FACULDADE DE COMPUTAÇÃO - FACOM
UNIVERSIDADE FEDERAL DO MATO GROSSO DO SUL – UFMS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

VITOR MESAQUE ALVES DE LIMA

Linha de Produtos de Software no Domínio de Portais e-Gov Acessíveis

CAMPO GRANDE
2013

VITOR MESAQUE ALVES DE LIMA

Linha de Produtos de Software no Domínio de Portais e-Gov Acessíveis

Dissertação apresentada ao Programa de Mestrado Stricto Sensu em Ciência da Computação da Faculdade de Computação, mantido pela Universidade Federal do Mato Grosso do Sul, como requisito para a obtenção do título de Mestre em Ciência da Computação (Área de Concentração: Engenharia de Software).

Orientador: Prof. Dr. Marcelo Augusto Santos Turine

CAMPO GRANDE
2013

Agradecimentos

Agradeço primeiramente a Deus, por ter me dado inteligência e sabedoria para desenvolver este trabalho.

A mulher da minha vida, minha esposa Michelly, por todo amor e cuidado, me apoiando e incentivando em todos os momentos.

Aos meus pais Josimar e Linda, por tudo que eles representam para mim, um exemplo de fé e amor. Tudo que sou hoje é fruto do amor e dedicação deles.

Aos meus irmãos Vinícius e Milca, pelo companheirismo e amizade verdadeira em todos os momentos da minha vida.

Ao meu orientador e amigo, professor Marcelo Turine, pela dedicação, apoio e incentivo em todas as minhas conquistas acadêmicas e profissionais.

Ao Camilo, pelas importantes contribuições durante o desenvolvimento deste trabalho.

A todos os meus amigos do LEDES, pelo aprendizado, conquistas e bons momentos que passamos juntos.

A todos os professores da Facom que participaram da minha formação.

À CAPES pelo apoio financeiro.

“Tudo é possível ao que crê”.

Jesus Cristo

Resumo

As Tecnologias de Informação e Comunicação (TIC) têm se desenvolvido com intensidade nos últimos anos, espalhando-se pelo setor governamental por meio do que é chamado e-Gov ou Governo Eletrônico, representado pela informatização de suas atividades internas e pela comunicação com o público externo. As diretrizes do Governo Eletrônico brasileiro estimulam a cidadania, inclusão digital e no que tange ao desenvolvimento de software enfatizam a utilização de tecnologias *open source*, racionalização de recursos e o desenvolvimento colaborativo. Uma forma para essa informatização é a construção de portais governamentais. Os portais e-Gov são um meio para a democratização da informação e devem viabilizar o acesso a ela para todos os cidadãos sem distinção. Entretanto, grande parte dos portais e-Gov desenvolvidos não seguem os padrões Web e nem padrões de acessibilidade, o que inviabiliza o acesso à informação por pessoas com necessidades especiais. O reúso de software tem como objetivo principal o desenvolvimento de produtos de qualidade e economicamente viável utilizando artefatos já especificados, implementados e testados. Nesse contexto, visando consolidar uma plataforma para o desenvolvimento de portais e-Gov acessíveis, é apresentada uma plataforma baseada em software livre e tecnologias que promovem o reúso, racionalizando recursos e estimulando a colaboração entre órgãos governamentais. Dentre as diversas abordagens de reúso encontradas na literatura, optou-se, nesta proposta, por utilizar um processo de Linha de Produtos de Software (LPS), automatizado por meio do uso de *frameworks*, geradores de aplicação e repositório de reúso. Com a utilização de tais tecnologias, pretende-se desenvolver uma plataforma de LPS automatizada no domínio de portais e-Gov acessíveis, promovendo o reúso de software e o desenvolvimento colaborativo.

Palavras-chave: Linha de Produtos de Software, Geradores de Aplicação, *Framework*, Repositórios de Reúso, Governo Eletrônico e Acessibilidade.

Abstract

The Information and Communication Technologies (ICT) have been developed intensively in recent years, spreading across the government sector through what is called e-Gov or Electronic Government, represented by the informatization of its internal activities and communication with the external public. The guidelines of the Brazilian Electronic Government stimulate the citizenship, digital inclusion, and with respect to software development, emphasize the use of open source technologies, resources rationalization and collaborative development. A way to the informatization is the construction of government portals. The e-Gov portals promote the information democratization and should facilitate access to care for all citizens without distinction. However, most e-Gov portals developed not follow web standards and accessibility standards, preventing access to people with disability. The software reuse has as main objective the development of quality products and economically viable products, using artifacts already specified, implemented and tested. In this context, in order to consolidate an development process for the acessible e-Gov domain, is presented a platform based on open source software and technologies that promote reuse, resource rationalization and collaboration between government agencies. Among the various techniques of reuse found in the literature, we decided on this proposal by using a process of Software Product Line (SPL), automated through the use of frameworks, application generators and reuse repository. With the use of such techniques, we intend to develop a automated SPL platform for generation of accessible e-Gov portals by promoting the software reuse and collaborative development.

Palavras-chave: Software Product Line, Application Generators, Framework, Reuse repositor, Eletronic Governament and Accessibility.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Capítulo 1

Introdução

1.1Contextualização

O desenvolvimento das Tecnologias de Informação e Comunicação (TIC) nos últimos anos tem viabilizado diversas possibilidades de formação e disponibilização de informação, inclusive na área governamental. As TICs proporcionam o estabelecimento de uma maior transparência e participação popular na esfera governamental, e têm se espalhado pelo setor por meio do que é chamado de e-Gov ou Governo Eletrônico [1], representado pela informatização de suas atividades internas e pela comunicação com o público externo. A necessidade de investimento em Governo Eletrônico é justificada pela aparição de um novo espaço público, criado a partir do surgimento de novas tecnologias de comunicação, como a Internet (PINHO, 2005) (PINHO, 2008).

A Internet é um instrumento ideal para promover a democracia, fornecendo fácil acesso à informação política e permitindo que os cidadãos sejam quase tão bem informados quanto seus líderes, possibilitando a eles a solicitação de informações e expressão de opiniões (CASTELLS, 2003). A Internet é provavelmente a mais sofisticada forma de expandir a informação e comunicação atualmente disponível para a sociedade, em função de sua organização e de seus impactos nas esferas tecnológicas, social, econômica e política. Ela proporciona a infraestrutura necessária para uma de suas maiores e mais conhecida aplicação: a Web, que é a grande responsável pela popularização da Internet, a ponto de serem confundidas (PINHO, 2008).

Internet e Web são, portanto, conceitos distintos. A Web pode ser definida, de maneira geral, como a parte da Internet acessada por meio de navegadores, ou *browsers*, já a Internet

[1] O termo Governo Eletrônico, em inglês Electronic Government, possui, na língua inglesa, uma expressão simplificada – e-government – que é comumente utilizada. A expressão ganhou uma versão em português – e-governo – e também uma abreviatura frequentemente encontrada tanto em português como em inglês – e-gov. (VILELLA, 2003)

impulsionou o surgimento e o crescimento de uma nova geração de softwares na Web denominados WebApps (*Web based Applications*) (CGI.br e NIC.br, 2010)[2].

Nesse cenário governamental e tecnológico, aparecem os portais e-Gov, por intermédio dos quais os governos mostram sua identidade, seus propósitos, suas realizações, possibilitando a concentração e disponibilização de serviços e informações aos cidadãos. Os portais e-Gov são, portanto, WebApps que atuam na promoção da democratização da informação, propondo-se a viabilizar o acesso a ela para todos os cidadãos, em qualquer dispositivo e lugar, de forma segura e confiável (PINHO, 2008).

Entretanto, a realidade atual dos portais de órgãos públicos brasileiros não reflete o ideal esperado, uma vez que para garantir a universalização do acesso é necessário que os portais sejam construídos em conformidade com os padrões Web e de acessibilidade. Pesquisas divulgadas em 2010 mostram que 98% dos portais e-Gov não seguem esses padrões (CGI.br e NIC.br, 2010), inviabilizando o acesso à informação por pessoas com necessidades especiais. Assim, se faz necessário o emprego de princípios e tecnologias que promovam a construção de WebApps acessíveis. O governo então assume um papel importante na elaboração de iniciativas que promovam a inclusão e a universalização do acesso a serviços e informações disponibilizadas nos sítios e portais governamentais.

Perante as demandas tecnológicas supracitadas, o programa de Governo Eletrônico brasileiro, criado para regulamentar a utilização das TICs no setor governamental, propôs um conjunto de diretrizes que apontam para a promoção da cidadania, inclusão digital, utilização de software livre, gestão de conhecimento, racionalização de recursos, padronização e integração de informações (E-GOV, 2012). Sob essa perspectiva, pode-se perceber que além de aspectos políticos e sociais, o governo também está interessado nos custos associados às questões tecnológicas. É conhecido que as organizações, inclusive governamentais, buscam cada vez mais a redução de esforço, custo e tempo no desenvolvimento de software (VINCENZO e GIOVANNI, 2010). Assim, várias pesquisas na área de Engenharia de Software estão sendo realizadas objetivando propor tecnologias que supram tais demandas. Dentre as tecnologias propostas destacam-se as abordagens de reúso de software, tais como, Linhas de Produtos de Software (LINDEN *at al*., 2007) (CLEMENTS e NORTHROP, 2002), *frameworks* (FAYAD, 1999) (BRAGA, 2003), geradores de aplicação (CLEAVELAND, 1988) (CZARNECKI e EISENERCKER, 2002), repositório de reúso (BURÉGIO *et al*.,

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

2007), dentre outras. Essas técnicas, quando empregadas de forma sistemática, visam alcançar níveis ideais de qualidade, produtividade e manutenibilidade no processo de desenvolvimento e nos produtos. É consenso na literatura que a partir do reúso de modelos, de projeto e de partes da implementação de software, pode-se construir novos produtos em menor tempo, custo e confiabilidade (BENGTSSON *et al*., 1999) (GIMENES e TRAVASSOS, 2002) (BRAGA, 2003) (BALZERANI *et al*., 2005).

A abordagem de LPS, escopo da presente pesquisa, é uma solução para a construção de softwares numa perspectiva de reutilização em larga escala, a partir de um conjunto de artefatos comuns de um domínio em particular (LINDEN at al., 2007) (CLEMENTS e NORTHROP, 2002). Carromeu *et al*. (2010) definem um processo de LPS para a construção de WebApps no domínio e-Gov, abrangendo sítios e portais governamentais, permitindo a reutilização sistemática de artefatos por meio de uma arquitetura baseada em componentes e utilização de ferramentas automatizadas. Após estudos e análises do processo de LPS proposto por Carromeu *et al*. (2010) foram identificados pontos que poderiam ser melhorados e adaptados para alcançar níveis ideais de reúso e aumentar o nível de acessibilidade das aplicações criadas.

Diante deste panorama, surge uma oportunidade para propor e explorar novas tecnologias na implementação de produtos de software reutilizáveis com qualidade utilizando LPS.

1.2 Motivação e Justificativa

Desde meados dos anos 90, a Web brasileira tem mostrado acentuado crescimento, tanto no número de usuários como na abrangência de serviços e aplicações oferecidas por meio da Internet. É notável o avanço de seu uso pela população brasileira: de 37 milhões de usuários, em 2005, passou para aproximadamente 65 milhões em 2009, conforme mostram as pesquisas direcionadas pelo Comitê Gestor de Internet no Brasil (CGI.br) e pelo Núcleo de Informação de Coordenação do Ponto BR (NIC.br). É também igualmente perceptível a mudança de comportamento do cidadão, que utiliza cada vez mais serviços transacionais em ambientes virtuais. Contudo, o crescimento acelerado da Internet trouxe consigo também o aumento de conteúdos inacessíveis na Web, devido a falta de suporte a acessibilidade das WebApps desenvolvidas (CGI.br e NIC.br, 2010).

Pesquisa recente do World Wide Web Consortium Brasil (W3C)[3] mostra que o indicador de conformidade aos padrões de acessibilidade dos sítios e portais de órgãos públicos brasileiros alcança a média de apenas 2% do total de mais de 6 milhões de páginas Web analisadas (CGI.br e NIC.br, 2010). Esse número indica que há uma distância considerável a percorrer em busca de acessibilidade na web.

Comprometido com a inclusão digital, o governo brasileiro buscou, por meio da elaboração do Modelo de Acessibilidade de Governo Eletrônico (e-MAG) (e-MAG, 2011), guiar o desenvolvimento de sítios e portais do governo brasileiro, a partir de um conjunto de recomendações e diretrizes que devem ser aplicadas para que as barreiras de acesso sejam eliminadas ou reduzidas. Tais técnicas permitem que navegadores e tecnologias assistivas, como leitores de tela, compreendam o conteúdo de forma adequada e apresentem ao usuário um resultado claro, inteligível e sem empecilhos.

Preocupado ainda com a qualidade das aplicações e-Gov, o governo propôs o documento e-PWG (Padrões Web em Governo Eletrônico) (e-PWG, 2011), que fornece recomendações de boas práticas de desenvolvimento, com relação aos aspectos de apresentação, organização, navegação, usabilidade e acessibilidade dos portais e-Gov. Segundo o e-PWG (2011), a adoção dos padrões propostos traz benefícios na criação e na gestão de sítios e portais, tais como, a diminuição de tempo e custo de desenvolvimento, bem como melhoria no nível de qualidade e manutenção das páginas web.

De acordo com o e-PWG, os órgãos da Administração Pública Federal devem implementar ferramentas de controle editorial das informações publicadas nos sítios e portais. Tais ferramentas devem a) permitir o monitoramento da inclusão e atualização de conteúdo; b) organizar as informações em bancos de dados administrados por módulo de gestão descentralizado; c) estruturar as informações e serviços de modo a permitir seu manuseio e manutenção, independente de conhecimento técnico; e d) publicar a data da informação e periodicidade de sua atualização.

As diretrizes supracitadas apontam para a adoção de um Sistema de Gerenciamento de Conteúdo (*Content Management Systems* - CMS), que é um sistema que integra uma série de soluções para a administração, manutenção e evolução de sítios e portais. Por automatizarem o processo de gestão e publicação, permitem que usuários não técnicos possam criar

[3] O W3C é um consórcio internacional com a missão de conduzir a Web ao seu potencial máximo, criando padrões e diretrizes que garantam a sua evolução permanente.

conteúdos com maior facilidade (e-PWG, 2011). Nesse sentido, de acordo com pesquisas recentes (BURZAGLI *et al*., 2008) (FOGLI, 2009) (FOGLI E SACCO, 2010) (LOPEZ *et al*., 2012) algumas soluções *open source* têm sido adotadas, tais como, Joomla! (JOOMLA, 2012), Drupal (DRUPAL, 2012), PLONE (PLONE, 2013), OpenCMS (OPENCMS, 2013), Typo3 (TYPO3, 2013), dentre outras. No âmbito deste trabalho foi utilizado o Titan Framework, um *framework* para a criação de CMSs, concebido a partir de um processo de LPS para o desenvolvimento de WebApps no domínio[4] e-Gov (Carromeu *et al*., 2010). Diversos portais institucionais e governamentais foram construídos com sucesso utilizando o Titan Framework (Carromeu *et al*., 2010).

Na liteturatura, por meio um mapeamento sistemático conduzido durante o período deste trabalho pode-se constatar que há um crescente interesse em pesquisas relacionadas à utilização de CMSs acessíveis no domínio e-Gov. Foram evidenciadas as dificuldades relacionadas ao desenvolvimento e utilização de CMSs acessíveis, bem como a carência de abordagens sistemáticas de reutilização.

Assim, objetivando atingir níveis ideais de reúso e portais e-Gov com alto nível de qualidade e acessibilidade, este trabalho de mestrado explora a utilização de LPS, automatizada por meio de uma plataforma composta por *frameworks*, geradores de aplicação e repositório de reúso, a fim de sistematizar uma abordagem de reúso, potencializando o reaproveitamento de recursos e esforços despendidos em desenvolvimentos anteriores, empregando técnicas e mecanismos que promovem a acessibilidade nos portais gerados.

1.3 Objetivos

O objetivo geral deste trabalho é propor uma solução eficiente para a construção de portais e-Gov acessíveis, por meio de uma plataforma automatizada de LPS, combinando a utilização de *frameworks*, geradores de aplicação e repositório de reúso com o objetivo de tornar o processo de criação de portais e-Gov mais ágil e menos suscetível a erros.

Os objetivos específicos deste trabalho são:

- Estender o *framework* Titan com o objetivo de facilitar a criação de portais e-Gov;

[4] Neste trabalho, o termo domínio será utilizado para denotar uma área de conhecimento ou atividade caracterizada por um conjunto de conceitos e terminologia compreendidos pelos participantes dessa área (BOOCH *et al*., 2000).

- Projetar e implementar o gerador de aplicações Titan FrontEnd utilizado na automação da plataforma da LPS;
- Projetar e implementar o repositório de reúso Titan Extensions, um ambiente que possibilitará a disponibilização de extensões (*extensions*)[5] para o *framework* Titan;
- Avaliar a plataforma proposta por meio de um questionário de avaliação aplicado a especialistas.

Os resultados deste trabalho visam apoiar o processo de desenvolvimento de software na construção e evolução de portais e-Gov acessíveis.

1.4 Organização do Texto

Este texto está dividido em mais cinco capítulos. No Capítulo 2 são apresentados os conceitos fundamentais que compõem a base teórica deste trabalho, que inclui uma visão geral sobre Governo Eletrônico, acessibilidade Web e reúso de software, em especial as abordagens de LPS, *frameworks*, geradores de aplicação e repositórios de reúso. No Capítulo 3 são apresentados os resultados de pesquisas realizadas com o intuito de investigar o problema na prática, bem como um mapeamento sistemático conduzido com o objetivo de identificar lacunas na literatura. No Capítulo 4 é apresentada a plataforma de LPS proposta, bem como o detalhamento dos processos e ferramentas utilizadas. No Capítulo 5 são apresentados os resultados de uma avaliação conduzida com a finalidade de avaliar a plataforma de LPS proposta. Por fim, no Capítulo 6, são apresentadas as conclusões, contribuições e limitações deste trabalho, assim como as possibilidades de trabalhos futuros são discutidas.

[5] Neste trabalho, o termo *extensions* será usado para representar os ativos de softwares reutilizáveis disponíveis no repositório Titan Extensions, sendo portanto, extensões do *framework* Titan.

Capítulo 2

Embasamento Teórico

2.1.1 Considerações Iniciais

O Governo Eletrônico tem utilizado a Internet para cumprir com os seus propósitos, em particular, por meio dos portais governamentais. É imprescindível que tais portais estejam em conformidade com as recomendações de acessibilidade para garantir o acesso a todos os cidadãos. Grande parte dos órgãos governamentais ainda não possuem portais, e dos que possuem, poucos são acessíveis e muitos ainda utilizam tecnologias proprietárias (CGI.br e NIC.br, 2010). Neste cenário, é possível perceber a importância de pesquisas relacionadas ao desenvolvimento de portais e-Gov acessíveis e a redução de custos nesse processo. É provado que técnicas de reúso de software podem reduzir significamente custos e esforços na construção de novas aplicações (BENGTSSON *et al*., 1999) (GIMENES e TRAVASSOS, 2002) (BRAGA, 2003) (BALZERANI *et al*., 2005).

Neste capítulo são apresentados os principais conceitos utilizados neste trabalho. Nas Seções 2.2 e 2.3 é apresentada uma visão geral sobre Governo Eletrônico e acessibilidade Web, respectivamente. Na Seção 2.4 são apresentadas as principais definições sobre reusabilidade de software e algumas abordagens de sua aplicação. Nas Seções 2.5, 2.6, 2.7 e 2.7.4 são apresentados os principais conceitos utilizados neste trabalho, sendo eles, *frameworks* orientados a objetos, geradores de aplicação, LPS e repositórios de reúso, respectivamente, bem como o detalhamento das ferramentas já existentes que irão compor a plataforma proposta. Por fim, na Seção 0, são apresentadas as considerações finais do capítulo.

2.2 Governo Eletrônico

As definições para Governo Eletrônico (e-Gov) variam desde a mais genérica — "uso de TICs e sua aplicação pelo governo para prestação de informações e serviços públicos às pessoas" (CURTIN, 2007); "qualquer utilização de TICs em serviços e administração pública" (BANNISTER, 2007) — para as mais específicas — "a entrega de informações governamentais e serviços *online* por meio da Internet ou outros meios digitais" (WEST,

2004); “prestação de serviços do governo por meio da internet em geral, e a Web em particular” (BANNISTER, 2007). 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.

No contexto brasileiro, o e-Gov surgiu no ano 2000, quando foi criado um Grupo de Trabalho Interministerial com a finalidade de examinar e propor políticas, diretrizes e normas relacionadas às novas formas eletrônicas de interação. A política de e-Gov segue um conjunto de diretrizes baseada em três frentes fundamentais: junto ao cidadão; na melhoria da sua própria gestão interna; na integração com parceiros e fornecedores. A seguir é apresentada uma síntese desse conjunto de diretrizes disponibilizadas no portal oficial do Programa de Governo Eletrônico brasileiro (E-GOV, 2012).

2.2.1 Diretrizes do Governo Eletrônico (E-GOV, 2012)

1. A prioridade do Governo Eletrônico é a promoção da cidadania

A política de governo eletrônico do governo brasileiro 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 equidade na oferta de serviços e informações.

2. A 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. A inclusão digital 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.

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

O software livre deve ser entendido como opção tecnológica do governo federal. Onde possível, deve ser promovida sua utilização. 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 entendida somente como motivada por aspectos econômicos, mas pelas possibilidades que oferece no campo da produção e circulação de conhecimento, no acesso a novas tecnologias e no estímulo ao desenvolvimento de software em ambientes colaborativos e ao desenvolvimento de software nacional.

4. A 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 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.

5. O Governo Eletrônico deve racionalizar o uso de recursos

O governo eletrônico não deve significar aumento dos dispêndios do governo federal 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 através do compartilhamento de recursos entre órgãos públicos. Deve merecer destaque especial o

desenvolvimento compartilhado em ambiente colaborativo, envolvendo múltiplas organizações.

6. **O 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 que cubram uma série de fatores críticos para o sucesso das iniciativas.

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.

A plataforma de LPS proposta neste trabalho é coerente com as diretrizes supracitadas, visto que os portais gerados a partir da instanciação da linha promoverão a cidadania, inclusão digital e gestão de conhecimento, oferecendo serviços e informações aos cidadãos. A plataforma de LPS também proporcionará a racionalização de recursos e o desenvolvimento colaborativo, reduzindo o custo e o esforço na construção de portais e permitindo o compartilhamento de artefatos de software desenvolvidos durante a instanciação da LPS. Além disto, propiciará a utilização de tecnologias *open source*, uma vez que a plataforma proposta é composta em sua completude por tecnologias que seguem a abordagem de Software Livre.

2.3 Acessibilidade Web

De acordo com o W3C (W3C/WAI, 2005), acessibilidade Web permite que pessoas com deficiência sejam capazes de utilizar a Internet, tornando possível a percepção, entendimento, navegação, e interação com a Web.

Segundo Park (2012), além dos aspectos políticos, sociais e legais, a acessibilidade Web provê benefícios técnicos e financeiros: reduzindo tempo e custo de desenvolvimento e

manutenção de aplicações; reduzindo a exigência de recursos de computação e carga do servidor; aumentando a interoperabilidade Web e independência de dispositivos; e preparação para novas tecnologias.

Uma alternativa para a diminuição dos obstáculos de acesso ao conteúdo Web é a utilização de recomendações que direcionam a promoção da acessibilidade. Essas recomendações têm como objetivo encorajar projetistas a desenvolverem sítios e portais em conformidade com as especificações, possibilitando que tecnologias assistivas sejam empregadas com sucesso a fim de permitir aos usuários com deficiência a interação com o conteúdo Web (HARPER e YESILADA, 2008).

A principal recomendação de acessibilidade Web existente é o WCAG (WCAG 2.0, 2008), criado pelo W3C/WAI (*World Wide Web Consortium/Web Accessibility Initiative*). No Brasil, o governo elaborou o e-MAG, com o objetivo de facilitar o acesso às informações e serviços disponibilizados nos sítios e portais do governo brasileiro a todas as pessoas, sem distinção (e-MAG, 2011).

O e-MAG consiste em um conjunto de recomendações a ser considerado para que o processo de acessibilidade dos sítios e portais do governo brasileiro seja conduzido de forma padronizada e de fácil implementação. O e-MAG é coerente com as necessidades brasileiras e em conformidade com os padrões internacionais. Foi formulado para orientar profissionais que tenham contato com publicação de informações ou serviços na Internet a desenvolver, alterar e/ou adequar páginas, sítios e portais, tornando-os acessíveis ao maior número possível de pessoas (GOV, 2012).

Para a elaboração da versão 2.0 do e-MAG foi realizado um estudo das regras de acessibilidade por meio de um método comparativo entre as normas adotadas por diversos países, como Estados Unidos, Canadá, Irlanda, Portugal, Espanha, entre outros. Também foi realizada uma análise detalhada das regras e pontos de verificação do WAI/W3C, presentes na WCAG 1.0. A elaboração da versão 3.0 foi embasada na versão anterior do e-MAG, apoiando-se na WCAG 2.0, lançada em dezembro de 2008, e considerando as novas pesquisas na área de acessibilidade Web. Apesar de utilizar a WCAG como referência, o e-MAG 3.0 foi desenvolvido e pensado para as necessidades locais, visando atender as prioridades brasileiras e mantendo-se alinhado ao que existe de mais atual neste segmento (e-MAG, 2011).

A plataforma de LPS apresentada neste trabalho tem como objetivo principal a geração de portais e-Gov que atendam aos padrões brasileiros estabelecidos no e-MAG 3.0.

2.4 Reúso de Software

A reutilização de software surgiu no final dos anos 60 como uma alternativa para superar a crise do software e tem como objetivo principal desenvolver software com qualidade e economicamente viável utilizando artefatos já especificados, implementados e testados. Com a crescente demanda por sistemas cada vez maiores e complexos, acompanhada da necessidade de desenvolver e mantê-los em prazos cada vez menores sem comprometer a qualidade, tornou-se necessária a utilização de tecnologias que promovem o reúso (PRESSMAN, 2005).

O reúso caracteriza-se pela utilização de artefatos de software numa situação diferente daquela para a qual foram originalmente construídos (PRESSMAN, 2005) (CHEESMAN e DANIELS, 2001). Segundo Frakes e Kang (2005), a reutilização pressupõe o uso de artefatos existentes ou o conhecimento para criação de novos softwares. Com o aumento da demanda e da complexidade dos sistemas nas últimas décadas surgiram também muitos problemas inerentes ao desenvolvimento de software, tais como: atrasos nos prazos de entrega; aumento exponencial da dificuldade de desenvolvimento em decorrência do aumento da complexidade; a baixa qualidade dos produtos; a manutenção e evolução do software tornaram-se mais difíceis (PRESSMAN, 2005). Nesse cenário, pode-se perceber a importância da aplicação de técnicas de reúso no processo de desenvolvimento.

O reúso de software se propõe a prover soluções aos problemas supracitados, reduzindo o esforço de desenvolvimento e a manutenção de novos sistemas, reaproveitando qualquer tipo de conhecimento sobre outro sistema similar. Várias técnicas de reúso têm sido propostas na literatura, com o objetivo de aproveitar os esforços e recursos despendidos quando se iniciam novos desenvolvimentos. Dentre as técnicas propostas têm-se as abordagens de geradores de aplicação, *frameworks*, LPS, e repositórios de reúso.

2.5 Frameworks Orientados a Objetos

Os *frameworks* são definidos como aplicações semi-completas e reutilizáveis que, quando especializadas, produzem aplicações personalizadas dentro de um domínio específico

(FOOTE e JOHNSON, 1988). Para Johnson (1997), um *framework* pode ser definido como sendo o esqueleto de uma aplicação, podendo ser instanciado por um desenvolvedor de aplicações. A estrutura de um *framework* é composta por um conjunto de classes que contém o projeto abstrato de soluções para uma família de problemas, propiciando o reúso com granularidade maior do que classes (FOOTE e JOHNSON, 1988). É formado por uma coleção de classes abstratas e concretas e de interfaces entre elas, representando o projeto de um subsistema (SOMMERVILLE, 2011).

De acordo com Fayad e Schmidt (1997) os frameworks podem ser classificados, segundo seu escopo, em três grupos: a) *Frameworks* de infraestrutura do sistema: simplificam o desenvolvimento da infraestrutura de sistemas portáveis e eficientes como, por exemplo, sistemas operacionais, sistemas de comunicação, interfaces com o usuário e ferramentas de processamento de linguagem; b) *Frameworks* de integração middleware: geralmente usados para integrar aplicações e componentes distribuídos, são projetados para melhorar a habilidade de desenvolvedores em modularizar, reutilizar e estender a infraestrutura de software para funcionar em um ambiente distribuído; e (c) *Frameworks* de aplicação: voltados para domínios de aplicação e são base para atividades de negócios das empresas, tais como, sistemas de telecomunicações e aviação.

Os *frameworks* são compostos de partes fixas e partes variáveis. As partes fixas são denominadas pontos-fixos (*frozen-spots*), os quais são estáveis e imutáveis durante o uso do *framework* em diversas aplicações. As partes variáveis são denominadas pontos variáveis (*hot-spots*) e devem ser adaptadas de forma particular em cada instanciação do *framework* (SHIMABUKURO, 2006).

Quanto à forma de reúso, os *frameworks* podem ser classificados como caixa-branca (*whitebox*), caixa-preta (*blackbox*) ou caixa-cinza (*graybox*). As funcionalidades de um *framework* caixa-branca são reusadas e estendidas por meio de herança. Desta forma, o usuário necessita conhecer detalhes das estruturas internas do *framework* para utilizá-lo. Já o *framework* caixa-preta permite estender suas funcionalidades por composição ou definição de interfaces para os componentes. Assim, o usuário deve entender apenas a interface para usar este tipo de *framework*. Por fim, o *framework* caixa-cinza é uma combinação de ambos, caixa-branca e caixa-preta, ou seja, o reúso é feito por meio de herança e pela definição de interfaces. Ele deve possuir flexibilidade e extensibilidade suficiente, e a habilidade de ocultar de seus usuários informações desnecessárias (Fayad *et al*., 1999).

Resumidamente, um *framework* é um grande modelo de um domínio de aplicações e, desta forma, pode ser desenvolvido a partir de um conjunto de aplicações do domínio que atuam como fontes de informação deste domínio (Silva, 2000).

A plataforma da LPS proposta neste trabalho é composta por dois *frameworks* de aplicação (Titan Framework e Zend Framework). Ambos possuem como característica serem flexíveis e extensíveis, e quanto à forma de reúso, o Titan e o Zend podem ser classificados como caixa-cinza e caixa-branca, respectivamente.

2.5.1 Titan Framework

O Titan é um *framework* de aplicação que foi especificado e implementado pelo grupo de pesquisa do Laboratório de Engenharia de Software (LEDES) da Faculdade de Computação (FACOM) da Universidade Federal de Mato Grosso do Sul (UFMS) para instanciar aplicações Web no domínio de CMS. A partir de uma arquitetura genérica comum e de um conjunto de artefatos que compõe a arquitetura, são desenvolvidos os produtos de software. Cada produto é, portanto, uma instância parametrizada do Titan. O Titan recebe como entrada uma linguagem de marcação estendida (XML) e a transforma, em tempo de execução, em um gerenciador de conteúdo.

- **Arquitetura**

A arquitetura do *framework* Titan é composta por um núcleo e um repositório, conforme ilustrado na Figura 2.1. O núcleo (*core*) tem a particularidade de ser imutável e é responsável por receber como entrada os arquivos de configuração da instância (XML e SQL) e gerar uma aplicação em tempo de execução. Desta forma, apenas um núcleo é necessário para executar todas as aplicações instanciadas pelo *framework* em um mesmo servidor. O núcleo contém características e funcionalidades, tais como um sistema de autenticação e segurança, *log*, controle de revisões e autoria. Além do núcleo existe um conjunto de ativos de software localizados em um repositório (*repository*).

