO: FRAMEWORK DE APLICAÇÕES WEB PARA PLATAFORMAS E-GOV

HERCULES DA COSTA SANDIM

Dissertação de Mestrado em Ciência da Computação

Área de Concentração: Engenharia de Software



Departamento de Computação e Estatística

HERCULES DA COSTA SANDIM

PANTANEIRO: FRAMEWORK DE APLICAÇÕES WEB PARA PLATAFORMAS E-GOV

Dissertação apresentada como trabalho final para obtenção do título de Mestre em Ciência da Computação, Curso de Pós-Graduação em Ciência da Computação, Departamento de Computação e Estatística da Fundação Universidade Federal de Mato Grosso do Sul.

Orientador: Prof. Dr. Marcelo Augusto Santos Turine

Junho, 2009

Agradecimentos

Ao longo desses anos de estudo tenho muito a agradecer. Primeiramente, a Deus que nunca me deixou fraquejar e desistir.

Agradeço muito meus pais, que desde criança só fizeram me incentivar e mostrar a importância do estudo e aperfeiçoamento contínuo, os únicos que acompanharam de perto, e com muita preocupação, as inúmeras noites mal dormidas (ou não dormidas), e que a cada dia são mais e mais recompensadas.

À Maiany, minha esposa, pelo apoio incondicional em todos os momentos, apresentando sempre uma palavra de incentivo e carinho e à minha enteada, Kethellyn, que nos últimos meses trouxe uma imensa felicidade em nosso lar.

Ao meu orientador, Prof. Dr. Marcelo Augusto Santos Turine que com apoio, disposição e firmeza conduziu este trabalho. Obrigado pela amizade e confiança sempre.

A Profª Drª. Débora Maria Barroso Paiva e a Profª Drª Maria Istela Cagnin pelas contribuições no trabalho.

A todos os amigos, colegas e professores, tais como o Prof. Msc. Cristiano Costa Argemon Vieira, Prof. Msc. Camilo Carromeu, Msc. Carlos Juliano Moura Viana e Marcio Roberto Silva, que direta ou indiretamente contribuíram para a realização deste trabalho.

A equipe de TI da SGI, que colaborou bastante com o desenvolvimento deste trabalho.

A Fundação de Apoio ao Desenvolvimento do Ensino, Ciência e Tecnologia do Estado de Mato Grosso do Sul (FUNDECT), pelo auxílio financeiro concedido para realização do mestrado.

“A persistência é o caminho do êxito.”

Charles Chaplin

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AJAX	*Asynchronous Javascript And XML*
API	*Application Programming Interface*
CASE	*Computer-Aided Software Engineering*
CGI	*Common Gateway Interface*
CRUD	*Create, Retrieve, Update and Delete*
CSS	*Cacading Style Sheets*
DBPL	*Database Programming Language*
E-gov	Governo Eletrônico
EORM	*Enhaced Object Relationship Model*
HDM	*Hypermedia Design Method*
HMBS	*Hypermedia Model Based on Statecharts*
HMBS/M	*Hypermedia Model Based on Statecharts/Method*
HTML	*Hypertext Markup Language*
HySCharts	*Hypermedia System based on Statecharts*
IBGE	Instituto Brasileiro de Geografia e Estatística
IHC	interação Humano-Computador
JSP	*Java Server Pages*
LEDES	Laboratório de Engenharia de Software
NCSA	*National Center for Supercomputing Applications*
ODBC	*Open Data Base Connectiviy*
OO-H METHOD	*Object-Oriented Hypermedia Method*
OOHDM	*Object-Oriented Hypermedia Design Method*
PHP	*PHP: Hypertext Preprocessor*
PREG	Pró-Reitoria de Ensino de Graduação
PUC/RJ	Pontifícia Universidade Católica do Rio de Janeiro
RDF	*Resource Description Framework*
RMDM	*Relationship Management Data Model*
RUP	*Rational Unified Process*
SCXML	*Statescharts XML*
SDMX	*Statistical Data and Metadata Exchange*
SERPRO	Serviço Federal de Processamento de Dados

SGI	Superintendência de Gestão da Informação
SGBD	Sistema Gerenciador de Banco de Dados
SGBD-R	Sistema Gerenciador de Banco de Dados Relacional
SGML	*Standard Generalized Markup Language*
SINAES	*Sistema Nacional de Avaliação da Educação Superior*
SMIL	*Synchronized Multimedia Integration Language*
SSI	*Server-Side Includes*
SVG	*Scalable Vectorial Graphics*
MathML	*Mathematical Markup Language*
MVC	*Model-View-Controller*
MIT	*MassachussettsInstitute of Technology*
NCL	*Nested Content Language*
TERESA	*Transformation Environment for Interactive Systems Representations*
TI	Tecnologia da Informação
TIC	Tecnologias da Informação e Comunicação
UFMS	Universidade Federal de Mato Grosso do Sul
UML	*Unified Modeling Language*
UWE	*UML based Web Engineering*
W3C	*World Wide Web Consortium*
WebApps	Aplicações baseadas na Web
WebML	*Web Modeling Language*
WSDM	*Web Services Distributed Management*
WWW	*World Wide Web*
XBRL	*eXtensible Business Reporting Language*
XHTML	*eXtensible Hypertext Markup Language*
XML	*eXtensible Markup Language*
XSIL	*eXtensible Scientific Interchange Language*
XSLT	*eXtensible Stylesheet Language Transformations*

PANTANEIRO: FRAMEWORK DE APLICAÇÕES WEB PARA PLATAFORMAS E-GOV

RESUMO

O governo, em suas diferentes esferas (municipal, estadual e federal), tem o desafio de utilizar a Tecnologia de Informação e Comunicação para compartilhar informações, dar maior transparência na gestão pública e melhorar a qualidade de prestação de serviços ao cidadão. Com o avanço da tecnologia, o governo eletrônico (e-gov) é uma das principais estratégias para tornar disponível serviços online ao cidadão, criar e fortalecer uma área de interação e participação entre governo e sociedade. Neste contexto, o presente trabalho tem como objetivo especificar e implementar um framework intitulado Pantaneiro para facilitar e agilizar o desenvolvimento de portais corporativos em uma plataforma e-gov. O framework é baseado em técnicas de reuso e fornece suporte ao método orientado a objetos HMBS/M (*Hypermedia Model Based on Statecharts/Method*), que tem como principal característica o uso do formalismo *Statecharts* para modelar o comportamento de uma aplicação Web (WebApp). Para instanciar os portais por meio do framework Pantaneiro, foi desenvolvido o Wizard-Pantaneiro, composto pelos ambientes de Autoria, de Projeto Navegacional e de Publicação fornecendo suporte às quatro fases do método subjacente: modelagem conceitual, modelagem navegacional, modelagem da interface e publicação/teste. Durante o processo de modelagem da aplicação, o framework incentiva o desenvolvimento incremental e iterativo, sendo que, em cada fase, modelos são construídos ou enriquecidos. Para simular o comportamento navegacional dos portais, o Wizard-Pantaneiro possui um módulo que gera a especificação em *Statecharts* da aplicação em XML (padrão SCXML/W3C) para ser visualizada no simulador de *Statecharts*, denominado SCXML Viewer–Pantaneiro, desenvolvido neste trabalho. Para validar a proposta do framework, é apresentada a experiência da equipe de TI da Superintendência de Gestão de Informação (SGI) do Governo do Estado de Mato Grosso do Sul que está usando a 1ª versão do framework Pantaneiro e já instanciou 54 portais de secretarias e/ou órgãos do governo do estado de Mato Grosso do Sul.

Palavras-chave: governo eletrônico, framework, Engenharia Web.

PANTANEIRO: WEB APPLICATIONS FRAMEWORK FOR E-GOV PLATFORM

ABSTRACT

The government, in their different spheres (local, state and federal), has the challenge to use Information and Communication Technologies to share information, provide greater transparency in public administration and improve the quality of services for the citizens. With advance of technology, electronic government (e-gov) is a major strategy to provide online services for citizens, create and strengthen an interaction and participation area between government and society. In this context, this work aims to specify and implement a framework entitled Pantaneiro, to facilitate and expedite the development of corporate portals in an e-gov platform. The framework is based on reuse techniques and provides support for object oriented method HMBS/M (Hypermedia Model Based on Statecharts / Method), whose main characteristic is the use of Statecharts formalism for modeling the behavior of a Web application (WebApp). To instantiate the portals through the Pantaneiro framework, the Wizard-Pantaneiro was developed. It is composed of the environment Author, Navigational Design and Publication, providing support to the four phases of the underlying method: conceptual modeling, navigational modeling, modeling of the interface, and publication / test. During the modeling implementation process, the framework encourages incremental and iterative development, where at each stage, models are built or enriched. To simulate the navigational behavior of e-gov portal, the Wizard-Pantaneiro has a module that generates a specification in Statecharts implementation of XML (default SCXML/W3C) to be displayed on the Statecharts simulator, called SCXML Viewer-Pantaneiro, also developed in this work. To validate the proposed framework, the experience of the Superintendent of Management Information (SGI) IT team of the Government of the State of Mato Grosso do Sul is presented, which is using the 1st version of the framework Pantaneiro and has 54 instances of portals desks and / or government agencies in the state.

Key-words: electronic government, framework, Web Engeneering.

Introdução

1.1 – CONSIDERAÇÕES INICIAIS

É inegável o crescimento acelerado e a relevância das Tecnologias de Informação e Comunicação (TIC) nos últimos anos. De acordo com Glover et al. (2002), Tarapanoff (2001) e Koh et al. (2005) as mudanças que vêm ocorrendo nas organizações convergem para a quebra de um paradigma histórico e, por meio dele, configura-se um novo estágio do desenvolvimento social: a sociedade da informação e do conhecimento. A informação como matéria-prima das organizações é um insumo comparável à energia que alimenta uma organização; o conhecimento é utilizado na agregação de valor a produtos e serviços; a tecnologia constitui um elemento vital para as mudanças.

A internet, como uma das tecnologias emergentes, trouxe enormes possibilidades de mudança social. Motivou o crescimento de uma nova geração de softwares na Web denominados WebApps[1] (*Web based Applications*), tornando-se um dos meios mais efetivos e atrativos de divulgação, comunicação, negociação e disponibilização de bens e serviços (BALASUBRAMANIAN *et. al.*, 1997, BIEBER *et. al.*, 1998, CONALLEN, 2000, GLOVER *et. al.*, 2002; SILVA; FREITAS, 2006).

As organizações, ou mais especificamente os governos, estão buscando diferenciais e vantagens sustentáveis em seu ambiente de negócio. As TICs têm se espraiado pelo setor governamental por meio do que se chama e-gov ou governo eletrônico, que representa a informatização das atividades internas do governo e sua

[1] A sigla WebApp corresponde neste texto a denominações diversas encontradas na literatura, como aplicação hipermídia na Web, hiperdocumento, hipertexto, hipermídia, hiperbase, site e Website.

comunicação com o público externo: cidadãos, fornecedores, empresas, ou outros setores do governo e da sociedade (PINHO, 2008). A estratégia principal dessa informatização são os portais governamentais, por intermédio dos quais os governos mostram sua identidade, seus propósitos, suas realizações, torando disponível serviços e informações, facilitando a realização de negócios e o acesso à identificação das necessidades dos cidadãos.

Tais avanços estão fundamentados nas características intrínsecas que as novas TICs possuem, que permitem e aceleram a comunicação e a interação entre sociedade e governo. No âmbito da política, o e-gov objetiva tornar a máquina pública transparente, diminuindo os custos da burocracia, evitando corrupção e aumentando a confiança do cidadão no governo. Na esfera econômica, possibilita o aumento da arrecadação tributária pela simplificação das declarações eletrônicas. De forma complementar, a sociedade tem demandado que o acesso a serviços e informações públicas seja cada vez mais facilitado, integrado, racional e menos oneroso. O desafio de cumprir tais expectativas é parte do conjunto de resultados que se espera de projetos de governo eletrônico. Enfim, para os governos, são vários os benefícios da existência de ferramentas e tecnologias para a adequação ao processo e-gov.

Em relação as pesquisas em Computação e Informática, a Engenharia Web por meio do tema e-gov ingressou nas agendas governamentais com grande visibilidade, indicando a importância do uso das modernas TICs na administração pública. Por exemplo, o portal de serviços e informações do Governo Brasileiro, o Rede Governo[2], oferece ao usuário uma variedade de aplicações que permitem realizar transações, tais como, declaração do imposto de renda, emissão de certidão de pagamento de impostos, denúncias, compras eletrônicas, abertura e acompanhamento de processos previdenciários, dentre outros.

Diante deste panorama e das tendências tecnológicas, várias pesquisas estão sendo realizadas a fim de propor modelos, técnicas, métodos e ferramentas para auxiliar o processo de desenvolvimento de portais e-gov e, conseqüentemente, uma gestão de conhecimento compartilhada e efetiva das informações. Surge, assim, oportunidades na área de Engenharia de Software para investigar arquiteturas e-gov de forma gradual e consistente, constituindo-se em uma plataforma para facilitar o

[2] Acesse o endereço http://www.redegoverno.gov.br.

reuso de soluções computacionais e antever a realização de estágios mais avançados de e-gov.

Assim, no presente projeto é utilizado métodos e modelos de especificação de WebApps para facilitar a geração automática de portais corporativos em uma plataforma e-gov, permitindo o reuso de um conjunto de produtos de software com características similares e parametrização de suas diferenças.

1.2 – MOTIVAÇÕES

A acelerada evolução das TICs e sua aplicação nos mais diversos contextos sociais têm se apresentado como verdadeiros desafios. Na última década, principalmente, tornou-se evidente o desafio proposto pelas tecnologias na gestão pública, surgindo vários esforços de como transformar o governo e de como governar (FREY, 2000).

As principais motivações para o desenvolvimento deste projeto são:

(1) retrato estatístico da pequena quantidade de portais e-gov no Brasil;

(2) necessidade de softwares livre para auxiliar na construção e gestão de portais e-gov;

(3) pesquisa desenvolvida em parceria com a Secretaria de Gestão da Informação (SGI) do governo do Estado de Mato Grosso do Sul;

(4) pesquisa em tecnologias ou modelos de reuso para o desenvolvimento de WebApps; e

(5) necessidade de modelos e ferramentas para auxiliar o processo de desenvolvimento de WebApps.

De acordo com os dados quantitativos do Instituto Brasileiro de Geografia e Estatística (IBGE), o Brasil possui hoje 5.564 municípios (IBGE, 2009), e apenas 25% destes apresentam portais e-gov. Se considerar apenas acesso à internet, entre os municípios com população de até 20 mil habitantes, o acesso a internet está em apenas 5,5%. Na medida em que o número de habitantes aumenta, cresce o número de municípios que tem acesso à internet. Enquanto cerca de 70% dos municípios com mais de 50 mil habitantes podem acessar a internet, todos os municípios com mais de 1 milhão de habitantes possuem tal serviço.

No final de 2006 e início de 2007, o IBGE fez um levantamento inédito da existência de planos ou políticas de inclusão digital nos municípios (MUNIC, 2006). A pesquisa identificou que em metade deles (52,9%) foi implantada uma iniciativa

deste tipo. A preocupação com a inclusão digital esteve presente em 33 dos municípios com mais de 500 mil habitantes (91,7%), índice superior à média nacional (52,9%). Entre as grandes regiões, o destaque foi para o Sul (59,4%), seguido do Sudeste (57,9%) e Centro-Oeste (52,6%). O Norte e o Nordeste ficaram abaixo da média nacional, com 35,6% e 48,4%, respectivamente. A pesquisa também investigou a implantação de pontos de inclusão digital, ou telecentros, essenciais ao desenvolvimento de redes. Em todo o país, a rede pública municipal de ensino foi a que mais concentrou computadores em rede, visando à inclusão digital (61,8%). Dentre as prefeituras com plano ou política de inclusão digital, 45,7% optaram pela criação de telecentros, e 40,7% disponibilizaram computadores com acesso à internet para uso do público em geral.

Desta forma, nos últimos anos o desafio é a implantação de portais e-gov na internet. Porém, é fundamental oferecer ferramentas simples e de custo reduzido para as unidades gestoras e técnicas dos governos para auxiliar no processo de implantação de plataformas e-gov. Neste contexto, surgiram inúmeras iniciativas de software livre para e-gov, dentre elas:

- **OpenACS**: framework de desenvolvimento Web para construir aplicações que suportam comunidades (OpenACS, 2009; CALVO *et al.*, 2006);
- **e-Proinfo**: ambiente colaborativo de aprendizagem desenvolvido pela Secretaria de Educação a Distância do Ministério da Educação (PHITON; BROCHADO, 2006);
- **i3Geo**: software livre para disponibilização de dados ao público aliados a um conjunto de ferramentas de navegação, geração de análises, compartilhamento e geração de mapas sob demanda (I3GEO, 2009);
- **LightBase**: banco de dados textual multimídia que abrange um ambiente de desenvolvimento rápido de aplicações e um servidor tridimensional para recuperação textual, possibilitando um rápido acesso a qualquer informação da base de dados (LIGHTBASE, 2009);
- **i-Educar**: gerenciador de informações do sistema educacional municipal, que diminui a necessidade de uso de papel, a duplicidade de documentos, o tempo de atendimento ao cidadão, racionalizando o trabalho do servidor público. Com o i-Educar é possível controlar todo o cadastro de alunos, com seus dados pessoais, familiares e pedagógicos. Funcionalidades

como: matrículas, transferências, emissão de certificados e diplomas, suspensões, quadro de horários, são realizados de forma integrada (I-EDUCAR, 2009);

- **Ases**: ferramenta que permite avaliar, simular e corrigir a acessibilidade de páginas, websites e portais, sendo de grande valia para os desenvolvedores e publicadores de conteúdo (ASES, 2009); e
- **Banco de Talentos**: desenvolvido para mapear os talentos da Câmara dos Deputados e identificar o potencial humano da instituição, de modo a facilitar uma análise contínua da evolução funcional, por meio da disponibilização de informações prestadas pelos próprios servidores (TALENTOS, 2009).

Em 2005, o governo do Estado de Mato Grosso do Sul iniciou o projeto Pantaneiro[3] a fim de definir uma política de padronização do portal governamental. Cada órgão do governo tinha autonomia para desenvolver seu portal, o que aumentava o gasto público, além de utilizar tecnologias distintas, dificultando e impedindo qualquer estratégia de integração das informações produzidas. Inúmeras informações e serviços eram publicados pelos órgãos de governo, porém não existia possibilidade de estabelecer comparações ou criar um gráfico de desvio de crescimento, além do uso incorreto, muitas vezes, de informações e imagens governamentais, da impossibilidade de prever quantas informações encontravam-se prescritas e ainda publicadas em duplicidade. Tudo isso devido à inexistência de diretrizes e padronizações de uma plataforma e-gov, não permitindo uma integração e consistência da base de dados entre os vários órgãos do governo.

Em 2005, o Laboratório de Engenharia de Software da Universidade Federal de Mato Grosso do Sul (LEDES/UFMS) e o governo do Estado de Mato Grosso do Sul iniciaram um projeto de pequisa a fim de propor ou adotar um framework para a instanciação e gerência das WebApps governamentais, com uma base de dados centralizada e um forte envolvimento de seus funcionários em uma gerência de conteúdo compartilhada entre os órgãos. Devido à dificuldade de uso, insuficiência de documentação e custo dos vários frameworks existentes, tais como Drupal

[3] Acesse o endereço http://www.sgi.ms.gov.br/frameworkpantaneiro/index.php

(DRUPAL, 2009), JOOMLA (JOOMLA, 2009), Microsoft CMS (MCMS, 2000), dentre outros, optou-se por definir um novo framework, que foi intitulado Pantaneiro.

Em 2006, iniciou-se a implantação da primeira versão do framework Pantaneiro no governo sul-matogrossense (TURINE *et. al.* 2005; TURINE *et. al.* 2006; SANDIM *et. al.* 2006). Porém, essa versão implantada não faz uso de um método de processo para o desenvolvimento do portal e-gov, além de dificuldades de reuso das WebApps geradas.

Embora o reuso de software seja uma das práticas mais indicadas da Engenharia de Software, a sua aplicação específica em e-gov é inexistente. Atualmente, técnicas de reuso de software (FRAKES; TERRY, 1995; BRAGA; MASIERO, 2001; NOGUEIRA, CAPRA, 2003; ANDRETO; AMARAL, 2006) estão sendo utilizadas para melhorar a produtividade, a manutenibilidade e a qualidade tanto do software quanto do processo do desenvolvimento em domínios gerais.

A busca por técnicas e ferramentas para auxiliar o projeto e o desenvolvimento ágil de WebApps com maior qualidade e em menor tempo é uma das preocupações da Engenharia de Software. Inúmeros métodos e ferramentas para o desenvolvimento de WebApps surgiram, tais como os métodos **HDM** (GARZOTTO *et. al.* 1995), **EORM** (LANGE, 1994), **RMM** (ISAKOWITZ *et. al.* 1995), **MATILDA** (LOWE; GINIGE, 1996), OO-Method (PASTOR *et al.* 1997), **HDM-lite** (FRATERNALI; PAOLINI, 1998), **OOHDM** (SCHWABE; ROSSI, 1998), **WSDM** (DE TROYER; LEUNE, 1998), **HMBS/M** (CARVALHO, 1998; CARVALHO *et al.* 1999), **HFPM** (KOCH, 1999), **WebML** (CERI *et al.* 2000), **OO-H Method** (GOMEZ *et al.* 2001), **ADM** (ATZENI; PARENTE, 2001), **UWE** (KOCH, 2001), **HMBS/M Estendido** (BRITO, 2003).

Além dos métodos, ferramentas automatizadas são essenciais para facilitar a utilização correta destes, auxiliando o processo de projeto e desenvolvimento de aplicações (CAVALLARO *et al.* 1993; GARZOTTO *et al.* 1994, 1995; KNAPP *et al.* 2003; WEBRATIO, 2009). Com essa necessidade, inúmeras ferramentas para desenvolvimento de WebApps foram desenvolvidas, tais como **RMCASE** (DIAZ *et al.* 1995), **HyScharts** (TURINE *et al.* 1999), **OOHDM-WEB** (SCHWABE *et al.* 1999), **WEBRATIO** (WEBRATIO, 2009), **ArgoUWE** (KNAPP, 2003), **VisualWADE** (GOMEZ *et al.* 2005), **WebScharts** (BRITO, 2003), entre outras.

O método HMBS/M Estendido (*Hypermedia Model Based on Statecharts/Method*) e a ferramenta WebSCharts são as referências teóricas iniciais deste projeto, pois são tecnologias propostas e em desenvolvimento pelo grupo de pesquisa da UFMS. Brito (2003) propôs várias extensões ao método HMBS/M, criando o HMBS/M estendido, que é constituído por quatro fases: modelagem conceitual, navegacional, de interface e publicação/teste. Essas fases devem ser realizadas segundo uma abordagem mista de desenvolvimento incremental e iterativo, de modo que os produtos de cada fase são construídos, incrementados ou enriquecidos na fase posterior. O método incentiva a utilização de protótipos no final de cada fase para facilitar a validação e verificação dos requisitos e é automatizado por meio da ferramenta WebScharts.

1.3 – OBJETIVOS

O objetivo principal deste trabalho é especificar e implementar o framework intitulado Pantaneiro para especificar, instanciar e gerenciar WebApps em uma plataforma e-gov institucional. O Wizard-Pantaneiro é necessário para facilitar e agilizar o processo de instanciação dos portais e-gov. A proposta tem como diferencial o uso do formalismo Statecharts, subjacente ao método HMBS/M estendido, para modelar o comportamento das WebApps, além de técnicas para facilitar o reuso das aplicações desenvolvidas.

O framework Pantaneiro, juntamente com o Wizard-Pantaneiro, vem suprir uma necessidade do governo eletrônico brasileiro, que é a de ser uma solução de software livre completa para a criação de portais e-gov desburocratizados (desburocratizando o processo de desenvolvimento e o resultado das aplicações instanciadas), transparentes, com compartilhamento de informações, com um repositório de componentes e-gov, um gerenciador de conteúdo baseado em necessidades reais do contexto e-gov e que contempla o desenvolvimento de WebApps desde a concepção, passando pela instanciação dos componentes, dos metadados e dos conteúdos, até a fase de publicação.

Os objetivos específicos deste projeto são:

- Conhecer as tecnologias existentes;
- Estudar e comparar as soluções existentes para o problema;

- Utilizar o HMBS/M estendido, proposto por Brito (2003), incorporando novos conceitos de portais e-gov para o framework Pantaneiro;
- Projetar e implementar uma arquitetura para o Wizard-Pantaneiro formado por cinco grandes módulos: Gerência de Componentes, Gerência de Conteúdo, Gerência de Permissões, Gerência de Interfaces, Gerência Navegacional e Instanciador de WebApps. Estes módulos são base para a execução das quatro fases do HMBS/M estendido: modelagem conceitual, navegacional, de interfaces e de publicação/teste da WebApp;
- Projetar e implementar um repositório de componentes e-gov a fim de facilitar o reuso em diferentes plataformas, tais como: Notícias, Agenda, Eventos, Galeria de Imagens, Fórum, Enquete, Banners, entre outros componentes a serem especificados;
- Especificar a modelagem comportamental das WebApps geradas no formalismo Statecharts, no padrão SCXML, possível de ser simulado usando o *Commons SCXML*, uma API JAVA para executar máquinas de estado; e
- Avaliar o uso do Pantaneiro no Governo do Estado de Mato Grosso do Sul por meio de experimentos e relatos de uso.

1.4 – ORGANIZAÇÃO DO TEXTO

Este texto está organizado em sete capítulos. No Capítulo 1 foram apresentados os problemas a serem investigados, as motivações e os objetivos do presente trabalho. No Capítulo 2 é apresentada uma visão geral sobre governo eletrônico, modelos de classificação existentes e plataformas e-gov brasileiras. O Capítulo 3 contém uma revisão da literatura sobre métodos e ferramentas de Engenharia Web. No Capítulo 4 é apresentada uma breve revisão da literatura sobre frameworks. No Capítulo 5 é apresentado o framework Pantaneiro e o Wizard-Pantaneiro, ferramenta utilizada para a instanciação. O Capítulo 6 apresenta a avaliação da proposta, apresentando os resultados da implantação no Estado de Mato Grosso do Sul, enquanto que o Capítulo 7 apresenta as contribuições e limitações do projeto, as dificuldades encontradas e os possíveis trabalhos futuros. Por fim, é apresentada a revisão bibliográfica. Os Anexos I, II, III e IV contém vários modelos de documento e

formulários utilizados no governo do Estado de Mato Grosso do Sul para a implantação de portais e-gov usando o Pantaneiro.

Governo Eletrônico

2.1– CONSIDERAÇÕES INICIAIS

Segundo Koh *et. al.* (2005) e Vilella (2003), e-gov pode ser definido como o uso da internet e das TICs para simplificar ou melhorar o método pelo qual cidadãos, funcionários, parceiros e outras organizações de governo interagem e realizam seus negócios.

Segundo Pinho (2008), dado o avanço da tecnologia, entende-se que o e-gov não deve ser visto apenas como meio de disponibilização de serviços online, mas também, como uma área de interação e participação entre governo e sociedade e pelo compromisso de transparência por parte dos governos. As TICs têm enorme potencial democrático, desde que haja definição política no sentido da participação popular e da transparência.

O governo eletrônico deve prover canais de comunicação adicionais entre governos, organizações e cidadãos, aglutinando a preocupação de atender a um conjunto mais complexo de agentes que operam nas esferas de influência cívica, política e de negócios, além de aumentar a transparência e participação da sociedade nas ações governamentais (GRANT; CHAU, 2005; PINHO, 2008). Essa rede de influências entrelaçada nas iniciativas de governo eletrônico, possibilitada pelo uso da internet e das TICs explicitam o papel potencialmente transformador do uso das tecnologias para a reinvenção da forma de governar.

Para viabilizar o compartilhamento de informações e o estabelecimento de espaços de cooperação, as soluções e-gov devem ser abertas, flexíveis, robustas e,

sobretudo, abranger o interesse de todos os atores envolvidos no domínio (PACHECO *et. al*., 2007).

Assim, o que parece se caracterizar como um novo desafio é, na realidade, a necessidade de atuação dos governos em um novo cenário, marcado por novas exigências impostas pelos cidadãos e, de forma acentuada, pela própria multiplicidade e velocidade de desenvolvimento das soluções tecnológicas, que acabam, por sua vez, impondo aos gestores públicos a realização de uma análise ainda mais criteriosa dos objetivos, estratégias e metas dos governos para o uso das inúmeras possíveis soluções tecnológicas (VILELLA, 2003).

Neste capítulo são apresentados os princípios e as diretrizes da Política de Governo Eletrônico do governo federal, modelos de classificação e-gov, além de exemplos de plataformas e-gov brasileiras.

2.2– PRINCÍPIOS E DIRETRIZES

No Brasil, a política de Governo Eletrônico[4] segue um conjunto de diretrizes que atuam em três frentes fundamentais: junto ao cidadão; na gestão interna; e na integração com parceiros e fornecedores. O Programa de Governo Eletrônico brasileiro tem como objetivo transformar as relações do governo com os cidadãos, empresas e também entre os órgãos do próprio governo de forma a aprimorar a qualidade dos serviços prestados; promover a interação com empresas e indústrias; e fortalecer a participação cidadã por meio do acesso a informação e a uma administração mais eficiente.

Em 2000, o Governo Brasileiro lançou as bases para a criação de uma sociedade digital ao criar um Grupo de Trabalho em Tecnologia da Informação (GTTI) Interministerial com a finalidade de examinar e propor políticas, diretrizes e normas relacionadas com as novas formas eletrônicas de interação.

Em 29 de outubro de 2003, a Presidência da República publicou um Decreto criando oito Comitês Técnicos de Governo Eletrônico com a finalidade de coordenar e articular o planejamento e a implementação de projetos e ações nas respectivas áreas de competência:

1. Implementação do Software Livre;
2. Inclusão Digital;

[4] Acesse o endereço http://www.governoeletronico.gov.br

3. Integração de Sistemas;
4. Sistemas Legados e Licenças de Software;
5. Gestão de Sítios e Serviços Online;
6. Infra-Estrutura de Rede;
7. Governo para Governo - G2G, e
8. Gestão de Conhecimentos e Informação Estratégica.

As sete diretrizes apresentadas a seguir para implantação e operação do Governo Eletrônico no âmbito dos Comitês Técnicos de Governo Eletrônico e de toda a Administração Pública Federal são referência para estruturar as estratégias de intervenção, adotadas como orientações para todas as ações de governo eletrônico, gestão do conhecimento e gestão da TI no governo federal (GOVERNO ELETRÔNICO, 2009).

1 - Prioridade do Governo Eletrônico é a promoção da cidadania

A política de governo eletrônico brasileiro tem como referência os direitos coletivos e uma visão de cidadania que não se restringe à somatória dos direitos dos indivíduos. Assim, incorpora a promoção da participação e do controle social e a indissociabilidade entre a prestação de serviços e sua afirmação como direito dos indivíduos e da sociedade.

Essa visão, evidentemente, não abandona a preocupação em atender as necessidades e demandas dos cidadãos individualmente, mas a vincula aos princípios da universalidade, da igualdade perante a lei e da eqüidade na oferta de serviços e informações. Neste contexto, pode-se considerar o framework Pantaneiro como um agente capaz de promover a cidadania, principalmente relacionado à desburocratização no momento em que as agências governamentais necessitam disponibilizar serviços e informações de forma transparente ao cidadão.

2 - Inclusão Digital é indissociável do Governo Eletrônico

A inclusão digital deve ser tratada como um elemento constituinte da política de governo eletrônico, para que esta possa configurar-se como política universal. Esta visão funda-se no entendimento da inclusão digital como direito de cidadania e, portanto, objeto de políticas públicas para sua promoção.

Entretanto, a articulação à política de governo eletrônico não pode levar a uma visão instrumental da inclusão digital. Esta deve ser vista como estratégia para construção e afirmação de novos direitos e consolidação de outros pela facilitação de acesso a eles. Além disso, enquanto a inclusão digital concentra-se apenas em indivíduos, ela cria benefícios individuais, mas não transforma as práticas políticas. Não é possível falar de práticas políticas sem que se fale também da utilização da TI pela sociedade civil em suas interações com os governos, o que evidencia o papel relevante da transformação dessas mesmas organizações pelo uso de recursos tecnológicos.

O framework Pantaneiro tem como objetivo impulsionar a inclusão digital, pois promoverá a cidadania e tornará disponíveis informações e serviços aos cidadãos, onde mais de 41,5 milhões têm acesso à internet (IBOPE/Net Ratings, 2008).

3 - Software Livre é um recurso estratégico para a implementação do Governo Eletrônico

O software livre deve ser entendido como opção tecnológica de governo. Para tanto, deve-se priorizar soluções, programas e serviços baseados em software livre que promovam a otimização de recursos e investimentos em tecnologia da informação. Entretanto, a opção pelo software livre não pode ser motivada somente por aspectos econômicos, mas pelas possibilidades que abre no campo da produção e divulgação de conhecimento, no acesso a novas tecnologias e no estímulo ao desenvolvimento de software em ambientes colaborativos.

O framework Pantaneiro é uma solução de software livre, disponível no sourceforge.net[5], um software de controle de desenvolvimento colaborativo capaz de criar e centralizar uma comunidade virtual de colaboradores para agregar novas soluções e enriquecer o framework.

4- Gestão do conhecimento é um instrumento estratégico de articulação e gestão das políticas públicas do Governo Eletrônico

A Gestão do Conhecimento é compreendida, no âmbito das políticas de governo eletrônico, como um conjunto de processos sistematizados, articulados e intencionais, capazes de assegurar a habilidade de criar, coletar, organizar, transferir

[5] Acesse o endereço http://pantaneiro.sourceforge.net

e compartilhar conhecimentos estratégicos que podem servir para a tomada de decisões, para a gestão de políticas públicas e para inclusão do cidadão como produtor de conhecimento coletivo.

O framework Pantaneiro possui um gerenciador de conteúdo que foi projetado de acordo com as necessidades do Governo do Estado de Mato Grosso do Sul, baseado em um workflow de publicação de conteúdo colaborativo. Assim, a gestão do conhecimento é organizada, onde cada gestor possui seu papel no processo de publicação, podendo agir de forma estratégica na publicação de informações.

5 - Governo Eletrônico deve racionalizar o uso de recursos

O governo eletrônico não deve significar aumento de dispêndios para o governo na prestação de serviços e em tecnologia da informação. Ainda que seus benefícios não possam ficar restritos a este aspecto, é inegável que deve produzir redução de custos unitários e racionalização do uso de recursos. Grande parte das iniciativas de governo eletrônico pode ser realizada por meio do compartilhamento de recursos entre órgãos públicos.

As WebApps geradas com o framework Pantaneiro conseguem facilmente compartilhar recursos entre si. Por exemplo, uma foto do Governador do Estado de Mato Grosso do Sul é disponibilizada apenas uma vez e referenciada em todos os locais necessários, diminuindo custos de desenvolvimento, custos com o armazenamento de informações, padronizando, unificando e centralizando a publicação de informações em WebApps de uma mesma plataforma e-gov.

6- Governo Eletrônico deve contar com um arcabouço integrado de políticas, sistemas, padrões e normas

O sucesso da política de governo eletrônico depende da definição e publicação de políticas, padrões, normas e métodos para sustentar as ações de implantação e operação do Governo Eletrônico.

O framework Pantaneiro foi desenvolvido para ser extensível as mais variadas realidades regionais e nacionais, mas os aspectos de padronização foi prioridade em seu projeto. Todas as WebApps possuem a mesma semântica e lógica de funcionamento, sendo personalizada apenas a forma de apresentação das informações. Com isso, há uma maior facilidade de integrar as políticas de governo

ao framework Pantaneiro, sem a necessidade de replicação manual das alterações necessárias.

7 - Integração das ações de Governo Eletrônico com outros níveis de governo e outros poderes

A implantação do governo eletrônico não pode ser vista como um conjunto de iniciativas de diferentes atores governamentais que podem manter-se isoladas entre si. Pela própria natureza do governo eletrônico, este não pode prescindir da integração de ações e de informações.

As aplicações instanciadas pelo framework Pantaneiro podem ser gerenciadas em quaisquer níveis de governo, dependendo das permissões. Por exemplo, um gestor de TI poderá gerenciar quaisquer aplicações da plataforma e-gov do Estado do Mato Grosso do Sul, que contempla secretarias de governo, órgãos estaduais da polícia, grupamentos dos Bombeiros, agências estaduais, centrais de compras, centros culturais, conselhos, defensorias, fundações, procuradorias, portais informativos, dentre outras aplicações.

2.3– MODELOS DE CLASSIFICAÇÃO e-GOV

A maioria dos modelos de classificação e-gov utiliza a teoria de estágios dos sistemas de informação, ou seja, as iniciativas e-gov passam por uma série de estágios e níveis de complexidade para se transformar em uma estrutura completa, madura e de qualidade (KOH *et al.* 2005).

Segundo Cruz (2002) existem basicamente cinco tipos de portais e-gov:

- **Portal generalista**: também conhecido como portal horizontal. Possui um grande volume de dados, informações e conhecimento coletados de uma grande variedade de fontes. Sua finalidade é atender ao maior número possível de necessidades. Um exemplo de portal generalista é o portal da UOL - Universo On line[6].
- **Portal vertical ou Vortal:** serve para criar e disponibilizar cadeias produtivas verticalizadas por tipo de negócio. São portais especializados em bens ou serviços de um único tipo. Esse tipo de portal pode conter informações sobre os sistemas de produção, com seus diversos

[6] Acesse o endereço http://www.uol.com.br

processos, desde a obtenção de matéria-prima até o produto final. Um exemplo de vortal é o Portal da Construção[7].

- **Portal de conhecimento**: esse tipo não é igual ao generalista, pois não existem dados e informações, mas somente conhecimento, ou seja, nele os dados e as informações já estão contextualizados. Um exemplo é o Portal do Conhecimento Nuclear[8].
- **Portal de Negócio**: esse tipo é específico para transações comerciais. Pode conter apenas uma das pontas da cadeia produtiva, isto é, ter apenas a relação cliente-empresa, empresa-fornecedor, empresa-parceiro ou pode contemplar a relação que vai dos fornecedores-parceiros-empresa-clientes aos acionistas, podendo ser vertical ou horizontal. Um exemplo é o Submarino[9].
- **Portal Composto:** esse portal pode ter todos os quatro tipos anteriores dentro de seus domínios. Um exemplo é o Mercado Livre[10].

Segundo a classificação de Cruz (2002), as WebApps instanciadas pelo framework Pantaneiro na plataforma e-gov podem ser caracterizadas como portais generalistas ou verticais. As secretarias do Governo do Estado de Mato Grosso do Sul concentram informações e serviços específicos de suas áreas de atuação, podendo ser classificadas em portais verticais, como por exemplo, a WebApp da Secretaria de Estado de Educação do MS[11]. Já a WebApp MS-GOV[12] é um portal generalista que concentra informações e serviços de todas as outras WebApps de sua plataforma e-gov.

Para cada tipo de portal existe um conjunto de ferramentas, softwares, módulos, linguagens de programação e sistemas que sustentam as funcionalidades criadas. Porém, como afirmam Terra e Gordon (2002), independentemente do tipo de portal, as necessidades básicas são as mesmas: “Todos os portais requerem uma infra-estrutura que possa crescer com a expansão do negócio, um potente e flexível ambiente de apresentação e que também permita o desenvolvimento de novos componentes com grande facilidade. Necessitam também, de características

[7] Acesse o endereço https://www.vortal.biz/econstroi
[8] Acesse o endereço http://portalnuclear.cnen.gov.br
[9] Acesse o endereço http://www.submarino.com.br
[10] Acesse o endereço http://www.mercadolivre.com.br
[11] Acesse o endereço https://www.sed.ms.gov.br
[12] Acesse o endereço https://www.ms.gov.br

avançadas de personalização, permitindo que o portal disponibilize informações relevantes para cada usuário, aumentando sua eficiência e produtividade".

