Linha de Produtos de Software no Processo de Geração de Sistemas Web de Apoio a Gestão de Fomento de Projetos

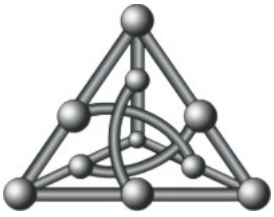
Camilo Carromeu

Dissertação de Mestrado

Orientação:Prof. Dr. Marcelo Augusto Santos Turine

Área de Concentração: Engenharia de Software

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



Departamento de Computação e Estatística
Centro de Ciências Exatas e Tecnologia
Universidade Federal de Mato Grosso do Sul
29 de novembro de 2007

Aos heróis da minha vida,
os meus pais

Agradecimentos

À minha amada família por seu apoio e presença em todos os momentos. Em especial, aos meus pais, Manoel e Elisabeth, que sempre foram o exemplo a ser seguido e aos meus irmãos Cassiano, Caio e Cauê pelas lições de todos os dias.

Aos meus familiares que me apoiaram e sempre acreditaram em mim.

À minha namorada Alice pelo apoio e estímulo durante todo o mestrado.

Ao meu orientador e amigo professor Marcelo Turine, pela atenção e confiança no decorrer e conclusão deste trabalho, sem as quais ele não seria realidade.

Aos amigos Rodrigo e Humberto, companheiros de tantos projetos.

Aos meus amigos "ogros", sempre presentes desde a minha infância e nos momentos em que mais precisei.

A todos os membros do Laboratório de Engenharia de Software (LEDES) que colaboraram com o desenvolvimento deste projeto e aos amigos que lá fiz.

Aos amigos do curso de Ciência da Computação com os quais criei fortes laços de amizade.

A todos os professores e funcionários do Departamento de Computação e Estatística (DCT) e do Núcleo de Informática (NIN) da Universidade Federal de Mato Grosso do Sul (UFMS), que me acolheram sempre que precisei.

À Fundect e ao CNPq pelo apoio financeiro para realização do mestrado.

Resumo

Atualmente, um dos desafios da administração pública é atender de maneira eficiente, rápida e satisfatória as demandas crescentes e diferenciadas advindas da sociedade. Os governos têm sido pressionados a apresentar maior eficiência na aplicação do recurso público, maior efetividade nos resultados esperados dos serviços e programas sociais, além de prestar informações e serviços à sociedade de forma transparente e democrática. Porém, tal meta é impossível de ser alcançada sem um processo de informatização das operações internas nos órgãos governamentais. A necessidade de técnicas e ferramentas para auxiliar o projeto e o desenvolvimento ágil de softwares de maior qualidade e em menor tempo é uma das preocupações da Engenharia de Software. Muitos produtos são desenvolvidos em função de artefatos já especificados e implementados, utilizando técnicas de reutilização. Neste contexto, a técnica Linha de Produtos de Software (*Software Product Line* - LPS) surge como uma proposta de construção e reutilização sistemática de software baseado em um domínio específico. O objetivo principal desta pesquisa é a instanciação e a automatização de um processo LPS orientado a família de produtos no domínio Sistemas Web de Apoio à Gestão de Fomento de Projetos (SAGF) utilizados por agências estaduais de fomento a projetos (Fundações de Amparo à Pesquisa - FAP). O processo LPS utilizado é baseado na abordagem PLUS (*Product Line UML-Based Software Engineering*) e enfatiza o uso de padrões e *frameworks* na definição de um conjunto de sistemas que compartilham similaridades e variabilidades que satisfazem as necessidades específicas do domínio. Para automatizar o processo de LPS são implementadas e integradas a ferramenta Fênix e o *framework* Titan em um ambiente de geração de aplicação que permite instanciar e gerar aplicações Web para auxiliar a submissão, gestão, monitoramento e avaliação de propostas eletrônicas de projeto a serem avaliadas pelas FAPs. Para implementação do ambiente são utilizadas tecnologias de software livre Java (JSP, Struts e Hibernate), PHP, SVG (*Scalable Vector Graphics*) e PostgreSQL. Para testar, avaliar e validar o ambiente de geração de aplicações é realizado um estudo de caso para o Sistema Web da Fundação de Apoio ao Desenvolvimento do Ensino, Ciência e Tecnologia do Estado de Mato Grosso do Sul (FUNDECT).

Palavras-chave: Reuso de Software, Linha de Produtos de Software, Gerador de Aplicação Web, *Framework* e Gestão de Fomento de Projetos.

Abstract

Currently, one of the public administration challenges is meet more efficiently, fast and satisfactory rising and differentiated demands stemming of society. Governments have been pressured to provide more efficiency in the implementation of public resource, greater effectiveness in the expected results of services and social programs, and providing information and services to society in a transparent and democratic system. However, this goal is impossible to be reached without a process of computerization of internal operations and services provided by government agencies. The need for techniques and tools to aid the design and development of software for agile higher quality and less time is one of the concerns of Software Engineering. Many of software products are developed on the basis of artifacts already defined and implemented, utilizing techniques for software reuse. In this context, the technique of Software Product Line (LPS) emerges as a proposal for software reuse construction and systematic based on a specific domain. The primary purpose of this research is the implementation of process automation to LPS oriented family of products in Support for Management Promotion Projects Web Systems domain. The process is based on PLUS LPS approach (Product Line UML-Based Software Engineering), emphasizing the use of standards and frameworks, and defines a set of systems that share a group of similar and variables characteristics that satisfy the domain needs and specific requirements. To automate the LPS process was implemented the Fênix tool and will be extended the Titan framework, which allow instanciate and generating Web applications to assist management (submission, evaluation, monitoring and completion) of electronic propose they are evaluated by agencies of promotion. The implementation will be based on free software technology Java (JSP, Struts and Hibernate), PHP, SVG (Scalable Vector Graphics) and PostgreSQL. To LPS architecture and Fênix tool test, assess and validate will be made domain case studies.

Keywords: Reuse, Software Product Line, Generative Programming, Framework and Management Promotion Projects.

Conteúdo

1 Introdução **11**

1.1 Contextualização 11

1.2 Motivações 13

1.3 Objetivos 14

1.4 Organização do Texto 15

2 Reusabilidade de Software **16**

2.1 Considerações Iniciais 16

2.2 Categorização 16

2.3 Padrões e *Frameworks* 18

2.4 Geradores de Aplicação 20

2.5 Linha de Produtos de Software 22

2.6 Abordagens de LPS 23

2.7 Considerações Finais 25

3 Aplicação do Processo LPS no Domínio de SAGF **26**

3.1 Considerações Iniciais 26

3.2 Família de Produtos de SAGF 26

3.3 Visão Geral do Processo LPS 33

3.4 Engenharia da LPS 35

3.4.1 Documento de Requisitos 36

3.4.2 Modelo de Casos de Uso 43

3.4.3 Modelo de Features 47

3.4.4 Modelo Estático e Modelo Dinâmico 52

3.4.5 Arquitetura da LPS . . . 53
3.5 Engenharia da Aplicação . . . 53
3.6 Considerações Finais . . . 54

4 Proposta do Ambiente de Geração de Aplicações 55
4.1 Considerações Iniciais . . . 55
4.2 Arquitetura . . . 55
4.3 *Framework* Titan . . . 56
4.3.1 Arquitetura . . . 57
4.3.2 Características . . . 63
4.4 Ferramenta Fênix . . . 68
4.4.1 Arquitetura . . . 69
4.4.2 Sistema de Segurança . . . 70
4.4.3 Sistema de Gestão . . . 71
4.4.4 Gerador de Código . . . 77
4.5 Tecnologias adotadas . . . 77
4.6 Considerações Finais . . . 78

5 Estudo de Caso: Fundect OnLine 79
5.1 Considerações Iniciais . . . 79
5.2 Descrição do Problema . . . 79
5.3 Geração da Aplicação . . . 80
5.4 Considerações Finais . . . 89

6 Conclusão 90
6.1 Resumo do Trabalho Realizado . . . 90
6.2 Contribuições . . . 91
6.3 Limitações da Proposta . . . 92
6.4 Trabalhos Futuros . . . 92
6.5 Trabalhos decorrentes desta pesquisa . . . 93

Referências Bibliográficas 94

Lista de Figuras

2.1 Transformações vertical, horizontal e oblíqua [34]. 21
2.2 Representação de uma LPS ([3], p.12). 22
2.3 Exemplo de modelo de *features* ([64], p.3). 23
2.4 Atividades essenciais de uma LPS ([29], p.30). 24

3.1 FAPs por estado. 27
3.2 Tela de submissão de propostas eletrônicas do SAGe. 31
3.3 Tela de visualização de eventos do JEMS-SBC. 32
3.4 Tela de preenchimento da proposta de projeto da Fundect OnLine. 33
3.5 Tela de submissão de propostas eletrônicas de ações de extensão do SIEX. 34
3.6 Ciclos do Processo ESPLEP ([56], p. 45). 35
3.7 Fases da Engenharia da LPS ([56], p. 46). 37
3.8 Diagrama de casos de uso para o ator Consultor *AdHoc*. 46
3.9 Diagrama de casos de uso para o ator Gestor. 47
3.10 Diagrama de casos de uso para o ator Pesquisador. 48
3.11 Diagrama de casos de uso para o ator Temporizador. 48
3.12 Pacote de casos de uso para a *feature* comum Locadora Kernel. 49
3.13 Pacote de casos de uso para a *feature* opcional Consultor *AdHoc*. 49
3.14 Pacotes de casos de uso para as *features* opcionais Pendências, Membros, Atividades e Recursos de Outras Fontes. 50
3.15 Pacote de casos de uso das *features* relacionadas a Orçamento. 50
3.16 Dependências entre as *features* da variabilidade de Orçamento. 51
3.17 Dependências de *features* da Linha de Produto de SAGF. 51
3.18 Modelo estático de classes de entidade. 52

3.19 Arquitetura baseada em componentes da LPS de SAGF. 53

4.1 Ambiente de geração de aplicações. 56
4.2 Tela de uma instância do Titan-FAPESP no domínio de SAGF. 58
4.3 Arquitetura do *framework* Titan. 58
4.4 Tela da seção "Notícias", na ação "Criar Notícia"de uma instância do Titan. 63
4.5 Tela dos relatórios quantitativos de uma instância do *framework* Titan. . . 65
4.6 Tela de controle de versões e auditoria de um item em uma instância do *framework* Titan. 66
4.7 Salas de chat em uma instância do *framework* Titan. 68
4.8 Arquitetura do Fênix. 70
4.9 Diagrama de casos de uso do Sistema de Segurança do Fênix. 71
4.10 Interface para gerência de aplicações. 72
4.11 Menu de seleção de gestores de configuração. 73
4.12 Diagrama de casos de uso para o Sistema de Gestão do Fênix. 73
4.13 Estrutura de um Módulo. 75

5.1 Website da FUNDECT-MS. 80
5.2 "Passo 1.1: Proposta"do formulário eletrônico de Projeto de Pesquisa da Fundect OnLine. 81
5.3 Criação de formulário eletrônico para a modalidade "Projeto de Pesquisa"e configuração do primeiro passo. 82
5.4 Diagrama entidade-relacionamento gerado para o formulário eletrônico da modalidade "Projeto de Pesquisa". 83
5.5 Passo 1.1 do formulário eletrônico para a modalidade "Projeto de Pesquisa"da instância gerada. 85
5.6 Formulário de cadastro de Pesquisador gerado pelo Titan embutido no Portal Fundect OnLine. 86
5.7 Criação do *workflow* para o trâmite de Projetos de Pesquisa. 87

Lista de Tabelas

2.1 Taxonomia de reutilização ([71], p.52). 17

3.1 Glossário. 43

3.2 Identificação dos Casos de Uso 46

Capítulo 1

Introdução

1.1 Contextualização

No campo do desenvolvimento científico e tecnológico brasileiro, um dos aspectos positivos proporcionados pela constituição de 1988 foi o estímulo à criação de agências estaduais de fomento à pesquisa denominadas FAPs (Fundação de Amparo à Pesquisa). Estas agências foram propostas aproveitando a bem sucedida experiência da FAPESP (Fundação de Amparo a Pesquisa do Estado de São Paulo - http://www.fapesp.br), criada pela lei estadual n° 5.918 de 18 de outubro de 1960 e instituída pelo decreto n° 40.132, de 23 de maio de 1962, e objetivam dotar o Estado de um organismo de apoio ao desenvolvimento científico e tecnológico autônomo, eficiente e ágil nas decisões [57]. Entre os desafios das FAPs está a necessidade de conhecer o potencial científico e tecnológico do estado, mapear demandas no setor produtivo e fomentar intercâmbio universidade-empresa nas diversas áreas de pesquisa e desenvolvimento. É neste cenário que a construção de bases de dados e de sistemas de informação pode servir de subsídio à atividade de indução de desenvolvimento sustentável no âmbito estadual.

Segundo o Conselho Nacional das Fundações Estaduais de Amparo à Pesquisa (CONFAP) do Ministério de Ciência e Tecnologia do Governo Brasileiro (http://www.mct.gov.br) até agosto de 2007 existem 23 instituições de fomento à pesquisa no Brasil. A maioria utiliza meios não digitais para a gerência de fomento de projetos, pois não há softwares genéricos devido a complexidade e a inexistência de padrões pré-definidos neste domínio. A partir do levantamento destas FAPs constatou-se a existência dos sete sistemas informatizados para apoiar a gestão de fomento de projetos: SAGe (Sistema de Apoio à Gestão de Fomento), GIPA (Gerenciamento Informatizado de Programas e Ações), SIGEP (Sistema de Gerenciamento de Projetos), AgilFAP, inFAPERJ, Patronage e FUNDECT OnLine.

Assim, faz-se necessário implantar e consolidar nas FAPs sistemas informatizados de gestão e acompanhamento de projetos a fim de auxiliar a tomada de decisões e políticas de investimento no âmbito estadual. Tais sistemas devem possuir a propriedade de assegurar um registro fiel da produção científica regional e garantir, de forma transparente, o acompanhamento de investimentos. Porém, a complexidade no desenvolvimento destes sistemas, as variabilidades existentes eles e a necessidade de constante interação com a

comunidade envolvida dificultam sua construção e implantação de forma rápida e eficaz. Esta problemática motiva a busca por soluções de inovação tecnológica, modelos, metodologias e ferramentas que agilizem tal processo. Neste contexto, a necessidade de técnicas e ferramentas para auxiliar o projeto e o desenvolvimento ágil de softwares de maior qualidade e em menor tempo é uma das preocupações da Engenharia de Software.

Muitos produtos de software são desenvolvidos em função de artefatos já especificados e implementados, utilizando técnicas de reutilização de software para melhorar a produtividade, a manutenibilidade e a qualidade tanto do software quanto do processo do desenvolvimento. O reuso caracteriza-se pela utilização de artefatos[1] de software numa situação diferente daquela para a qual foram originalmente construídos [76] [27]. As técnicas de reutilização usualmente não representam um sistema completo e sim especificações e/ou implementações de soluções parciais, tornando o processo de reuso uma atividade apenas de parametrização de variáveis.

É consenso na literatura que a partir do reuso de modelos, de projeto e de partes da implementação de software, pode-se construir novos produtos em menor tempo e com maior confiabilidade [13] [71] [55] [14] [5]. Diversas técnicas de reuso têm sido propostas na literatura, tais como, engenharia de domínio [86], *frameworks* [60] [44] [41] [75] [15] [18] [14], padrões [48] [19], linguagem de padrões [50] [51] [43] [14] [3], desenvolvimento baseado em componentes (DBC) [35] [33], geradores de aplicação [47] [7], linha de produtos de software (LPS) [29] [65] [11] [12] e aspectos [62] [92]. A abordagem LPS está em evolução na comunidade de Engenharia de Software, sendo que o primeiro workshop foi promovido pelo Instituto de Engenharia de Software (*Software Engineering Institute* - SEI) [81] em dezembro de 1996, em Pittsburgh, Pensilvânia, e foram sintetizados vários fatores envolvidos na prática de LPS, tais como, arquitetura de software, pessoal, organização, administração e modelos empresariais [6].

Tais características motivam a integração de técnicas e de ferramentas de reutilização de software no processo de desenvolvimento de Aplicações Web (WebApps) e, mais especificamente, na informatização da gestão de fomento de projeto das FAPs. Assim, no presente projeto é proposto utilizar LPS para especificar uma maneira sistemática de reuso de software, permitindo criar um conjunto de produtos de software em um mesmo domínio de negócio com características similares, por meio da definição de uma infra-estrutura comum dos artefatos que compõem os produtos, e da parametrização das diferenças entre eles. A partir de uma arquitetura genérica comum, denominada arquitetura da LPS, e de um conjunto de artefatos que povoam a arquitetura, são desenvolvidos os produtos de software. Segundo Gimenes, Lazilha e Price [54], LPS é aplicável a domínios de informação que demandam por produtos específicos com características comuns e aspectos variáveis bem definidos.

Segundo Clements e Northrop [29], organizações que trabalham com processos manufaturados utilizam constantemente uma ou mais linha de produtos na produção. Por exemplo, Boeing, Ford, Dell e McDonald's aproveitam as similaridades, ou seja, as características comuns nas linhas de produtos de diferentes maneiras. A Boeing desenvolve as

[1]Segundo França e Staa [47], um artefato é qualquer item criado como parte da definição, manutenção ou utilização de um processo de software, incluindo descrições de processo, planos, procedimentos, especificações, projeto arquitetural, projeto detalhado, código, documentação para o usuário, entre outros.

aeronaves 757 e 767 em série, e a sobreposição de partes comuns para essas duas diferentes aeronaves é de 60%, mostrando a importância da utilização de linha de produtos.

Na literatura existem poucos trabalhos sobre abordagens desenvolvidas para LPS e todas compreendem, isoladamente, técnicas de reutilização baseadas em componentes de software, *frameworks* e padrões, tais como: PLP (*Product Line Practice*) [29], Synthesis/RSP (*Reuse-Driven Software Process*) [20], PuLSE (*Product Line Software Engineering*) [9], FODA (*Feature-Oriented Domain Analysis*) [30], FOOM (*Feature Object Oriented Modeling*) [1], FAST (*Family-Oriented Abstraction, Specification, and Translation*) [65], KobrA (*Component-based Application Development*) [4] e PLUS (*Product Line UML-Based Software Egineering*) [56]. Porém, o uso de LPS na comunidade de Engenharia de Software está emergindo rapidamente como um paradigma viável para o desenvolvimento ágil de software.

Diante deste panorama e das tendências tecnológicas, várias pesquisas estão sendo realizadas objetivando propor tecnologias de reuso eficientes e eficazes para auxiliar o processo de desenvolvimento de software. Surge assim, uma oportunidade para propor e explorar métodos, técnicas, modelos de processo e ferramentas para especificar, projetar e implementar produtos de software reutilizáveis e com qualidade utilizando LPS - escopo da presente pesquisa.

1.2 Motivações

Segundo estudo da ONU[2], a evolução do governo eletrônico em um país está fortemente relacionado à sua composição social, política e econômica. Fatores como boa infra-estrutura de telecomunicações, capital humano, vontade política do governo e comprometimento de lideranças são cruciais para que o governo tenha uma visibilidade de qualidade na Internet, características normalmente encontradas em países desenvolvidos.

O mundo atual apresenta elevado grau de competição, de mudanças e de adaptações constantes. Dentro deste contexto, os projetos vêm assumindo um papel relevante nas organizações, principalmente, governamentais. Segundo Cheesman [26], um projeto pode ser definido como um empreendimento temporário com o objetivo de criar um produto ou serviço único.

A gerência de projetos e a gestão de fomento de projetos são domínios não necessariamente comuns. A gerência de projeto compreende a gestão dos diferentes caminhos que o projeto pode assumir desde sua concepção até sua execução e conclusão. A concepção de metas e objetivos do projeto, a elaboração e execução de um plano, a revisão e controle do projeto são suas características. Por fim, oferece grande variedade de princípios, procedimentos, habilidades, ferramentas e técnicas que são necessários para que se possa atingir os objetivos previamente planejados. Por outro lado, a gestão de fomento de projetos envolve decisões sobre sua viabilidade, relevância, como e onde aplicar recursos, apoio a

[2] *Benchmarking E-government: a Global Perspective*, da Divisão de Administração Pública da ONU e da *American Society for Public Administration*. Disponível em: [http://www.un.org/]. Acesso em: 02/08/2007.

planos estratégicos, entre outras características.

Assim, os desafios do domínio de gestão de fomento de projetos motivam o desenvolvimento deste projeto, cuja hipótese é que a adoção integrada de abordagens de LPS, *frameworks* e geradores de aplicação permite a instanciação de WebApps de qualidade. Para se conceber uma LPS é necessário definir conceitos que generalizam as aplicações do domínio e que permitam instanciações para aplicações específicas. Neste sentido, os *frameworks* podem ser utilizados para projetar arquiteturas de software semidefinidas que consistem de um conjunto de unidades individuais e de interconexões entre elas, de tal forma a criar uma infra-estrutura de apoio pré-fabricada para o desenvolvimento de aplicações em um ou mais domínios específicos [44]. Da mesma forma, geradores de aplicação são sistemas onde a aplicação final é gerada a partir de um padrão, definido internamente pelo gerador [69].

Assim, a investigação e a aplicação de LPS no domínio de gestão de fomento de projetos que compreenda técnicas de *frameworks* e geradores de aplicação motivam o presente trabalho. Braga [14] usou *frameworks* no domínio de Gestão de Recursos de Negócio (SGRN) que visa atender e solucionar os problemas de administração e ou de comercialização dos negócios de uma organização. Da mesma maneira, neste projeto é proposto usar LPS, *frameworks* e geradores de aplicação para instanciar Sistemas Web de Apoio à Gestão de Fomento (SAGF) que visam atender e solucionar os problemas de submissão, acompanhamento e gerenciamento de propostas de projetos nas FAPs. A iniciativa do estudo pela Família de Produtos de Software (FPS) no domínio de SAGF surgiu da necessidade de geração de sistemas on-line de gestão de propostas de projetos de pesquisa.