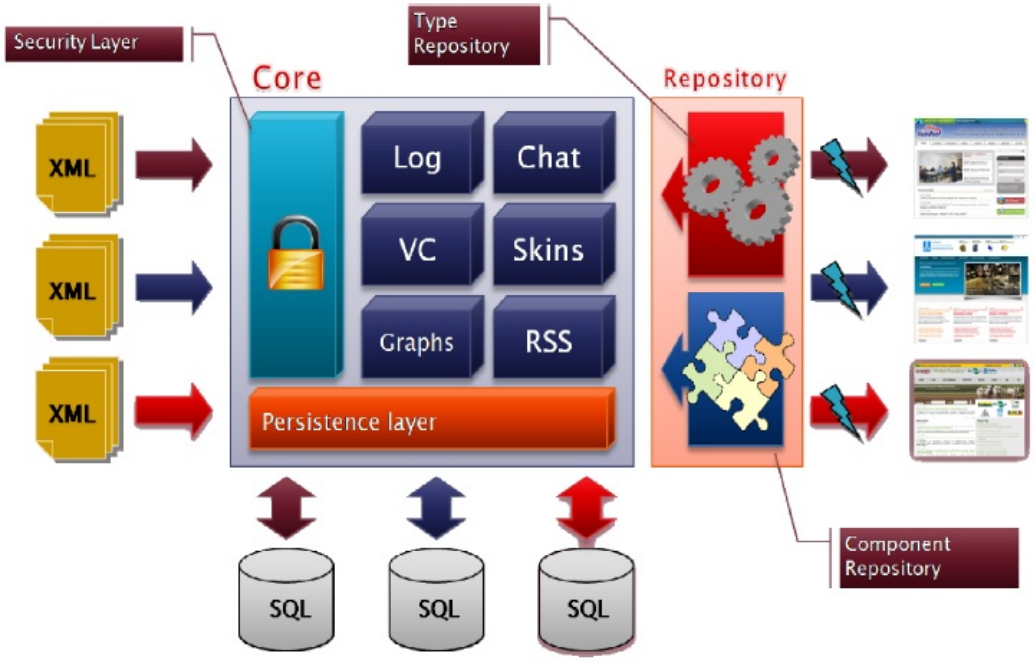

Figura 2.1: Arquitetura do Titan Framework (CARROMEU *et al*., 2010).

O repositório do *framework* Titan é composto por:

- Tipos: classes responsáveis por tratar os tipos de dados no *framework*.
- *Icon*: classes responsáveis por tratar e direcionar ações sobre itens dentro de uma seção do *framework*.
- Ldap: classes responsáveis por implementar diversas interfaces de comunicação Ldap.
- Menu: classes que representam funcionalidades dentro de uma seção do *framework*. Por exemplo, o menu *Pdf* é uma funcionalidade que quando acionada gera um arquivo PDF para *download* contendo os dados exibidos;
- *Packages*: conjunto de arquivos que compõe uma pré-configuração que pode ser utilizada para instanciar uma seção em uma instância do Titan. Esses arquivos são compostos basicamente dos arquivos XML e, caso necessário interação com o banco de dados, o arquivo SQL para a criação das tabelas da seção.
- Componentes: para gerenciar cada uma das seções da aplicação existe uma entidade seção no *framework* Titan. Por exemplo, uma possível área de notícias do portal é gerenciada por uma seção no CMS, enquanto uma área de agenda e eventos é gerenciada por outra seção específica para esse fim. Cada

> seção da instância possui um conjunto de arquivos de configuração próprio e faz, obrigatoriamente, uso de um componente do *framework*. Os componentes são, portanto, artefatos de código parametrizáveis que recebem como entrada arquivos XML gerando diferentes seções.

O núcleo e o repositório agregam todos os artefatos responsáveis pela renderização da aplicação, que acontece em tempo real por meio dos arquivos XML de entrada que o parametrizam.

O *framework* Titan possui em sua arquitetura artefatos reutilizáveis, por exemplo, os componentes. No contexto de um portal e-Gov, uma seção pode ser definida como uma área específica em função de seu contexto. Por exemplo, algumas seções existentes em vários portais são: notícias, eventos, galeria de fotos, enquete, fale conosco e contato. Para gerenciar cada uma das seções da aplicação existe uma entidade seção (*section*) no *framework* Titan. Cada seção da instância possui um conjunto de arquivos de configuração próprio e faz, obrigatoriamente, uso de um componente do *framework*. Os componentes são, portanto, artefatos de código com funcionalidades específicas e que, quando parametrizados, geram seções. Várias seções da instância podem utilizar ao mesmo tempo o mesmo componente, isto ocorre quando se deseja tratar de forma semelhante seções distintas, onde mudam-se apenas campos ou rótulos. Dessa forma, modificando-se os parâmetros de entrada dos componentes têm-se diferentes seções (CARROMEU, 2009).

Exemplificando, o componente *global.generic* do Titan permite efetuar a listagem, criação, edição e remoção de itens de uma seção. Nesse componente estão embutidas uma série de funcionalidades, tais como, busca, ordenação, paginação, validação, impressão de relatórios quantitativos com gráficos, formulários com múltiplos passos, controle de versões, RSS, entre outras. Assim, esse único componente consegue atender as diferentes seções de uma aplicação. No entanto, quando uma seção que possui funcionalidades muito distintas, como uma seção de galeria de fotos, necessita ser gerenciada, faz-se necessário criar um novo conjunto de artefatos que possibilite a instanciação dessas funcionalidades.

Cada seção do CMS é formada por uma coleção de ações. Cada ação é uma funcionalidade bem específica da seção. Por exemplo, a listagem de itens de uma seção é uma ação, enquanto a criação de novos itens ou edição de itens existentes são outras ações. Cada ação de uma seção faz uso de um motor (*engine*), que fisicamente é um conjunto de três arquivos que renderizam as ações e têm funções bem definidas: preparação (*prepare*),

visualização (*view*) e submissão (*commit*). Os arquivos de preparação separam a camada de execução, persistência e acesso ao SGBD (Sistema de Gerenciamento de Banco de Dados) da camada de visualização da ação. O arquivo de submissão tem como finalidade capturar e salvar as modificações e interações do usuário com a aplicação. Várias ações da seção podem fazer uso de um único motor (CARROMEU, 2009).

Todo o código do Titan está disponível em um repositório público com controle de versões. Dessa forma, além de se obter o código fonte, é possível atualizar versões do *framework* com um único comando do controlador de versões.

- ## Características

A arquitetura do *framework* Titan possui como característica ser caixa-cinza, flexível e extensível, possibilitando que componentes sejam criados através de sua API orientada a objetos (CARROMEU *et al*., 2010). Por ter um único *core*, novas funcionalidades são herdadas automaticamente para cada instância gerada e as atualizações dos arquivos do *core* e *repository* são feitas por um gerenciador de versões, tornando mais fácil a atualização periódica do *framework*. Dentre as diversas características nativas das instâncias do *framework* Titan, pode-se destacar:

- controle de acesso e autenticação;
- controle de versões e auditoria;
- geração automática de manual do usuário das aplicações instanciadas;
- sistema de *backup on demand*;
- sistema de alertas;
- sistema de *scheduler jobs*;
- gerador de gráficos e relatórios quantitativos;
- sistema de tradução automática de instâncias;
- exportador de dados;
- sistema de *log*;

A principal característica do Titan em relação a outras soluções no domínio de CMSs, tais como, Pantaneiro (SANDIM, 2009), Joomla! (JOOMLA, 2012), Drupal (DRUPAL, 2012), PLONE (PLONE, 2013), OpenCMS (OPENCMS, 2013) e Typo3 (TYPO3, 2013), motivo pela qual optou-se por utilizá-lo, é que além de possibilitar a instanciação de um CMS,

o Titan permite a construção de sistemas complexos e completos por meio de sua arquitetura extensível, ou seja, o Titan pode ser estendido também para atender outras famílias de produtos, fornecendo maiores possibilidades de extensão. Por exemplo, no contexto de portais e-Gov, quando uma instância de um portal necessitar de alguma funcionalidade mais complexa, por exemplo, a emissão e validação de certidões, alvarás e licenças, marcação de consultas médicas, matrículas escolares, entre outras, tais funcionalidades podem ser implementadas utilizando o *framework* Titan, e uma vez implementadas, podem ser reutilizadas em qualquer instância do *framework* com a utilização do Titan Extensions (Seção 4.3). Já em outras soluções, uma funcionalidade desse tipo, quando é possível ser desenvolvida, é feita de forma limitada ou o desenvolvimento é complexo e custoso, necessitando alterar as partes fixas (*frozen-spots*) do *framework*.

Um CMS criado a partir do *framework* Titan é responsável apenas pelo gerenciamento do conteúdo de um portal, e desta forma não provê funcionalidades para criação do projeto de interface (*design* gráfico) do *Front-End*. Diante dessa carência, é proposto neste trabalho a ferramenta Titan FrontEnd, que proverá meios para a criação do projeto de interface gráfica por meio do uso de *templates* de *layouts* renderizados através de um motor de *templates* (*Template Engine*).

2.6 Geradores de Aplicação

A automação no processo de construção de artefatos de software é uma atividade que pode diminuir o tempo, custo e esforço no desenvolvimento de aplicações (SHIMABUKURO, 2006). Uma das técnicas que podem ser utilizadas na automação da geração de artefatos de software são os geradores de aplicação. Com o uso de geradores, o engenheiro de aplicação insere apenas as informações sobre “o que” deve ser feito, e a ferramenta decide “como” as informações devem ser transformadas em código fonte (CLEAVELAND, 2001).

Geradores de aplicação são sistemas de software que transformam especificações em uma aplicação (CLEAVELAND, 1988). Segundo Czarnecki e Eisenercker (2002), um gerador de aplicação é uma ferramenta que aceita como entrada uma especificação de uma tarefa ou problema a ser selecionado e gera automaticamente os artefatos necessários para solucionar o problema. O termo gerador de aplicações pode assumir diferentes significados, tais como: compiladores, pré-processadores, meta-funções que geram classes e procedimentos, *wizards* e geradores de código. Um *wizard*, por exemplo, é um programa

gráfico que recebe uma especificação em alto nível e transforma essa informação em software (CZARNECKI e EISENERCKER, 2002).

Os geradores de aplicação podem ajudar na construção de múltiplos produtos de uma família com mais facilidade do que a maneira de implementação tradicional. Além do código, os geradores podem produzir documentação do usuário e do software, casos de teste, diagramas e figuras (CLEAVELAND, 2001).

Segundo Czarnecki e Eisenercker (2002), um gerador de aplicação contempla as seguintes tarefas:

- Realizar a validação da especificação de entrada e relatar erros ou avisos de inconsistências;
- Completar a especificação utilizando as configurações padrão, caso necessário;
- Realizar otimizações; e
- Gerar artefatos de software.

Existem duas possibilidades para se construir um gerador de aplicação: construir um compilador ou um compositor. Construir um compilador significa criar um analisador léxico, sintático e semântico para uma linguagem. Construir um compositor significa criar um projeto de software, derivar um conjunto de gabaritos (*templates*) a partir desse projeto, criar um mapeamento entre a especificação e esses gabaritos, para em seguida uma ferramenta utilizar as especificações junto com os gabaritos para gerar artefatos (WEISS e LAI, 1999).

A utilização de geradores durante a engenharia de aplicação torna o processo de codificação mais ágil e menos suscetível a erros humanos do que o processo tradicional, ou seja, os geradores podem produzir código de forma sistemática e mais segura em relação aos métodos tradicionais de programação (CLEAVELAND, 2001) (CZARNECKI e EISENERCKER, 2002).

Neste trabalho foi desenvolvido o gerador de aplicações Titan FrontEnd, que de acordo com as definições de Czarnecki e Eisenercker (2002) e Weiss e Lai (1999), pode ser classificado como um *wizard* e um compositor.

2.6.1 Titan Architect

O Titan Architect (NACER, 2009) é um *wizard*, gerador de aplicações, que permite a

instanciação automática de sistemas gerenciadores de conteúdo através do Titan Framework. Esse gerador de aplicações é baseado em passos, cujo objetivo é criar para o engenheiro de aplicação os arquivos XML, PHP e SQL, sendo parametrizados no Titan Framework para a criação de novas aplicações.

O Titan Architect foi construído como um componente do Titan Framework, desta forma pode ser adicionado em qualquer instância do *framework*, sendo responsável pela geração automática de arquivos XML e criação das tabelas no banco de dados. Portanto, qualquer instância do *framework* pode ser um gerador de aplicações, bastando apenas que possua o componente Titan Architect. O Titan Architect, por ser um componente do *framework*, utiliza o banco de dados do próprio *framework*. Além das tabelas nativas do *framework*, o Titan Architect adiciona uma nova tabela para gerenciar as aplicações geradas por ele, desta forma o engenheiro de aplicação pode reabrir posteriormente qualquer projeto criado no Titan Architect (NACER, 2009).

Conforme ilustrado na Figura 2.2, a configuração de novas instâncias do domínio é realizada de forma interativa, através de quatro passos de configuração. No primeiro passo são definidas as informações e configurações principais e visuais da aplicação. No segundo passo são configurados os atores da aplicação, seus dados e como será seu acesso ao sistema. No terceiro passo são configuradas as funcionalidades do sistema, ou seja, são definidos os parâmetros que serão passados aos componentes. O último passo efetua a geração e download da nova aplicação (CARROMEU at al., 2010).

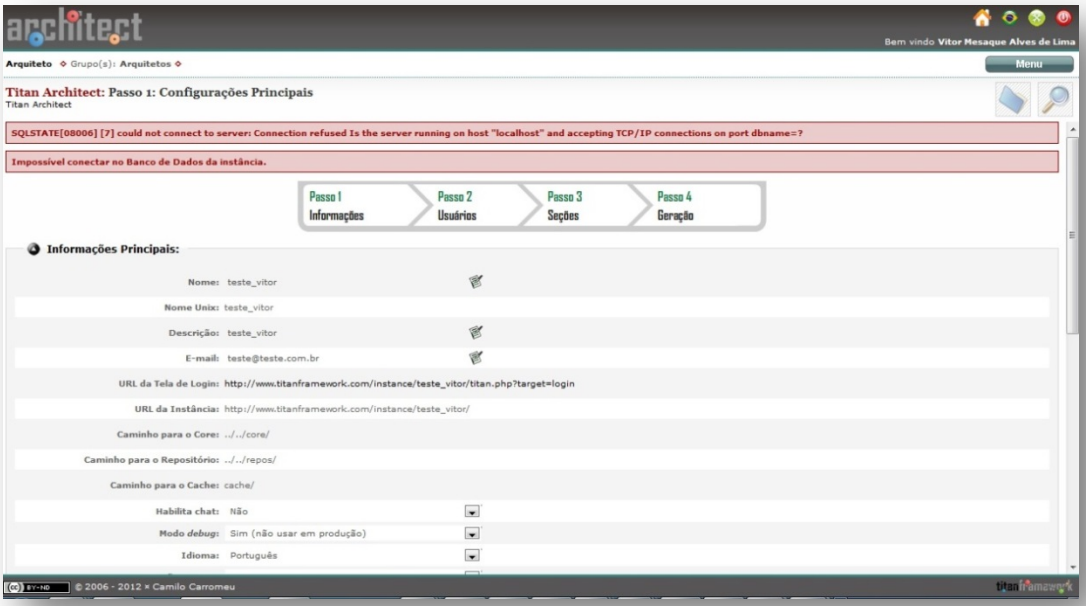

Figura 2.2: Titan Architect (NACER, 2009).

O uso do Titan Architect elimina a necessidade programação, porém a parametrização das variabilidades é limitada (NACER, 2009). Já o Titan Framework possibilita a configuração de regras de negócios e parâmetros da nova aplicação por meio da edição da linguagem de marcação de entrada e, caso não seja suficiente, permite o desenvolvimento de novos artefatos de software.

Com a utilização do Titan Framework e do Titan Architect é possível instanciar de forma automatizada o *Back-End* de um portal, porém o *Front-End*, até então, era construído de forma totalmente manual, dispendendo tempo e esforço em trabalhos repetitivos, uma vez que todo o desenvolvimento do *Back-End* teria que ser refeito em outra abstração, agora para os usuários finais. Na Seção 4.4 do Capítulo 4 é apresentado o gerador de aplicações Titan FrontEnd, proposto neste trabalho, que será responsável pela geração automática do *Front-End* da aplicação criada a partir da instanciação do Titan Framework.

2.7 Linha de Produtos de Software

A abordagem de LPS ganhou uma crescente atenção nos últimos anos devido à competitividade na área de desenvolvimento de software. As considerações econômicas de empresas de software, tais como redução de custos e tempo de entrega (*time-to-market)*, assim como melhoria na qualidade, motivam a transição de desenvolvimento de produtos únicos para a abordagem de linha de produtos, na qual os produtos são desenvolvidos numa

perspectiva de reutilização em larga escala (GRISS, 2001) (POHL *et al*., 2005) (LINDEN *et al*., 2007). A abordagem de LPS é cada vez mais reconhecida na indústria e no governo como sendo uma grande e importante estratégia no desenvolvimento de software. Estudos indicam que, se três ou mais sistemas com certo grau de funcionalidades comuns são desenvolvidos, o desenvolvimento de uma LPS se torna significativamente mais rentável do que o desenvolvimento de sistemas individuais a partir do zero (GOMAA, 2004).

Uma LPS é um conjunto de sistemas de software, que compartilham um conjunto de funcionalidades comuns e que satisfazem as necessidades de um segmento de mercado particular ou missão e que são desenvolvidos de forma sistemática, a partir de um conjunto de ativos[6] base (CLEMENTS e NORTHROP, 2002). Segundo Pohl *et al*. (2005), uma LPS pode ser definida como um conjunto de aplicações desenvolvidas utilizando plataformas e customização em massa. Uma LPS permite gerar um conjunto de aplicações similares pertencentes a um mesmo domínio, desenvolvidas a partir de uma arquitetura genérica de LPS formada por artefatos de software e de um conjunto de componentes que compõem a arquitetura. O paradigma de utilização de linha de produtos surge como uma proposta de construção e reutilização sistemática de software baseada em uma família de produtos. Segundo Weiss e Lai (1999), uma família de produtos é um conjunto de itens que possuem aspectos comuns e variabilidades previsíveis.

Segundo Clements e Northrop (2002), um processo de desenvolvimento de LPS é formado por três atividades essenciais, conforme ilustrado na Figura 2.3:

- Engenharia do domínio ou desenvolvimento do núcleo de artefatos: compreende as atividades de análise de domínio para definir o contexto da LPS, a arquitetura e os componentes reutilizáveis, permitindo determinar o plano de produção da aplicação;

- Engenharia de aplicação ou desenvolvimento do produto: compreende o modelo do domínio que serve como base para identificar os requisitos do cliente, e um modelo de instanciação da arquitetura da LPS para especificar os membros da família de produtos a partir de um conjunto de componentes reutilizáveis; e

[6] Ativos são artefatos de qualquer natureza, gerados em qualquer momento do processo de desenvolvimento (Ezran, 2002).

- Gerenciamento da LPS: define regras de gestão implícitas durante todo o processo a fim de garantir o sucesso na construção e manutenção da LPS.

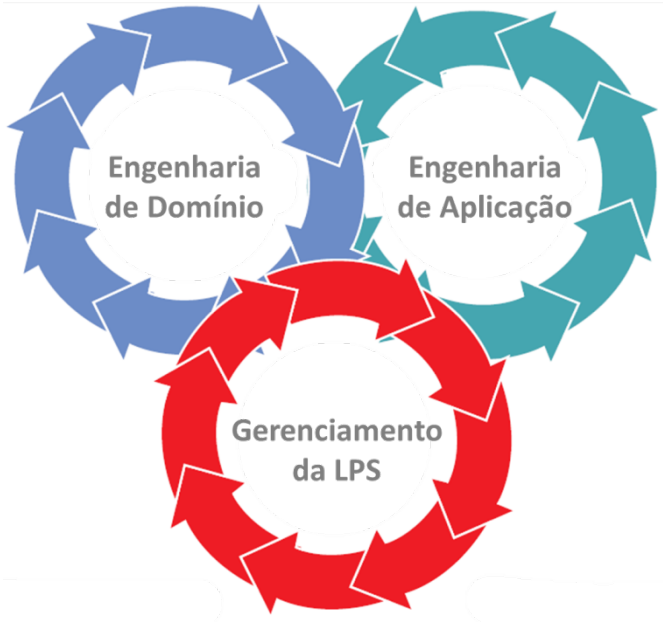

Figura 2.3: Atividades essenciais de uma LPS (adaptado de CLEMENTS e NORTHROP, 2002, p. 30).

2.7.1 Modelagem de Variabilidades em LPS

A modelagem de variabilidades é uma atividade essencial da Engenharia de Domínio no desenvolvimento de uma LPS, pois a partir dela é possível identificar os aspectos variáveis de cada produto da linha, dando suporte à fase de Engenharia de Aplicação da LPS.

As variabilidades são descritas por meio de pontos de variação e variantes. Um ponto de variação é um local específico, em um artefato de uma LPS, no qual uma decisão de projeto está relacionada. Cada ponto de variação está associado a um conjunto de variantes que correspondem às alternativas de modelagem para resolver a variabilidade (Linden *et al*., 2007) (Pohl *et al*., 2005).

Segundo Gomaa (2004), a abordagem amplamente difundida pela comunidade científica para modelagem de variabilidade de requisitos de software é o modelo de *features* (característica). *Feature* é um conceito importante em LPS, pois indicam requisitos ou características reusáveis de uma linha de produtos. Um requisito é usado para representar uma necessidade da LPS e, portanto, pelo menos um membro da linha de produtos deve ser capaz de satisfazê-lo. Uma vez que a LPS capturou esse requisito, ele é identificado como uma *feature* fornecida pela LPS (GOMAA, 2004).

O modelo de *features* surgiu na fase de análise do domínio da abordagem FODA (*Feature-Oriented Domain Analysis*) (COHEIN et. al, 1990) e tem como objetivo representar de forma hierárquica o relacionamento entre as características de um domínio de aplicação. Na Figura 2.4, é apresentado um exemplo de modelo de *features* para um carro, que compreende um conjunto de características representadas por retângulos e relacionamentos entre eles definidas por linhas hierarquicamente distribuídas. As *features* são identificadas por um nome e classificadas em raiz, obrigatórios e opcionais. Existe somente uma *feature* raiz que é o nó principal da estrutura, por exemplo, a *feature* "carro". Uma *feature* obrigatória é representada por um círculo preenchido na parte superior, e a opcional é representada por um círculo vazio. A notação de um triângulo vazio representa a abstração do tipo "um de", indicando que somente uma característica é utilizada, e o triângulo preenchido define o tipo "mais de um", quando são utilizadas uma ou mais *features*.

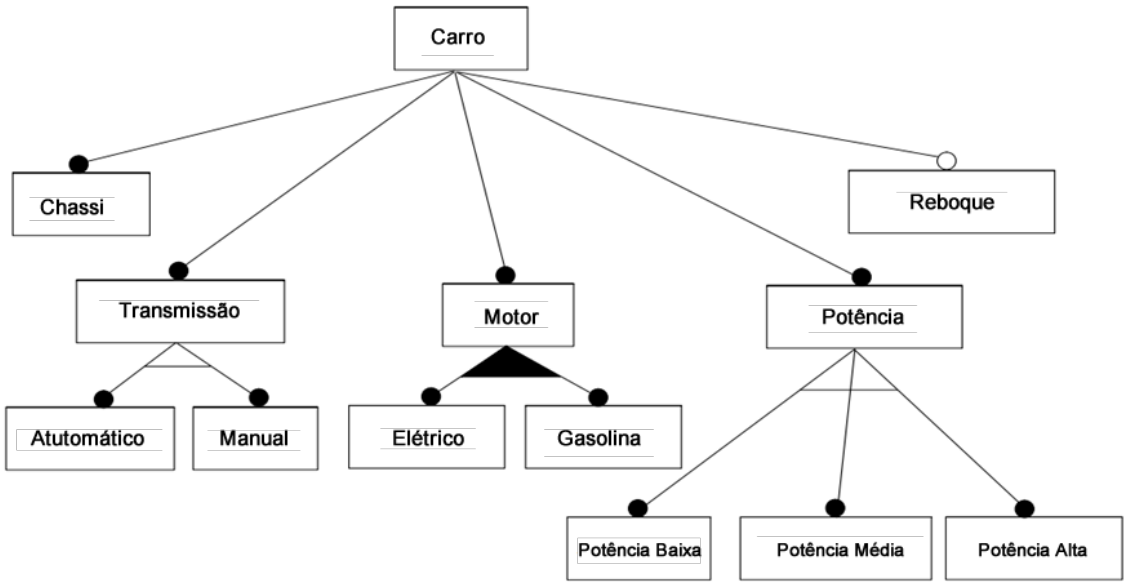

Figura 2.4: Exemplo de modelo de *features* (KLINT e VAN DEURSEN, 2002, p.3).

Embora o modelo de *features* auxilie na identificação dos pontos variáveis da LPS, somente a sua utilização não fornece diretivas suficientes para as etapas de projeto e implantação de uma LPS. A seguir são apresentadas algumas abordagens existentes na literatura que apoiam o processo de desenvolvimento de LPS.

2.7.2 Abordagens para Desenvolvimento de LPS

Diversas abordagens para o desenvolvimento de LPS são propostas na literatura, a seguir é apresentada uma visão geral sobre algumas técnicas:

- FAST (*Family-oriented Abstraction, Specification, and Translation*) (Weiss e Lai, 1999): é uma alternativa ao processo de desenvolvimento de software tradicional. É aplicável em qualquer contexto que possui múltiplas versões de um produto que compartilha um conjunto de atributos, assim como comportamento, interface e código comuns;

- PuLSE (*Product Line Software Engineering*) (Bayer *et al*., 1999): desenvolvida pelo *Institute Experimental Software Engineering,* define uma metodologia de construção e utilização de LPS que inclui as diferentes fases de desenvolvimento: iniciação, construção de infraestrutura, utilização da infraestrutura, evolução e gerenciamento;

- PLP (*Product Line Practice*) (SEI, 2012): é uma iniciativa, desenvolvida pelo *Software Engineering Institute* (SEI), proposta por Clements e Northrop (2002). Não contém um método de construção de LPS, mas se destaca por caracterizar e uniformizar os vários conceitos de LPS. A estratégia é utilizada para implementação de LPS que propõe o uso de práticas de negócio e aquisição de tecnologias para conduzir as deficiências existentes em projeto de LPS em relação a requisitos, arquitetura, modelos, descrições de processos, documentação, testes, programações, orçamentos e outros artefatos reutilizáveis;

- PLUS (*Product Line UML-Based Software Engineering*) (GOMAA, 2004): estende os métodos de modelagem baseados em UML (*Unified Modeling Language*) usados no desenvolvimento de sistemas únicos para LPS. Destaca-se por ser compatível com o RUP (*Rational Unified Process*) e com o modelo em espiral. O processo utilizado no PLUS é denominado ESPLEP (*Evolutionary Software Product Line Engineering Process*), utiliza o modelo de *features* e gera um projeto baseado em componentes.

Dentre as abordagens LPS citadas, a PLUS foi utilizada no processo de LPS proposto por Carromeu *et al*. (2010) e será utilizada neste trabalho.

2.7.3 Abordagem PLUS

A abordagem PLUS utiliza o processo ESPLEP, conforme ilustrado na Figura 2.5, apresentando dois ciclos de vida: Engenharia da LPS (chamada também de Engenharia de Domínio) e Engenharia da Aplicação. Os dois ciclos são desenvolvidos de forma iterativa e incremental, permitindo o desenvolvimento de artefatos reutilizáveis que são armazenados no Repositório da LPS. Destaca-se que o processo ESPLEP elimina a distinção entre desenvolvimento e manutenção, pois as atividades de correção de erros, adaptação e prevenção são novas iterações do ciclo da Engenharia da LPS (GOMAA, 2004).

No ciclo da Engenharia da LPS, a partir dos requisitos levantados, os artefatos desenvolvidos são armazenados no repositório da LPS, são eles: modelo de caso de uso, modelo de *features*, modelo de análise, arquitetura da LPS e componentes reusáveis. Na Engenharia da Aplicação, a partir dos requisitos do engenheiro da aplicação e dos artefatos do Repositório da LPS é desenvolvida uma aplicação, ou seja, uma nova instância derivada da LPS utilizada pelos clientes. O modelo de casos de uso da LPS é adaptado dando origem ao modelo de casos de uso da aplicação; o modelo de análise da LPS é adaptado originando o modelo de análise da aplicação e a arquitetura da LPS é adaptada para derivar a arquitetura da aplicação. Dessa forma, a partir da arquitetura da aplicação e dos componentes reutilizáveis do repositório, a aplicação é construída.

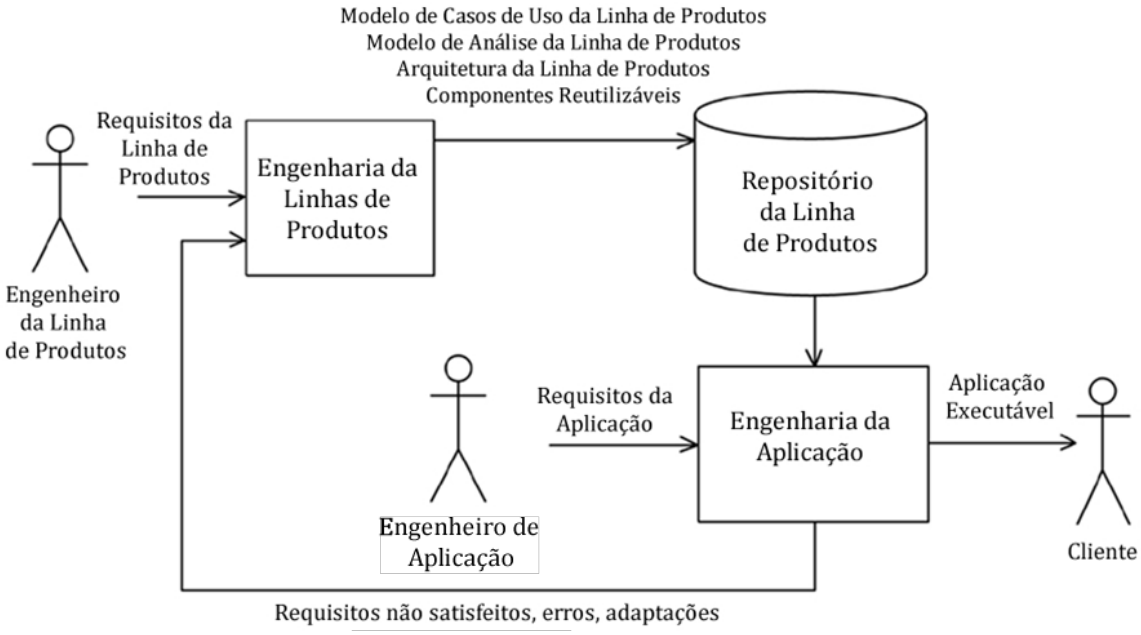

Figura 2.5: Ciclos do Processo ESPLEP (GOMAA, 2004, p. 45).

2.7.4 LPS no domínio e-Gov

O processo de LPS no domínio e-Gov proposto por Carromeu *et al*. (2010), utilizado neste

trabalho, é baseado na abordagem PLUS (GOMAA, 2004). Segundo Carromeu *et al*. (2010), a PLUS foi escolhida por ser evolutiva, compatível com o RUP (*Rational Unified Process*) e com o modelo em espiral (GOMAA, 2004) (BOEHM, 1988), utiliza modelos simples e gera um repositório de artefatos reutilizáveis.