Outra classificação para as iniciativas e-gov sintetizada por Pacheco *et. al* (2007) explicita dois tipos: uma por um nível de agregação de serviços (NEC3) (HOLMES, 2001) e outra por nível de interação governo-cidadão, proposta por Belanger e Hiller (2006). Os níveis de classificação NEC3 são:

- **Primeiro nível**: Portal que provê informação e esconde a complexidade organizacional, para mostrar o governo da forma que o cidadão quer vê-lo.
- **Segundo nível**: Portal que oferece transações online, como é o caso dos diversos serviços disponíveis no portal Rede Governo[13].
- **Terceiro nível**: Portal que permite ao cidadão passar de um serviço a outro sem identificação a partir da colaboração e compartilhamento de serviços entre diversos departamentos.
- **Quarto nível**: Portal que coleta a informação necessária para uma transação de todas as fontes governamentais disponíveis, requerendo colaboração entre diversas organizações, além de tecnologias de interconexão (*middleware*) e pré-processamento de informações analíticas (*data warehousing*).
- **Quinto nível**: Portal que permite ao cidadão ter acesso aos serviços governamentais segundo seus interesses, por exemplo, em vez de lhe impor o acesso a um departamento de veículos automotores, oferece-lhe um ícone "meu carro" no portal governamental com serviços amplos, incluindo licenciamento e pagamento de multas e seguro, informações sobre condições de tráfego, *recalls* relacionado ao modelo de seu veículo, alertas de datas de vencimento, dentre outros serviços, inclusive com saída em dispositivos móveis.

De acordo com os níveis de classificação NEC3, as WebApps instanciadas pelo framework Pantaneiro podem ser classificadas nos seguintes níveis:

- Primeiro Nível: portais simplesmente informativos, como por exemplo, a WebApp da Central de Compras do MS[14];

[13] Acesse o endereço http://www.redegoverno.gov.br
[14] Acesse o endereço http://www.centraldecompras.ms.gov.br

- Terceiro Nível: usuários autenticados podem ter acesso a quaisquer WebApps ou serviços da plataforma, por exemplo o Nosso Portal[15];
- Quarto Nível: portais que centralizam informações de várias WebApps, como por exemplo o Portal Notícias.MS[16].

A classificação proposta por Belanger e Hiller (2006) evidencia o potencial transformador da relação proporcionada por um projeto e-gov. Sendo assim, serviços de governo podem aumentar a participação do cidadão e a transparência da administração pública, como apresentada a seguir:

- **Informação**: Disseminação da informação do governo para os seus constituintes. É a forma mais simples de governo eletrônico, definida de forma semelhante ao primeiro nível do NEC3.
- **Comunicação de duas vias**: Neste estágio, é possível que os cidadãos se comuniquem com o governo e façam solicitações simples e mudanças cadastrais ou de outra natureza. É uma definição menos ambiciosa do que o segundo nível do NEC3.
- **Transação**: Neste estágio, governos possuem ambientes *online* disponíveis para transações com os seus cidadãos. Indivíduos interagem e conduzem transações com o governo completamente *online*, de forma semelhante ao que propõe o segundo nível do NEC3.
- **Integração**: Neste estágio, todos os serviços de governo estão integrados. Os cidadãos entram em um único portal que confere acesso a serviços independentemente da agência ou departamento ofertante. Esta definição leva o terceiro nível do NEC3 ao limite. A maior dificuldade para a integração é a fragmentação dos sistemas *online* e de *back-office.*
- **Participação política**: Dizem respeito aos portais que possibilitam a votação *online* e o encaminhamento de comentários acerca deste processo de forma completamente *online*. Segundo os autores, embora essas funções possam fazer parte do estágio Transação, têm importância que justifica sua caracterização como um estágio adicional.

Segundo a classificação de Belanger e Hiller, as WebApps instanciadas pelo framework Pantaneiro podem ser classificadas por WebApps de Informação (como

[15] Acesse o endereço http://www3.sefaz.ms.gov.br
[16] Acesse o endereço http://www.noticias.ms.gov.br

por exemplo, a WebApp Banco da Gente[17]) e ainda por WebApps de Comunicação de duas vias, pelo simples fato de oferecer serviços de Fale Conosco e Fórum (como por exemplo, a WebApp do Procon de MS[18])

Uma forma mais genérica de pensar as funcionalidades dos portais e-gov é proposta por Detlor (2000), quando define os portais como potencializadores de um "rico e complexo espaço de trabalho e de informação compartilhados, para a criação, troca, retenção e reutilização do conhecimento". No diagrama da Figura 1 são apresentados os três maiores componentes de um portal corporativo: conteúdo, comunicação e coordenação.

Figura 1 – Portal corporativo como um espaço de trabalho e informação compartilhados (DETLOR, 2000, p.93).

No que se refere ao **espaço do conteúdo**, destaca-se a capacidade de prover às organizações acesso a uma variedade de fontes de informação, dados estruturados e não-estruturados, sistemas legados, servidores Web, ou seja, fontes de dados heterogêneas. Em relação ao **espaço da coordenação**, os portais corporativos devem prover trabalho cooperativo e, para isso, os usuários precisam de funcionalidades que apóiem o fluxo automático de informações e as rotinas de trabalho de forma coordenada, assim como a facilidade de identificação de especialistas e profissionais que possam cooperar na realização de determinada

[17] Acesse o endereço http://www.bancodagente.ms.gov.br
[18] Acesse o endereço http://www.procon.ms.gov.br

tarefa. No **espaço da comunicação**, os portais devem prover canais de informação que ajudem os usuários a interagir em conversações e negociações com outros usuários.

A partir dos modelos de classificação citados anteriormente, Pacheco e Kern (2003) conceberam uma arquitetura conceitual para plataformas e-gov, que parte dos seguintes princípios e pressupostos:

- **Abrangência de Usuários**: levar em consideração os interesses de todos os atores ou interessados no domínio público ao qual o projeto atende. Esse risco é iminente em projetos cujos gestores públicos patrocinadores não fazem distinção entre a melhoria unilateral do órgão público que administram e a transformação do governo em provedor de informações e serviços ao cidadão.
- **Construção Colaborativa**: promover o trabalho colaborativo e a realização de projetos em rede, tanto em nível tecnológico (na definição e manutenção dos padrões das informações públicas) como operativo (na prática e uso dos serviços).
- **Perspectiva Internacional**: adotar padrões internacionais estabelecidos de forma colaborativa, de forma a permitir comparabilidade e extensibilidade.
- **Multi-Plataforma**: não impor a adoção de uma mesma tecnologia específica por todos os interessados, ou seja, ser flexível em nível tecnológico ao ponto de permitir troca de padrões técnicos e/ou conectividade com outras soluções.
- **Respeito aos Atores e Processos de Domínio**: não impor mudança nos princípios fundamentais de trabalho de qualquer ator do sistema. Ocorre tipicamente quando gestores públicos e/ou responsáveis técnicos têm na obrigatoriedade do uso do serviço o único recurso para que o mesmo seja executado.

O framework Pantaneiro se enquadra nos princípios e pressupostos de Pacheco e Kern, pois torna possível instanciar WebApps construídas de forma totalmente colaborativas, onde todos os servidores de governo são capazes de alimentar a WebApp com informações e contribuições, resguardando-se da autoria das mesmas. Além disso, é multi-plataforma e centraliza a geração das WebApps no

framework, o que torna fácil a adoção de padrões internacionais (como por exemplo, padrões de acessibilidade e usabilidade) e quaisquer alterações na estrutura das WebApps.

Os princípios e pressupostos anteriores estão em conformidade com os requisitos necessários a um arcabouço de governo eletrônico, segundo Grant e Chau (2005):

- Prover uma representação de governo eletrônico sem vieses que venham a beneficiar grupos de interesse;
- Possibilitar a identificação e articulação de objetivos e metas do governo eletrônico;
- Identificar o hiato entre o estado atual e futuro do uso do governo eletrônico;
- Prever tendências futuras que podem afetar as iniciativas de governo eletrônico;
- Ser transferível entre diferentes contextos de aplicação;
- Apoiar uma representação em sistema de agendas estratégicas e esforços de implementação;
- Prover uma representação funcional dos objetivos do governo eletrônico; e
- Apoiar a reusabilidade e a capacidade de expansão dos construtos do framework.

2.4– APLICAÇÕES e-gov

Uma das ações da Política de Governo Eletrônico Brasileira é a proposta da arquitetura e-PING (Padrões de Interoperabilidade de Governo Eletrônico) que define um conjunto mínimo de premissas, políticas e especificações técnicas que regulamentam a utilização da TIC no governo federal, estabelecendo as condições de interação com os demais Poderes e esferas de governo e com a sociedade em geral (e-PING, 2008).

Um governo moderno e integrado exige sistemas igualmente modernos e integrados, interoperáveis, trabalhando de forma íntegra, segura e coerente em todo o setor público. Nesse contexto, a interoperabilidade de tecnologia, processos, informação e dados é condição vital para o provimento de serviços de qualidade,

tornando-se premissa para governos em todo o mundo, como fundamento para os conceitos de governo eletrônico, o e-gov. A interoperabilidade permite racionalizar investimentos em TIC, por meio do compartilhamento, reuso e intercâmbio de recursos tecnológicos.

Alguns dos benefícios que a arquitetura e-PING pode trazer ao governo e à sociedade em geral são a unificação dos cadastros sociais, a unificação dos sistemas de segurança, a unificação dos Detrans, entre outros, levando em consideração ainda a existência de um legado de sistemas, de plataformas de hardware e software instaladas. Tem por meta a consideração de todos os fatores para que os sistemas possam atuar cooperativamente, fixando as normas, as políticas e os padrões necessários para consecução desses objetivos.

Neste contexto, todos os portais e-gov desenvolvidos deveriam tratar questões de interoperab ilidade. Exemplos de projetos de iniciativas e-gov que estão abordanto tais aspectos e referências na comunidade são: Plataforma Lattes[19] (LATTES, 2009), Portal Inovação[20] (INOVAÇÃO, 2009), o Portal SINAES[21] (SINAES, 2009), Plataforma da Receita Federal[22] (RECEITA, 2009), Rede Governo[23] (REDEGOVERNO, 2009), Portal Federativo[24] (FEDERATIVO, 2009), Portal MS-GOV[25] (MSGOV, 2009), dentre outros,

O Portal Inovação promove a cooperação entre os atores da cadeia de inovação e serve à gestão da informação e do conhecimento em inovação. O Portal SINAES é aplicado à gestão do sistema nacional de ensino superior. A Plataforma *Lattes* reúne informações e gera novos conhecimentos sobre o sistema científico-tecnológico. A Plataforma da Receita Federal, dentre outras aplicações, é uma plataforma que reúne funcionalidades como: a administração dos tributos internos e do comércio exterior; a gestão e execução das atividades de arrecadação, lançamento, cobrança administrativa, fiscalização, pesquisa, investigação fiscal e controle da arrecadação administrada; a gestão e execução dos serviços de administração, fiscalização e controle aduaneiro. O Portal Federativo foi

[19] Acesse o endereço http://lattes.cnpq.br
[20] Acesse o endereço http://www.portalinovacao.mct.gov.br/pi
[21] Acesse o endereço http://www.sinaes.gov.br
[22] Acesse o endereço http://www.receita.fazenda.gov.br
[23] Acesse o endereço http://www.redegoverno.gov.br
[24] Acesse o endereço http://www.portalfederativo.gov
[25] Acesse o endereço http://www.ms.gov.br

desenvolvido para agregar informações sobre os temas que envolvem os entes federados e as relações estabelecidas entre a União, os Estados, o Distrito Federal e os Municípios. O Portal MS-GOV foi desenvolvido para agregar informações sobre os temas que envolvem todas as secretarias de estado de Mato Grosso do Sul. O Portal MS-GOV foi desenvolvido com a primeira versão do framework Pantaneiro, em 2006, versão esta que está instalada nos servidores do Estado de Mato Grosso do Sul, servindo hoje 54 portais de secretarias e órgãos do Estado de Mato Grosso do Sul.

2.5 – CONSIDERAÇÕES FINAIS

Este capítulo abordou temas relacionados com WebApps no domínio e-gov, relacionando os princípios e diretrizes estabelecidos pelo governo brasileiro na construção de WebApps. Além disso, foi realizada uma revisão bibliográfica sobre os modelos de classificação e-gov segundo Cruz (2002), Holmes (2001) e Belanger e Hiller (2006) e Pacheco et al. (2007). Em cada um dos modelos, o framework proposto foi categorizado. Por fim foi elencada sete aplicações e-gov brasileiras, dando enfâse a plataforma e-gov do Estado de Mato Grosso do Sul, instanciada pelo framework Pantaneiro, com 54 portais em operacionalização.

Engenharia Web

3.1 – CONSIDERAÇÕES INICIAIS

Engenharia Web é o processo utilizado para especificar, sistematizar e implementar aplicações na internet de qualidade. É um novo paradigma, que define um processo para construir aplicações hipermídia na internet com base em critérios e requisitos básicos (NOGUEIRA; CAPRA, 2003). De acordo com Pastor (2004), o objetivo principal da Engenharia Web é desenvolver aplicações corretas, nas quais suas estruturas, funcionalidades, aspectos navegacionais e de interação com o usuário estejam representados de forma apropriada.

A necessidade de métodos para o projeto e desenvolvimento de WebApps está relacionada ao fato de que o processo de desenvolvimento deve ser estruturado, permitir o reuso e o rastreamento das decisões tomadas, além de facilitar a construção de ferramentas automatizadas (MURUGESAN *et al*. 1999). Segundo Graef e Gaedke (2000), o avanço da Engenharia Web deve promover um enfoque sistemático, disciplinado e quantificável para o desenvolvimento e evolução de WebApps, com alta qualidade e a um custo efetivo.

Assim, as atuais metodologias de Engenharia de Software estão sendo adaptadas para especificar as WebApps, que mudam e crescem rapidamente em seus requisitos, conteúdo e funcionalidade durante seu ciclo de vida (GINIGE; MURUGESAN, 2001).

Neste capítulo serão apresentadas metodologias, modelos e ferramentas que apóiam a especificação e implementação de WebApps, explorando principalmente o método HMBS/M Estendido e a ferramenta WebSCharts utilizadas neste trabalho.

3.2 – MÉTODOS DE APOIO

A busca por técnicas e ferramentas para auxiliar o projeto e o desenvolvimento ágil de WebApps com maior qualidade e em menor tempo é uma das preocupações da Engenharia de Software (CONALLEN, 2003), e inúmeros métodos e ferramentas para o desenvolvimento de WebApps existem, conforme Tabela 1.

Tabela 1 – Métodos de desenvolvimento de WebApps.

Sigla	Nome	Ano	Ferramenta Suporte	Referência
HDM	*Hypertext Design Model*	1993	-	(GARZOTTO *et al.* 1993; GARZOTTO *et al.* 1995)
EORM	*Enhanced Object-Relationship Model*	1994	-	(LANGE, 1994)
RMM	*Relationship Management Methodology*	1995	*RMCase*	(ISAKOWITZ *et al.* 1995)
MATILDA	*Multimedia Authoring Through Intelligent Linking and Directed Assistance*	1996	MATILDA *Prototype*	(LOWE; GINIGE, 1996)
OO- Method	*OO- Method*	1997	-	(PASTOR *et al.,* 1997)
HDM – lite	*Hypertext Design Model lite*	1998	-	(FRATERNALI; PAOLINI, 1998)
OOHDM	*Object Oriented Hypermedia Design Model*	1998	OOHDM-*Web*	(SCHWABE; ROSSI, 1998)
WSDM	*Web Site Design Method*	1998	-	(DE TROYER; LEUNE, 1998)
HMBS/M	*Hypermedia Model Based on Statecharts Method*	1998	*HysCharts*	(CARVALHO, 1998; CARVALHO *et al.* 1999)
HFPM	*Hypermedia Flexible Process Modeling*	1999	-	(KOCH, 1999)
WebML	*Web Modeling Language*	2001	*WebRatio*	(CERI *et al.* 2000)
OO-H Method	*OO-H Method*	2001	*VisualWADE*	(GÓMEZ *et al. 2*001)
ADM	*ARANEU Design Methodology*	2001	-	(ATZENI; PARENTE, 2001)
UWE	*UML-based Web Engineering*	2001	*ArgoUWE*	(KOCH, 2001)
HMBS/M Estendido	*Extended Hypermedia Model Based on Statecharts Method*	2001	*WebScharts*	(BRITO, 2003)
ICDM	*internet Commerce Development Methodology*	2002	-	(STANDING, 2002)
ADM	*Ariadne Development Method*	2004	-	(DIÁZ et al. 2004)

Analisando a Tabela 1 nota-se que os métodos HMBS/M, OOHDM, RMM, UWE, WebML, MATILDA, OO-H Method e HMBS/M Estendido possuem ferramentas que automatizam fases do processo de desenvolvimento. Mais recentemente, DIÁZ *et al.* (2004) propuseram o método ADM (*Ariadne Development Method)* que apóia as diferentes visões de modelagem hipermídia: Projeto Navegacional, de Apresentação, Estrutural, Comportamental, de Processos e de Segurança.

Nessa perspectiva, a fim de ilustrar o estado da arte nesta área, será apresentada a seguir uma breve descrição dos métodos HMBS/M Estendido, WebML, OO-H Method, UWE e OOHDM e RMM.

3.2.1 – HMBS/M ESTENDIDO

O método HMBS/M tem como objetivo principal apoiar o projeto e o desenvolvimento estruturado de WebApp (CARVALHO, 1998). Utiliza primitivas do método orientado a objetos Fusion (COLEMANN *et al.* 1994) para representar o domínio da aplicação e o modelo HMBS (TURINE *et al.* 1999; PAULO *et al.* 1999; OLIVEIRA *et al.* 2001) para a modelagem navegacional.

O modelo HMBS (*Hyperdocument Model Based on Statecharts*), foco deste trabalho, utiliza a técnica de especificação formal Statecharts (HAREL, 1987) para definir a estrutura organizacional e a semântica de navegação de uma aplicação Web. Permite separar as informações referentes à estrutura das representações físicas ou conteúdo da aplicação, possibilitando a descrição de aspectos de concorrência da aplicação.

Em 2003, Brito propôs mudanças/extensões ao método HMBS/M, que passou a ser denominado de HMBS/M Estendido. O processo geral do HMBS/M estendido, ilustrado na Figura 3, é constituído por quatro fases denominadas modelagem conceitual, modelagem navegacional, modelagem de interface e publicação/teste. Essas fases devem ser realizadas segundo uma abordagem mista de desenvolvimento incremental e iterativo, de modo que os produtos de cada fase são construídos, incrementados ou enriquecidos na fase posterior. A abordagem iterativa determina um incremento na compreensão do problema por meio de refinamentos sucessivos em diversos ciclos, permitindo ao projetista identificar e corrigir mais rapidamente as falhas de requisitos organizacionais e navegacionais da WebApp. O

método incentiva a utilização de protótipos no final de cada fase para facilitar a validação e verificação dos requisitos.

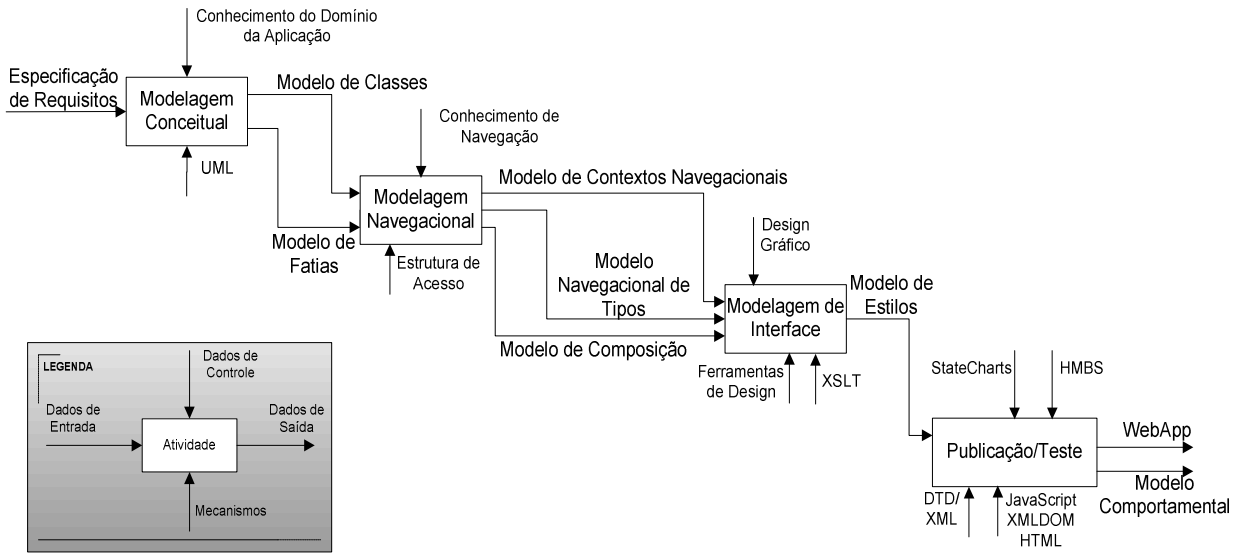


Figura 3 – Fases do Método HMBS/M Estendido (BRITO, 2003, p.29).

Na fase de modelagem conceitual, o objetivo é analisar o domínio da WebApp e representá-lo em dois modelos: de classes e de fatias. No modelo de classes são representadas as informações relevantes do domínio e o relacionamento entre elas. O modelo de fatias enriquece o modelo de classes com a primitiva de modelagem denominada fatia, que contém as unidades de informação da WebApp que serão apresentadas e exploradas durante a navegação. Tais modelos utilizam como base o documento de especificação de requisitos que define os objetivos e os requisitos necessários para a WebApp. O método não oferece técnicas e ferramentas para realizar a tarefa de especificação de requisitos.

Na fase de modelagem navegacional, os modelos da fase anterior são reorganizados, considerando os aspectos navegacionais da WebApp por meio da especificação de três modelos: de contextos navegacionais, navegacional de tipos e de composição. Na proposta original do HMBS/M não têm o modelo de composição. No modelo de contextos navegacionais são especificados os contextos de navegação que definem as possíveis estruturas de navegação a serem utilizadas na WebApp para orientar o usuário. Os contextos definidos e os diagramas das fases anteriores são combinados para a criação do modelo navegacional de tipos, que define os caminhos de navegação que orientam o usuário na WebApp.

Na modelagem de interface, é especificada e implementada a interface de cada uma das páginas da WebApp utilizando a linguagem XSLT (*Extensible Stylesheet Language Transformations*) (XSLT, 1999), uma das partes da família de recomendações da W3C para apresentação e transformação de documentos XML (XML, 2003). O conjunto de todos os templates de interface forma o modelo de estilos da WebApp.

A última fase do método é a de Publicação/Teste, na qual deve ser cadastrado o conteúdo da aplicação de forma estruturada, tendo como produto a WebApp povoada. Para a fase de implementação, o HMBS/M Estendido é automatizado pela ferramenta WebScharts (BRITO, 2003), descrita na seção 3.3.5.

3.2.2– WEBML

A WebML (*Web Modeling Language*) (CERI *et al.* 2000) é uma linguagem conceitual que suporta várias atividades do projeto de WebApp complexas, especificando as características em quatro modelos específicos: modelo estrutural, de hipertexto, de apresentação e de personalização. O modelo estrutural descreve a organização conceitual dos dados compatível com o MER e o diagrama de classes da UML. O modelo de hipertexto possibilita a definição das páginas e links dos hipertextos que definem a aplicação, sendo subdividido em modelos de composição e de navegação. O modelo de apresentação descreve o layout das páginas e, normalmente, está associado a uma linguagem de apresentação como, por exemplo, HTML. No modelo de personalização os usuários e os grupos de usuários são modelados em entidades pré-definidas, que são utilizadas para armazenar informações específicas de um usuário ou grupo.

A navegação e a abstração da composição são baseadas em um número restrito de componentes de hipertexto (units). Durante a implementação, as páginas e units são automaticamente traduzidas para templates por meio de scripts do servidor que permitem a representação dinâmica dos dados recuperados de diversas fontes.

Os principais objetivos da WebML são expressar a estrutura de uma aplicação Web em uma descrição de alto nível, que pode ser utilizada para pesquisa, evolução e manutenção; fornecer múltiplas visões do mesmo conteúdo; separar o conteúdo de informação dos aspectos de navegação e apresentação, podendo evoluir separadamente; e armazenar a estrutura da informação em um

repositório, que pode ser utilizado para a geração dinâmica da WebApp. Uma característica relevante é a existência da ferramenta comercial WebRatio (WEBRATIO, 2009), descrita na seção 3.3.1.

3.2.3– OO-H METHOD

O método *Object-Oriented Hypermedia Method* (GOMEZ *et al.* 2001) permite a criação de WebApps utilizando diagramas de classes padrões e, com base nestes, diagramas de navegação e diagramas de acesso para cada tipo de usuário. Usando estes diagramas de navegação, uma interface web padrão, destinada a prototipagem rápida, pode-se gerar uma prototipagem rápida.

Para WebApps mais sofisticadas, um diagrama de apresentação abstrato é derivado do diagrama de navegação, sendo posteriormente refinado pelo desenvolvedor da aplicação web. Essencialmente, o diagrama de apresentação se apresenta no mecanismo de templates.

Esta abordagem é interessante e deixa o desenvolvedor livre para criar passos, como usabilidade e testes com protótipos. A usabilidade é muito relevante pois permite a criação de diferentes diagramas de navegação para diferentes tipos de usuários. A utilização de padrões caminha no sentido correto, porém os modelos disponíveis são muito concretos, e todos os diagramas de apresentação poderiam ser facilmente substituídos por modelos mais tradicionais, como por exemplo as soluções SSI (*Server-Side Includes*) (ATTERER, 2005).

A ferramenta *VisualWADE,* que será descrita na seção 3.3.6, oferece excelente suporte para todo seu processo de desenvolvimento, incluindo a geração automática de protótipos.

3.2.4– UWE

O método UWE (*UML based Web Engineering*) (KOCH 2001) tem como foco principal a definição de um projeto sistemático, a personalização e a geração semi-automática de WebApp. O UWE é orientado a objeto, baseado em uma abordagem iterativa e incremental. Descreve uma metodologia de projeto que utiliza exclusivamente as técnicas, a notação e as extensões da UML.

O processo do UWE é baseado no RUP (*Rational Unified Process*) (KRUCHTEN, 2000) e incorpora todos os aspectos necessários ao desenvolvimento de WebApp. As fases de modelagem são análise de requisitos, modelagem conceitual, navegacional e de apresentação. Modelos de tarefas e Statecharts são

incluídos para modelar os aspectos dinâmicos da aplicação. Para a modelagem conceitual e navegacional são definidos *profiles* da UML, que permitem estender a notação para incorporar novos aspectos por meio de estereótipos. Um estereótipo possibilita estender a semântica para adicionar novas condições. Para o modelo conceitual é utilizado o modelo de classes da UML que representa graficamente a aplicação por meio de uma visão estática dos elementos do domínio.

A modelagem navegacional da WebApp é constituída de dois modelos: modelo navegacional de espaço e modelo navegacional de estrutura. O primeiro tem a função de especificar quais objetos podem ser alcançados pela navegação. O segundo define como estes objetos são alcançados. Para a fase de modelagem de apresentação é utilizado um tipo particular de diagrama de classes.

Na seção 3.3.4 é descrita a ferramenta CASE ArgoUWE (KNAPP, 2003) que suporta a fase de modelagem estrutural do método UWE.

3.2.5– OOHDM

O método OOHDM (*Object Oriented Hypermedia Design Method*) (SCHWABE; ROSSI, 1998), descendente do HDM (Hypermedia Design Method), faz uso de um processo de engenharia em cinco etapas bem distintas: levantamento de requisitos, modelagem conceitual, modelagem navegacional, projeto de interface abstrata e implementação. Em cada uma dessas etapas, um modelo novo é criado ou enriquecido. Entre todas as etapas previstas por este método é possível efetuar o rastreamento para trás ou para frente a fim de mapear as modificações de um modelo para o outro (ROSSI *et al.* 1997).

Devido à separação do processo de desenvolvimento em diversas etapas de modelagem, o OOHDM é indicado como forma de comunicação entre as várias pessoas envolvidas no processo de criação da aplicação como: projetistas, designers, desenvolvedores e usuários. O modelo produzido pelo OOHDM pode ser implementado em qualquer tipo de ambiente de desenvolvimento disponível no mercado, seja este orientado a objeto ou não (HENNRICHS; MAZZOLA, 2005).

Na seção 3.3.3 é descrito o ambiente OOHDM-Web (SCHWABE *et al.* 1999; PONTES, 1997) que permite a prototipagem rápida de aplicações hipermídias projetdas com o OOHDM.

3.2.6– RMM

O RMM é baseado no Modelo de Entidade Relacionamento (MER) (CHEN, 1976; ELMASRI, 2000) e no HDM (GARZOTTO; POLINI; SCHWABE, 1993; GARZOTTO; MAINETTI; PAOLINI, 1995). O núcleo do método é o modelo de dados RMDM (*Relationship Management Data Model*), que fornece uma linguagem gráfica para especificar a semântica do domínio de aplicação (via primitivas do domínio) e sua estrutura navegacional (via primitivas de acesso) (ISAKOWITZ; STOHR; BALASUBRAMANIAM, 1995; ISAKOWITZ; KAMIS; KOUFARIS 1997; DIAZ et al., 1995).

O RMM destina-se prioritariamente ao projeto de aplicações que possuem informações estruturadas e que necessitam de freqüentes atualizações. O projeto de aplicações Web, segundo o RMM, consiste em sete fases em um processo iterativo: projeto de entidade-relacionamento, projeto de entidade, projeto navegacional, projeto de interface com o usuário, projeto de protocolo de conversão, projeto de comportamento em tempo de execução e construção/teste. Cada fase gerencia objetos de projeto (entidades, relacionamentos, atributos, slices e estruturas de acesso) de forma particular, sendo estes não exclusivos de uma fase de desenvolvimento específica.

3.3 – FERRAMENTAS DE APOIO

Segundo Fraternali (1999), as ferramentas existentes para o desenvolvimento de *WebApps* podem ser agrupadas em seis grandes categorias:

- Gerenciadores de sites e editores visuais de HTML;
- Ferramentas de autoria hipermídia adaptadas para *Web*;
- Integradores Web-DBPL (*Database Programming Language*);
- Ferramentas de edição de formulários *Web*, geradores de relatórios e assistentes de publicação de banco de dados;
- Ferramentas multiparadigmas; e
- Geradores de aplicação dirigidos a modelo.

Os gerenciadores de sites e editores visuais de HTML como, por exemplo, *Macromedia DreamWeaver*, *Microsoft FrontPage*, *SoftQuad HotMetal* e *Adobe PageMill*, permitem uma edição visual (WYSIWYG) das páginas HTML, eliminando a

complexidade de programação. São úteis em pequenas e médias aplicações em que a utilização intensiva de banco de dados não é o principal requisito.

Na segunda categoria, as ferramentas de autoria hipermídia adaptadas para Web, tais como, *Asymetrix's Toolbook*, *Macromedia Director*, *Autoware* e *Flash*, possuem editores visuais HTML, porém com origem nas ferramentas existentes de publicação hipermídia *off-line*. Permitem a criação de interfaces sofisticadas com o usuário com alto grau de precisão na sincronização multimídia.

Os integradores Web-DBPL, *Allaire Cold Fusion*, *Microsoft Active Server Pages* (ASP) e *Personal Home Page* (PHP), permitem integrar informações Web com forte gerenciamento em banco de dados.

Na quarta categoria, as ferramentas de edição de formulários Web, geradores de relatórios e assistentes de publicação de banco de dados facilitam a migração de aplicações cliente/servidor baseadas em formulários Web. Exemplos dessa categoria são *Microsoft Visual InterDev*, *Borland IntraBuilder* e *Oracle Developer*.

As ferramentas multiparadigmas combinam ambientes sofisticados de edição visual com assistentes de gerenciamento de banco de dados como, por exemplo, *Lotus Domino* e *NetObject Fusion*.

Os geradores de aplicação dirigidos a modelo abrangem todas as atividades de desenvolvimento Web desde a análise até a implementação. As ferramentas WebRatio (WEBRATIO, 2009), RMCASE (DIAZ *et al.* 1995), OOHDM-Web (SCHWAB *et al.* 1999), ArgoUWE (KNAPP, 2003), *HySCharts* (TURINE, 1998) e *WebScharts* (BRITO, 2003) são apresentadas a seguir.

3.3.1– WEBRATIO

O WebRatio (WEBRATIO, 2009) é uma ferramenta CASE visual para especificação de WebApp utilizando o método WebML. Gera a aplicação nas plataformas J2EE e/ou *Microsoft* .NET.

A ferramenta provê uma interface para a camada de dados que auxilia o projetista a mapear as entidades, atributos e relacionamento de nível conceitual para estruturas físicas do banco de dados. É produzido um conjunto de templates de páginas e unidades descritoras que possibilitam a execução da aplicação. O template é um arquivo de código, por exemplo JSP, que expressa o conteúdo da página na linguagem de marcação escolhida. A unidade descritora é um arquivo XML que expressa a dependência da unidade da WebML sobre a perspectiva dos

dados. A Figura 4 apresenta a tela da ferramenta WebRatio na fase de composição da WebApp, onde são definidas as páginas, suas unidades e os relacionamentos.

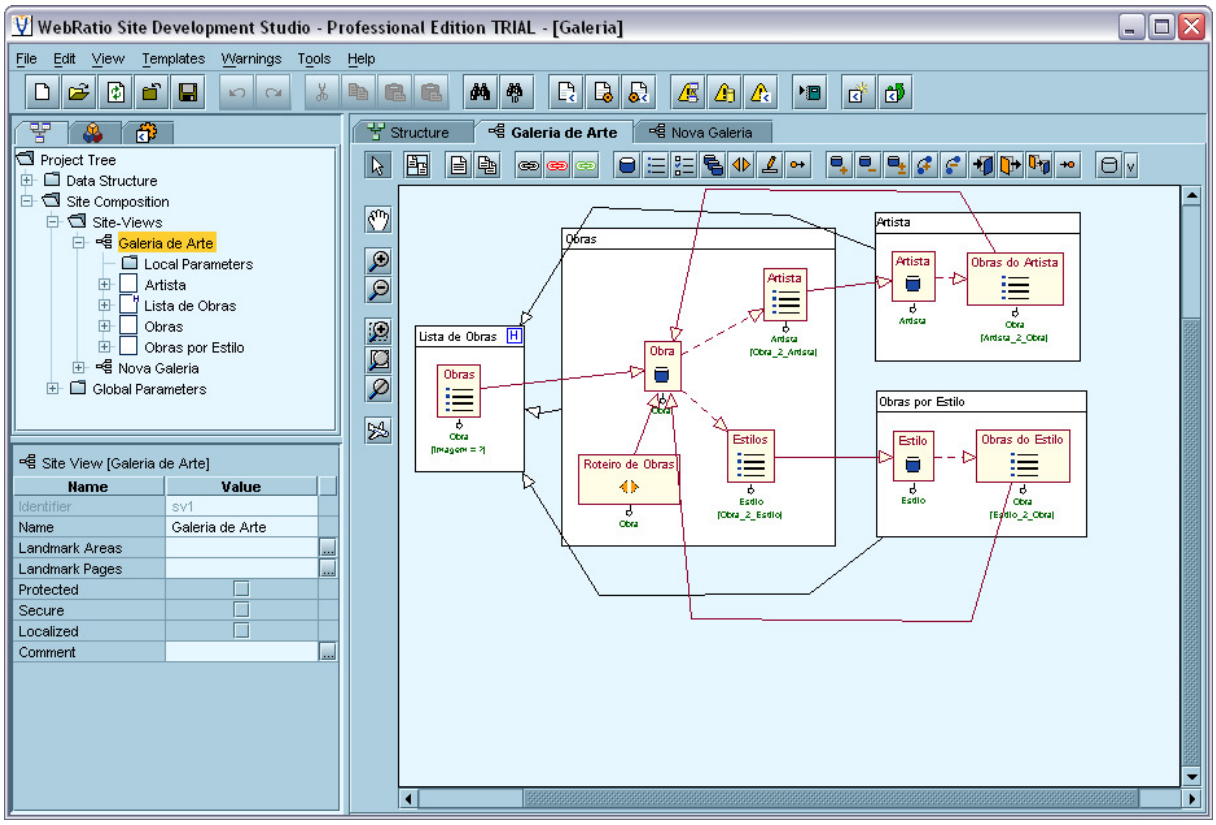


Figura 4 – Exemplo de tela da ferramenta WebRatio.

3.3.2– RMCASE

A ferramenta RMCase (*Relationship Management CASE Tool*), uma das primeiras desenvolvidas, dá suporte a geração de protótipos de aplicações desenvolvidas segundo a abordagem RMM. Implementa todas as fases do método RMM, auxiliando na geração de protótipos. A aplicação especificada pode somente ser apresentada na própria ferramenta.

Várias aplicações utilizando essa ferramenta foram projetadas e simuladas para mostrar a potencialidade do método: *LINKBase* apresenta informações bibliográficas da *ACM SIGLINK* extraídas de uma base de dados relacional, *JMIS WebSite* apresenta as informações da revista *Journal of Management Information Systems* (ISAKOWITZ *et al.* 1997), e o catálogo Web do Departamento de Sistemas de Informação da Escola de Administração *Leonard Stern* (*ISWeb*) (ISAKOWITZ *et al.* 1995). A tela de projeto navegacional da ferramenta RMCase é apresentada na Figura 5.

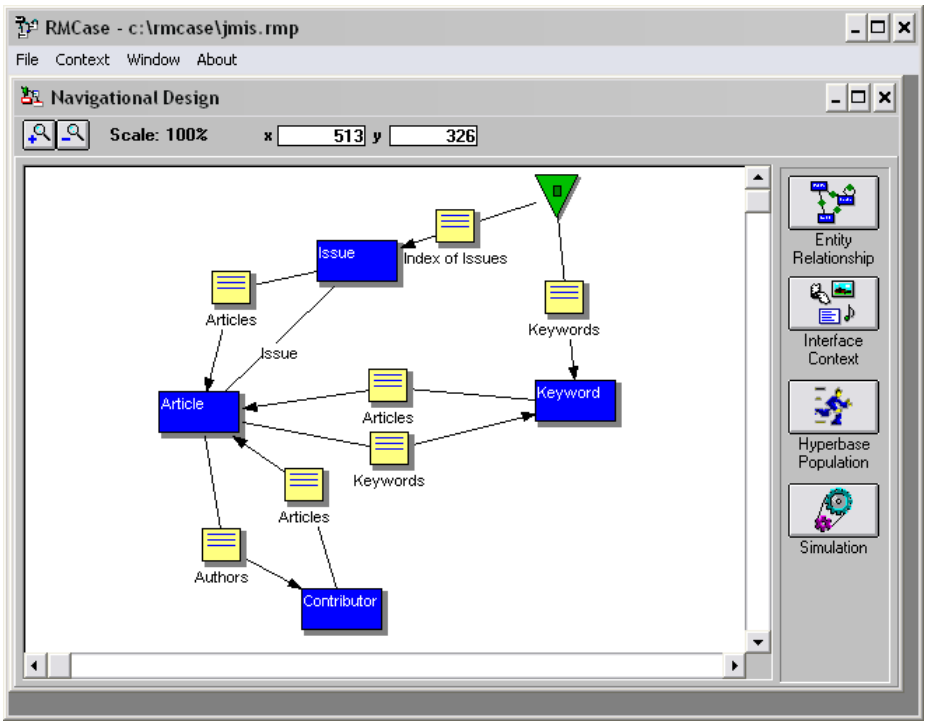


Figura 5 – Exemplo de tela da ferramenta RMCase.

3.3.3– OOHDM-WEB

O ambiente OOHDM-Web (SCHWABE *et al.* 1999; PONTES, 1997) permite a prototipagem rápida de aplicações hipermídia projetadas com o OOHDM. Este ambiente fornece um mapeamento direto de construções de navegação e de interface para uma biblioteca de funções no ambiente de programação CGI (*Common Gateway Interface*) estendido para acesso a bancos de dados. Permite a implementação de aplicações na forma de scripts CGI que produzem páginas geradas dinamicamente.

O OOHDM-Web utiliza o ambiente de script CGI Lua, escrito na linguagem LUA (IERUSALIMSHY *et al.* 1996) e que mantém um número global de variáveis e funções que podem ser acessadas para facilitar a geração das páginas.

A arquitetura geral do OOHDM-Web é ilustrada na Figura 6. O projetista define o esquema de navegação e os templates da interface por meio da primeira interface (Ambiente de Autoria), gerando as definições do banco de dados. Na segunda interface (Ambiente de Navegação), o projetista inclui as instâncias do banco de dados. Na terceira (servidor HTTP), é apresentado o resultado no navegador.