1.3 Objetivos

O objetivo principal deste trabalho é aplicar um processo LPS para gerar WebApps no domínio da família de produtos de SAGF, desenvolvendo e aprimorando técnicas para a sistematização de um processo de reutilização. Uma arquitetura genérica de LPS para o gerenciamento específico de sistemas de fomento deve auxiliar a criação do software gerador de sistemas neste domínio. Esta arquitetura envolve um *framework* para gerenciamento de conteúdo, denominado Titan, e o desenvolvimento de uma ferramenta que instancia este *framework* gerando WebApps neste domínio, denominada Fênix.

De forma a tornar viável a abstração das características e dos padrões necessários à especificação do *framework*, serão analisadas e comparadas as similaridades e as variabilidades de três sistemas de apoio à gestão de fomento das instituições Fundação de Apoio ao Desenvolvimento do Ensino, Ciência e Tecnologia do Estado de Mato Grosso do Sul (FUNDECT) [49], Pró-Reitoria de Extensão, Cultura e Assuntos Estudantis da Universidade Federal de Mato Grosso do Sul (PREAE/UFMS) [73] e Fundação de Amparo a Pesquisa do Estado de São Paulo (FAPESP) [40]. Para testar, validar e avaliar a aplicação do processo LPS, a ferramenta Fênix e o *framework* Titan propõe-se um estudo de caso para implementar um novo sistema para a FUNDECT, sendo possível comparar as facilidades de adaptação e manutenção com o atual sistema Fundect OnLine.

1.4 Organização do Texto

Este texto está dividido em seis capítulos. Neste Capítulo são apresentados o problema a ser investigado, o contexto, as motivações e os objetivos principais do presente trabalho. No Capítulo 2 são descritas as técnicas de reuso utilizadas e os aspectos principais e abordagens referentes a LPS. No Capítulo 3 é apresentada a aplicação do processo LPS, baseado na abordagem PLUS, no domínio SAGF bem como a definição de sua arquitetura, similaridades e variabilidades. No Capítulo 4 é apresentada a proposta do ambiente de geração de aplicações, a arquitetura do *framework* Titan e da ferramenta Fênix. O estudo de caso realizado para validar o trabalho é discutido no Capítulo 5 e, por fim, as contribuições e os trabalhos futuros são apresentados no Capítulo 6.

Capítulo 2

Reusabilidade de Software

2.1 Considerações Iniciais

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. Segundo Frakes e Terry [46], a reutilização de software pressupõe o uso de artefatos existentes do software ou o conhecimento para criação de novos softwares. É uma estratégia de solução de problemas utilizada na maioria das atividades do ser humano. Neste Capítulo são apresentados os principais conceitos utilizados neste projeto e uma visão geral sobre reutilização de software, enfatizando as diferentes abordagens de LPS.

2.2 Categorização

De Paez [71] estende a categorização das técnicas de reutilização apresentada por Frakes e Terry [46] em cinco aspectos de comparação, conforme ilustrado na Tabela 2.1: aproximação tecnológica, enfoque metodológico, modificação, alcance do desenvolvimento e alcance do domínio. A partir dos aspectos propostos determina-se o contexto de uso da técnica de reutilização. No caso específico da LPS, a taxonomia auxilia na determinação da abordagem a ser utilizada.

Aspecto	**Tipo**	**Descrição**
Aproximação Tecnológica	Por geração	Parte de uma especificação do problema que será refinada em iterações sucessivas.
	Por composição	Utiliza pequenas partes de software como invariantes para prover nova funcionalidade por meio de sua integração.
Enfoque Metodológico	Desenvolvimento para reutilização	Centrado no desenvolvimento de componentes adequados e específicos para um problema específico - visão provedor.
	Desenvolvimento com reutilização	Centrado no processo de construção de soluções de software a partir de componentes armazenados numa biblioteca - visão demanda.
Modificação	Caixa-branca	Componentes são utilizados por modificação e adaptação.
	Caixa-preta	Componentes que são reutilizados sem nenhuma modificação.
	Adaptativo	Utiliza grandes estruturas de software como invariantes e restringe a variabilidade a um conjunto de argumentos ou parâmetros.
Alcance do Desenvolvimento	Interno	Mede o nível de reutilização de componentes de um repositório do mesmo projeto.
	Externo	Mede o nível de reutilização de componentes que provêm de repositórios externos ou a proporção de produtos que foram adquiridos.
Alcance do Domínio	Vertical	Reutilização dentro do mesmo domínio de aplicação.
	Horizontal	Reutilização dentro de diferentes domínios de aplicação.

Tabela 2.1: Taxonomia de reutilização ([71], p.52).

A aproximação tecnológica refere-se à técnica utilizada para desenvolver componentes reutilizáveis que se dividem em uma aproximação por geração, ou seja, baseado em um desenvolvimento cíclico e em partes do problema; e aproximação por composição, que compreende um trabalho de integração das partes. O enfoque metodológico representa a maneira como aplicar a reutilização no processo de desenvolvimento, podendo ser "para reutilização"e "com reutilização". No primeiro caso, os componentes de reutilização são criados, e no segundo define-se o aproveitamento e o reuso de componentes ou estruturas já existentes. O aspecto modificação sugere a forma de acesso e manipulação dos componentes para adequá-los a novas necessidades, classificando-se em técnicas Caixa-Branca, as quais compreendem componentes que serão modificados para sua reutilização; técnicas Caixa-Preta, que contêm componentes que não serão modificados para sua reutilização; e técnicas Adaptativas, ou seja, uma abordagem mista das duas anteriores. O alcance de desenvolvimento indica se a reutilização somente se aplica a componentes internos

no domínio ou permite integrar componentes de outras bibliotecas, classificando-se em técnicas internas e externas. O alcance do domínio delimita a reutilização em uma família de sistemas (tipo Vertical), ou permite seu desenvolvimento entre diferentes domínios de aplicação (tipo Horizontal).

2.3 Padrões e *Frameworks*

Com o objetivo de reuso em diferentes níveis de abstracão, surgiram na década de 90, os padrões de software [19] [32] [51], que tentam captar a experiência adquirida no desenvolvimento de software e sintetizá-la em forma de problema e solução. Além de prover o reuso das soluções, os padrões ajudam a melhorar a comunicação entre desenvolvedores, que podem conduzir suas discussões com base na identificação dos padrões [51]. Para desenvolvedores que conhecem os padrões, a simples citação de seu nome traz consigo um conteúdo semântico significativo, o que dispensa a explicação dos detalhes envolvidos na solução.

Durante o processo de resolução de um problema particular, raramente os especialistas inventam uma nova solução diferente das já existentes. Diversas soluções são conhecidas de acordo com experiência própria ou de outros profissionais. Assim, ao confrontar-se com novos problemas, é comum recuperar e lembrar soluções antigas, pensando em pares "problema/solução"[19] [51]. Os padrões de software descrevem soluções de processo para problemas que ocorrem com freqüência no desenvolvimento de software, e tem por função recuperar e documentar a experiência adquirida pelos desenvolvedores durante a prática profissional. Segundo Fayad, Johnson e Schmidt [42], 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.

Os *frameworks* são definidos como aplicações semi-completas e reutilizáveis que, quando especializadas, produzem aplicações personalizadas dentro de um domínio específico [45]. Apesar de serem tipicamente voltados para o reuso, *frameworks* possuem características, como dependência de interfaces bem definidas, reuso de projeto e arquitetura, uso de padrões, que podem auxiliar no desenvolvimento de sistemas com arquiteturas mais organizadas para permitir maior facilidade de adaptação e extensão. A estrutura de um *framework* pode ser definida como um conjunto de classes que contém o projeto abstrato de soluções para uma família de problemas, propiciando o reuso com granularidade maior do que classes [45]. É composto por uma coleção de classes abstratas e concretas e de interfaces entre elas, representando o projeto de um subsistema [86].

Os *frameworks* são compostos de partes fixas e partes variáveis. As partes fixas são denominadas pontos-fixos (*frozen-spots*) e 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* [82].

De acordo com Fayad e Schmidt [44] os *frameworks* podem ser classificados, segundo seu escopo, em três grupos:

- *Frameworks* de infra-estrutura do sistema: simplificam o desenvolvimento da infra-estrutura de sistemas portáveis e eficientes como, por exemplo, sistemas operacionais, sistemas de comunicação, interfaces com o usuário e ferramentas de processamento de linguagem;
- *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 infra-estrutura de software para funcionar em um ambiente distribuído; e
- *Frameworks* de aplicação empresarial: voltados para domínios de aplicação amplos e são base para atividades de negócios das empresas, tais como, sistemas de telecomunicações, aviação, manufatura e engenharia financeira.

Considerando a forma de reuso, os *frameworks* podem ser classificados como *framework* 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. Dessa forma, o usuário precisa conhecer detalhes das estruturas internas do *framework* para utilizá-lo. 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*. Finalmente, o *framework* caixa-cinza é uma combinação do caixa-branca e do caixa-preta, ou seja, o reuso é feito por herança e pela definição de interfaces. Ele deve possuir flexibilidade e extensibilidade suficiente, e a habilidade de ocultar de seus usuários informações desnecessárias [42].

Como inicialmente é difícil conhecer todo o domínio da aplicação e prever todos os casos de uso para desenvolver um *framework* caixa-preta, as primeiras versões de um *framework* são geralmente caixa-branca. Na medida em que vai sendo utilizado o *framework* é melhorado sistematicamente até ser transformado em um *framework* caixa-cinza e, posteriormente, caixa-preta [42] [98].

De acordo com Bosch *et al.* [13], as seguintes fases são identificadas durante o desenvolvimento de um sistema baseado em frameworks:

- Desenvolvimento do *framework*: essa fase geralmente consome maior parte do tempo de desenvolvimento do software, e tem como objetivo produzir um projeto reusável dentro de um domínio que suporte a implementação do sistema desejado;
- Uso do *framework*: essa fase, também identificada por fase de instanciação ou fase de desenvolvimento de aplicações, corresponde à instanciação do *framework* para um ou mais sistemas específicos; e
- Evolução e Manutenção do *framework*: nessa fase o *framework* evolui para incorporar novas funcionalidades do domínio ou é modificado para corrigir problemas que surjam a partir do seu uso real.

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 [83]. As propostas de metodologias de desenvolvimento de *frameworks* mais relevantes para este trabalho são:

- Projetos dirigidos por exemplos [61]: a abstração do domínio, que é o próprio *framework*, é obtida por meio de casos e sistema reais, ou seja, as aplicações em uma família de produtos;
- Projeto dirigido por *Hot Spot* [74]: os *hot spots* são as partes flexíveis do *framework*, e a essência desta metodologia é identificar os *hot spots* na estrutura de classes de um domínio e, a partir disto, construir o *framework*; e
- Projeto dirigido por Linguagem de Padrões [14]: nesta abordagem o desenvolvimento de um *framework* tem como base uma linguagem de padrões em um particular domínio.

O desenvolvimento do *framework* proposto neste projeto é baseado no modelo dirigido por exemplos, permitindo abstrair os padrões a partir de sistemas no domínio da família de produtos de SAGF.

2.4 Geradores de Aplicação

Várias técnicas podem ser utilizadas para a produção automática de artefatos de software, e têm como objetivo fornecer meios para o engenheiro especificar uma aplicação em uma linguagem de alto nível e, a partir dessa especificação, uma ferramenta realizará a construção automática de artefatos, tais como código-fonte, casos de teste e documentação [82]. Os geradores de aplicação são uma das técnicas de geração de artefatos de software. Estas ferramentas têm como entrada uma especificação abstrata e, a partir dela, as atividades rotineiras de codificação, teste e documentação são feitas automaticamente. Com o uso de geradores, o desenvolvedor insere apenas as informação sobre "o que"deve ser feito e a ferramenta decide "como"tais informações são transformadas em código fonte [85] [28].

Os geradores de aplicação ajudam a construir múltiplos produtos de uma família com maior facilidade do que pela maneira tradicional. Além de código, os geradores podem produzir documentação do usuário e do software, casos de teste, digramas e figuras [28]. O processo de codificação que utiliza geradores é mais rápido e menos suscetível a erros humanos do que o processo tradicional de codificação, ou seja, os geradores podem produzir código de forma sistemática e mais segura em relação aos métodos tradicionais de programação [85] [28] [34].

De maneira geral, um gerador de aplicação tem as seguintes tarefas [34]:

- Realizar a validação da especificação de entrada e relatar erros ou avisos de inconsistências;

- Completar a especificação utilizando as configurações-padrão, caso seja necessário;
- Realizar otimizações; e
- Gerar os 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 [65].

Existem duas classificações para os tipos de transformações que um gerador pode fazer: transformações verticais e transformações horizontais. Uma transformação vertical recebe como entrada uma representação de alto nível e fornece como saída uma transformação de baixo nível, deixando a modularidade de alto nível preservada. Cada transformação vertical implementa um módulo de alto nível em um ou mais módulos de baixo nível. Esses geradores são chamados de geradores de composição [34].

As transformações horizontais redefinem a estrutura modular das representações de alto nível, ou seja, modificam a estrutura modular da representação de alto nível por meio da inserção, remoção ou integração dos módulos. As transformações que são tanto horizontais quanto verticais são denominadas transformações oblíquas [34]. Um diagrama esquemático das transformações horizontais, verticais e oblíquas é apresentado na Figura 2.1

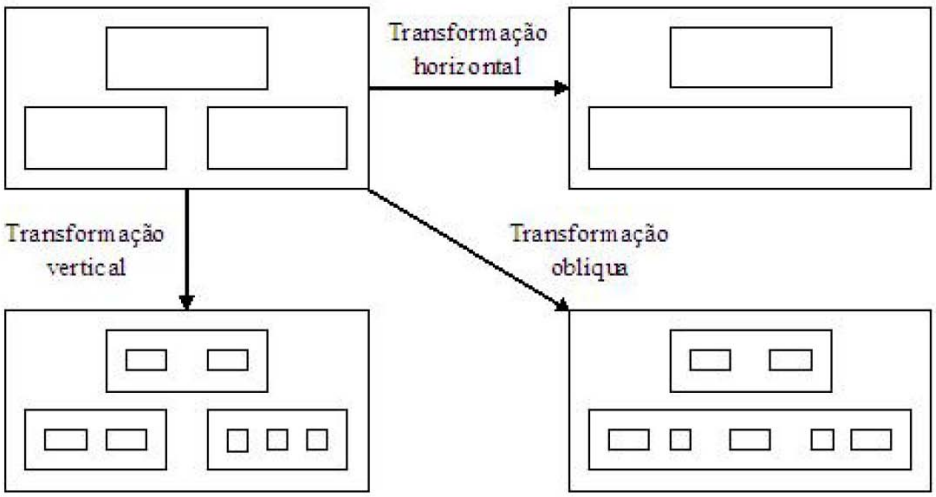


Figura 2.1: Transformações vertical, horizontal e oblíqua [34].

2.5 Linha de Produtos de Software

Segundo Chastek [25], uma LPS define uma FPS em um domínio que compartilha *features*. Segundo Gimenes e Travassos [55], uma FPS define um conjunto de produtos de software com características similares para permitir a definição de uma infra-estrutura comum de estruturação de artefatos que compõem os produtos e a parametrização de suas diferenças. Conforme ilustrado na Figura 2.2, uma LPS permite gerar um conjunto de aplicações similares (App 1, App 2, App 3, ..., App n) pertencentes a um mesmo domínio, desenvolvidas a partir de uma arquitetura genérica de LPS formada por artefatos de software (por exemplo, Modelo de Classes, Casos de Uso, Padrões e modelos próprios da LPS, como o modelo de *features*) e de um conjunto de componentes que povoam a arquitetura.

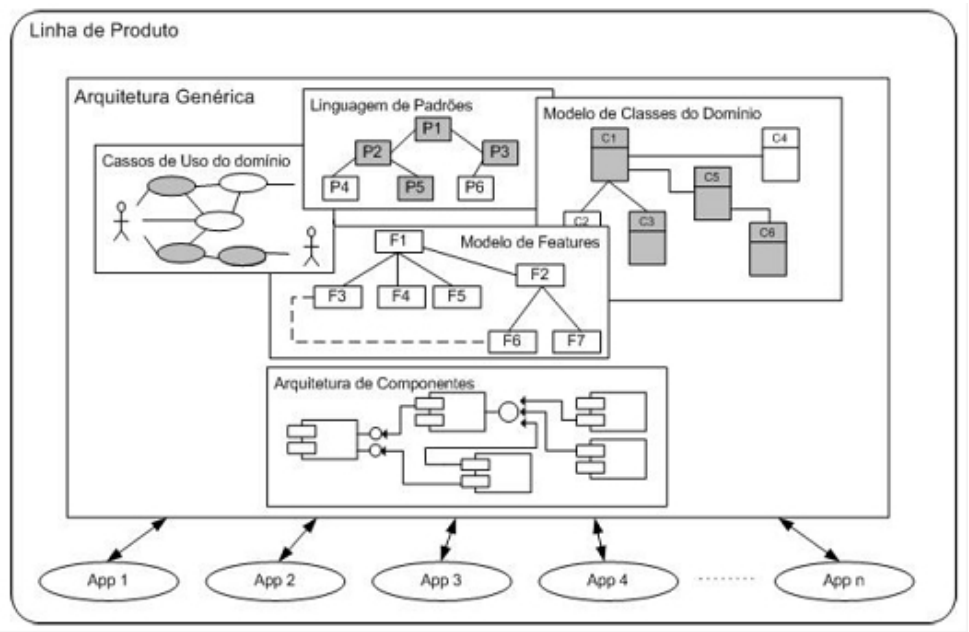


Figura 2.2: Representação de uma LPS ([3], p.12).

O modelo de *features* é utilizado na LPS para especificar e representar as variabilidades e similaridades existentes nos produtos de uma FPS e surgiu na fase de análise do domínio da abordagem FODA (*Feature-Oriented Domain Analysis*) [30]. Um exemplo de modelo de *features* para um carro é ilustrado na Figura 2.3 e compreende um conjunto de *features* representados por retângulos e relacionamentos entre eles representados por linhas em algum nível de abstração (*feature* filho e pai). Os *features* são identificados por um nome e são classificados em raiz, obrigatórios e opcionais. Existe somente um *feature* raiz que é o nó principal da estrutura, por exemplo, o *feature* "carro". Um *feature* obrigatório é representado por um círculo preenchido na parte superior do *feature*, e o opcional é representado por um círculo vazio. A notação de um triângulo vazio representa a abstração do tipo "um de"(*one of*), indicando que somente uma *feature* é utilizada, e o triângulo preenchido define o tipo "mais de um"(*more-of*), quando são utilizadas uma ou mais *features*.

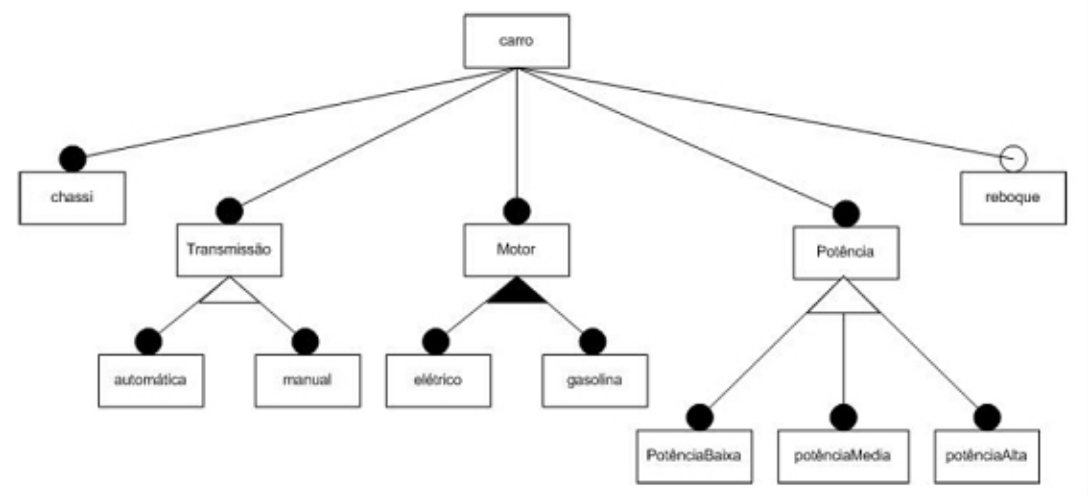


Figura 2.3: Exemplo de modelo de *features* ([64], p.3).

O desenvolvimento da LPS está determinado pela FPS que substitui o domínio de negócio no qual os recursos e os requisitos se encontram. Os padrões auxiliam na documentação e os *frameworks* definem o processo de construção da arquitetura de LPS. Segundo Clements e Northrop [29], um processo de desenvolvimento de LPS é formado por três atividades essenciais, conforme ilustrado na Figura 2.4:

- 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 de LPS para especificar os membros da FPS a partir de um conjunto de componentes reutilizáveis; e
- Gerenciamento da LPS: define regras de gestão implícitas durante todo o processo a fim de garantir o sucesso na construção e manutenção da LPS.

A seguir são apresentadas as abordagens existentes na literatura para auxiliar o processo de desenvolvimento de LPS.

2.6 Abordagens de LPS

Na literatura existem várias abordagens e estratégias para uso de LPS. Algumas são baseadas em *features* que apresentam uma solução abrangente para questões relacionadas à representação de características dentro da engenharia do domínio e da aplicação. Outras são baseadas em famílias de produtos que fornecem técnicas centradas nos aspectos de definição da família, construção da infra-estrutura arquitetural básica ou mesmo a

Figura 2.4: Atividades essenciais de uma LPS ([29], p.30).

implementação de componentes reutilizáveis. Estas estão usualmente integradas a outro método específico de engenharia de software como, por exemplo, análise de *features* e desenvolvimento baseado em componentes ao longo do processo de desenvolvimento de software.