Segundo Carromeu *et al*. (2010), o modelo de *features* e a arquitetura inicial da LPS foram concebidas a partir de padrões abstraídos de diversas WebApps do domínio e-Gov. Segundo Fayad *et al*. (1999), os padrões descrevem soluções recorrentes aprovadas durante seu tempo de uso, permitindo que sejam evidenciadas características comuns que podem ser reutilizadas em novos projetos e em diferentes níveis de abstração, não somente de código, mas de processo, de arquitetura, de análise, de projeto, de programação, dentre outros.

Na Figura 2.6 é apresentada a criação da engenharia da LPS e da versão inicial do repositório de componentes a partir da abstração dos padrões identificados e na Figura 2.7 é apresentado o processo ESPLEP da LPS para o domínio e-Gov.

Contudo, para sistematizar o processo de reutilização, implementar os componentes e permitir a geração de novas WebApps, é fundamental o uso de ferramentas automatizadas. Inicialmente a plataforma proposta por Carromeu *et al*. (2010) era composta por duas ferramentas: o *framework* Titan e o gerador de aplicações Titan Architec.

Após diversas instanciações da LPS no domínio de portais e-Gov, identificou-se a necessidade de automatizar partes do processo, com o objetivo de aumentar o nível de reúso e acessibilidade das aplicações geradas. Assim, este trabalho propõe uma plataforma para a automação da LPS composta por dois *frameworks* (Titan Framework e Zend Framework), dois geradores de código (Titan Architect e Titan FrontEnd) e um repositório de reúso (Titan Extensions). A arquitetura da plataforma LPS e as ferramentas mencionadas são detalhadas nas seções subsequentes deste trabalho.

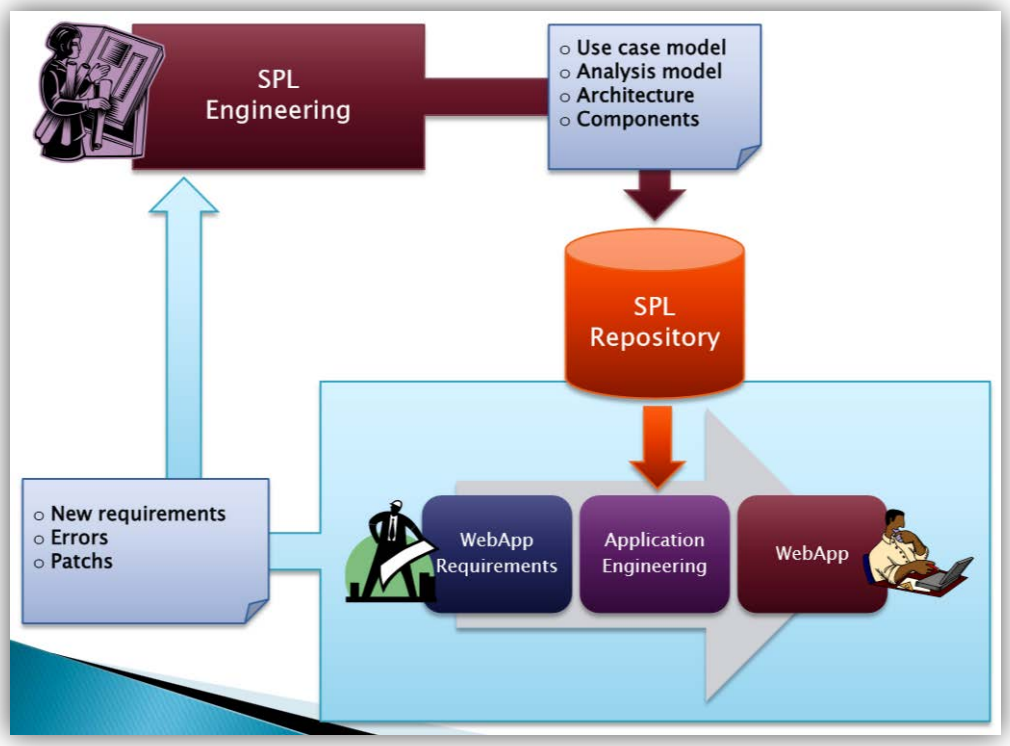

Figura 2.6: Abstração dos padrões do domínio e especificação da Engenharia da LPS (CARROMEU *et al.*, 2010).

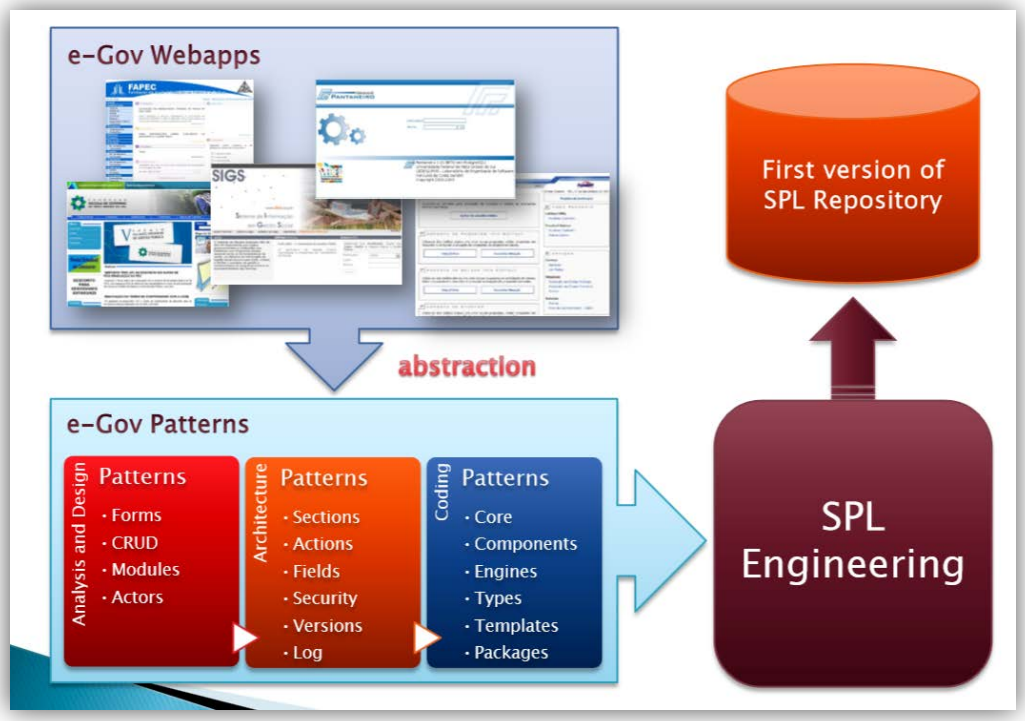

Figura 2.7: Processo ESPLEP da LPS para o domínio e-Gov (CARROMEU *et al.*, 2010).

2.8 Repositórios de Reúso

É conhecido que as organizações estão sempre em busca de alcançar o desenvolvimento de produtos cada vez mais rápidos, baratos e melhores. Neste sentido, a importância do reúso de software tem sido evidenciada e tornou-se uma abordagem indispensável para atingir esses objetivos. No entanto, o sucesso das soluções de reúso de software depende de uma aplicação eficaz dos aspectos técnicos e não técnicos (ALMEIDA *et al*., 2004). Os aspectos técnicos incluem, entre outras coisas, a criação de um repositório de reúso de software que dá suporte aos engenheiros de software e outros usuários no processo de desenvolvimento de software *para* e *com* reúso (BURÉGIO *et al*., 2007).

Segundo D'Souza (1999), ativos de software (*assets*) são quaisquer artefatos que podem ser utilizados em mais de um projeto e que podem ser armazenados em um repositório de reúso. Mili *et al*. (1998) afirmam que um repositório de reúso é uma coleção de ativos reutilizáveis e provê funcionalidades, tal como mecanismos de pesquisa e recuperação. Segundo Niranjan e Rao (2011), a utilização de um repositório de reúso se faz necessário para atingir uma grande variedade de componentes de alta qualidade, organizados de uma maneira eficiente utilizando técnicas de classificação e deve ser capaz de recuperar os melhores componentes que correspondem aos requisitos do utilizador. O utilizador deve acessar os componentes apropriados com precisão e rapidez e, se necessário, deve ser capaz de modificá-los.

Em qualquer processo de desenvolvimento baseado em reúso, é significativo distinguir pelo menos duas atividades: o desenvolvimento para reúso e o desenvolvimento com reúso. De acordo com Apperly (2001), tais atividades caracterizam um processo *produção-gerência-consumo*. Nesse processo, um repositório deve conter serviços que satisfaçam os interesses de três partes interessadas: produtores, consumidores e gerentes. A fim de satisfazer tais interesses e apoiar o reúso de software de forma sistemática, um repositório pode assumir as seguintes funções (ALMEIDA *et al*., 2004):

- *Repositório como um meio de comunicação entre as partes interessadas*. Representa um canal de distribuição de ativos e deve servir como um canal de comunicação de duas vias: o desenvolvimento para o reúso (produção) e o desenvolvimento com reúso (consumo);
- *Repositório como um suporte ao gerenciamento.* Um repositório pode ajudar as organizações a controlar a sua utilização de ativos, com o objetivo de alinhar iniciativas, projetos e medir o impacto nos negócios do reúso;

- *Repositório como um promotor de reúso*. Repositórios podem disseminar a cultura de reúso dentro da organização, fornecendo visibilidade em seus ativos e disseminando notícias sobre as iniciativas de reúso;
- *Repositório como uma garantia de qualidade.* Repositórios podem ajudar no alcance do objetivo da organização de manter a qualidade em todo o ciclo de vida de desenvolvimento de software, fornecendo ativos de software provados e apoiando um processo de certificação eficiente de ativos.

Burégio *et al*. (2008) descrevem dezessete requisitos básicos que um repositório de reúso deve atender, conforme apresentados resumidamente no Quadro 2.1.

Quadro 2.1: Lista de Requisitos de Repositórios de Reúso (BURÉGIO *et al*., 2008).

Requisitos	Descrição
1. Descrição do Ativo	Possibilitar o cadastro de metadados de ativos
2. Inserção	Possibilitar a inserção de ativos
3. Publicar Especificação	Permitir apenas a publicação da especificação do ativo
4. Navegação	Possibilitar a navegação pelos ativos por categoria
5. Busca	Fornecer mecanismos de busca
6. Geração de Relatórios	Possibilitar a geração de relatórios sobre a utilização do repositório
7. Notificação de usuários	Registrar preferências de usuários e notifica-los sobre os interesses comuns
8. Serviços de Manutenção	Implementar serviços administrativos, tais como, manter classificadores de ativos, tipos de artefatos de ativos, hierarquias de catálogos.
9. Controle de Versões	Realizar controle de versões dos ativos
10. Gerência de Dependência	Possibilitar o gerenciamento de dependência
11. Serviços de *Feedback*	Fornecer serviços de *feedback* sobre a utilização de ativos
12. Serviços de Anúncios	Possibilitar o envio de anúncios de serviços
13. Múltipla Fonte de Ativos	Suportar ativos de diversos repositórios
14. Processo de Certificação	Apoiar um processo de certificação
15. Métricas	Coletar métricas de reúso
16. Controle de Acesso	Permitir controle de acesso
17. Controle de Mudanças	Permitir controle de mudanças

2.8.1 Repositório do Titan Framework

O Titan Framework possui em sua arquitetura artefatos de software reutilizáveis como, por exemplo, os componentes. Os componentes do Titan são armazenados no repositório global e quando não conseguem atender a demanda de uma determinada funcionalidade nova, podem ser criados novos componentes ou estendidos os existentes.

O gerenciamento dos ativos disponíveis no repositório do Titan é feito pela ferramenta de gerenciamento de configuração Subversion (APACHE, 2012). Com a utilização de uma ferramenta de gerenciamento de configuração é possível ter controle de acesso, mudança e versões, por outro lado, não é possível a definição, publicação e busca de ativos reutilizáveis de uma maneira eficiente e organizada.

Da forma como está implementado o repositório atualmente, quando é necessário o desenvolvimento de um novo ativo para uma instância específica do Titan, o mesmo é tratado como um ativo local, e não fica disponível para ser utilizado em outras instâncias. Apenas um grupo de usuários administradores do *framework* tem permissão para adicionar novos ativos ao repositório, impossibilitando que outros usuários os disponibilizem, impedindo dessa forma a colaboração de equipes de desenvolvimento distintas.

Deste modo, visando otimizar e maximizar o reúso de novos artefatos de software desenvolvidos como extensões do *framework* Titan, é proposto neste trabalho o repositório de reúso Titan Extensions, que consiste em um ambiente público de desenvolvimento colaborativo de ativos reutilizáveis. Estes ativos são denominados de *extensions* (extensões).

2.9 Considerações Finais

A reusabilidade de software promete aumentar a qualidade e a produtividade, reduzindo o *time-to-market* e os custos no processo de desenvolvimento de software. Porém, para alcançar tais benefícios é necessário um processo de reutilização sistemático, motivando a utilização de abordagens como a LPS. A abordagem de LPS por sua vez estimula a utilização de ferramentas que automatizem a linha de produção, como os *frameworks* e geradores de aplicação. Os repositórios de reúso auxiliam nesse processo, facilitando a identificação e recuperação dos ativos reutilizáveis.

Capítulo 3

Investigação do Problema e Mapeamento Sistemático

3.1 Considerações Iniciais

Neste capítulo são apresentados os resultados obtidos em pesquisas realizadas durante o período de desenvolvimento deste trabalho, com o objetivo de investigar e comprovar a necessidade de soluções em acessibilidade para portais e-Gov, bem como as soluções e dificuldades encontradas na literatura no contexto deste trabalho.

Na Seção 3.2 é apresentada uma pesquisa realizada no estado de Mato Grosso do Sul com objetivo de evidenciar o problema concernente à acessibilidade em portais e-Gov na prática. Na Seção 3.3 é apresentado o mapeamento sistemático realizado no âmbito deste trabalho, com o intuito de analisar as pesquisas disponíveis no contexto de CMSs acessíveis, produto final instanciado pela plataforma de LPS proposta neste trabalho. As etapas de planejamento, condução, seleção e análise do mapeamento sistemático realizado também estão documentadas na Seção 3.3. Por fim, na Seção 3.4 são apresentadas as considerações finais do capítulo.

3.2 Investigação do Problema na Prática

De acordo com o censo publicado em 2010 pelo CGI.br e NIC.br, o Brasil possui 11.856 portais e sítios e-Gov e um total de 6.331.256 páginas HTML (*HyperText Markup Language*). Apenas 2% estão em conformidade com os padrões de acessibilidade e só 3% disponibilizam a opção de navegação em outros idiomas. Os dados ainda mostram que mais de 30% dos portais e sítios e-Gov brasileiros ainda utilizam tecnologias proprietárias (CGI.br e NIC.br, 2010). Segundo os dados quantitativos do Instituto Brasileiro de Geografia e Estatística (IBGE), o Brasil possui 5.565 municípios, e 40% desses ainda não apresentam portais e-Gov (IBGE, 2010). Dos 60% (3.339) que possuem portais, apenas 780 oferecem uma página, que além de informativa e interativa, possibilita o uso de serviços, como concessão de alvará e matrícula escolar.

Para fins de comprovação da necessidade de implantações de melhoria no desenvolvimento de portais para o domínio e-Gov em conformidade com as diretrizes de acessibilidade do e-MAG e visando atingir um nível maior de detalhamento quanto aos erros de acessibilidade mais infligidos, foi realizada uma pesquisa no estado de Mato Grosso do Sul no âmbito deste trabalho, abrangendo os setenta e nove municípios do estado e investigando quais prefeituras possuem um portal e quais deles respeitam os requisitos de acessibilidade.

3.2.1 Metodologia

Para a realização da pesquisa foi desenvolvido um *Web Crawler* [7] (CHEONG, 1996) (HEATON, 2002), que recebe como entrada os nomes dos municípios e verifica de forma automática se a prefeitura do município possui um portal ou não. O programa assume uma formatação padronizada das URL's dos portais das prefeituras (*municipio.ms.gov.br*). Com isso obteve-se sucesso para 94,9% das entradas, sendo que apenas 5,1% foram verificadas manualmente.

Após a identificação dos portais disponíveis, a acessibilidade de todos foi validada pela ferramenta ASES (Avaliador e Simulador de Acessibilidade de Sítios) (ASES, 2012). Apenas as páginas iniciais de cada portal foram validadas e os erros encontrados foram classificados em três níveis de prioridade de acordo com o e-MAG, conforme mostrados no Quadro 3.2. Os erros detectados também foram mapeados em correspondência com as diretrizes do e-MAG, detalhados no Apêndice A.

Quadro 3.2: Níveis de Prioridades do ASES.

	Prioridade 1	**Prioridade 2**	**Prioridade 3**
Pontos que os criadores de conteúdo na Web:	**devem** satisfazer inteiramente	**deveriam** satisfazer	**podem** satisfazer
Com relação ao acesso a documentos disponíveis na Web, A satisfação desse tipo de pontos :	é um requisito básico para determinados grupos	promoverá a remoção de barreiras significativas	melhora o acesso

[7] *Web crawler* é um programa de computador que navega pela World Wide Web de uma forma metódica e automatizada. Um *Web crawler* é um tipo de robô de Internet ou agente de software. Em geral, ele visita uma lista de URL's para obter informações específicas. Outros termos para Web crawlers são indexadores automáticos, *bots*, *web spiders*, *Web robot*, ou *Web scutter* (CHEONG, 1996) (HEATON, 2002).

3.2.2 Análise

A Tabela 3.1 mostra os resultados da pesquisa realizada, evidenciando que dos setenta e nove municípios do Estado do Mato Grosso do Sul, 94% possuem portais, enquanto 6% do total não possuem. Dentre os que possuem, 97% estão disponíveis e 3% estão *off-line*.

Tabela 3.1: Panorama dos portais das prefeituras do MS.

	Sites	**Percentagem**
Existe	74	94%
Não Existe	5	6%
Total	79	100%

Fonte: elaborado pelo autor a partir dos dados coletados.

Quando os portais foram validados pelo ASES, notou-se que o erro mais comum são os erros da categoria de Prioridade 2 (P2), com 2368 erros, os erros de Prioridade 1 (P1) foram detectados 2100 vezes e os erros de Prioridade (P3) somaram 21% do total de erros, conforme mostra a Tabela 3.2.

Tabela 3.2 - Erros por categoria de prioridade.

	Número de Erros	**Percentagem**
P1	2100	37%
P2	2368	42%
P3	1157	21%
Total	5625	100%

Fonte: elaborado pelo autor a partir dos dados coletados.

Na avaliação dos portais, concernente aos erros P1, todos os portais das prefeituras apresentaram erros. Além disso, fazendo uma análise detalhada, é possível perceber que o erro mais comum é o *não fornecimento de um equivalente textual a cada imagem*, 98,6% das cidades apresentaram esse tipo de problema. Os sites mais problemáticos, tendo por base essa perspectiva, foram os das cidades de Pedro Gomes (169 erros), Angélica (102 erros), Iguatemi (96 erros), Nioaque (84 erros), Chapadão do Sul (75 erros), Rio Brilhante (75 erros), Itaquiraí (68 erros). Esses resultados podem ser visualizados na Figura 3.8.

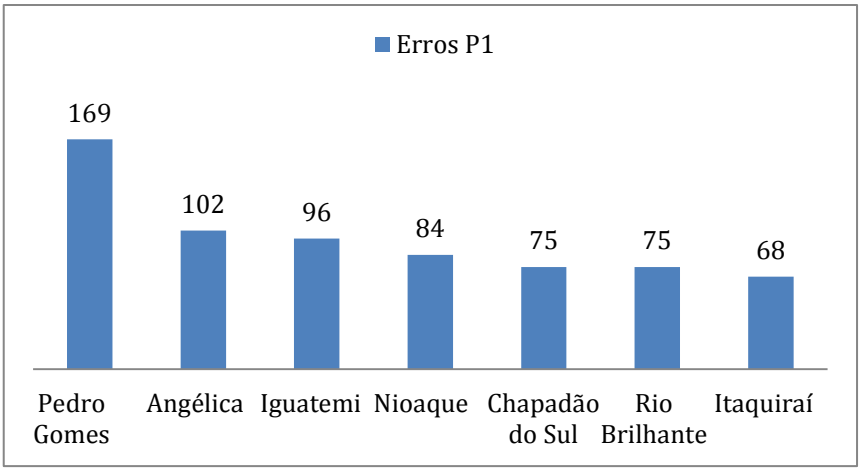

Figura 3.8: Frequência de erros da categoria P1.

Fonte: elaborado pelo autor a partir dos dados coletados.

Dentre os erros da categoria P2 o que mais se destaca é a *utilização de unidades relativas e não absolutas nos valores dos atributos de tabelas, textos, etc*. Esse tipo de problema foi constado em 84,1% dos portais. O portal do município de Chapadão do Sul foi o que apresentou mais erros desse tipo P2, seguido por Rio Brilhante. Os portais que apresentaram a maior quantidade de erros podem ser visualizados na Figura 3.9.

Segundo, ainda, os dados levantados na pesquisa, os erros P3 foram encontrados mais comumente nos portais dos seguintes municípios: Angélica (207 erros), Pedro Gomes (162 erros), Nioaque (146 erros), Santa Rita do Rio Pardo (142 erros), Sonora (91 erros), Cassilândia (80 erros), Ponta Porã (56 erros). Conforme mostra a Figura 3.10.

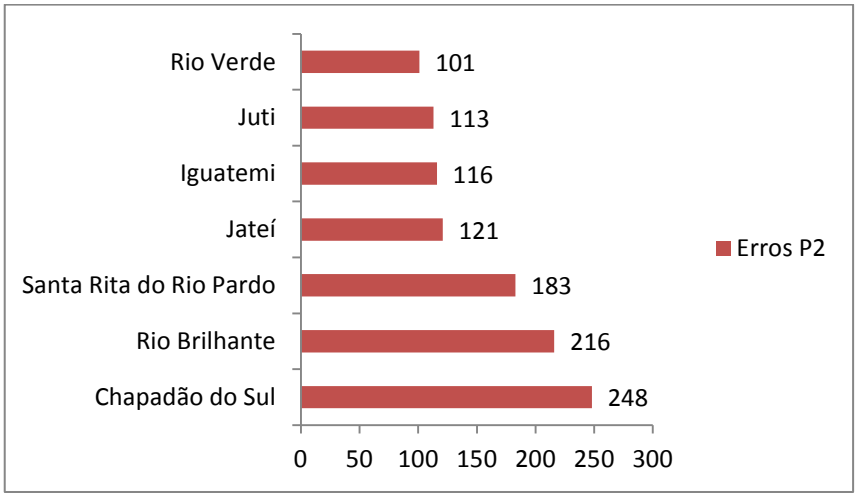

Figura 3.9: Frequência de erros da categoria P2.

Fonte: elaborado pelo autor a partir dos dados coletados.

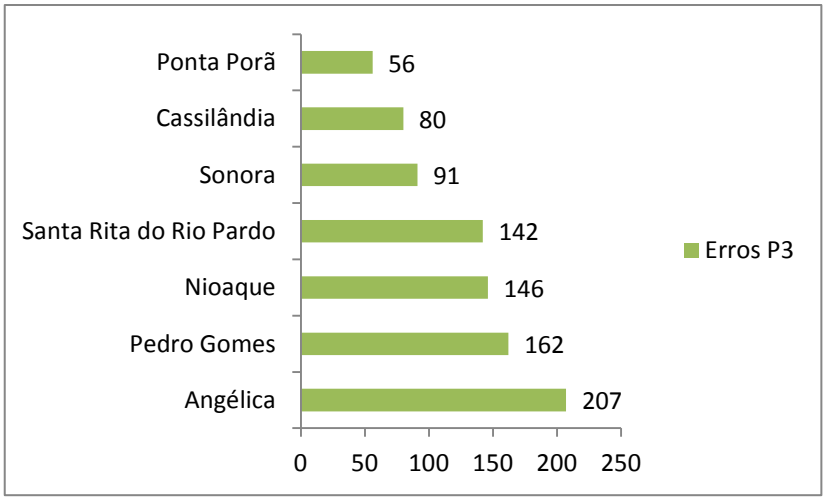

Figura 3.10: Frequência de erros da categoria P3.

Fonte: elaborado pelo autor a partir dos dados coletados.

Todos os municípios apresentam erros e, somando todos os erros das categorias P1, P2 e P3, observou-se que o portal da prefeitura de Pedro Gomes é o que apresenta a pior acessibilidade em relação aos padrões estabelecidos no e-MAG, seguido por Angélica, Santa Rita do Rio Pardo, Chapadão do Sul, Rio Brilhante, Nioaque e Iguatemi.

Em um panorama geral, é possível perceber que os sítios e portais governamentais não estão em conformidade com as diretrizes de acessibilidade estabelecidas pelo Governo Federal. Portanto, os dados evidenciam a necessidade da utilização de práticas e tecnologias que promovam a construção e o desenvolvimento de portais e-Gov acessíveis, bem como apontam para a identificação das fragilidades e infrações de acessibilidade de maiores incidências nos portais analisados.

3.3 Mapeamento Sistemático

Segundo Kitchenham (2004), revisão sistemática é um meio de identificar, avaliar e interpretar toda pesquisa disponível relevante a uma questão de pesquisa em particular, ou área temática ou fenômeno de interesse. A revisão sistemática pode ser efetuada para sumarizar evidências relacionadas a uma tecnologia, identificar qualquer lacuna em pesquisas atuais, sugerir áreas para futuras investigações ou para fornecer um embasamento, a fim de posicionar apropriadamente novas atividades de pesquisa.

Mapeamento sistemático é uma forma de revisão sistemática e tem como objetivo identificar e categorizar a pesquisa disponível em uma área temática específica (KITCHENHAM *et al.*, 2010). Um mapeamento sistemático se difere de uma revisão

sistemática em alguns pontos: i) questão de pesquisa: enquanto no mapeamento sistemático a questão é geral, relacionada a tendências de pesquisa, na revisão sistemática, é específica, relacionada a resultados de estudos empíricos; ii) processo de busca: no mapeamento sistemático é definido por área temática, já na revisão sistemática é definido pela questão de pesquisa (KITCHENHAM *et al.*, 2010).

A revisão sistemática e o mapeamento também diferem entre si quanto aos requisitos da estratégia de busca, o mapeamento é menos rigoroso se apenas as tendências de pesquisa são de interesse. Além disso, não é essencial, no mapeamento, efetuar avaliação da qualidade. Por fim, os resultados esperados em um mapeamento são, normalmente, um conjunto de artigos relacionados a uma área temática e a contagem de artigos em várias categorias, enquanto na revisão sistemática, busca-se resposta a uma questão de pesquisa específica.

O produto final gerado pela plataforma de LPS proposta é um portal e-Gov acessível, sendo um CMS. A fim de identificar e categorizar os trabalhos relacionados à adoção de CMSs no processo de construção de portais acessíveis e técnicas utilizadas para promover uma reutilização sistemática neste processo, tais como as utilizadas neste trabalho, foi realizado um mapeamento sistemático.

3.3.1 Planejamento do Mapeamento Sistemático

De acordo com Kitchenham (2004), o planejamento de um mapeamento sistemático descreve o Protocolo de Revisão que foi estabelecido. É necessário especificar os seguintes elementos: (i) questões de pesquisa; (ii) estratégia de busca; (iii) critérios de inclusão e exclusão; e (iv) extração de dados e métodos de síntese.

1. Questões da Pesquisa:

As seguintes questões de pesquisa (QP) foram formuladas para conduzir a análise dos estudos primários encontrados:

QP1: Quais os CMSs acessíveis propostos recentemente na literatura?
QP2: Quais os CMSs citados nos estudos e quais são acessíveis?
QP3: Qual o tipo de licença dos CMSs citados nos estudos?
QP4: Os CMSs acessíveis são reutilizados sistematicamente utilizando LPS?
QP5: As novas tecnologias como HTML5 e CSS3 têm sido utilizadas em CMSs acesíveis?

QP6: Em quais domínios CMSs têm sido aplicados?

2. Estratégia de Busca:

A estratégia de busca define as fontes de pesquisa (bases de busca) e a *string* de busca que será aplicada para localizar os estudos primários para o mapeamento sistemático. O Quadro 3.3 apresenta as bases selecionadas como fontes de pesquisa para a condução deste mapeamento sistemático. As fontes selecionadas são citadas em Dyba *et al*. (2007) como confiáveis no contexto da Engenharia de Software.

Quadro 3.3: Fontes de Pesquisa para a Mapeamento Sistemático.

Bases de Busca	
ACM Digital Library	http://dl.acm.org/
IEEE Xplore Digital Library	http://ieeexplore.ieee.org/
Scopus	http://www.scopus.com/
ScienceDirect	http://www.sciencedirect.com/
SpringerLink	http://www.springerlink.com/

A construção da *string* de busca foi concebida por meio da identificação dos conceitos chaves que se deseja investigar: Sistema de Gerenciamento de Conteúdo e Acessibilidade. A partir disso, foram selecionados os termos utilizados na condução do mapeamento sistemático.

- Termos para: **Sistema de Gerenciamento de Conteúdo**:
    - *Content Management System*
    - *CMS*
- Termos para: **Acessibilidade**:
    - *Accessibility*
    - *Accessible*

Após a identificação dos termos apresentados, a string de busca *(“Content Management System”) AND (“Accessibility” OR “Accessible”)* foi construída para a execução do mapeamento sistemático proposto.

3. Critérios de Inclusão e Exclusão

Os Critérios de Inclusão (CI) foram escolhidos de forma a permitir que estudos primários relevantes sejam incluídos na pesquisa e os Critérios de Exclusão (CE) foram identificados com o objetivo de descartar estudos primários irrelevantes no contexto deste mapeamento. Os critérios selecionados são descritos no **Erro! Fonte de referência não encontrada.** e Quadro 3.4, respectivamente.

Critérios de Inclusão	
CI1	O estudo apresenta um CMS acessível.
CI2	O estudo apresenta um mecanismo ou abordagem para melhorar a acessibilidade em CMSs.
CI3	O estudo mostra a utilização de um CMS para obter ganhos na acessibilidade.
CI4	O estudo apresenta uma abordagem de desenvolvimento de um CMS acessível.
CI5	O estudo apresenta uma avaliação sobre acessibilidade em CMS.

Quadro 3.4: Critérios de Exclusão (CE)

Critérios de Exclusão	
CE1	O estudo não está em inglês ou em português;
CE2	O estudo está diretamente relacionado a outro estudo primário do mesmo autor;
CE3	O estudo já foi selecionado por meio de outra fonte.
CE4	O estudo não contempla satisfatoriamente todos os itens de inclusão.
CE5	O estudo foi publicado fora do período de 2007 a 2013.
CE6	O trabalho não foi publicado em anais de eventos, em revistas, como relatórios técnicos, capítulos de livros, teses ou dissertações.
CE7	O trabalho é um estudo secundário, ou seja, o trabalho apresenta uma revisão da literatura, um survey ou uma revisão sistemática.

4. Extração de Dados e Métodos de Síntese

Para a extração de dados, serão construídas tabelas relacionadas a cada questão de pesquisa, visando sintetizar resultados para facilitar as conclusões.

3.3.2 Condução da Busca

As buscas foram executadas com base na *string* de busca definida no protocolo do mapeamento sistemático. Adicionalmente, a *string* precisou ser adaptada à sintaxe padrão de cada base, sem divergir com o protocolo da revisão. Um guia elaborado por Landre (2012), detalhando as especificidades de busca em cada fonte de pesquisa, foi utilizado no processo de condução da busca deste mapeamento sistemático.

No Apêndice A são apresentados todos os resultados das buscas de cada fonte de pesquisa, bem como os critérios de inclusão e exclusão associados aos resultados. Todos os resultados encontrados foram armazenados, organizados e catalogados utilizando a ferramenta Mendeley Desktop (MENDELEY, 2012).

3.3.3 Seleção dos Estudos Primários

Na Tabela 3.3 são apresentados os estudos primários selecionados de acordo com os critérios de inclusão e exclusão definidos no planejamento do mapeamento sistemático.

Tabela 3.3 Estudos primários selecionados.