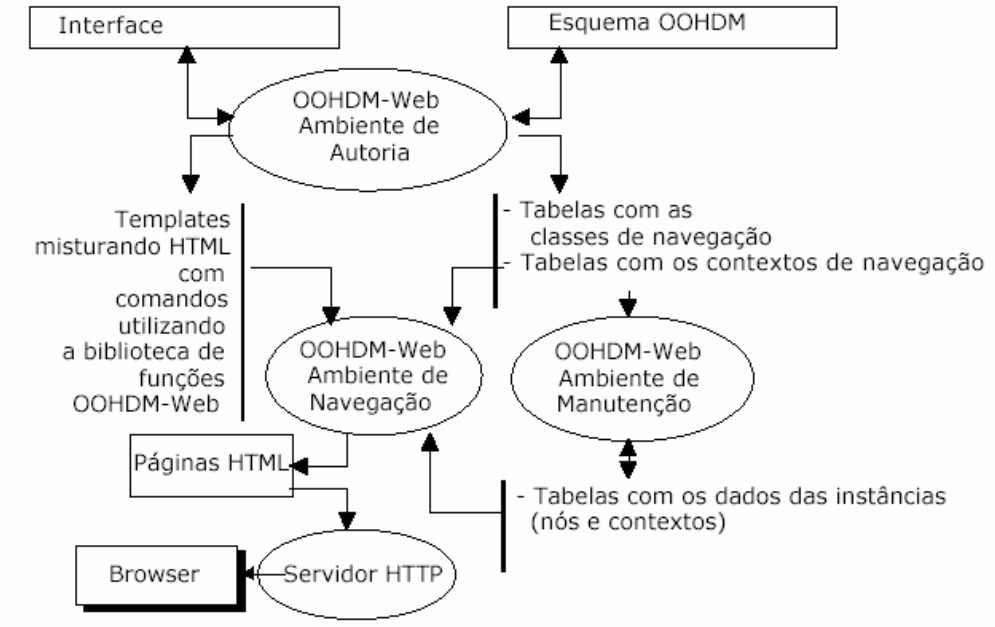


Figura 6 – Arquitetura da ferramenta OOHDM-Web (SCHWABE *et al.* 1999, p. 7).

3.3.4– ARGOUWE

A ferramenta CASE ArgoUWE (KNAPP, 2003) suporta somente a fase de modelagem estrutural do método UWE. É implementada como um *plug-in* da ferramenta de modelagem UML *opensource ArgoUML* (ARGOUML, 2009). Exemplo de tela de especificação do diagrama de classes é apresentada na Figura 7. Novos tipos de diagramas foram adicionados ao *ArgoUML* para incluir a representação dos modelos definidos pelo método UWE: conceitual, navegacional e de apresentação.

A ferramenta suporta a geração semi-automática dos modelos. De posse do modelo conceitual, é gerado o modelo navegacional da *WebApp*. Após completar o modelo navegacional o ArgoUWE também possibilita a geração do modelo de apresentação.

A possibilidade de verificação de consistência dos modelos pela ferramenta auxilia o projetista a garantir que as regras definidas no UWE serão aplicadas. Algumas regras são aplicadas automaticamente, outras necessitam ser ativadas pelo projetista.

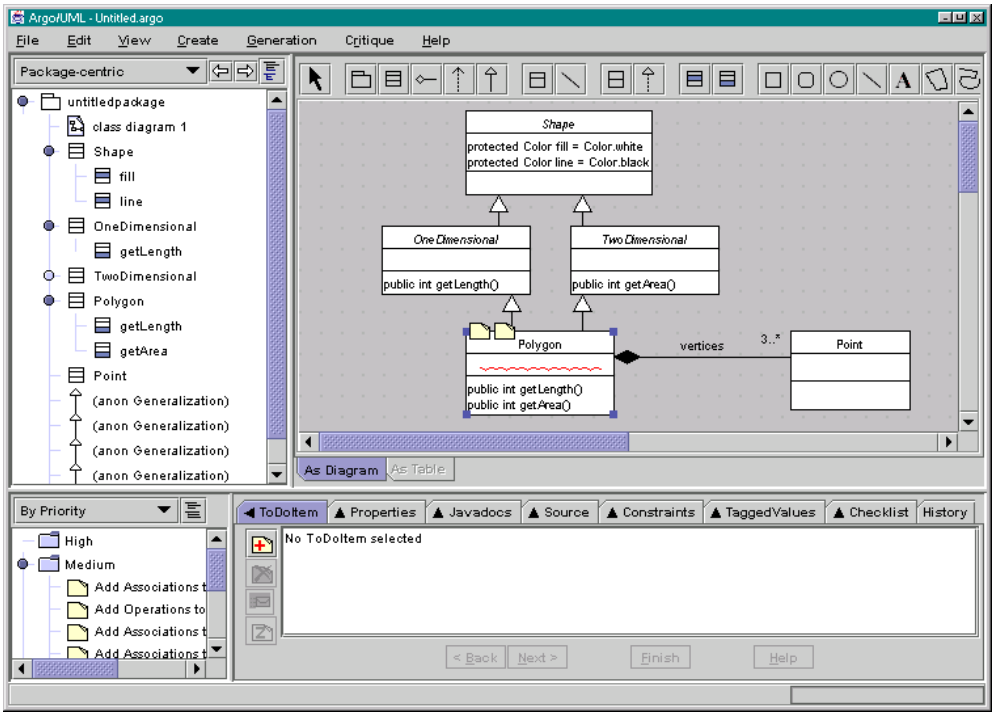


Figura 7 – Exemplo de tela da ferramenta *ArgoUML* com o *plug-in* ArgoUWE.

3.3.5– WEBSCHARTS

A arquitetura da ferramenta *WebSCharts* é composta por três camadas principais, denominadas camadas de aplicação, de estrutura e de armazenamento. A camada de aplicação permite ao projetista criar a estrutura organizacional e navegacional da WebApp e, ao usuário, navegar pela aplicação. Esta camada está subdividida nos ambientes de autoria, de projeto navegacional e de publicação. A camada de estrutura é considerada o núcleo da ferramenta, pois compreende a estrutura interna subjacente à formalização da WebApp com base nos modelos de classes, de fatias, de contextos navegacionais, navegacional de tipos, de composição, de estilos e HMBS. Toda a representação desses modelos é armazenada em bases de dados gerenciadas por funções da camada de armazenamento, que contém as bases internas da ferramenta capaz de gerar o conteúdo XML da WebApp (BRITO, 2003). A arquitetura da ferramenta *WebSCharts* é apresentada na Figura 8.

No ambiente de autoria, o projetista baseia-se no documento de especificação de requisitos da aplicação e especifica os modelos da fase de modelagem conceitual do HMBS/M utilizando os editores de classe e de fatias. As primitivas do modelo de classes são armazenadas na base de dados Classe e as primitivas do modelo de fatias na base Fatias.

No ambiente de projeto navegacional, os editores de modelo de contextos navegacionais, de modelo navegacional de tipos e de modelo de composição são utilizados para especificar os modelos da fase de modelagem navegacional. A partir da definição dos contextos, o modelo navegacional de tipos é construído. Este modelo é a primeira visão abstrata da estrutura navegacional da WebApp. A especificação das páginas Web e o relacionamento entre estas são definidos no modelo de composição. Desta forma a WebApp pode ser gerada automaticamente no ambiente de publicação. Na Figura 9 é apresentada a estrutura dos editores do WebScharts (BRITO, 2003).

O cadastro/atualização do conteúdo da WebApp é feito diretamente no *WebSCharts* ou utilizando uma ferramenta de povoamento desenvolvida especialmente para este fim. A aplicação é gerada em documentos XML que são apresentados em navegadores sem a realização de nenhum tipo de conversão ou mapeamento.

Finalmente, para gerar a aplicação e verificar as consistências é utilizado o gerador de aplicação, que por meio das estruturas criadas anteriormente e da associação de templates XSLT aos documentos XML gera a WebApp. Para especificar os aspectos comportamentais utiliza-se o modelo HMBS. Assim, uma vez povoada a aplicação, o projetista pode automaticamente gerar uma versão comportamental da WebApp por meio do gerador de especificação comportamental.

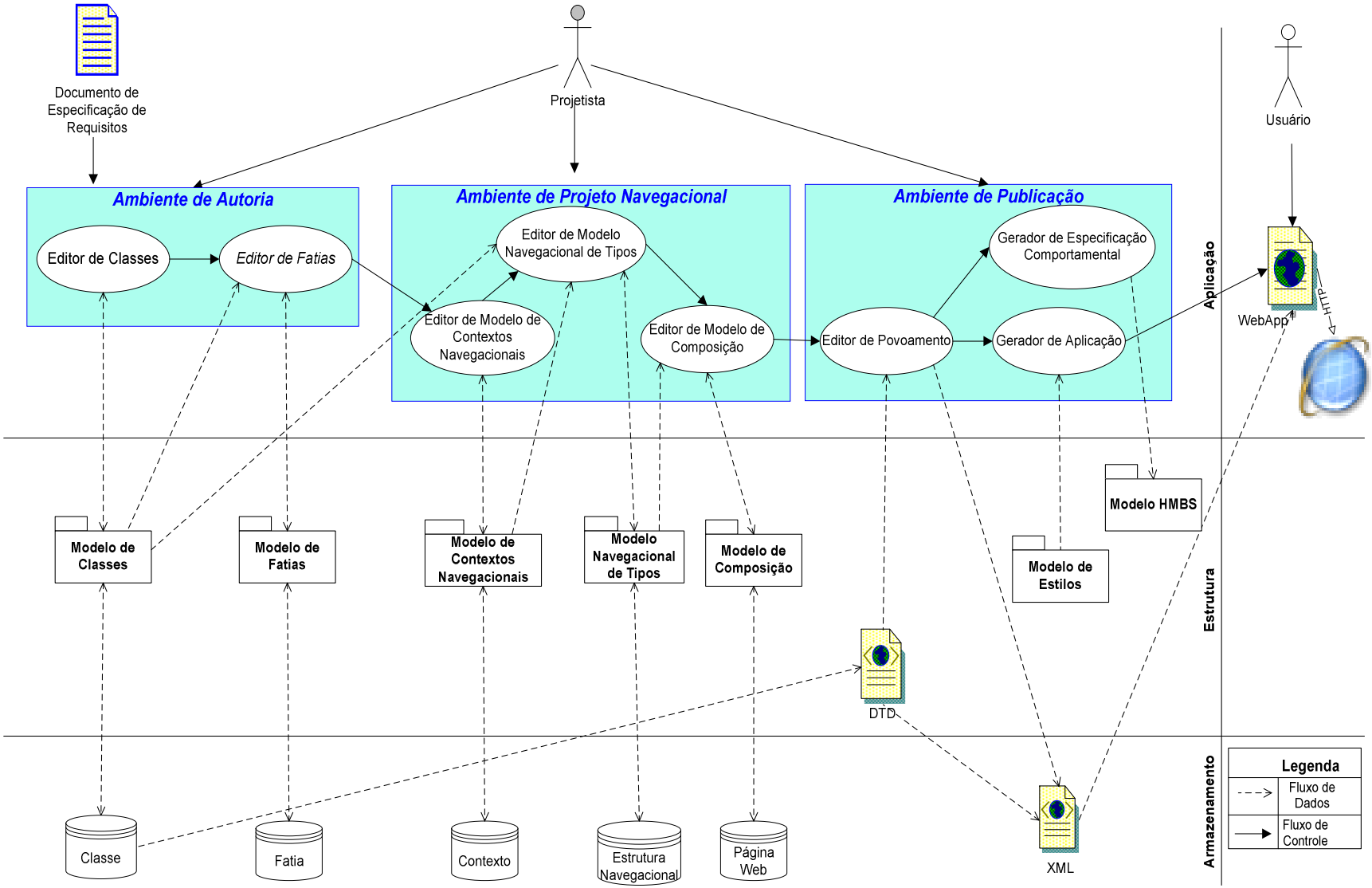


Figura 8 – Arquitetura do WebScharts (BRITO, 2003), p.52

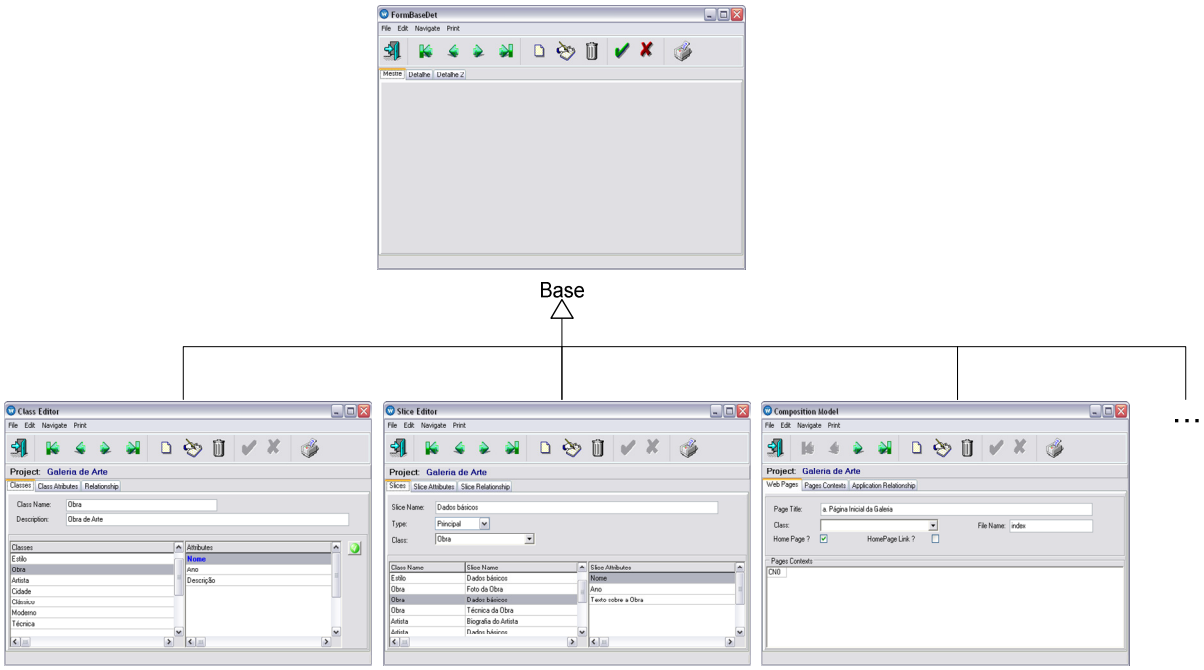


Figura 9 – Estrutura dos editores do *WebSCharts.*

3.3.6– VISUALWADE

A ferramenta CASE *VisualWade* (GOMEZ *et al.* 2005) tem como objetivo desenvolver WebApps com base nos conceitos de sistemas de informação, sendo capaz de gerar código-fonte na linguagem PHP a partir de modelos de projeto. É possível projetar e desenvolver WebApp, com suporte a conexão com bancos de dados relacionais, tais como *MySQL*, *PostgreSQL* ou *ORACLE*. O *VisualWade* possui *plug-in* para a plataforma *Rational Rose*.

Com a *VisualWADE,* uma WebApp é modelada em três grandes perspectivas: estrutural, que descreve a organização da informação, em termos de gestão de conteúdo do domínio e suas relações semânticas; navegacional, que se preocupa com as formas de acesso à informação e a navegação em toda a aplicação; e de apresentação, que determina a maneira pela qual a aplicação e os comandos de navegação são apresentados ao usuário (MELIÁ; GÓMEZ, 2005). A Figura 10 apresenta a tela do *VisualWade*.

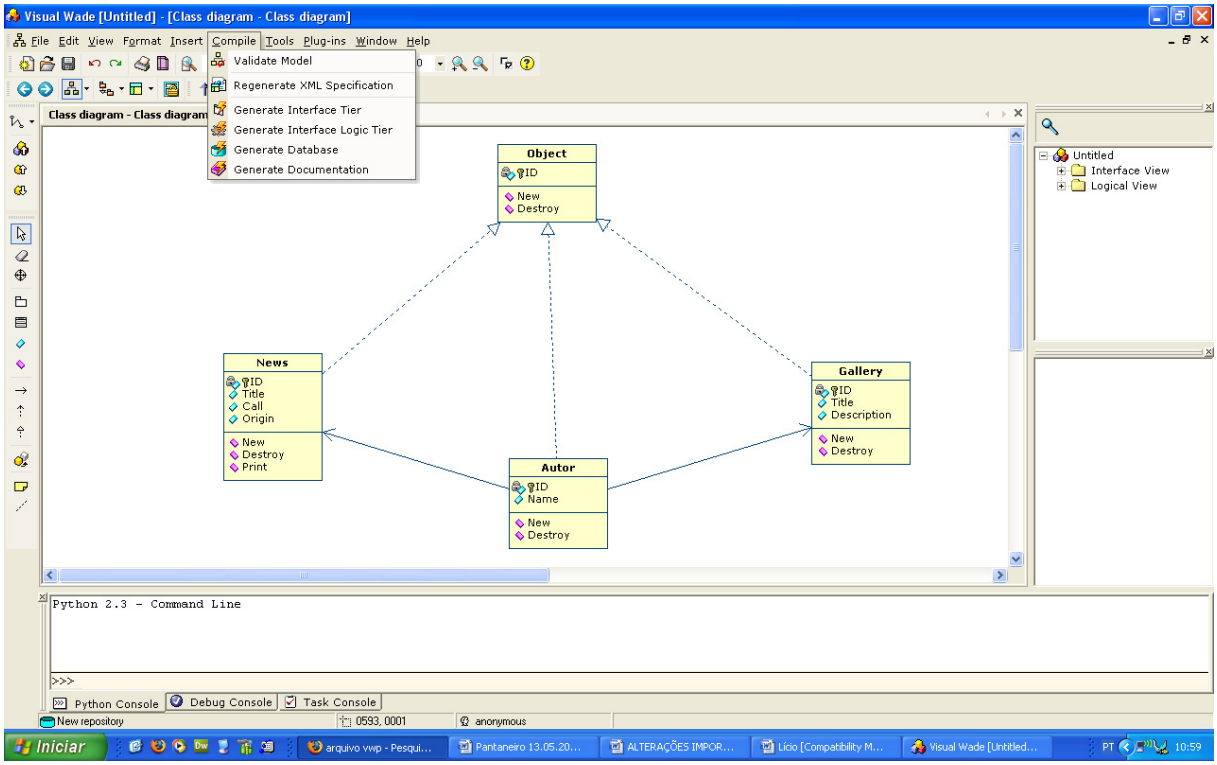


Figura 10 – Exemplo de tela da ferramenta *VisualWade*.

3.4– CONSIDERAÇÕES FINAIS

A utilização de métodos, modelos e ferramentas da Engenharia Web propicia melhorias significativas na qualidade e produtividade no processo de desenvolvimento de *WebApp*. A partir da revisão bibliográfica realizada, poucas ferramentas automatizadas para WebApps auxiliam o projetista no desenvolvimento de portais e-gov. Foi enfatizada a pesquisa nas ferramentas OODHM-Web, RMCase, ArgoUWE, *VisualWade,* WebRatio, HysCharts e WebScharts pois elas se baseiam nos principais métodos citados na literatura OOHDM, RMM, UWE, WebML, OO-H Method, HMBS/M e HMBS/M Estendido, respectivamente. Dentre as ferramentas descritas, apenas WebRatio e *VisualWade* tem publicações atuais que mostram continuidade nos trabalhos. As ferramentas *WebRatio, WebSCharts* e *VisualWade* estão disponíveis na *Web* e estão sendo utilizadas como referência neste trabalho.

Frameworks Orientados a Objetos

4.1 - CONSIDERAÇÕES INICIAIS

O desenvolvimento de frameworks orientados a objetos[26] tem sido motivo de pesquisa desde a década de 80, com o surgimento do paradigma de orientação a objetos. Para Silva (2000), a abordagem desses frameworks utiliza o paradigma de orientação a objetos para produzir uma descrição do domínio a ser reutilizado. A reusabilidade por meio de um framework se dá através da herança e ou composição e, como todas as formas de reuso de software, possui limitações.

Neste capítulo é apresentado o conceito de framework, suas principais características, bem como suas classificações. São descritas as metodologias mais utilizadas no desenvolvimento de um framework e as formas de utilização. Este capítulo apresenta as bases para a discussão da proposta do framework Pantaneiro no domínio e-gov.

4.2 - CONCEITUAÇÃO

É de consenso na literatura a definição que um framework é um conjunto de classes abstratas e concretas usadas para o desenvolvimento de uma aplicação em um domínio específico. Para Mattsson (1996), framework é uma arquitetura desenvolvida com o objetivo de se obter a máxima reutilização, representada como um conjunto de classes abstratas e concretas, com grande potencial de especialização. Para Johnson (1997), pode ser definido como sendo o esqueleto de uma aplicação, que pode ser instanciado por um desenvolvedor de aplicações.

[26] Neste trabalho, o termo “framework orientado a objetos” será substituído por “framework” para efeitos de simplificação.

Johnson e Foote (1988) afirmaram que um framework trata-se de uma aplicação semicompleta reutilizável que, quando especializada, produz aplicações personalizadas. É um conjunto de objetos que colaboram com o objetivo de cumprir um conjunto de responsabilidades para uma aplicação ou um domínio de um subsistema. Silva (2000) define framework como uma estrutura de classes que se relacionam dando origem a uma aplicação inacabada que podem gerar um conjunto de aplicações de um determinado domínio.

De acordo com Braga (2002), a base do framework é fornecida pelas classes abstratas, a partir das quais classes concretas ou outras classes abstratas podem ser implementadas. Para um melhor entendimento, a seguir definem-se ambas as classes:

- **Classes Abstratas:** são as classes que não possuem objetos instanciados a partir dela. Segundo Freiberg (2002), classe abstrata é um projeto que especifica uma classe e a árvore de subclasses que poderá ser produzida a partir dela.
- **Classes Concretas:** são classes que permitem a instanciação de seus objetos. Freiberg (2002) afirma que as classes concretas definem uma estrutura de dados e métodos construídos para satisfazer uma necessidade específica.

Framework é considerado uma forma de reuso de software, pois pode possibilitar o reuso do projeto, do código e da análise, quando estiver disponível. Johnson e Foote (1988) afirmaram que um framework orientado a objetos representa uma categoria de artefatos de software potencialmente capazes de promover reuso de alta granularidade. Os autores salientam que a característica mais importante dos frameworks é a inversão de controle, pois usualmente um desenvolvedor utiliza um programa principal para chamar as bibliotecas que deseja reutilizar.

De acordo com Johnson (1997), o desenvolvedor decide quando chamar os componentes, sendo responsável pela estrutura global e fluxo de controle do programa. No caso do framework, o programa principal é reusado e o desenvolvedor pode construir novos componentes e ligá-los ao programa principal, sendo o framework quem determina a estrutura global e o fluxo de controle do programa.

Em Taligent (1994) são apresentadas as principais diferenças de reutilização de bibliotecas e frameworks, ilustradas na Figura 11.

Biblioteca	Framework
Conjunto de classes instanciadas pelo usuário	Cuida da personalização por meio de subclasses
Usuário chama funções	Chama funções do cliente
Fluxo de controle não é pré-definido	Controla o fluxo de execução
Interação não pré-definida	Define a interação dos objetos
Não tem comportamento "*default*"	Tem comportamento "*default*"

Figura 11 - Principais diferenças entre Biblioteca e Framework (Adaptado de TALIGENT, 1994).

4.3 - CLASSIFICAÇÕES DE FRAMEWORKS

Os frameworks podem ser classificados de acordo com o seu escopo e sua extensão. Fayad *et al.* (1997) propuseram uma classificação quanto ao escopo, estabelecendo as seguintes categorias:

- Framework de infra-estrutura (*System Infrastructure framework*): utilizado na construção de sistemas operacionais, sistemas de comunicação, interface com o usuário e ferramentas de processamento de imagem. Este tipo de framework não é voltado para usuários finais.
- Framework de integração de *middleware* (*Middleware Integration framework*): utilizado para integrar aplicações e componentes distribuídos, facilitando a tarefa do desenvolvedor, pois aumenta a possibilidade de modelar, reusar e estender a infra-estrutura do *software*. O exemplo mais conhecido é o framework ORB (*Object Request Broker*) criado pelo OMG[27] (*Object Management Group*).
- Framework de Aplicação Empresarial (*Enterprise Application framework*): voltado para as aplicações comerciais e para a área de negócios. Este tipo de framework requer um investimento financeiro maior para ser desenvolvido. No entanto, garante o retorno do investimento, uma vez que permite o desenvolvimento de aplicações e produtos para usuários finais.

[27] O OMG é uma organização sem fins lucrativos que promove a padronização de tecnologias orientadas a objetos publicando diretrizes e especificações (DEITEL e DEITEL, 2005).

A forma como se efetiva o reuso por meio de um framework gera uma classificação de acordo com sua extensão, que pode ser baseada em uma das três categorias: caixa preta (*black box*), caixa branca (*white box*) e caixa cinza (*gray box*).

Nos frameworks caixa branca o reuso acontece por herança e são orientados à arquitetura. Silva (2000) afirmou que em um framework orientado à arquitetura, a aplicação deve ser gerada a partir da criação de subclasses das classes do framework. Segundo Fayad *et al.* (1999), frameworks caixa branca tendem a produzir sistemas relacionados aos detalhes específicos das hierarquias de herança do framework. Segundo Roberts e Johnson (1996), o framework pode evoluir e se tornar um framework caixa preta, e isso deve ser feito de forma gradativa, implementando-se várias alternativas que possam ser aproveitadas na instanciação do framework.

Em um framework caixa preta o reuso acontece por composição, ou seja, sua extensibilidade é garantida definindo interfaces para os componentes, por composição de objetos. Fayad *et al.* (1999) afirmaram que frameworks caixa preta são estruturados usando composição de objeto e delegação ao invés de herança. O desenvolvedor pode construir sua aplicação combinando as classes concretas do framework e, para isso, terá que se preocupar apenas com sua interface. Segundo Freiberger (2002), o framework caixa preta é orientado a dados. Silva (2000) afirmou que em um framework orientado a dados, diferentes aplicações são produzidas a partir de diferentes combinações de objetos, instâncias das classes presentes no framework. Esta categoria de framework é mais abstrata e menos flexível, porém a manutenção das aplicações derivadas é complicada e demanda muito tempo. Segundo Fayad *et al.* (1997), os frameworks caixa preta são mais difíceis de desenvolver, pois o desenvolvedor deve prever o maior número possível de aplicações no domínio.

Os frameworks caixa cinza são constituídos por classes abstratas e concretas, ou seja, é uma mistura dos frameworks caixa branca e preta, mantendo a flexibilidade do caixa branca e a facilidade de extensão do caixa preta. Sua forma de reuso é obtida por meio de herança, por ligação dinâmica e por interface de definição. Segundo Fayad *et al.* (1999), um bom framework caixa cinza tem muita flexibilidade e extensibilidade, além da habilidade de ocultar informações desnecessárias ao desenvolvedor da aplicação.

A extensibilidade de um framework é definida pelas partes fixas e variáveis. Pree *et al.* (1995) denominam as partes flexíveis (*hot spots*) de pontos de especialização. As partes fixas (*frozen spots*) são aquelas que não mudam, ou seja, são constantes em todas as instanciações do framework, pois possuem aspectos do domínio da aplicação. Esses pontos são representados pelas classes que podem ser adaptadas em novas aplicações determinando, assim, o grau de extensibilidade do framework. Froehlich *et al.* (1997) nomeiam as partes variáveis de gancho (*hook*), afirmando que são pontos do framework passíveis de mudança e as formas de mudança utilizadas são preenchimento de parâmetros ou criação de subclasses.

4.4 – PROCESSO DE DESENVOLVIMENTO DE UM FRAMEWORK

O desenvolvimento de frameworks tem sido pesquisado nos últimos anos em relação aos requisitos de alterabilidade e extensibilidade. A construção de um framework trará, em longo prazo, uma economia de tempo no desenvolvimento de novas aplicações e uma diminuição significativa de erros e, por conseqüência, um aumento de qualidade do produto final que, nesse caso, é um framework ou um conjunto de aplicações derivadas dele. Segundo Silva (2000), a principal característica buscada ao desenvolver um framework é a generalidade em relação a conceitos e funcionalidades do domínio tratado. Na Figura 12 é ilustrado o método tradicional de desenvolver uma aplicação orientada a objetos e na Figura 13 mostra-se o desenvolvimento por meio do framework, podendo notar também a extensibilidade do framework.

Figura 12 - Desenvolvimento tradicional de uma aplicação orientada a objetos (JOHNSON, 1993).

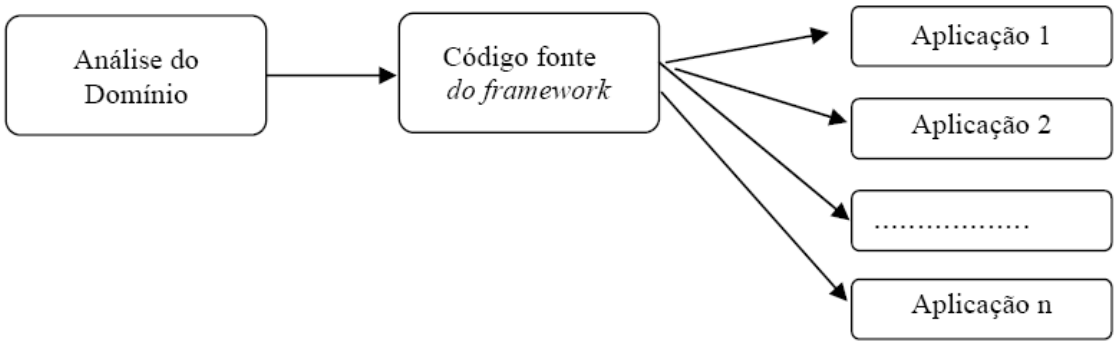


Figura 13 – Desenvolvimento baseado em *frameworks* (MATTSSON, 1996).

A necessidade de utilizar uma metodologia no desenvolvimento de um framework foi apontada por Silva (2000) e Braga (2002). Silva (2000) utilizou em seu trabalho três metodologias: Projeto Dirigido por Exemplos, proposto por Johnson (1993); Projeto Dirigido por *Hot Spot* e de Pree (1994 *apud* BRAGA, 2002). Braga (2002) citou as metodologias de Pree *et al.* (1999), Schmid (1997, 1999), Roberts e Johnson (1996) e Bosch *et al.* (1999). No presente trabalho foram pesquisadas as principais características destas metodologias, que são apresentadas a seguir.

Johnson (1993) propõe o Projeto Dirigido por Exemplos, que sintetiza o conceito de framework como o resultado da análise do domínio, a decomposição do problema e a representação deste problema por meio de um programa, com a possibilidade de reusar a análise, o projeto e o código de uma aplicação já existente, sendo a abstração do domínio o próprio framework.

O processo de desenvolvimento consiste em três fases: análise, projeto e teste:

- **Fase da análise do domínio:** nesta fase os procedimentos envolvidos são o entendimento das abstrações já conhecidas, coleta de exemplos de possíveis programas a serem desenvolvidos a partir do framework e adequação de cada exemplo.
- **Fase do projeto:** a hierarquia das classes é projetada de forma que possa ser especializada para abranger todos os exemplos.
- **Fase de testes:** os testes são feitos usando o framework para desenvolver os exemplos e cada exemplo deve ser uma aplicação separada.

Outra metodologia é chamada “Evolução de Frameworks” e se preocupa tanto com a construção quanto com a instanciação. Nesta abordagem, os autores utilizam

uma Linguagem de Padrão que possui nove padrões. Estes, ao serem aplicados, geram um framework caixa preta com todas as facilidades para sua instanciação. O processo de desenvolvimento dessa metodologia é baseado na aplicação dos padrões. O primeiro padrão sugere que três aplicações concretas sejam construídas e depois generalizadas. A regra geral é: construir a primeira aplicação, construir a segunda aplicação ligeiramente diferente da primeira, e por último construir a terceira aplicação diferente das duas anteriores, porém, todas dentro de um mesmo domínio. Assim, as abstrações comuns ficarão evidentes.

Os padrões seguintes levam, através de iterações, um framework caixa branca a se transformar de forma gradual em um framework caixa preta. Para isso utilizam bibliotecas de componentes, *hot spots*, objetos plugáveis e objetos e granularidade menor. Quando os componentes são acrescentados à biblioteca percebe-se que freqüentemente partes fixas dos códigos são requisitadas e que há a necessidade de procurar os *frozen spots*, pois estas partes mudam de aplicação para aplicação. A criação de objetos plugáveis é uma forma de encapsular os *hot spots* no framework. E, por último, os padrões sugerem o desenvolvimento de um construtor visual e ferramentas de linguagem para facilitar o uso do framework.

Bosch *et al.* (1999) afirmaram que o desenvolvimento de um framework é diferente de uma aplicação padrão, pois o projeto de um framework necessita agregar todas as características pertinentes em um domínio. Baseando-se nos problemas experimentados e identificados durante o desenvolvimento de frameworks, os autores criaram seis atividades para auxiliar no desenvolvimento de um framework simples:

- **Análise de domínio:** esta atividade tem como objetivo descrever o domínio que será coberto pelo framework, capturar seus requisitos e identificar os conceitos. Seu resultado consiste em um modelo de análise de domínio. Muitos desenvolvedores recorrem a aplicações que já foram desenvolvidas no domínio, ou a padrões existentes para este domínio;
- **Projeto arquitetural:** baseado no modelo de análise de domínio cria-se o projeto arquitetural do framework;
- **Projeto do framework:** nessas atividades as classes são refinadas. O resultado desta atividade consiste na extensão da funcionalidade dada pelo projeto do framework;

- **Implementação do framework:** as classes abstratas e concretas são implementadas utilizando a linguagem de programação escolhida;
- **Teste do *framework*:** essa atividade tem como objetivo testar o framework a fim de verificar se possuem as funcionalidades planejadas e avaliar a sua usabilidade. Dependendo do tipo de aplicação, diferentes aspectos do framework podem ser avaliados. Baseando os testes nas aplicações desenvolvidas a partir do framework, pode-se verificar a necessidade de remodelar o framework; e
- **Documentação:** esta atividade é uma das mais importantes no processo de desenvolvimento de um framework, pois se torna possível a sua utilização plena somente por meio da documentação completa, onde se descreve como usar o framework, um manual do usuário e um documento do projeto do framework.

Pree (1999) propõe uma metodologia na qual a construção inicia-se com a definição do modelo de objetos específicos de uma aplicação e as outras atividades são repetidas sucessivamente até que o framework se torne satisfatório. Na Figura 14 são ilustradas as etapas e a forma cíclica de refinamento do projeto.

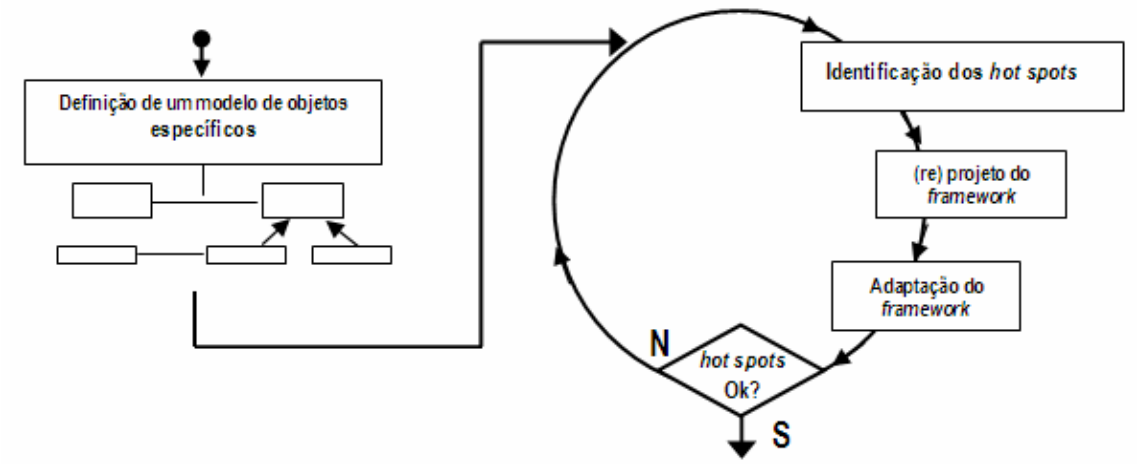


Figura 14 - Processo de desenvolvimento dirigido por *Hot spots* (PREE *et al.*, 1999).

Nessa metodologia há a necessidade de um desenvolvedor e de um especialista no domínio da aplicação e ambos serão co-responsáveis em todo o processo. As etapas são descritas a seguir:

- **Primeira etapa**: o especialista no domínio fornece todas as informações sobre o domínio para o desenvolvedor do framework, para que este defina a estrutura de classes;
- **Segunda etapa**: os especialistas no domínio ajudam os desenvolvedores de framework a identificarem os *hot spots*, que são documentados por meio de cartões de pontos variáveis (*hot spots cards*). Esses cartões têm como principal objetivo facilitar a comunicação entre especialistas de domínio e desenvolvedores, ou seja, padronizar a informação;
- **Terceira etapa**: após os especialistas no domínio identificarem e documentarem os *hot spots*, o desenvolvedor pode ter a necessidade de modificar o modelo de objeto para garantir a flexibilidade que os *hot spots* indicam. Nessa etapa o desenvolvedor utiliza os Padrões de Projetos. Os *hot spots* são projetados e implementados e o framework é testado para verificar se os requisitos do domínio são satisfeitos;
- **Quarta etapa**: consiste em um refinamento da estrutura do framework a partir de novas sugestões dos especialistas no domínio. Alguns pontos variáveis podem se descobertos nesta etapa, levando à repetição da etapa de identificação de *hot spots*. Caso o framework foi avaliado como satisfatório, estará concluída uma versão do framework.

Schmid (1997, 1999) afirma que o mais importante não é iniciar o projeto de um framework tentando modelar sua variabilidade/flexibilidade. Ao invés disso, deve-se projetar uma aplicação fixa dentro do domínio do framework e, após o seu total entendimento, pode-se iniciar a generalização. Schmid (1999) criou quatro atividades para a construção de um framework, como segue abaixo (Figura 15):

- **Análise de alto nível dos *hot spots*:** esta atividade identifica todos *hot spots* do domínio, descrevendo-os brevemente e colecionando-os em forma de cartões e com eles é realizada uma avaliação quanto a variabilidade e flexibilidade.
- **Análise detalhada e especificação dos *hot spots*:** as atividades de análise detalhada e de especificação são feitas separadamente para cada *hot spot*. Analisa-se um *hot spot* e descrevem-se suas características, com base no conhecimento do domínio e nos protótipos de aplicação. Se o resultado da

análise de *hot spot* for um número muito grande de alternativas, deve-se concentrar nos *hot spots* elementares.

- **Projeto de alto nível de subsistemas dos *hot spots*:** com as informações da atividade anterior são gerados vários subsistemas de *hot spots*. Esta atividade deriva um projeto baseando-se na especificação destes *hot spots*. Isto acontece separando este subsistema do resto da estrutura de classe do framework. Nesta atividade são utilizados Padrões de Projetos, pois estes provêm algumas vantagens, tais como: um nível de descrição mais abstrato, menos esforço no desenvolvimento, melhor comunicação de documentação.
- **Transformação do modelo de classes por generalização com subsistemas dos *hot spots*:** essa atividade consiste em transformar a estrutura de classes do framework e generalizá-la, tendo como base uma única característica. Antes da generalização, a estrutura de classes contém normalmente esta característica como um *frozen spot* na forma de uma classe especializada ou de uma responsabilidade especializada como parte de uma classe, a qual possui relações diretas e fixas a outras classes. A transformação de generalização introduz a variabilidade e flexibilidade de um *hot spot* dentro da estrutura de classes.

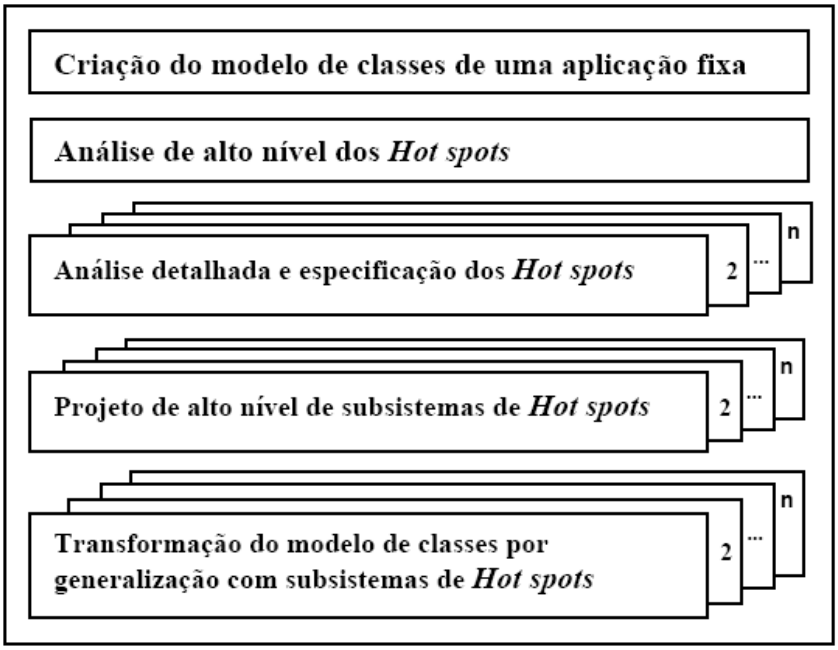


Figura 15 - Atividades para a construção de um *framework* (SCHMID *et al.*, 1999).