As principais abordagens existentes na literatura são:

- PLP (*Product Line Practice*): é uma estratégia, desenvolvida pelo SEI [29], utilizada para implementação de LPS que propõe o uso de práticas de negócio e de aquisição de tecnologias para conduzir as deficiências existentes em projetos de LPS em relação a requisitos, arquitetura, modelos, descrições de processos, documentação, testes, programações, orçamentos e outros artefatos reutilizáveis;
- Synthesis/RSP: o método Synthesis/RSP (*Reuse-Driven Software Process*) [20], desenvolvido pelo *Software Productivity Consortium*, é uma aproximação sistemática para identificar similaridades e variabilidades por meio das exigências do negócio e explorar casualidades nas famílias das aplicações;
- PuLSE (*Product Line Software Engineering*): desenvolvida pelo *Institute Experimental Software Engineering* [9], define uma metodologia de construção e utilização de LPS que inclui as diferentes fases de desenvolvimento: iniciação, construção de infra-estrutura, utilização da infra-estrutura e evolução e gerenciamento;
- FODA (*Feature-Oriented Domain Analysis*): documentada pelo SEI [30], foi criada para a análise de domínios específicos e é uma técnica orientada a *features*;
- FOOM (*Feature Object Oriented Modeling*): é uma síntese do método FODA e do modelo de reengenharia Horseshoe do SEI. Proporciona um Processo Unificado de

Desenvolvimento do Software para desenvolver e descrever cada um dos modelos da arquitetura de linha de produtos. Uma parte integral desta padronização é a adoção da UML para representar os vários recursos desenvolvidos no processo [1];

- FAST (*Family-Oriented Abstraction, Specification, and Translation*): 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 [66] [65];

- KobrA (*Component-based Application Development*): utiliza uma abordagem baseada em componentes em cada fase do processo de desenvolvimento. Seu processo de implementação provê *guidelines* que especificam como os artefatos UML produzidos na fase de especificação da LPS serão usados na fase de implementação [4]; e

- PLUS (*Product Line UML-Based Software Egineering*): destaca-se por ser compatível com o RUP (*Rational Unified Process*) e com o modelo em espiral [56] [10]. 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 pesquisadas e citadas, a PLUS foi escolhida para ser aplicada neste projeto.

2.7 Considerações Finais

Neste Capítulo foi apresentada uma taxonomia de reutilização de software e diferentes técnicas existentes na literatura: *frameworks*, geradores de aplicação e LPS. Além disso, concentrou-se no processo genérico de desenvolvimento de LPS baseado em *features* que será utilizado na aplicação no domínio de SAGF. As principais abordagens de LPS foram descritas, sendo que a PLUS é a utilizada e será descrita no próximo capítulo.

Capítulo 3

Aplicação do Processo LPS no Domínio de SAGF

3.1 Considerações Iniciais

A aplicação de um processo LPS para gerar WebApps no domínio da família de produtos de Sistemas Web de Apoio à Gestão de Fomento (SAGF) é o objetivo principal deste projeto, contribuindo para o desenvolvimento de técnicas para a sistematização de um processo de reutilização. Dentre as abordagens de LPS pesquisadas, a PLUS é utilizada neste projeto por ser evolutiva, compatível com o RUP e que utiliza modelos simples e gera um repositório de componentes reusáveis.

Neste capítulo é apresentada uma revisão de todos os sistemas da família de produtos no domínio de gestão de fomento de projetos utilizados pelas agências de fomento estaduais (FAPs) no Brasil. Tais sistemas são analisados a fim de recuperar as similaridades e as variabilidades para especificar a arquitetura da LPS. Uma visão geral do processo LPS aplicado utilizando a abordagem PLUS, organizada em dois ciclos de vida (Engenharia da LPS e Engenharia da Aplicação), e a descrição dos modelos da Engenharia da LPS são apresentadas de forma contextualizada no domínio da FPS do SAGF.

3.2 Família de Produtos de SAGF

Com o objetivo de recuperar e conhecer todos os sistemas computacionais no domínio de SAGF, foram pesquisadas e descritas, resumidamente, as 24 FAPs no Brasil (existentes até agosto de 2007). O processo utilizado para tal diagnóstico foram análises do website de cada FAP e o envio de e-mails de forma informal, questionando a existência ou não de ferramentas para auxiliar a gestão de fomento de projetos. Na Figura 3.1 é ilustrada a lista das FAPs por estado e, em seguida, são apresentadas as principais características em relação as tecnologias utilizadas pelas FAPs:

1. Fundação de Tecnologia do Estado do Acre - FUNTAC

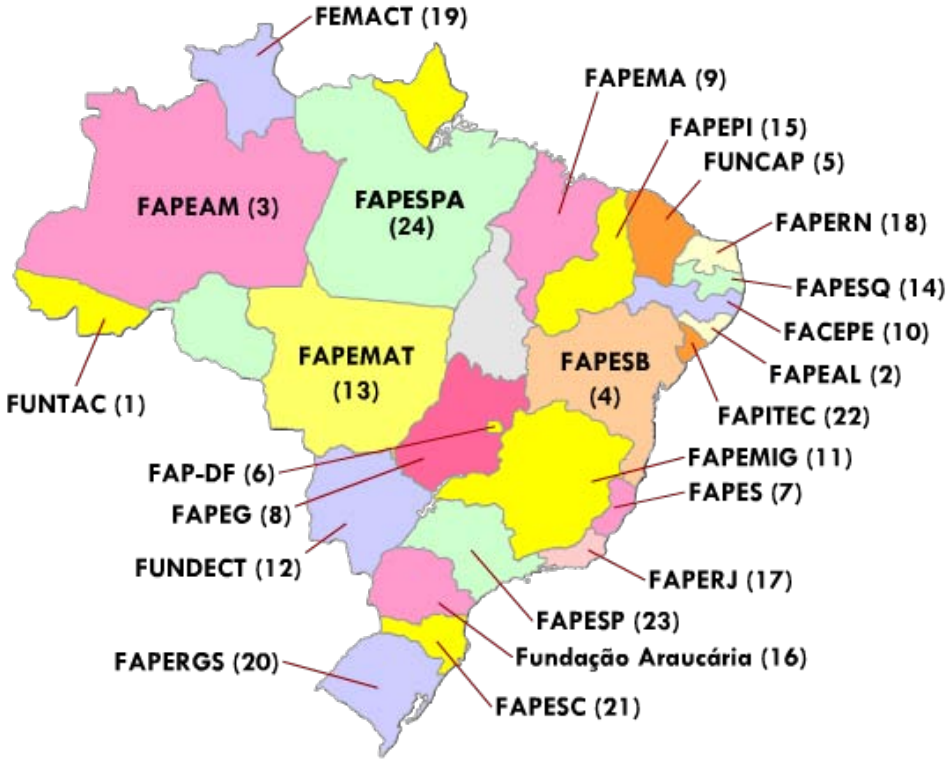


Figura 3.1: FAPs por estado.

(http://www.funtac.ac.gov.br): website criado com o *framework* Joomla![1]. O website exige cadastro para acesso, mas não oferece formulários eletrônicos para submissão de propostas de projeto.

2. Fundação de Amparo à Pesquisa do Estado de Alagoas - FAPEAL (http://www.fapeal.br): tem uma ferramenta para gerenciar pesquisadores chamada Banco de Contexto Científico e Tecnológico. O cadastro como pesquisador não permite a submissão de propostas eletrônicas, apenas auxilia a construção de mala direta dos pesquisadores.

3. Fundação de Amparo à Pesquisa do Estado do Amazonas - FAPEAM (http://www.fapeam.am.gov.br): utiliza o sistema GIPA (Gerenciamento Informatizado de Programas e Ações), que aparece como versão em desenvolvimento. Apenas o cadastro de pesquisador pode ser feito pelo GIPA, e os formulários de submissão de propostas estão disponíveis na página principal em formato DOC e SXW (Open Office).

4. Fundação de Amparo à Pesquisa do Estado da Bahia - FAPESB (http://www.fapesb.ba.gov.br): utiliza sistema on-line próprio de gerência com o mesmo nome da instituição. Seu sistema possui cadastro on-line de pesquisador, formulários eletrônicos para requisição de apoio financeiro e possibilidade de visualizar os apoios solicitados (mas sem possibilitar a edição). Não permite salvar os

[1] *Framework* em PHP encontrado em http://www.joomla.org/.

formulários para posterior alteração e submissão, além de não permitir o acompanhamento da execução dos projeto.

5. Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico - FUNCAP (http://www.funcap.ce.gov.br): disponibiliza os editais em formato PDF no website da fundação. Formulários para solicitação de apoio financeiro de bolsas em formato DOC e muitos links quebrados para o sistema de formulários on-line AgilFap, que provavelmente está em fase de implantação. A ferramenta AgilFap está sendo desenvolvida pela FACEPE e está sendo implantada em outras FAPs, listadas a seguir.

6. Fundação de Amparo à Pesquisa do Distrito Federal - FAP-DF (http://www.fap.df.gov.br): os formulários de solicitação de apoio financeiro estão disponíveis em formato DOC.

7. Fundação de Apoio a Ciência e Tecnologia do Estado do ES - FAPES (http://www.sect.es.gov.br): os editais estão disponibilizados em formato PDF no website da fundação. Formulários de requisição de apoio financeiro estão disponíveis em formato DOC.

8. Fundação de Apoio à Pesquisa do Estado de Goiás - FAPEG (http://www.fapeg.go.gov.br): o website da fundação foi desenvolvido por meio da ferramenta Simple PHP Blog[2]. Disponibiliza os editais em formato PDF e possui um sistema de gestão em desenvolvimento intitulado FAPEGestor (http://www.fapeg.go.gov.br/gestor/), mas não existe descrição técnica no portal. O sistema exige um login e senha, mas não apresenta formulário para efetuar cadastro.

9. Fundação de Amparo à Pesquisa Científica e Tecnológica do Estado do Maranhão - FAPEMA (http://www.fapema.br): possui um sistema completo de gerência de "processos"denominado Patronage, tendo um "Quadro de Avisos"na forma de Correio e modalidades de apoio organizadas. Foram encontradas várias dificuldades para ter acesso aos editais, não permite navegar entre os passos de formulários eletrônicos sem preencher os campos obrigatórios e não permite salvar o formulário durante a edição.

10. Fundação de Amparo à Pesquisa do Estado de Pernambuco - FACEPE (http://www.facepe.br): tem um website bem estruturado e é responsável por desenvolver o sistema AgilFAP.

11. Fundação de Amparo a Pesquisa do Estado de Minas Gerais - FAPEMIG (http://www.fapemig.br): utiliza o sistema AgilFap.

12. Fundação de Apoio ao Desenvolvimento do Ensino, Ciência e Tecnologia do Estado de Mato Grosso do Sul - FUNDECT (http://www.fundect.ms.gov.br): utiliza um sistema de gerência de projetos desenvolvido pelo Grupo de Engenharia de Software do Laboratório de Engenharia de Software (LEDES - http://www.ledes.net) do

[2]Ferramenta de blog em PHP encontrada em http://www.simplephpblog.com/.

DCT/UFMS. O sistema é denominado Fundect OnLine, e possui suporte completo a criação e divulgação de editais, submissão e acompanhamento de propostas eletrônicas bem como julgamento por meio de um módulo de consultores *adhoc*. Uma série de recursos adicionais como correio eletrônico, gerenciador de ofícios e equipamentos também estão disponíveis no sistema.

13. Fundação de Amparo à Pesquisa do Estado de Mato Grosso - FAPEMAT (http://www.fapemat.mt.gov.br): disponibiliza os editais em formato PDF e os manuais de prestação de contas disponíveis em formato DOC no website da fundação.

14. Fundação de Apoio à Pesquisa do Estado da Paraíba - FAPESQ (http://www.fapesq.rpp.br): possui apenas um *hotsite* de notícias dentro do website do governo da Paraíba.

15. Fundação de Amparo à Pesquisa do Estado Piauí - FAPEPI (http://www.fapepi.pi.gov.br): não possui sistema de gerenciamento on-line. Os editais e os resultados são disponibilizados em formato DOC no website da fundação.

16. Fundação Araucária de Apoio ao Desenvolvimento Científico e Tecnológico do Paraná (http://www.fundacaoaraucaria.org.br): utiliza um sistema próprio denominado SIGEP.

17. Fundação Carlos Chagas Filho de Amparo a Pesquisa do Estado do Rio de Janeiro - FAPERJ (http://www.faperj.br): utiliza um sistema próprio denominado in-FAPERJ.

18. Fundação de Apoio a Pesquisa do Estado do Rio Grande do Norte - FAPERN (http://www.fapern.rn.gov.br): não possui formulários eletrônicos e os editais e os formulários para solicitação de auxílio estão disponíveis em formato PDF e DOC no website da fundação.

19. Fundação de Meio Ambiente, Ciência e Tecnologia de Roraima - FEMACT (http://www.femact.rr.gov.br): não possui sistema de submissão de propostas on-line e os editais e resultados de seleção estão disponíveis em formato PDF e DOC no website da fundação.

20. Fundação de Amparo a Pesquisa do Estado do Rio Grande do Sul - FAPERGS (http://www.fapergs.rs.gov.br): não possui sistema de formulários eletrônicos e os editais e formulário estão disponíveis em formato PDF e DOC no website da fundação.

21. Fundação de Apoio a Pesquisa Científica e Tecnológica do Estado de Santa Catarina - FAPESC (http://www.fapesc.rct-sc.br): possui cadastro on-line de pesquisadores, mas a única finalidade é para enviar mala direta. Os editais estão disponíveis em arquivos PDF e DOC para download no website da fundação, e os resultados de editais em arquivos PDF.

22. Fundação de Apoio a Pesquisa e a Inovação Tecnológica do Estado de Sergipe - FAPITEC (http://www.fap.se.gov.br): não possui sistema de submissão de propostas eletrônicas e os editais estão disponíveis para download em formato PDF no website da fundação.

23. Fundação de Apoio a Pesquisa do Estado de São Paulo - FAPESP (http://www.fapesp.br): tem um website bem organizado e utiliza um sistema próprio denominado SAGe, desenvolvido em Java e JSP pelo grupo CESAR de Pernanbuco/PE.

24. Fundação de Amparo à Pesquisa do Estado do Pará - FAPESPA (http://www.fapespa.pa.gov.br/): os editais estão disponibilizados em HTML no website da fundação. Existe a opção de submissão de formulário eletrônico, mas sem identificação por login e senha de pesquisador e, portanto, sem histórico de submissões e acompanhamento de propostas.

A partir deste levantamento foram pesquisados os sistemas encontrados: SAGe, GIPA, SIGEP, AgilFAP, inFAPERJ e Fundect OnLine. Cada um destes sistemas foi desenvolvido dentro de diferentes realidades e necessidades de sua fundação. A existência destes diferentes sistemas dentro do domínio colaboraram para a extração das similaridades e variabilidades da família de produtos, além da identificação de problemas e soluções. Além destes sistemas foram analisadas também duas outras aplicações que não pertencem a fundações de âmbito estadual: SIEX (Sistema de Informação em Extensão Universitária da UFMS - http://siex.ledes.net) e JEMS (*Journal and Event Management System*) (https://submissoes.sbc.org.br/). A seguir são descritos, resumidamente, cada um destes sistemas.

A FAPESP implantou em 2005 o Sistema de Apoio à Gestão do Fomento (SAGe - http://www.fapesp.br/sage), cujo objetivo é informatizar os procedimentos de apresentação, análise, gestão de contratos, acompanhamento e avaliação dos programas financiados pela FAPESP [40]. Na Figura 3.2 é apresentada uma das telas de submissão de propostas eletrônicas do SAGe, que destaca os passos do formulário da proposta, opções para visualizar e submeter, além de informações sobre andamento do processo. É um dos sistemas de referência na gestão de fomento de projetos no contexto das FAPs.

O projeto GIPA (Gerenciamento Informatizado de Programas e Ações) (http://gipa.inpa.gov.br) é um exemplo de esforço em desenvolvimento para solucionar essa problemática no domínio de gerência de projetos submetidos à agência de fomento. O sistema não está finalizado, mas tem a finalidade ser utilizado em ambiente Intranet/Internet e baseado em tecnologia de software livre para o gerenciamento de dados e informações dos programas e ações da FAPEAM. Tem como objetivo permitir a automação e distribuição das propostas, alocando-as de maneira balanceada aos avaliadores, facilitando o acesso e o envio dos pareceres pelos avaliadores, ambos realizados via Web, tendo em vista a maior agilidade ao processo, e o acompanhamento técnico e financeiro das atividades de cada projeto, além de informações estatísticas sobre os projetos em execução, melhorando a comunicação e a interação entre os gestores e os coordenadores dos projetos [39].

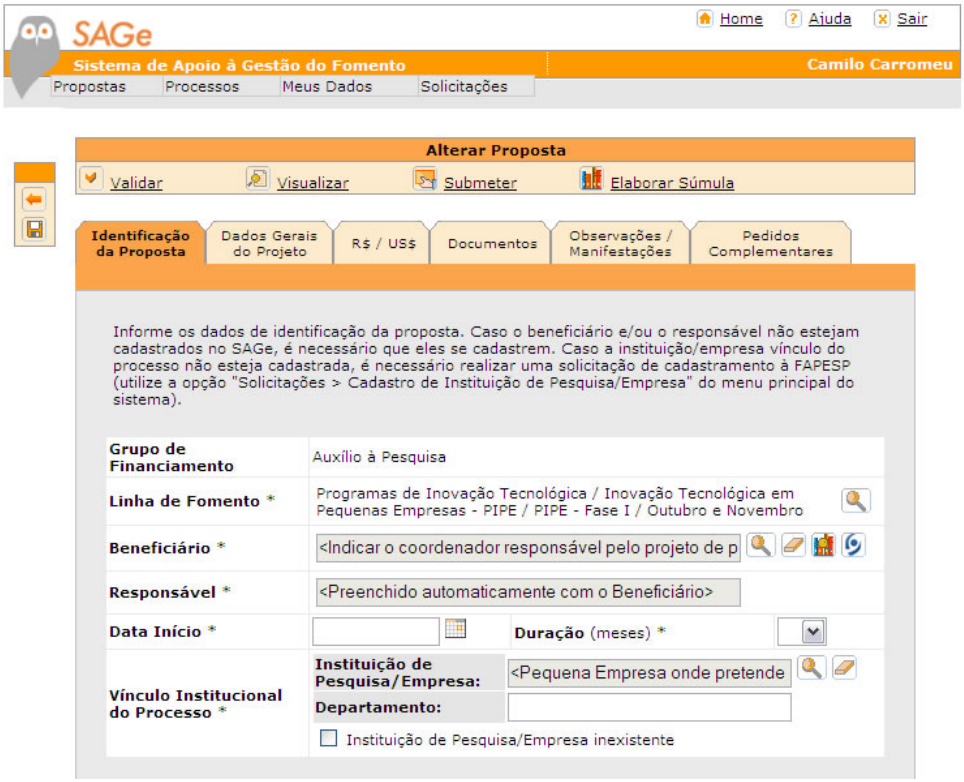


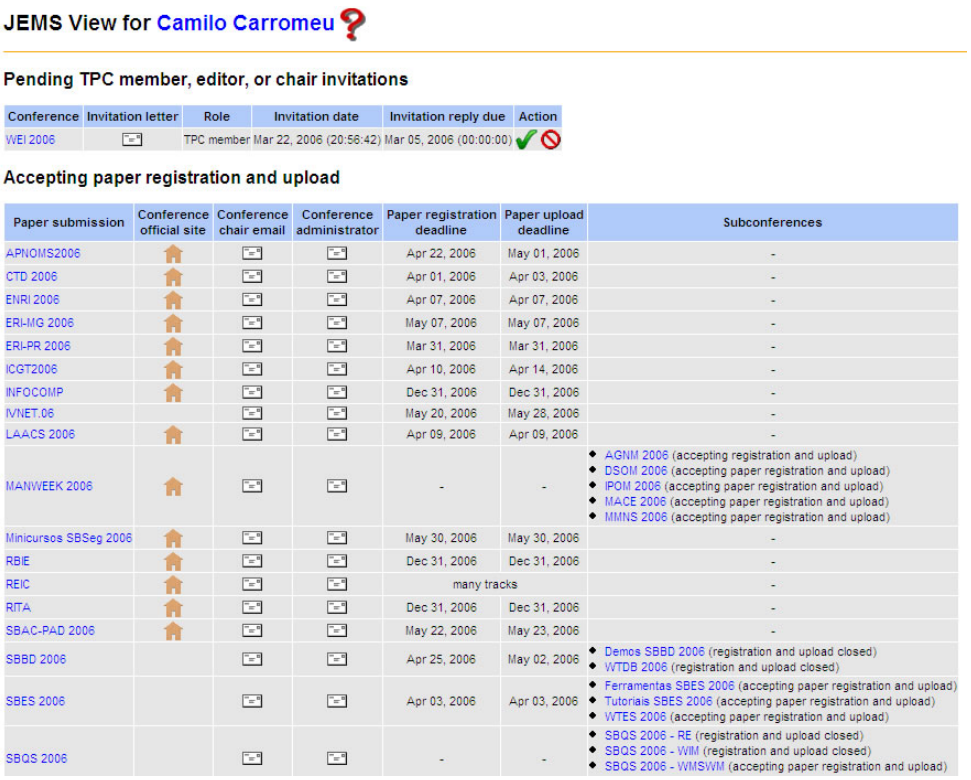
Figura 3.2: Tela de submissão de propostas eletrônicas do SAGe.

O SIGEP (Sistema de Gerenciamento de Projetos) (http://www.sigep-araucaria.pr.gov.br) é uma aplicação informatizada da Fundação Araucária para a gestão estratégica dos recursos destinados à pesquisa científica no Estado do Paraná. Permite a edição de propostas eletrônicas com formulários baseados em passos, onde cada proposta é vinculada a um edital que deve estar vigente. Dentre os problemas encontrados neste sistema estão o fato de apresentar alguns links quebrados e o formulário de submissão de propostas apresentar poucas opções, não sendo possível verificar pendências durante o preenchimento do mesmo.