Id	Título	Critério	Ano
ACM4	Methodology for identifying and solving accessibility related issues in web content management system environments.	CI2	2012
SC10	Using web content management systems for accessibility: The experience of a research institute portal.	CI3	2008
SC13	Toward the creation of a green content management .	C1	2011
SC18	End-user development for E-government website content creation .	CI2	2009
SC19	Managing accessibility in local e-government websites through end-user development: A case study.	CI2	2010
SC27	Evaluating the accessibility of three open-source learning content management systems: A comparative study.	CI5	2011
SC38	Influence of web content management systems in web content accessibility.	CI3	2011
SC39	Engineering accessibility in web content management system environments.	CI2	2009
SC40	Beyond standards: Unleashing accessibility on a learning content management system.	CI1	2011
SC45	A software solution for accessible e-government portals.	CI1	2008
SC46	Implementation concept for an accessible web CMS.	CI2	2010
SC51	Using a CMS to create fully accessible web sites.	CI1	2009

Na Tabela 3.4 são sumarizados todos os estudos primários incluídos e excluídos por base de busca.

Tabela 3.4: Quantidade de estudos primários incluídos e excluídos.

	Incluídos	Excluídos	Total	% Incluídos
ACM Digital Library	1	6	7	14,2%
IEEEXplore Digital Library	0	6	6	0%
ScienceDirect	0	6	6	0%
Scopus	11	54	65	16,9%
Springerlink	0	12	12	0%

Na Tabela 3.5 é apresentado o total de estudos primários incluídos e excluídos durante o processo de seleção do mapeamento sistemático.

Tabela 3.5: Quantidade de estudos primários incluídos, excluídos e total, pelas bases de busca.

	Incluídos	Excluídos	Total	% Incluídos
Total	12	84	96	12,5%

3.3.4 Análise

Com o intuito de responder as questões de pesquisa estipuladas no planejamento do mapeamento sistemático, os estudos primários foram lidos e analisados. A seguir é apresentada uma síntese das análises realizadas com base nas questões de pesquisa estabelecidas.

- **CMS Acessíveis Propostos na Literatura**

Nos estudos primários selecionados foram encontradas três propostas de CMSs acessíveis, mostrados no Quadro 3.5.

Quadro 3.5: CMSs acessíveis propostos na literatura.

Id	**CMS**	**Tecnologia**	**Licença**	**Ano**
SC13	Greeen CMS (GCMS)	Perl/ MySql/ Apache	*Open Source*	2011
SC45	RiS-Kommunal	ASP	Comercial	2008
SC51	Edimaster Plus	--	Comercial	2009

Ao analisar o Green CMS (GCMS) (CIMAN *et al*., 2011) apresentado no estudo SC13, observou-se que ele tem suporte a acessibilidade, porém não provê mecanismos de verificação e automação que garantam a acessibilidade. Os autores afirmam que a instanciação deve ser realizada exclusivamente por especialistas e não há nenhuma interface que possibilita a configuração do código HTML, pois muitas soluções para a implementação de acessibilidade exige um domínio total do código. Os autores afirmam que o CMS é dividido em partes fixas e variáveis, sendo as partes variáveis codificadas por um programador experiente, conhecedor dos padrões web. Os autores analisaram três CMSs (Joomla!, Drupal e Typo3) e afirman que nenhum deles tem suporte a acessibilidade. O GCMS é *open source* e está disponível para *download* no repositório Source forge.

O RiS-Kommunal (Nedbal e Petz, 2009) é um CMS amplamente usado por alguns

municípios da Europa Central, em países como Austria, Itália e Alemanhã. O Ris-Hommunal é um CMS voltado para o domínio e-Gov e possui um banco de dados com mais de cinquenta módulos, projetados para atender as especificidades do domínio e-Gov. Os portais criados por meio do Ris-Hommunal ficam hospedados em um único servidor. Em sua versão inicial não havia suporte a acessibilidade, o estudo primário SC45 descreve a metodologia utilizada na implantação de melhorias de acessibilidade no Ris-Hommunal. O estudo enfatiza os desafios enfrentados na construção de um CMS acessível com o advento de novas tecnologias.

O CMS Edimaster Plus desenvolvido pela Netic Hipermidia, utiliza mecanismos automatizados e campos pré-formatados, fornecendo uma base para permitir a criação de um conteúdo acessível. O Edimaster Plus armazena em um banco de dados os metadados da informação, contendo a sua estrutura. Assim, o conteúdo acessível no *Front-End* é gerado a partir dos metadados armazenados no banco de dados. Os autores afirmam que dessa forma é possível atualizar os padrões do código e disponibilizar o conteúdo acessível.

- **CMS Citados nos Estudos**

No Quadro 3.6 são mostrados todos os CMSs citados nos estudos primários, bem como as informações com relação à licença, linguagem de programação utilizada e o suporte a acessibilidade. O atributo relacionado ao suporte a acessibilidade foi extraído com base nas avaliações realizadas nos estudos, sendo apresentada a avaliação mais atual. A sigla "*ND*" foi utilizada para informar que o valor do atributo não foi definido no estudo.

Quadro 3.6: Tipos de licenças dos CMSs citados nos estudos.

CMS	**Id. Estudo Prim.**	**Tipo de Licença**	**Ling. de Programação**	**Suporte a Acessibilidade**	**Ano do Estudo**
EzPublish	SC38	*Open Source* / Comercial	PHP	*ND*	--
Sakai	SC27	*ND*	*ND*	Parcial	2011
Moodle	SC38	*Open Source*	PHP	Parcial	2011
Fruibile	SC10	*Open Source*	ASP	Sim	2008
Drupal	SC10 – SC38	*Open Source*	PHP	Parcial	2008
Joomla!	SC10 – SC38	*Open Source*	PHP	Não	2008
Typo3	ACM4 – SC38	*Open Source*	PHP	Não	2012
Plone	SC10 – SC38	*Open Source*	Python	Parcial	2008
Greeen CMS	SC13	*Open Source*	ND	Parcial	2011
OpenACS	SC27	*Open Source*	PHP	ND	ND
ATutor	SC27 – SC40	*Open Source*	PHP	Parcial	2011
RiS-Kommunal	SC45- SC46	*ND*	*ND*	Sim	2008
Edimaster Plus	SC51	Comercial	*ND*	Sim	2009
OpenCMS	ACM4	*Open Source*	Java	Não	2012

- **Tecnologias Utilizadas**

Após a análise dos estudos verificou-se que nenhum dos CMS citados e propostos tem suporte às novas versões do HTML e CSS (HTML5 e CSS3), que introduzem novos conceitos ao desenvolvimento web, tornando páginas mais semânticas. Com relação às linguagens de programação utilizadas, observou-se que a linguagem mais utilizada é a PHP, dentre outras linguagens utilizadas, como ASP, Python e Java.

- **Reusabilidade na utilização de CMSs**

Nos estudos analisados não foi encontrada nenhuma evidência da utilização de um processo de reutilização sistemática na instanciação de CMSs, como a abordagem de LPS proposta neste trabalho.

- **Domínios da Aplicação de CMSs**

No Quadro 3.7 são apresentados os resultados relacionados ao domínio de aplicação no qual os CMSs foram aplicados nos estudos. O resultado mostra o interesse em pesquisas relacionadas à utilização de CMSs acessíveis no domínio e-Gov, evidenciando a relevância deste trabalho de mestrado.

Quadro 3.7: Domínios de aplicação dos CMSs.

Id	**Domínio de Aplicação**	**Tipo de estudo**
ACM4	Geral	Metodologia
SC10	e-Inclusion	Instanciação
SC13	Geral	CMS
SC18	e-Gov	Abordagem
SC19	e-Gov	Estudo de Caso
SC27	e-Learning	Estudo Comparativo
SC38	Geral	Estudo Compatativo
SC39	Geral	Metodologia
SC40	e-Learning	CMS
SC45	e-Gov	CMS
SC46	e-Gov	Instanciação
SC51	Geral	CMS

3.4 Outros Trabalhos Relacionados

Durante o desenvolvimento desta pesquisa foram identificados alguns trabalhos diretamente relacionados, porém não foram encontrados nos resultados das buscas realizadas no

mapeamento sistemático.

Sandim (2009) apresenta o *framework* CMS, intitulado Pantaneiro, desenvolvido para facilitar a geração de aplicações Web e portais corporativos em uma plataforma e-gov, além de gerenciar e compartilhar o conteúdo dinamicamente. O projeto Pantaneiro foi iniciado em 2005 por meio de uma parceria entre o LEDES/UFMS e o governo do estado do Mato Grosso do Sul, instanciando cerca de cinquenta portais de secretarias e/ou órgãos do governo estadual (SANDIM, 2009). Contudo, foi identificado o fato de que as páginas geradas pelo Pantaneiro apresentavam problemas de acessibilidade, tais como: menu inacessível pelo teclado, imagens sem descrição textual e código HTML fora dos padrões estabelecidos pelo W3C (Maia, 2010). Maia (2010) apresenta um processo para o desenvolvimento de aplicações Web Acessíveis, e durante o desenvolvimento do projeto realizou significativas modificações de acessibilidade no Pantaneiro.

O Pantaneiro (SANDIM, 2009) estendido por Maia (2010) é relevante e coerente com as propostas deste trabalho, porém as abordagens são distintas. Enquanto no *framework* Pantaneiro os portais instanciados executam em apenas um servidor e são gerenciadas por um banco de dados único, a plataforma de LPS proposta neste trabalho gera aplicações únicas, com banco de dados e código próprio, ou seja, cada portal instanciado pela LPS é um produto.

3.5 Considerações Finais

Os estudos mostram que atualmente um grande número de organizações tem utilizado CMSs para gerenciar o conteúdo em seus portais (Burzagli et. al, 2008)(Fogli e Sacco, 2010) (Lopez et. al, 2012), porém muitas vezes os CMSs não satisfazem as necessidades de publicação em sua totalidade, em particular com relação a acessibilidade (Fogli e Sacco, 2010)

Lopez et al. (2011) apresentam um estudo sobre alguns CMSs mais utilizados no domínio e-Gov e propósitos gerais, chegando a conclusão que todos os CMSs analisados podem ser adaptados para torná-los acessíveis. No entanto, é necessário modificar as classes internas do CMS e realizar a modelagem do conteúdo web de forma acessível. Para a maioria dos CMSs existe também a possibilidade de incluir *plug-ins*, que podem ser utilizados como ferramentas para melhorar a acessibilidade. De qualquer forma, todas essas soluções exigem conhecimento profundo e exaustivo sobre cada CMS. Além disso, é importante a utilização de

mecanismos de monitoramento de acessibilidade para verificar a acessibilidade do conteúdo da web gerenciado.

Na análise dos estudos primários foi possível identificar que a construção de portais acessíveis depende do conhecimento de tecnologias, apoiadas por metodologias que auxiliem no processo de desenvolvimento, assim como uma boa familiaridade com as diretrizes de acessibilidade (Burzagli et. al, 2008) (Lopez et. al , 2012). Acessibilidade em CMSs depende também dos usuários que alimentam o conteúdo (Burzagli et. al, 2008) (Fogli, 2009), os Usuários Finais de Desenvolvimento (UFD). Fogli (2009) propõe uma abordagem para estender CMSs com técnicas UFD, visto que os CMSs geralmente não satisfazem todas as exigências e necessidades de um domínio de aplicação. Portanto, a adoção de técnicas UFD pode representar uma possível solução para as limitações atuais dos CMSs (Fogli, 2009).

Os estudos mostram também que a acessibilidade deve ser considerada e gerenciada em um processo contínuo. Embora um CMS seja capaz de cumprir todas as diretrizes de acessibilidade, é completamente possível que um portal instanciado não seja acessível. Mesmo que todas as possibilidades técnicas sejam previstas, um administrador bem treinado de um portal é crucial para um portal sustentável acessível. Por isso, é inevitável a elaboração de políticas de conscientização sobre o tema em geral (Nedbal e Petz, 2008) (Fogli, 2009) (Lopez et al., 2011).

As evidências encontradas na literatura sugerem que o advento de novas tendências e tecnologias traz um grande desafio para a pesquisa na área de acessibilidade web (Burzagli et. al, 2008) (Nedbal e Petz, 2008) (Lopez et al., 2009). A questão permanece, como essas novas tendências e tecnologias podem ser combinadas com as diretrizes de acessibilidade para satisfazer as necessidades de todos os usuários.

Capítulo 4

Plataforma da LPS no Domínio de Portais e-Gov Acessíveis

4.1 Considerações Iniciais

Pretendendo consolidar um processo de desenvolvimento ágil para o domínio de portais e-Gov acessíveis, reaproveitando de forma sistemática os esforços e recursos despendidos na construção e evolução das aplicações, é proposta uma plataforma de LPS automatizada. Os artefatos gerados nesse processo são armazenados em um repositório de reúso, onde poderão ser reutilizados no desenvolvimento de novos portais e-Gov acessíveis.

Neste capítulo é apresentada a plataforma de LPS proposta, bem como o detalhamento da sua arquitetura e as tecnologias utilizadas. A organização do capítulo está disposta da seguinte forma: na Seção 4.2 é apresentada a arquitetura da plataforma de geração de portais e-Gov acessíveis; na Seção 4.3 é apresentado o repositório Titan Extensions; na Seção 4.4 é apresentado o gerador de aplicações Titan FrontEnd; e por fim as considerações finais deste capítulo são apresentadas na Seção 4.5.

4.2 Arquitetura da Plataforma de Geração de Portais e-Gov Acessíveis

A arquitetura da plataforma de automação da LPS proposta está dividida em dois blocos principais: *Front-End* e *Back-End*. Conforme ilustrado na Figura 4.1, nos dois blocos são utilizados geradores de aplicação e *frameworks* para automatizar o processo de instanciação, de modo que o primeiro bloco a ser instanciado é o *Back-End*, e a partir dele, o segundo bloco *Front-End* é instanciado.

Na prática, o produto gerado na instanciação do bloco *Front-End* da plataforma consiste na área pública de um portal, onde os usuários finais podem acessar o conteúdo disponível e interagir com a aplicação Web. Por outro lado, o bloco *Back-End* contempla a área de gestão de conteúdo da aplicação, onde as informações são inseridas, armazenadas e recuperadas para exibição no *Front-End*.

Para Cleaveland (1988), geradores de aplicação são sistemas de software que

transformam especificações em uma aplicação. As especificações descrevem o problema ou a tarefa que deve ser realizada pelo gerador. Essas especificações podem ser modeladas de forma gráfica, escritas em alguma linguagem intermediária ou ainda criadas de forma interativa, em que o usuário seleciona as características desejadas por meio de escolhas em uma sequência de formulários ou menus. Dessa forma, conforme ilustrado na Figura 4.1, a instanciação do bloco *Back-End* é realizada de forma interativa pelo engenheiro de aplicação através do gerador de aplicações Titan Architect, que gera os arquivos XML (*Extensible Markup Language*) e SQL (*Structured Query Language*) necessários para a instanciação do Titan Framework. Da mesma forma, assim como o Titan Architect é utilizado para instanciação do Titan Framework, o gerador de aplicações proposto nesse trabalho, denominado Titan FrontEnd, é responsável pela geração do bloco *Front-End* da plataforma, instanciando automaticamente o Zend Framework. A combinação da utilização desses geradores de aplicação e *frameworks* será detalhada nas próximas seções.

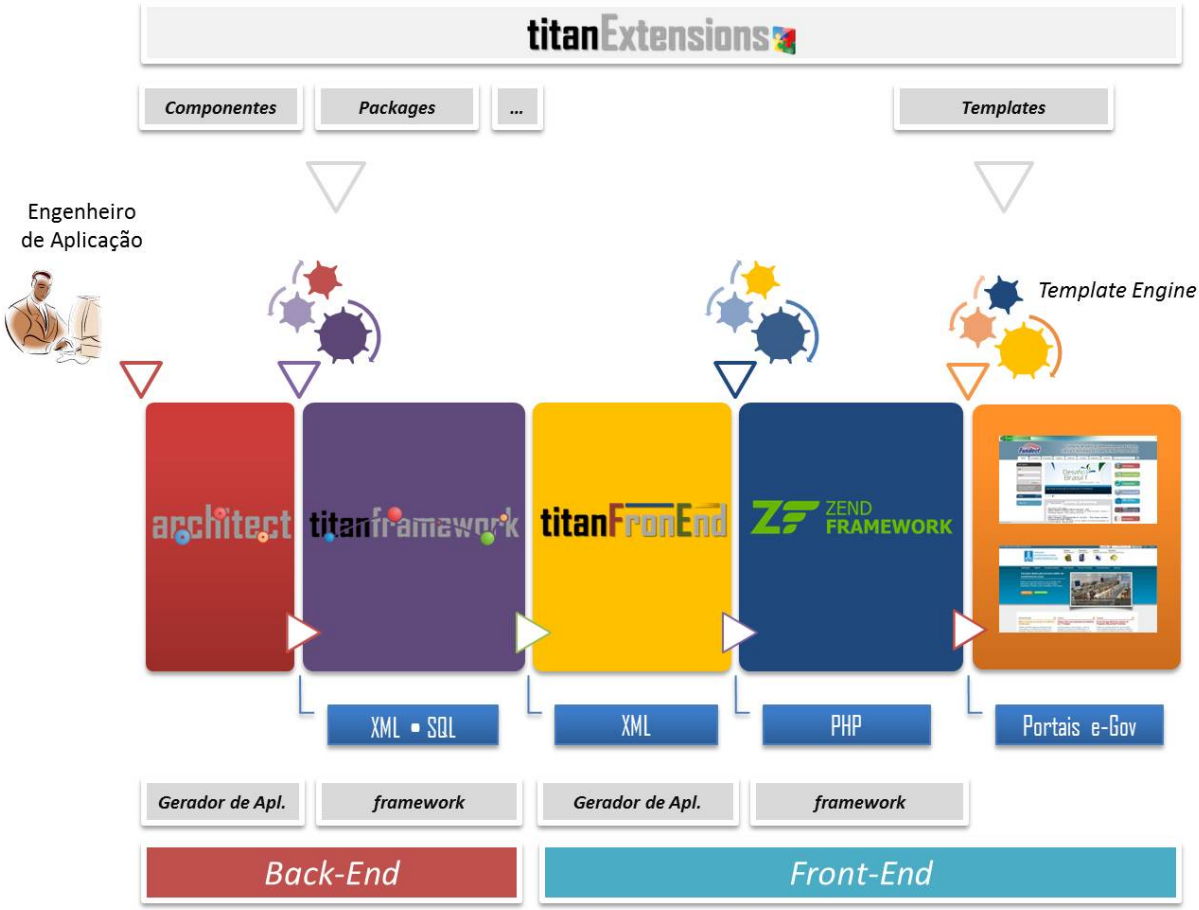

Figura 4.1: Processo de Geração de Portais e-Gov.

Adicionalmente, para otimizar e maximizar a reutilização de artefatos construídos na Engenharia de Aplicação dos diversos produtos instanciados pela LPS, bem como promover

o desenvolvimento colaborativo, foi implementado um repositório de reúso em ambiente Web, denominado Titan Extensions, onde é possível disponibilizar os artefatos produzidos na Engenharia de Domínio, assim como as *extensions* (*Componente*, *Types*, *Packages*, *Templates*, Idiomas, etc.) desenvolvidas para o *framework* Titan durante esse processo. Assim, o repositório atua como uma base de artefatos de software que alimentam todo o processo de instanciação da LPS. Adicionamente, foi desenvolvida uma ferramenta *wizard* para instalação automática de *extensions*, ou seja, o desenvolvedor poderá realizar o *download* de *extensions* disponíveis nesse repositório e executar a instalação através de um *wizard*, sem a necessidade de programação.

Uma característica importante da plataforma está relacionada ao processo de manutenção e evolução da LPS. Em todas as aplicações instanciadas pela LPS, o manual do usuário é gerado automaticamente em tempo de execução, por meio de uma ferramenta automatizada implementada pelo *framework* Titan. Dessa forma, havendo alguma alteração na aplicação após a instanciação pela LPS, a documentação é atualizada automaticamente.

A plataforma proposta utiliza tecnologias *open source* e faz uso de técnicas e ferramentas baseadas no conceito de desenvolvimento de comunidades de software livre. Todas as ferramentas da plataforma foram desenvolvidas na linguagem de programação PHP, e utilizam o banco de dados PostgreSQL, executando sobre o servidor Web Apache com PHP. As tecnologias utilizadas automatizam a Engenharia de Aplicação e provêem uma maneira dinâmica e eficiente de reutilização de artefatos. A seguir são detalhadas as ferramentas Titan Extensions e Titan FrontEnd, respectivamente.

4.3 Titan Extensions

Um repositório de reúso é uma coleção de ativos reutilizáveis que provê funcionalidades, tal como mecanismos de pesquisa e recuperação (MILI *et al*., 1998). Apperly (2001) afirma que um repositório de reúso desempenha um papel crucial na comunicação entre produtores, que estão interessados em publicar ou disponibilizar ativos reutilizáveis, e consumidores, que estão preocupados em encontrar e recuperar ativos para o desenvolvimento de software. Sob essa perspectiva, o ambiente do repositório Titan Extensions é dividido em duas visões: a) Visão do Produtor e b) Visão do Consumidor.

O Titan Extensions é um ambiente que possibilita a publicação de ativos desenvolvidos por diferentes programadores em diversos projetos durante o ciclo de vida da

LPS, de tal forma que as extensões podem ser utilizadas em outras aplicações por qualquer engenheiro de aplicação. Sendo assim, um engenheiro de aplicação pode atuar nas duas visões do repositório, produzindo e consumindo ativos durante a instanciação da LPS.

A concepção da implementação do Titan Extensions se deu através da identificação de limitações no repositório da arquitetura proposta por Carromeu *et al*. (2010), que segundo a abordagem PLUS, implementa o Repositório da LPS alimentando a construção de novas aplicações.

No contexto da plataforma proposta neste trabalho, dentre os dezessete requisitos propostos por Burégio *et al*. (2008) apresentados no capítulo anterior, para a implementação da versão inicial foram selecionados os seguintes requisitos mostrados no Quadro 4.1. Uma visão geral do ambiente do repositório é ilustrada na Figura 4.2.

Quadro 4.1: Requisitos Selecionados para Implementação do Titan Extensions

RP	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17
RS	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✔	✘	✔	✔

RP: Requisitos Propostos | **RS**: Requisitos Selecionados

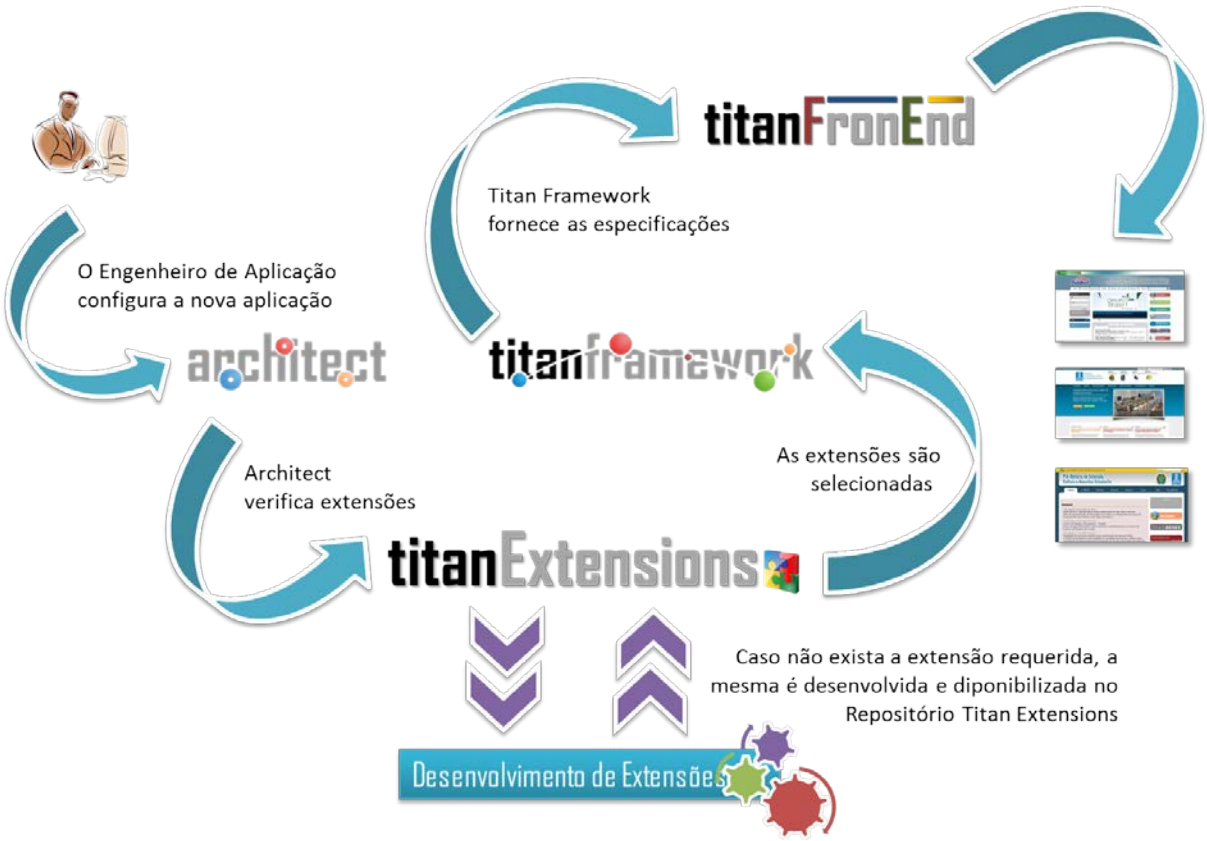

Figura 4.2: Visão geral de funcionamento do repositório Titan Extensions.

O Titan Extensions implementa os seguintes requisitos:

- Possibilita o cadastro de metadados de ativos;
- Possibilita a inserção de ativos;
- Permite apenas a publicação da especificação do ativo;
- Possibilita a navegação pelos ativos por categoria;
- Fornece mecanismos de busca;
- Possibilita a geração de relatórios sobre a utilização do repositório
- Registra preferências de usuários e notifica-os sobre os interesses comuns;
- Implementa serviços administrativos, tais como, classificadores de ativos e tipos de artefatos de ativos;
- Realiza controle de versões dos ativos;
- Possibilita o gerenciamento de dependência;
- Fornece serviços de *feedback* sobre a utilização de ativos;
- Coleta métricas de reúso;
- Permite controle de acesso; e
- Permite controle de mudanças.

4.3.1 Visão do Produtor (VP)

A Visão do Produtor (VP) possibilita que um engenheiro de aplicação disponibilize os artefatos de software, gerados durante o processo de instanciação da LPS, no repositório através uma interface gráfica organizada em etapas. Em sua versão inicial, é possível fazer a publicação de extensões do tipo: *Component*, *Type*, *Package* e *Template*.

O Titan Extensions foi desenvolvido utilizando o *framework* Titan, sendo, portanto uma instância do *framework,* conforme ilustrado na Figura 4.3. A instância pode ser acessada publicamente através da URL http://extensions.titanframework.com/manager, onde é possível

realizar um cadastro e se tornar um usuário produtor registrado. Apenas os usuários registrados poderão publicar extensões no repositório e cada usuário poderá gerenciar apenas as extensões inseridas por ele. Para a instanciação da VP no repositório foi necessária a criação de um componente, denominado *extension.creator*. O componente *extension.creator* implementa as *engines* responsáveis pela insersão, validação e publicação das extensões.

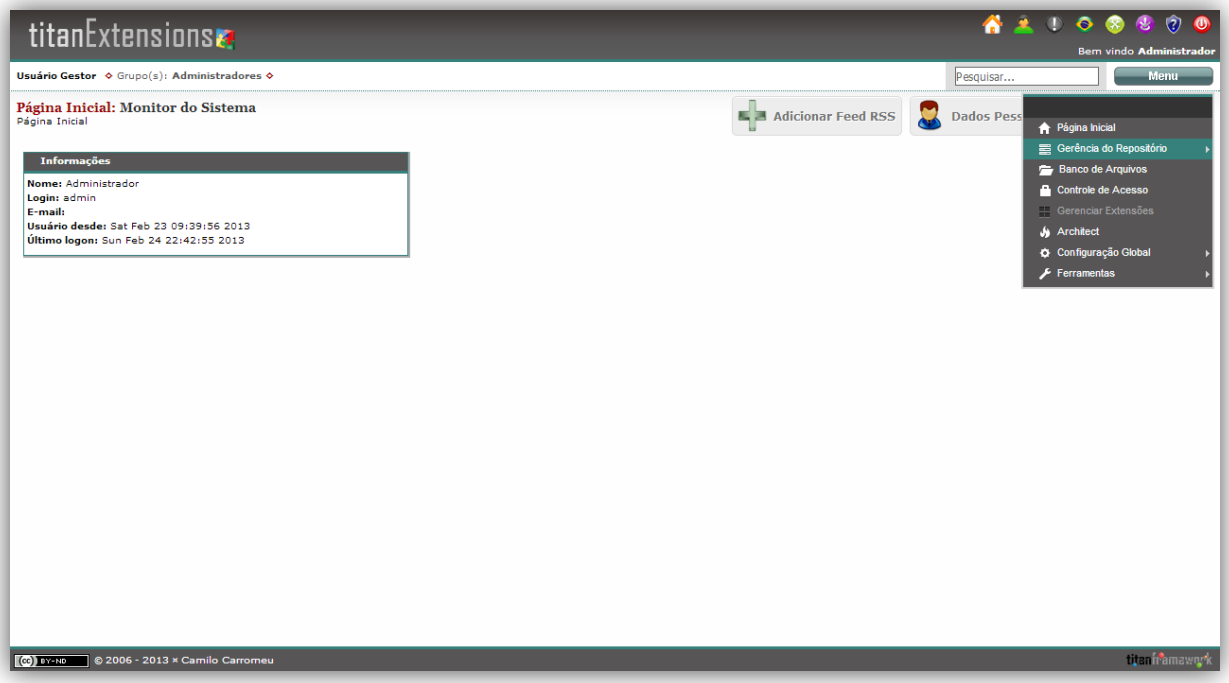

Figura 4.3: Tela inicial do Titan Extensions.

Ao inserir uma nova extensão no repositório, além do artefato de código, o usuário produtor deve informar os metadados do ativo, conforme mostra o Quadro 4.2.

Quadro 4.2: Metadados de um ativo do repositório.

Metadado	**Descrição**
Nome	O nome do ativo que será exibido no ambiente de navegação do repositório.
Versão	A versão relacionada às *features* disponibilizada s na *release* do ativo.
Licença	Licença de uso.
Website	Website do autor do ativo.
Documentação	*Link* externo para documentação/manual sobre a utilização do ativo.
Categoria	Categoria do ativo (e-Gov, Multimidia, Design, Comunicação, Financeiro, etc).
Descrição	A descrição do ativo que será exibida no ambiente de navegação do repositório.
Imagem	A imagem que será utilizada no ambiente de navegação do repositório.

As extensões do tipo *Type*, *Component* e *Template* podem ser adicionadas ao repositório em apenas uma etapa de configuração, já a inserção de uma extensão do tipo *Package* é subdividida em quatro etapas, conforme ilustrado na Figura 4.4.

Um *Package* é composto por uma ou mais seções, que por sua vez utiliza diversos *Types* e fazem uso de um ou mais *Component*. Por exemplo, o *package* Notícias faz utilização de três *types* do repositório (*String, Date* e *Fck*) e um *component* (*global.generic*), conforme detalhado na Tabela 4.1.

Tabela 4.1: Package de Notícias.

Package	***Type***	***Component***
Notícias – possui os campos: [Título], [Data]. [Texto] e [Fonte].	*String* – tipo de dado responsável pelas informações do campo [Título] e [Fonte] da notícia.	*global.generic* – permite efetuar a listagem, criação, edição e remoção de uma notícia. Implementa funcionalidades de busca, ordenação, paginação, validação, controle de versão, etc.
	Date – tipo de dado responsável pela informação do campo [Data] da notícia.	
	Fck – tipo de dado responsável pela informação do campo [Texto] da notícia.	