4.5 – UTILIZAÇÃO DE UM FRAMEWORK

Utilizar um framework significa desenvolver aplicações a partir dele. Segundo Johnson (1997) e Fayad *et al.* (1999), há várias maneiras de se utilizar um framework, sendo que a maneira mais fácil é conectar os componentes existentes. Isto não muda o framework ou cria qualquer nova subclasse concreta. Reusam-se as interfaces do framework e suas regras para conectar componentes. O desenvolvedor da aplicação só tem que saber que o objeto do tipo "A" é conectado ao objeto de tipo "B", porém não precisa saber a especificação exata de "A" e "B".

O usuário do framework é o desenvolvedor da aplicação a partir do framework e deve conhecer sua estrutura interna. O conhecimento profundo do framework é necessário para que toda a potencialidade de reuso seja aproveitada e esse nível de conhecimento varia de acordo com o tipo de framework.

No caso dos frameworks dos tipos caixa branca e caixa cinza, as classes que precisam ser conhecidas são as classes que devem ser geradas e as classes que podem ser geradas, pois estas herdam algumas de suas características e as tornam adaptadas às exigências da nova aplicação. É necessário diferenciar quais são as classes do framework e quais são as classes geradas a partir dele. Já nos frameworks do tipo caixa preta, além das classes concretas disponíveis é preciso também conhecer suas interfaces, relacionamentos e responsabilidades.

Para todos os tipos de framework, Silva (2000) levantou três questões-chave que devem ser respondidas pelo usuário do framework: a) Quais classes do framework podem ser reutilizadas? b) Quais métodos podem ser reutilizados? c) O que os métodos fazem?

De acordo com Johnson (1993), o usuário deve compreender quais as aplicações que podem ser desenvolvidas a partir do framework e ser capaz de desenvolver aplicações básicas, além de conhecer detalhadamente o projeto do framework. Sobre a utilização de um framework, Silva (2000, p. 75) afirma ainda que:

> *"a motivação para produzir aplicações a partir de frameworks é a perspectiva de aumento de produtividade e qualidade, em função da reutilização promovida. Para produzir uma aplicação sob um framework deve-se estender sua estrutura, de modo a cumprir os requisitos da aplicação.*"

A reusabilidade pode ir além do reuso de linhas de código. Os artefatos de software desenvolvidos com finalidade de reutilização, segundo Freiberger (2002) são: classe concreta, classe abstrata, Padrões de Projetos, frameworks e componentes. Segundo Fayad *et al.* (1997), o reuso de componentes de framework pode render melhorias significativas na produtividade da programação, aumentando qualidade, desempenho, confiança e interoperabilidade do software.

Como o framework é uma das formas de reuso de software, este deve ser bem documentado, para facilitar a sua utilização por outros desenvolvedores. A forma mais utilizada de documentação no caso de frameworks são os chamados *cookbooks*. Para auxiliar o desenvolvimento de aplicações baseadas em framework, Bosch *et al.* (1999) criou um conjunto de atividades que servem como guia, baseado nas atividades de Mattsson (1996) (Tabela 2).

Tabela 2 - Comparativo: As atividades de Mattsson (1996) e os métodos de Bosch *et al.* (1999)

Descrição das atividades de Mattsson	Métodos de Bosch
Atividade 1: definição de um *framework* conceitual para a aplicação e isso se dá através da decomposição funcional e da análise preliminar das características da aplicação. Levantamento das associações entre os componentes e suas funcionalidades, relacionamentos e colaborações.	**Análise dos requisitos:** visa coletar e analisar as exigências da aplicação que deve ser construída usando um ou mais *frameworks*.
Atividade 2: seleção do *framework* para a aplicação. Baseando-se no levantamento da atividade anterior será selecionado um *framework* que atenda aos requisitos da aplicação.	**Arquitetura Conceitual:** Baseado nos requisitos coletados, uma arquitetura conceitual da aplicação é definida. Essa arquitetura possui as associações funcionais entre os componentes, relacionamentos e colaborações.
Atividade 3: mapeamento do *framework* conceitual para o *framework* selecionado.	**Seleção do *framework:*** com base na arquitetura da fase anterior, um ou mais *frameworks* são selecionados, levando em consideração as funcionalidades que devem ser oferecidas à aplicação.
Atividade 4: especificação ou revisão do relacionamento e colaboração entre os subsistemas do *framework*. Essas relações são pré-definidas pelo *framework* podendo ser revisadas ou manter-se sem alterações.	**Seleção de reúso ou extensão:** após decidir qual *framework* será usado, é decidido quais partes do *framework* serão usadas e quais serão definidas pela aplicação.
Atividade 5: estruturação dos sub*frameworks*. Após o mapeamento das funcionalidades do *framework* conceitual, cada componente deve ser associado a um *subframework* ou a um componente ou ainda a um padrão e projeto inserido no *framework*.	**Desenvolvimento das extensões:** decididas às partes que serão desenvolvidas, essas serão especificadas e implementadas. Nessa fase as regras estabelecidas na documentação do *framework* devem ser obedecidas.
Atividade 6: estruturação dos componentes dos *subframeworks*. Nesse caso a atenção é voltada para os *hot spots* e *frozen spots*.	**Teste das extensões:** teste das partes desenvolvidas individualmente.
Atividade 7: implementação da aplicação.	**Desenvolvimento da aplicação:** elaboração do código-fonte de acordo com a tecnologia escolhida.

4.6 – VANTAGENS E DESVANTAGENS

Como apresentado nas seções anteriores, um framework possui características das linguagens de programação orientadas a objetos: herança, polimorfismo e abstração de dados.

Se o framework já estiver pronto, há a maximização do reuso, ou seja, reutiliza-se, em geral, análise, projeto, código e testes e, com isso, acontece uma redução significativa do tempo de desenvolvimento de novas aplicações. As outras vantagens apontadas são:

- diminuição das linhas de código;
- possibilidade de utilizar outras técnicas de reuso em conjunto, por exemplo, Padrões de Projetos (*design patterns*) e componentes aumento da qualidade de software, facilidade na manutenção, pois quando um erro é corrigido no framework, automaticamente, ele é corrigido nas aplicações desenvolvidas a partir deste framework;
- diminuição dos erros no código já que ele é usado em várias aplicações; e
- empacotamento do conhecimento dos especialistas sobre o domínio do problema, evitando a dependência de um único especialista. Os desenvolvedores ficam garantidos caso algum especialista ou pesquisador saia do projeto.

Algumas desvantagens apontadas na literatura são:

- dificuldade em desenvolver um framework;
- se o framework não possuir uma documentação apropriada, certamente ele não será bem utilizado;
- o processo de depuração pode ser complicado porque é difícil distinguir quando o erro é do framework ou da aplicação. Caso o erro esteja no framework pode ser impossível o usuário conseguir corrigi-lo;
- a generalidade e flexibilidade do framework podem atrapalhar sua eficiência em uma aplicação particular; e
- dependência do desenvolvedor do framework, pois têm características que somente o desenvolvedor conhece e a inversão de controle e a falta de fluxo de controle explícito podem dificultar a depuração de alguns erros.

4.7 – FRAMEWORKS EXISTENTES

Esta seção descreve alguns frameworks que podem ser utilizados para a construção de WebApps e-gov, tais como: Drupal (DRUPAL, 2009), Joomla (JOOMLA, 2009), CakePHP (CakePHP, 2009) e Demoiselle (DEMOISELLE, 2009).

4.7.1 – DRUPAL

O Drupal é um framework gerenciador de conteúdo flexível, modular e escalável. É um sistema colaborativo de código aberto, liberado sob a licença GPL[28], foi criado justamente com dois grandes objetivos: usuários e a colaboração entre eles. Esta ferramenta foi desenvolvida na linguagem PHP, é mantida e desenvolvida por uma comunidade ativa, além de um sistema gerenciador de conteúdo onde desenvolvedores web utilizam-na para criar sistemas complexos que vão desde comércio eletrônico até produção de livros colaborativos (RAUNI, 2007).

O Drupal é uma ferramenta desenvolvida com o objetivo de ser um sistema de gerenciamento de portais e um sistema de gerenciamento de comunidades online.

Esta ferramenta divide-se em duas frentes de desenvolvimento, Drupal 5.x e Drupal 6.x. O Drupal 5.x é leve, estável, e destaca-se por possuir praticamente todos os módulos disponíveis. O Drupal 6.x conta com algumas inovações e novos recursos que o tornam mais complexo, porém, alguns dos módulos essenciais para este projeto encontram-se indisponíveis ou estão em fase de desenvolvimento. A compatibilidade entre as versões ainda é um problema que torna o uso do Drupal 6.x bastante arriscado.

4.7.2 – JOOMLA

O nome Joomla tem origem da palavra *Jumla* que significa todos juntos. Criado em 2005, herdou as características do Mambo (MAMBO, 2009), um sistema gerenciador de conteúdo que se acabou por conflitos de ideologias, mas que deu base para a criação dos componentes do Joomla. Sua comunidade de desenvolvimento vem crescendo rapidamente, assim como também vem crescendo a quantidade de

[28] A GPL (GNU Public Licence) é a licença com maior utilização por parte de projetos de software livre, em grande parte devido à sua adoção para o Linux.

componentes disponíveis. Sua base de desenvolvimento está em PHP e utiliza-se de banco de dados MySQL.

A proposta do Joomla é um framework gerenciador de conteúdo de fácil instalação, customização e manutenção, mas exige do usuário certo conhecimento técnico. A proposta é que o conhecimento necessário para a manutenção de uma WebApp seja adquirido de forma fácil a qualquer pessoa que se dedique ao assunto. A indicação feita é que os módulos principais do Joomla sejam instalados restando ao usuário adicionar os módulos conforme surgirem necessidades. Quanto à personalização existem muitos temas gratuitos, mas para criação de seu próprio design o usuário necessitará de um webdesign.

A perspectiva principal do Joomla consiste na capacidade de publicação de conteúdo do próprio usuário (administrador e colaboradores) por meio da edição e publicação do conteúdo sem conhecimento de programação. O Joomla é caracterizado pelo código livre, mas também pelas suas funcionalidades como gerenciamento de banners, arquivamento do conteúdo não utilizado, sistema de buscas, grupos de usuário e sistema de enquetes. Também possui recursos para gerar estatísticas de visitas e avaliações. Quanto às funcionalidades do Joomla se destaca a publicação de notícias e artigos, criação de catálogos de produtos, criação fóruns com álbum de fotos e blogs. Também é possível a facilidade de navegação através de menus e submenus gerados pelo sistema (CICHINI, 2008).

4.7.3 – CAKEPHP

CakePHP é um framework que usa padrões de desenvolvimento conhecidos, tais como, *ActiveRecord*, *MVC* (*Model-View-Controller*), entre outros. Foi baseado nas idéias do framework *Ruby on Rails* (RUBY, 2009) e sua principal meta é ser um framework estruturado que permita a usuários PHP de todos os níveis desenvolverem WebApps robustas sem perda da flexibilidade (MINETTO, 2007).

Algumas características relevantes são:

- **Baseado na arquitetura MVC**: a utilização do MVC facilita o desenvolvimento, e por consequência, proporciona melhor organização dos códigos-fonte, facilita o trabalho em equipe, dentre outras vantagens;

- **Licença Flexível**: utiliza a licença MIT[29], que é mais flexível que outras licenças como a GPL;
- **Compatibilidade com PHP4 e PHP5**: Isso é importante para desenvolvedores que já têm uma base de aplicativos executados em servidor com PHP4 e não tem como migrar o servidor para a nova versão da linguagem , ou que precisem hospedar suas aplicações em um servidor com a versão 5;
- **Validação de Campos**: possui formas de validar os dados digitados pelos usuários em formulários ou URLs. Isso ajuda o desenvolvedor a evitar dados incorretos ou problemas de segurança;
- **Scaffolding**: analisa uma tabela na base de dados e cria automaticamente botões e formulários para realizar as operações básicas de inclusão, alteração e exclusão (CRUD – *Create, Retrieve, Update and Delete*);
- **Lista de Controle de Acesso**: essa característica permite que sejam gerenciadas permissões de acesso nas aplicações, gerenciando dois conceitos: algo (geralmente, usuários) que deseja utilizar alguma coisa e algo que é desejado (aplicações, por exemplo);
- **Componentes**: possui vários componentes que podem ser utilizados pelos desenvolvedores. Dentre os existentes, pode-se citar componentes para segurança, sessões, tratamento de requisições, entre outros. Além disso, pode-se desenvolver componentes que poderão ser utilizados em diversos projetos; e
- **Geração de código-fonte**: possibilidade de geração de código-fonte para evitar esforço repetitivo. Possui uma ferramenta chamada *bake* que permite a criação dos diretórios da aplicação, além da geração de código CRUD.

4.7.4 – DEMOISELLE

O framework Demoiselle tem esse nome em homenagem ao primeiro avião fabricado em série no mundo, construído por Santos-Dumont nos moldes de "código-livre". Santos-Dumont presenteou a todos, sem cobrar nada, todas as suas dezenas de invenções. O framework é uma recente iniciativa do Governo Federal

[29] MIT (*Massachusetts Institute of Technology*) é uma licença que permite a reutilização de software licenciado em programas livres ou proprietários.

Brasileiro, por meio da SERPRO (Serviço Federal de Processamento de Dados). É um framework integrador para JAVA que busca padronizar o desenvolvimento de softwares para o Governo Federal.

O Demoiselle será utilizado como ferramenta base para o desenvolvimento por todas as empresas que fornecem software à administração pública. Lançado como software livre em abril de 2009, deverá ser adotado como o framework oficial do Governo Federal Brasileiro. Não fornece funcionalidades de um gerenciador de conteúdo para portais e-gov, porém oferece uma API bem definida e documentada para a criação de componentes de integração para serviços de governo. Esses componentes são implementados por meio de colaboração de uma comunidade, como ilustrado nas Figuras 16 e 17.

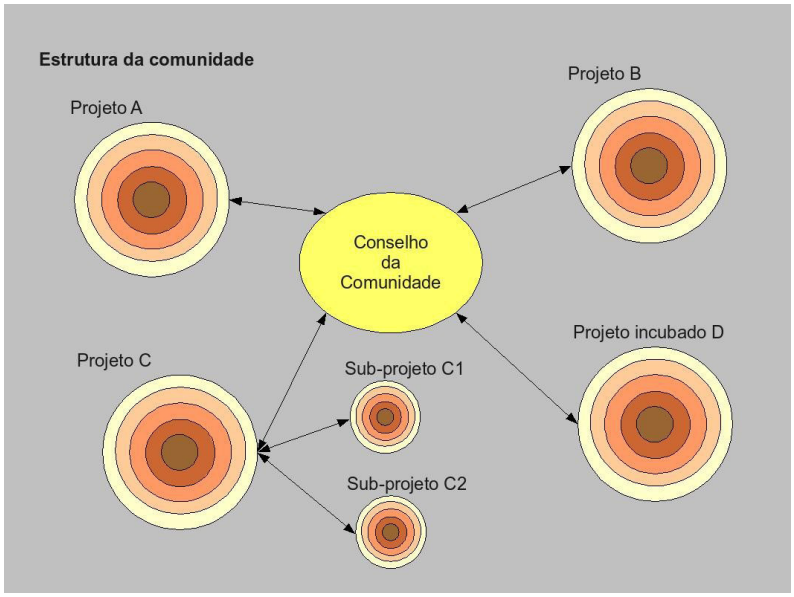


Figura 16 – Estrutura da Comunidade do framework Demoiselle (DEMOISELLE, p.29 - 2009).

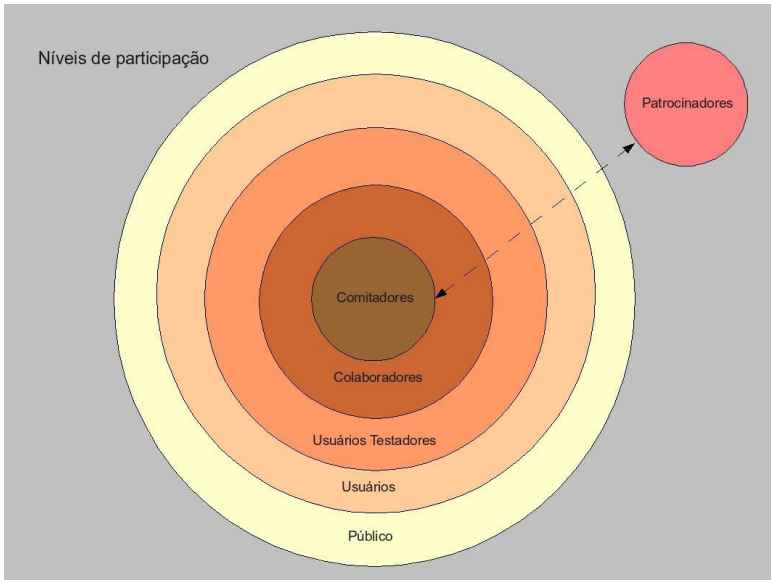


Figura 17 – Níveis de participação dos contribuidores da comunidade (DEMOISELLE, p.30 -2009).

O ciclo de vida das contribuições é descrito a seguir:

1. O contribuidor tem uma idéia e apresenta à comunidade;
2. O contribuidor refina a idéia com o retorno da comunidade;
3. O contribuidor cria um documento com a nova proposta;
4. O contribuidor edita o documento e adiciona todos os detalhes;
5. O contribuidor termina o primeiro esboço;
6. O contribuidor submete a proposta revisada para os comitadores;
7. O comitador move o documento para a seção “Pendente de Recomendação” e avisa o contribuidor;
8. O comitador faz uma recomendação;
9. O comitador aprova a incorporação do código depois que testes e documentação suficientes foram escritos; e
10. O contribuidor prototipa, refina a idéia, e então a submete novamente.

O framework foi construído com as premissas de ser extensível, fácil de usar, estável, configurável, confiável e ter sua documentação publicada. A intenção do Demoiselle é atingir padronização, redução da curva de aprendizagem, maior produtividade, simplificação dos processos, reutilização de códigos e uma manutenção simplificada.

4.8 – CONSIDERAÇÕES FINAIS

Este capítulo apresentou o conceito de framework sob a ótica de vários autores. As formas de classificação de frameworks foram apresentadas, juntamente com as metodologias existentes, descritas com intuito de apresentar as várias possibilidades de implementação. A utilização de um framework é tão importante quanto a sua construção e por isso, foram descritos atividades e métodos que auxiliam e otimizam a sua utilização. Foram apresentadas, ainda, as vantagens e desvantagens na construção e utilização de um *framework*, além da descrição de frameworks existentes. Observa-se que na literatura brasileira, não existem frameworks utilizados comumente em vários órgãos de governo. As iniciativas do governo federal com o framework Demoiselle é extramamente relevante, sendo consistente e coerente com as propostas do framework Pantaneiro.

Pantaneiro: Uma Proposta de Framework e-gov

5.1 – CONSIDERAÇÕES INICIAIS

O principal tema investigado neste trabalho é como utilizar tecnologias de reutilização de software, usabilidade e banco de dados a fim de especificar e implementar o framework Pantaneiro de forma a facilitar a geração de plataformas e-gov e Webapps de qualidade, além de gerenciar e compartilhar o conteúdo dinamicamente, facilitando aos gestores corporativos a manipulação das informações na web, mesmo sem um conhecimento técnico em informática.

O framework Pantaneiro utiliza o método HMBS/M estendido para modelar o comportamento das WebApps, além de técnicas para facilitar o reuso das aplicações desenvolvidas. Uma solução robusta, formal, adaptativa e de fácil utilização e operacionalização para os órgãos públicos.

As seções seguintes apresentam o framework Pantaneiro, seu processo de construção e seu processo de instanciação por meio da ferramenta Wizard-Pantaneiro.

5.2– FRAMEWORK PANTANEIRO

O framework Pantaneiro pode ser definido como sendo o esqueleto de uma WebApp no domínio e-gov ou no contexto de organizações corporativas. As plataformas e-gov podem ser instanciadas por meio do **Wizard-Pantaneiro**, que consiste em uma

ferramenta integrada a um repositório de componentes que funciona como um *wizard* (facilitador) para instanciar as aplicações.

Segundo a forma como se efetiva o reuso no framework, é classificado como um framework caixa cinza (*gray box*), pois proporciona tanto o uso de herança de suas classes abstratas quanto o uso de classes concretas, responsáveis por exemplo, pela implementação do repositório de componentes.

De acordo com a extensibilidade do framework, possui partes fixas (*frozen spots*) e partes variáveis (*hot spots*). Exemplos de partes fixas são aquelas funcionalidades que todas WebApps instanciadas possuem, sendo elas:

- **Menu**: permite navegação rápida entre os componentes;
- **Repositório de Componentes**: permite que a WebApp possa utilizar componentes pré-instanciados e disponíveis de forma livre e configurável;
- **Página Inicial**: Uma WebApp sempre terá uma Página Inicial, que indicará qual é a primeira página de acesso a WebApp;
- **Página Interna**: Uma WebApp sempre terá uma Página Interna, que indicará quais são as páginas visualizadas quando o usuário navega dentro da WebApp;
- **Página Restrita**: permite restringir alguns conteúdos de usuários comuns, necessitando que, para visualização completa do conteúdo de alguns componentes, seja necessário efetuar *logon* na Página Restrita;

Exemplos de partes variáveis são aquelas funcionalidades que diferenciam uma WebApp de outras, sendo elas:

- **Componentes**: em cada WebApp pode-se definir quais componentes do repositório deverá usar, além de definir e instanciar componentes específicos, podendo criar toda a estrutura de um componente novo e específico de sua WebApp. Esse componente específico é chamado de componente “Derivado”, pois sempre será um componente que herdará características de um componente do repositório, este chamado de componente “Base”. Essa estrutura permite que WebApps diferentes consigam compartilhar conteúdos entre si, por meio da especificação das interfaces dos componentes “Base”;
- **Configuração da Página Principal**: cada WebApp terá sua própria configuração de Página Principal, na qual são indicados quais os

componentes serão visualizados, além de definir a posição, a dimensão e o estilo de cada um deles;

- **Configuração da Página Interna**: cada WebApp terá sua própria configuração de Página Interna, na qual serão indicados uma "Área Mutável" da Página Interna e quais os componentes serão visualizados, além de definir a posição, a dimensão e o estilo de cada um deles. Na Página Interna, configura-se quais componentes são sempre exibidos, enquanto que a "Área Mutável" será a área onde as informações internas da WebApp serão visualizadas;
- **Configuração dos Componentes**: cada componente possui configurações para possibilitar sua apresentação correta, sendo elas:
    - Configuração de Fatias: cada "Fatia" consiste de uma possível representação de um dado componente. Essa representação é uma combinação de atributos, cada um configurado com posição, dimensão e estilo de apresentação;
    - Configuração de Contextos Navegacionais: um Contexto Navegacional agrupa informações do domínio da aplicação que estão relacionados segundo algum critério. Vários contextos podem ser especificados para cada "Fatia", definindo, assim, os diferentes pontos de acesso dos componentes as várias formas de se navegar pela aplicação.
- **Configuração de Navegação**: utiliza as Fatias e os Contextos Navegacionais para definir e orientar a navegação;
- **Configuração de Estilos**: define quais características, seguindo a representação CSS (*Cascading Styles Sheets*).

5.3– O PROCESSO DE CONSTRUÇÃO

O framework Pantaneiro foi concebido segundo a abordagem de construção de frameworks proposta por Bosch *et al.* (1999), onde o processo de construção é composto de seis fases: análise do domínio, projeto arquitetural, refinação do projeto arquitetural, implementação, teste e validação e por último, a documentação do projeto e seu guia de usuário. Nas subseções a seguir são apresentadas a análise de domínio e-gov realizada, o projeto arquitetural e as tecnologias utilizadas na implementação do framework, respectivamente.

5.3.1 – ANÁLISE DO DOMÍNIO

A análise de domínio é definida como o processo em que a informação utilizada no desenvolvimento de sistemas de software é identificada, capturada e organizada com o objetivo de fazê-la reusável durante a criação de novos sistemas (PRIETO-DIAZ, 1990; CZARNECKI; EISENERCKER, 2002; WEISS; LAIS, 1999; PRIETO-DIAZ, 1990; VALERIO *et al.*, 1997; GREENFIELD; SHORT, 2004).

Esta atividade concentrou-se em descrever o domínio e-gov que abrange a Estrutura de Aplicação. Para capturar os requisitos e identificar os conceitos, utilizou-se aplicações previamente desenvolvidas, consultou-se especialistas e padrões existentes para o domínio. O resultado desta atividade é um modelo da análise contendo os requisitos, os conceitos, e as relações entre estes conceitos no domínio coberto pela Estrutura de Aplicação (CAMARGO; CARVALHO, 2002).

A análise de domínio deste trabalho foi realizada juntamente com os analistas e gestores da SGI (Superintência de Gestão da Informação do Estado de Mato Grosso do Sul), levantando os requisitos já conhecidos de outras ferramentas e novos requisitos de extrema importância para o Estado de Mato Grosso do Sul, sempre idealizando a concepção de maneira mais genérica possível dentro do contexto e-gov para futuras utilizações do framework em outras unidades federativas.

Dentre os requisitos levantados na análise de domínio, alguns foram identificados como funcionalidades essenciais:

1. O framework deve ser implementado para execução em plataforma Web.
2. O framework deve permitir a instanciação de quantas WebApps forem necessárias.
3. O framework deve permitir que as WebApps compartilhem informações e serviços.
4. O framework deve centralizar as informações de todas as WebApps instanciadas em um único banco de dados.
5. O framework deve visualizar a WebApp construindo-a dinamicamente.
6. As WebApps construídas com o framework devem ser compostas da união de componentes pré-instanciados.
7. O framework deve possuir mecanismos eficientes de busca por informações nas WebApps instanciadas.

8. O framework deve possui um repositório colaborativo de componentes, que armazene componentes próprios do domínio e-gov, tais como: Enquete, Fale Conosco, Fórum, Galeria de Imagens, Banco de Arquivos, Banners, Notícias, Agenda de Eventos, entre outros.
9. Toda WebApp instanciada pelo framework poderá ter acesso aos componentes do repositório.
10. Toda WebApp instanciada poderá possuir seus próprios componentes para atender necessidades específicas, e estes deverão ser totalmente configuráveis.
11. O framework deverá possuir uma ferramenta de instanciação automatizada, para que usuários leigos em linguagens de programação sejam capazes de instanciá-lo. O processo de instanciação deverá seguir uma metodologia de desenvolvimento de WebApps conhecida na literatura.
12. O framework deverá possuir um gerenciador de conteúdo baseado em fluxo de informações (*workflows*), controle de versões, controle por log de alterações, entre outras funcionalidades pertinentes ao domínio e-gov.
13. O framework deverá possuir mecanismos de segurança para garantir integridade e sigilo das informações.
14. O framework deverá possuir mecanismos de controle capazes de gerar relatórios gerenciais relativos ao número de acessos a WebApp, o número de acessos a um dado componente e o número de acessos a uma dada informação.
15. O framework deverá possuir mecanismos de integração entre as WebApps instanciadas e outros serviços e-gov.

De acordo com os requisitos apresentados anteriormente, foi realizada uma análise comparativa entre algumas soluções existentes de frameworks para geração de WebApps. Para esta análise, foi levado em consideração somente frameworks com a funcionalidade de gerenciador de conteúdo. A Tabela 3 traz esta análise comparativa, comparando os frameworks Drupal, Joomla, Microsoft CMS e Pantaneiro (em seu estado finalizado).

Tabela 3 – Análise comparativa entre alguns frameworks com gerenciador de conteúdo.

CARACTERÍSTICAS	DRUPAL	JOOMLA	MICROSOFT CMS	PANTANEIRO
Sistema Operacional	Qualquer	Qualquer	Windows	Qualquer
Linguagem de Programação	PHP	PHP	ASP	PHP
Servidor Web	Apache, IIS	Apache	IIS	Apache, IIS
Classificação quanto ao tipo de extensão	Caixa Cinza	Caixa Cinza	Caixa Cinza	Caixa Cinza
Acesso ao Banco de Dados	MySQL, PostgreSQL	MySQL	SQL Server	Framework Metabase (Independente de SGBD)
Autenticação LDAP	Componente Pago	Componente Pago	Sim	Sim
Documentação	Sim	Sim	Sim	Sim
Gerenciamento de privilégios	Sim	Não	Sim	Sim
Fórum	Nativo	Permite adicionar	Nativo	Nativo
Sistema de Busca	Nativo	Nativo	Nativo	Nativo
Enquete	Nativo	Nativo	Nativo	Nativo
Galeria de Imagens	Nativo	Nativo	Nativo	Nativo
Download/Upload de arquivos	Componente Pago	Componente Pago	Nativo	Nativo
Calendário de Eventos	Componente Pago	Componente Pago	Nativo	Nativo
Ajuda online	Sim	Sim	Sim	Sim
Domínio de abrangência	WebApps Genéricas	WebApps Genéricas	WebApps Genéricas	WebApps e-gov

Controle de Versão nas Instâncias	Sim	Sim	Sim	Sim
Workflow para publicação de instâncias	Não	Sim	Sim	Sim
Software Livre	Sim	Sim	Não	Sim
Estabilidade	Sim	Sim	Sim	Sim
Gerenciamento de Múltiplas WebApps	Nativo	Componente Pago	Nativo	Nativo
Templates	Reduzido	Muitos Templates	Muitos Templates	Gerador dinâmico
Metodologia de desenvolvimento	Não possui	Não possui	Não	HMBS/M estendido
Simulação da aplicação gerada	Não	Não	Não	Sim
Suporte a geração de modelos de desenvolvimento da WebApp	Não	Não	Não	Sim
Compartilhamento de instâncias entre WebApps geradas	Não	Não	-	Sim

Dentre todas as características comparadas, os frameworks Drupal e Joomla são soluções de software livre e atendem parcialmente aos requisitos levantados, porém a generalidade de domínio desses frameworks acaba não oferecendo recursos que são essenciais ao domínio e-gov. Por outro lado, o Microsoft CMS (*Microsoft Content Management Server*) se mostrou como uma poderosa ferramenta, porém proprietária. Anteriormente ao framework Pantaneiro, o Governo do Estado de Mato Grosso do Sul utilizava o Microsoft CMS, porém o alto custo de licenciamento (cerca de R$ 200.000,00 anual – dados de 2003 para a versão Microsoft CMS 2000) inviabilizou o uso desta ferramenta.

Dessa forma, o framework Pantaneiro foi concebido para atender aos requisitos apresentados, ser uma ferramenta baseada no paradigma de software livre e possuir uma metodologia de desenvolvimento para as WebApps instanciadas, visando facilitar o acesso do cidadão à informação e compartilhar informações entre órgãos públicos, evitando a duplicação da mesma informação em todas as páginas do governo.

5.3.2 – PROJETO ARQUITETURAL

A arquitetura do framework Pantaneiro (Figura 18a) é subdividida em cinco camadas: de Aplicação, de Dados, de Acesso ao Banco de Dados, de Segurança e a de Metadados, descritas a seguir:

- **Camada de Dados:** responsável pela persistência das instâncias dos componentes. Cada componente, a partir de sua criação, armazena em sua estrutura de metadados, a referência para o local físico (tabela no SGBD escolhido) onde suas instâncias serão armazenadas. É desse local que as instâncias dos componentes serão recuperadas;
- **Camada de Acesso ao Banco de Dados:** serve de apoio à aplicação, para prover a comunicação entre a Camada de Dados e as demais camadas. A Camada de Acesso ao Banco de Dados é implementada utilizando-se o framework de acesso a banco de dados METABASE (METABASE, 2009), tornando a aplicação independente do SGBD escolhido;
- **Camada de Metadados:** responsável pela persistência da estrutura de controle de componentes, interfaces, dados, apresentação e papéis do sistema. Os metadados podem ser vistos como pontos variáveis do framework Pantaneiro. A camada de Metadados é subdivida em:

  a) Metadados de Componentes: são as informações sobre a estrutura de controle dos componentes, tais como suas características, seus atributos e seu local físico para a persistência das instâncias;

  b) Metadados de Conteúdo: são as informações de controle sobre os dados, tais como as características pertinentes à instância, como por exemplo, se uma instância encontra-se publicada ou não, ou se ela é uma informação restrita ou não;

c) Metadados de Interface e Navegação: são as informações que controlam a correta apresentação dos dados e navegação entre as páginas, de acordo com a interface especificada, a disposição dos componentes na aplicação e a estrutura navegacional configurada.

- **Camada de Segurança:** responsável por fazer o controle de permissões sobre os dados e metadados da aplicação. Ela tanto controla os papéis dos usuários da ferramenta de desenvolvimento, através de um protocolo de permissões adotado, como controla o acesso às aplicações ou parte delas;
- **Camada de Aplicação:** consiste em classes concretas do framework responsáveis pela visualização das páginas da WebApp. Tais classes utilizam-se de códigos em PHP para mesclar informações advindas da Camada de Metadados, da Camada de Dados e da Camada de Segurança, para possibilitar a visualição das páginas da WebApp.

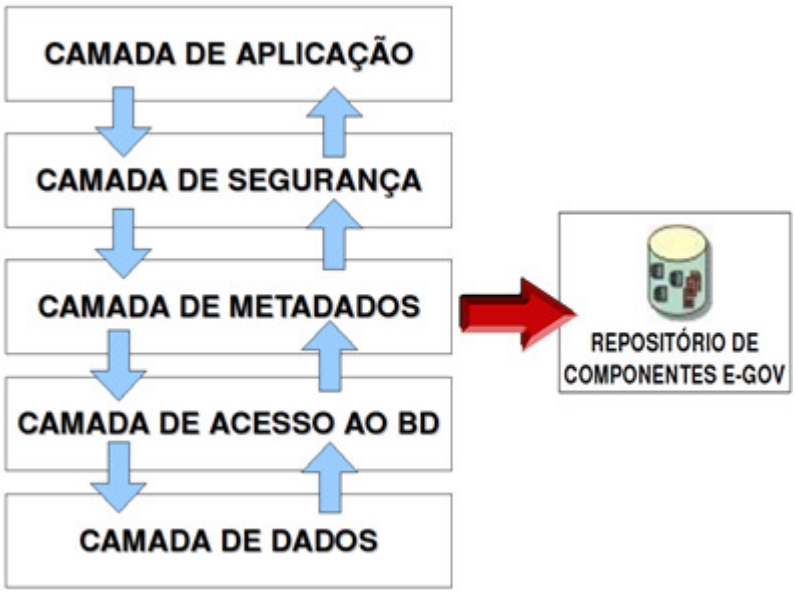


Figura 18a – Arquitetura do framework Pantaneiro.

A Figura 18b ilustra a arquitetura geral da proposta, englobando o framework Pantaneiro, o Wizard-Pantaneiro e o SCXML Viewer. A Figura 19 ilustra um diagrama de classes (com seus pacotes) simplificado do projeto arquitetural do framework Pantaneiro.

Figura 18b – Arquitetura geral da Proposta.

5.3.3 – TECNOLOGIAS ADOTADAS

Para o desenvolvimento dos artefatos de software do framework Pantaneiro, foram adotadas as seguintes tecnologias, tendo como base o paradigma de software livre e as necessidades do Governo do Estado de Mato Grosso do Sul:

- **PHP** (um acrónimo recursivo para "PHP: Hypertext Preprocessor") é uma linguagem de programação de computadores interpretada, livre e muito utilizada para gerar conteúdo dinâmico na Web. Neste trabalho foi utilizada a versão 5 por esta possuir suporte ao paradigma OO (orientação a objetos).
- **JAVASCRIPT** é uma linguagem de programação criada pela Netscape em 1995, que a princípio se chamava LiveScript, para atender, principalmente, as seguintes necessidades: Validação de formulários no lado cliente (programa navegador); Interação com a página.
- **CSS** (*Cascading Style Sheets*) é uma linguagem de estilo utilizada para definir a apresentação de documentos escritos em uma linguagem de marcação, como HTML ou XML. Seu principal benefício é prover a separação entre o formato e o conteúdo de um documento.

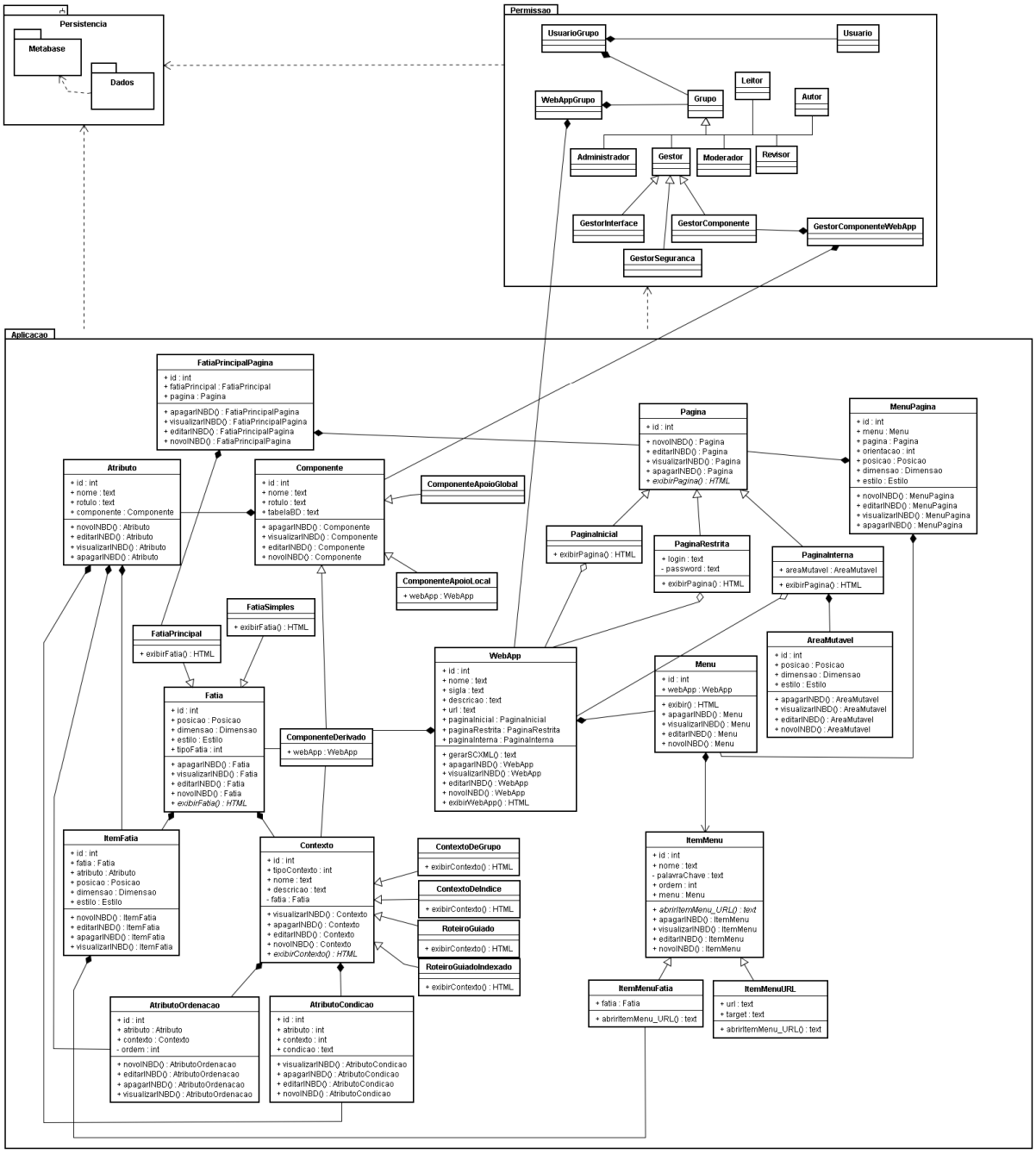


Figura 19– Diagrama de Classes (com seus pacotes) simplificado do projeto arquitetural do framework Pantaneiro.