O sistema inFAPERJ (https://infaperj.faperj.br) foi idealizado para agilizar o trabalho administrativo da FAPERJ e diminuir a utilização de documentos em papel durante a solicitação de auxílios, bolsas e editais. O sistema permite o cadastro eletrônico dos pesquisadores e este pode utilizar o sistema para participar de editais bem como acompanhar o andamento dos processos. Este sistema apresenta uma interface com alta usabilidade, intuitiva e possui um sistema de correio eletrônico de fácil utilização. É um dos sistemas de referência na gestão de fomento de projetos no contexto das FAPs.

O AgilFAP é um software de gestão de informações e logística desenvolvido especificamente para as Fundações de Amparo à Pesquisa. É uma ferramenta desenvolvida pela FACEPE e está sendo implantada em outras instituições, tais como, FAPEMAT, FUNCAP, FAPERGS e FAPEMIG. Este sistema apresenta alguns problemas como o fato de não permitir que formulários em edição possam ser salvos e submetidos em outro momento.

O JEMS (https://submissoes.sbc.org.br) é um sistema baseado na Web para controle de trabalhos em eventos (congressos ou workshops). Permite submeter artigos e trabalhos acadêmicos a serem julgados por avaliadores cadastrados pelo Comitê Organizador do respectivo evento. O JEMS possibilita um controle integral e acompanhamento de toda a organização do evento por parte da comissão responsável bem como a interação da comunidade acadêmica com os organizadores. Atualmente o JEMS vem sendo utilizado pela Sociedade Brasileira de Computação (SBC) para controle de todos os eventos relacionados à Computação e Informática no país [80]. A tela onde o pesquisador pode observar os congressos e os eventos acadêmicos promovidos pela SBC é ilustrada na Figura 3.3.



JEMS View for Camilo Carromeu

Pending TPC member, editor, or chair invitations

Conference	Invitation letter	Role	Invitation date	Invitation reply due	Action
WEI 2006		TPC member	Mar 22, 2006 (20:56:42)	Mar 05, 2006 (00:00:00)	

Accepting paper registration and upload

Paper submission	Conference official site	Conference chair email	Conference administrator	Paper registration deadline	Paper upload deadline	Subconferences
APNOMS2006				Apr 22, 2006	May 01, 2006	-
CTD 2006				Apr 01, 2006	Apr 03, 2006	-
ENRI 2006				Apr 07, 2006	Apr 07, 2006	-
ERI-MG 2006				May 07, 2006	May 07, 2006	-
ERI-PR 2006				Mar 31, 2006	Mar 31, 2006	-
ICGT2006				Apr 10, 2006	Apr 14, 2006	-
INFOCOMP				Dec 31, 2006	Dec 31, 2006	-
IVNET.06				May 20, 2006	May 28, 2006	-
LAACS 2006				Apr 09, 2006	Apr 09, 2006	-
MANWEEK 2006				-	-	AGNM 2006 (accepting registration and upload) DSOM 2006 (accepting paper registration and upload) IPOM 2006 (accepting paper registration and upload) MACE 2006 (accepting paper registration and upload) MMNS 2006 (accepting paper registration and upload)
Minicursos SBSeg 2006				May 30, 2006	May 30, 2006	-
RBIE				Dec 31, 2006	Dec 31, 2006	-
REIC				many tracks		-
RITA				Dec 31, 2006	Dec 31, 2006	-
SBAC-PAD 2006				May 22, 2006	May 23, 2006	-
SBBD 2006				Apr 25, 2006	May 02, 2006	Demos SBBD 2006 (registration and upload closed) WTDB 2006 (registration and upload closed)
SBES 2006				Apr 03, 2006	Apr 03, 2006	Ferramentas SBES 2006 (accepting paper registration and upload) Tutoriais SBES 2006 (accepting paper registration and upload) WTES 2006 (accepting paper registration and upload)
SBQS 2006				-	-	SBQS 2006 - RE (registration and upload closed) SBQS 2006 - WIM (registration and upload closed) SBQS 2006 - WMSWM (accepting paper registration and upload)

Figura 3.3: Tela de visualização de eventos do JEMS-SBC.

A FUNDECT iniciou em 2003 o desenvolvimento do sistema Fundect Online em parceria com o grupo de pesquisa do LEDES do DCT (Departamento de Computação e Estatística) da UFMS [49]. Esse sistema foi implementado em linguagem PHP e tem como objetivo automatizar todo o processo de submissão, julgamento e acompanhamento de propostas de projetos para a FUNDECT. Por meio deste sistema, um pesquisador do estado de MS pode se cadastrar, elaborar e submeter propostas de projetos de pesquisa ou de eventos regionais. O sistema oferece suporte para que as propostas sejam julgadas por meio de consultores *adhoc*, pré-cadastrados no sistema, e que, caso aprovadas, possam ser financiadas pela fundação. Permite o acompanhamento durante a execução do projeto, disponibilizando de forma transparente às partes envolvidas todo o acompanhamento de gastos realizados, relatórios e prestação de contas. Na Figura 3.4 é ilustrada uma das telas do formulário de submissão de propostas eletrônicas da Fundect OnLine, que contém os três passos principais (1.Principal, 2.Equipe e 3.Orçamento) para o preenchimento do formulário do projeto.

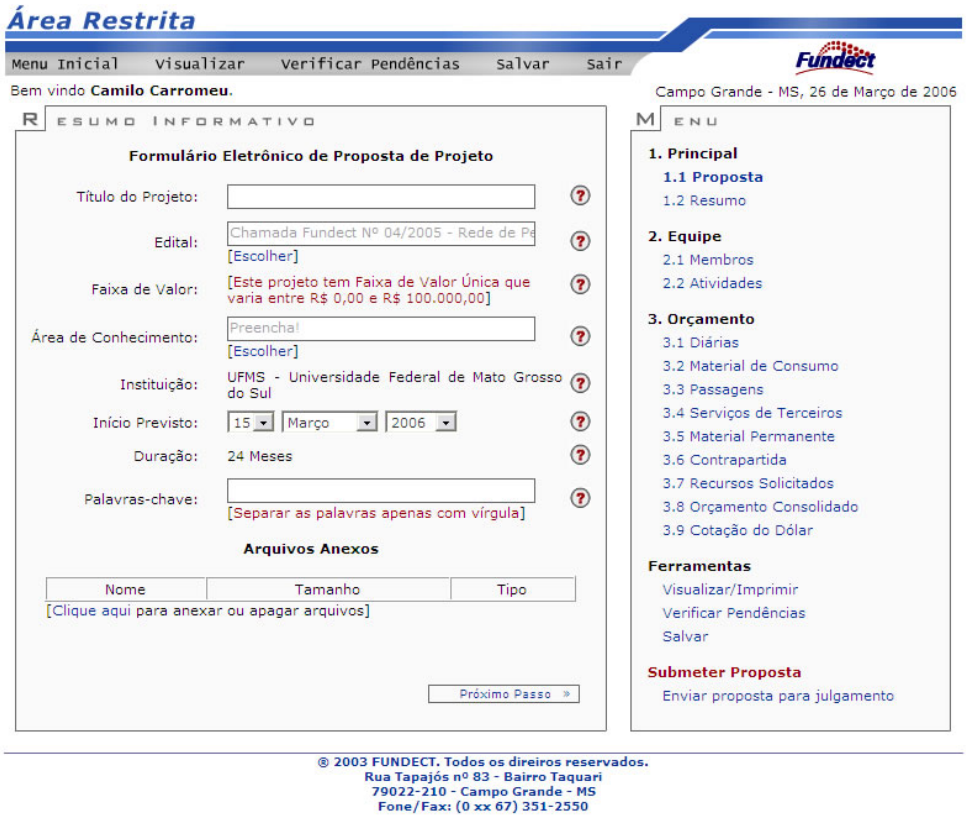


Figura 3.4: Tela de preenchimento da proposta de projeto da Fundect OnLine.

Além do Fundect OnLine, um outro sistema desenvolvido pelo LEDES-DCT/UFMS no domínio de gestão de fomento de projetos foi o SIEX da Pró-Reitoria de Extensão, Cultura e Assuntos Estudantis (PREAE/UFMS) e [73] do Fórum de Pró-Reitores de Extensão das Universidades Públicas Brasileiras - FORPROEX (http://www.renex.org.br). O SIEX está sendo utilizado por várias instituições de ensino superior e foi definido com a SESu/MEC, em novembro de 2007, como sistema brasileiro para gerenciar a Extensão Universitária. Tem como objetivo principal agilizar o processo de registro e envio das ações de extensão por meio da Internet, além de gerenciar os pareceres técnicos da Comissão Técnica, acompanhando as ações durante as fases de planejamento, execução e avaliação. Assim, como o sistema da Fundect OnLine, o SIEX vem sendo utilizado como estudo de caso e análise de requisitos para desenvolvimento do *framework* Titan e da ferramenta Fênix, propostas desta pesquisa. Na Figura 3.5 é ilustrada a tela de submissão de propostas eletrônicas de ações de extensão do SIEX.

3.3 Visão Geral do Processo LPS

O processo de LPS utilizado é baseado na abordagem PLUS [56] e tem como objetivo permitir o desenvolvimento ágil de sistemas na família de produtos SAGF de forma iterativa, compatível com o USDP (*Unified Software Development Process*), o RUP (*Rational Unified Process*) e com o modelo de processo em espiral de Boehm [10]. Além disso, usa

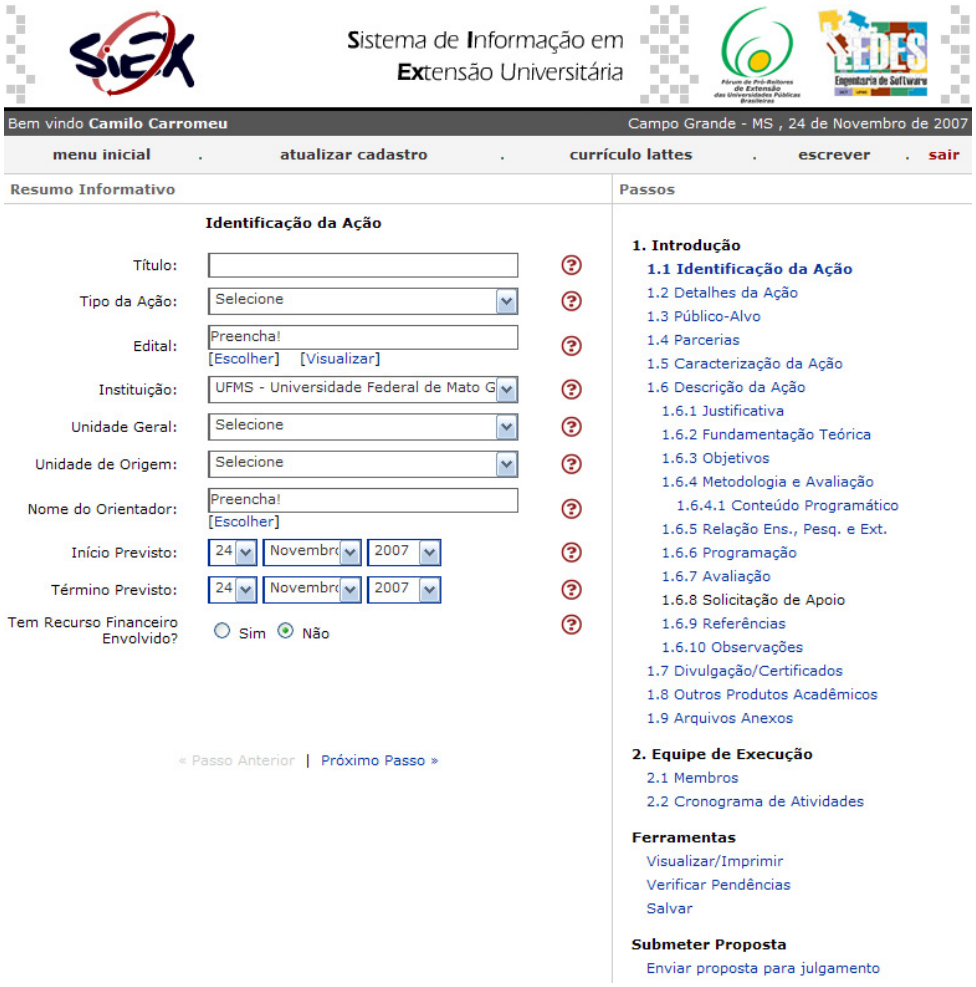


Figura 3.5: Tela de submissão de propostas eletrônicas de ações de extensão do SIEX.

o modelo de *features* e gera um projeto baseado em componentes.

Na Figura 3.6 são ilustrados os dois ciclos de vida do processo ESPLEP (*Evolutionary Software Product Line Engineering Processo*) que o PLUS utiliza: Engenharia da Linha de Produtos de Software e Engenharia da Aplicação. Os dois ciclos são desenvolvidos de forma iterativa e incremental, permitindo o desenvolvimento de artefatos reutilizáveis armazenados no Repositório de Linha de Produtos de Software, além de sistemas executáveis para famílias de produtos de software de acordo com as necessidades do cliente. 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.

No ciclo da Engenharia da LPS o engenheiro da LPS define as similaridades e as variabilidades a partir da análise de requisitos da FPS do domínio de negócio. Os artefatos da LPS desenvolvidos durante este ciclo são modelo de caso de uso, modelos 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, um membro da FPS utilizado 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 dando origem a arquitetura da aplicação. Assim, tendo a arquitetura da aplicação e os componentes reutilizáveis do repositório, a aplicação é construída.

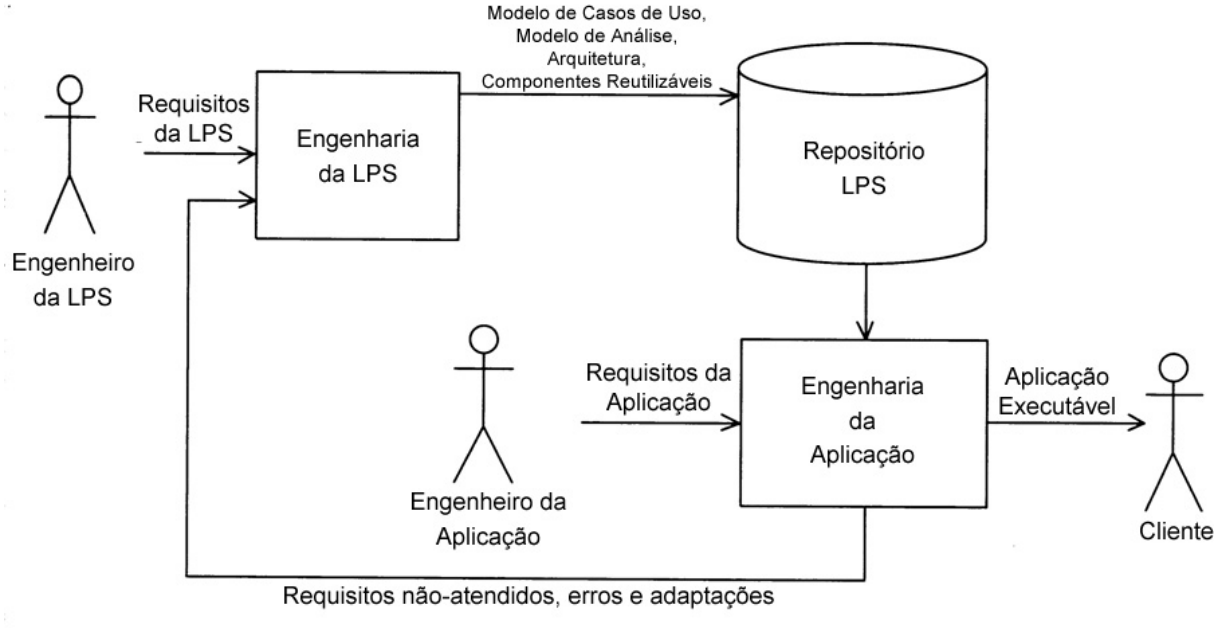


Figura 3.6: Ciclos do Processo ESPLEP ([56], p. 45).

Segundo Gomma [56], as duas estratégias para o desenvolvimento de uma LPS são: Engenharia Evolutiva Avante e Engenharia Evolutiva Reversa. Em ambos os casos o desenvolvimento é incremental e evolucionário. Quando não existir sistemas no domínio da FPS para guiar o desenvolvimento da LPS é sugerido utilizar a Engenharia Evolutiva Avante, cuja ênfase inicial é desenvolver o núcleo da LPS (*kernel*) composto pelos casos de uso das similaridades dos membros da LPS. A Engenharia Evolutiva Reversa é mais aplicável quando já existem sistemas para análise e modelagem. Os modelos de casos de uso de cada sistema individual da FPS são analisados e documentados inicialmente. Estes modelos são integrados em um modelo de caso de uso da LPS. Os casos de uso que são usados apenas em um sistema individual ou um subconjunto dos membros da FPS são classificados como casos de uso opcionais ou alternativos.

3.4 Engenharia da LPS

O processo da Engenharia da LPS é composto por cinco fases, ilustradas na Figura 3.7, sendo que em cada fase modelos são produzidos e armazenados no Repositório da LPS. A seguir são descritas cada uma destas fases e os modelos gerados:

1. Modelagem de Requisitos: Durante esta fase um modelo de requisitos é elaborado e

consiste em um documento de requisitos, um modelo de casos de uso e um modelo de *features*. O modelo de casos de uso define requisitos funcionais da LPS em termos de atores e casos de uso e, posteriormente, é estendido para um modelo de similaridades e variabilidades por meio do desenvolvimento de casos de uso do núcleo, opcionais e alternativos. Cada *feature* do modelo de *features* é um requisito que define uma similaridade ou uma variabilidade que pode ser parametrizada;

2. Modelagem de Análise: Nesta fase são desenvolvidos os modelos estático e dinâmico. O modelo estático define as relações estruturais entre as classes do domínio, e o modelo dinâmico é desenvolvido a partir dos casos de uso e mostra os objetos que participam de cada caso de uso e como eles interagem;

3. Modelagem Arquitetural: Nesta fase é especificada a arquitetura da LPS baseada em componentes. O modelo de análise é mapeado para o modelo arquitetural considerando-se os padrões de software;

4. Implementação Incremental de Componentes: Nesta fase um subconjunto da LPS é selecionado para ser implementado em cada iteração. A implementação inicia-se com os casos de uso do núcleo e a implementação dos casos de uso opcionais e alternativos, de acordo com a seqüência estabelecida no modelo dinâmico. A implementação de cada caso de uso consiste no detalhamento da arquitetura, codificação e teste de unidade dos componentes; e

5. Teste: Nesta fase são realizados o teste de integração e de funcionalidade de componentes da LPS.

Nesta seção será apresentada uma versão resumida de cada um dos modelos do processo para contextualizar a Engenharia da LPS no domínio de SAGF.

3.4.1 Documento de Requisitos

A partir de sistemas existentes, é elaborado um documento de requisitos inicial da LPS que contém a definição dos requisitos funcionais e não funcionais da LPS, destacando as variabilidades dos sistemas. Além disso, são descritos os principais termos do domínio da FPS no glossário.

A - REQUISITOS FUNCIONAIS

A1 - Armazenamento

1. O sistema deve permitir a inclusão, alteração e remoção de **gestores** da instituição com os seguintes itens de informação: nome, login (identificação), cpf, e-mail, telefone, celular, cargo, cidade, estado e dados do endereço.

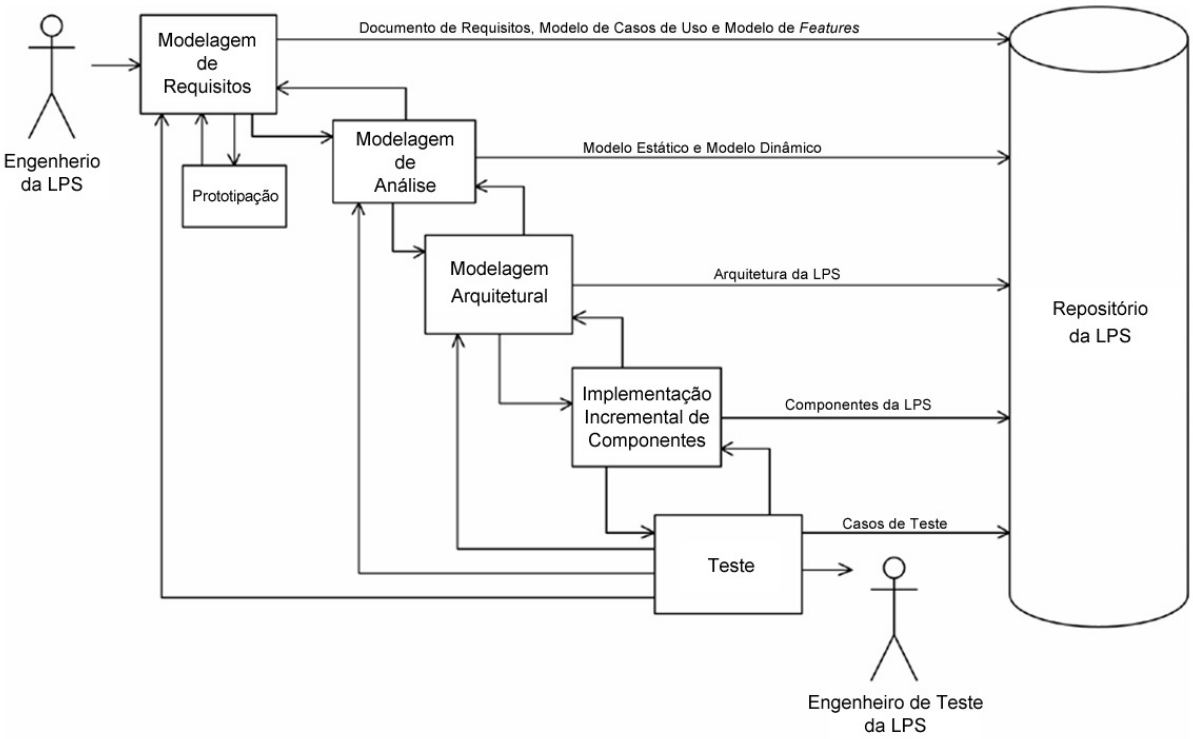


Figura 3.7: Fases da Engenharia da LPS ([56], p. 46).