A seguir são detalhadas as etapas para a inserção de um *Package* no repositório:

- **Etapa 1:** os metadados do *Package* são inseridos e o código fonte contendo os XML's de parametrização é enviado em um arquivo *.zip* com a seguinte estrutura:

```
package/
        mypackage/
        _repos/
                component/
                type/
        package.xml
        installation.xml
```

- **Etapa 2 e 3:** as informações contidas no arquivo *package.xml* inserido na primeira etapa são recuperadas com o objetivo de obter informações sobre os componentes e tipos utilizados pelo *package*. Caso algum componente ou tipo detectado não seja nativo e não esteja no repositório Titan Extensions, o sistema direciona o usuário para inserir o arquivo *.zip* com o conteúdo do componente ou tipo (arquivos *.php*, *.js*, *.css*, etc). Caso os componentes ou tipos sejam nativos ou já estejam disponíveis no repositório, as informações

dos componentes ou tipos são apenas exibidas, conforme ilustrado na Figura 4.5.

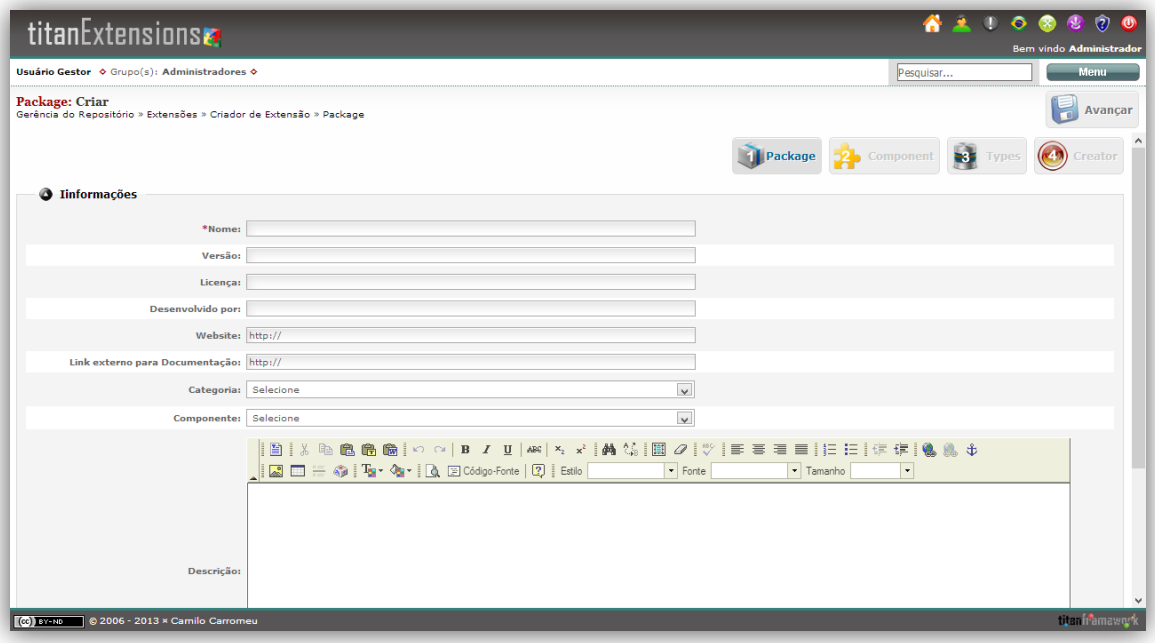

Figura 4.4: Inserção de um Package no Titan Extensions.

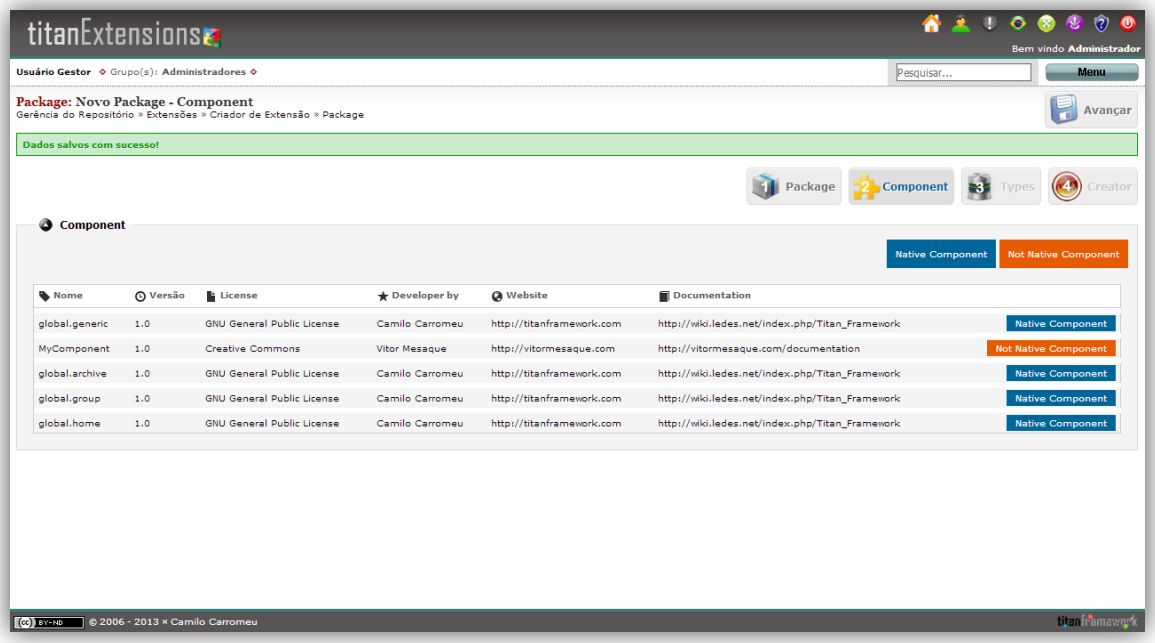

Figura 4.5: Segunda etapa do processo de inserção de um package no repositório.

- **Etapa 4:** na última etapa o sistema realiza todas as verificações de pendências

e validações necessárias para a criação do *package*, apresentando um quadro resumo de todas as configurações realizadas.

A cada etapa de inserção de uma nova extensão no repositório, o sistema faz diversas verificações para garantir que não haja inconsistências no processo, impedindo a disponibilização da extensão caso haja alguma incoerência.

Todas as configurações e relacionamentos realizados no processo de inserção de uma extensão são armazenados no banco de dados do Titan Extension, permitindo dessa forma o gerenciamento de dependência dos ativos do repositório. Para otimizar os recursos do servidor Web, o código fonte de cada extensão é armazenado individualmente. Assim, quando o usuário consumidor efetua o *download* de uma extensão, o sistema é responsável por recuperar as dependências da extensão requerida e montar o arquivo para *download* em tempo de execução. O resultado desse processo é um arquivo *.phar*[8] com todo o conteúdo necessário para instalação da extensão. Adicionalmente, foi implementado uma ferramenta denominada *Extension Installer*, que permite a instalação e configuração de extensões, sem a necessidade de programação, por meio de um wizard (Figura 4.6).

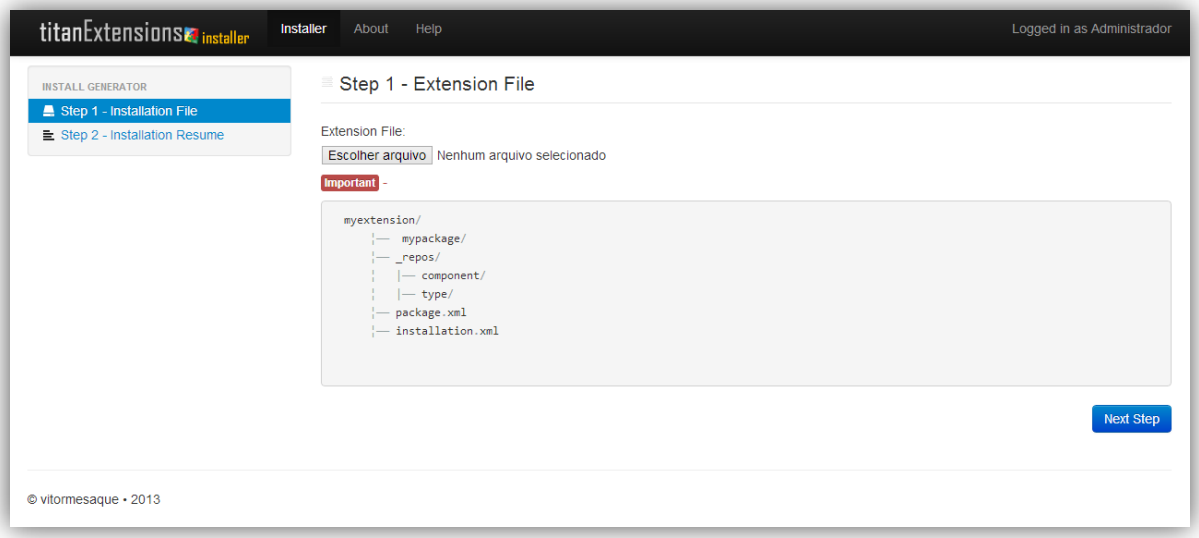

Figura 4.6: Ferramenta Extension Installer

Após a inserção de uma extensão no repositório, é necessário que um usuário moderador do repositório valide a extensão para que ela possa ficar disponível para utilização pelos usuários consumidores do repositório.

[8] Phar é um arquivo que fornece uma maneira de distribuir uma aplicação PHP completa em um único arquivo e executá-lo a partir desse arquivo, sem a necessidade de extraí-lo para o disco (PHP, 2012).

É importante salientar que o *core* do Titan Framework está sob a licença Creative Commons - (*by-nd*) (CC, 2013), que permite a redistribuição e o uso para fins comerciais e não comerciais, contanto que a obra seja redistribuída sem modificações e completa, e que os créditos sejam atribuídos ao autor. Porém todas as extensões nativas do *framework* estão sob a licença GNU General Public License (GPL, 2013), o que permite aos utilizadores adaptar as extensões as suas necessidades e a liberdade para redistribuir.

4.3.2 Visão do Consumidor (VC)

A Visão do Consumidor é um portal disponível publicamente através da URL **http://extensions.titanframework.com/**, onde as extensões disponibilizadas na VP são categorizadas e expostas de forma a permitir a fácil navegação e busca por extensões. Na VC os usuários consumidores podem realizar o *download* de extensões e contribuir fornecendo informações de *feedback* sobre as extensões disponíveis no repositório (Figura 4.7, Figura 4.8 e Figura 4.9). Dentre as funcionalidades disponíveis no Titan Extensions - VC pode-se destacar:

- Avaliação das extensões pelos usuários consumidores;
- Classificação das extensões por categorias:
    - melhores extensões;
    - extensões mais utilizadas; e
    - extensões atualizadas recentemente.
- Compartilhamento de extensões em redes sociais;
- Ambiente central para disponilização de documentação da API do Titan Framework, bem como o fornecimento de informações relacionadas a funcionalidades liberadas nas *releases*.

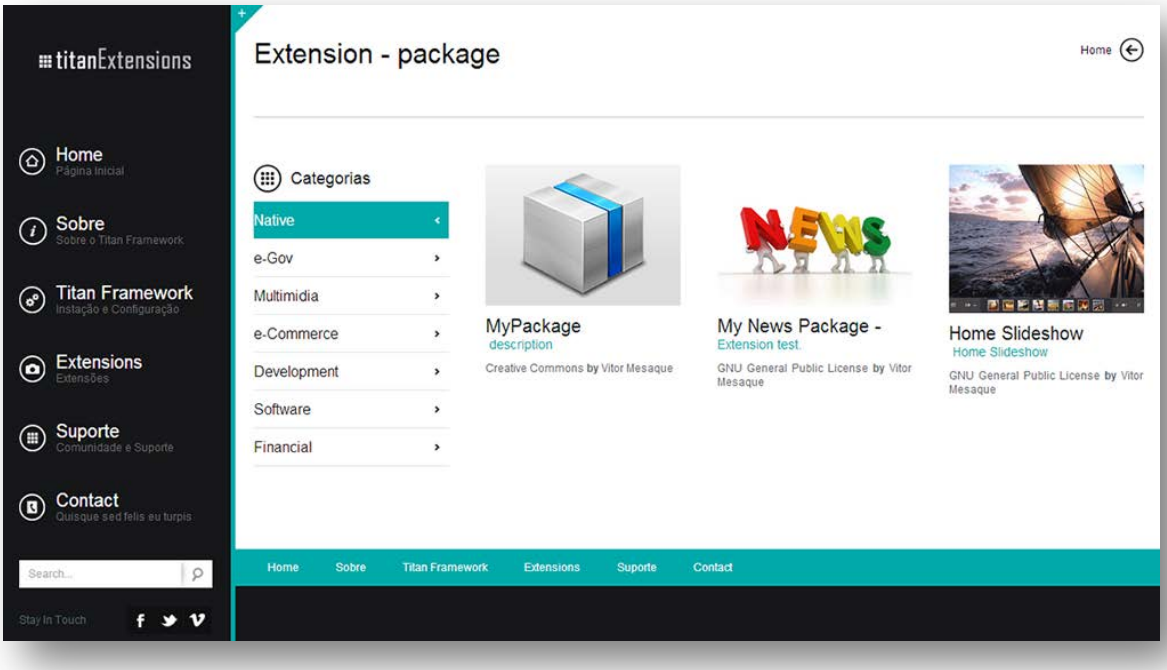

Figura 4.7: Visão do Consumidor do Titan Extensions.

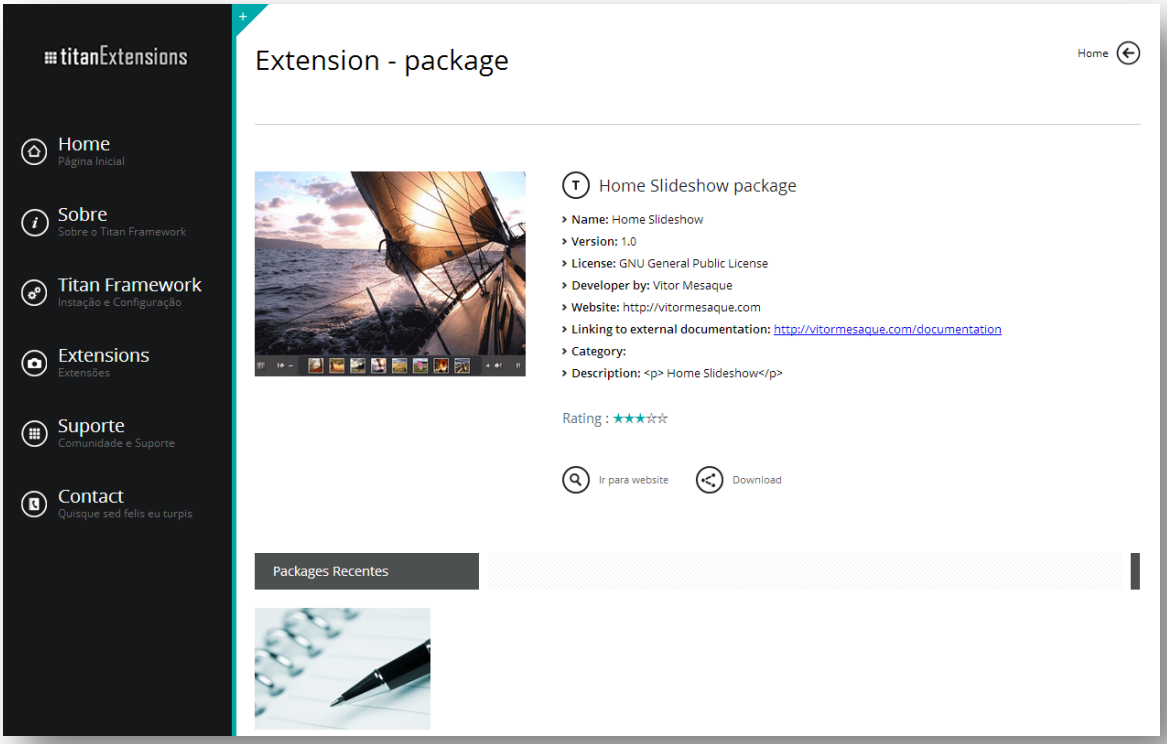

Figura 4.8: Tela de download de um Package.

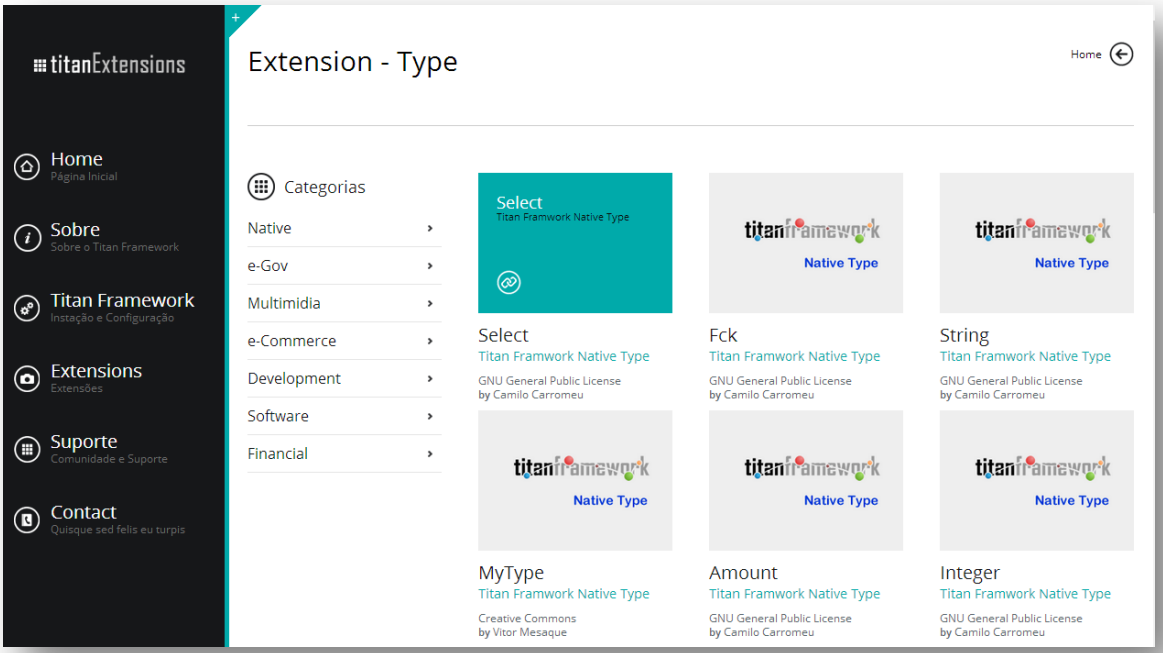

Figura 4.9: Tela da listagem de types disponíveis no repositório.

4.3.1 Benefícios do Titan Extensions

O repositório Titan Extensions provê um meio de comunicação entre as partes interessadas em produzir e consumir ativos reutilizáveis, representando um canal de comunicação de duas vias: o desenvolvimento para o reúso (produção) e o desenvolvimento com reúso (consumo). Além disso, o repositório atua como um suporte ao gerenciamento, ajudando os engenheiros a controlar a sua utilização de ativos, com o objetivo de alinhar iniciativas, projetos e medir o impacto do reúso nos negócios.

O Titan Extensions também atua como um promotor de reúso, disseminando a cultura de reúso no processo de desenvolvimento de software, fornecendo visibilidade em seus ativos e disseminando notícias sobre as iniciativas de reúso. O repositório atua ainda como uma garantia de qualidade, auxiliando no alcance do objetivo da organização de manter a qualidade em todo o ciclo de vida de desenvolvimento de software, fornecendo ativos de software provados e apoiando um processo de certificação eficiente de ativos.

4.4 Titan FrontEnd

O Titan FrontEnd consiste em um *wizard*, gerador de aplicações, que é responsável por receber especificações relacionadas a uma determinada instância (gerada pelo bloco *Back-End* da plataforma) e gerar o código fonte, em tempo de execução, para a instanciação do bloco *Front-End.*

4.4.1 Zend Framework

Com o objetivo de minimizar a complexidade do código gerado e aproveitar soluções existentes, optou-se por utilizar o *framework* de aplicações Web Zend Framework (ZEND TECHNOLOGIES, 2013), que em conjunto com o Titan FrontEnd compõe a automação do processo de instanciação do bloco *Front-End*. O Zend Framework, criado no início de 2005, é orientado a objetos e implementado em PHP que utiliza o padrão de projeto *Model-View-Controller* (MVC) (BURBECK, 1987) (GAMMA, 1995). O *framework* Zend foi escolhido para ser usado neste trabalho por implementar as seguintes características:

- Implementa componentes de software genéricos, fracamente acoplados e extensíveis;
- Implementa o padrão MVC extensível suportando *layouts* e *templates*;
- Suporte a internacionalização;
- Suporte a criação de formulários utilizando XML;
- Possui o componente nativo (*Zend_Layout*) que permite a criação *templates* de in terface.

Desta forma, o gerador de aplicações Titan FrontEnd, a partir das especificações de entrada, instancia as classes providas pelo *framework* Zend que implementam as camadas de controle, modelo e visão da aplicação. Em síntese, o Titan FrontEnd gera código para o Zend Framework, garantindo um processo de codificação mais ágil e menos suscetível a erros humanos, visto que os geradores podem produzir código de forma sistemática e mais segura em relação aos métodos tradicionais de programação (CLEAVELAND, 2001) (CZARNECKI e EISENERCKER, 2002).

4.4.2 Arquitetura do Titan FrontEnd

O Titan FrontEnd foi construído como uma ferramenta (*tool*) do *framework* Titan, sendo implementado utilizando a estrutura do próprio *framework*, facilitando a recuperação dos dados contidos nos arquivos XML's utilizados no processo de geração. Na arquitetura do Titan, uma *tool* é um artefato de código provido pelo *framework* que é utilizado com o objetivo de facilitar ou automatizar uma tarefa realizada pelo desenvolvedor da aplicação, por exemplo, a *tool* de tradução automática de instâncias.

De uma maneira geral, o Titan FrontEnd faz uma varredura em todo o diretório *section* do *framework* Titan, procurando por arquivos *frontend.xml*, *view.xml* e *all.xml*, nessa precedência, recuperando as informações das seções do Titan. Outras informações sobre a aplicação, utilizadas no processo de geração, são obtidas por meio dos arquivos *titan.xml* e *business.xml*, tais como, informações de conexão com o banco de dados, nome e descrição da aplicação e das seções, entre outras. As informações obtidas por meio dos arquivos XML's são combinadas com as parametrizações realizadas pelo engenheiro de aplicação, alimentando o processo de geração. O Titan FrontEnd faz a importação das bibliotecas (Zend Framework e Twitter Bootstrap) utilizadas na instância gerada e fornece ao gerador um conjunto de gabaritos de código que são parametrizadas dando origem aos artefatos de código da WebApp. Uma visão geral da arquitetura do Titan FrontEnd pode ser visualizada na Figura 4.10.

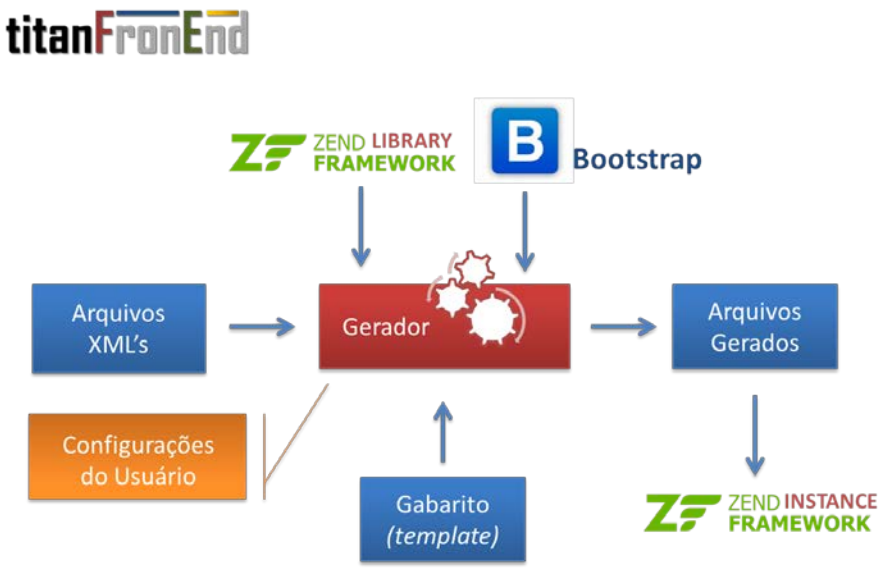

Figura 4.10: Arquitetura do Titan FronEnd.

A entidade seção do Titan foi mapeada para a entidade módulo do Zend, ou seja, no processo de geração da interface da aplicação, cada seção do Titan (bloco *Back-End*) foi transformada em um módulo do Zend. A Figura 4.11 mostra a árvore de diretório e os arquivos gerados para um módulo do Zend a partir de uma seção do Titan.

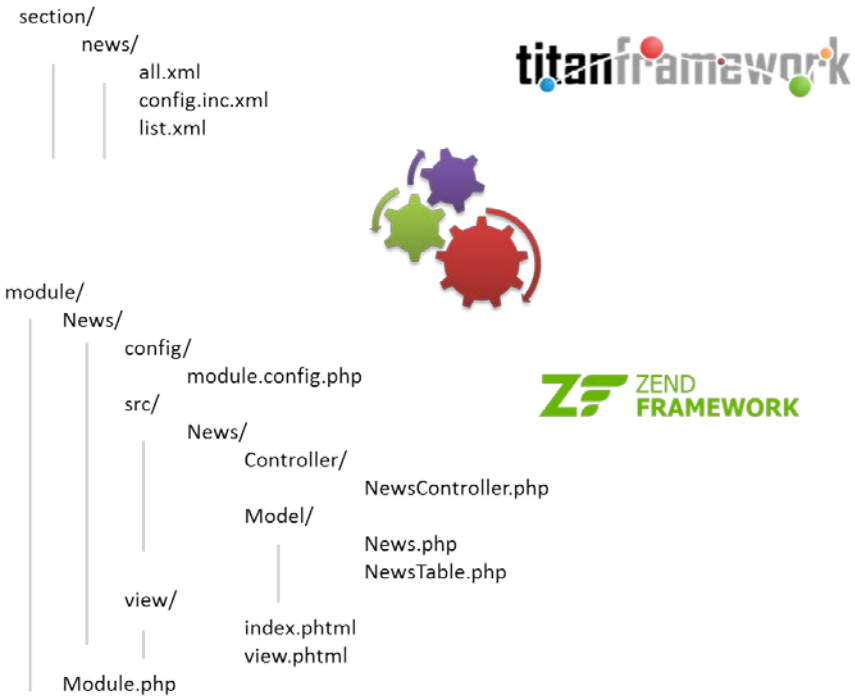

Figura 4.11: Árvore de diretório gerada.

Com a recuperação das informações dos arquivos XML's contidos nas seções do Titan, o gerador Titan FrontEnd recebe um *template* contendo um código base para a parametrização e instanciação dos módulos correspondentes para o Zend Framework. Para exemplificar, a seguir é apresentada a geração de um bloco de código de uma seção desenvolvida para o Titan Extensions (Visão Produtor), a seção *package*. Esta seção contém as informações relacionadas à inserção/edição/visualização de um Package no repositório.

Na Figura 4.12 é apresentado um XML de entrada *(package/all.xml)* e nas Figuras 4.13 e 4.14 o *template* de entrada e o código gerado, respectivamente.

XML de entrada - *(package/all.xml)*

```xml
<?xml version="1.0" encoding="UTF-8"?>
<form table="public.package" primary="id">
    <go-to flag="success" action="component" />
    <go-to flag="fail" action="[same]" />
    <go-to flag="cancel" action="[default]" />
    <go-to flag="componentView" action="componentView" />
    <group label="Info | pt_BR: Iinformações | es_ES: ">
        <field type="String" column="name" required="true" label="Name | pt_BR: Nome | es_ES: "/>
        <field type="String" column="version" label="Version | pt_BR: Versão | es_ES: " />
        <field type="String" column="license" label="License | pt_BR: Licença | es_ES: " />
        <field type="String" column="developer" label="Developer by | pt_BR: Desenvolvido por | es_ES: " />
        <field type="String" column="website" label="Website | pt_BR: Website | es_ES: " value="http://" />
        <field type="String" column="documentation"
                        label="Linking to external documentation | pt_BR: Link externo para Documentação | es_ES: " value="http://" />
        <field type="Fck" column="description" label="Description | pt_BR: Descrição | es_ES: " />
        <field type="File" column="image" label="Image | pt_BR: Imagem | es_ES: " tip="JPEG/GIF/PNG/BMP">
            <mime-type>image/jpeg</mime-type>
            <mime-type>image/gif</mime-type>
            <mime-type>image/pjpeg</mime-type>
            <mime-type>image/png</mime-type>
            <mime-type>image/x-bitmap</mime-type>
            <mime-type>image/photoshop</mime-type>
            <mime-type>image/bmp</mime-type>
        </field>
        <field type="String" column="unix_name" required="true" label="Unix Name | pt_BR: Nome Unix | es_ES: " tip="Package directory name."/>
        <field type="File" column="archive" label="Section Folder | pt_BR: Pasta da Seção | es_ES: " required="true"
                        help=" | pt_BR: -- | es_ES: " tip="Upload Section Folder with XML's Files (MyPackage/section/*.xml)">
            <mime-type>application/zip</mime-type>
            <mime-type>application/x-rar-compressed</mime-type>
            <mime-type>application/x-zip-compressed</mime-type>
            <mime-type>application/x-7z-compressed</mime-type>
        </field>
    </group>
</form>
```

Figura 4.12: Aquivo *all.xml* da seção *Package* do Titan Extensions.

***Template* de Entrada**

```php
<?php
namespace {$module}\Model;

class {$module}
{
    [_MODEL_BLOCK_1_]

    public function exchangeArray($data)
    {
        [_MODEL_BLOCK_2_]
    }

    [_MODEL_BLOCK_3_]

}
```

Figura 4.13: Template de entrada para o gerador.

Arquivo de Saída – *Model/Package.php*

```php
<?php
namespace Package\Model;

class Package
{
    public $id;
    public $name;
    public $version;
    public $license;
    public $developer;
    public $website;
    public $documentation;
    public $category;
    public $component;
    public $description;
    public $image;
    public $archive;

    public function exchangeArray($data)
    {
        $this->id = (isset($data['id'])) ? $data['id'] : null;
        $this->name = (isset($data['name'])) ? $data['name'] : null;
        $this->version = (isset($data['version'])) ? $data['version'] : null;
        $this->license = (isset($data['license'])) ? $data['license'] : null;
        $this->developer = (isset($data['developer'])) ? $data['developer'] : null;
        $this->website = (isset($data['website'])) ? $data['website'] : null;
        $this->documentation = (isset($data['documentation'])) ? $data['documentation'] : null;
        $this->category = (isset($data['category'])) ? $data['category'] : null;
        $this->component = (isset($data['component'])) ? $data['component'] : null;
        $this->description = (isset($data['description'])) ? $data['description'] : null;
        $this->image = (isset($data['image'])) ? $data['image'] : null;
        $this->archive = (isset($data['archive'])) ? $data['archive'] : null;

    }

    public function getId(){ return $this->id; }
    public function getName(){ return $this->name; }
    public function getVersion(){ return $this->version; }
    public function getLicense(){ return $this->license; }
    public function getDeveloper(){ return $this->developer; }
    public function getWebsite(){ return $this->website; }
    public function getDocumentation(){ return $this->documentation; }
    public function getCategory(){ return $this->category; }
    public function getComponent(){ return $this->component; }
    public function getDescription(){ return $this->description; }
    public function getImage(){ return $this->image; }
    public function getArchive(){ return $this->archive; }


}
```

Figura 4.14: Arquivo de saída gerado pelo Titan FrontEnd.