- **XML** (*eXtensible Markup Language*) é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais. É um subtipo de SGML (acrônimo de *Standard Generalized Markup Language*, ou Linguagem Padronizada de Marcação Genérica) capaz de descrever diversos tipos de dados. Seu propósito principal é a facilidade de compartilhamento de informações por meio da internet. Entre linguagens baseadas em XML incluem-se XHTML (formato para páginas Web),

RDF,SDMX ,SMIL, MathML (formato para expressões matemáticas), NCL, XBRL, XSIL e SVG (formato gráfico vetorial).

- **AJAX** (acrônimo em língua inglesa de *Asynchronous Javascript And XML*) é o uso metodológico de tecnologias como Javascript e XML, providas por navegadores, para tornar páginas mais interativas com o usuário, utilizando-se de solicitações assíncronas de informações. AJAX não é somente um novo modelo, é também uma iniciativa na construção de aplicações web mais dinâmicas e criativas. AJAX não é uma tecnologia, são realmente várias tecnologias conhecidas trabalhando juntas, cada uma fazendo sua parte, oferecendo novas funcionalidades.
- **PostgreSQL** é um dos SGBD-R atualmente homologados para o framework Pantaneiro. Foi escolhido por seguir o paradigma de software livre e se enquadrar como banco de dados robusto e estável.
- **Microsoft SQL Server** é um dos SGBD-R atualmente homologados para o framework Pantaneiro. Foi escolhido por opção da SGI-MS por fornecer uma plataforma de dados confiável, produtiva e inteligente, permitindo a execução de aplicações de missão crítica mais exigentes e sendo utilizado atualmente por sistemas corporativos dos mais diversos portes. A instalação do framework Pantaneiro no Estado de Mato Grosso do Sul utiliza este SGBD-R.
- **Framework Metabase** é um framework em PHP que implementa um conjunto de classes que controlam a organização, o armazenamento, a gerência e a recuperação de dados de um banco de dados relacional, independente do SGBD-R escolhido. O Metabase permite que o framework Pantaneiro possa homologar quaisquer SGBD-Rs homologados pelo Metabase, que hoje são: Interbase, Firebird, Informix, Mini SQL, Microsoft SQL SEVER, MySQL, Microsoft Access, PostgreSQL, Oracle, SQL-Lite, além de quaisquer outros SGBD-Rs que utilizem a interface ODBC (*Open Data Base Connectivity*).
- **JAVA**: é uma linguagem de programação orientada a objetos desenvolvida na década de 90. A linguagem JAVA é distribuida com um vasto conjunto de bibliotecas (API - *Application Programming Interface*). As principais APIs utilizadas neste trabalho foram: *Apache Commons*

BeanUtils, *Apache Commons Collections*, *Apache Commons Digester*, *Apache Commons Jexl*, *Apache Commons Logging*, *Apache Commons Scxml*, *Serializer*, *Xalan*, *XercesImpl* e *Xml-Apis* para o desenvolvimento da aplicação SCXML Viewer – Pantaneiro.

- **Apache Server** é o mais bem sucedido servidor web livre. Foi criado em 1995 por Rob McCool, então funcionário do NCSA (*National Center for Supercomputing Applications*). Numa pesquisa realizada em dezembro de 2007, foi constatado que a utilização do Apache representa 47.20% dos servidores ativos no mundo.

5.4 – O PROCESSO DE INSTANCIAÇÃO DO FRAMEWORK

Um framework é uma técnica poderosa para aumentar o reuso de software, visto que inúmeras aplicações distintas podem ser obtidas por meio de sua instanciação. Entretanto, o processo de instanciação é, na maioria das vezes, muito complexo, exigindo conhecimento considerável a respeito do projeto e implementação do framework (BRAGA, 2002). Existem pelo menos cinco tipos de abordagens para instanciação de frameworks: instanciação pelo uso da documentação do framework, instanciação por exploração de exemplos, instanciação por meio de padrões, instanciação por meio de *cook-books*[30] e instanciação com a utilização de um *wizard*[31].

O processo de instanciação do framework Pantaneiro é realizado pela ferramenta Wizard-Pantaneiro, que usa a abordagem da utilização de um wizard, uma ferramenta de desenvolvimento de aplicações, que permite a instanciação dinâmica e transparente do framework.

5.4.1 – O WIZARD-PANTANEIRO

O Wizard-Pantaneiro é composto por três ambientes: de Autoria, de Projeto Navegacional e de Publicação. Resumidamente, no ambiente de Autoria tem como objetivo realizar a modelagem da aplicação, a definição dos componentes, a modelagem da interface e as configurações necessárias para a gerência de

[30] *Cook-books*: do inglês *livro-de-receita*, uma espécie de documentação passo-a-passo de utilização do framework.
[31] *Wizard*: do inglês *assistente*, é um padrão de projeto de software amplamente utilizado em interface gráfica do usuário para prover um meio simples de realizar tarefas complexas em sistemas computacionais, através de um esquema passo-a-passo.

permissões sobre os componentes e interfaces geradas. No Ambiente de Projeto Navegacional é onde o gestor da WebApp instanciada projeta a navegação existente entre os componentes criados no Ambiente de Autoria, além de manipular dinamicamente o menu da aplicação, configurando aspectos de navegação sintetizada pelo “mapa do site”. No Ambiente de Publicação são realizadas as fases de instanciação e publicação das WebApps , além da fase de povoamento e publicação das instâncias dos componentes, sobre a ideologia de um gerenciador de conteúdo baseado em *workflows*.

5.4.1.1 – AMBIENTE DE AUTORIA

O ambiente de autoria é composto pelos módulos de Gerência de Componentes, de Gerência de Permissões e de Gerência de Interfaces. Na Figura 27 é ilustrado o diagrama de Casos de Uso do Ambiente de Autoria.

GERÊNCIA DE COMPONENTES

O termo componente é utilizado neste trabalho como um conjunto de funcionalidades (características e comportamentos) necessárias para reproduzir um determinado processo, como por exemplo, a organização institucional de uma empresa é modelada como um componente. Outros exemplos são: notícias, enquetes, fóruns e todas as seções que estão presentes em um sítio.

A Gerência de Componentes é um módulo que organiza a estruturação dos componentes, bem como suas características e funcionalidades. Todos os componentes criados neste módulo preservarão as características de integridade, segurança, disponibilidade e durabilidade, além da capacidade de serem reutilizados em ambientes externos à da arquitetura do framework Pantaneiro.

A Gerência de Componentes atua na Camada de Metadados, persistindo as configurações, atributos e relações dos componentes, além de fazer acesso à Camada de Segurança, filtrando o acesso do usuário autenticado apenas aos componentes que ele possui permissão.

Nas Figuras 20 e 21 são ilustrados o ambiente de autoria de um componente no framework Pantaneiro, cujo usuário autenticado tem as permissões de Gestor de Componentes. A Figura 20 ilustra a tela onde se define os metadados básicos de um

componente, tais como “nome do componente”, “descrição”, “rótulo”, “webapp”, “tipo de componente” e “publicação”.

Na Figura 21 é ilustrada a tela para a criação de atributos. Esses atributos são tipificados em: “Número Inteiro”, “Número Real”, “Texto Curto”, “Texto Longo”, “Recurso”, “Relacionado”, “Banner”, dentre outros. O tipo “Recurso” permite inserir atributos que se vinculam a arquivos do “Banco de Arquivos” do framework Pantaneiro. O tipo “Relacionado” permite inserir atributos para “relacionar” o componente a um outro componente. O tipo “Banner” permite inserir atributos para “relacionar” o componente com um Banner, que é um componente “permanente” do framework Pantaneiro, presente no repositório de componentes.

Figura 20 – Tela da “Etapa 1” para a criação de componentes.

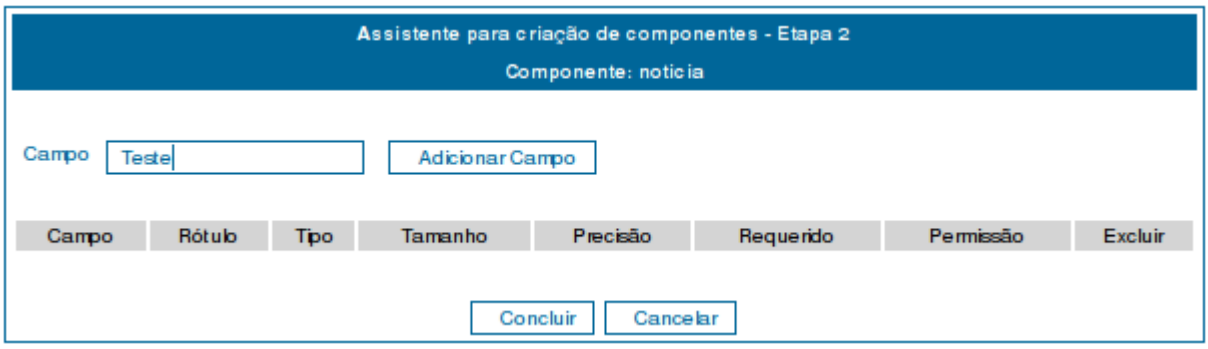


Figura 21 – Tela da “Etapa 2” para a criação de atributos.

Este módulo é responsável pela composição do esquema do banco de dados da WebApp. Abstraindo a idéia de componentes em classes, com seus atributos e inter-relações, a Gerência de Componentes permite a geração dos modelos de classes e modelos de fatias do HMBS/M estendido. Dessa forma, a Gerência de Componentes abrange a modelagem conceitual da aplicação, segundo o HMBS/M estendido.

GERÊNCIA DE PERMISSÕES

A integridade e a consistência serão asseguradas nos níveis de criação, alimentação e visualização das informações contidas nos componentes, bem como na manipulação do esquema de um componente. Em cada um destes níveis, os privilégios de manipulação serão verificados por meio do papel definido para o usuário que está manipulando as instâncias e/ou esquemas do componente.

Devido ao grande compartilhamento de informações, a existência de um protocolo de permissões comprometido em manter a integridade e consistência do conteúdo da ferramenta é fundamental.

Com o objetivo de facilitar a manutenção das permissões na ferramenta, as permissões são atribuídas a grupos de usuários. Cada grupo contém vários usuários e está associado a um papel. Este papel corresponde ao tipo de permissão com a qual o grupo atuará na ferramenta. Dependendo do papel, o grupo possuirá um conjunto de websites e/ou um conjunto de componentes. Caso o tipo de permissão seja de esquema, apenas os websites deverão estar presentes no grupo. Caso o tipo de permissão seja de instância, deverão estar presentes as informações sobre que componentes de quais websites o grupo terá privilégios.

Os papéis existentes podem ser classificados em duas categorias, de acordo com a semântica de seus privilégios. Cada uma das categorias é estruturada hierarquicamente. Na Figura 22 é apresentada a estrutura hierárquica das duas categorias.

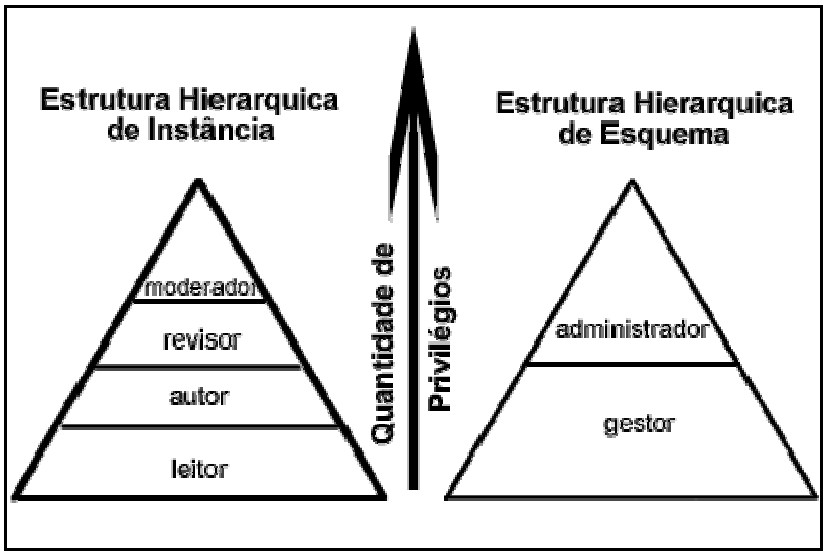


Figura 22 – Estrutura hierárquica das categorias de papéis existentes no Pantaneiro.

Além de seus privilégios específicos, o papel situado no nível superior, ou possui todos os privilégios dos papéis situados nos níveis inferiores, ou é capaz de realizar a auto-atribuição de tais permissões. As categorias e seus papéis são:

- **Esquema**: esta estrutura hierárquica está relacionada aos privilégios existentes durante a manipulação dos esquemas dos componentes. É composta pelos seguintes papéis:
  - **Administrador**: este papel é responsável pela realização de todas as atividades necessárias para manter o funcionamento da ferramenta.
  - **Gestor**: este papel é responsável por realizar todas as atividades necessárias para manter o funcionamento de uma aplicação específica. De acordo com suas responsabilidades é dividido em três categorias distintas, podendo agrupar uma ou mais características dentro deste papel. São elas:
    - **Gestor de Segurança** – responsável por criar e atribuir permissões aos grupos de usuários dentro da aplicação.
    - **Gestor de Componentes** – responsável por instanciar e publicar componentes na aplicação.

- **Gestor de Interface** – define a interface com o qual a aplicação será apresentada, organiza os componentes para visualização da aplicação e escolhe as interfaces da aplicação.

- **Instância**: esta estrutura hierárquica está relacionada aos privilégios existentes no momento da manipulação das instâncias dos componentes. Para os papéis desta estrutura é necessário definir sobre quais componentes de quais aplicações o grupo atua. Ele pode atuar sobre um ou mais componentes da aplicação. Esta estrutura é composta pelos seguintes papéis:
  - **Leitor Restrito**: este papel possui o privilégio de visualizar as instâncias restritas contidas nos componentes das aplicações que o grupo tem permissão. A visualização é feita na interface de apresentação da aplicação.
  - **Autor**: este papel possui o privilégio do papel "Leitor Restrito". Além disso, pode inserir informações nos componentes das aplicações que o grupo tem permissão e alterar e remover apenas as informações inseridas por ele e pelos outros usuários do grupo.
  - **Revisor**: este papel possui os privilégios do papel "Autor". Além disso, pode inserir informações nos componentes das aplicações que o grupo tem permissão, alterar e remover as informações pertencentes aos componentes dos sites sobre os quais tem permissão.
  - **Moderador**: este papel possui os privilégios do papel "Revisor". Além disso, pode publicar, compartilhar e ainda restringir o acesso a algumas instâncias dos componentes pertencentes às aplicações para as quais o grupo tem permissão.

Na Figura 23 é ilustrada a tela de cadastro de usuários do framework Pantaneiro, enquanto que a Figura 24 ilustra a edição de papéis (“grupos”) do framework Pantaneiro, configurando “nome”, “descrição” e “permissão”. Além disso, a tela de edição de papéis, ainda possibilita a vinculação de “usuários” ao “papel” criado e dar permissões de acesso aos componentes da WebApp.

GERÊNCIA DE INTERFACES

Os gerenciadores de interface possibilitam manipular e organizar os elementos da aplicação, por meio de uma área de edição gráfica, que integra o usuário com a aplicação (JACOBS, 2006).

Alguns geradores de código produzem esqueletos para que os desenvolvedores escrevam o código que será integrado à interface. Em alguns casos comuns, as interfaces são padronizadas utilizando-se templates, o qual surge do conceito de gabaritos. E com o uso de templates se estabelece uma pré-definição aos elementos que serão utilizados na interface alvo.

Uma proposta de gerenciador de interface encontrado na literatura é a ferramenta TERESA – *Transformation Enviroment for Interactive Systems Representations* (TERESA, 2006), um ambiente semi-automático para a construção de interfaces baseadas em transformações de modelos. Úteis para o projetista construir, analisar seus projetos em diferentes níveis de abstração e conseqüentemente, gerar a interface concreta para tipos específicos de plataforma.

Outro, o ambiente *TransformiXML* realiza mapeamentos, através de relacionamentos entre expressões matemáticas e permite a definição e aplicação de regras de transformação. Estas regras representam associação entre modelos para gerar interface do usuário. Esse ambiente adota o padrão *UsiXML* (UsiXML, 2009) para os modelos de IHC (Interface Humano-Computador) e para as regras que mapeiam tais modelos.

A Gerência de Interfaces do Ambiente de Autoria é composta por:

- **Modelagem da Interface**: Antes de começar a configurar a interface, o usuário do Pantaneiro deve desenhar um rascunho da interface desejada. Nesse passo são escolhidos quais componentes deverão ser apresentados, associando-os às várias características de interface/estilo e posicionando-os na tela.
- **Geração de Gabaritos**: É responsável por salvar no banco de dados todas as configurações realizadas no passo anterior. Após salvar tais características, é possível a renderização da aplicação final. Uma interface criada no Pantaneiro pode ser salva como gabarito para futuras utilizações, criando um repositório de interfaces. Com tais informações, o módulo de Gerência de Interfaces possibilita a construção do modelo de estilos do HMBS/M estendido.

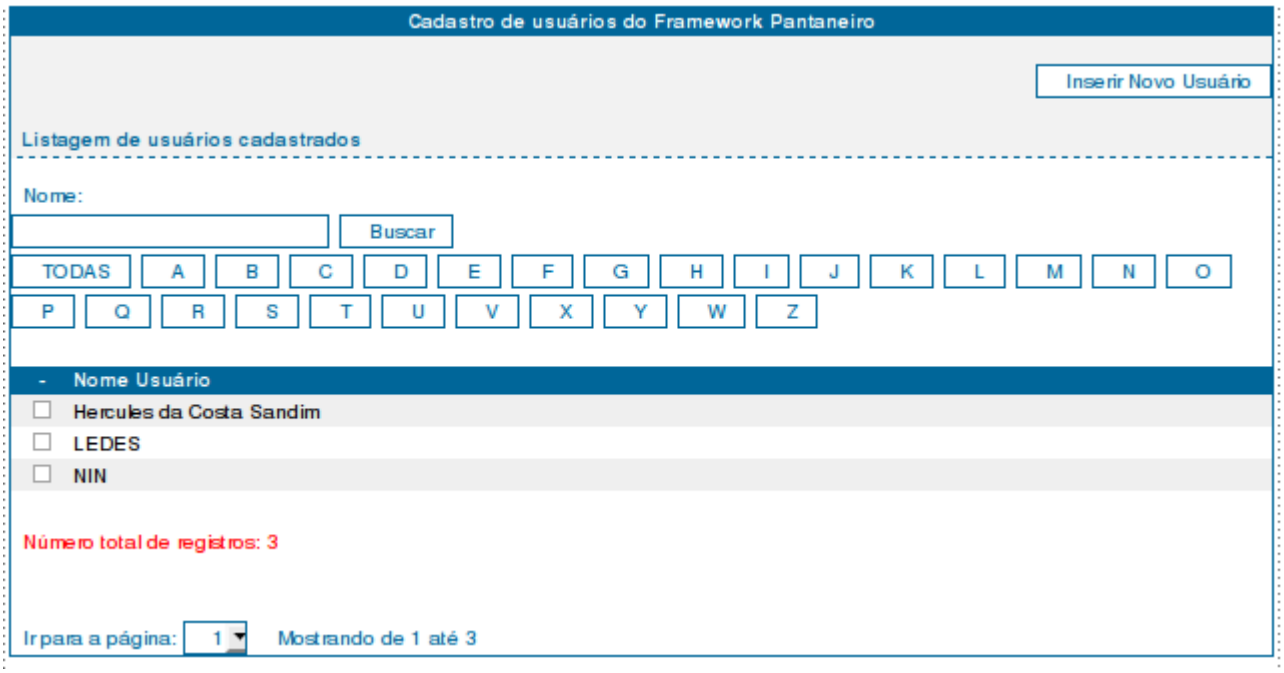


Figura 23– Tela de cadastro de usuários do framework Pantaneiro.

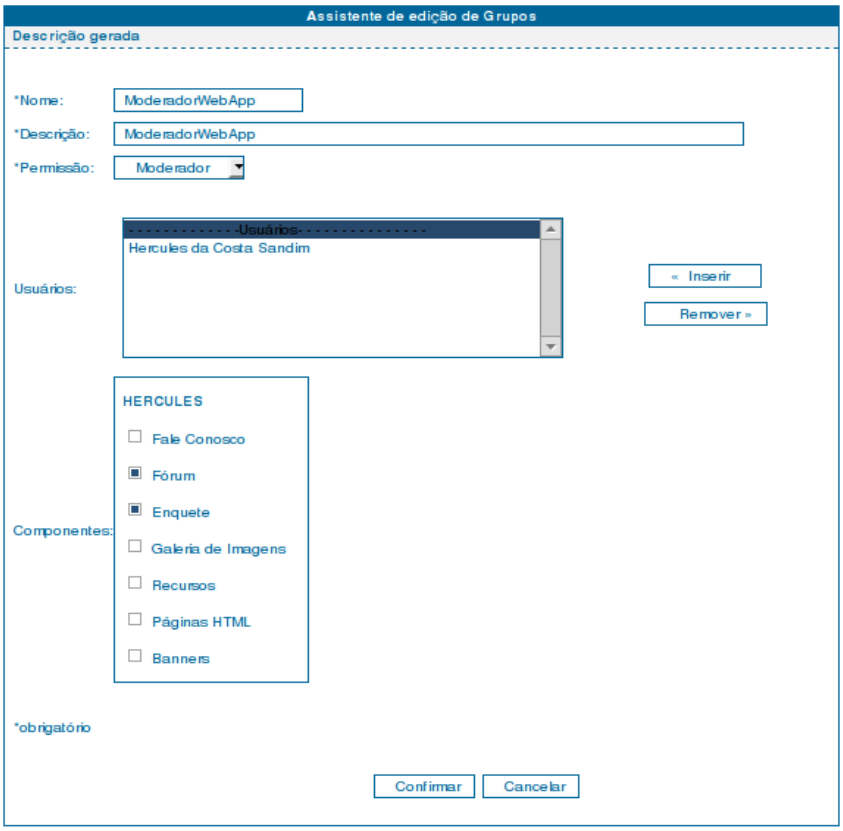


Figura 24 – Tela de edição de papéis do framework Pantaneiro.

- **Geração da aplicação**: O Gerador de Aplicação atua direto na Camada de Metadados, sobre os metadados de interface e de navegação, e é composto de códigos nas linguagens PHP, HTML, JAVASCRIPT com AJAX para a geração da WebApp em tempo de execução.

Dessa forma, a Gerência de Interfaces do framework Pantaneiro abrange a fase de modelagem de interface do HMBS/M estendido. Na Figura 25 é apresentada a área de edição da interface, enquanto que a Figura 26 apresenta a área de edição de estilos.

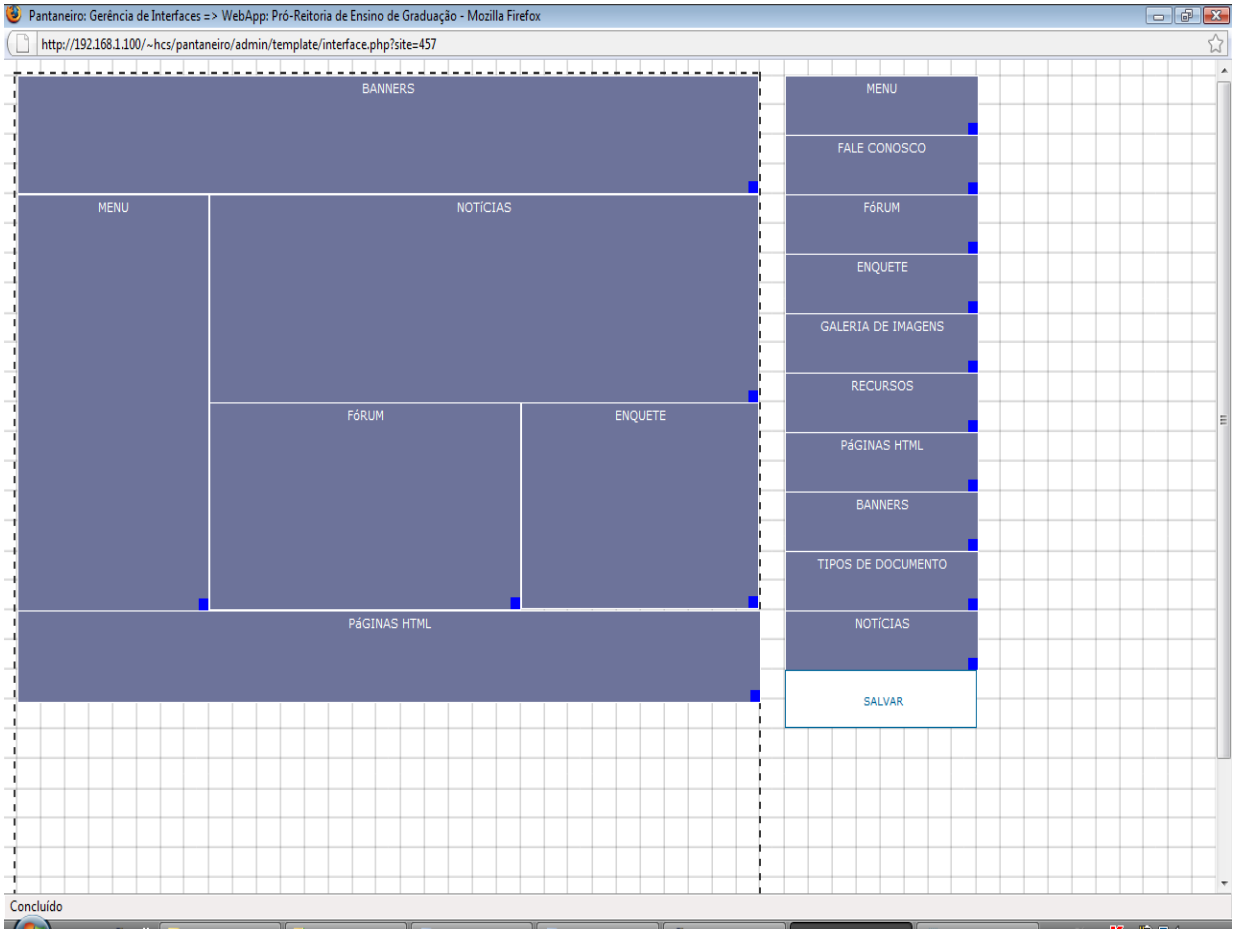


Figura 25 – Área de edição de interfaces do framework Pantaneiro.

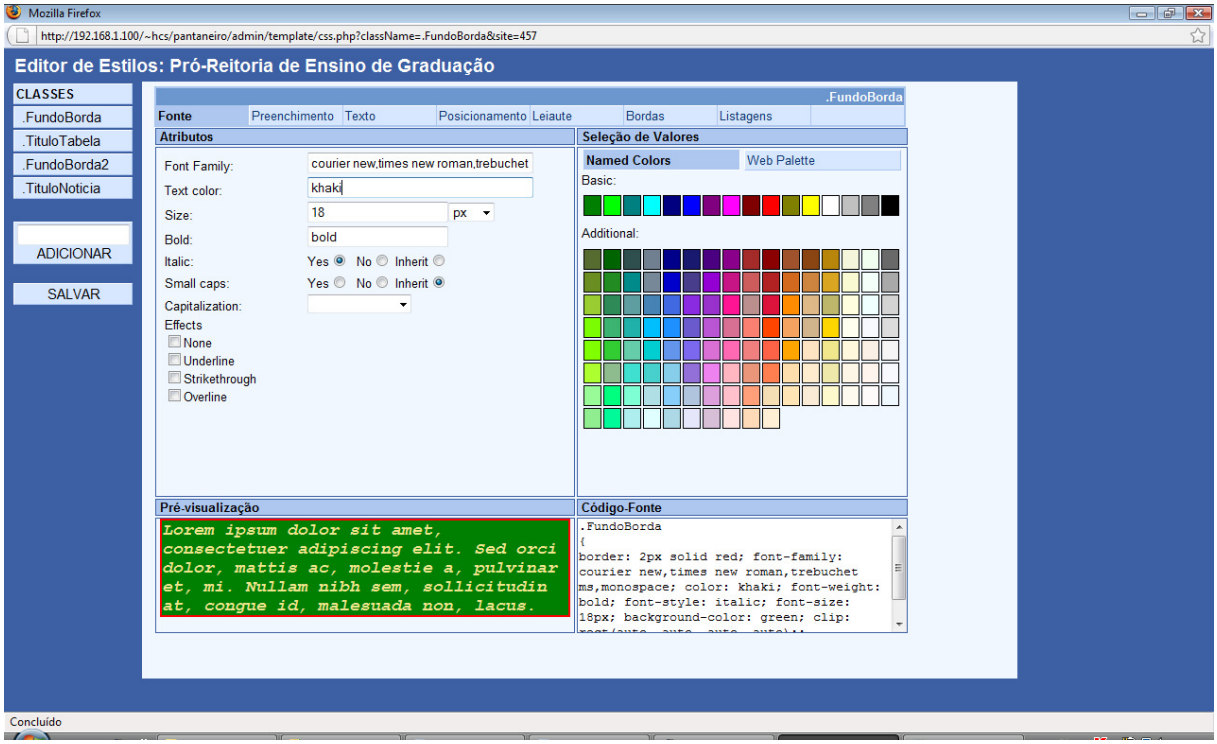


Figura 26 – Área de edição de estilos do framework Pantaneiro.

A Figura 27 ilustra um diagrama de caso de uso simplificado do Ambiente de Autoria do Wizard-Pantaneiro.

5.4.1.2 – AMBIENTE DE PROJETO NAVEGACIONAL

O ambiente de projeto navegacional é composto pelo módulo de Gerência Navegacional que será apresentado a seguir.

GERÊNCIA NAVEGACIONAL

A Gerência Navegacional utiliza os modelos de fatias gerados pela Gerência de Componentes para compor as mais variadas formas de visualização de um componente na WebApp gerada. Este módulo também configura a navegação entre os componentes para proporcionar a criação dos modelos de composição, navegacional de tipos e de contextos navegacionais da WebApp. Dessa forma, a Gerência Navegacional abrange a fase de modelagem navegacional do HMBS/M estendido.

Além de modelar a navegação entre os componentes, a gerência navegacional ainda é responsável pela composição do “menu” da WebApp, que juntamente com a navegação entre os componentes, irá compor o “mapa do site”. Na Figura 28 é apresentada a gerência navegacional na edição do “menu”.

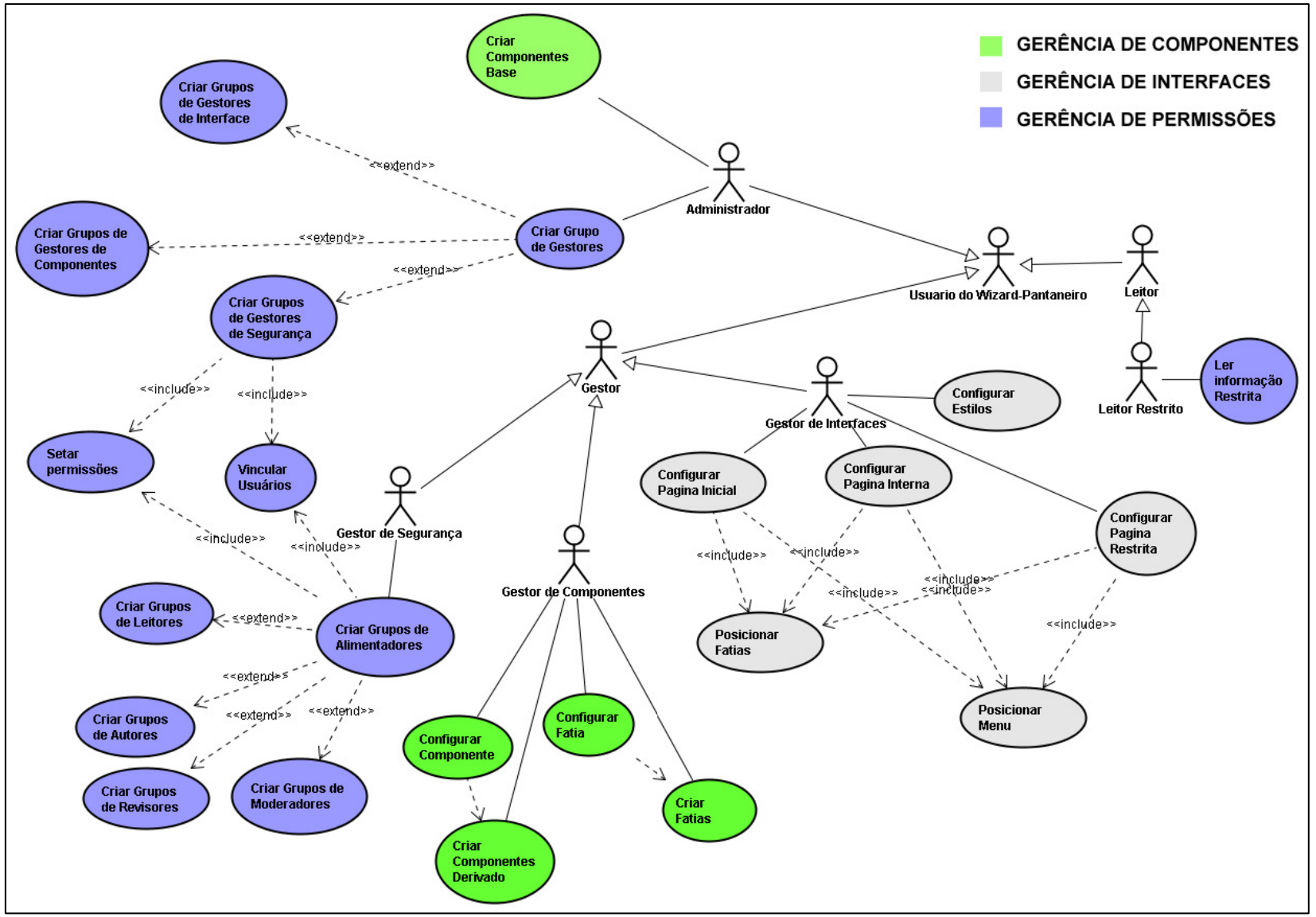


Figura 27 - Diagrama de Caso de Uso simplificado do Ambiente de Autoria.

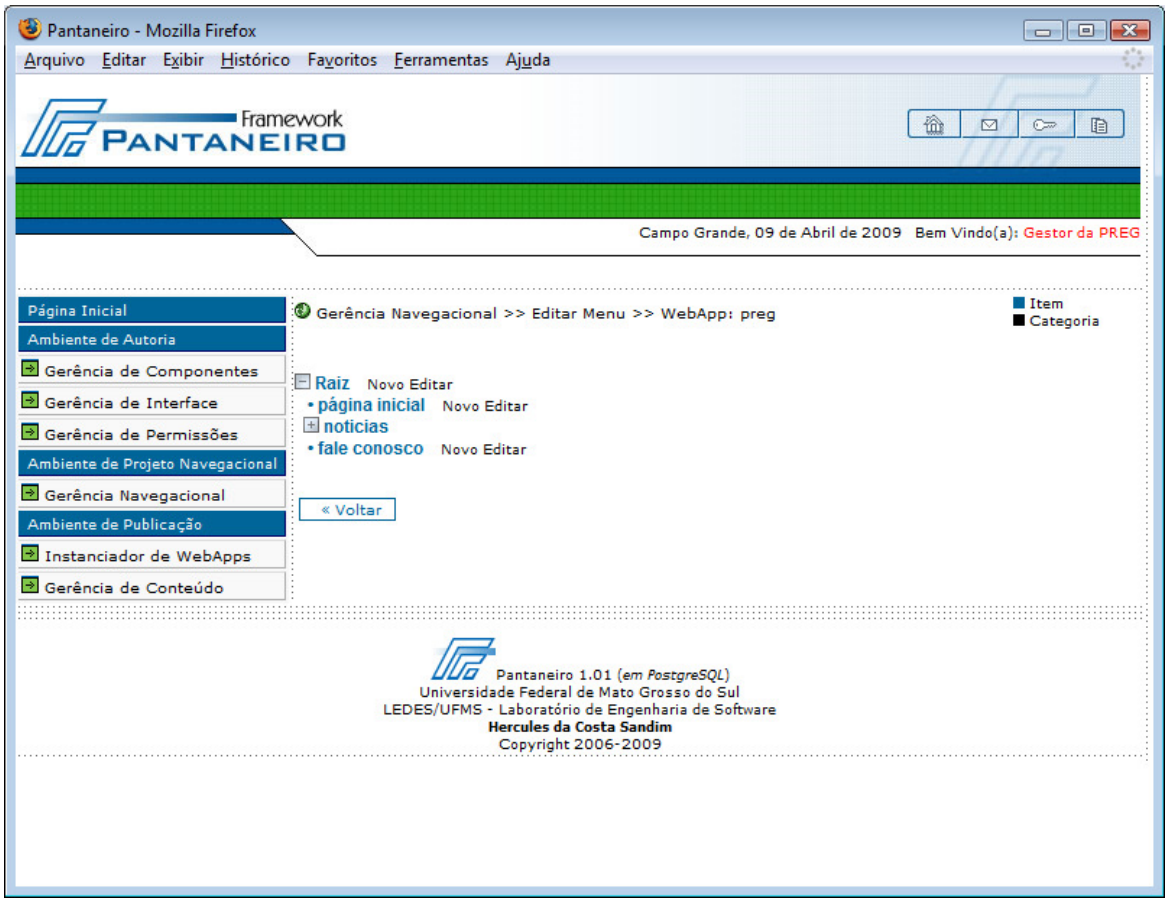


Figura 28 – Edição do “menu” da WebApp.

A Figura 29 ilustra um diagrama de caso de uso simplificado do Ambiente de Projeto Navegacional do Wizard-Pantaneiro.

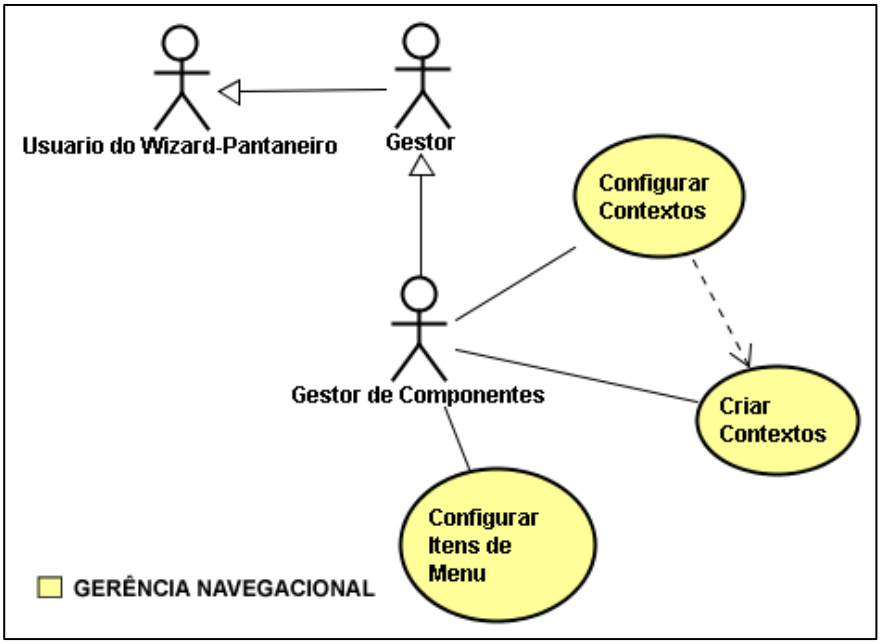


Figura 29 – Diagrama de Caso de Uso simplificado do Ambiente de Projeto Navegacional.

5.4.1.3 – AMBIENTE DE PUBLICAÇÃO

O ambiente de publicação é composto pelos módulos de Gerência de WebApps, Gerência de Conteúdo e Gerador de SCXML, que serão apresentados a seguir.

GERÊNCIA DE WEBAPPS

O módulo Gerência de WebApps é o módulo onde o administrador do framework Pantaneiro instancia, edita ou exclui as WebApps. As WebApps instanciadas serão portais de órgãos corporativos em uma dada plataforma e-gov, onde existe um "Ambiente Compartilhado" onde as WebApps conseguem realizar a comunicação entre si, compartilhando instâncias de componentes, recursos, banners e até mesmo componentes "inteiros".