2. O sistema deve permitir a inclusão, alteração e remoção de **pesquisadores** da instituição com os seguintes itens de informação: nome, login (identificação), cpf, rg, sexo, data de nascimento, e-mail, nível acadêmico, telefone, celular, instituição, dados do endereço residencial e institucional.

3. O sistema deve permitir a inclusão, alteração e remoção de **consultores *adhoc*** da instituição com os seguintes itens de informação: nome, login (identificação), cpf, rg, sexo, data de nascimento, e-mail, nível acadêmico, telefone, celular, instituição, dados do endereço residencial e institucional. Ver variabilidade *V1 - Consultores AdHoc*.

4. O sistema deve permitir a inclusão, alteração e remoção de **instituições** com os seguintes itens de informação: nome, sigla, cnpj, e-mail, telefone, dados do endereço e representante legal.

5. O sistema deve permitir a inclusão, alteração e remoção de **editais** de financiamento de projetos. Cada edital possui os seguintes itens de informação: título, identificação, recursos, datas de vigência, data de divulgação dos resultados, data de liberação de recursos e duração dos projetos vinculados. Cada edital possui um atributo "tipo"que especifica para qual modalidade de projeto é o edital.

6. O sistema deve gerenciar os editais segundo quatro classificações: inativo (quando a vigência do edital ainda não foi alcançada pela data atual); aberto (quando o atributo "vigência do edital"engloba a data atual); encerrado (quando a data atual

se encontra entre a última data de vigência e a data de divulgação de resultados); e finalizado (quando a data atual é maior à data de divulgação de resultados).

7. O sistema deve permitir a inclusão, alteração e remoção de **modalidades de fomento** das instituições de fomento. Tipicamente, podem ser: projetos de pesquisa, de extensão, eventos, bolsas (Mestrado, Doutorado, Pesquisador convidado) e publicações. Cada modalidade possui os seguintes itens de informação: título, edital, área de conhecimento, instituição, início previsto, duração, palavras-chave, orientador, duração da bolsa e documentos de anexo.

8. O sistema deve permitir a inclusão, alteração e remoção de **propostas de projetos** em cada modalidade, vinculadas necessariamente a um "edital". Sempre que uma proposta é criada o sistema vincula automaticamente o pesquisador autor da proposta como seu "coordenador".

9. O sistema deve permitir a inclusão, alteração e remoção de **tipos de publicação**, que podem ser: boletim técnico, folder, livro e manual técnico.

10. O sistema deve permitir a inclusão, alteração e remoção de **membros** vinculados a uma proposta de projeto. Cada membro está vinculado a um pesquisador pré-cadastrado no sistema e terá um atributo "função" que especifica a função do membro no projeto. O membro de uma proposta terá a possibilidade de visualizar esta proposta em qualquer momento, mas não poderá efetuar alterações. Ver variabilidade *V3 - Membros*.

11. O sistema deve permitir a inclusão, alteração e remoção de **atividades** do cronograma físico vinculadas a uma proposta de projeto. Cada atividade possui os seguintes itens de informação: nome, duração, mês de início, carga horária semanal e responsável. O responsável é um dos membros ou o coordenador da proposta. Além destas informações, o sistema deve possibilitar o vínculo de membros à atividade. Ver variabilidade *V4 - Atividades*.

12. O sistema deve permitir a inclusão, alteração e remoção de **diárias** vinculadas a uma proposta de projeto. Cada diária possui os seguintes itens de informação: localidade, número de diárias e custo unitário. Ver variabilidade *V5 - Orçamento*.

13. O sistema deve permitir a inclusão, alteração e remoção de **materiais de consumo** vinculados a uma proposta de projeto. Cada material de consumo possui os seguintes itens de informação: descrição, quantidade e custo unitário. Ver variabilidade *V5 - Orçamento*.

14. O sistema deve permitir a inclusão, alteração e remoção de **passagens** vinculadas a uma proposta de projeto. Cada passagem possui os seguintes itens de informação: trecho, quantidade, tipo (aérea ou terrestre) e custo unitário. Ver variabilidade *V5 - Orçamento*.

15. O sistema deve permitir a inclusão, alteração e remoção de **serviços de terceiros** vinculados a uma proposta de projeto. Cada serviço de terceiros possui os seguintes itens de informação: descrição e custo total. Ver variabilidade *V5 - Orçamento*.

16. O sistema deve permitir a inclusão, alteração e remoção de **materiais permanentes** vinculados a uma proposta de projeto. Cada material permanente possui os seguintes itens de informação: descrição, tipo (nacional ou importado), quantidade e custo unitário. Ver variabilidade *V5 - Orçamento*.

17. O sistema deve permitir a inclusão, alteração e remoção de **recursos de outras fontes** vinculados a uma proposta de projeto. Cada recurso possui os seguintes itens de informação: entidade, valor e moeda. Ver variabilidade *V6 - Recursos de Outras Fontes*.

18. O sistema deve permitir a inclusão, alteração e remoção de **relatórios técnicos** financeiros ou não das propostas de projetos por usuários gestores a fim de permitir acompanhamento e encerramento da proposta.

19. O sistema deve permitir a inclusão, alteração e remoção de **modelos de parecer técnico** por usuários gestores. Cada modelo é uma coleção de itens de informação, cada qual com um peso, uma descrição, um rótulo e nota. Quando um consultor *adhoc* for julgar uma proposta de projeto deve preencher um **formulário de parecer** que o sistema gera dinamicamente por meio do modelo vinculado ao edital da proposta. Ver variabilidade *V1 - Consultores AdHoc*.

A2 - Movimentações

20. O sistema deve permitir a submissão de propostas de projetos para julgamento. Quando uma submissão for efetuada o sistema deve verificar as pendências da proposta e se o edital escolhido encontra-se "aberto". O processamento da **submissão** possui os seguintes itens de informação: data e hora da submissão e número de protocolo. O número de protocolo é gerado dinamicamente e deve ser um identificador único no sistema. Ver variabilidade *V2 - Pendências*.

21. O sistema deve possibilitar a atribuição de diversas **situações** às propostas de projetos. As situações podem ser: rascunho; enquadrada; não-enquadrada; sob avaliação de consultores *adhoc*; sob avaliação final; não aprovada; aprovada e não contratada; aprovada, contratada e em andamento; aprovada, contratada e concluída; e, aprovada, contratada e cancelada. Quando a proposta é criada o sistema atribui sua situação como "rascunho". Quando ela for enviada para julgamento o sistema atribui automaticamente sua situação como "enquadrada".

22. O sistema deve possibilitar a edição e exclusão da proposta pelo pesquisador sempre que sua situação for "rascunho". Em qualquer outra situação o sistema deve possibilitar a edição da proposta pelos gestores e apenas a visualização pelo pesquisador. O sistema deve possibilitar a visualização das propostas de projetos pelos consultores *adhoc* sempre que sua situação for "sob avaliação de consultores *adhoc*".

A3 - Consultas

23. O sistema deve permitir que seja gerada uma versão para impressão da proposta de projeto em qualquer momento.

24. O sistema deve permitir a visualização de relatórios de **orçamento consolidado** e **recursos solicitados** da proposta de projeto a qualquer momento.

25. O sistema deve permitir a consulta de **verificação de pendências** por parte do pesquisador, em qualquer momento antes da submissão da proposta.

26. O sistema deve permitir aos gestores consultar todas as propostas de projeto submetidas para julgamento, ou seja, todas cuja situação forem diferentes de "rascunho". O sistema deve, para cada proposta, tornar disponível links de acesso a cada uma das cópias de backup geradas durante mudança de situação.

27. O sistema deve permitir aos pesquisadores consultar todas as propostas de projetos nas quais o usuário é coordenador ou membro.

28. O sistema deve permitir a geração de relatórios quantitativos das propostas de projeto. Para cada edital o relatório deve gerar uma lista de projetos, pesquisadores e instituições envolvidas, bem como o total de cada argumento.

29. O sistema deve permitir a geração de relatórios qualitativos, levando em consideração filtros e cruzamento de informações de todos os projetos cujo a situação seja diferente de "rascunho".

B - REQUISITOS NÃO-FUNCIONAIS

B1 - Confiabilidade

30. O sistema deve salvar, automaticamente, a proposta de projeto a cada mudança de passo do formulário de edição.

31. O sistema deve permitir que a proposta seja salva a qualquer momento durante uma edição.

32. O sistema deve gerar automaticamente uma cópia de backup da proposta de projetos toda vez que ela mudar de situação. Assim, haverão cópias da mesma proposta em todas as situações.

B2 - Segurança

33. O sistema deve possuir senha de acesso e identificação para cada usuário. As áreas de logon devem ser diferenciadas para cada tipo de usuário: pesquisador, consultor *adhoc* e gestor.

B3 - Eficiência

34. O sistema deve responder a consultas em menos de oito segundos.

35. O sistema deve apresentar relatórios solicitados dentro de no máximo trinta segundos após sua requisição.

B4 - Portabilidade

36. O sistema deve ser executado em um servidor Pentium IV 1GHz ou superior, com sistema operacional Linux com kernel 2.4 ou superior, com *webserver* Apache 2 ou superior e com banco de dados PostgreSQL 8.1 ou superior.

37. O sistema deve ser apropriado as máquinas clientes do tipo computadores Pentium III 500MHz ou superior, com qualquer sistema operacional e navegador Internet Explorer 5.5 ou superior, Mozilla Firefox 1.5 ou superior ou Opera 8.0 ou superior.

C - VARIABILIDADES DO SISTEMA

V1 - Consultor AdHoc A presença de consultores *adhoc* pode ser opcional para o sistema. Uma família da linha de produtos pode optar por:

i não gerenciar consultores *adhoc*; ou

ii gerenciar consultores *adhoc* permitindo que o usuário Consultor se cadastre no sistema e emita parecer. Inclui a possibilidade do usuário Gestor utilizar as funcionalidade de gerenciar Consultores *AdHoc* e gerenciar os pareceres.

V2 - Pendências A presença de verificação de pendências pode ser opcional para o sistema. Uma família da linha de produtos pode optar por:

i não verificar pendências para enquadramento da proposta durante a submissão; ou

ii verificar pendências e, caso a proposta tenha pendências, não permitir a submissão.

V3 - Membros A presença de outros membros, além do coordenador, vinculados à proposta é opcional para o sistema. Uma família da linha de produtos pode optar por:

i não permitir o vínculo de outros membros à proposta; ou

ii permitir o vínculo de outros pesquisadores à proposta na forma de membros do projeto com funções específicas.

V4 - Atividades A presença de uma listagem detalhada de atividades de um cronograma físico vinculadas à proposta é opcional para o sistema. Uma família da linha de produtos pode optar por:

i não permitir o vínculo de atividades à proposta; ou

ii permitir o vínculo de atividades que são realizadas no decorrer da execução do projeto.

V5 - Orçamento A presença de um gerenciador de orçamento vinculado à proposta de determinada modalidade é opcional para o sistema. Uma família da linha de produtos pode optar por:

i não gerenciar orçamento; ou

ii permitir a gerência de diárias; e/ou

iii permitir a gerência de material de consumo; e/ou

iv permitir a gerência de passagens; e/ou

v permitir a gerência de material permanente; e/ou

vi permitir a gerência de serviços de terceiros (pessoa jurídica ou física).

V6 - Recursos de Outras Fontes A presença de recursos de outras fontes vinculado à proposta de determinada modalidade é opcional para o sistema. Uma família da linha de produtos pode optar por:

i não gerenciar recursos de outras fontes; ou

ii gerenciar recursos de outras fontes.

Glossário

Para todo documento de requisitos há necessidade de um Glossário de Termos técnicos do domínio de negócio. Um exemplo de Glossário para o sistema descrito no documento de requisitos é apresentado a seguir.

Termo	**Descrição**	**Sinônimos**
Backup	Cópia de segurança ou cópia de salvaguarda.	
Pesquisador	Usuário do sistema que pode submeter propostas de projetos.	
Gestor	Usuário do sistema e funcionário da fundação de fomento responsável pelo acompanhamento das propostas de projeto.	
Consultor *adhoc*	Usuário do sistema responsável por julgar e emitir parecer sobre propostas de projetos	
Modalidades de apoio	São os tipos de propostas de projetos que o sistema aceita.	Tipicamente são: *projetos de pesquisa*, *eventos* e *publicações*.

Tabela 3.1: Glossário.

3.4.2 Modelo de Casos de Uso

A partir do documento de requisitos são identificados os seguintes casos de uso, conforme Tabela 3.2. Uma vez identificado os casos de uso é necessário identificar os Atores principais do sistema, conforme apresentado a seguir:

- **Pesquisador**:
  - Responsável por criar e submeter propostas de projetos para julgamento.
  - Responsável por fazer consultas sobre seus projetos ou os que participa como membro.
  - Responsável em atualizar a proposta com informações de acompanhamento de projeto em andamento.
- **Gestor**:
  - Responsável por gerenciar Pesquisadores e Consultores *AdHoc*, caso o sistema apresente esta variabilidade.
  - Responsável por gerar relatórios quantitativos e qualitativos.
  - Responsável por gerenciar propostas de projetos submetidas para julgamento por pesquisadores.
  - Responsável por atribuir propostas de projetos a Consultores *AdHoc*, caso o sistema apresente esta variabilidade.
  - Responsável por gerenciar instituições.

  - Responsável por gerenciar tipos de publicações.

- **Consultor *AdHoc***:

  - Responsável por julgar e emitir parecer técnico sobre as propostas de projetos.

- **Temporizador**:

  - Dispositivo responsável por verificar pendências em propostas de projetos.
  - Dispositivo responsável por emitir alertas e notificações do sistema.

Caso de Uso	**Tipo**	**Categoria**	**Descrição**
Gerenciar Pesquisadores	Primário	Núcleo	Consulta, edição e ativação/desativação de pesquisadores
Gerenciar Gestores	Primário	Núcleo	Criação, alteração, busca e remoção de usuários gestores
Gerenciar Consultores *AdHoc*	Primário	Alternativo	Consulta, edição e ativação/desativação de consultores
Submeter Propostas	Primário	Núcleo	Criação, edição, remoção e submissão de propostas de projetos
Gerenciar Membros	Secundário	Opcional	Criação, edição, busca e remoção de membros vinculados as propostas
Gerenciar Atividades	Secundário	Opcional	Criação, edição, busca e remoção de atividades vinculadas as propostas
Gerenciar Diárias	Secundário	Alternativo	Criação, edição, busca e remoção de diárias vinculadas as propostas
Gerenciar Material de Consumo	Secundário	Alternativo	Criação, edição, busca e remoção de material de consumo vinculados as propostas
Gerenciar Passagens	Secundário	Alternativo	Criação, edição, busca e remoção de passagens vinculados as propostas
Gerenciar Material Permanente	Secundário	Alternativo	Criação, edição, busca e remoção de material permanente vinculados as propostas
Gerenciar Serviços de Terceiros	Secundário	Alternativo	Criação, edição, busca e remoção de serviços de terceiros vinculados as propostas
Gerenciar Recursos de Outras Fontes	Secundário	Alternativo	Criação, edição, busca e remoção de recursos de outras fontes vinculados as propostas
Gerência de Propostas	Primário	Núcleo	Busca e edição de propostas de projetos
Gerência de Editais	Primário	Núcleo	Criação, busca, edição e remoção de editais
Mudar Situação	Primário	Núcleo	Alterar a situação de uma proposta submetida
Atribuir Propostas a Consultores	Primário	Opcional	Atribuir propostas de projetos para serem avaliadas por consultores
Emitir Parecer	Primário	Opcional	Emissão de parecer para propostas de projetos
Gerenciar Modelos de Pareceres	Primário	Opcional	Criação, edição, busca e remoção de modelos de parecer a serem vinculados a editais
Atribuir Parecer	Primário	Opcional	Atribuição de um modelo de parecer a um edital

Caso de Uso	**Tipo**	**Categoria**	**Descrição**
Gerenciar Instituição	Secundário	Núcleo	Criação, edição, busca e remoção de instituições
Gerenciar Tipos de Publicações	Secundário	Opcional	Criação, edição, busca e remoção de tipos de publicações.
Gerenciar Tipos de Anexo	Secundário	Opcional	Criação, edição, busca e remoção de tipos de anexo.
Cadastro de Pesquisador	Primário	Núcleo	Cadastro e edição de dados pessoais de pesquisador.
Cadastro de Consultor *AdHoc*	Primário	Núcleo	Edição de dados pessoais de consultores *adhoc*.
Gerenciar Anexos	Secundário	Núcleo	Criação, edição, busca e remoção de anexos vinculados a propostas
Verificar Pendências	Primário	Opcional	Verifica se os campos de determinada proposta estão devidamente preenchidos
Emitir Alertas	Primário	Núcleo	Emite alertas aos usuários do sistema em função de determinadas ações realizadas ou eminentes
Gerenciar Alertas	Primário	Núcleo	Gerencia quais alertas estarão ativos

Tabela 3.2: Identificação dos Casos de Uso

Para exemplificar, os diagramas de casos de uso a seguir foram elaborados a partir das informações apresentadas e são ilustrados nas Figuras 3.8, 3.9, 3.10 e 3.11.

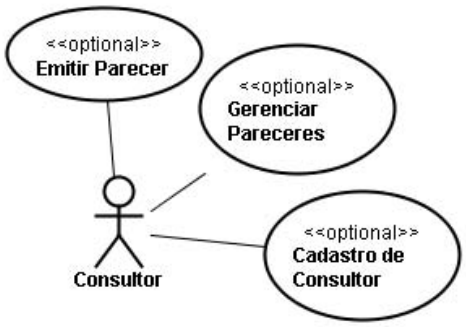


Figura 3.8: Diagrama de casos de uso para o ator Consultor *AdHoc*.

Cada caso de uso apresenta um estereótipo para representar a sua categoria de reuso. Os casos com estereótipo <<*kernel*>> fazem parte do núcleo do sistema, portanto, estão presentes em todos os membros da LPS; aqueles com estereótipo <<*optional*>> podem ser ou não usados por um membro da LPS; e, aqueles com estereótipo <<*alternative*>> correspondem a escolhas alternativas que podem ser feitas para cada membro da LPS.

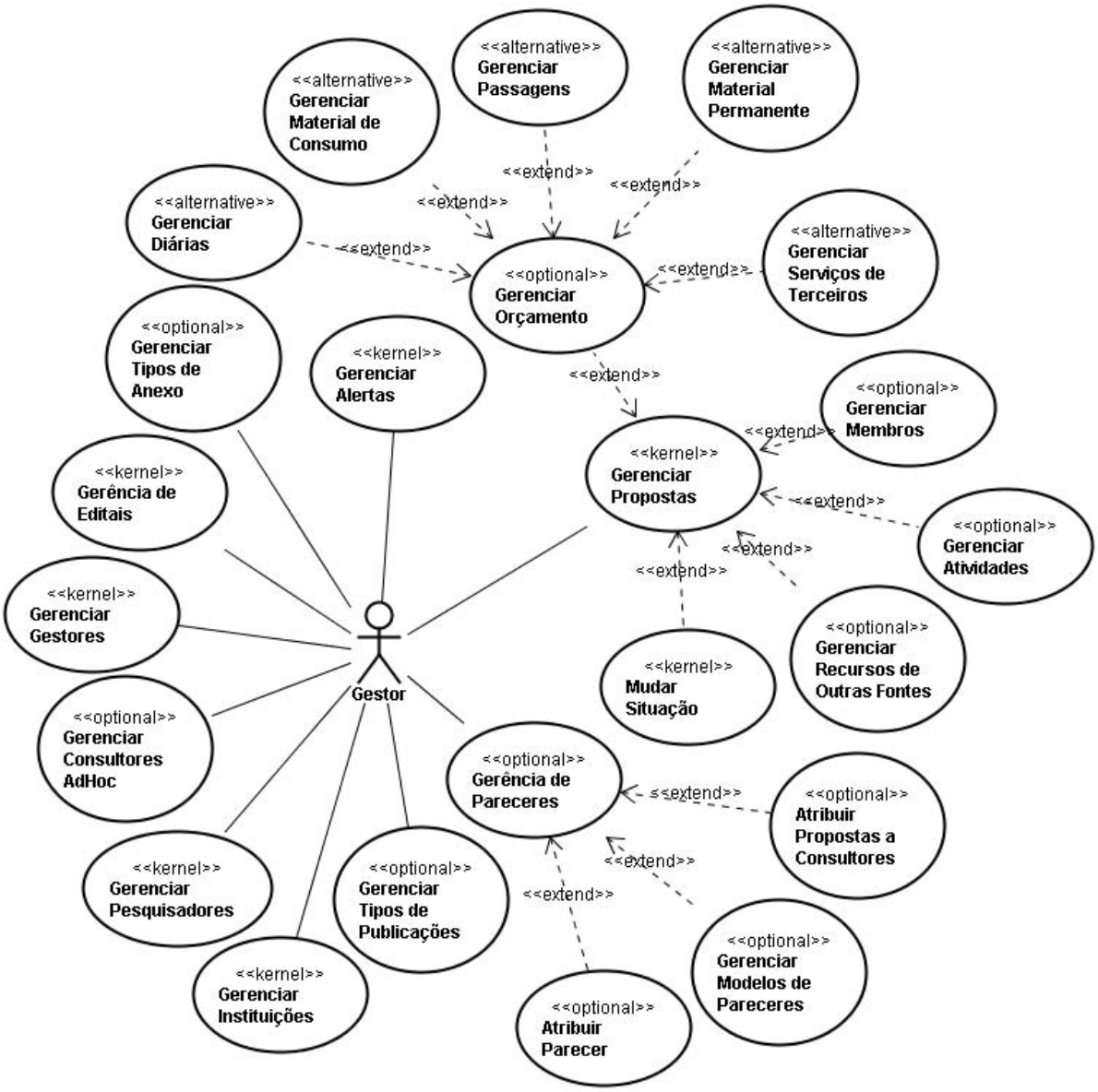


Figura 3.9: Diagrama de casos de uso para o ator Gestor.