Além dos módulos gerados, o Titan FrontEnd gera os arquivos de configuração do Zend responsáveis pela conexão com o banco de dados e metadados da aplicação, conforme mostra a Figura 4.14.

```php
<?php
return array(
    'modules' => array(
        'Application',
        'About',
        'Access',
        'Category',
        'Component',
        'Manager',
        'Package',
        'Type',
        'Slideshow',

    ),
    'module_listener_options' => array(
        'config_glob_paths'    => array(
            'config/autoload/{,*.}{global,local}.php',
        ),
        'module_paths' => array(
            './module',
            './vendor',
        ),
    ),
);
```

```php
<?php

return array(
    'db' => array(
                'driver' => 'Pdo_Pgsql',
                'dsn'    => 'pgsql:host=192.168.56.2;port=5432;dbname=titanextensions',
          ),
    'service_manager' => array(
        'factories' => array(
            'Zend\Db\Adapter\Adapter'
                    => 'Zend\Db\Adapter\AdapterServiceFactory',
        ),
    ),
    'navigation' => array(
                            'default' => array(
                                array( 'label' => 'Home', 'route' => 'home', ),
                                array( 'label' => 'About ','route' => 'about',),
                                array( 'label' => 'Access Control ', 'route' => 'access',),
                                array( 'label' => 'Category ', 'route' => 'category',),
                                array( 'label' => 'Component ', 'route' => 'component',),
                                array( 'label' => 'Managers Users ', 'route' => 'manager',),
                                array( 'label' => 'Package ', 'route' => 'package',),
                                array( 'label' => 'Slideshow ', 'route' => 'slideshow',), ))
);
```

Figura 4.15: Arquivos de configuração do Zend Framework gerados pelo Titan FrontEnd.

O Titan FrontEnd possui um motor de *template* (*template engine*) que renderiza o *layout* da aplicação a partir de *templates* implementados utilizando o *framework* CSS Twitter Bootstrap (BOOTSTRAP, 2013). O Twitter Bootstrap é uma biblioteca *open source* que contém modelos de elementos HTML5 e CSS3 para tipografia, formulários, botões, tabelas, navegação entre outros componentes, bem como extensões de *JavaScript*. O Titan FrontEnd importa a biblioteca do Twitter Bootstrap no momento da geração e possibita a escolha de *templates* de *layout* pré-defindos pelo gerador, ou *templates* importados do repositório Titan Extensions, gerando uma aplicação renderizada com o *template* de *layout* escolhido pelo engenheiro de aplicação, podendo ser customizado após a geração por um *web designer*, uma vez que a arquitetura do código gerado separa a camada de negócio, da camada de visualização.

Através de uma interface bem definida, *templates* podem ser construídos utilizando o Twitter Bootstrap e disponibilizados no Titan Extensions para reutilização, de forma que o engenheiro de aplicação possa alterar o layout das páginas sem a necessidade de codificação.

4.4.3 Processo de Geração do FrontEnd

O processo de geração do Titan FrontEnd é realizada em seis etapas, conforme ilustrado na Figura 4.3 e detalhadas a seguir.

Etapa 1 (Zend Configuration): na primeira etapa o engenheiro de aplicação informa o diretório de instalação do Zend Framework e o *Virtual Host* que responde por ele. O Titan FrontEnd irá verificar se o Zend já está instalado no diretório informado, caso não haja nenhuma instalação, o Titan FrontEnd faz o *download* e instala

automaticamente o Zend Framework no diretório informado pelo engenheiro de aplicação (Figura 4.16);

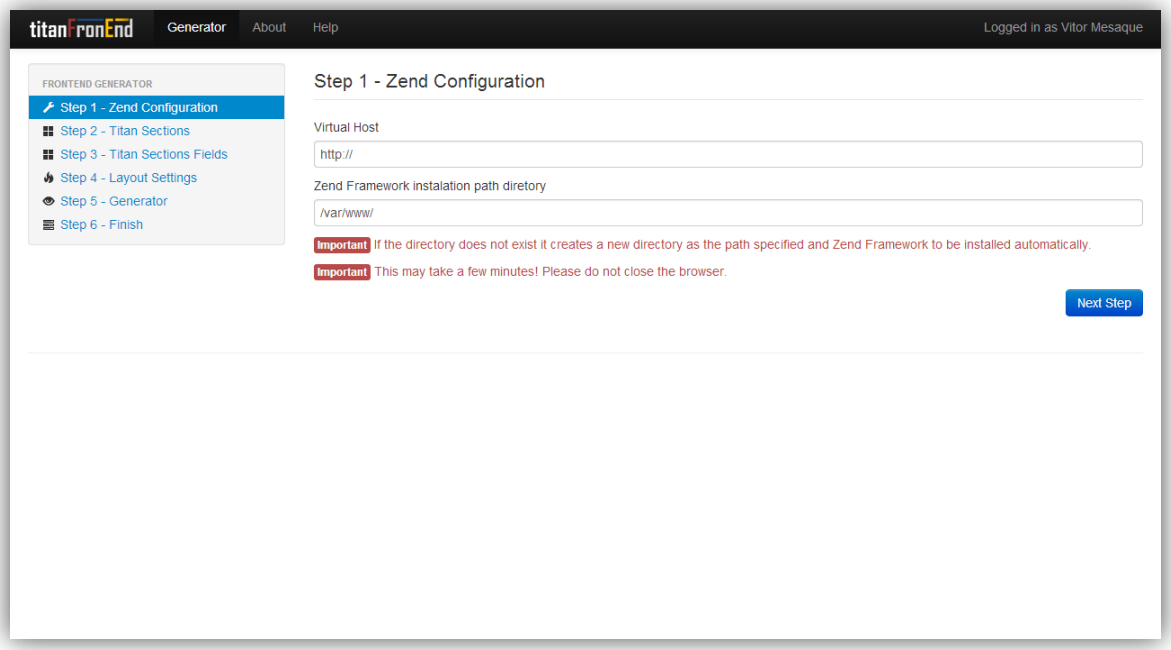

Figura 4.16: Tela Inicial do Titan Frontend.

Etapa 2 (Titan Sections): na segunda etapa o Titan FrontEnd lista todas as seções encontradas no diretório *section* da instância do Titan Framework (bloco *Back-End*), possibilitando que o engenheiro de aplicação selecione as seções que serão geradas para compor o bloco *Front-End* (Figura 4.17);

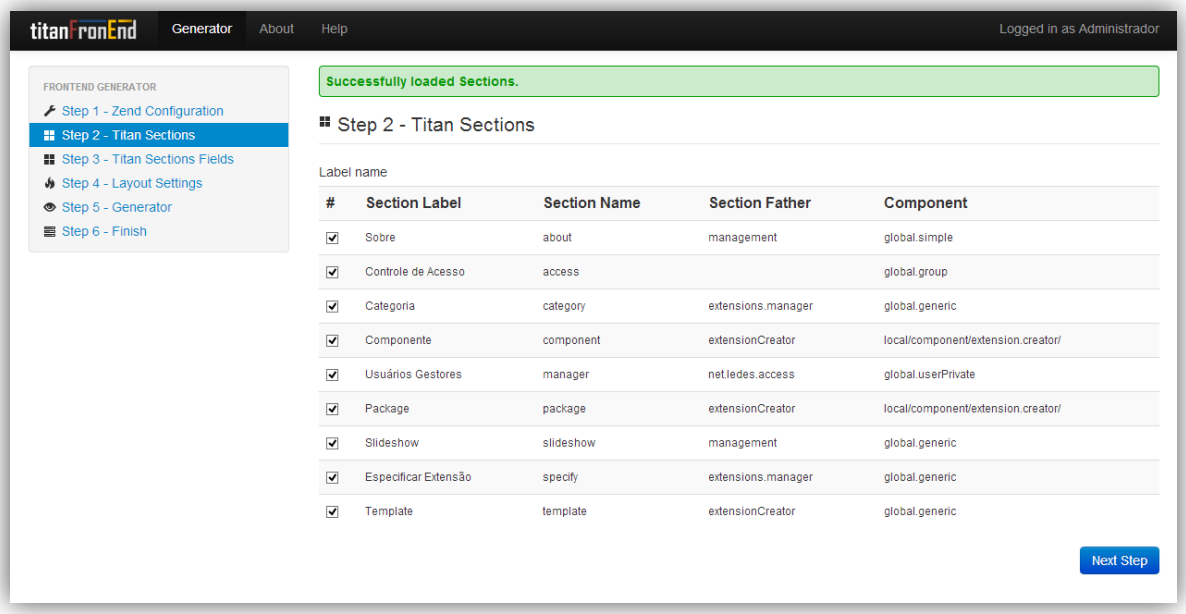

Figura 4.17: Seleção das seções que serão geradas.

Etapa 3 (Titan Sections Field): na terceira etapa as seções selecionadas anteriormente são detalhadas ao engenheiro de aplicação, possibilitando a escolha de quais atributos/dados serão gerados para as ações de listagem e visualização (Figura 4.18);

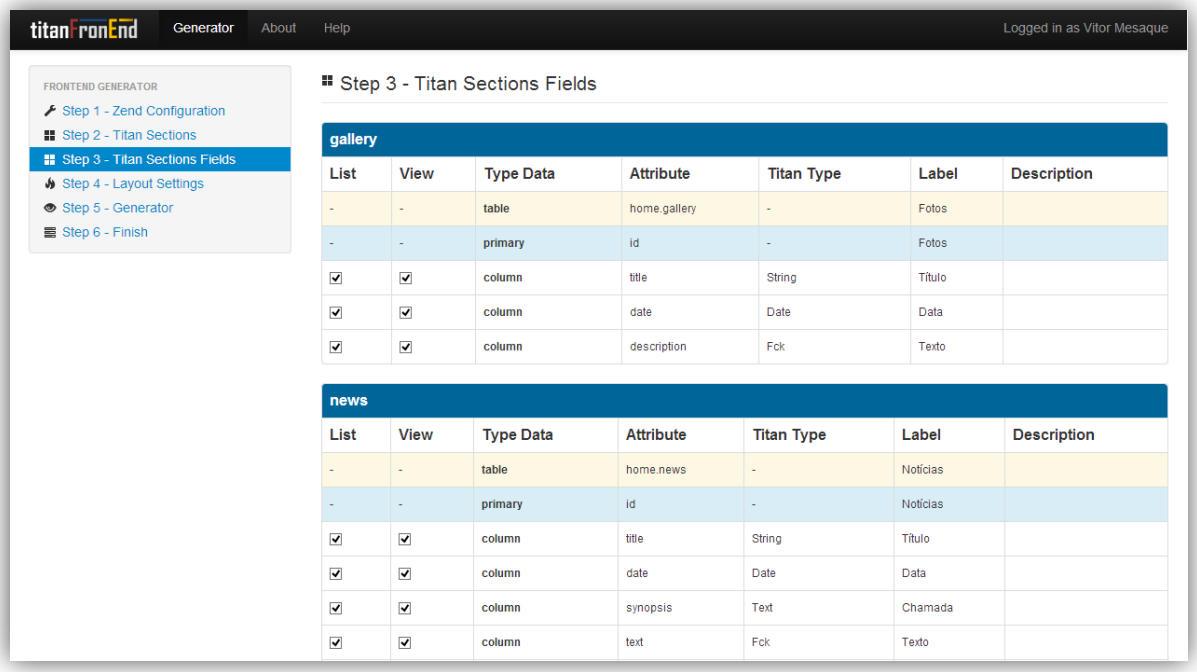

Figura 4.18: Seleção de atributos que serão gerados.

Etapa 4 (Layout Settings): na quarta etapa o engenheiro de aplicação escolhe as opções de layout, selecionando o *template* que será renderizado no momento da

geração. O Titan FrontEnd permite a escolha de templates pré-definidos e a importação de novos *templates* providos pelo Titan Extensions (Figura 4.19);

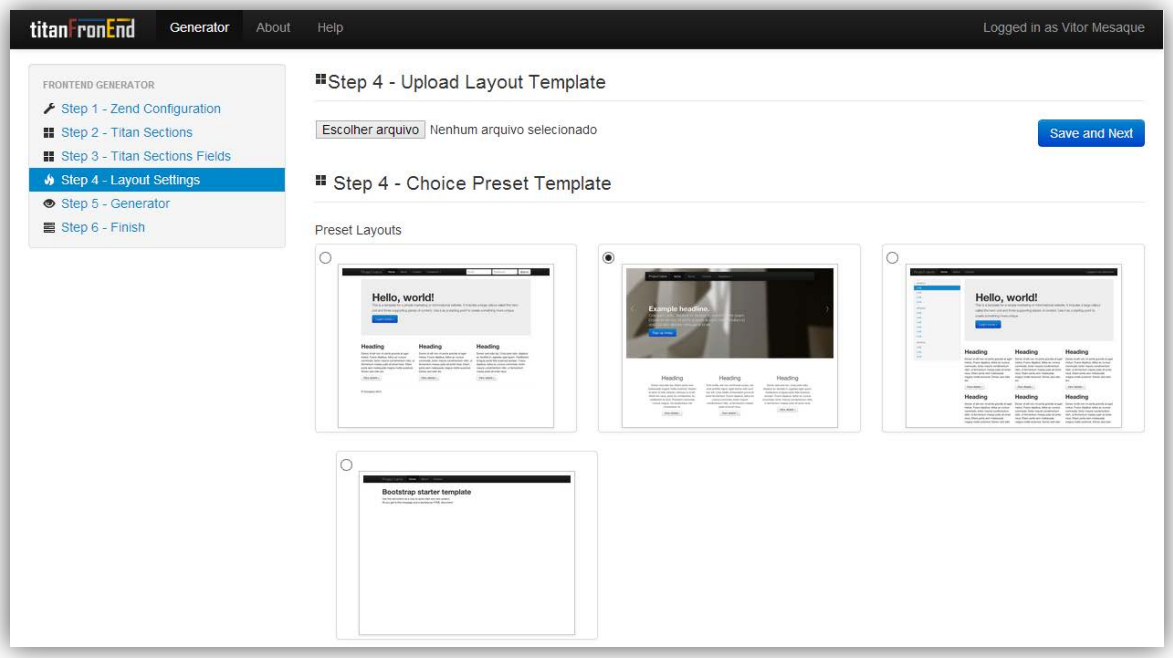

Figura 4.19: Selação do template utilizado na renderização da interface.

Etapa 5 (Generator): após todas as configurações realizadas, o Titan FrontEnd exibe um resumo de informações importantes ao engenheiro de aplicação referentes ao processo de geração, solicitando a autorização da geração (Figura 4.20);

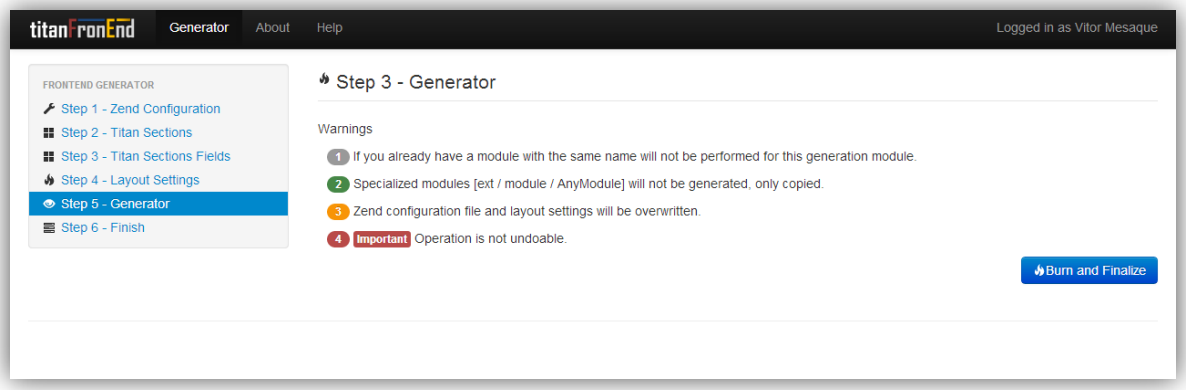

Figura 4.20: Informações sobre a geração.

Etapa 6 (Finish): por fim, o sistema exibe um log das operações realizadas durante o processo de geração, e permite a pré-visualização da aplicação (Figura 4.21).

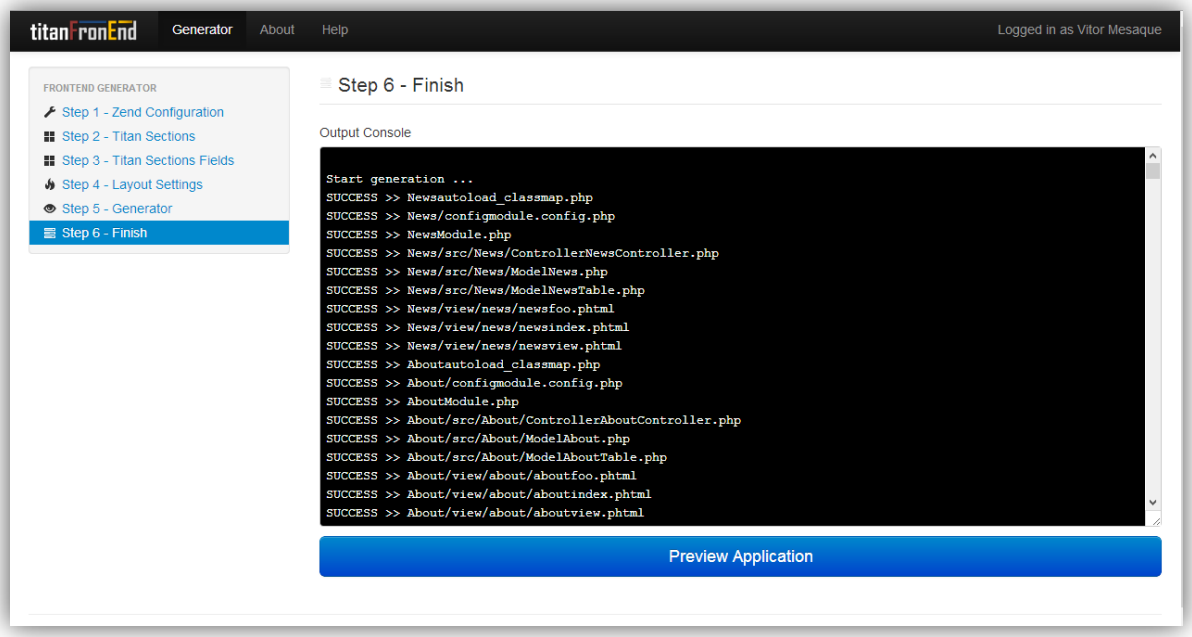

Figura 4.21: Log exibido na última etapa do processo de geração.

A cada etapa do processo de geração, as verificações e validações necessárias relacionadas às especificações de geração são realizadas de forma a garantir um processo seguro e livre de falhas.

Inicialmente, como forma de validar o gerador de aplicações Titan FrontEnd, o processo de geração do bloco *Front-End* foi realizado para o o repositório Titan Extensions (Visão Consumidor). Ao final o processo de geração, clicando na opção "*Preview Application*", o resultado exibido pode ser visualizado nas Figuras (4.0.22, 4.0.23 e 4.0.24).

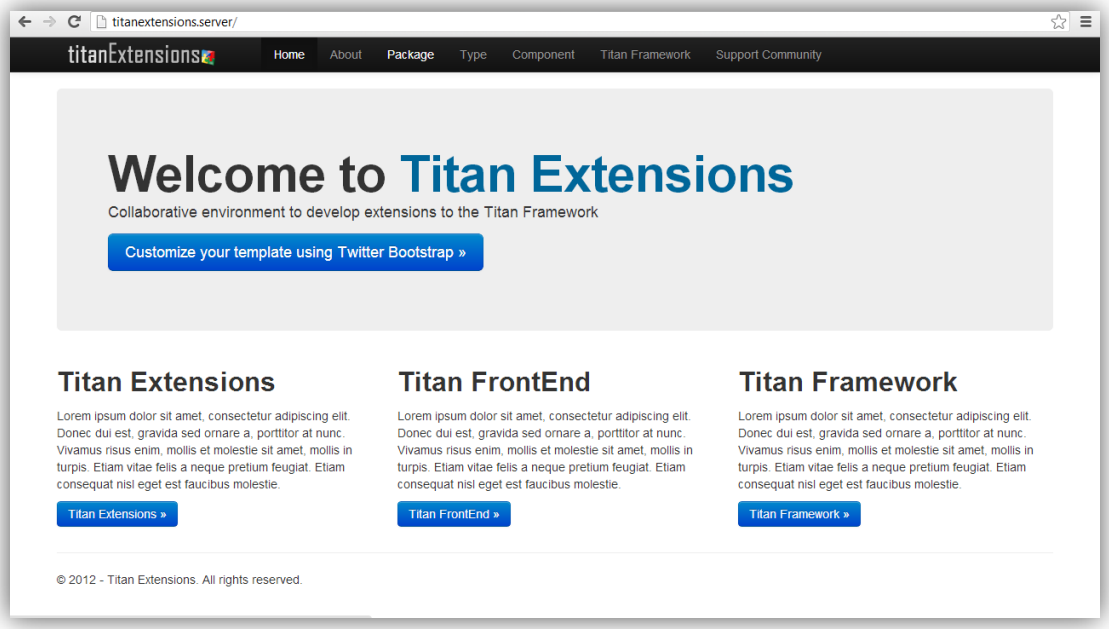

Figura 4.22: Página inicial do Titan Extensions gerada pelo Titan FrontEnd.

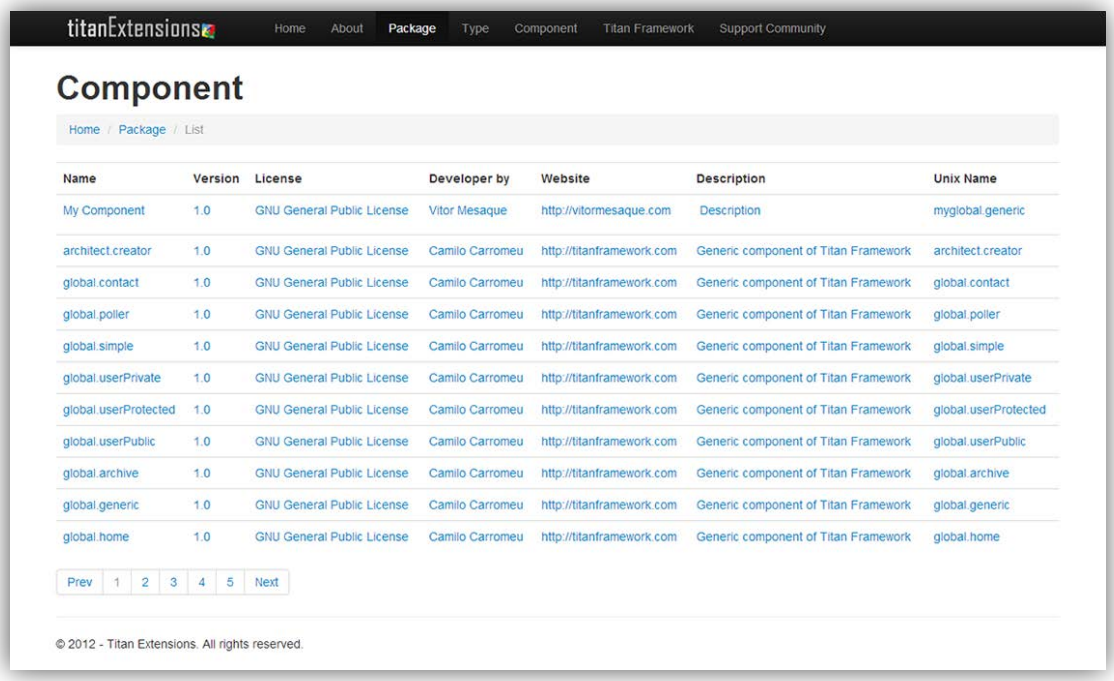

Figura 4.23: Listagem padrão de itens de um módulo.

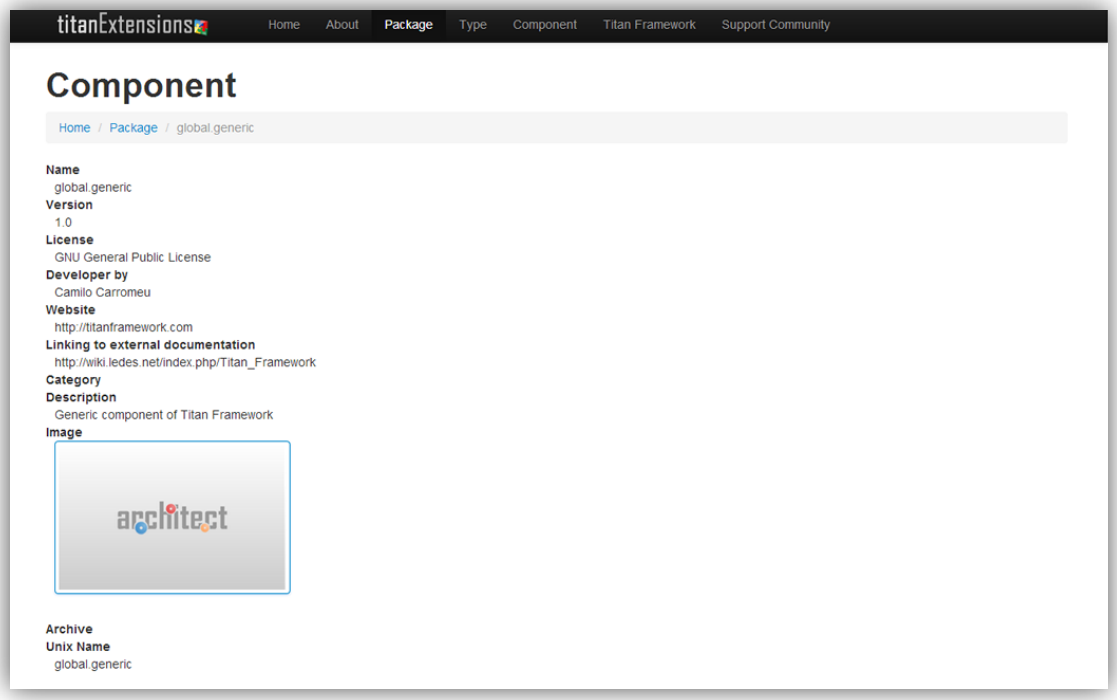

Figura 4.24: Visualização padrão de um item gerado pelo Titan FrontEnd.

É importante salientar que os exemplos explorados neste capítulo apresentam uma interface na língua inglesa, porém o Titan FrontEnd, assim como o Titan Extensions, foram projetados para suportar a internacionalização, possibilitando a exibição na linguagem escolhida.

4.4.4 Arquitetura MVC das Aplicações Geradas

O *Front-End* gerado segue o padrão de projeto de software MVC, que separa a aplicação nas camadas de modelagem de dados (*Model*), interface gráfica (*View*) e controle de fluxo da aplicação (*Controller*), conforme ilustradas na Figura 4.25. No modelo MVC, o usuário interage com a *View*, enquanto o *Controller* espera até que algum evento ocorra. Quando um evento ocorre, o *Controller* é notificado e se responsabiliza por processar aquele evento e invocar alterações no *Model*. Após a atualização do *Model*, o mesmo estará em um estado consistente para a utilização pela *View*, reiniciando o fluxo (GAMMA, 1995).

Um dos objetivos deste trabalho é garantir que as aplicações geradas pela plataforma atendam aos requisitos de acessibilidade, sendo a camada de visão (*View*) o ambiente onde as soluções para este fim devem ser aplicadas.

A seguir são apresentados os mecanismos utilizados para garantir a acessibilidade nos

portais e-Gov gerados pela plataforma.

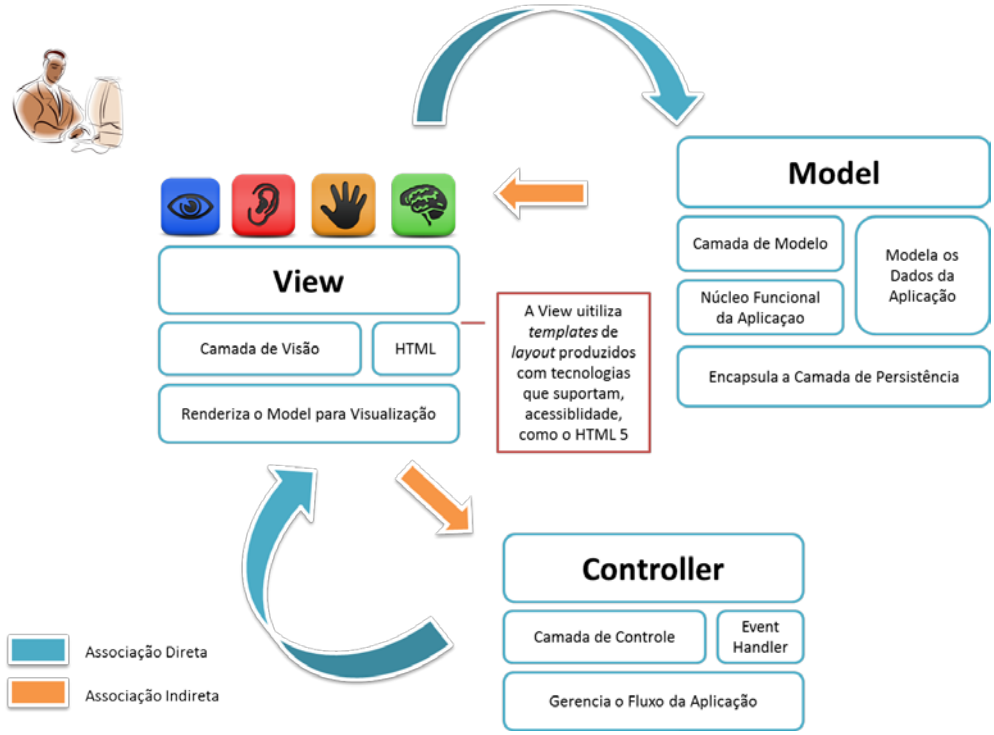

Figura 4.25: Arquitetura MVC da aplicação gerada pelo Titan FrontEnd.

4.4.5 Acessibilidade nos Portais e-Gov instanciados pela LPS

Os resultados obtidos por meio do mapeamento sistemático conduzido durante o desenvolvimento deste trabalho evidenciam as dificuldades em atingir níveis ideais de acessibilidade em CMS. Mesmo que a automação do processo seja realizada, gerando código acessível, os estudos mostram que parte da responsabilidade é transferida para os usuários finais não técnicos, gerenciadores dos CMSs, e *web designers* que produzem os *templates* de *layout* para os CMSs. Foi possível evidenciar também que nenhum dos CMSs investigados implementam as tecnologias: HTML5 e CSS3.

Nesse contexto, a plataforma proposta neste trabalho provê uma solução híbrida, que combina utilização de mecanismos automatizados, verificações de acessibilidade e o enfoque na utilização das tecnologias HTML5 e CSS3, permitindo alcançar níveis satisfatórios de acessibilidade e possibilitando *layouts* customizáveis e flexíveis.

As novas versões do HTML e CSS, utilizadas no processo de geração dos portais provêem páginas mais semânticas e acessíveis, proporcionando mais interatividade sem a necessidade de instalação de *plugins* e perda de desempenho. Permite a criação de código

interoperável, pronto para futuros dispositivos e que facilita a reutilização da informação de diversas formas (W3C BRASIL, 2012).

A seguir são apresentados os mecanismos utilizados para garantir a acessibilidade nos portais criados a partir da instanciação da LPS. É importante ressaltar que os mecanismos propostos neste trabalho devem ser adotados em sua totalidade durante toda a instanciação da LPS.