Na Figura 30 é ilustrada a tela onde se realiza o processo de instanciação (processo inicial de criação) de uma nova WebApp. Esta tela consiste de um formulário para a inserção da "URL" da WebApp, da "Instituição" a qual a WebApp pertence (o framework Pantaneiro permite gerenciar mais de uma plataforma e-gov), a "sigla", o "Nome" e a "Descrição" da WebApp, além de um campo para marcar a publicação ou não da WebApp, um campo para marcar a exibição ou não do mapa do site na WebApp e a possível escolha de tornar a WebApp uma aplicação restrita, marcando-a como sendo uma aplicação "interna" a uma outra WebApp. O fato de uma WebApp **A** ser "interna" a uma WebApp **B**, significa que só poderão acessar a WebApp **A**, usuários que possuam permissões de gestor, moderador, revisor, autor ou leitor na WebApp **A**, após realizado *logon* na WebApp **A** através de uma interface de *login* que existirá na WebApp **B**.

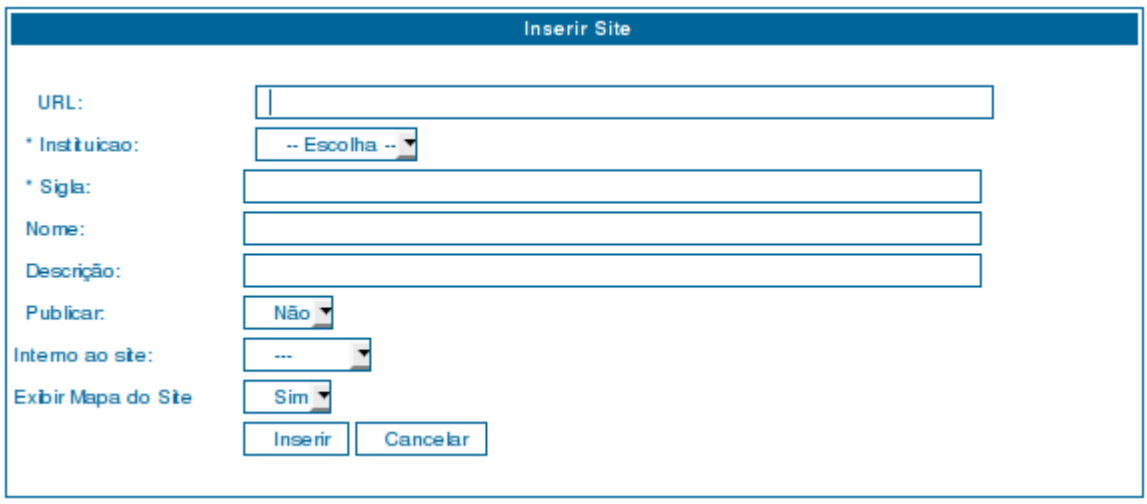


Figura 30 – Tela de instanciação da nova WebApp.

GERÊNCIA DE CONTEÚDO

Enquanto os serviços de portais corporativos se tornam mais volumosos e complexos, as administrações necessitam melhorar suas potencialidades da gerência. Uma das tarefas principais é a gestão de informação, um campo interdisciplinar que extraia e combine habilidades e recursos da tecnologia de informação, da gerência de registros e arquivos e da gerência geral (KLISCHEWSKI; JEENICKE, 2004).

Existem diversas ferramentas que possibilitam o gerenciamento de conteúdo de portais tais como Zope (ZOPE, 2009) e OpenCMS (ALKACON, 2009). Entretanto, segundo Souza *et al.* (2004), os resultados de busca em grandes portais que utilizam tais ferramentas, apresentam baixa revocação (razão entre o número de documentos recuperados relevantes e o número de documentos relevantes existentes) e baixa precisão (razão entre o número de documentos recuperados relevantes e o número de documentos recuperados).

Além disso, percebe-se que a informação recuperada é de baixa qualidade, pois os resultados recuperados geralmente diferem daquilo que efetivamente se busca. A principal causa deste comportamento é a falha dos mecanismos de busca em organizar conceitualmente os documentos em uma maneira mais próxima às necessidades do usuário. Por esse motivo o uso de ferramentas com um propósito mais específico, ou mesmo de propósito mais genérico, mas que consiga abranger o maior número de funcionalidades possível é imprescindível.

Na Figura 31 é ilustrada a tela de listagem de instâncias para o componente Enquete (outro componente do repositório de componentes do framework Pantaneiro). Na listagem de instâncias de informações de um componente, existe uma “legenda” informando o “status” da informação, em “Nova”, “Devolvidas”, “Recebidas” e “Publicadas”.

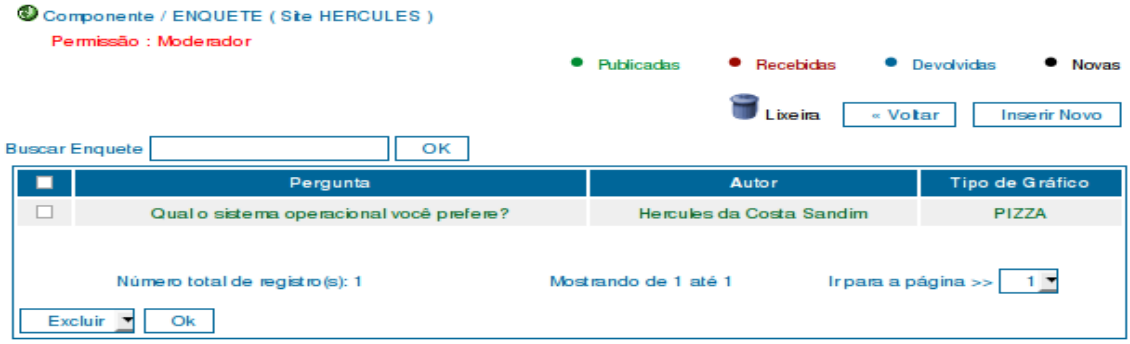


Figura 31 – Tela de listagem de enquetes.

A edição na instância de um componente permite visualizar o *log* de alterações realizadas naquela instância, além de possibilitar o controle de versões sobre a instância, criando uma nova versão ou reutilizando uma versão anterior. Esse controle de versão é uma funcionalidade própria deste módulo e não se utiliza de outras ferramentas. A Figura 32 ilustra a edição de uma instância do componente Enquete.

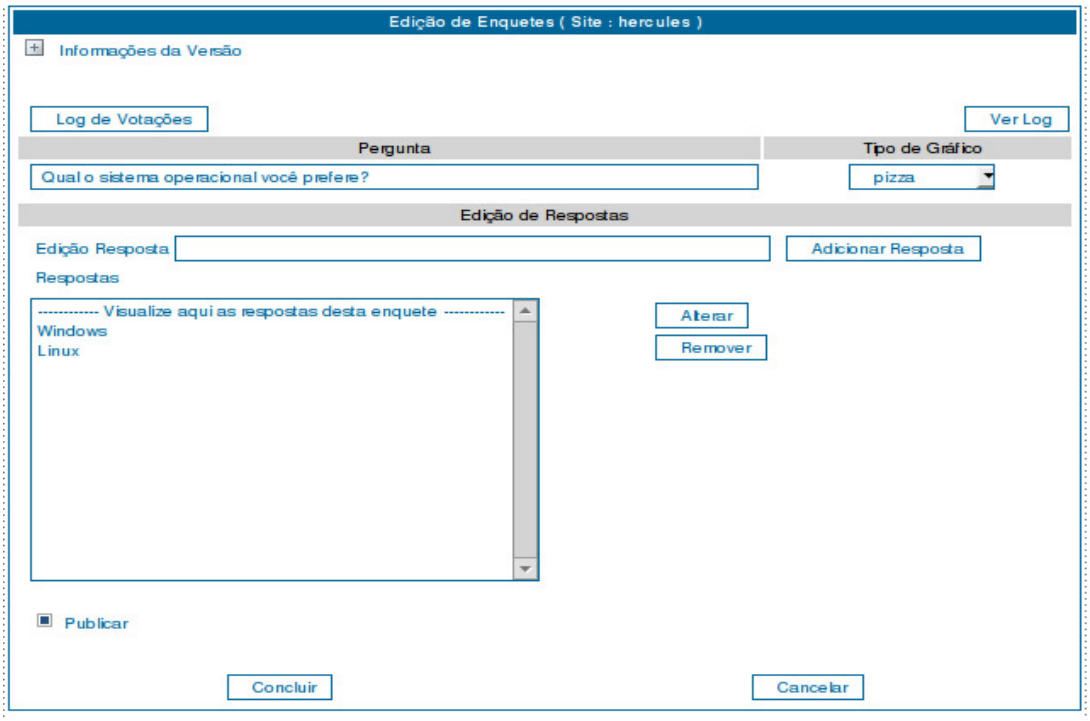


Figura 32 – Edição de uma instância do componente Enquete.

O WORKFLOW DE PUBLICAÇÃO

Workflows podem ser definidos como qualquer conjunto de atividades executadas de forma coordenada, em série e/ou em paralelo, por indivíduos ou grupos com um objetivo em comum. Sistemas de Gerência de Workflow (SGWs) visam automatizar

as atividades de administração/coordenação relativas à execução de workflows (MORO, 2009).

Devido ao grande fluxo e compartilhamento de informações, é fundamental que exista um protocolo de permissões comprometido a manter a integridade e consistência do conteúdo da ferramenta. A integridade e a consistência devem estar asseguradas nos níveis de criação, alimentação e visualização das informações contidas nos componentes. Em cada um destes níveis, os privilégios de manipulação são verificados através do papel definido para o usuário que está manipulando os dados.

O workflow de publicação do Ambiente de Publicação garante a integridade e a origem das informações. O autor da informação original é sempre preservado, bem como os autores das inúmeras alterações ao longo desse workflow. Esse controle é garantido por meio do desenvolvimento de um mecanismo de controle de versões.

No Pantaneiro, são contemplados os papéis de “autor”, “revisor” e “moderador”. A funcionalidade de cada um desses papéis, que compõem o workflow de publicações, é ilustrada na Figura 33.

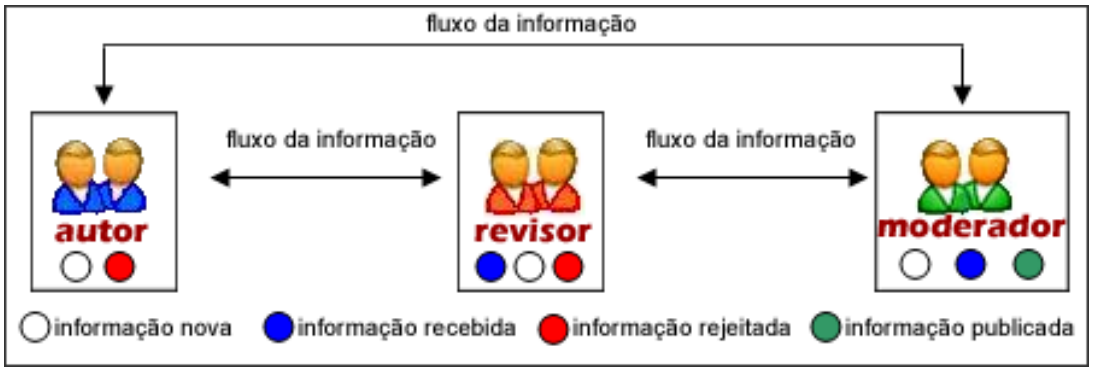


Figura 33 – *Workflow* de publicação das informações do Pantaneiro.

Dessa forma, a Gerência de Conteúdo abrange a fase de Publicação/Teste do HMBS/M estendido. Na Figura 34 é ilustrado um diagrama de caso de uso simplificado do Ambiente de Publicação do Wizard-Pantaneiro. Na Figura 35 é ilustrado o diagrama de caso de uso simplificado do Wizard-Pantaneiro.

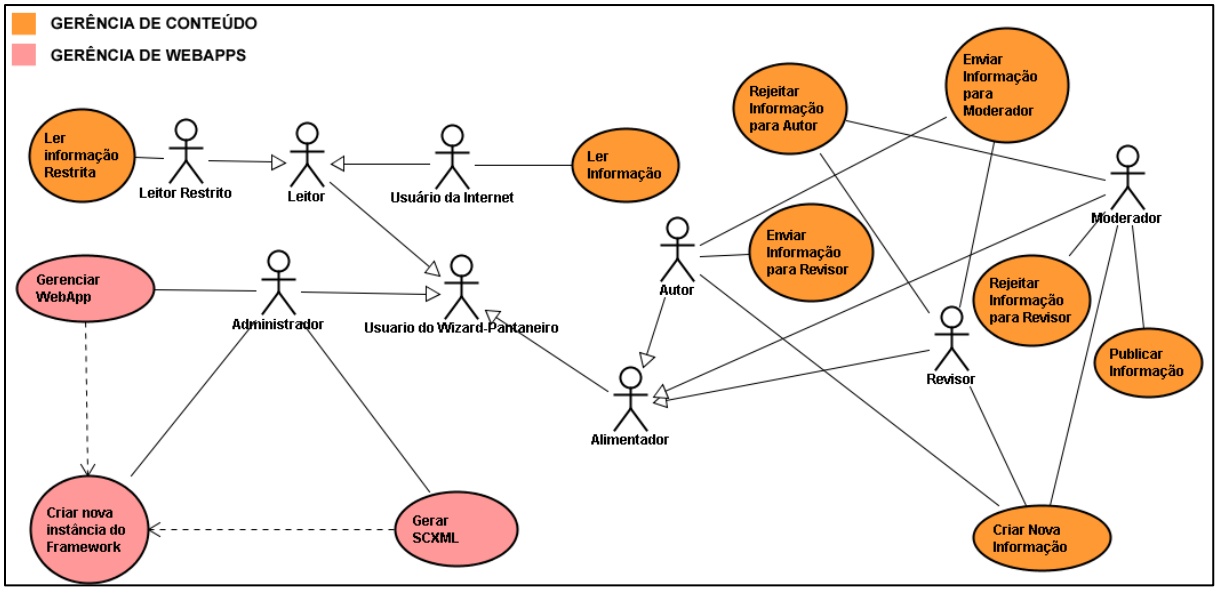


Figura 34 – Diagrama de Caso de Uso simplificado do Ambiente de Publicação.

GERADOR DE SCXML

O módulo Gerador de SCXML (*StateChart eXtensible Markup Language*) é responsável por gerar o modelo comportamental da WebApp, representado pelo formalisto de *Statecharts* (HAREL, 1987). O modelo comportamental é gerado em formato XML sobre o esquema SCXML (SCXML, 2009) definido pela W3C que padroniza a representação internacional de um diagrama de estados usando XML.

O padrão SCXML é composto pelos seguintes elementos:

- **scxml**: representa o elemento raiz, indicando a versão, o estado inicial e a notação de possíveis linguagens embarcadas. As propriedades desse elemento são:
  - **initialstate**: representa o estado inicial padrão para um diagrama de estados;
  - **xmlns**: representa a notação de linguagens embarcadas; e
  - **version**: representa a versão do documento SCXML.
- **state**: representa um estado do diagrama de estados. É composto das seguintes propriedades:
  - **id**: representa um "id" único de representação do estado.
  - **onentry**: representa o comportamento de entrada do estado, uma ou mais ações que são executadas sempre que a

execução entrar nesse estado, independentemente da transição que o leve até ele;

- **onexit**: representa o comportamento de saída do estado, uma ou mais ações que são executadas sempre que a execução deixar o estado, independentemente da transição que o faça sair dele; e
- **transition**: representa um conjunto de transições possíveis a partir de um determinado estado. Transições são relacionamentos direcionados entre um estado origem e um estado destino. Sua especificação inclui:
  - **event**: representa o evento (gatilho) que disparam a transição;
  - **cond**: uma condição (expressão *booleana*) que é avaliada quando uma ocorrência de evento capaz de provocar a transição é notificada. Assim, mesmo ocorrendo o evento, a transição só ocorre se sua "restrição de guarda" for satisfeita; e
  - **target**: representa o estado destino da máquina de estados, caso a transição ocorra.

O modelo comportamental (simplificado) da aplicação "Portal MS"[32] da plataforma e-gov do Governo do Estado de Mato Grosso do Sul, desenvolvida utilizando o framework Pantaneiro e o Wizard-Pantaneiro, é descrito em SCXML pelo Código-Fonte 1.

Este trabalho utilizou-se de APIs JAVA para implementar uma aplicação JAVA de visualização gráfica do diagrama de estados gerados pelo módulo Gerador de SCXML. Essa aplicação é denominada SCXML Viewer – Pantaneiro

[32] *Portal MS – Notícias.* Acesse http://www.ms.gov.br.

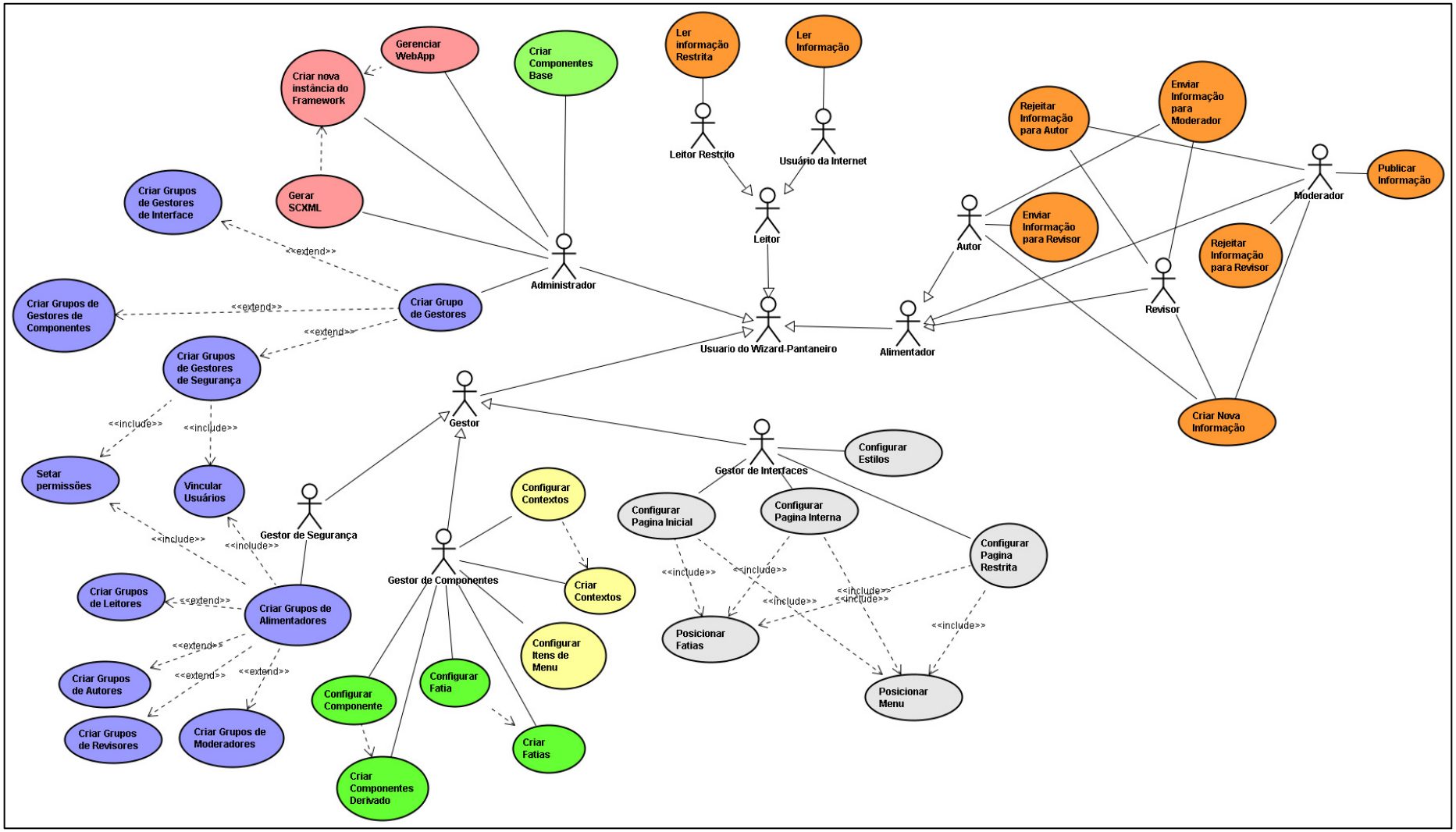
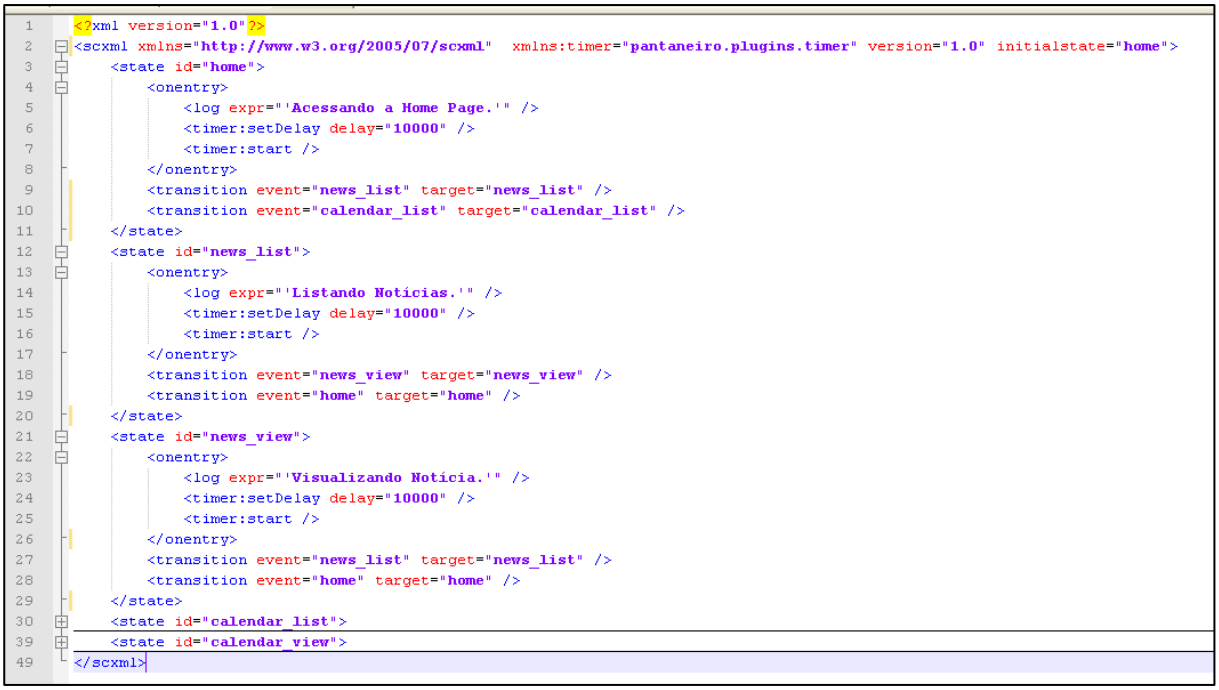


Figura 35 – Diagrama de Caso de Uso simplificado do Wizard-Pantaneiro.



```
<?xml version="1.0"?>
<scxml xmlns="http://www.w3.org/2005/07/scxml"  xmlns:timer="pantaneiro.plugins.timer" version="1.0" initialstate="home">
    <state id="home">
        <onentry>
            <log expr="'Acessando a Home Page.'" />
            <timer:setDelay delay="10000" />
            <timer:start />
        </onentry>
        <transition event="news_list" target="news_list" />
        <transition event="calendar_list" target="calendar_list" />
    </state>
    <state id="news_list">
        <onentry>
            <log expr="'Listando Noticias.'" />
            <timer:setDelay delay="10000" />
            <timer:start />
        </onentry>
        <transition event="news_view" target="news_view" />
        <transition event="home" target="home" />
    </state>
    <state id="news_view">
        <onentry>
            <log expr="'Visualizando Noticia.'" />
            <timer:setDelay delay="10000" />
            <timer:start />
        </onentry>
        <transition event="news_list" target="news_list" />
        <transition event="home" target="home" />
    </state>
    <state id="calendar_list">
    <state id="calendar_view">
</scxml>
```

Código-Fonte 1: Representação simplificada do SCXML da WebApp “Portal MS”.

A representação gráfica do SCXML exemplo (Código-Fonte 1) foi visualizada com a aplicação SCXML Viewer – Pantaneiro e é representada na Figura 36.

Pelo fato do Wizard-Pantaneiro adotar o padrão SCXML da W3C, o modelo comportamental pode ser visualizado em quaisquer ferramentas que implemente a visualização gráfica de arquivos SCXML. Este trabalho implementou uma aplicação simplificada em JAVA para mostrar q possibilidade de tal visualização gráfica do modelo.

5.5 – CONSIDERAÇÕES FINAIS

Este capítulo abordou os principais temas relacionados ao framework Pantaneiro. Sinteticamente, foram abordados o processo de construção do framework Pantaneiro, cujos frutos foram a análise do domínio e o projeto arquitetural do framework, as tecnologias adotadas no desenvolvimento do framework e o processo de instanciação do framework, sendo este realizado todo através de uma ferramenta de desenvolvimento denominada Wizard-Pantaneiro.

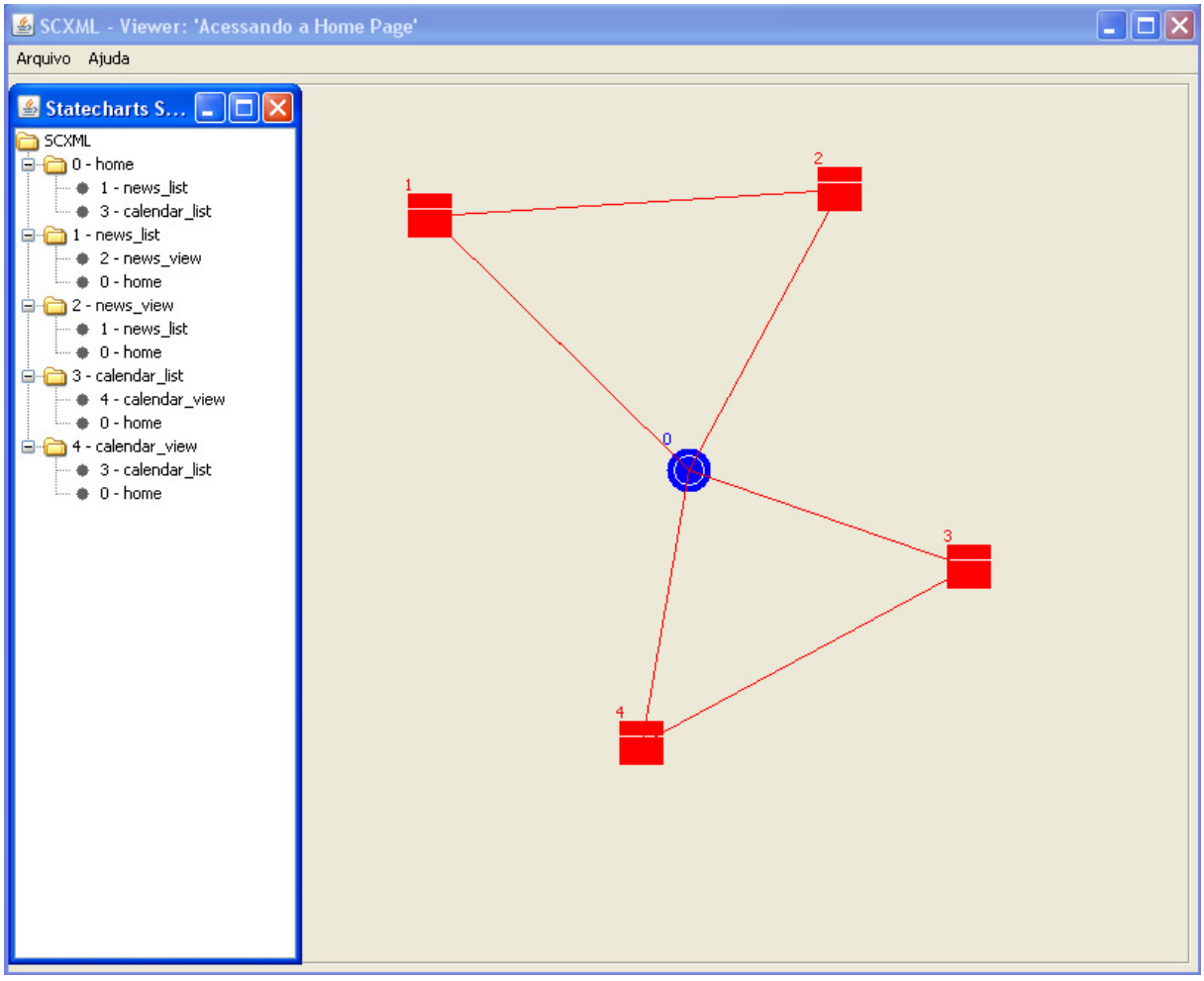


Figura 36 – Representação gráfica do SCXML simplificado (Código-Fonte 1).

Avaliação do Framework

6.1 – CONSIDERAÇÕES INICIAIS

Conforme apresentado nas motivações, este trabalho é fruto de uma parceria desde o ano de 2004 entre o Laboratório de Engenharia de Software e o Governo do Estado de Mato Grosso do Sul. No Governo de Mato Grosso do Sul, o framework Pantaneiro é visto como um framework de gerenciamento de conteúdo web, propiciando a disseminação de informações públicas de forma ágil, e quando necessário, em tempo real. No caso do Governo do Estado de Mato Grosso do Sul, o Pantaneiro cumpre também a proposta de padronizar a publicação das informações dos sites do Poder Executivo.

Na implantação realizada no Governo do Estado de Mato Grosso do Sul, para que não houvesse duplicidade de informações sobre os usuários, o Wizard-Pantaneiro utiliza a autenticação de usuários sobre uma base de diretórios LDAP (LDAP, 2009) centralizada e que armazena toda a estrutura organizacional de usuários da plataforma e-gov de Mato Grosso do Sul.

No Estado de Mato Grosso do Sul, o framework Pantaneiro foi implantado para testes em Junho de 2005, tendo seu uso impulsionado em 2006.

Segundo a equipe que utiliza o framewok Pantaneiro, primeiramente instalada na SGI e atualmente instalada na Governadoria do Estado de MS, dentre as principais funcionalidades do framework Pantaneiro, estão:

- Cadastro de várias WebApps com administrações independentes;

- Gerência de Componentes que permite facilmente a criação e extensão de componentes;
- Gerência de Interfaces ágil que facilita a modificação do layout/visual da WebApp, bem como distribuição dos componentes na tela;
- Gerência Navegacional para administração do menu e da navegação das WebApps;
- Gerência de Conteúdo com *workflow* baseado em grupos de usuários;
- Estrutura que possibilita a visualização de relatórios de estatísticas das WebApps;
- Controle de versão e *Log* para controle de alterações sobre as informações armazenadas;
- Possibilidade de expansão da ferramenta conforme demanda dos órgãos principalmente pelo fato do framework Pantaneiro ser um software-livre e de código-fonte aberto;
- Padronização das WebApps do Governo do Estado de Mato Grosso do Sul;
- Possibilidade de compartilhamento de informações entre os órgãos; e
- Possibilidade de publicação de conteúdos restritos e/ou instanciação de WebApps na intranet.

6.2 – PROCESSO DE INSTANCIAÇÃO

O processo de instanciação do framework Pantaneiro no Governo do Estado de Mato Grosso do Sul segue um processo muito bem definido que envolve o órgão que deseja um portal e-gov no framework e a equipe de TI da SGI. O processo, ilustrado pela Figura 37, é descrito a seguir:

- O órgão **A** toma conhecimento do framework Pantaneiro e a possibilidade de sua utilização;
- O órgão **A** estuda a documentação do framework Pantaneiro, composta de manuais, apostilas, vídeo-tutoriais, tutoriais e guia de utilização;
- O órgão **A** preenche o Formulário de Solicitação, conforme modelo de ofício apresentado no Anexo I, e envia à SGI;
- O órgão **A** preenche o Formulário de Proposta de Projeto, conforme modelo descrito no Anexo II;

- A equipe de TI da SGI e a equipe de TI do órgão **A** avaliam:
  - Viabilidade de instanciação da WebApp; e
  - Se órgão **A** possui demandas de serviços não contemplados pelo framework Pantaneiro.
- Ambas as equipes de TI traçam um Cronograma de Instanciação: cada projeto de nova WebApp terá um período previsto para que as equipes de TI viabilizem sua instanciação.
- O órgão **A** preenche o Formulário de Solicitação de Validação e Publicação, conforme modelo de ofício no Anexo III; e
- A SGI valida a WebApp instanciada e caso esteja funcional e atenda as necessidades e padrões do Governo do Estado de Mato Grosso do Sul, a WebApp é publicada.

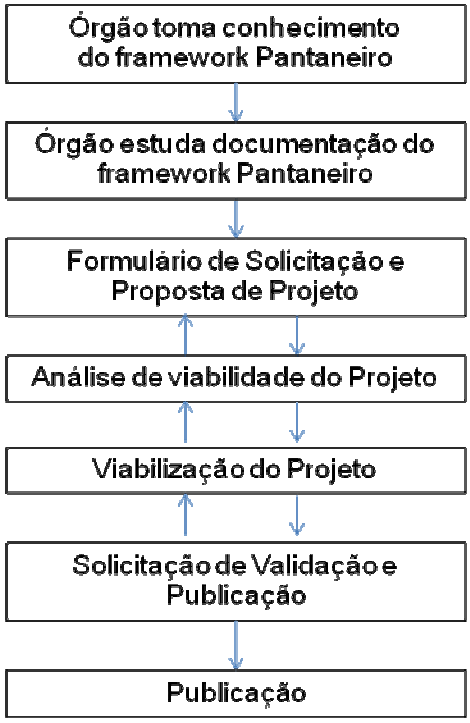


Figura 37 – Processo de instanciação do framework Pantaneiro no Governo do Estado de Mato Grosso do Sul.

Algumas informações importantes para o preenchimento do Formulário de Proposta de Projeto são descritas a seguir:

- É importante que o levantamento do conteúdo da WebApp já tenha sido feito antes do preenchimento do formulário;

- O formulário do projeto é um resumo do que a WebApp conterá, portanto o órgão pode optar por ter um projeto mais completo;
- Em todas as áreas que estiver pedindo e-mail dos usuários é necessário informar preferencialmente o e-mail institucional;
- O tipo de WebApp é importante para classificar como sendo uma WebApp de internet (público) ou WebApp de intranet (acessado somente por login). Em geral haverá uma WebApp intranet a partir de um endereço de uma WebApp internet, porém isso não é obrigatório. O órgão pode optar por ter inicialmente só uma WebApp de intranet também;
- A parte sobre "informações restritas" significa que o usuário pretende publicar informações na WebApp internet/intranet que serão lidas somente por algumas pessoas/grupos. Essa informação é útil para que se possa definir os acessos às suas informações;
- A WebApp que tiver a área interativa do Fale Conosco deverá já informar no projeto quais são os assuntos e seus responsáveis. Isso não significa que no futuro o Fale Conosco não possa ser alterado, mas para que a equipe da SGI visualize se todos usuários responsáveis possuem contas de e-mail institucional;
- O "Mapa do Site" e "Detalhes de cara área do site" é a parte do formulário que coletará informações para "análise do conteúdo da WebApp **x** Componentes do Framework Pantaneiro". É importante que coloquem no projeto o tipo do dado que será colocado em cada área (texto, imagem, download, etc.); e
- Para o planejamento do *workflow* de criação e aprovação de conteúdo, assinalar um ou mais papéis que serão utilizados na WebApp.

6.3 – PLATAFORMA E-GOV MS

Até junho de 2009, o framework Pantaneiro instanciou 54 WebApps na plataforma e-gov de Mato Grosso do Sul, compreenendo WebApps de secretarias, agências, grupamento de bombeiros, batalhões policiais, centros culturais, coordenadorias, conselhos, defensorias, fundações, procuradorias, entre outros órgãos de governo, como indicadas abaixo:

1. 14º Batalhão de Polícia Militar Rodoviária (PRE): http://www.pre.ms.gov.br;
2. 15º Batalhão de Polícia Militar Ambiental (PMA): http://www.pma.ms.gov.br;
3. 6º Grupamento de Bombeiros de MS (6GB): http://www.bombeiros.ms.gov.br/6gb;
4. Agência de Desenvolvimento Agrário e Extensão Rural de Mato Grosso do Sul (AGRAER): http://www.agraer.ms.gov.br;
5. Agência Estadual de Administração do Sistema Penitenciário (AGEPEN): http://www.agepen.ms.gov.br;
6. Agência Estadual de Defesa Sanitária Animal e Vegetal (IAGRO): http://www.iagro.ms.gov.br;
7. Agência Estadual de Gestão de Empreendimentos (AGESUL): http://www.agesul.ms.gov.br;
8. Agência Estadual de Imprensa Oficial de Mato Grosso do Sul: http://www.imprensaoficial.ms.gov.br;
9. Agência Estadual de Regulação de Serviços Públicos de MS (AGEPAN): http://www.agepan.ms.gov.br;
10. Banco da Gente: http://www.bancodagente.ms.gov.br;
11. Central de Compras: http://www.centraldecompras.ms.gov.br;
12. Centro de Convenções "Arquiteto Rubens Gil de Cammilo": http://www.centrodeconvecoes.ms.gov.br;
13. Centro Cultural José Otávio Guizzo: http://www.fundacaodecultura.ms.gov.br/centrocultural;
14. Centro Integrado de Operações de Segurança (CIOPS): http://www.ciops.ms.gov.br;
15. Centro Estadual de Monitoramento do Tempo, Clima Recursos Hidricos: http://www.agraer.ms.gov.br/cemtec;
16. Companhia Indepedente de Polícia Militar de Trânsito (CIPTRAN): http://www2.pm.ms.gov.br/ciptran;
17. Coordenadoria Especial de Políticas Públicas para Juventude: http://www.juventude.ms.gov.br;
18. Coordenadoria de Fiscalização de Mercadorias em Trânsito (COFIMT/SEFAZ): http://www3.sefaz.ms.gov.br/cofimt;

19. Coordenadoria de Polícia Comunitária/SEJUSP: http://www.sejusp.ms.gov.br/polcomunitaria;
20. Conhecimento de Transporte Eletrônico CT-e/SEFAZ: http://www.cte.ms.gov.br;
21. Conselho Estadual de Educação de Mato Grosso do Sul (CEE-MS): http://www.cee.ms.gov.br;
22. Conselho Estadual de Transito de Mato Grosso do Sul (CETRAN-MS): http://www.cetran.ms.gov.br;
23. Corpo de Bombeiros Militar de Mato Grosso do Sul: http://www.bombeiros.ms.gov.br;
24. Defesa Civil: http://www.defesacivil.ms.gov.br;
25. Defensoria Pública-Geral: http://www.defensoria.ms.gov.br;
26. Empresa de Gestão de Recursos Humanos e Patrimônio: http://www.egrhp.ms.gov.br;
27. Escola de Saúde Pública (ESP): http://www.esp.ms.gov.br;
28. Escrituração Fiscal Digital (EFD): http://www.efd.ms.gov.br;
29. Framework Pantaneiro (SGI): http://www.sgi.ms.gov.br/frameworkpantaneiro;
30. Fundação de Cultura de Mato Grosso do Sul: http://www.fundacaodecultura.ms.gov.br;
31. Fundação de Turismo (FUNDTUR): http://www.turismo.ms.gov.br;
32. Fundação do Desporto e Lazer (FUNDESPORTE): http://www.fundesporte.ms.gov.br;
33. Fundação Escola de Governo (EscolaGov): http://www.escolagov.ms.gov.br;
34. JUCEMS - Junta Comercial do Estado de Mato Grosso do Sul: http://www.jucems.ms.gov.br;
35. Notícias: http://www.noticias.ms.gov.br;
36. Nosso Portal (SEFAZ): http://www3.sefaz.ms.gov.br;
37. Polícia Civil (PCMS): http://www.pc.ms.gov.br;
38. Portal MS: http://www.ms.gov.br;
39. PROCON - Superintendência para Orientação e Defesa do Consumidor/SETAS: http://www.procon.ms.gov.br;
40. Procuradoria Geral do Estado (PGE): http://www.pge.ms.gov.br;
41. PROERD / SEJUSP: http://www.proerd.ms.gov.br;

42. Secretaria de Estado de Adminstração de Mato Grosso do Sul (SAD): http://www.sad.ms.gov.br;
43. Secretaria de Estado de Educação (SED): http://www.sed.ms.gov.br;
44. Secretaria de Estado de Habitação de Mato Grosso do Sul (SEHAB): http://www.sehab.ms.gov.br;
45. Secretaria de Estado de Justiça e Segurança Pública (SEJUSP): http://www.sejusp.ms.gov.br;
46. Secretaria de Estado do Meio Ambiente, das Cidades, do Planejamento, da Ciência e Tecnologia (SEMAC): http://www.semac.ms.gov.br;
47. Secretaria de Estado de Obras Públicas (SEOP): http://www.seop.ms.gov.br;
48. Secretaria de Estado de Desenvolvimento Agrário, da Produção, da Indústria, do Comércio e do Turismo (SEPROTUR): http://www.seprotur.ms.gov.br;
49. Servidor do Estado de Mato Grosso do Sul: http://www.servidor.ms.gov.br;
50. Secretaria de Trabalho, Assistência Social e Economia Solidária (SETASS): http://www.setass.ms.gov.br;
51. Superintendência de Gestão da Informação - SGI/SEFAZ: http://www.sgi.ms.gov.br;
52. Departamento de Operações de Fronteira (DOF): http://www.dof.ms.gov.br;
53. Tribunal Administrativo Tributário (TAT/SEFAZ): http://www3.sefaz.ms.gov.br/tat; e
54. Programa de Zoneamento Ecologico-Economico de Mato Grosso do Sul (ZEEMS): http://www.semac.ms.gov.br/zeems.