3.4.3 Modelo de Features

O objetivo nesta fase é integrar os casos de uso em pacotes, segundo algum critério. Todos os casos de uso que são núcleo (estereótipo <<*kernel*>>) ficam em um mesmo pacote denominado Fomento Kernel, como observado na Figura 3.12.

Em seguida, para cada variabilidade possível da LPS são projetadas as *features* em separado. A variabilidade *V1 - Consultor AdHoc* é mostrada em uma *feature* opcional, sendo que seu pacote é formado por oito casos de uso, conforme ilustrado na Figura 3.13.

As variabilidades *V2 - Pendências*, *V3 - Membros*, *V4 - Atividades* e *V6 - Recursos de Outras Fontes* são *features* opcionais e seus casos de uso respectivos são ilustrados na

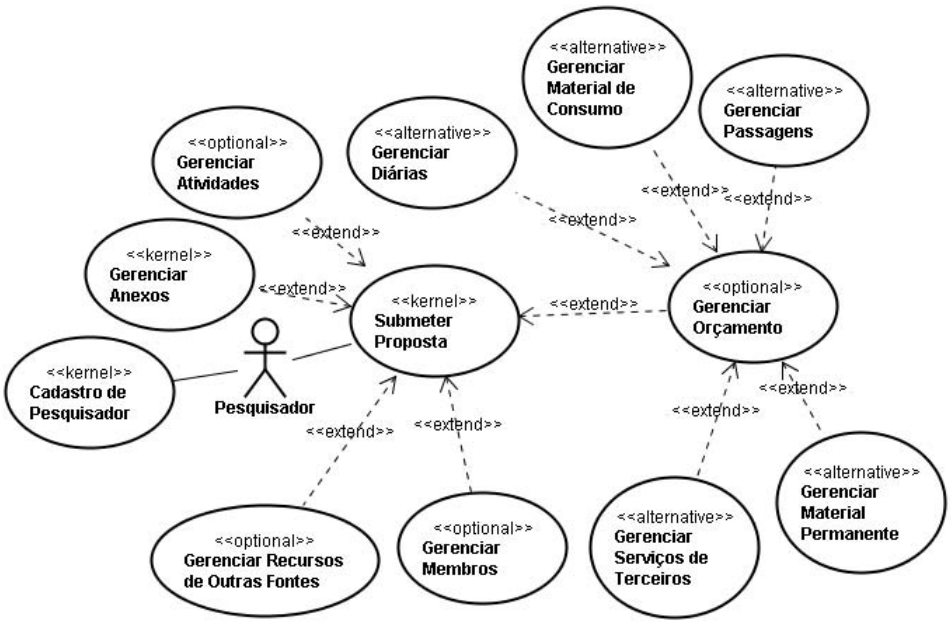


Figura 3.10: Diagrama de casos de uso para o ator Pesquisador.

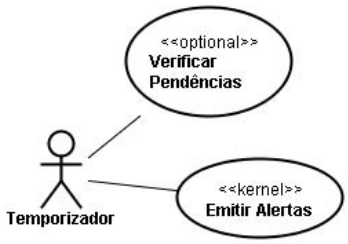


Figura 3.11: Diagrama de casos de uso para o ator Temporizador.

Figura 3.14.

A variabilidade *V5 - Orçamento* é dividida em seis *features*, a *feature* opcional Orçamento e as *features* alternativas Diárias, Material de Consumo, Passagens, Serviços de Terceiros e Material Permanente, como visto na Figura 3.15.

Identificar dependências entre features

Após identificar as relações entre casos de uso e *features* são identificadas as dependências entre as *features*. Inicialmente, verifica-se que todas as variabilidades requerem a parte núcleo da LPS, ou seja, a <<*common feature*>> Fomento Kernel. Dessa forma, tem-se para cada uma das variabilidades:

<<*optional feature*>> Consultor *AdHoc* {*prerequisite* = Fomento Kernel}

<<*optional feature*>> Pendências {*prerequisite* = Fomento Kernel}

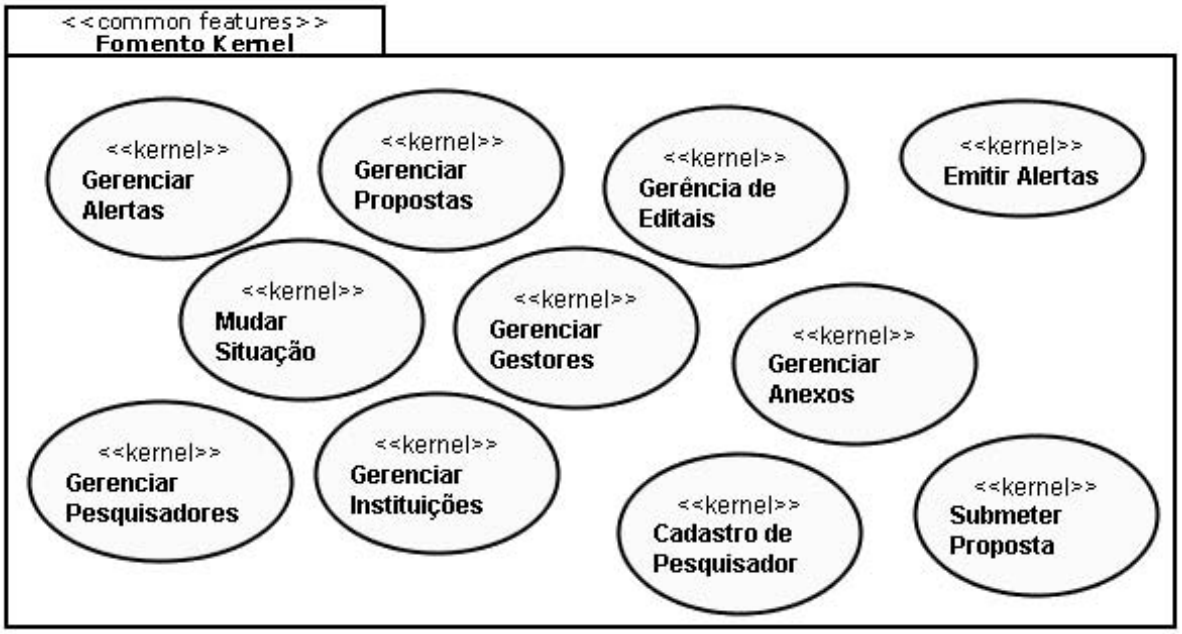


Figura 3.12: Pacote de casos de uso para a *feature* comum Locadora Kernel.

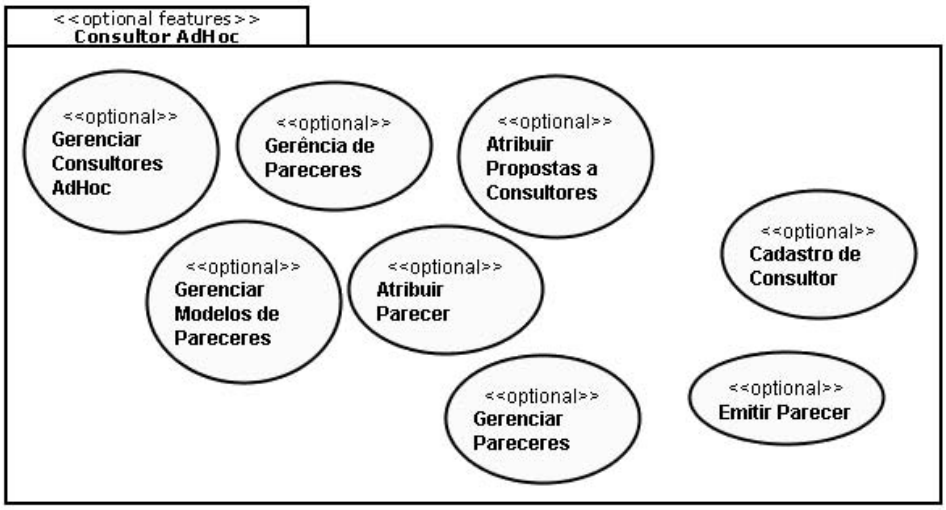


Figura 3.13: Pacote de casos de uso para a *feature* opcional Consultor *AdHoc*.

<<*optional feature*>> Membros {*prerequisite* = Fomento Kernel}

<<*optional feature*>> Atividades {*prerequisite* = Fomento Kernel}

<<*optional feature*>> Orçamento {*prerequisite* = Fomento Kernel}

<<*optional feature*>> Recursos de Outras Fontes {*prerequisite* = Fomento Kernel}

Para a variabilidade *V3 - Orçamento*, existem cinco *features* alternativas: Diárias, Material de Consumo, Passagens, Serviços de Terceiros e Material Permanente. Elas têm como pré-requisito a *feature* opcional Orçamento. Portanto, tem-se:

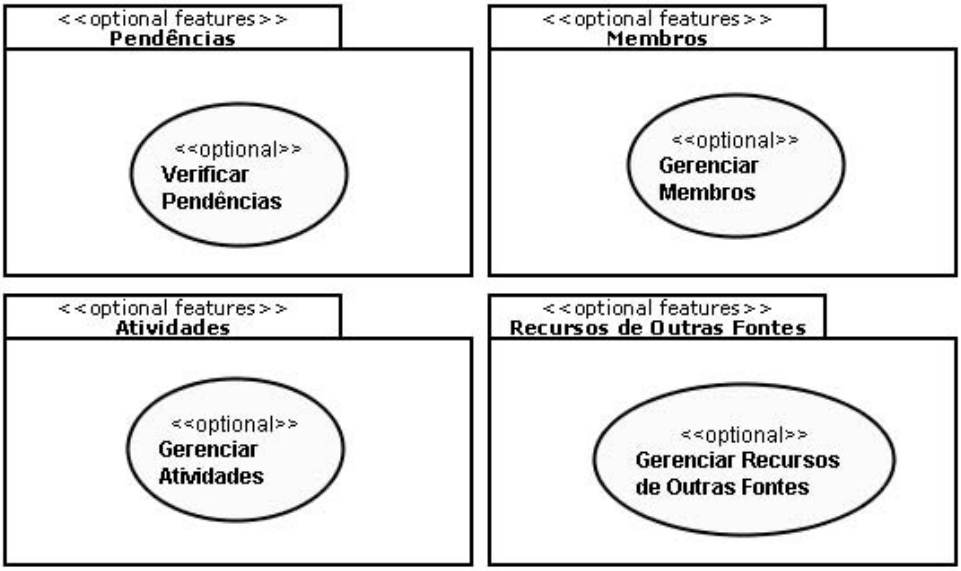


Figura 3.14: Pacotes de casos de uso para as *features* opcionais Pendências, Membros, Atividades e Recursos de Outras Fontes.

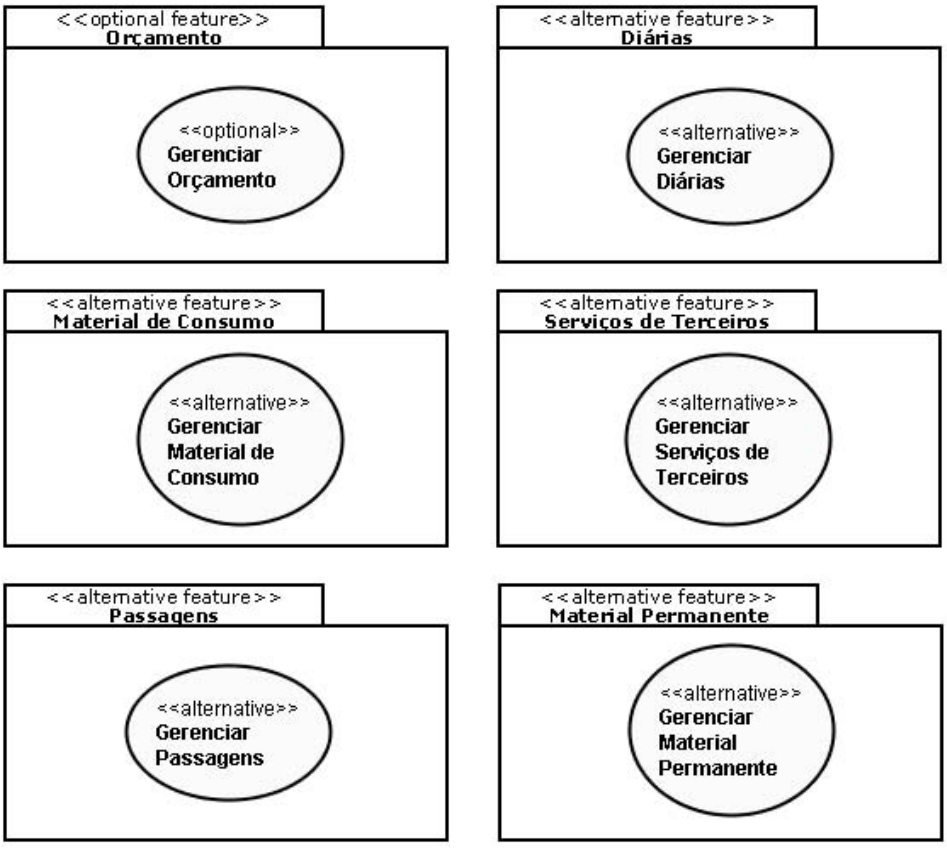


Figura 3.15: Pacote de casos de uso das *features* relacionadas a Orçamento.

<<*alternative feature*>> Diárias {*prerequisite* = Orçamento}

<<*alternative feature*>> Meterial de Consumo {*prerequisite* = Orçamento}

<<*alternative feature*>> Passagens {*prerequisite* = Orçamento}

<<*alternative feature*>> Serviços de Terceiros {*prerequisite* = Orçamento}

<<*alternative feature*>> Material Permanente {*prerequisite* = Orçamento}

Existem, conforme ilustrado na Figura 3.16, cinco modalidades de orçamento. Caso a *feature* opcional Orçamento seja selecionada deve existir pelo menos uma modalidade e as cinco *features* alternativas podem co-existir em qualquer membro da LPS.

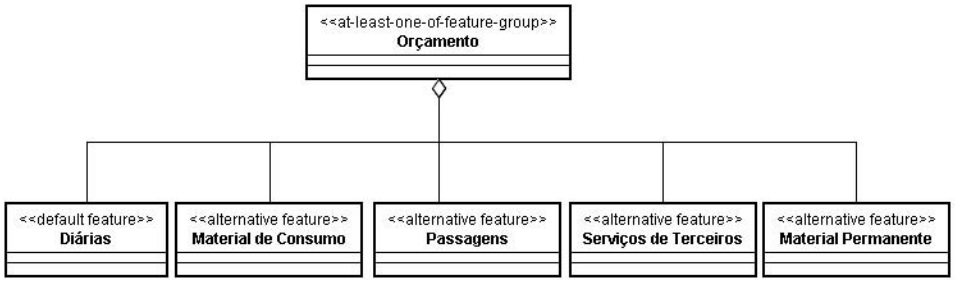


Figura 3.16: Dependências entre as *features* da variabilidade de Orçamento.

Associando todas as *features* e grupos de *features*, pode-se chegar ao modelo de dependências de *features* da LPS da família de SAGF, conforme Figura 3.17.

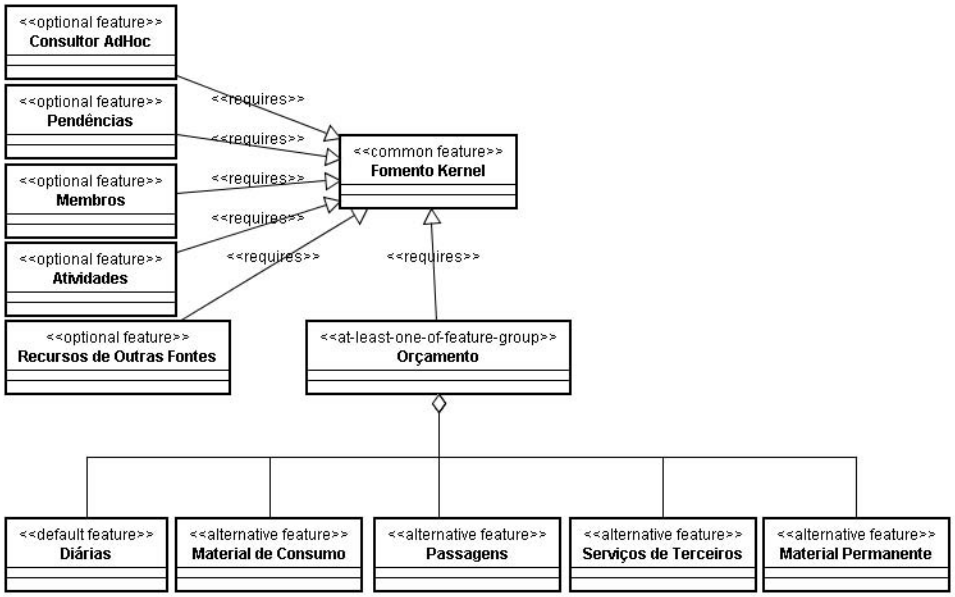


Figura 3.17: Dependências de *features* da Linha de Produto de SAGF.

3.4.4 Modelo Estático e Modelo Dinâmico

A abordagem de modelagem estática inicia com a modelagem de classes do mundo real, que são determinados a partir do domínio do problema, fazendo assim o modelo estático de conceitos. Em seguida é feita a modelagem de classes de software da linha de produtos de software.

No modelo de conceitos da linha de produtos para SAGF, as classes ficam com o estereótipo <<*entity*>>. As classes apresentam um segundo estereótipo representando a categoria de reúso, como foi feito anteriormente para os casos de uso e para as *features*, com a mudança de que , ao invés de <<*alternative*>> é usado <<*variant*>>. Após realizar o modelo de classes do mundo real, pode-se partir para o modelo das classes de software para a linha de produtos. Permanecem apenas as classes que tinham o estereótipo <<*entity*>>. O modelo estático de classes de entidade é mostrado na Figura 3.18.

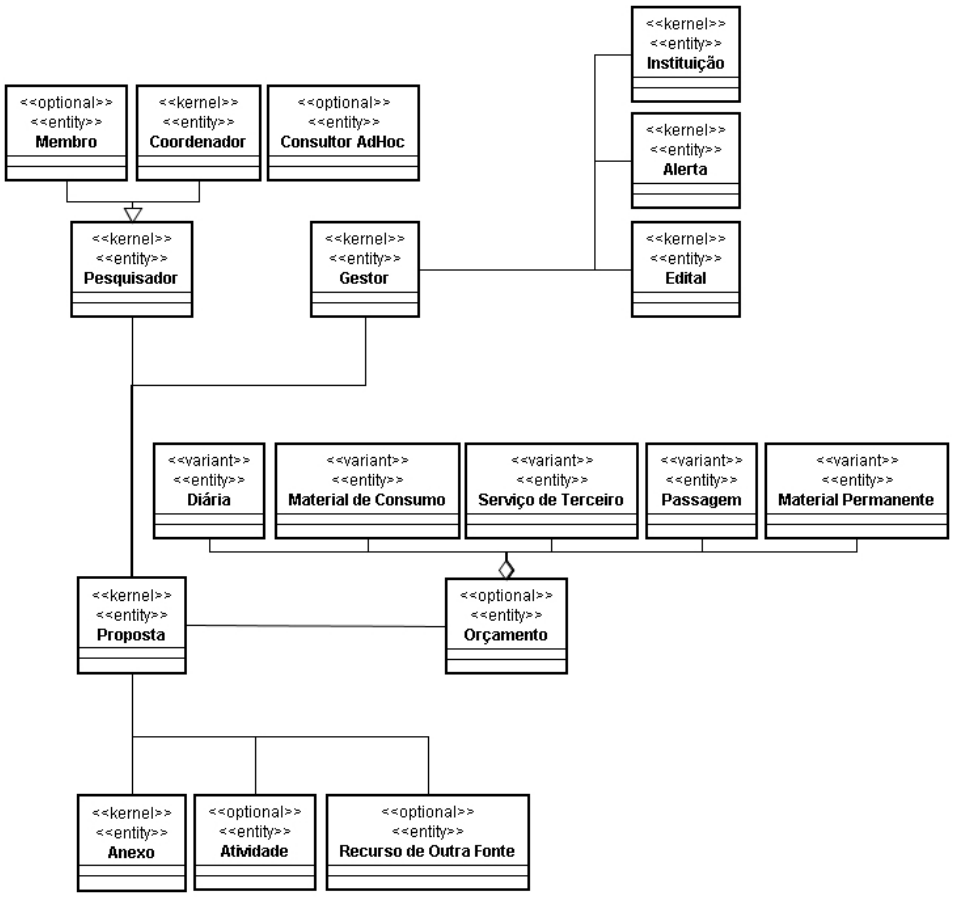


Figura 3.18: Modelo estático de classes de entidade.

Nos diagramas deste modelo são mostrados os componentes relacionados ao componente principal do caso de uso, assim como, todos os componentes de variabilidades possíveis, ou seja, componentes opcionais e alternativos relacionados. Para o caso de um membro que não tenha alguma dessas variabilidades, desconsideram-se os componentes das variabilidades e as mensagens relacionadas a eles.