4.4.6 Geração de Código Acessível

O Titan FrontEnd gera código utilizando o Twitter Bootstrap como *framework* de estilos, que por sua vez implementa HTML5 e CSS3. O código inicial gerado pelo Titan FrontEnd a partir dos *layout*s pré-definidos pelo gerador é acessível, garantindo a inexistência de erros que inviabilizem o acesso ao conteúdo. Porém, a utilização de *templates* de *layout* customizados pode ocasionar a inserção de erros de acessibilidade no código. Para isso, um mecanismo de verificação de ativos é proposto, descrito na próxima subseção.

4.4.7 Ativos acessíveis no repositório Titan Extensions

O Titan Extensions fornece artefatos de software durante todo o processo de instanciação da LPS, sendo, portanto, um agente fundamental para a promoção de artefatos acessíveis. Uma forma de garantir acessibilidade nos portais instanciados pela LPS é garantir que todos os ativos reutilizáveis disponíveis no repositório tenham suporte à acessibilidade.

Dessa forma, foi implementado um mecanismo de verificação de acessibilidade nos ativos do repositório. Sempre que um novo ativo é disponibilizado no repositório, caso contenha arquivos HTML, o mesmo é verificado automaticamente para detectar a versão do HTML utilizada. Apenas códigos escritos em HTML5 serão aceitos no repositório, estimulando o desenvolvimento de páginas mais semânticas e acessíveis.

Algumas pesquisas foram realizadas no decorrer deste trabalho procurando formas de validar automaticamente e remotamente a acessibilidade de um código por meio de *web services*. Algumas soluções implementadas como *web crawlers* foram encontradas, mas no âmbito deste trabalho não foram utilizadas. O Governo Federal ainda não disponibilizou nenhum serviço permitindo a validação por meio de *webservice* ao avaliador ASES (ASES, 2012), que faz validações de acordo com as diretrizes do e-Mag. Dessa forma, um mecanismo

manual foi adotado inicialmente. Os códigos devem ser validados em alguma ferramenta de validação por um usuário moderador do repositório.

Adicionalmente, dois serviços foram implementados no repositório Titan Extensions, são eles:

- Mecanismo de Denúncia: todo e qualquer ativo do repositório Titan Extensions poderá receber avaliações quanto à acessibilidade por meio dos usuários consumidores;
- Mecanismo de Habilitação: todo e qualquer ativo disponibilizado no repositório Titan Extensions só será liberado para utilização após a validação de acessibilidade do mesmo.

4.5 Considerações Finais

Neste capítulo foi apresentada a plataforma de LPS proposta, bem como a sua arquitetura e as ferramentas que a compõe. Foram apresentados também os mecanismos adotados no suporte a acessibilidade das aplicações instanciadas pela LPS.

Em síntese, o processo de instanciação da LPS é iniciado com a seleção das extensões no repositório Titan Extensions. A partir da análise de requisitos de um novo portal e-Gov, o engenheiro de aplicação faz a verificação de quais ativos disponíveis no repositório Titan Extensions atendem aos requisitos. Caso algum requisito não seja atendido em sua totalidade pelos ativos disponíveis no repositório, o engenheiro de aplicação inicia o desenvolvimento de um novo ativo e o disponibiliza no repositório, continuando o processo. Assim, as extensões selecionadas, somadas aos artefatos nativos do *framework* Titan, fornecem os parâmetros de entrada necessários para o Titan Architect. Com uma interface gráfica subdividida em passos, o Architect permite a parametrização e a instanciação do *Back-End* de um portal de forma automática, resultando em um CMS, sendo uma instância do Titan Framework. Após a geração do *Back-End*, o Titan FrontEnd pode ser utilizado para a geração automática do *Front-End* do portal. Dessa forma, semelhantemente ao Architect, o Titan FrontEnd fornece um processo de geração subdividido em passos de configuração. A acessibilidade das páginas geradas é garantida por meio de mecanismos de automação e verificação.

Com a automação provida pela plataforma da LPS proposta, é possível gerar de forma rápida e fácil novos produtos no domínio de portais e-Gov acessíveis. A plataforma

apresentada nesse capítulo automatiza a fase de Engenharia da Aplicação, guiando o engenheiro de aplicação no processo de geração das aplicações, garantindo que os caminhos certos sejam seguidos, fazendo as devidas consistências para assegurar que os processos e componentes aplicados sejam coerentes entre si. Portanto, o engenheiro tem condições de saber exatamente onde começar e onde terminar a instanciação.

Capítulo 5

Avaliação da Plataforma

5.1 Considerações Iniciais

Com o objetivo de avaliar a plataforma proposta neste trabalho foi conduzida uma pesquisa com vinte e um participantes por meio de um questionário de avaliação. Todos os participantes são especialistas na área de desenvolvimento Web e trabalham em projetos de quatro instituições: 1) LEDES/UFMS; 2) PLEASE Lab/Embrapa; 3) Unimed; e 4) SGI (Superintendência de Gestão da Informação)/SEFAZ-MS (Secretaria de Estado de Fazenda de Mato Grosso do Sul).

Neste capítulo são apresentados os resultados da pesquisa de avaliação. Na Seção 5.2 é descrita a metodologia utilizada para a condução da avaliação. Na Seção 5.3 é apresentada uma síntese da análise dos dados obtidos a partir dos dados coletados. Na Seção 5.4 as limitações do estudo são avaliadas e, por fim, as considerações finais deste capítulo são apresentadas na Seção 5.5.

5.2 Metodologia

Inicialmente foi realizada uma avaliação piloto com três participantes, a fim de identificar e eliminar ambiguidade e incoerências no questionário. Algumas adaptações feitas após a realização da avaliação piloto.

Antes da aplicação do questionário, a proposta de trabalho foi apresentada aos participantes e os principais conceitos teóricos utilizados foram explanados. A apresentação durou aproximadamente sessenta minutos, incluindo a demostração e exploração de exemplos práticos da utilização da plataforma.

A avaliação foi conduzida por meio de um questionário de avaliação contendo questões abertas e de múltiplas escolhas organizadas em três categorias: a) perfil do especialista; b) entendendo a situação atual; e c) avaliação da plataforma, conforme apresentado no Apêndice C.

5.3 Análise dos Dados

Para a análise dos dados foram utilizados métodos de síntese estatística e os resultados foram compilados em gráficos e tabelas separados em categorias para a melhor visualização e compreensão dos dados. O questionário, juntamente com todas as respostas dos participantes, podem ser visualizados no apêndice C e D, respectivamente. A seguir é apresentada uma síntese da análise dos dados considerados relevantes para a pesquisa.

Titan Extensions

No que se refere às extensões disponibilizadas no Titan Extensions para construção de novas WebApps, 62% dos desenvolvedores afirmaram que definitivamente as utilizariam e 33% provavelmente utilizariam. Apenas 5% provavelmente não utilizaria e nenhum entrevistado afirmou que definitivamente não utilizaria, como mostrado na Figura 5.1.

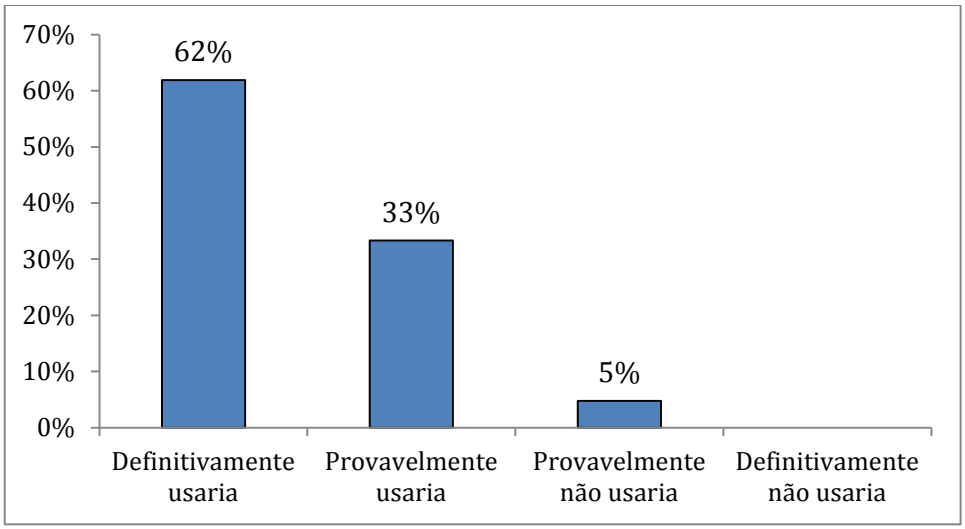

Figura 5.1 - Utilização das extensões disponibilizadas no Titan Extensions na construção de novas WebApps.

Fonte: elaborado pelo autor a partir dos dados coletados.

Na Figura 5.2, é possível visualizar que, quando os desenvolvedores foram perguntados se eles utilizariam as extensões disponibilizadas no Titan Extensions na construção de novas WebApps, 95% dos entrevistados afirmaram que usariam. Dentre os que afirmaram que usariam 52% definitivamente usariam e 43% provavelmente usariam. Apenas 5% dos entrevistados não utilizariam as extensões disponibilizadas no Titan Extensions.

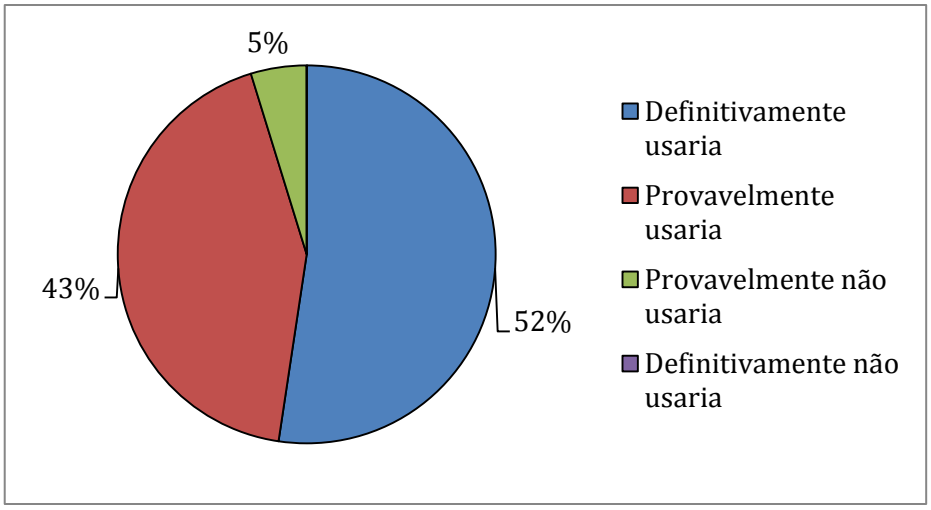

Figura 5.2 – Utilização do Titan FrontEnd para gerar o Front-End de uma WebApp.

Fonte: elaborado pelo autor a partir dos dados coletados.

Titan FrontEnd

A Figura 5.3 ilustra a avaliação do gerador Titan FrontEnd, por parte dos desenvolvedores. Constatou-se que a maioria dos entrevistados avaliou bem o Titan FrontEnd, 48% consideraram excelente, 48% consideraram como sendo muito bom e 5% como sendo bom. Nenhum desenvolvedor avaliou o Titan FrontEnd como sendo razoável ou ruim.

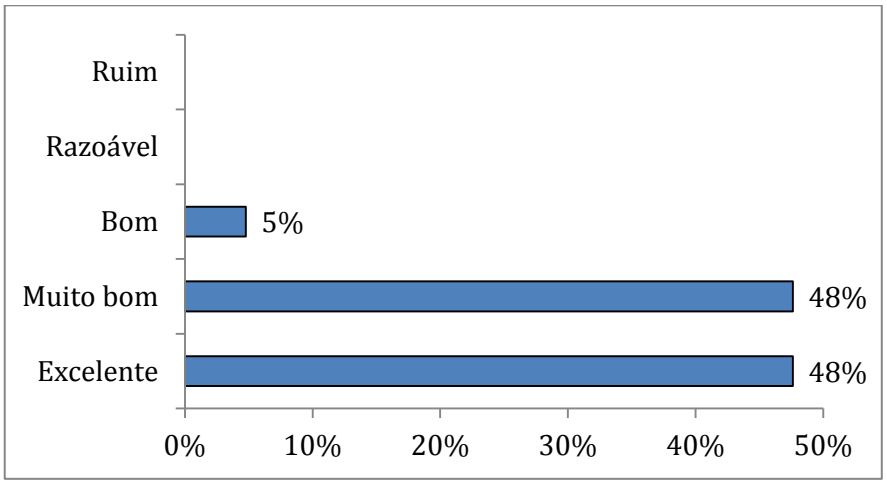

Figura 5.3 - Avaliação do gerador Titan FrontEnd.

Fonte: elaborado pelo autor a partir dos dados coletados.

No que tange a avaliação dos desenvolvedores em relação ao repositório Titan Extensions, 52% dos entrevistados considerou esse quesito como sendo excelente, 33% disseram que o Titan Extensions é muito bom e 10% o avaliaram como sendo bom. Assim

como na avaliação do Titan FrontEnd, nenhum desenvolvedor avaliou o Titan Extensions como sendo razoável ou ruim. Os resultados podem ser visualizados na Figura 5.4.

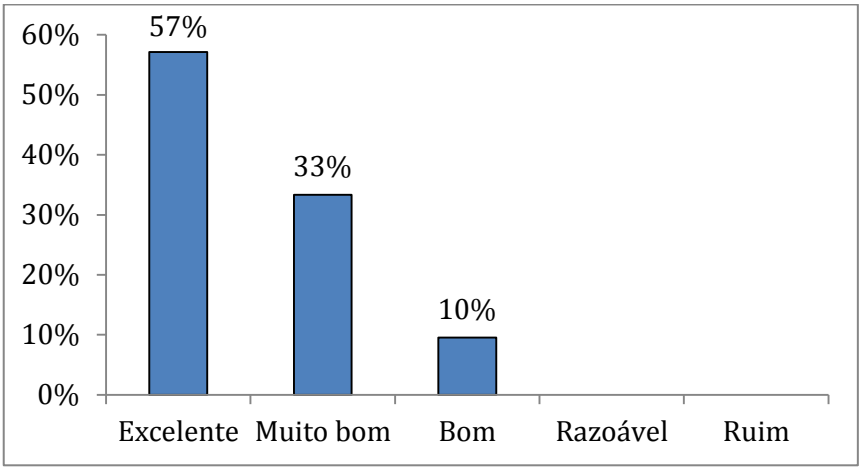

Figura 5.4 - Avaliação do repositório Titan Extensions.

Fonte: elaborado pelo autor a partir dos dados coletados.

Acessibilidade

Quando os entrevistados foram questionados sobre a preocupação com as diretrizes de acessibilidade na construção do websites/portais apenas 5% relataram que se preocupam totalmente, enquanto 38% não se preocupam. Do total de entrevistados, 14% declararam nem conhecer essas diretrizes de acessibilidade, e 43% conhece parcialmente. Conforme apresenta a Tabela 5.1.

Tabela 5.1: Preocupação com as diretrizes de acessibilidade ao construir um website/portal.

Alternativas	Percentagem
Não	38%
Não conheço as diretrizes de acessibilidade	14%
Sim, parcialmente	43%
Sim, totalmente	5%
Total	100%

Fonte: elaborado pelo autor a partir dos dados coletados.

Com relação à acessibilidade das WebApps geradas, os especialistas foram questionados se eles acreditam que a utilização da plataforma proposta traria benefícios nesse sentido. Todos os participantes responderam que sim e dentre os que deram essa resposta 67% relataram que provavelmente utilizariam e 33% que definitivamente utilizariam. Os resultados podem ser visualizados na Figura 5.5.

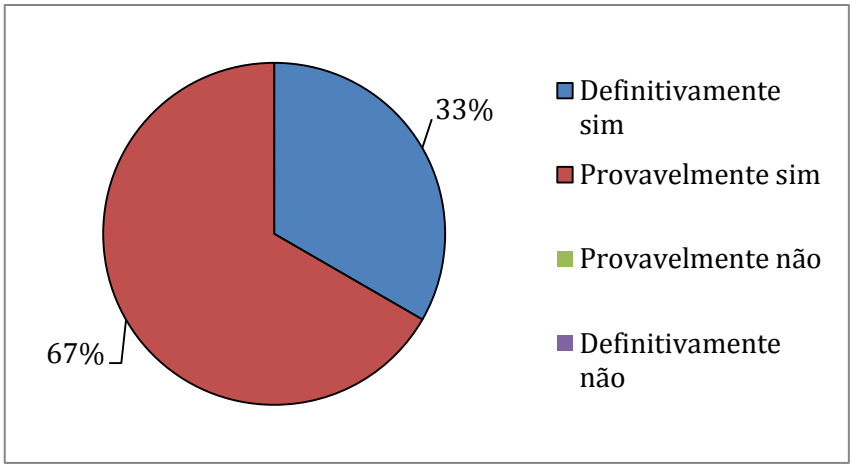

Figura 5.5 - Benefícios com relação à acessibilidade das WebApps geradas pela utilização da plataforma.

Fonte: elaborado pelo autor a partir dos dados coletados.

Plataforma da LPS

A Figura 5.6 ilustra a resposta dos entrevistados quando foram questionados se usariam em seus projetos a plataforma apresentada. Verificou-se que a maioria dos entrevistados usaria a plataforma, 43% afirmaram que provavelmente usariam e 38% disseram que definitivamente usariam. Apenas 5% dos desenvolvedores disseram que provavelmente não usariam e 14% relataram que podem ou não usar.

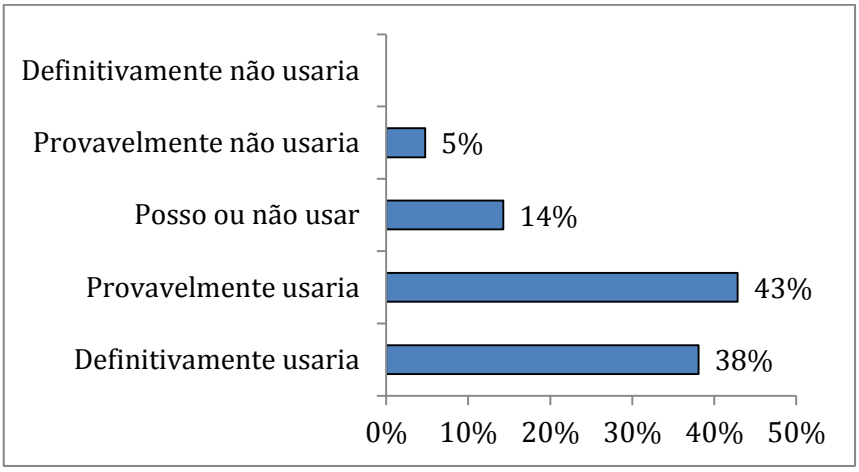

Figura 5.6 – Utilização da plataforma apresentada no projeto dos entrevistados.

Fonte: elaborado pelo autor a partir dos dados coletados.

De uma maneira geral os desenvolvedores avaliaram bem a plataforma de LPS na geração de portais e-Gov Acessíveis. A Figura 5.7 mostra que 62% dos entrevistados consideraram a plataforma como sendo muito boa e 38% como sendo excelente. Nenhum desenvolvedor avaliou essa plataforma como sendo razoável ou ruim.

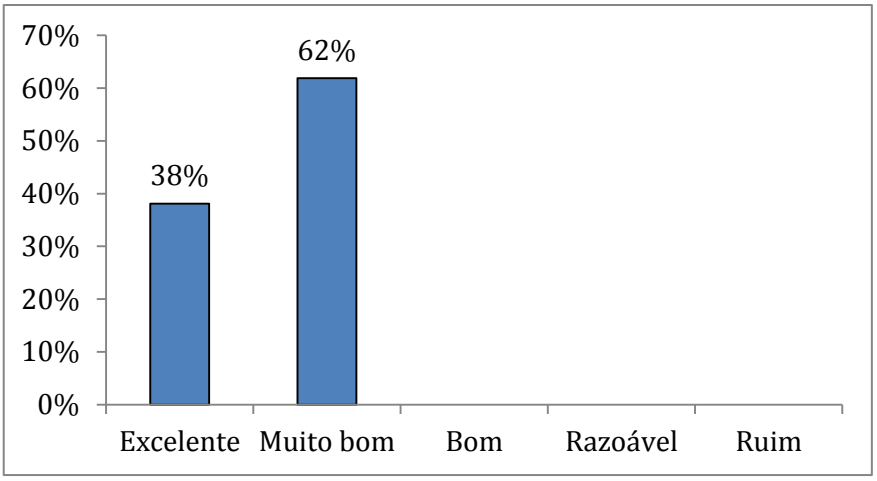

Figura 5.7 - Avaliação da plataforma de LPS na geração de portais e-Gov Acessíveis.

Fonte: elaborado pelo autor a partir dos dados coletados.

5.4 Limitações da Avaliação

Algumas limitações foram identificadas na pesquisa de avaliação do trabalho, entre elas pode-se destacar: i) os participantes não utilizaram ferramentas apresentadas na prática; e ii) o repositório Titan Extensions ainda não apresenta ativos reutilizáveis reais disponíveis, limitando a demonstração de algumas funcionalidades do repositório na prática.

5.5 Considerações Finais

Neste capítulo foram apresentados os resultados obtidos por meio de uma pesquisa conduzida com o intuito de avaliar a plataforma de LPS no domínio de portais e-Gov acessíveis.

Apesar das limitações da avaliação, o resultado final foi satisfatório e pontos importantes foram identificados. Primeiramente, o estudo mostrou que apenas 5% dos especialistas estão totalmente preocupados com relação à acessibilidade no desenvolvimento de suas aplicações. Questionados quanto à dificuldade de se construir uma WebApp acessível, a maioria dos especialistas alegaram a dificuldade na sua construção, assim como a falta de ferramentas automatizadas e a pouca familiaridade com as diretrizes de acessibilidade.

Conforme mostram os dados, foi constatado que, de uma maneira geral, as ferramentas foram muito bem avaliadas e de fato poderiam ser utilizadas na prática. Com relação as vantagens que a plataforma traria ao trabalho diário dos especialistas, foi evidenciado na maioria das respostas que a utilização da plataforma poderia levar a um aumento significativo

na produtividade e na qualidade dos produtos, bem como a redução no custo e esforço e aumento no reúso.

Capítulo 6

Conclusão

5.6 Considerações Iniciais

As conclusões apresentadas neste capítulo estão organizadas da seguinte maneira: na Seção 6.2 são apresentadas as contribuições observadas com a realização deste trabalho; na Seção 6.3 são discutidas as principais limitações observadas no trabalho e na Seção 6.4 os trabalhos futuros relacionados ao trabalho desenvolvido são elencados.

5.7 Contribuições

O presente trabalho propôs uma solução eficiente para a construção de portais e-Gov acessíveis, por meio de uma plataforma de LPS automatizada, combinando a utilização de *frameworks*, geradores de aplicação e repositório de reúso, tornando o processo de instanciação de portais e-Gov mais ágil e menos suscetível a erros.

Os resultados deste trabalho consolidam um processo de desenvolvimento para o domínio de portais e-Gov acessíveis, reaproveitando de forma sistemática os esforços e recursos despendidos na construção e evolução dos portais por meio de uma plataforma de LPS automatizada. Os artefatos gerados nesse processo são armazenados no repositório de reúso Titan Extensions e reutilizados no desenvolvimento de novos portais e-Gov acessíveis.

O gerador de aplicações Titan FrontEnd apresenta uma solução automatizada, reduzindo significativamente os custos associados ao desenvolvimento do *Front-End* de um portal e garantindo um processo de geração sistemático e seguro. O repositório Titan Extensions por sua vez, é um agente fundamental na plataforma da LPS proposta, alimentando todo o processo de desenvolvimento, provendo artefatos de software reutilizáveis e potencializando a reutilização e a qualidade dos artefatos durante a instanciação da LPS. As tecnologias utilizadas e os mecanismos implementados garantem a geração de um código acessível, reduzindo as barreiras de acessibilidade nos portais gerados.

A LPS explorada neste trabalho é coerente com as diretrizes de Governo Eletrônico brasileiro, apresentando uma plataforma totalmente *open source*, proporcionando a racionalização de recursos e o desenvolvimento colaborativo, reduzindo o custo e o esforço na construção de novos portais e-Gov e permitindo o compartilhamento de artefatos de software desenvolvidos no processo de instanciação da LPS. Os portais gerados pela plataforma promoverão a cidadania, inclusão digital e gestão de conhecimento, oferecendo serviços e informações aos cidadãos.

5.8 Limitações

As seguintes limitações foram identificadas em relação à plataforma de LPS proposta e consequentemente a este trabalho de mestrado:

- Não foi possível aplicar a plataforma da LPS em cenários reais para avaliar o seu desempenho;
- O Titan FrontEnd em sua versão inicial não permite configurações avançadas em relação ao *layout* de interface gerado automaticamente;
- O Titan Extensions não permite a desinstalação automática de uma extensão, o que leva ao processo manual de exclusão de uma extensão instalada;
- O mecanismo de controle de acessibilidade apresentado precisa ser testado na prática para identificar melhorias no processo; e
- O mecanismo de verificação de acessibilidade dos ativos do repositório Titan Extensions não é totalmente automatizada.

5.9 Trabalhos Futuros

Abaixo estão elencados sugestões de trabalhos futuros decorrentes do desenvolvimento deste trabalho:

- Avaliar a plataforma proposta em um cenário real, com o objetivo de medir o nível de reúso, coletando métricas sobre a utilização dos artefatos de software reutilizáveis disponíveis no repositório;
- Explorar novas tecnologias para melhorar o processo de geração de portais e-Gov acessíveis; e

- Implementar uma ferramenta de geração automática de manuais técnicos da aplicação, visando apoiar o processo de manutenção e evolução da LPS. Desse modo, sempre que uma nova aplicação for criada, o manual técnico será gerado automaticamente em tempo de execução, ou seja, havendo alguma alteração na instância a documentação será atualizada automaticamente. Atualmente o *framework* Titan já possibilita a geração automática de manual do usuário da aplicação.

Referências

ALMEIDA, E.; ALVARO, A.; LUCREDIO, D.; GARCIA, V.; MEIRA, S. *RiSE Project: Towards a Robust Framework for Software Reuse*, IEEE-IRI-2004, Las Vegas, USA, Novembro, p. 48-53, 2004.

APACHE. *Apache Subversion*. Disponível em http://subversion.apache.org/. Acesso em: 2 de Jul. 2012.

APPERLY, H. *Component-Based Software Engineering: Putting the Pieces Together*. Addison Wesley, 2001.

ASES. *Avaliador e Simulador de Acessibilidade de Sítios*. Disponível em <http://www.governoeletronico.gov.br/acoes-e-projetos/e-MAG/ases-avaliador-e-simulador-de-acessibilidade-sitios>. Acesso em: 30 de Mai. 2012.

BALZERANI, L.; DI RUSCIO, D.; PIERANTONIO, A.; DE ANGELIS, G. *A product line architecture for web applications*. ACM Symposium on Applied Computing, 2005.

BANNISTER, F. *The curse of the benchmark: An assessment of the validity and value of e-Government comparisons*. International Review of Administrative Sciences, v. 73, n. 2, p. 171–188, 2007.

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.

BOEHM, B. *A Spiral Model of Software Development and Enhancement*. IEEE Computer 21(5): 61-72, 1988.

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

BURBECK, S. *Application Programming in Smalltalk-80: How to use Model-View-Controller (MVC).* Smalltalk Archive, 1987, 1992. Disponível em <http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html>. Acesso em 26 de Jun. 2012.

BURÉGIO, V. A.; ALMEIDA, E. S. D.; LUCRÉDIO, D.; MEIRA, S. R. D. L. *Specification, Design and Implementation of a Reuse Repository*. COMPSAC, v. 1, p. 579-582, 2007.

BURZAGLI, L.; GABBANINI, F.; NATALINI, M.; PALCHETTI, E.; AGOSTINI, A. *Using Web Content Management Systems for Accessibility: The experience of a Research Institute Portal*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 5105 LNCS, p. 454-461, 2008.

CARROMEU, C. *Linha de Produtos de Software no Processo de Geração de Sistemas Web de Apoio à Gestão de Fomento de Projetos*. Dissertação de Mestrado em Ciência da Computação. Departamento de Computação e Estatística. Universidade Federal de Mato Grosso do Sul. Campo Grande, MS, Brasil, 2007.

CARROMEU, C.; PAIVA, D.M.B.; CAGNIN, M.I.; RUBINSZTEJN, H.K.S.; TURINE, M.A.S.; BREITMAN, K. *Component-Based Architecture for e-Gov Web Systems Development*. Engineering of Computer Based Systems (ECBS), 17th IEEE International Conference and Workshops, p.379-385, 2010.

CARROMEU, C.; VIEIRA, C. A. *Instanciação de Sistemas de Informação para Educação a Distância com o Framework Titan*. Trabalho de Conclusão do Curso de Especialização em Planejamento e Tutoria em EAD. Universidade Federal de Mato Grosso do Sul - UFMS, 2009.

CASTELLS, M. *A Galáxia da Internet – reflexões sobre a internet, os negócios e asociedade*. Rio de Janeiro. Jorge Zahar, 2003.

CC. *Creative Commons*. Disponível em http://creativecommons.org.br/as-licencas/. Acesso em: 10 de Jan. 2013.

CGI.br (Comitê Gestor da Internet no Brasil); NIC.br (Núcleo de Informação e Coordenação do Ponto BR). *Dimensões e características da web brasileira: um estudo do .gov.br*. Brasília, 2010. Disponível em: <http://www.cgi.br/publicacoes/ pesquisas/govbr/cgibrnicbr-censoweb-govbr-2010.pdf>. Acesso em: 24 de Jul. 2012.

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

CHEONG, F. *Internet Agents: Spiders, Wanderers, Brokers, and Bots*. New Riders Publishing, Indianapolis, IN, USA, 1996.

CIMAN, M.; GAGGI, O.; SBRIGNADELLO, M. *Toward the Creation of a Green Content Management System*. WEBIST 2011 - Proceedings of the 7th International Conference on Web Information Systems and Technologies. Anais...2011.

CLEAVELAND, J. C. *Building application generators*. IEEE Software, p. 25-33, 1988.

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, 2002.

COHEIN S.; HESS J.; KANG K.; NOVAK W.; PETERON A. *Feature-oriented domain analysis (FODA) feasibility study*. Pittsburgh, Pennsylvania, USA, 1990.

CURTIN, G. *Encyclopedia of political communications*. Thousand Oaks, CA: Sage Publications, 2007.

CZARNECKI, K; EISENERCKER, U. W. *Generative Programming*. Addison-Wesley, 2002.

DRUPAL. Disponível em <http://drupal.org>. Acesso em: 30 de Jul. 2012.

D'SOUZA, D. F.; WILLS, A. W. *Objects, Components and Frameworks with UML - The Catalysis Approach*. Addison-Wesley, 1999.

DYBA, T.; DINGSOYR, T.; HANSSEN, G. K. *Applying Systematic Reviews to Diverse Study Types: An Experience Report*. First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007). Anais... [S.l.]: IEEE. Set 2007.

E-GOV. *Programa de Governo Eletrônico Brasileiro*. Disponível em: < http://www.governoeletronico.gov.br>. Acesso em: 30 de Mai. 2012.

E-MAG. *e-MAG: Modelo de Acessibilidade em Governo Eletrônico*, 2011. Disponível em: < http://www.governoeletronico.gov.br/acoes-e-projetos/e-MAG>. Acesso em 30 de Mai. 2012.

E-PWG. *e-PWG: Padrões Web em Governo Eletrônico*, 2011. Disponível em: <http://www.governoeletronico.gov.br/acoes-e-projetos/padroes-brasil-e-gov>. Acesso em: 25 de Jun. 2012.

FAYAD, M. E.; JOHNSON, R. E.; SCHMIDT, D. C. *Building Application Frameworks: Object-Oriented Foundations of Framework Design*. Wiley & Sons, 1999.

FAYAD, M.; SCHMIDT, D. C. *Object-oriented application frameworks*. Commun. ACM, v. 40, n. 10, p. 32-38, 1997.