6.4 – PORTAL DA PREFEITURA MUNICIPAL DE PONTA PORÃ

Como estudo de caso foi desenvolvido um portal para a Prefeitura Municipal de Ponta Porã utilizando o Pantaneiro a fim de realizar uma análise comparativa contra seu atual site. A Figura 38 apresenta a interface da página inicial do atual site da Prefeitura Municipal de Ponta Porã.

Atualmente o site da Prefeitura Municipal de Ponta Porã apresenta os seguintes itens: Notícias, Banners, Menu, Galeria de Imagens (não funcional), Secretarias Municipais, Governo, Legislação e Fale Conosco. Com a nova modelagem para um portal da Prefeitura Municipal de Ponta Porã no Pantaneiro, os seguintes componentes tornaram-se disponíveis:

Figura 38 – Atual site da Prefeitura Municipal de Ponta Porã.

- **Notícias**: componente derivado do componente NotíciasBase (do repositório de componentes e-gov do framework Pantaneiro), o que permite o compartilhamento de suas informações com os demais portais desenvolvidos com o Pantaneiro. Armazena notícias informativas da Prefeitura Municipal de Ponta Porã e de seus parceiros;
- **Agenda**: componente derivado do componente AgendaBase (do repositório de componentes e-gov do framework Pantaneiro) similarmente ao componente notícias. Armazena informações de eventos da Prefeitura Municipal de Ponta Porã;
- **Fórum**: componente (do repositório de componentes e-gov do framework Pantaneiro) que implementa fóruns de discussões à Prefeitura Municipal de Ponta Porã;

- **Banners**: componente (do repositório de componentes e-gov do framework Pantaneiro) que torna disponíveis as *logos* e imagens no portal da Prefeitura Municipal de Ponta Porã;
- **Menu Horizontal**: componente que realiza a apresentação do menu no portal da Prefeitura Municipal de Ponta Porã. Este componente sintetiza o "mapa do site";
- **Busca**: componente que torna disponível a busca rápida por informações contidas no portal da Prefeitura Municipal de Ponta Porã;
- **Fale Conosco**: componente (do repositório de componentes e-gov do framework Pantaneiro) que torna disponível um canal de comunicação de duas vias entre a população de Ponta Porã (ou geral) e os servidores municipais de Ponta Porã;
- **Recursos para *download***: componente (do repositório de componentes e-gov do framework Pantaneiro) que apresenta todos os arquivos disponíveis para *download*;
- **Secretarias**: componente que representa/apresenta informações sobre as Secretarias Municipais da Prefeitura Municipal de Ponta Porã;
- **Galeria de Imagens**: componente (do repositório de componentes e-gov do framework Pantaneiro) que implementa a visualização de galerias de imagens da Prefeitura Municipal de Ponta Porã; e
- **Enquetes**: componente (do repositório de componentes e-gov do framework Pantaneiro) que implementa a ferramenta Enquete.

Para modelar o portal da Prefeitura Municipal de Ponta Porã foram criadas as seguintes fatias:

- **Notícias**: Listagem de Notícias; Visualização de Notícias e Listagem com Imagens;
- **Eventos**: Listagem de Eventos da Página Inicial; Listagem geral de eventos e Visualização de Eventos; e
- **Secretarias:** Listagem de Secretarias e Visualização de Secretarias.

As Figuras 39 e 40 ilustram a representação gráfica das fatias Listagem de notícias e Visualização de eventos. Com as fatias criadas, foram criados os seguintes contextos:

- **Notícias**: Listagem de todas as notícias; Visualização de notícia e Listagem de Destaques;
- **Eventos**: Listagem dos últimos eventos e Visualização de eventos; e
- **Secretarias**: Listagem de todas as secretarias e Visualização de secretaria.

Figura 39 – Fatia Listagem de Notícias (com seus atributos).

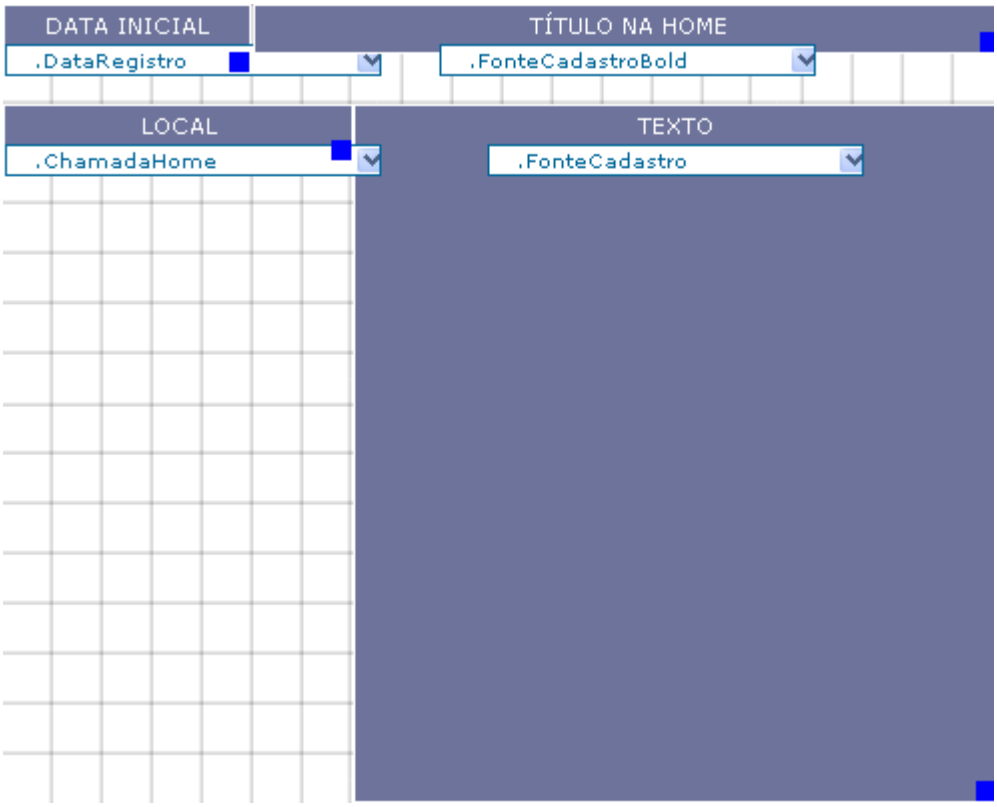


Figura 40 – Fatia Visualização de evento (com seus atributos).

O contexto Visualização de secretaria é formado das seguintes configurações:

- **Componente**: Secretaria.
- **Nome**: Visualização de secretaria.
- **Atributos**: Título, Chamada, Foto, Nome, Endereço, Telefone e Email.
- **Limite de registros**: 1.
- **Ordenação**: sem critérios.
- **Condição**: sem condições.

- **Contexto de Navegação**: Listagem de secretarias.

A Figura 41 ilustra a representação gráfica do contexto Visualização de secretaria no Pantaneiro.

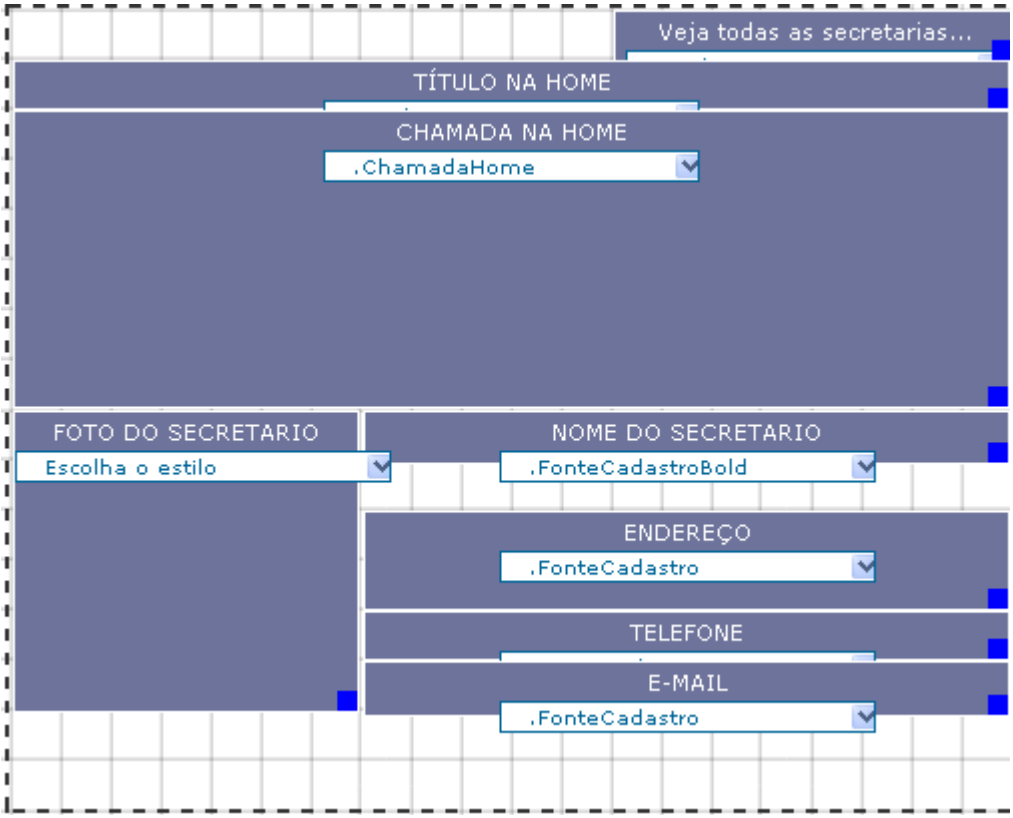


Figura 41 – Representação gráfica do contexto Visualização de secretaria.

Pode-se perceber que nas representações gráficas de configuração das fatias e dos contextos no Pantaneiro, que há uma associação entre classe de estilos (CSS) e atributos do componente, configurando a apresentação do contexto (ou fatia) na interface final da aplicação. A Figura 42 ilustra a apresentação do contexto da Figura 41 na interface final da aplicação. O mesmo é deslumbrado para o contexto Listagem de destaques na Figura 43.

A Figura 44 ilustra a configuração de estilos. Estes estilos são os mesmos que são utilizados para configurar a apresentação dos atributos (nos contextos e fatias). A Figura 45 ilustra a configuração da página inicial, onde são configurados a disposição e dimensão dos componentes na página.

Figura 42 – Apresentação final do contexto Visualização de secretaria.

Figura 43 – Representação gráfica e apresentação final do contexto Listagem de Destaques.

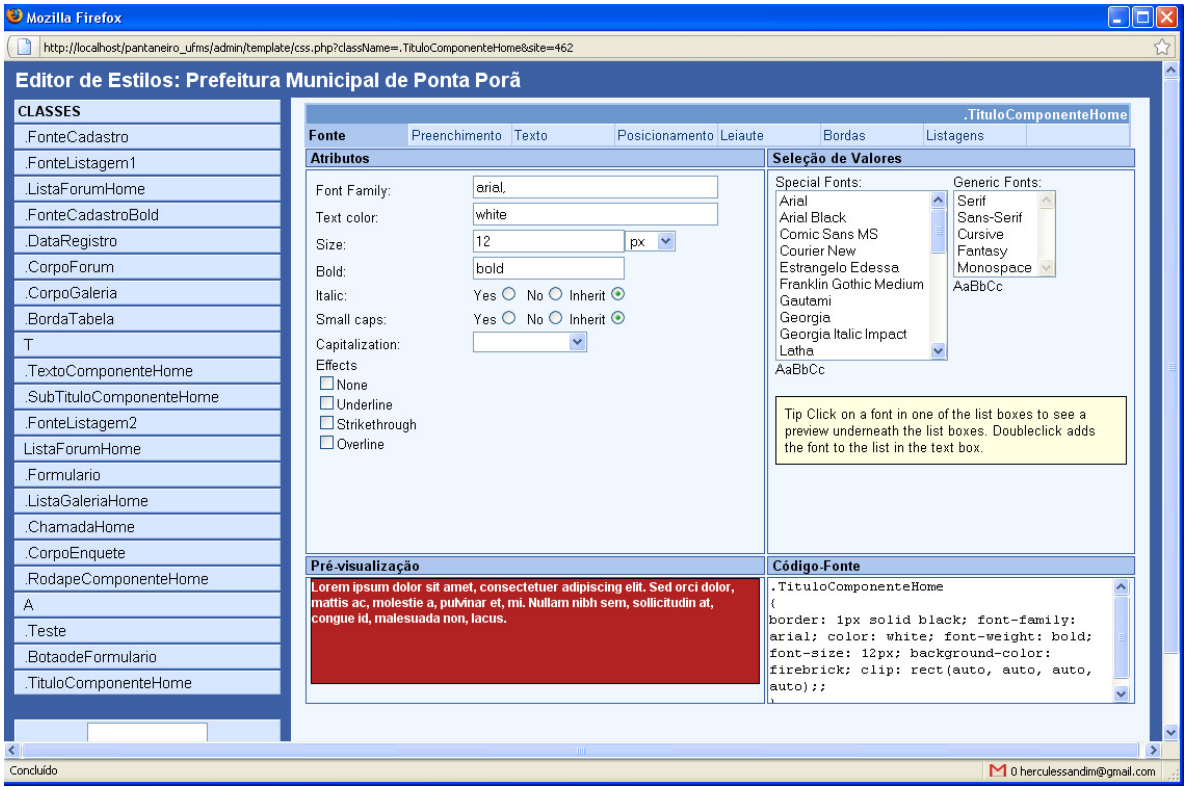


Figura 44 – Configuração de estilos.

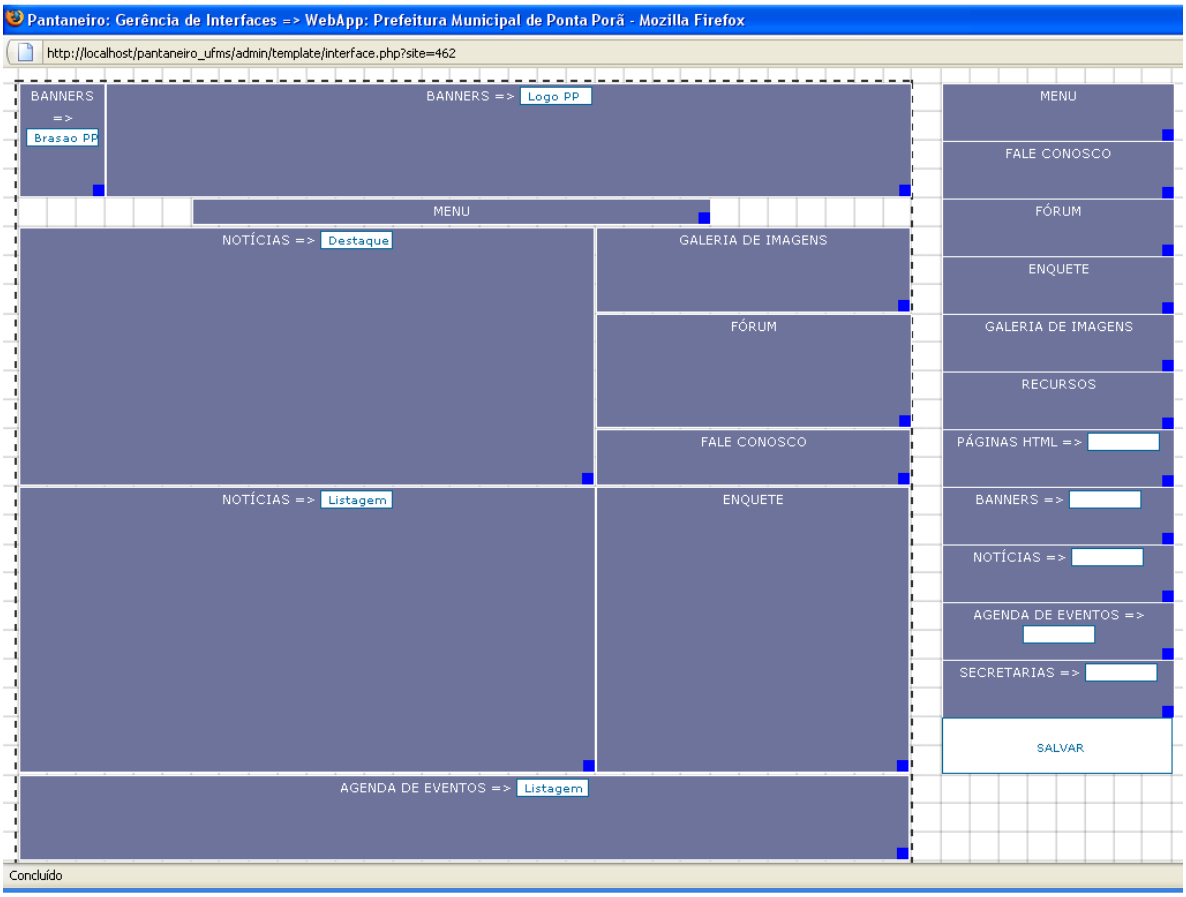


Figura 45 – Configuração da Interface Página Inicial.

A Figura 46 ilustra a configuração das páginas internas da aplicação. Nesta configuração, toma-se como base a configuração da página inicial e define-se uma área de navegação (ÁREA FIXA), que será a região onde os contextos serão apresentados seguindo a navegação configurada nos passos anteriores.

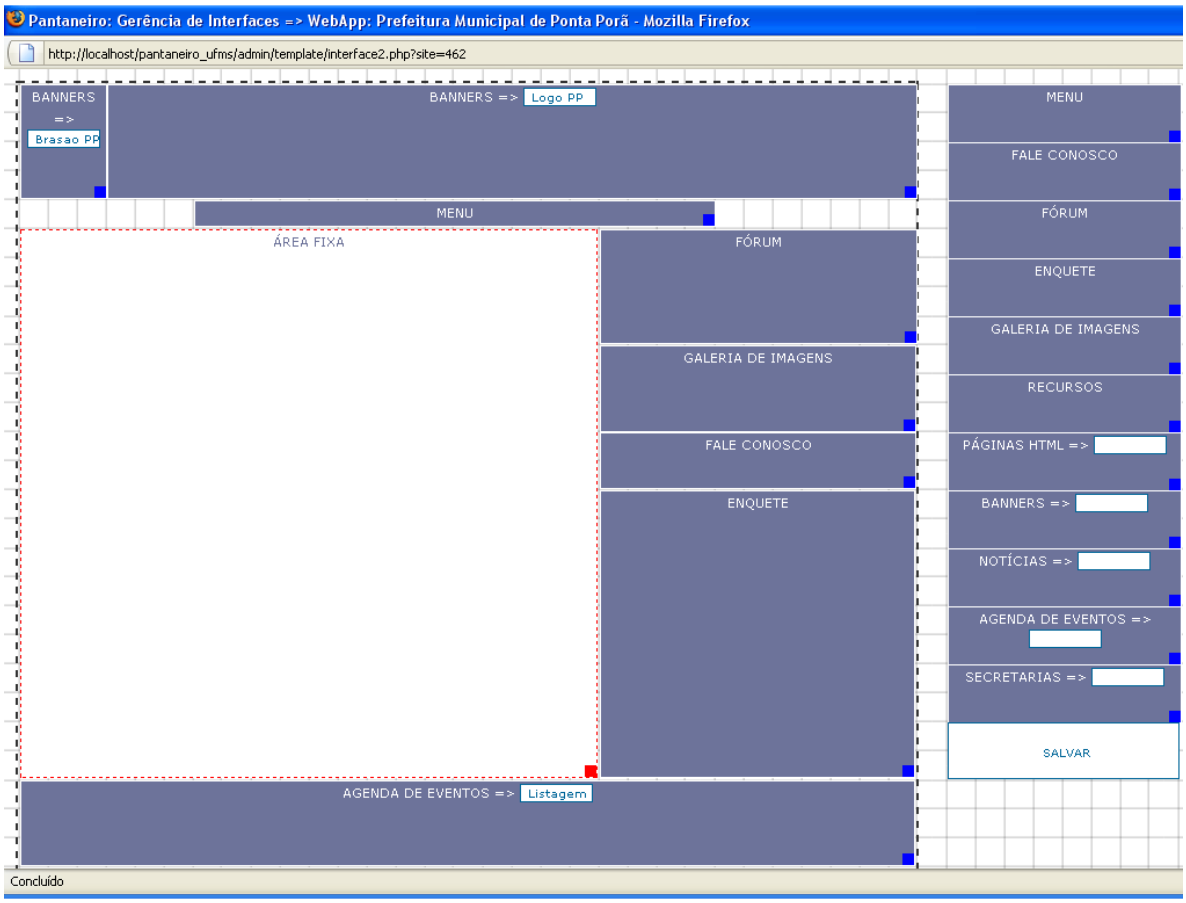


Figura 46 – Configuração da Interface das páginas internas.

Um exemplo do povoamento da aplicação é visualizado na Figura 47, com a gerência de conteúdo do componente Secretarias.

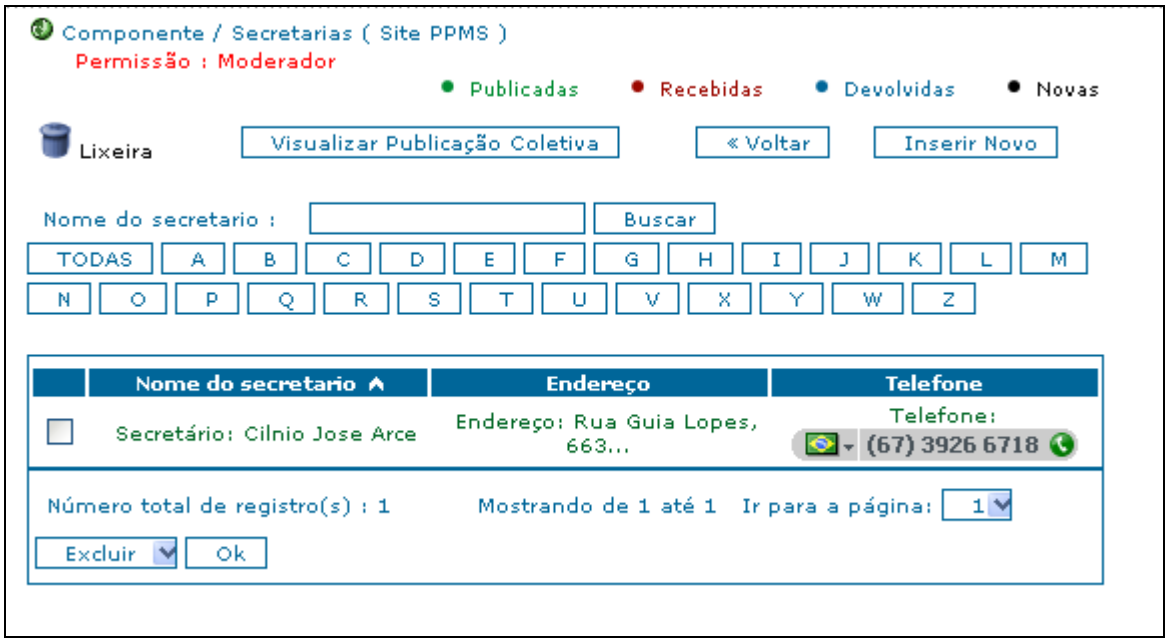


Figura 47 – Gerência de Conteúdo do componente Secretarias.