3.4.5 Arquitetura da LPS

Ao se fazer os diagramas de comunicação pode-se montar a arquitetura de software da linha de produto baseada em componentes. Os componentes são agrupados em subsistemas de acordo com os seus objetivos e as comunicações entre componentes são feitas pelas portas de comunicação. A Figura 3.19 mostra as classes que são agrupadas para formar componentes. Por exemplo, o componente **Usuário** é formado pelas classes Pesquisador, Gestor e Consultor *AdHoc*.

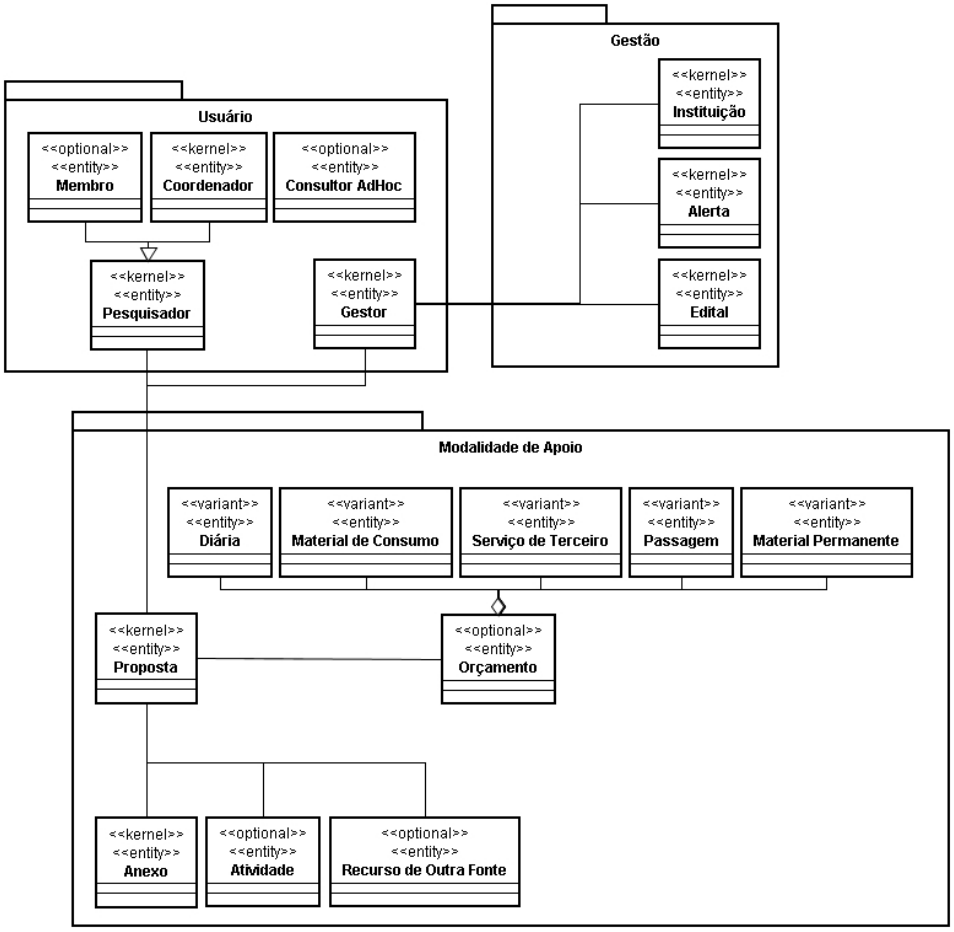


Figura 3.19: Arquitetura baseada em componentes da LPS de SAGF.

3.5 Engenharia da Aplicação

Durante a engenharia da aplicação é desenvolvida uma aplicação individual, membro da FPS. Este processo faz uso dos artefatos desenvolvidos durante a engenharia da LPS. As fases da engenharia da aplicação são:

1. Modelagem de Requisitos: O modelo de requisitos da aplicação é definido em termos de atores e casos de uso. Os requisitos são comparados com o modelo de *features* da

LPS para determinar quais *features* serão incorporadas à aplicação. Uma aplicação típica consiste de todas as *features* do núcleo e algumas *features* opcionais e alternativas. Um protótipo pode ser desenvolvido para ajudar na análise dos requisitos;

2. Modelagem de Análise: Nesta fase são desenvolvidos os modelos estático e dinâmico da aplicação. Somente as classes e os relacionamentos da LPS relevantes para este membro da FPS são selecionados;

3. Modelagem Arquitetural: Nesta fase a arquitetura da aplicação é adaptada da arquitetura da LPS. Para as *features* parametrizadas, os parâmetros correspondentes são aplicados;

4. Implementação Incremental da Aplicação: A abordagem de implementação incremental da aplicação consiste na seleção de um subconjunto da aplicação para ser implementado em cada iteração. Alguns componentes já existem no repositório LPS, outros deverão ser construídos. A implementação de um novo componente consiste do desenvolvimento da arquitetura detalhada, codificação e teste das novas classe do subconjunto; e

5. Teste: Esta fase inclui o teste de integração e funcional da aplicação. No teste de integração as interfaces dos componentes que participam de cada caso de uso são testados. No teste funcional a aplicação é testada em função dos requisitos funcionais.

Enfim, no próximo Capítulo é apresentada a proposta do ambiente de geração de aplicações que contém ferramentas para contribuir com a engenharia da aplicação, automatizando a criação de novos membros da FPS.

3.6 Considerações Finais

Neste Capítulo foram apresentados os principais sistemas utilizados pelas FAPs que compõem a FPS do domínio de negócio explorado neste projeto. Foi observado que existem poucos sistemas semelhantes, e cada membro da FPS possui funcionalidades específicas das necessidade regionais de cada FAP, o que mostra o potencial da proposta de LPS no domínio. Também foi apresentada uma visão geral do processo LPS e da abordagem PLUS utilizada para aplicar a LPS no domínio de SAGF. Os vários modelos da Engenharia de LPS foram ilustrados a fim de compreender a aplicação do processo ESPLEP.

Capítulo 4

Proposta do Ambiente de Geração de Aplicações

4.1 Considerações Iniciais

A aplicação do processo PLUS para gerar artefatos reutilizáveis no domínio da família de produtos de SAGF é fundamental para definir os padrões e especificar os componentes reutilizáveis do Repositório da LPS. Porém, para implementar os componentes do Repositório a fim de sistematizar um processo de reutilização é fundamental a existência de ferramentas automatizadas.

Neste Capítulo é apresentada a arquitetura do Ambiente de Geração de Aplicações formada por duas ferramentas: o *framework* caixa-branca para gerenciamento de conteúdo denominado **Titan** que permite a especificação e a implementação dos artefatos no Repositório no domínio de SAGF e a ferramenta **Fênix** que funciona como um *Wizard* para instanciação do *framework* caixa-branca com base nos padrões existentes, gerando WebApps neste domínio. Assim, os elementos que compõem a arquitetura da LPS são: (1) os modelos da FPS gerados pela abordagem PLUS, que auxiliam a especificação e o projeto de novos produtos da LPS; (2) os padrões do domínio de gestão de fomento de projetos; e (3) o *framework* Titan e a ferramenta Fênix que formam o ambiente de geração de aplicações.

4.2 Arquitetura

A ferramenta Fênix e o *framework* Titan são artefatos de software que automatizam a Engenharia da Aplicação provendo uma maneira dinâmica de reusar os componentes do Repositório da LPS. Conforme ilustrado na Figura 4.1, a arquitetura do ambiente gerador de aplicações provê uma interface para auxiliar o desenvolvimento de novos membros da FPS. Utilizando a ferramenta Fênix, o Engenheiro da Aplicação pode instanciar o *framework* Titan gerando uma nova aplicação no domínio de SAGF. Este processo envolve a geração de código (XML e SQL) que parametriza componentes do repositório da LPS.

A ferramenta e o *framework* fazem uso de componentes do repositório e podem, por meio dele, obter novos componentes com novas funcionalidades na medida em que a LPS evolui.

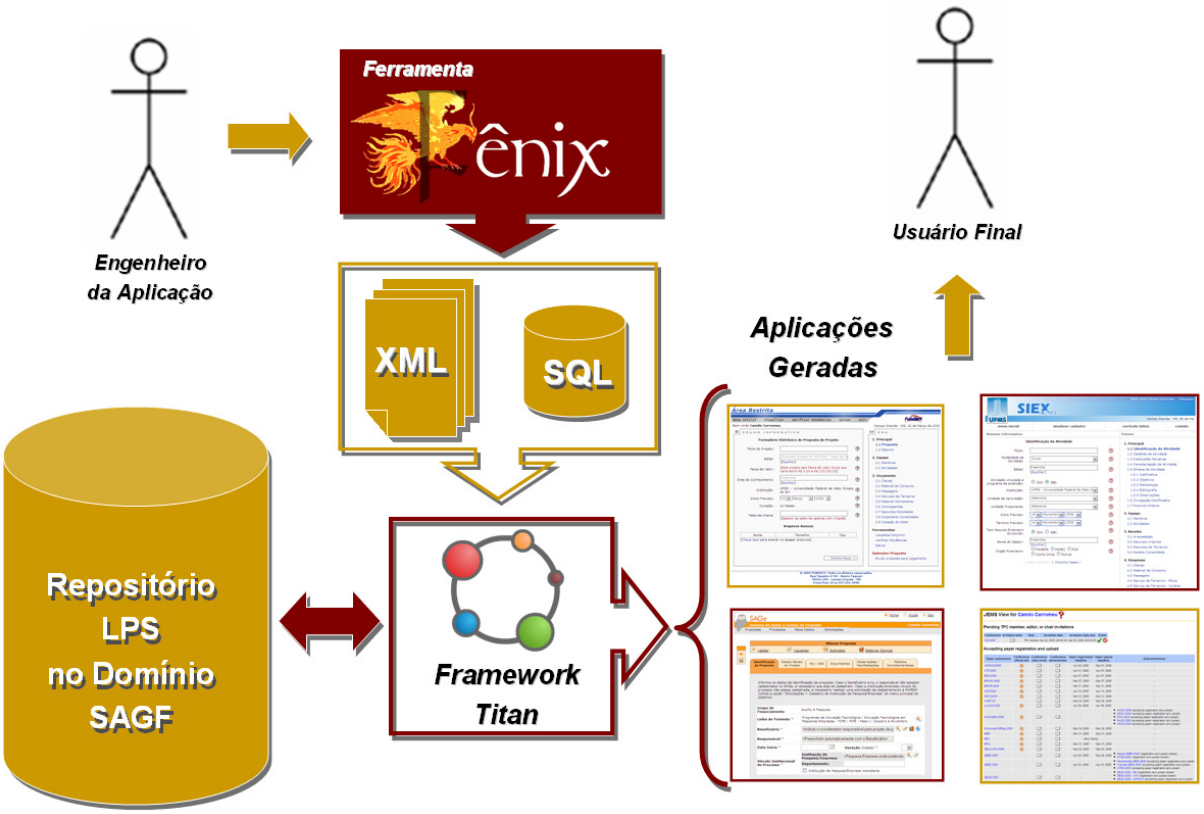


Figura 4.1: Ambiente de geração de aplicações.

A seguir são detalhados os artefatos de software *framework* Titan e ferramenta Fênix.

4.3 *Framework* Titan

O Titan é um *framework* de aplicações criado e desenvolvido pelo grupo de pesquisa do LEDES-DCT/UFMS como sistema gerenciador de conteúdo para aplicações Web. Um sistema gerenciador de conteúdo (*Content Management System* - CMS) têm por finalidade separar o gerenciamento das páginas de conteúdo do projeto de interface (design gráfico). O design do layout das páginas é definido utilizando modelos (*templates*), enquanto o conteúdo é armazenado em banco de dados ou em arquivos textos independentes do *template*. Quando um usuário solicita o carregamento de uma página, as partes são combinadas para produzirem a página HTML padrão. A página resultante pode incluir conteúdos de diferentes fontes.

Segundo Pereira e Bax [72], o processo subjacente dos CMSs é organizado em três etapas básicas: criação, gestão e publicação. O CMS tem como objetivo permitir que os próprios usuários colaboradores, no papel de autores, alimentem o sistema com novos dados sem a necessidade de intermediários ou usuários com conhecimento de tecnologia.

Em seguida, estes dados são armazenados em repositórios para serem tratados (gerenciados, padronizados, formatados e publicados) pelo CMS. O CMS deve gerir também as revisões, atualizações e o controle de acesso, garantindo confiabilidade ao que é publicado e segurança quanto à propriedade e a autoria dos dados.

Para a especificação do *framework* Titan foi utilizada a metodologia de desenvolvimento de projetos dirigidos por exemplos proposta por Johnson [61]. Por meio da revisão de uma série de sistemas de gerenciadores de conteúdo anteriormente implementados pôde-se especificar o *framework* Titan, que permite especificar uma arquitetura de software que recebe como entrada uma linguagem de marcação estendida (XML - *Extensible Markup Language*) e a transforma, em tempo de execução, em um gerenciador de conteúdo. A arquitetura do Titan possui como característica ser caixa-branca, flexível e estensível para outros domínios. Desta forma, se uma determinada funcionalidade não pode ser instanciada pelo *framework* é possível criar um novo componente para a respectiva funcionalidade. A linguagem genérica de entrada se encarrega de garantir que a configuração deste novo componente possa ser realizada como nos demais e que funcione de maneira harmônica com os outros componentes.

Neste trabalho, o Titan foi estendido e adaptado para as necessidades da LPS. O desenvolvimento do processo LPS envolveu, portanto, a extensão do *framework* Titan, criando-se novos componentes e tipos para englobar os padrões do domínio de gestão de fomento de projetos. Por ser um *framework* do tipo caixa-branca este processo torna-se possível e sustentável. Para tanto foi realizado um mapeamento entre os padrões obtidos por meio da análise dos sistemas da FPS no domínio SAGF e as funcionalidades já existentes no *framework*. Após ser estendido, foi realizada a implementação de uma instância do Titan, utilizando os novos padrões e criando novos componentes para atender ao domínio de aplicações de SAGF. Esta instância serviu de protótipo para o desenvolvimento da ferramenta Fênix que automatiza todo o processo de instanciação. Um exemplo de instância do *framework* Titan para o domínio de sistemas da FAP é ilustrada na Figura 4.2, que apresenta uma tela para o sistema da FAPESP-São Paulo no contexto de submissão de propostas de projetos de pesquisa.

4.3.1 Arquitetura

A arquitetura do *framework* é formada por um núcleo (*core*) e um repositório, conforme ilustrada na Figura 4.3. O núcleo tem como característica ser imutável, indiferente da configuração da instância no domínio. Ele é responsável por receber como entrada os arquivos de configuração da instância (XML e SQL) e gerar uma aplicação em tempo de execução. Assim, 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, que são descritos nesta seção. Além do núcleo existe um outro conjunto de código, que não precisa ser replicado, denominado repositório (*repository*), responsável por armazenar as classes que podem ser estendidas do *framework*. Estas classes estão organizadas de acordo com os contextos (tipos e componentes) explicados nesta seção.

Figura 4.2: Tela de uma instância do Titan-FAPESP no domínio de SAGF.

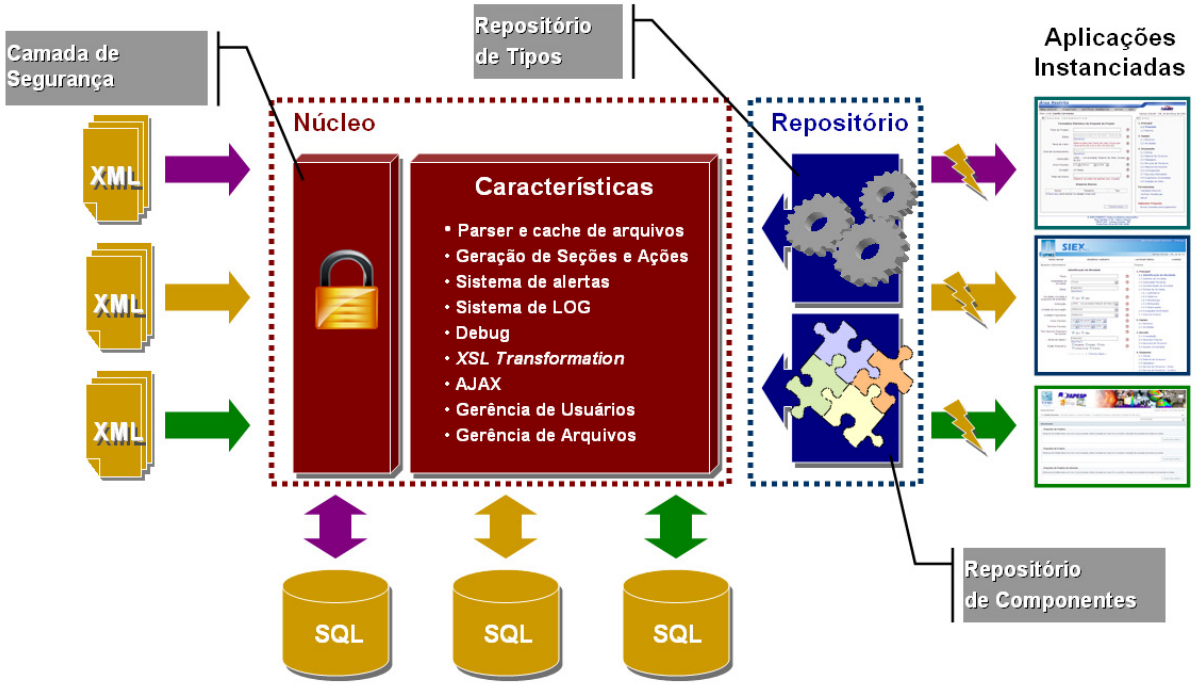


Figura 4.3: Arquitetura do *framework* Titan.

Assim, 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. Na instância fica também um único arquivo de execução que efetua chamadas ao núcleo. Após a execução deste arquivo, ele irá buscar o arquivo principal de configuração, onde estão os caminhos para o núcleo e para o repositório, e irá incluir os ar-

quivos de ambos de acordo com a necessidade. Um pedaço deste arquivo de configuração, localizado na instância em *configure/titan.xml*, pode ser observado abaixo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<titan-configuration
name="Fundect OnLine"
description="Sistema de gestão do Portal Fundect OnLine"
url="http://www.fundect.ms.gov.br/"
e-mail="fundect@fundect.ms.gov.br"
login-url="http://www.fundect.ms.gov.br/titan.php?target=login"
core-path="../../core/"
repos-path="../../repository/"
cache-path="../../cache/"
debug-mode="false"
session="3801792ef7a619386fc19eca65bb45f7"
use-chat="false">

<database
sgbd="PostgreSQL"
host="localhost"
name="fundect"
user="fundect"
password="*****"
schema="titan"
/>
...
</titan-configuration>
```

O Titan foi construído com uso de tecnologias de Engenharia de Software Livre (*Open Source*) [84] e fez uso de técnicas e ferramentas baseadas no conceito de desenvolvimento de comunidades de software livre, sendo que seu código e sua documentação estão disponíveis na wiki do projeto (http://wiki.ledes.net). A linguagem de programação utilizada é o PHP [31] e optou-se por utilizar a linguagem de marcação XML para configuração das instâncias. Assim, todas as instâncias no servidor usam o mesmo conjunto de código de execução (arquivos PHP) localizado na mesma pasta do servidor e cada instância tem uma pasta separada com suas específicas configurações (arquivos XML). Além de diminuir o espaço em disco no servidor eliminando repetição de código, esta estratégia permite que uma atualização do código do *framework* afeta ao mesmo tempo todas as instâncias no servidor, independente da sua configuração. Todo o código do Titan está disponível em um repositório público com controle de versões. Desta forma, além de se obter o código fonte, é possível atualizar versões do *framework* com um único comando do controlador de versões.

Tipos

Os Tipos (*types*) são classes responsáveis por tratar os tipos de dados no *framework*. Quando um formulário é criado ou é necessário carregar um dado da base de dados, este deve ser convertido para um objeto do *framework* e em seguida o Titan consegue manipular seus diferentes usos, tais como, em formulários HTML, arquivos PDF, cálculos e validação. Por exemplo, se o usuário deseja gerenciar um dado do tipo DATA (mais tecnicamente, um tipo *timestamp*), o *framework* converte este tipo para um objeto da classe *Date*, aplicando conversores de tipos e máscaras necessárias. Após ter sido carregado o respectivo tipo pode-se utilizá-lo em diferentes situações na aplicação, por exemplo, no cálculo de períodos de tempo ou idade.

Componentes

Uma WebApp possui vários propósitos ou categorias (Informativas, Interativas, Transacionais, *Workflow*, Colaborativas, Comunidades on-line e Portais). Uma **seção** da WebApp é definida como uma área específica em função de seu contexto. Por exemplo, alguma seções existentes em vários portais são: notícias, eventos, enquete, fale conosco e contato.

Para gerenciar cada uma das seções da aplicação existe uma entidade seção no *framework* Titan. Por exemplo, uma possível área de notícias do website 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 com funcionalidades específicas.

Os componentes do Titan são armazenados no Repositório de Componentes e quando os componentes existentes não conseguem atender a demanda de determinada funcionalidade do website podem ser criados novos componentes ou estendidos os existentes. 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.

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

A definição das seções da instância e o mapeamento destas com seus respectivos componentes são realizados pela configuração do arquivo *configure/titan.xml*. A seguir é apresentada a definição de seis seções, sendo que para cada seção propriedades são configuráveis. Uma destas propriedades (*component*) define o mapeamento para o respectivo componente do Repositório.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<section-mapping>
```

```
<section
label="Página Inicial"
name="home"
component="global.home"
default="true"
/>

<section
label="Notícias"
name="news"
component="global.generic"
/>

<section
label="Agenda"
name="calendar"
component="global.generic"
/>

<section
label="Controle de Acesso"
name="access"
component="global.group"
admin="true"
/>

<section
label="Usuários Gestores"
name="manager"
component="global.userPrivate"
father="access"
/>

<section
label="Usuários Colaboradores"
name="farmer"
component="global.userProtected"
father="access"
/>
</section-mapping>
```

Cada seção possui uma pasta na instância que parametriza o componente. O caminho para esta pasta é determinado pelo nome da seção. Por exemplo, para a seção Notícias acima, a pasta está localizada em *section/news/*. Cada pasta de configuração possui um arquivo principal denominado *config.inc.xml*. Este arquivo mapeia as ações da seção com seu respectivo motor. A seguir são apresentadas as ações da seção Notícias.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<action-mapping>
<!-- Seção Notícias -->
<action
name="list"
engine="list"
label="Listar Notícias"
default="true"
description="">
<menu action="create" />
<menu function="search" />
<menu function="rss" />
</action>

<action
name="create"
engine="create"
label="Criar Notícia"
```

```
description="">
<menu function="save" />
<menu action="list" image="close.png" />
</action>

<action
name="edit"
label="Editar Notícia"
description="">
<menu function="save" />
<menu action="list" image="close.png" />
</action>

<action
name="delete"
label="Apagar Notícia"
description="">
<menu function="delete" />
<menu action="list" image="close.png" />
</action>

<action
name="view"
label="Visualizar Notícia"
description="">
<menu action="list" />
<menu function="print" />
<menu action="edit" />
<menu action="delete" />
</action>

<action
name="version"
label="Controle de Versões">
<menu function="search" />
<menu action="list" />
</action>
</action-mapping>
```

O Titan busca no repositório pela pasta do componente especificado na configuração da seção e, mais internamente, pelos arquivos de preparação, visualização e submissão que formam o motor, renderizando com o uso deles a ação. Cada ação, por sua vez, faz uso de outros arquivos XML localizados na pasta da seção de acordo com a necessidade. Cada um desses arquivos podem definir diretrizes, configurar formulários de inserção, modificação ou visualização de itens desta seção, definir listagem de itens, formulários de busca, entre outros. Um exemplo da configuração de um formulário de uma ação de criação de um novo item para a seção Notícias é apresentado abaixo. Neste exemplo, o formulário possui cinco campos e usa a propriedade *<go-to />* para controle da navegação. Além disso, nota-se o mapeamento com o SGBD por meio das propriedades *table* do formulário e *column* dos campos. Cada campo possui ainda uma propriedade *type* que referencia como o dado será tratado em função do seu tipo no *framework*.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<form table="portal.news" primary="id">
<go-to flag="success" action="[default]" />
<go-to flag="fail" action="[same]" />
<field type="String" column="title" label="Título" required="true" max-lenght="256" help="Título da notícia." />
<field type="Date" column="date" label="Data" help="Data em que a notícia será publicada." />
<field type="Text" column="synopsis" label="Chamada" help="Chamada da notícia." />
<field type="Fck" column="text" label="Texto" help="Notícia na íntegra." />
<field type="String" column="source" label="Fonte" max-lenght="256" help="Fonte da qual a notícia foi obtida." />
</form>
```

Enfim, após a configuração dos recursos acima, o *framework* renderiza a tela ilustrada na Figura 4.5.