FOGLI, D. .; COLOSIO, S. .; SACCO, M. *Managing Accessibility in Local E-government Websites Through End-user Development: A case study*. Universal Access in the Information Society, v. 9, n. 1, p. 35-50, 2010.

FOGLI, D. *End-user development for E-government website content creation*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 5435 LNCS, p. 126-145, 2009.

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

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

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 Brasileira de Computação, 2002.

GOMAA, H. *Design software product lines with UML: From use cases to pattern-based software architetures*. Addison-Wesley, 2004.

GPL. *GNU General Public License*. Disponível em http://www.gnu.org/licenses/gpl-2.0.html. Acesso em: 10 de Jan. 2013.

HARPER, S.; YESILADA, Y. *Web Accessibility and Guidelines. Web Accessibility – A Foundation for Research*. Springer-Verlag London Limited. p. 61-78, 2008.

HEATON, J. *Programming Spiders, Bots, and Aggregators in Java (1st ed.)*. SYBEX Inc., Alameda, CA, USA, 2002.

IBGE. *Instituto Brasileiro de Geografia e Estatística*. Disponível em: http://www.ibge.gov.br. Acesso em: 03 de abr 2012.

IGLESIAS, A.; MORENO, L.; MARTÍNEZ, P.; CALVO, R. *Evaluating the Accessibility of Three Open-source Learning Content Management Systems: A comparative study*. Computer Applications in Engineering Education, [s.d.].

J. BAYER, O. FLEGE, P. KNAUBER, R. LAQUA, D. MUTHIG, K. SCHMID, e T. WIDEN. *Pulse: A methodology to develop software product lines*. Proceedings of the 1999 Symposium on Software Reusability. Los Angeles, California, USA: ACM Press, 1999.

JOOMLA. *Joomla!*. Disponível em http://www.joomla.org/. Acesso em: 30 de Jul. 2012.

KITCHENHAM, B. Procedures for Performing Systematic Reviews. Keele UK Keele University, v. 33, n. TR/SE-0401, p. 28, 2004.

KITCHENHAM, B.; BUDGEN, D.; BRERETON, O. P. *The Value of Mapping Studies - A Participant-observer Case Study*. Proceedings of Evaluation and Assessment of Software Engineering EASE, v. 2, n. 1, p. 1-9, 2010.

KLINT, P.; VAN DEURSEN, A. *Domain-specific language design requires feature descriptions*. Journal of Computing and Information Technology, 2002.

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.

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

LANDRE, G. *GLPN – Uma Abordagem para Gestão de Linhas de Processos de Negócios*. Dissertação de Mestrado em Ciência da Computação. Faculdade de Computação. Universidade Federal de Mato Grosso do Sul. Campo Grande, MS, Brasil, 2012.

LINDEN, F. J.; SCHMID, K.; ROMMES, E. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer-Verlag, 2007.

LÓPEZ, J. M.; PASCUAL, A.; MASIP, L.; GRANOLLERS, T.; CARDET, X. *Influence of Web Content Management Systems in Web Content Accessibility*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 6949 LNCS, n. PART 4, p. 548-551, 2011.

LÓPEZ, J. M.; PASCUAL, A.; MENDUIÑA, C.; GRANOLLERS, T. *Methodology for Identifying and Solving Accessibility Related Issues in Web Content Management System Environments*. Proceedings of the International Cross-Disciplinary Conference on Web Accessibility. Anais...New York, NY, USA: ACM, 2012. Disponível em: <http://doi.acm.org/10.1145/2207016.2207043>

LÓPEZ, J. M.; PASCUAL, A.; GRANOLLERS, A. *Engineering Accessibility in Web Content Management System Environments*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 5802 LNCS, p. 597-604, 2009.

MAIA, L. S. *Um Processo para o Desenvolvimento de Aplicações Web Acessíveis*. Dissertação de Mestrado em Ciência da Computação. Departamento de Computação e Estatística. Universidade Federal de Mato Grosso do Sul. Campo Grande, MS, Brasil, 2010.

MENDELEY. *Release Notes for Mendeley Desktop v1.6*. Disponível em: <http://www.mendeley.com/release-notes/v1_6/>. Acesso em: 2 dez. 2012.

MILI, A., MILI, R., MITTERMEIR, R.T. *A survey of software reuse libraries*. Annals of Software Engineering. v. 5, p. 349–414, 1998.

MIRRI, S.; SALOMONI, P. ROCCETTI, M. GAY, G. R. *Beyond standards: Unleashing Accessibility on a Learning Content Management System*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 6530, p. 35-49, 2011.

NACER, J. F. R. *Titan Architect: Um Wizard para Instanciação de Gerenciadores de Conteúdo utilizando o Titan Framework. Trabalho de Conclusão de Curso*. Faculdade de Computação. Universidade Federal de Mato Grosso do Sul. Coxim, MS, Brasil, 2009.

Nedba, G. *A software solution for Accessible E-government Portals*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 5105 LNCS, p. 338-345, 2008.

NEDBAL, D.; PETZ, G. *Implementation concept for an accessible web CMS*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 6179 LNCS, n. PART 1, p. 456-463, 2010.

NIRANJAN, P.; GURU RAO, C. V. *A Model Software Reuse Repository with an Intelligent Classification and Retrieval Technique*. Computer Science and Engineering. v.1, n. 1, p. 15-21, 2011.

OPENCMS. *OpenCms*. Disponível em http://www.opencms.org/en/. Acesso em: 30 de Jan. 2013.

PARK, H. M. *The Web Accessibility Crisis of the Korea's Electronic Government: Fatal Consequences of the Digital Signature Law and Public Key Certificate*. System Science (HICSS), 2012 45th Hawaii International Conference. p. 2319-2328, 4-7 Jan. 2012.

PINHO, J. A. G. *Investigando portais de governo eletrônico de estados no Brasil:muita tecnologia, pouca democracia*. Revista de Administração Pública. Rio de Janeiro - RJ. v.42, n. 3, p. 471-93, Mai/Jun 2008.

PINHO, J. A. G.; IGLESIAS, D.; SOUZA, A. C. P. *Governo eletrônico, transparência, accountability e participação: o que portais de governos estaduais no Brasil mostram*. ENANPAD, 29. Brasília, 2005.

PLONE. *Plone*. Disponível em <http://plone.org>. Acesso em: 30 de Jul. 2012.

POHL, K.; BOCKLE, G.; LINDEN, F. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, 2005

PRESSMAN, R. S. *Software Engineering: A Practitioner's Approach. 6th ed*. New York, USA: McGraw Hill, 2005.

RAINVILLE-PITT, S. C; D'AMOUR, J.-M. *Using a CMS to Create Fully Accessible Web Sites*. Journal of Access Services, v. 6, n. 1-2, p. 261-264, 2009.

SANDIM, H. C. *Pantaneiro: Framework de Aplicações Web Para Plataformas E-Gov.* Dissertação de Mestrado em Ciência da Computação. Departamento de Computação e Estatística. Universidade Federal de Mato Grosso do Sul. Campo Grande, MS, Brasil, 2009.

SEI. *A framework for software product line practice*. Disponível em <http://www.sei.cmu.edu/productlines/frame_report/index.html>. Acesso em: 4 de Jun. 2012

SHIMABUKURO, E. K.; *Um Gerador de Aplicações Configurável*. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação, ICMC-USP, São Carlos, SP, Brasil, 2006.

SOMMERVILLE, I. *Software Engineering, 9th Edition*. Addison-Wesley, 2011.

TITAN. *Titan Framework*. Disponível em <http://www.titanframework.com/>. Acesso em: 30 de Jul. 2012.

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

VINCENZO, A.; GIOVANNI, A. C. *Software reuse in local public bodies: lessons learned in tuscany*. 9th IFIP WG 8.5 International Conference on Electronic Government (EGOV'10), p. 375-386, 2010.

W3C BRASIL. *World Wide Web Consortium Escritório Brasil*. Disponível em <http://www.w3.br/>. Acesso em: 2 de Jul. 2012.

W3C. *World Wide Web Consortium*. Disponível em <http://www.w3.org/>. Acesso em: 30 de Jul. 2012.

W3C/WAI. *Introduction to Web Accessibility. 2005*. Disponível em: http://www.w3.org/WAI/intro/accessibility.php. Acesso em: Mai. 2012.

WCAG 2.0. *Web Content Accessibility Guidelines (WCAG) 2.0*, 2008. Disponível em: http://www.w3.org/TR/2008/REC-WCAG20-20081211/. Acesso em: Mai. 2012.

WEST, D.M. *E-Government and the transformation of service delivery and citizen attitudes*. Public Administration Review, v.64, n. 1, p. 15–27, 2004.

WILLIAM.B. FRAKES, W. B.; KANG, K. *Software Reuse Research Status and Future*. IEEE transactions on Software Engineering, v. 31, n. 7, Julho 2005.

WORDPRESS. *WordPress.com*. Disponível em <http://wordpress.com/>. Acesso em: 30 de Jul. 2012.

ZEND TECHNOLOGIES. *Zend Framework*. Disponível em <http://framework.zend.com/>. Acesso em: 1 de Jul. 2012.

APÊNDICE A - Regras do e-Mag Inflingidas

No quadro A.1 deste apêndice são apresentadas as regras do e-Mag não atendidas nos portais das prefeituras do estado de Mato Grosso do Sul.

Quadro A.1: Regras do e-Mag não atendidas.

Regra	Descrição
1.1	Identificar o principal idioma utilizado nos documentos.
1.11	Fornecer um equivalente textual a cada imagem.
1.12	Fornecer links de texto redundantes relativos a cada região ativa de um mapa de imagem armazenado tanto no cliente quanto no servidor.
1.16	Assegurar a acessibilidade do conteúdo de frames, fornecendo uma página alternativa através do elemento "*noframes*".
1.17	Dar a cada frame um título que facilite a identificação dos frames e sua navegação.
1.19	Assegurar a acessibilidade de objetos programados, tais como programas interpretáveis e applets, garantindo que a resposta a eventos seja independente do dispositivo de entrada e que qualquer elemento dotado de interface própria possa funcionar com qualquer leitor de tela ou navegador que o usuário utilize.
2.1	Criar documentos passíveis de validação por gramáticas formais publicadas. Declarando o tipo de documento (atributo "*doctype*") no topo do código fonte de cada página do sítio.
2.4	Não criar páginas com atualização automática periódica, até que os leitores de tela ou navegadores possibilitem o controle da atualização para o usuário.
2.5	Não utilizar marcações para redirecionar as páginas automaticamente, até que os leitores de tela ou navegadores do usuário possibilitem interromper o processo.
2.10	Utilizar elementos de cabeçalho de forma lógica, organizando o conteúdo de acordo com uma hierarquia.
2.14	Incluir caracteres pré-definidos de preenchimento nas caixas de edição e nas áreas de texto, até que os navegadores tratem corretamente os controles vazios.
2.19	Em programas interpretáveis, especificar respostas a eventos, preferindo as rotinas dependentes de dispositivos (mouse, teclado, etc).

APÊNDICE B - Resultado das buscas do Mapeamento Sistemático

Na Figura B.1 dste apêndice é apresentada a lista de todos os resultados encontrados na execução das buscas nas fontes de pesquisa.

Figura B.1: Resultado das buscas do mapeamento sistemático

id	Título	Relevante	Critério
ACM (7)			
ACM1	Using a CMS to create fully accessible websites	Não	CE5
ACM2	Learning our ABCs: accessibility, bottlenecks, and control in an organized research unit's website	Não	CE5
ACM3	Multiple solutions for digitizing equations for students	Não	CE5
ACM4	Methodology for identifying and solving accessibility related issues in web content management system environments	Sim	CI2
ACM6	Engineering web information systems: a content management system-based approach	Não	CE4
ACM7	Experiences merging two university websites	Não	CE5
IEEE Xplore (7)			
IEX1	An architecture of automating production of cross media content for multi-channel distribution	Não	CE5
IEX2	Using learning content management systems as highly adaptive and efficient supporting tool for (recurrent) training #x2014; an applied perspective	Não	CE3,CE4
IEX3	Solving Frame-Based Accessibility Problems in Web Content Management	Não	CE5
IEX4	Mobile Search - Social Network Search Using Mobile Devices Demonstration	Não	CE5
IEX5	Collecting data about television contents in the OntoTV system	Não	CE4,CE3
IEX6	Assistive smartphone for people with special needs : The Personal Social Assistant	Não	CE4
IEX7	Accessibility solutions for visually impaired users of Web discussion boards	Não	CE5
Scopus (65)			
SC1	Using MILOS to build a multimedia digital library application: The PhotoBook experience.	Não	CE5
SC2	Experiences merging two university websites	Não	CE5
SC3	Adaptive IPTV services based on a novel IP Multimedia Subsystem.	Não	CE4
SC4	Web editing module for tagging metadata of the Fedora commons repository learning objects under DRD and LOM	Não	CE4
SC5	An architecture of automating production of cross media content for multi-channel distribution	Não	CE5
SC6	LCG MCDB-a knowledgebase of Monte-Carlo simulated events	Não	CE4
SC7	A Peer-to-Peer approach to content dissemination and search in collaborative networks	Não	CE5

SC8	Bringing scientists to the people -The Co-Extra website.	Não	CE5
SC9	The use of current content management systems for accessibility.	Não	CE5
SC10	Using web content management systems for accessibility: The experience of a research institute portal	Sim	CI3
SC11	Future distribution and playback options for digital talking books.	Não	CE5
SC12	A content citizen health management system: A tele-health and tele-care prototype portal for the public.	Não	CE5
SC13	Toward the creation of a green content management	Sim	
SC14	Towards accessible content management	Não	CE5
SC15	Cyber-physical planning support systems: Advancing participatory decision making in complex urban environments.	Não	CE4
SC16	How to realize the accessibility of websites (2005)	Não	CE5
SC17	Sign language online with signlink studio 2.0.	Não	CE4
SC18	End-user development for E-government website content creation	Sim	CI2
SC19	Managing accessibility in local e-government websites through end-user development: A case study.	Sim	CI2
SC20	Using learning content management systems as highly adaptive and efficient supporting tool for (recurrent) training #x2014; an applied perspective.	Não	CE3,CE4
SC21	The integration of cartographic information into a content management system	Não	CE5
SC22	Promoting accessibility by using metadata in the framework of a semantic-web driven cms .	Não	CE5
SC23	Developing a role-based dynamic content management system for sustainable agriculture outreach through integration .	Não	CE5
SC24	Cost and benefit. Quality management for websites	Não	CE5
SC25	Solving Frame-Based Accessibility Problems in Web Content Management	Não	CE5
SC26	Website creation and resource management: Developing collaborative strategies for asynchronous interaction with library users.	Não	CE4
SC27	Evaluating the accessibility of three open-source learning content management systems: A comparative study.	Sim	CI5
SC28	New web site, new opportunities: Enforcing standards compliance within a content management system.	Não	CE5
SC29	Using hyperdocuments to manage scientific knowledge: The prototype Encyclopedia of Southern Appalachian Forest Ecosystems. (2005)	Não	CE5
SC30	Semantic analytical reports: A framework for post-processing data mining results.	Não	CE4
SC31	Automated metadata creation to enhance search capabilities in GPO's Federal Digital System	Não	CE4
SC32	Investigation of best practices for maintaining section 508 compliance in U.S. federal web sites.	Não	CE4
SC33	CMF 4ALL: A Content Management framework for ALL.	Não	CE6
SC34	eBIOMICS: A website promoting good practices and specific methods for using bioinformatics resources	Não	CE4
SC35	EDCMS: A content management system for engineering documents	Não	CE5
SC36	The EUROMED 4 project ELAICH: E-Tools for a teaching environment on EU mediterranean cultural heritage.	Não	CE4, CE3
SC37	Methodology for identifying and solving accessibility related issues in web content management system environments	Não	CE3
SC38	Influence of web content management systems in web content accessibility.	Sim	CI3
SC39	Engineering accessibility in web content management system environments.	Sim	CI2
SC40	Beyond standards: Unleashing accessibility on a learning content management system.	Sim	CI1

SC41	Multiple solutions for digitizing equations for students	Não	CE5
SC42	Open source LMS customization a moodle stadistical control aplication	Não	CE4
SC43	Design and implementation of an institutional case report form library.	Não	CE4
SC44	DICOM Wiki: Web-based collaboration and knowledge database system for radiologists.	Não	CE5
SC45	A software solution for accessible e-government portals.	Sim	CI1
SC46	Implementation concept for an accessible web CMS.	Sim	CI2
SC47	Personal learning environments: Integration of web 2.0 applications and content management	Não	CE4
SC48	Web tools and standards in the web 2.0 era for activities dissemination	Não	CE4
SC49	Website development and web standards in the ubiquitous world: Where are we going?	Não	CE4
SC50	GEO information systems - An easier way.	Não	CE4
SC51	Using a CMS to create fully accessible web sites.	Sim	
SC52	Using a CMS to create fully accessible websites.	Não	CE5, CE3
SC53	Collecting data about television contents in the OntoTV system .	Não	CE4,CE3
SC54	Web content management - The practical view Experiences with Typo 3 at the University of Applied Sciences Potsdam	Não	CE5
SC55	Web compliance management: Barrier-free websites just by simply pressing the button? Accessibility and the use of content-management-systems.	Não	CE5
SC56	Engineering web information systems: a content management system-based approach	Não	CE4
SC57	Situational requirements engineering for the development of Content Management System-based web applications.	Não	CE5
SC58	Learning our ABCs: Accessibility, bottlenecks, and control in an organized research unit's website	Não	CE5
SC59	Issues in the implementation of computer-based medical algorithms.	Não	CE4
SC60	Assistive smartphone for people with special needs : The Personal Social Assistant	Não	CE4
SC61	Athena: A hybrid management system for multi-device educational content.	Não	CE4
SC62	W4A 2012 - International Cross-Disciplinary Conference on Web Accessibility.	Não	CE4
SC63	Accessibility solutions for visually impaired users of web discussion boards	Não	CE5
SC64	PhychSpider: Six years experience with a psychology search engine for web content	Não	CE5
SC65	Towards accessibility to digital cultural materials: An FRBRized approach.	Não	CE5
SpringerLink (12)			
SL1	EDCMS: A content management system for engineering documents	Não	CE5
SL2	PCML: A Pedagogy-oriented Content Markup Language	Não	CE5
SL3	Discovering Knowledge in a Large Organization through Support Vector Machines	Não	CE5
SL4	A software solution for accessible e-government portals	Não	CE5
SL5	Using web content management systems for accessibility: The experience of a research institute portal	Não	CE5, CE3
SL6	SQCB - Sustainability Quick Check Tool for Biofuels	Não	CE4
SL7	Engineering accessibility in web content management system environments	Não	CE3
SL8	Search Engine Optimization and Accessibility	Não	CE4
SL9	Investigation of best practices for maintaining section 508 compliance in U.S. federal web sites	Não	CE3

SL10	Adaptive IPTV services based on a novel IP Multimedia Subsystem	Não	CE4, CE3
SL11	A content citizen health management system: A tele-health and tele-care prototype portal for the public	Não	CE4
SL12	The EUROMED 4 project ELAICH: E-Tools for a teaching environment on EU mediterranean cultural heritage	Não	CE4,CE3
ScienceDirect (6)			
SD1	Future distribution and playback options for digital talking books	Não	CE5
SD2	DICOM Wiki: Web-based collaboration and knowledge database system for radiologists	Não	CE5
SD3	Using hyperdocuments to manage scientific knowledge: The prototype Encyclopedia of Southern Appalachian Forest Ecosystems	Não	CE5
SD4	Un portfolio électronique structuré pour la gestion et le suivi des dossiers d'évaluation des pratiques professionnelles en technique et médecine transfusionnelles	Não	CE5, CE1
SD5	LCG MCDB-a knowledgebase of Monte-Carlo simulated events	Não	CE5, CE3
SD6	The new on-line Czech food composition database	Não	CE4

APÊNDICE C - Questionário de Avaliação

Este apêndice apresenta o questionário aplicado aos especialistas com o objetivo de avaliar a plataforma proposta neste trabalho.

Questionário de Avaliação

Part I - Perfil

1. Em qual papel você atua dentro de um processo de desenvolvimento de software?

   ( ) Desenvolvedor Iniciante
   ( ) Desenvolvedor Avançado
   ( ) Líder de equipe
   ( ) Gerente de Projeto
   ( ) Outro (Qual?)

Parte II – Entendendo a Situação Atual

2. Você utiliza algum *framework* de aplicação para o desenvolvimento de WebApps? Qual?
3. Você utiliza algum repositório para reutilização de ativos de software? Qual?
4. Você utiliza alguma ferramenta que automatiza o processo de geração do FrontEnd de WebApps? Qual?
5. Ao construir um website/portal você se preocupa com as diretrizes de acessibilidade?

   ( ) Não.
   ( ) Sim, totalmente.
   ( ) Sim, parcialmente.
   ( ) Não conheço as diretrizes de acessibilidade
6. Na sua opinião, qual a dificuldade de se construir uma WebApp acessível?

Part III – Avaliação da Plataforma

7. Você utilizaria as extensões disponibilizadas no Titan Extensions na construção de novas WebApps?

   ( ) Definitivamente usaria
   ( ) Provavelmente usaria
   ( ) Provavelmente não usaria
   ( ) Definitivamente não usaria

8. Você utilizaria o Titan FrontEnd para gerar o *Front-End* de uma WebApp?

   ( ) Definitivamente usaria
   ( ) Provavelmente usaria
   ( ) Provavelmente não usaria
   ( ) Definitivamente não usaria

9. Como você avalia o gerador Titan FrontEnd?

   ( ) Excelente
   ( ) Muito Bom
   ( ) Bom
   ( ) Razoável
   ( ) Ruim

10. Como você avalia o repositório Titan Extensions?

    ( ) Excelente
    ( ) Muito Bom
    ( ) Bom
    ( ) Razoável
    ( ) Ruim

11. Você acredita que a utilização da plataforma proposta trará benefícios com relação à acessibilidade das WebApps geradas?

    ( ) Definitivamente sim
    ( ) Provavelmente sim
    ( ) Pode ou não usar
    ( ) Provavelmente não usaria
    ( ) Definitivamente não usaria

12. Como você avalia a plataforma de LPS apresentada?

    ( ) Excelente
    ( ) Muito Bom
    ( ) Bom
    ( ) Razoável
    ( ) Ruim

Part IV – Questões Conclusivas

13. Você usaria em seu projeto a plataforma apresentada?

    ( ) Definitivamente usaria
    ( ) Provavelmente usaria
    ( ) Pode ou não usar
    ( ) Provavelmente não usaria
    ( ) Definitivamente não usaria

14. A plataforma traria alguma vantagem no seu trabalho diário?

    ( ) Aumento na produtividade
    ( ) Redução no custo e esforço
    ( ) Aumento no reúso
    ( ) Melhoria na qualidade do produto

15. A plataforma traria alguma desvantagem no seu trabalho diário?

    ( ) Diminui a produtividade
    ( ) Dispêndio maior de tempo
    ( ) Não é relevante para o reúso
    ( ) Não tem impacto sobre a qualidade
    ( ) Outras (Qual?)

APÊNDICE D - Respostas da Pesquisa

Este Apêndice apresenta as respostas dadas pelos participantes ao questionário de avaliação aplicado. As respostas podem ser visualizadas nas Figuras D.1, D.2 e D.3.

Q1	Q2	Q3	Q4	Q5	Q6
Analista de Requisitos	Sim. MVC 3 Razor.	Sim. Sharepoint.	Não.	Não	Encontrar componentes que possibilitam o desenvolvimento de funcionalidades que precisam t acessibilidade
Desenvolvedor Avançado	No momento não utilizo framework, mas já participei de projetos que utilizaram framework Zend e Titan.	Não	Não	Sim, parcialmente	Falta de motivação, às vezes o próprio líder de equipe não foca nessa funcionalidade.
Desenvolvedor Avançado	CakePHP	Não	Não	Não	Apesar de importante e oneroso, vejo que o esforço despendido para a criação de aplicações acessíveis é muito grande e muitas vezes o público alvo que consumiria esse esforço é pequ
Desenvolvedor Avançado	Cake	não	sim, cake	Sim, parcialmente	Compreender os requisitos do usuário
Desenvolvedor Avançado	Sim. Microsoft .NET Framework	Não.	Sim. Microsoft Visual Studio	Não conheço as diretrizes de acessibilidade	Não conheço.
Desenvolvedor Avançado	CakePHP e Titan	GitHub	Bake do CakePHP	Não	Falta de Automatização
Desenvolvedor Iniciante	Titan	Nao	Nao	Não	Falta de ferramentas automaticas
Desenvolvedor Iniciante	Bootstrap, Cakephp.	github, sourceforge	bootstrap, cakephp	Não	Desinteresse, baixa quantidade de usuários que necessitam de sites acessíveis
Desenvolvedor Iniciante	Titan	não	bootstrap	Não	Falta de ferramenta
Desenvolvedor Iniciante	No momento não	No momento não	Não	Sim, parcialmente	Aumento na Complexidade
Desenvolvedor Iniciante	Titan Framework, Codeigniter	Sim	Não	Sim, parcialmente	Usar as técnicas recomendadas
Desenvolvedor Iniciante	Titan	nao	nao	Não conheço as diretrizes de acessibilidade	é muito complexo e dificulta a construção e implementação
Desenvolvedor Iniciante	Titan	nao	nao	Não	...
Desenvolvedor Iniciante	CakePHP	Sourceforge	Não	Não conheço as diretrizes de acessibilidade	Muito Dificil
Desenvolvedor Iniciante	Sim. Cakephp, Titan, Ruby on Rails	Sourceforge.net	Não	Sim, parcialmente	Falta mais conhecimento de técnicas para construir WebApps mais acessíveis
Desenvolvedor Iniciante	Atualmente não, mas já utilizei o Titan.	Não	Não	Sim, parcialmente	Falta de conhecimento de todas as diretrizes de acessibilidade.
Desenvolvedor Iniciante	Titan Framework e CakePhp	Nao	Nao	Sim, parcialmente	Falta de conhecimento especifico
Desenvolvedor Iniciante	Não	Não	Não	Sim, parcialmente	O trabalho adicional necessário para adequar seu código aos padrões
Gerente de Projeto	Titan	nao	nao	Sim, totalmente	nenhuma
Líder de equipe	CakePHP, Titan Framework, Zend Framework	Não	Não	Não	Conhecimento das normas de acessibilidade e a falta de ferramentas para apoiar o desenvolv WebApp acessível
Líder de equipe	Titan Framework	Por enquanto o Titan Framework	Não	Sim, parcialmente	A dificuldade está em manter uma versão da WebApp funcional e acessível dentre algumas compatibilidades necessárias como de browsers (IE, Firefox, Chrome) e de dispositivos (PC, T Smartphone) devido a evolução constantes deste elementos.

Apêndice D.1: Respostas das questões de 1 à 6.

Q7	Q8	Q9	Q10	Q11	Q12
Provavelmente usaria	Provavelmente usaria	Excelente	Excelente	Definitivamente sim	Provavelmente usaria
Provavelmente usaria	Definitivamente usaria	Muito Bom	Muito Bom	Provavelmente sim	Definitivamente usaria
Definitivamente usaria	Provavelmente usaria	Muito Bom	Muito Bom	Provavelmente sim	Definitivamente usaria
Provavelmente usaria	Provavelmente usaria	Excelente	Excelente	Provavelmente sim	Posso ou não usar
Provavelmente usaria	Provavelmente usaria	Muito Bom	Muito Bom	Provavelmente sim	Provavelmente usaria
Definitivamente usaria	Definitivamente usaria	Excelente	Excelente	Provavelmente sim	Provavelmente usaria
Definitivamente usaria	Provavelmente usaria	Muito Bom	Muito Bom	Definitivamente sim	Definitivamente usaria
Definitivamente usaria	Provavelmente usaria	Muito Bom	Excelente	Provavelmente sim	Definitivamente usaria
Definitivamente usaria	Definitivamente usaria	Excelente	Bom	Provavelmente sim	Definitivamente usaria
Provavelmente usaria	Definitivamente usaria	Excelente	Excelente	Definitivamente sim	Provavelmente usaria
Definitivamente usaria	Definitivamente usaria	Excelente	Excelente	Definitivamente sim	Definitivamente usaria
Provavelmente usaria	Definitivamente usaria	Muito Bom	Muito Bom	Provavelmente sim	Posso ou não usar
Definitivamente usaria	Provavelmente usaria	Muito Bom	Excelente	Provavelmente sim	Provavelmente usaria
Definitivamente usaria	Definitivamente usaria	Muito Bom	Excelente	Provavelmente sim	Provavelmente usaria
Definitivamente usaria	Provavelmente usaria	Muito Bom	Excelente	Definitivamente sim	Provavelmente usaria
Provavelmente usaria	Provavelmente usaria	Muito Bom	Muito Bom	Provavelmente sim	Posso ou não usar
Definitivamente usaria	Definitivamente usaria	Excelente	Excelente	Definitivamente sim	Definitivamente usaria
Provavelmente não usaria	Provavelmente não usaria	Excelente	Bom	Provavelmente sim	Provavelmente não usaria
Definitivamente usaria	Definitivamente usaria	Excelente	Excelente	Definitivamente sim	Definitivamente usaria
Definitivamente usaria	Definitivamente usaria	Excelente	Muito Bom	Provavelmente sim	Provavelmente usaria
Definitivamente usaria	Definitivamente usaria	Bom	Excelente	Provavelmente sim	Provavelmente usaria

Apêndice D.2: Respostas das questões de 7 a 12.

Q14	Q15	Q16
Aumento na produtividade, Redução no custo e esforço, Aumento no reuso, Melhoria na qualidade do produto	Não	Excelente
Redução no custo e esforço	Inicialmente poderia gastar maior esforço de tempo para se familiarizar com a plataforma.	Muito Bom
Aumento na produtividade, Redução no custo e esforço, Aumento no reuso, Melhoria na qualidade do produto	Nenhum	Muito Bom
Redução no custo e esforço	Dispêndio maior de tempo	Excelente
Aumento no reuso	Dispêndio maior de tempo	Muito Bom
Aumento na produtividade, Aumento no reuso	Não tem impacto sobre a qualidade	Excelente
Aumento no reuso	Nao	Muito Bom
Aumento na produtividade	nenhuma	Muito Bom
Redução no custo e esforço	Não tem impacto sobre a qualidade	Muito Bom
Aumento na produtividade	nenhuma desvantagem	Excelente
Aumento na produtividade	Não tem impacto sobre a qualidade	Excelente
Aumento na produtividade	Dispêndio maior de tempo	Muito Bom
Aumento no reuso	nenhuma	Excelente
Redução no custo e esforço	Não é relevante para o reuso	Muito Bom
Aumento na produtividade	Não tem impacto sobre a qualidade	Muito Bom
Aumento na produtividade, Redução no custo e esforço, Aumento no reuso	Não tem impacto sobre a qualidade	Muito Bom
Aumento na produtividade, Redução no custo e esforço, Aumento no reuso	nenhuma	Excelente
Aumento na produtividade, Redução no custo e esforço, Melhoria na qualidade do produto	Apesar de ter gostado da plataforma, eu prefiro ter um controle maior do código produzido, por isso prefiro balancear uso de frameworks e ferramentas: quanto menos melhor.	Muito Bom
Aumento na produtividade	Não é relevante para o reuso	Excelente
Aumento na produtividade, Redução no custo e esforço, Aumento no reuso, Melhoria na qualidade do produto	Nenhuma	Muito Bom
Aumento na produtividade, Redução no custo e esforço, Aumento no reuso	O produto final (Front End) gerado pela plataforma poderia ter mais customizações principalmente com relação ao layout final. Entretanto, com a versão gerada e atualmente disponibilizada já é bem melhor do que gerar uma aplicação e-Gov desde o início, utilizando o Titan como Back End, é claro.	Muito Bom

Apêndice D.3: Respostas das questões de 14 a 16.