Acessando o módulo Gerência de WebApps do Wizard-Pantaneiro é possível gerar o *Statecharts* do portal da Prefeitura Municipal de Ponta Porã. Este é gerado em XML (esquema SCXML) e é apresentado pelo Código-Fonte 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<scxml xmlns="http://www.w3.org/2005/07/scxml" version="1.0" initialstate="home">
<state id="home">
    <onentry>
        <log expr="'Pagina Inicial'" />
    </onentry>
    <transition event="FALECONOSCO" target="FALECONOSCO"/>
    <transition event="FORUM_LIST" target="FORUM_LIST"/>
    <transition event="FORUM_VIEW" target="FORUM_VIEW"/>
    <transition event="ENQUETE_VOTE" target="ENQUETE_VOTE"/>
    <transition event="ENQUETE_RESULT" target="ENQUETE_RESULT"/>
    <transition event="GALERIA_LIST" target="GALERIA_LIST"/>
    <transition event="GALERIA_VIEW" target="GALERIA_VIEW"/>
    <transition event="Visualizar_Noticia" target="Visualizar_Noticia"/>
    <transition event="Visualizar_Evento" target="Visualizar_Evento"/>
    <transition event="MENU_3" target="MENU_3"/>
    <transition event="MENU_4" target="MENU_4"/>
    <transition event="MENU_10" target="MENU_10"/>
    <transition event="MENU_11" target="MENU_11"/>
    <transition event="MENU_1" target="MENU_1"/>
    <transition event="MENU_2" target="MENU_2"/>
    <transition event="MENU_6" target="MENU_6"/>
    <transition event="MENU_13" target="MENU_13"/>
    <transition event="MENU_7" target="MENU_7"/>
    <transition event="MENU_8" target="MENU_8"/>
    <transition event="MENU_12" target="MENU_12"/>
</state>
<state id="FALECONOSCO">
<state id="FALECONOSCO_SEND">
<state id="FORUM_LIST">
<state id="FORUM_VIEW">
<state id="ENQUETE_RESULT">
<state id="GALERIA_LIST">
<state id="GALERIA_VIEW">
<state id="Visualizar_Noticia">
```

Código-Fonte 2 – SCXML do portal da Prefeitura Municipal de Ponta Porã.

A simulação gráfica do Código-Fonte 2 é apresentado na Figura 48. Configurados as fatias, os contextos, os estilos, o povoamento e as interfaces, já é possível acessar e visualizar a apresentação da nova página (portal) da Prefeitura Municipal de Ponta Porã, como mostra a Figura 49.

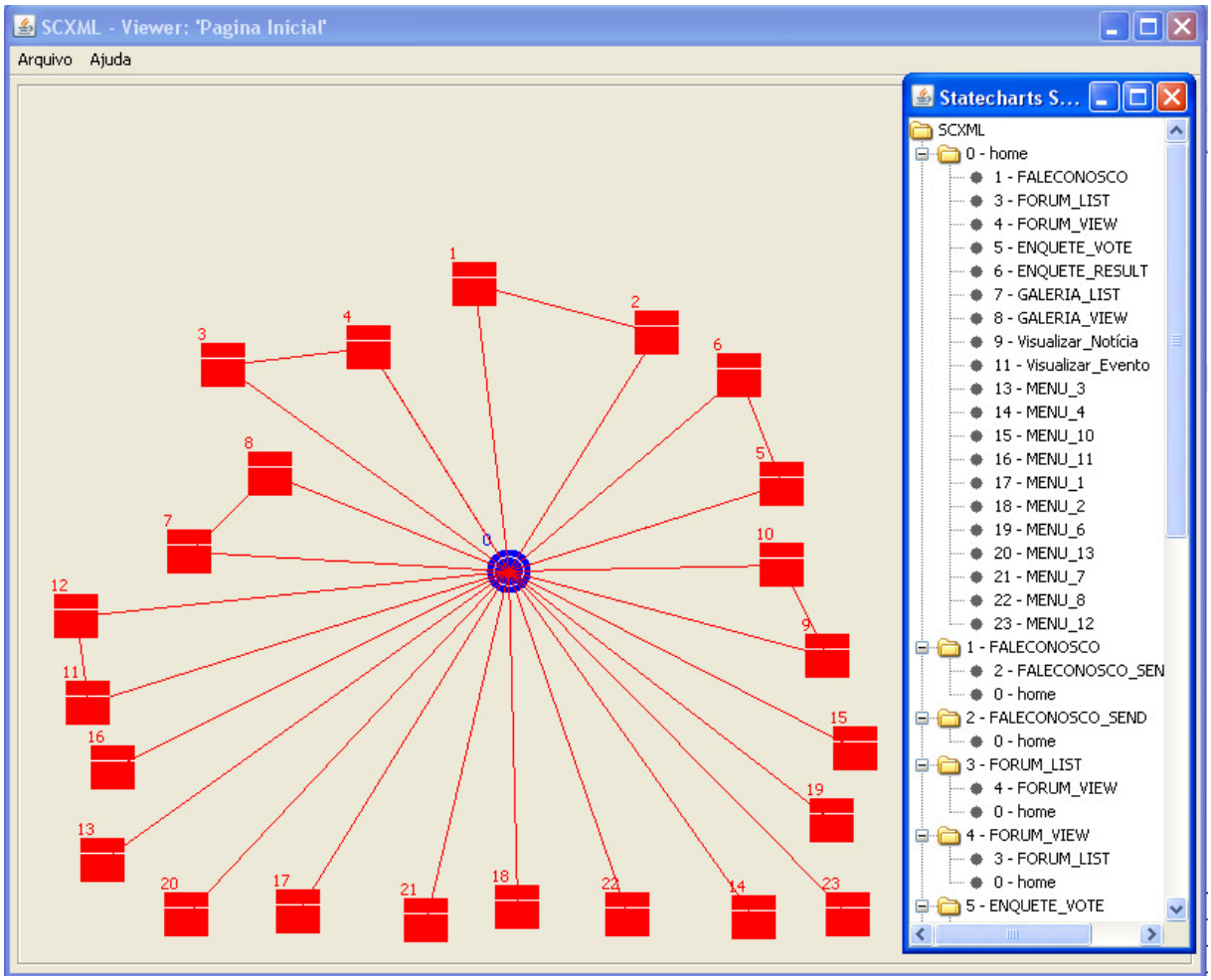


Figura 48 – Simulação gráfica do SCXML do Código-Fonte 2.

6.5 – CONSIDERAÇÕES FINAIS

Este capítulo apresentou a validação da proposta que foi realizada no Governo do Estado de Mato Grosso do Sul, sendo apresentado o processo (passo a passo) de instanciação de WebApps no framework Pantaneiro para a plataforma e-gov do Estado de Mato Grosso do Sul, a listagem do endereço das 54 WebApps já instanciadas com o uso do framework e o mapeamento de uma das WebApps mais completas do framework Pantaneiro em uso no Governo MS, que é o portal da SED. A partir das reuniões realizadas com a equipe de gestão do Pantaneiro no governo do Estado de Mato Grosso do Sul registrou-se uma satisfação no uso da ferramenta, pois agiliza o processo de desenvolvimento dos portais, padroniza estrutura e

conteúdo, além de ser simples, fácil de utilizar, estender e ser baseada em tecnologias de software livre, não tendo custos adicionais para o governo.

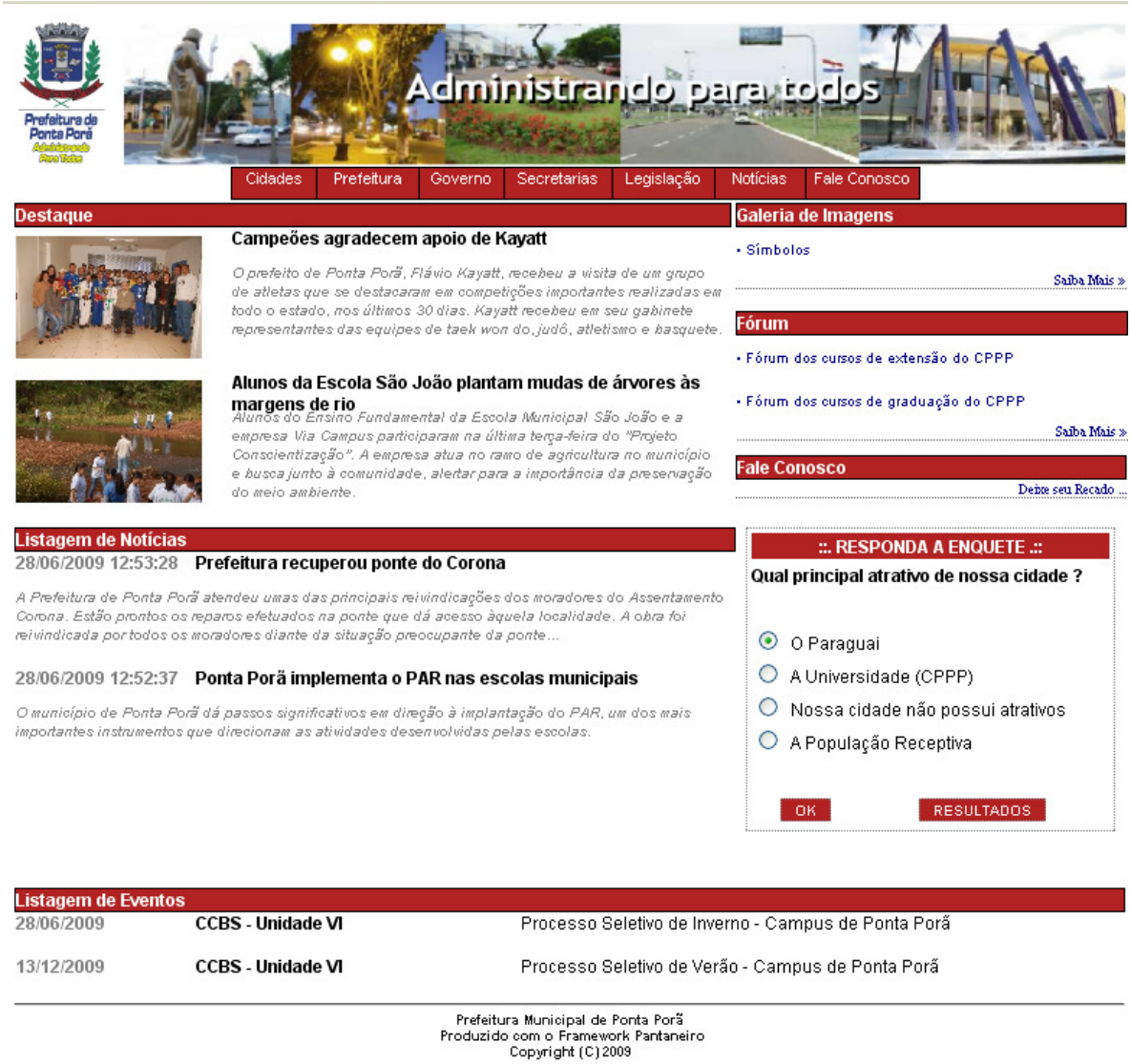


Figura 49 – Apresentação final da nova página da Prefeitura Municipal de Ponta Porã.

Conclusões

7.1 – CONSIDERAÇÕES INICIAIS

O desenvolvimento deste trabalho teve início com a especificação de um framework gerenciador de conteúdo e gerador de aplicações para o Governo do Estado de Mato Grosso do Sul, com o intuito de desburocratizar a disponibilização de WebApps entre os órgão de governo, distribuindo a gerência destas de forma colaborativa e compartilhada. Esse framework foi construído para ser leve e flexível o bastante para sofrer alterações e inclusões de novas funcionalidades que surgiriam de acordo com as demandas dos órgãos.

Com a implantação em 2006 do framework Pantaneiro no Governo do Estado de Mato Grosso do Sul, muitos requisitos que não haviam sido levantados foram surgindo e implementados de forma incremental e colaborativa. A primeira WebApp instanciada pelo framework Pantaneiro foi a WebApp da SGI (Superintendência de Gestão da Informação), órgão de TI do Governo de Mato Grosso do Sul, responsável por manter uma equipe técnica para o framework Pantaneiro. Depois desta iniciativa da SGI, a procura dos órgãos para ter sua própria WebApp foi aumentando em grande escala e hoje, o framework possui 54 WebApps instanciadas na plataforma e-gov MS. Neste período, o código-fonte passou por muitas modificações e otimizações para conseguir robustez e garantir a integridade das informações, além de sua documentação que passou a ser composta de manuais, guias do usuário, apostilas, apresentações, vídeo-tutoriais, tutoriais, entre outros documentos.

7.2 – CONTRIBUIÇÕES

Este trabalho, enquanto projeto de mestrado, se ateve a especificar, implementar e transformar o framework Pantaneiro em um software com forte base científica e com conhecimentos tecnológicos agregados. Artigos foram escritos e aceitos em congressos nacionais e internacionais, porém os avaliadores sempre aceitavam a publicação do artigo com ressalvas, pois faltava uma metodologia de desenvolvimento de WebApps subjacente ao framework. Isto impulsionou este trabalho, sendo que as principais contribuições e resultados alcançados foram:

a) modelagem do framework Pantaneiro para especificar, instanciar e gerenciar WebApps em uma plataforma e-gov institucional utilizando o formalismo Statecharts, subjacente ao método HMBS/M estendido.

b) projeto e implementação do Wizard-Pantaneiro, necessário para facilitar e agilizar o processo de instanciação dos portais e-gov usando o framework Pantaneiro. O Wizard-Pantaneiro é uma ferramenta de desenvolvimento (instanciação) de aplicações para o framework Pantaneiro. O Wizard-Pantaneiro é divido em três ambientes. O Ambiente de Autoria é um conjunto de módulos responsáveis pela instanciação de componentes, permissões e interfaces de apresentação da WebApp. O Ambiente de Projeto Navegacional é um conjunto de módulos responsáveis por definir a navegação entre os componentes da WebApp. O Ambiente de Publicação é um conjunto de módulos responsáveis pela inserção de novas WebApps, pelo processo de publicação colaborativa de informações e pela geração do modelo comportamental da WebApp publicada.

c) uso e adaptação do HMBS/M estendido para incorporar novos conceitos de e-gov para o framework Pantaneiro.

d) especificação e criação de um repositório de componentes e-gov no framework a fim de facilitar o reuso em diferentes plataformas: Notícias, Agenda, Eventos, Galeria de Imagens, Fórum, Enquete e Banners.

e) reengenharia do código-fonte da 1ª versão do Pantaneiro com a inserção da Camada de Acesso a Banco de Dados, transformando o framework Pantaneiro em um software independente ao SGBD-R escolhido. Ainda neste trabalho, o framework Pantaneiro homologou o uso dos SGBD-Rs Microsoft SQL Server (por necessidades do Governo do Estado de Mato Grosso do Sul) e PostgreSQL (por ser um SGBD-R muito robusto e com licenciamento de software-livre);

f) geração do modelo comportamental das WebApps instanciadas usando o formalismo de *Statecharts* (Diagrama de Estados) representada por um arquivo XML sobre o esquema SCXML (padrão internacional da W3C);

g) Visualização (gráfica) e simulação do comportamento das WebApps instanciadas por meio de uma aplicação JAVA desenvolvida neste trabalho, que recebe como entrada o modelo comportamental em SCXML exibindo na tela todos os estados e suas transições, além de simular a execução do diagrama de estados; e

h) avaliação, teste e validação do framework Pantaneiro em ambiente real de gestão do portal do governo do estado de Mato Grosso do Sul.

Além dos resultados e contribuições apresentadas, encontra-se em fase final um trabalho de mestrado (MAIA, 2008) que propõe um modelo de processo para a inserção das diretrizes de acessibilidade do governo eletrônico brasileiro nas WebApps instanciadas pelo framework Pantaneiro.

Este trabalho ainda serviu de base para dois Trabalhos de Conclusão de Curso (TCC) do curso de Bacharelado em Análise de Sistemas da Universidade Federal de Mato Grosso do Sul, sendo o primeiro intitulado “i-Faces: Geração de Interfaces Dinâmicas para o Pantaneiro” e o segundo “Reestruturaçao do portal PREAE/UFMS com o framework Pantaneiro”.

7.3 – LIMITAÇÕES DA PROPOSTA

O framework Pantaneiro que está implantado e com muito sucesso no governo do Estado de Mato Grosso do Sul ainda não está contemplado de todas as funcionalidades desenvolvidas neste trabalho. A partir de 2007, o framework

Pantaneiro passou a ter duas frentes de implementação, a primeira coordenada pela equipe de TI da SGI (baseada em manutenção, correção de *bugs* e novas funcionalidades segundo demandas dos órgãos de governo), enquanto que a segunda como resultado final deste trabalho. Assim, o framework Pantaneiro em produção na plataforma e-gov de MS possui algumas limitações, tais como:

- ausência do módulo Gerador de SCXML e consequentemente, a ausência da aplicação JAVA "SCXML Viewer – Pantaneiro";
- ausência da funcionalidade "gerência de fatias" do módulo Gerência de Componentes;
- ausência da funcionalidade "gerência de contextos" do módulo Gerência Navegacional;
- ausência de uma metodologia que dê suporte ao desenvolvimento das WebApps instanciadas;
- ausência da "Camada de Acesso ao Banco de Dados" que torna o framework Pantaneiro independente do SGBD-R escolhido. A implantação do framework Pantaneiro no Governo do Estado de Mato Grosso do Sul utiliza o SGBD-R Microsoft SQL Server;
- ausência do paradigma de orientação a objetos no framework Pantaneiro; e
- ausência de alguns artefatos de software, tais como diagrama de classe e diagrama de caso de uso.

No ínicio de implantação, a equipe de TI da SGI ainda encontrou inúmeras dificuldades com a utilização do framework Pantaneiro, dentre elas:

- o processo de depuração do código-fonte tornou-se complicado, pois muitas vezes era difícil distinguir quando o erro era do framework, da aplicação ou de alguma regra de negócio não conhecida pela equipe;
- a grande generalidade do framework (dentro do domínio e-gov) fez o framework perder em eficiência quando começou a ser exigido de muitas conexões simultâneas. Esse problema foi contornado com otimizações na implementação do código-fonte e na implementação de mecanismos de cache na estrutura de rede do Governo do Estado de Mato Grosso do Sul; e

- a solicitação de novos *templates* por parte dos órgãos governamentais demandou esforço da equipe de TI da SGI. Com isso, muitos templates foram criados, porém atualmente poucos são usados, devido a política de padronização de interface das WebApps do governo eletrônico de Mato Grosso do Sul;

Outra limitação do framewok Pantaneiro é a ausência de padrões e diretrizes de acessibilidade nas WebApps instanciadas. A acessibilidade das WebApps da administração pública na internet se tornou requisito obrigatório a partir do Decreto-Lei 5296 de 2 de dezembro de 2004.

7.3 – TRABALHOS FUTUROS

A partir dos resultados deste trabalho, dentre inúmeras possibilidades de trabalhos futuros, é possível vislumbrar as seguintes perspectivas:

1. Processamento dos metadados para visualização dos modelos de forma gráfica para representar o modelo de classes, o modelo de fatias, o modelo de contextos navegacionais, o modelo navegacional de tipos, o modelo de estilos e o modelo de composição das WebApps instanciadas pelo framework Pantaneiro;
2. Inserção de novos conceitos de *Statecharts* na simulação do SCXML Viewer, tais como as condições de guarda e comportamentos de entrada e saída dos estados do diagrama;
3. Adoção de todos os padrões do governo eletrônico brasileiro (e-Ping, e-Mag, entre outros) e padrões internacionais da W3C (CSS, XML, AJAX, entre outros) nas páginas geradas dinamicamento pelo framework;
4. Introdução aos conceitos da Web 2.0 tanto no desenvolvimento de novos módulos, quanto na reengenharia do código HTML gerado ao usuário final;
5. Publicações de artigos em anais de congressos nacionais e internacionais de Engenharia de Software, Engenharia Web, Governo Eletrônico e demais assuntos correlatos.

Referências Bibliográficas

ALEXANDER , C.; *The timeless way of building*. Oxford University Press, 1979.

ALKACON. *Interactive OpenCMS 5.0 documentation available from Alkacon Software.* Disponível em http://www.opencms.org. Acessado em 03 abr 2009.

ARAGON, C. R.; *Processo de Desenvolvimento de uma Linha de Produtos para Sistemas de Gestão de Bibliotecas*. Dissertação de Mestrado, Universidade Federal do Mato Grosso do Sul, Campo Grande, MS, Brasil, 2004.

ASES. Portal do Software Público Brasileiro. Disponível em http://www.softwarepublico.gov.br/ver-comunidade?community_id=8265263. Acessado em 22 jun 2009.

ANDRETO, D.E.; AMARAL, A.M.M.M.; *Engenharia de Software para Web* - Iniciação Científica - CESUMAR - Jul/Dez de 2006. v.08, n.02, p.157-166.

APPLETON, B.; *Patterns and Software: Essential Concepts and Terminology.* Hillside.Net, 2000.

ARGOUML. *UML design tool with cognitive support*. Disponível em: <http://argouml.tigris.org>. Acessado em: 03 abr. 2009.

ATTERER, R.; *Where Web Engineering Tool Support Ends: Building Usable Websites In Proceedings of the 20th Annual ACM Symposium on Applied Computing* (SAC05), ISBN 1-58113-964-0, Santa Fe, New Mexico, USA, pages 1684-1688, March 2005.

ATZENI P.; PARENTE, A.; *Specification of Web applications with ADM-2*. Dipartimento di Informatica e Automazione, Universit`a di Roma, Roma, 2001.

BALASUBRAMANIAN, V.; BASHIAN, A.; PORCHER, D. *A large-scale hypermedia application using document management and Web technologies*. In: Proceedings of Hypertext'97, VIII International ACM Hypertext Conference, Southampton, UK, April 6-11 1997, p.134-45.

BALZERANI, L.; RUSCIO, D. D.; PIERANTONIO, A.; DE ANGELIS, G.; *A product line architecture for web applications*. *In Proceedings of the 2005 ACM Symposium on Applied Computing* (Santa Fe, New Mexico, March 13 - 17, 2005). L. M. Liebrock, Ed. SAC '05. ACM, New York, NY, 1689-1693. DOI= http://doi.acm.org/10.1145/1066677.1067059 . 2005.

BELANGER, F.; HILLER, J. S.; *A framework for e-government: privacy implications*. Business Process Management Journal, v. 12, n. 1, p. 48-60, 2006.

BENGTSSON, P.; BOSCH, J.; MOLIN, P.; MATTSSON, M.; FAYAD, M.; *Framework problem and experiences in M. Fayad, R. Johnson, D. Schmidt. Building Application Frameworks: Object-Oriented Foundations of Framework Design, John Willey and Sons*, p. 55–82, 1999.

BIEBER, M.; GALNARES, R.; LU, Q. *Web Engineering and flexible hypermedia*. Proceedings of the 2nd Workshop in Adaptive Hypertext and Hypermedia, Hypertext 1998.

BIGGERSTAFF , T. J.; P ERLIS , A. J.; *Software reusability*. ACM Press, 1989.

BOSCH, J.; *Product-line architectures in industry: a case study. Proceedings of the 21st international conference on Software engineering*. Los Angeles, California, USA: IEEE Computer Society Press, 1999.

BOSCH, J.; *Software product lines: organizational alternatives. Proceedings of the 23rd international conference on Software engineering*. Toronto, Ontario, Canada: IEEE Computer Society, 2001.

BRAGA, R. T. V.; *Um Processo para Construção e Instanciação de Frameworks baseados em uma Linguagem de Padrões para um Domínio Específico*. Tese de Doutoramento, Instituto de Ciências Matemáticas e de Computação da USP-SC, São Carlos, São Paulo, Brasil, 2003.

BRAGA , R. T. V.; MASIERO , P. C.; *Identification of framework hot spots using pattern languages*. *In 15th Brazilian Symposium on Software Engineering*, p. 145—160, 2001.

BRAGA , R. T. V.; GERMANO, F. S. R.; MASIERO, P. C.; *A pattern language for business resource management. Proceedings of the 6th Annual Conference on Pattern Languages of Programs* (PLoP'99). Monticello, Illinois, USA, 1999.

BRITO, L.S.F.; *WEBSCHARTS: Uma ferramenta de desenvolvimento de aplicações Web baseada no HMBS/M.* Dissertação de Mestrado em Ciência da Computação – Universidade Federal de Mato Grosso do Sul. 2003.

BRUGALI, D.; SYCARA, K.; *Frameworks and pattern languages: an intriguing relationship*. ACM Computing Surveys, ACM Press, v. 32, n. 1, p. 2-7 , 1999.

BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAND, P.; STAL,M.; *Pattern-oriented software architecture - a system of patterns*. Wiley & Sons, 1996.

CALVO, R.A.; GHIGLIONE, E.;ELLIS, R.. "The OpenACS e-learning infrastructure". The University of Sydney, Bldg J03, Sydney, NSW 2006.

CAKEPHP. The rapid development php framework. Disponível em http://cakephp.org. Acessado em 29 mai 2009.

CARVALHO, M. R. C.; *HMBS/M - Um Método orientado a objetos para o projeto e o desenvolvimento de aplicações hipermídia.* Dissertação de Mestrado - Instituto de Ciências Matemáticas e de Computação (ICMC-USP). São Carlos-SP, 1998.

CARVALHO, M. R. C.; OLIVEIRA, M.C.F.; MASIERO, P. C.; *HMBS/M - an object oriented method for hypermedia design*. *Proc. V Brazilian Symposium on Multimedia and Hypermedia Systems* - SBMIDIA'99, Jun. 15-18, Instituto de Informática, UFG, Goiânia, p. 43-62, 1999.

CAVALLARO, U.; GARZOTTO, F.; PAOLINI, P.; TOTARO, D.; *HIFI: hypertext interface for information systems*. IEEE Software, v. 10, n. 6, p. 48-51, nov. 1993.

CERI, S.; FRATERNALI, P.; BONGIO, A.; *Web Modeling Language (WebML): a modeling language for designing Web sites*. *Proc WWW9 Conference, Amsterdam*, NL, (also in Computer Networks, 33 (2000), p. 137-157, 2000.

CHEN, PETER P. *The Entity Relationship Model - Toward A Unified View of Data*. ACM Transactions on Database Systems, v. 1, n. 1, p. 9-36 1976.

CHASTEK, G.; *Object technology and product lines*. OOPSLA 97: ACM SIG-PLAN conference on Object-oriented programming, systems, languages, and applications (Addendum). Atlanta, Georgia, USA: ACM Press, 1997.

CHEESMAN, J.; DANIELS, J.; *UML Components - A Simple Process for Specifying Component-based* . Addison-Wesley, 2001.

CLEAVELAND , J. C.; *Program generators with XML and JAVA*. Prentice Hall, 2001.

CLEMENTS, P. C.; NORTHROP, L.; *Software Product Lines: Practice and Patterns*. Addison-Wesley, 2001.

COLEMAN, D.; ARNOLD, P.; BODOFF, S.; DOLLIN, C.; GILCHIRST, H.; HAYES, F.; JEREMAES, P.; *Object-oriented development: The Fusion Method*, New Jersey, USA, Prentice Hall, Englewood Cliffs, p. 313, 1994.

COMMONS APACHE. Apache Commons: Home. Disponível em http://commons.apache.org. Acessado em 29 mai 2009.

CONALLEN, J.; *Desenvolvimento de aplicações Web com UML*. Rio de Janeiro, RJ: Campus. 2003.

CONALLEN, J.; *Building Web applications with UML*. Addison Wesley Object Technology Series, 2000.

COX, H.; *Users' guide to Whopper Cropper*. 2006. Disponível em: http://www.apsru.gov.au/. Acessado em: 03 abr 2009.

CRNKOVIC, I.; HNICH, B.; JONSSON,T.; KIZILTAN, Z.; *Specification, implementation, and deployment of components. Communications of the ACM*, ACM Press, New York, NY, USA, v. 45, n. 10, p. 35-40 , 2002.

CRUZ, T.; *Gerência do Conhecimento*. Ed. Cobra, São Paulo, 2002.

CZARNECKI , K.; E ISENERCKER , U. W.; *Generative programming*. Addison-Wesley, 2002.

D'SOUZA , D. F.; WILLS , A. C.; *Objects, components, and frameworks with UML - the catalysis approach*. Addison-Wesley, 339–381 p., 1999.

DEMOISELLE. SourceForge: Demoiselle Framework. Disponível em http://sourceforge.net/projects/demoiselle. Acessado em 29 mai 2009.

DE PAEZ, R. A.; *Un acercamiento a la reutilización en ingeniería de software*. EAFIT -Magazine, Medellín, Colombia, n. 114, p. 45-63 , 1999.

DETLOR, B.; *The corporate portal as information infrastructure: towards a framework for portal design. International Journal of Information Management*, v.20, p. 91-101, 2000.

DE TROYER, O.; LEUNE, C.; *WSDM: A User-Centered Design Method for Web Sites, in Computer Networks and ISDN systems, Proc. 7th International WWW Conference, Elsevier*, pp. 85-94. 1998.

DRUPAL. Community plumbing. Disponível em http://www.drupal.org. Acessado em 29 mai 2009.

e-PING. *Padrões de Interoperabilidade de Governo Eletrônico* – Governo Eletrônico Brasileiro. Documento de Referência da e-PING – Versão 4.0. - 2008.

DIAZ, A.; ISAKOWITZ, T.; MAIORANA, V.; GILABERT, G.; *RMC: A Tool To Design WWW Applications*, *Proceedings of the Fourth International World Wide Web Conference*, Boston, 1995.

DIÁZ, P.; MONTERO, S.; AEDO, I.; *Modelling hypermedia "GUIDELINES FOR EMPIRICAL WORK" in the Workshop and web applications: the Ariadne Development Method.* Information System. Article in Press. 2004.

ELMASRI R.; NAVATHE S. B. *Fundamentals of Database Systems*. 3. ed. Benjamin/Cummings, 2000.

FAYAD, M.E. *et al.*; *Building Application Frameworks*. John Wiley & Sons, Inc. New York, 1999.

FAYAD, M.E.; SCHMIDT, D. C.; *Object-oriented application frameworks. Communications of the ACM*, ACM Press, New York, NY, USA, v. 40, n. 10, p. 32-38, 1997.

FAYAD, M.E.; JOHNSON, R. E.; SCHMIDT, D. C.; *Software patterns. Communications of the ACM*, ACM Press, v. 39, n. 10, p. 37-39 , 1996.

FEDERATIVO; *Portal Federativo.* Disponível em http://www.portalfederativo.gov.br. Acessado em 03 abr de 2009.

FOOTE, B.; JOHNSON, R. E.; *Designing reusable classes*. *Journal of Object Oriented Programming*, v. 1, n. 2, p. 22-35 , 1988.

FRANCA, L. P. A.; STAA, A. V.; *Geradores de artefatos: Implementação e instanciação de frameworks*. Anais do XV SBES-2001 - Simpósio Brasileiro de Engenharia de Software. Rio de Janeiro, Brasil , 2001.

FRAKES , W.; ISODA , S.; Success factors of systematic reuse. IEEE Software, v. 11, n. 9, p. 14–19, 1994.

FRAKES , W.; TERRY, C.; *Software reuse and reusability metrics and models*. Relatório Técnico TR-95-07. 1995. Disponível em http://citeseer.ist.psu.edu/article/frakes96software.html. Acessado em 03 abr 2009.

FRATERNALI, P. *Tools and approaches for developing data-intensive Web applications*. ACM Computing Surveys, v. 31, n. 3, set. 1999.

FRATERNALI, P.; PAOLINI, P.; *A Conceptual Model and a Tool Environment for Developing More Scalable and Dynamic Web Applications*. *In H.-J. Schek, F. Saltor, I. Ramos, and G. Alonso Eds., Proc. Int. Conf. on Extending Database Technology*, EDBT98 (Valência, Espanha, Mar. 1998), p. 421-435, 1998.

FREY, K.; *Governança Eletrônica: experiências de cidades européias e algumas lições para países em desenvolvimento*. Belo Horizonte: Revista IP – Informática Pública, v.02, p.31-48. Maio/2000.

FRYE, J.; YODER, J.; *The hillside group*, 2001. Disponível em: http://hillside.net. Acessado em 03 abr 2009.

GAFFNEY Jr., J.E.; CRUICKSHANK, R. D.; *A General Economics Model of Software Reuse*. ICSE 1992: 327-337

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J.; *Design patterns - abstraction and reuse of object-oriented design. Lecture Notes in Computer Sciente,Springer-Verlag*, Berlin, Alemanha, n. 707, p. 406-431 , 1993.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J.; *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

GARZOTTO, F.; MAINETTI, L.; PAOLINI, P.; *Adding Multimedia Collections do the Dexter Model. Proceedings ECHT"94*, Edinburgh, set. 1994.

GARZOTTO, F.; MAINETTI, L.; PAOLINI, P.; *Hypermedia Design, Analysis and Evaluation Issues*, CACM: Communications of the ACM, v. 38, n. 8, p. 74-86, ago. 1995.

GARZOTTO, F.; PAOLINI, P.; SCHWABE, D.; *HDM-A Model-Based Approach to Hypertext Application Design*. ACM Transaction on Information Systems, v. 11, n. 1, p. 1-26, jan. 1993.

GINIGE, A.; MURUGESAN, S.; *Web Engineering: An Introduction*. IEEE Multimedia, v. 8, n. 1, p. 14-18, jan./mar. 2001.

GIMENES, I. M. S.; TRAVASSOS, G. H.; *O enfoque de linha de produto para desenvolvimento de software*. EVENTO INTEGRANTE DO XXII CONGRESSO DA SBC - SBC2002. XXI Jornada de Atualização em Informática. Sociedade Brasilera de Computação, 2002.

GÓMEZ; J.; CACHERO C.; PASTOR O.; *Conceptual Modeling of Device-Independent Web Applications*. IEEE Multimedia. v. 8, n. 2. p. 26-39. abr./jun. 2001.

GÓMEZ; J.; BIA, A.; PÁRRAGA, A.; *Tool Support for Model-Driven Development of Web Applications*. WISE 2005: 721-730.

GRAEF, G.; GAEDKE, G.; *Development and Evolution of Web-Applications using the WebComposition Process Model*. *International WorkShop on Web Engineering at 9th International, World Wide Web Conference (WWW9)*, Amsterdam, The Netherlands, 15 mai. 2000.

GRANT, G.; CHAU, D.; *Developing a generic framework for e-government. Journal of Global Information Management*, v. 13, n. 1, p. 1-30, 2005.

GREENFIELD, J.; SHORT, K.; *Software Factories - Assembling Applications with Patterns, Models, Frameworks, and Tools. Wiley Publishing, Inc.* 2004.

GLOVER, E.; TSIOUTSIOULIKLIS, K.; LAWRENCE, S.; PENNOCK, D.; FLAKE G. *Using Web structure for classifying and describing Web pages*. Proceedings of WWW2002, International Conference on the World Wide Web. 2002.

GOVERNO ELETRÔNICO. Princípios e Diretrizes – Governo Eletrônico. Disponível em http://www.governoeletronico.gov.br/o-gov.br/principios. Acessado em 29 mai 2009.

HAREL, D.; *Statecharts: a visual formalism for complex systems*. Science of Computer Programming, v. 8 (3), p. 231-274, jul. 1987.

HENNRICHS, J.C.; MAZZOLA, V.B.; *Avaliação da metodologia orientada a objetos OOHDM, para a modelagem e desenvolvimento de websites*. Monografia para conclusão de curso Especialização em Pós-Graduação em Ciência da Computação pela Universidade Federal de Santa Catarina. 2005.

HOLMES, D.; *EGov: eBusiness strategies for government*. Londres: Nicholas Brealey Publishing, 2001. Reimpressão.

I3GEO. Mapa Interativo do Ministério do Meio Ambiente. Disponível em http://mapas.mma.gov.br/i3geo/. Acessado em 22 jun 2009.

IBGE. *Instituto Brasileiro de Geografia e Estatística.* Disponível em: http://www.ibge.gov.br. Acessado em 03 abr 2009.

IBOPE/NET Ratings. Ibope, especialista em pesquisas de mercado, mídia e opinião. Disponível em http://www.ibope.com.br. Acessado em 29 mai 2009.

I-EDUCAR. Centro Tecnológico de Informação e Modernização Administrativa. Disponível em http://ctima.itajai.sc.gov.br/sistema_det.php?cod_sistema=13. Acessado em 22 jun 2009.

IERUSALIMSCHY R.; FIGUEIREDO L. H. de; CELES W.; *Lua – an extensible extension language. Software: Practice & Experience 26* #6 (1996) p. 635-652. Disponível em: <http://www.lua.org>. Acessado em: 03 abr 2009.

INOVAÇÃO; *Portal Inovação.* Disponível em http://www.portalinovacao.mct.gov.br/pi. Acessado em 03 abr de 2009.

ISAKOWITZ, T; KAMIS, A.; KOUFARIS, M.; *Extending the capabilities of RMM: Russian Dolls and hypertext*. *In: Proceedings of the 30th Annual Hawaii International Conference on System Sciences*, HICSS'97, 1997.

ISAKOWITZ, T.; STOHR, E.; BALASUBRAMANIAM, P.; *RMM: A methodology for structured hypermedia design*. CACM: Communications of the ACM, v. 8, p. 34-44, out. 1995.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J.; *The Unified Software Development Proccess. Reading*, MA: Addison-Wesley, 1999.

JOHNSON , R. E.; *Documenting frameworks using patterns*. OOPSLA, 1992.

JOOMLA. Joomla! Disponível em http://www.joomla.org. Acessado em 29 mai 2009.

KICZALES, G.; LAMPING, J.; MEDDHEKAR, A.; MAEDA, A.; LOPES, C. V.; LOINGTIER, J. M.; IRWIN, J.; *Aspect-oriented programming. In proceedings of the European Conference on Object-Oriented Programming* (ECOOP), Finland. Springer-Verlag LNCS , 1997.

KLISCHEWSKI, R.; JEENICKE, M.; *Semantic Web Technologies for Information Management within e-Government Services.* HICSS 2004.

KNAPP, A.; KOCH, N.; MOSER F.; ZHANG, G.; *ArgoUWE: A CASE Tool for Web Applications. First Int. Workshop on Engineering Methods to Support Information Systems Evolution* (EMSISE´03), set. 2003.

KOCH, N.; *A Comparative Study of Methods for Hypermedia Development. Ludwing-Maximilians-Universität München*, nov. 1999, Relatório Técnico 9905.

KOCH, N.; *Software Engineering for Adaptive Hypermedia Applications*, PhD. Thesis, Reihe Softwaretechnik 12, Uni-Druck Publishing Company, Munique, Alemanha, 2001.

KOH, C.E.; RYAN, S.; PRYBUTOK, V.R.; *Creating value through managing knowledge in an e-government to constituency (G2C) environment.* The Journal of Computer Information Systems, v.45, n.4, p.32-41, 2005.

KRUCHTEN, P.; *The Rational Unified Process An Introduction*, Second Edition, Addison Wesley, 2000.

LAI, C. T. R.; WEISS, D. M.; *Software Product-Line Engineering: A Family-Based Software Development Process*. Addison-Wesley, 1999.

LANGE, D.; *An Object-Oriented design method for hypermedia information systems*. *Proceedings of the 27th. Annual Hawaii International Conference on System Science*, jan. 1994.

LATTES; *Plataforma Lattes.* Disponível em http://lattes.cnpq.br. Acessado em 03 abr de 2009.

LDAP. Projeto LDAP Brasil. Disponível em http://www.ldap.org.br. Acessado em 29 mai 2009.

LEUNG, K.; HUI, L.; YIU S.; TANG, R.; *Modelling Web Navigation by Statechart, Department of Computer Science and Information Systems*, The University of Hong Kong, Hong Kong, 2000.

LIGHTBASE. Portal do Software Público Brasileiro. Disponível em http://www.softwarepublico.gov.br/ver-comunidade?community_id=3673574. Acessado em 22 jun 2009.

LIM , W.; *Effects of reuse on quality productivity and economics. Guest Editors Introduction*, IEEE Expert, p. 23–30, 1992.

LIMA, F.; SCHWABE, D.; *Application Modeling for the Semantic Web*, LA-WEB 2003 - First Latin American Web Conference, Santiago, Chile, IEEE-CS Press, 2003.

LOWE, D.; GINIGE A.; *Matilda: A framework for the representation and processing of information in multimedia systems*. *In 3rd International Interactive Multimedia Symposium*, Perth, Aust., jan. 21-25, 1996.

MAIA, L.; Acessibilidade no Processo de Desenvolvimento de Aplicações Web. Qualificação de Mestrado em Ciência da Computação. DCT-UFMS. 2008.

MCMS. Microsoft Content Management Server: Home. Disponível em http://www.microsoft.com/CMServer. Acessado em 29 mai 2009.

MELIA, S.; GÓMEZ, J.; *Applying WebSA to a case study: A travel agency system. Proceedings in Wokshop on Model-driven Web Engineering* (MDWE) 2005.

MENDES, E.; MOSLEY, N; *The Need for Web Engineering: an Introduction*, Web Engineering - Theory and Practice of Metrics and Measurement for Web Development, Springer-Verlag (accepted for publication). 2005.

MENDES, E.; MOSLEY, N; COUNSELL, S.; *Investigating Web size metrics for early Web cost estimation*, The Journal of System and Software, In Press, Corrected Proof, 2004.

METABASE. *Metabase (metabase, com, database form, blob) – PHP Classes.* Disponível em http://www.phpclasses.org/browse/package/20.html. Acessado em 03 abr 2009.

MINETTO, E.L.; Frameworks para desenvolvimento em PHP. Livro. Editora Novatec. ISBN: 9788575221242. 1ª Edição. 2007.

MORO, M. M.; *Workflow.* Disponível em http://www.inf.ufrgs.br/~mirella/workflow/work.html Acessado em 03 abr 2009.

MSGOV; *Portal MS-GOV.* Disponível em http://www.ms.gov.br. Acessado em 03 abr de 2009.

MUNIC, 2006 - Perfil dos Municípios Brasileiros em 2006. Disponível em : http://www.ibge.gov.br/home/estatistica/economia/perfilmunic/2006/default.shtm. Acessado em 03 abr 2009.

MURUGESAN, S.; DESHPANDE, Y.; HANSEN, S.; GINIGE A.; *Web Engineering: A New Discipline for Development of Web-based Systems*, Proc. First ICSE Workshop on Web Engineering, Los Angeles, CA, mai. 16-17, 1999.

NOGUEIRA, G.; CAPRA, S.; *Adaptação do RUP para Projetos de Software e-Commerce*. Trabalho de Conclusão de Curso, Centro de Ciências Exatas e da Natureza – Universidade da Amazônia, 2003.

OLIVEIRA, M.C.F.; TURINE, M.S.; MASIERO, P.C.; *A Statechart-Based Model for Modeling Hipermidia Applications*. ACM Transactions on Information Systems. v. 19, n. 1, p. 28-52, jan. 2001.

OPENACS. OpenACS Home. The Toolkit for Online Communities. Disponível em http://www.openacs.org/. Acessado em 22 jun 2009.

PACHECO, R.C.S.; KERN, V.M.; STEIL, A.V.; *Aplicações de arquitetura conceitual em plataformas e-GOV: da gestão da informação pública à construção da sociedade do conhecimento.* Ponto de Acesso, Salvador, v.1, n.1, p. 71-87, jun. 2007.

PACHECO, R.C.S.; KERN, V.M.; *Arquitetura conceitual e resultados da integração de sistemas de informação e gestão da ciência e tecnologia*. Datagramazero, v. 4, n. 2, 2003. Disponível em: <http://www.dgz.org.br/abr03/Art_03.htm>. Acesso em: 03 abr 2009.

PASTOR O.; INSFRÁN E.; PELECHANO V.; ROMERO J.; MERSEGUER J.; *OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods*. In CAiSE '97. *International Conference on Advanced Information Systems*, p. 145-158, 1997.

PASTOR O.; *Fitting the Pieces of the Web Engineering Puzzle*, Invited Talk, XVIII Simpósio Brasileiro de Engenharia de Software (SBES 2004), Brasília. 2004.

PAULO, F.; MASIERO, P; OLIVEIRA, M.; *Hypercharts, Extended Statecharts to Support Hypermedia Specifications*. IEEE Transactions on Software Engineering, USA, v. 25, n. 1, p 33-49, 1999.

PINHO, J.A.G.; *Investigando portais de governo eletrônico de estados no Brasil: muita tecnologia, pouca democracia.* Revista de Administração Pública – RAP. ISSN

0034 – 7612. FGV – Rio de Janeiro – RJ. 42 (3):471-93, MAI/JUN 2008.

PITHON, A.J.C.; BROCHADO, M.R.. "A Plataforma e-Proinfo como Ferramenta de Apoio a Aprendizagem Colaborativa". XXVI ENEGEP - Fortaleza, CE. 2006.

PONTES, R.; *Ambiente para Desenvolvimento de Aplicações WEB*. Rio de Janeiro-RJ, 1997, Dissertação de Mestrado – Departamento de Informática (DI-PUC-Rio).

PREE, W.; *Hot-spot-driven development. FAYAD, M.; JOHNSON, R.; SCHMIDT,D. (Ed.). Building Application Frameworks: Object-Oriented Foundations of Framework Design*. Wiley & Sons, 1999.

PRESSMAN, R.S. Software engineering - a practitioner's approach, v.5. McGraw-Hill, 2001

PRIETO-DIAZ, R. Domain analysis: An introduction. ACM SIGSOFT - Software Engineering Notes, v.15, p. 47-54, 1900

RECEITA; *Portal da Receita Federal.* Disponível em http://www.receita.fazenda.gov.br. Acessado em 03 abr de 2009.

REDEGOVERNO; *Portal da Rede Governo.* Disponível em http://www.redegoverno.gov.br. Acessado em 03 abr de 2009.

ROSSI, G.; SCHWABE, D.; GARRIDO, A.; *Design Reuse in Hypermedia Applications Development* in Proceedings of ACM Hypertext'97, Southampton, UK, 57-66, September 1997.

RUBY. Linguagem de Programação Ruby. Disponível em http://www.ruby-lang.org. Acessado em 29 mai 2009.

SANDIM, H. C. ; TURINE, M. A. S. ; VIEIRA, C. C. A.; *Pantaneiro: Um gerador de aplicações web no contexto e-gov. In: XXVI Congresso da Sociedade Brasileira de Computação.* Campo Grande. XXXIII Seminário Integrado de Software e Hardware, 2006.

SCHWABE, D.; ROSSI, G. V.; *An Object Oriented Approach To Web-Based Application Design. Theory and Practice of Object Systems*, New York, v. 4, n. 4, p. 207-225, 1998.

SCHWABE, D.; PONTES, R.; MOURA, I.; *OOHDM-Web: An Environment for Implementation of Hypermedia Applications in the WWW, ACM SigWEB Newsletter*, v. 8, n. 2, jun. 1999.

SCXML; *The Jakarta Project Commons SCXML*. Disponível em http://jakarta.apache.org/commons/scxml. Acessado em 03 abr 2009.

SHIBUYA, R.; LEIVA, W; OLIVEIRA, M; MASIERO, P.; *Automatic HTML generation from formal hypermedia specifications. In: Annual Hawaii International Conference on System Sciences*, 34., 2001, Maui, Hawaii. Abstracts and CD-ROM of full papers. Los Alamitos, California : IEEE Computer Society, 2001.

SHIMABUKURO Jr., E. K.; *Um gerador de aplicações configurável*. Dissertação de mestrado em Ciência da Computação pelo Instituto de Ciências Matemáticas e de Computação – ICMC/USP. São Carlos, SP. 2006.

SIAU, K.; LONG, Y.; Using social development lenses to understand e-government development. Journal of Global Information Management, v. 14, n. 1, p. 47-62, 2006.

SILVA, B.C.C.; FREITAS, S.A.A.; PIETROBON, C.A.M.; FALBO,R.A.; Processos e ferramentas para o desenvolvimento de software livre: um estudo de caso. Biblioteca Digital de Teses e Dissertações da UFES. 2006.

SINAES; *Portal SINAES.* Disponível em http://www.sinaes.gov.br. Acessado em 03 abr de 2009.

SMARAGDAKIS, Y.; BATORY, D.; *Application generators. Encyclopedia of Eletrical and Eletronics Engineering*, J. G. Webster (ed.), John Wiley and Sons, 2000.

SOMMERVILLE, I.; *Software Engineering*. Addison-Wesley, 2001.

SOUZA, T. B.; CATARINO, M. E.; SANTOS, P. C.; *Metadados: catalogando dados na internet*. Transinformação, v.9, n.2, 1997. Disponível em http://www.biblioestudantes.hpg.ig.com.br/146.htm. Acessado em: 03 abr 2009.

SOUZA, R.A.C.; VASCONCELOS, A.M.L.; *Uma extensão do fluxo de análise e projeto do RUP para o desenvolvimento de aplicações Web*. SBES2003, p.256-269. 2003.

TALENTOS. BANCO DE TALENTOS - Portal do Software Público Brasileiro. Disponível em http://www.softwarepublico.gov.br/ver-comunidade?community_id=10157501. Acessado em 22 jun 2009.

TALIGENT. Building object-oriented framework. Taligent Inc. Write paper, 1994, Disponível em : <http://www.taligent.com>. Acesso em : 22 jun 2009.

TANGARIFE, T., MONT'ALVÃO, C. R.; *Estudo Comparativo Utilizando uma Ferramenta de Avaliação de Acessibilidade para Web. In: 3o. Congresso Internacional de Pesquisa em Design*, Rio de Janeiro: ANPEDesign, 2005.

TARAPANOFF, K.; *Inteligência organizacional e competitiva.* Brasília: Editora da Unb, 2001. p. 344.

TARR, P.; OSSHER, H.; *Multi-dimensional separation of concerns and the hyperspace approach*. *Proceedings Architectures and Component Technology: The State-of-the-Art in Software Development*, 2000.

TERESA. *Transformation Environment for InteRactivE Systems RepresentAtions.* Disponível em: http://giove.cnuce.cnr.it/teresa.html. Acesso em: 22 jun 2009.

TERRA, J.C.; GORDON, C.; *Portais Corporativos: a Revolução na Gestão do Conhecimento*, São Paulo: Negócio Editora. 2002.

TURINE, M.A.S.; VIEIRA, C.C.A.; SANDIM, H.C.; *Uma Arquitetura Completa para WebApps Corporativas. In: Conferência IADIS Ibero Americana WWW/internet.* Murcia, Espanha. 2006.

TURINE, M. A. S. ; VIEIRA, C. C. A. ; ARAKAKI, A. A. ; SANDIM, H. C.; *Pantaneiro: Um Gerador de Aplicações Web Baseadas em Componentes. XI Simpósio Brasileiro de Multimídia e Web - WebMedia*, Poços de Caldas, Brasil, p. 210-213, 2005.

TURINE, M. A. S.; *HMBS - Um modelo Baseado em Statecharts para a Especificação Formal de Hiperdocumentos*. São Carlos-SP, 1998. Tese de Doutorado, Instituto de Física de São Carlos (IFSC).

TURINE, M.A.S.; OLIVEIRA, M.C.F.; MASIERO, P. C.; *HySCharts: A Statechart-Based Environment for Hyperdocument Autoring and Browsing. Multimedia Tools and Applications*, v. 8, p. 309-324, 1999.

UsiXML. Home of the User Interface eXtensible Markup Language. Disponível em http://www.usixml.org. Acessado em 29 mai 2009.

UOL. Universo Online. Disponível em http://www.uol.com.br. Acessado em 29 mai 2009.

VALERIO, A.; SUCCI, G.; FENAROLI, M. Domain analysis and framework-based software development. SIGAPP Appl. Comput. Rev., v.5, n.2, p. 4-15, 1997.

VILELLA, R. M.; *Conteúdo, usabilidade e funcionalidade: três dimensões para a avaliação de portais estaduais de governo eletrônico na web*. Tese de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil, 2003.

WEBRATIO; *WEBRATIO Tool Suíte*. Disponível em: http://www.webratio.com. Acessado em: 03 abr 2009.

XML; *Extensible Markup Language* (XML) 1.1 (Proposed Recommendation), W3C, eds. Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, John Cowan, François Yergeau. 5 nov. 2003. Disponível em: http://www.w3.org/TR/2003/PR-xml11-20031105/. Acessado em: 03 abr 2009.

XSLT; *XSL Transformations* Versão 1.0. W3C Recommendation. 16 nov. 1999. Disponível em: http://www.w3.org/TR/xslt. Acessado em: 03 abr 2009.

ZOPE. *Zope framework.* Disponível em http://www.zope.org. Acessado em 03 abr 2009.

ANEXO I

Modelo de Ofício para Solicitação de WebApp no framework Pantaneiro

Ofício nº

Campo Grande, _____ de _____ de 2007.

Senhor Secretário,

Solicito a criação de sítio oficial para a Secretaria / Órgão _____ _____.

Informamos a seguir os dados dos usuários para a ferramenta "Pantaneiro".

Responsável técnico pelo sítio:

Nome: _____

Setor: _____Telefone:_____

Prontuário: _____ CPF: _____RG: _____

Cargo: _____

E-mail: _____

A sugestão para endereço na web é www._____.ms.gov.br

Atenciosamente,

_____
Secretário
Secretaria de Estado .....

Ilmo Senhor
Guilherme Villalba Zurutuza Filho
Subsecretário de Comunicação
Secretaria de Estado de Governo
Nesta

ANEXO II

Modelo de Formulário de Projeto de WebApp no framework Pantaneiro

FRAMEWORK PANTANEIRO - PROJETO PARA CONSTRUÇÃO DE SITE

Órgão solicitante:

Equipe de trabalho	
• Responsável pelo Projeto	1. Nome completo: Cargo/função: Telefone: E-mail Notes: Outro e-mail:
• Técnicos (webmasters)	1. Nome completo: Cargo/função: Telefone: E-mail Notes: Outro e-mail:

Informações sobre o site	
• Informar subdomínio a ser utilizado	
• Tipo do site	[ ] internet [ ] Intranet
• Se tipo Intranet informar um apelido/nome curto para chamada do site	
• Poderá conter informações restritas para determinados usuários?	[ ] Sim [ ] Não
• Haverá área interativa de Fale Conosco?	[ ] Sim [ ] Não

- Se houver “Fale Conosco” informar destinatário por assunto:

Assunto:	Destinatário:
	e-mail Notes:
	Outro e-mail:

Assunto:	Destinatário:
	e-mail Notes:
	Outro e-mail:

Assunto:	Destinatário:
	e-mail Notes:
	Outro e-mail:

Finalidade do site

- *Descrever aqui.*

Mapa do site

- *Desenhar a estrutura hierárquica do site, dar nome às áreas a serem criadas, conforme características das informações.*

Detalhes de cada área do site

- *Descrever o conteúdo que será disponibilizado nas áreas do mapa do site. É importante colocar na descrição o tipo de informação que será publicada, por exemplo, texto, imagem, tabelas, anexos, etc.*

Para publicação de conteúdo serão utilizados quais papéis?		
• Moderador	[ ] Sim	[ ] Não
• Revisor	[ ] Sim	[ ] Não
• Autor	[ ] Sim	[ ] Não

Informações sobre Templates para o site

A ferramenta possui alguns templates já importados e prontos para uso, podendo ser utilizados conforme interesse do órgão. O órgão também poderá entregar uma nova proposta de template para publicar seu site no Framework Pantaneiro, para isso é importante que o desenhista / webdesigner:

1. Conheça o menu da ferramenta.
2. Construa o site em HTML/Table Less e entregue o arquivo com as imagens já trabalhadas.
3. As formatações sejam feitas por estilo (CSS) e tenham uma otimização do uso das classes.
4. Construa o modelo atendendo uma resolução de vídeo 800 X 600.
5. Sempre que possível troque informações com a equipe da UGSI responsável pela importação do novo template para a ferramenta.

ANEXO III

Modelo de Ofício de Solicitação de Validação e Publicação

Ofício nº

Campo Grande, _____ de _____ de 2007.

Senhor Subsecretário,

Informamos a conclusão dos trabalhos de construção do sítio .....na ferramenta Pantaneiro e solicitamos a validação e publicação do mesmo.

Atenciosamente,

_____
Secretário
Secretaria de Estado .....

Ilmo Senhor
Guilherme Villalba Zurutuza Filho
Subsecretário de Comunicação
Secretaria de Estado de Governo
Nesta