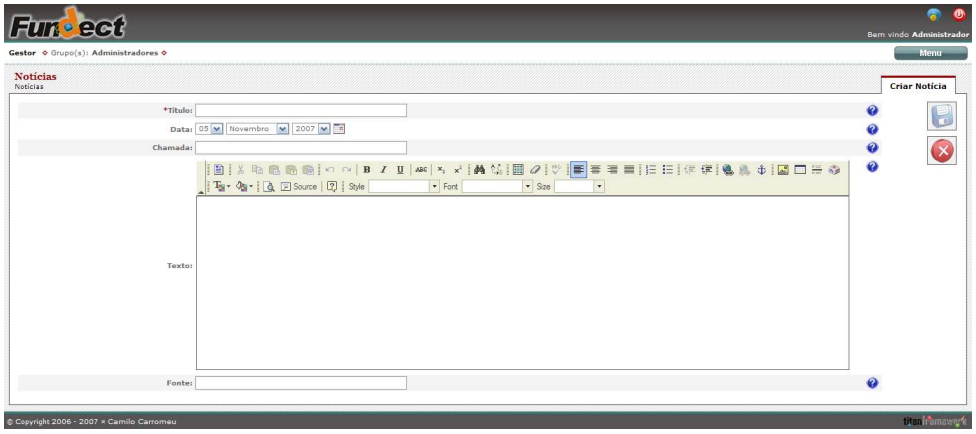


Figura 4.4: Tela da seção "Notícias", na ação "Criar Notícia"de uma instância do Titan.

4.3.2 Características

O *framework* Titan possui uma série de características e funcionalidades embutidas em sua API (*Application Programming Interface*) que permitem ao usuário criar gerenciadores de conteúdo baseados no paradigma Web 2.0 [70]. As principais características são apresentadas a seguir.

a. *Parser* e cache de arquivos de configuração

O *framework* efetua o parser dos arquivos de entrada XML, validando a estrutura por meio de arquivos DTD (*Document Type Definition*) e transformando as informações de configuração da instância em linguagem de máquina (*bytecode*). Para otimizar este processo o *framework* salva o arquivo que já foi realizado o parser para utilizá-lo nas próximas requisições, voltando a realizar o parser se este for modificado.

b. Controle de acesso e autenticação

O Titan possui uma estrutura de controle de acesso baseada em usuários, tipos e grupos. Um usuário do *framework* é uma entidade que representa um usuário real da instância. Esta entidade possui um login único no sistema, um e-mail e uma senha criptografada de forma irreversível [95]. Estas configurações são determinadas pelo tipo do usuário, que define a forma de cadastro de novos usuários daquele tipo, a quais grupos o usuário estará vinculado quando seu cadastro for realizado e quais os campos de dados estarão vinculados ao seu cadastro. Há três formas de cadastro possíveis:

- Pública (*public*): Qualquer usuário pode se cadastrar publicamente para ter acesso à instância;

- Protegida (*protected*): Qualquer usuário pode se cadastrar publicamente, mas requer que outro usuário, com as devidas permissões, habilite seu cadastro para que tenha acesso à instância; e
- Privado (*private*): O formulário de cadastro somente pode ser acessado por usuários da instância com as devidas permissões.

A configuração de tipos de usuários também pode definir se o tipo será autenticado por um serviço de diretórios pelo protocolo LDAP (*Lightweight Directory Access Protocol*) possibilitando que a instância se integre a bases de usuários pré-existentes ou que sirva de ferramenta gestora para estas bases.

O controle de acesso as seções e ações da aplicação instanciada é definido pelas permissões do usuário. As permissões de acesso, no entanto, são vinculadas a grupos e não diretamente a usuários. Cada usuário herda estas permissões dos grupos aos quais está vinculado. Para ter acesso a instância, cada usuário deve estar vinculado a pelo menos um grupo de usuários. Permissões podem ser vinculadas e desvinculadas aos grupos em qualquer momento, afetando todos os usuários do grupo.

c. Busca, paginação e ordenação

A estrutura de classes do *framework* foi implementada para atender o conceito CRUD (*Create, Read, Update and Delete*) de armazenagem persistente (*persistent storage*) [63]. Desta forma, cada componente implementado oferece uma maneira simples e usual de criar, visualizar, editar e apagar itens de suas seções. Além disso, o *framework* oferece, para cada componente baseado neste conceito, a busca, paginação e ordenação dos itens já criados.

d. Gráficos e relatórios quantitativos

Para cada componente CRUD é possível habilitar a geração automática de relatórios quantitativos, que são mostrados pela instância na forma de gráficos, conforme Figura 4.6. Os gráficos são gerados levando em consideração os campos dos formulários.

e. Monitoramento de modificações

Outra funcionalidade implícita às seções geradas por componentes CRUD é a possibilidade de acompanhar os últimos itens inseridos e modificados por meio da utilização de links RSS (*Really Simple Syndication*). Desta forma, o usuário pode utilizar agregadores RSS externos para acompanhar as modificações nas seções sem que seja necessário efetuar login na ferramenta.

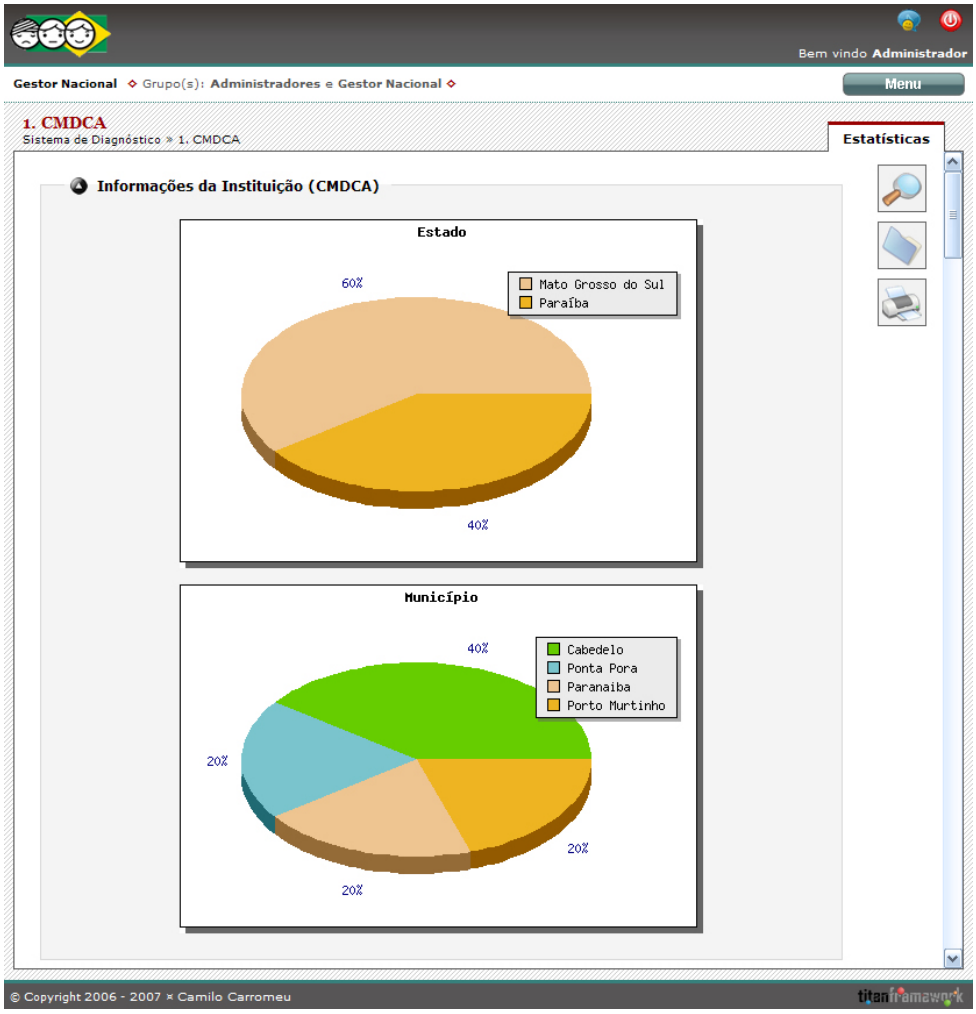


Figura 4.5: Tela dos relatórios quantitativos de uma instância do *framework* Titan.

f. *Asynchronous Javascript And XML*

Ajax (*Asynchronous Javascript And XML*) é uma técnica de desenvolvimento Web utilizada para criar WebApps interativas [52]. Sua utilização no *framework* Titan contribui para manipular dados entre o cliente e o servidor de forma invisível para o usuário. O uso desta técnica faz com que parte da carga de processamento da instância seja distribuída com o cliente, não sobrecarregando o servidor de atividades de renderização desnecessárias de páginas e evitando o recarregamento da página no cliente.

g. Controle de versões e auditoria

Pode ser ativada nas seções de componentes CRUD da instância do *framework* o suporte a auditoria e controle de versões. Quando este recurso está habilitado na seção, o *framework* grava automaticamente todas as modificações realizadas na seção. Isto inclui criação de novos itens, edição e remoção de itens existentes. Juntamente com cada modificação gravada é armazenado o autor da modificação e a data-hora em que ocorreu tal

modificação. Na Figura 4.6 é ilustrada uma tela com as revisões de um item de uma seção versionada em uma instância exemplo.

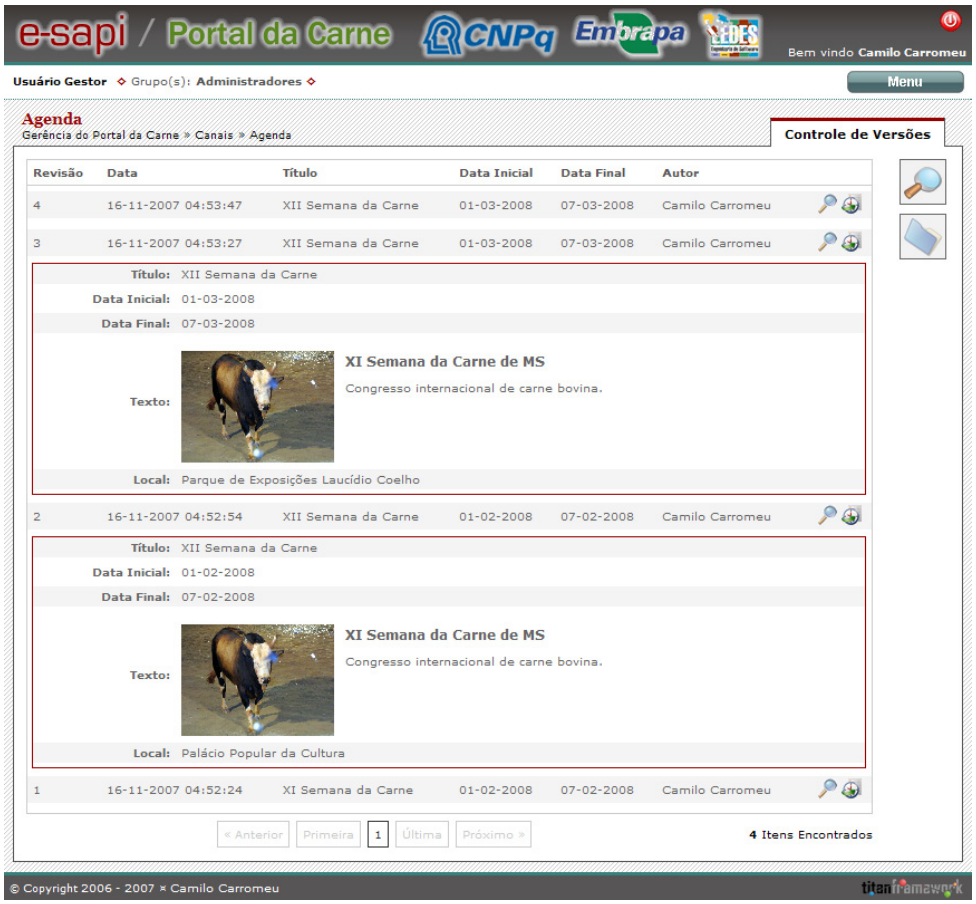


Figura 4.6: Tela de controle de versões e auditoria de um item em uma instância do *framework* Titan.

h. Sistema de *upload* de arquivos

O Titan possui um sistema integrado de vínculo de arquivos. Os arquivos enviados para a instância podem ser pré-estabelecidos por meio de um arquivo de configuração. O *framework* verifica o cabeçalho do arquivo durante seu envio para constatar se o arquivo pode ser enviado ou não ao servidor, aprimorando assim os mecanismos de segurança. Além disso, em qualquer momento arquivos enviados podem ser reutilizados, evitando o reenvio e o espaço em disco no servidor.

Os arquivos da instância podem ser tratados de três formas distintas:

- Imagem: Arquivos de imagem podem ser apresentados na forma de mini-imagens (*thumbnail*), que são pré-visualizações das imagens originais;
- Carregável: Os links gerados para este tipo de arquivo fazem chamada ao programa do cliente que normalmente abre o tipo de arquivo especificado; e

- Download: Os arquivos tratados como arquivos de download ficam disponíveis na instância apenas para serem salvos na máquina cliente.

i. Geração automática para formato de impressão e PDF

A visualização de itens das seções de componentes CRUD oferece, automaticamente, a visualização para impressão do item, removendo cabeçalhos, rodapé e outros itens desnecessários que aumentam o gasto de fita, tinta ou toner da impressora. Também é possível ativar a impressão PDF dos itens, gerando arquivos PDFs com o conteúdo exibido.

j. Sistema de alertas

O *framework* oferece um sistema de alertas de modificações por e-mail. A configuração deste sistema é realizada na instância vinculando o monitoramento das ações que desejar a grupos de usuários. Quando uma ação vinculada for ativada (como, por exemplo, a modificação de um item) os usuários vinculados àquele grupo recebem um alerta por e-mail.

k. Sistema de *debug* e *log*

A API do *framework* foi implementada com o conceito de exceções (*exception handling*). Desta forma, erros são capturados e mostrados ao usuário. Além disso, pode-se configurar a instância para exibir mensagens de erro com informações técnicas ou não (*debug mode*), possibilitando que o sistema reaja de forma diferente durante o processo de instanciação e de produção da aplicação. Caso o sistema esteja configurado para não exibir erros com informações técnicas ele automaticamente armazena tais informações em arquivos de *log*, mantendo registrada a ocorrência e possibilitando a consulta pela equipe técnica responsável.

l. Salas de Chat

Uma instância do *framework* pode ser configurada para habilitar salas de chat (do inglês *chatroom*) para propiciar a conferência síncrona por meio da troca de mensagens entre os usuários da aplicação. Cada grupo de permissões criado no controle de acesso pode ser configurado com uma sala de chat própria. Desta forma, todos os usuários de um grupo específico podem utilizar a sala deste grupo, de acesso restrito a usuários de outros grupos. Além disso, todos os usuários possuem acesso a uma sala denominada "Geral", e se o usuário receber mensagens enquanto o chat estiver fechado ele é avisado por meio de um alerta intermitente. Na Figura 4.7 é ilustrada a ferramenta de chat sendo utilizada na seção de Suporte Técnico.

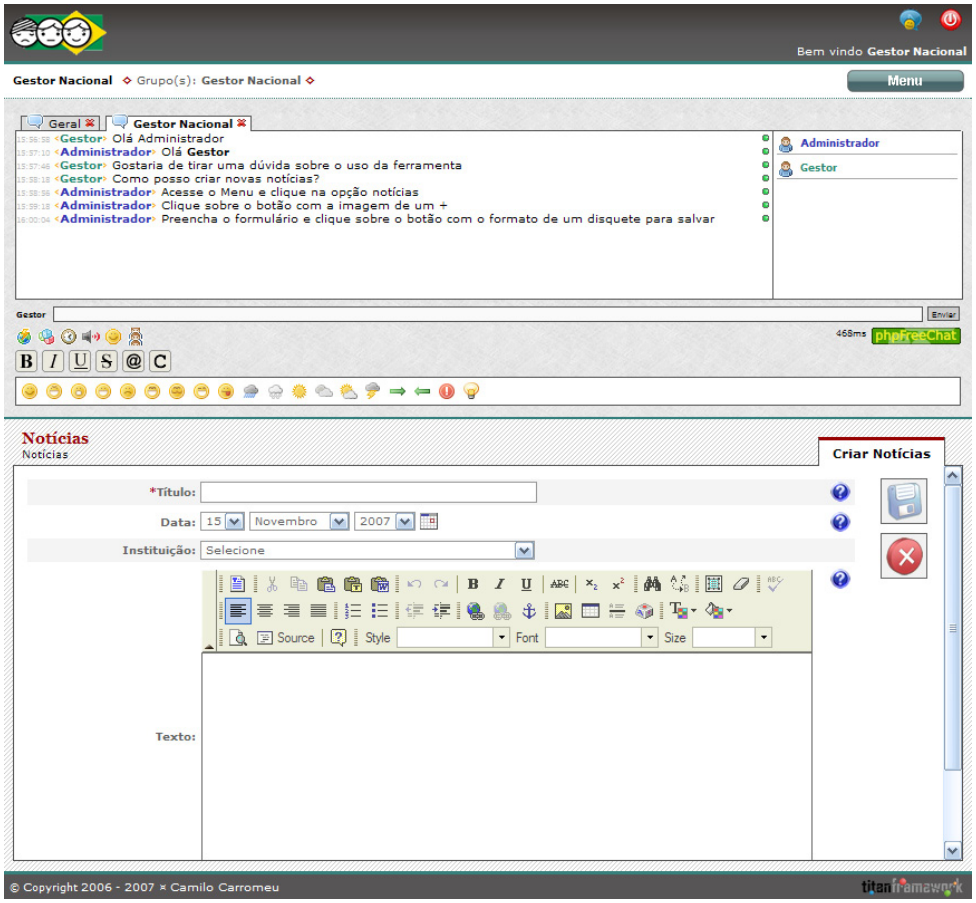


Figura 4.7: Salas de chat em uma instância do *framework* Titan.

m. *XSL Transformation*

A interface de uma instância do *framework* pode ser modificada usando as transformações XSLT (*Extensible Stylesheet Language Transformation*). Toda a saída resultante pode ser convertida para XML e, então, folhas de estilo e arquivos XSL podem ser aplicados para modificar a visualização da instância. Quando a mudança na interface não é necessária este recurso pode ser desabilitado para otimizar o desempenho.

4.4 Ferramenta Fênix

O Fênix é a ferramenta *wizard* que automatiza o processo de LPS para o desenvolvimento ágil de WebApps no domínio de SAGF. Provê uma maneira fácil e rápida para configurar uma instância do *framework* Titan, criando um novo membro para a FPS no domínio.

4.4.1 Arquitetura

A arquitetura do Fênix é baseada em dois sistemas autônomos (Figura 4.8): um sistema gerador e um sistema gerado. O sistema gerador é a própria ferramenta Fênix, enquanto o sistema gerado é a instância do *framework* Titan. O sistema gerador é subdividido em três subsistemas: Sistema de Segurança, Sistema de Gestão e o Gerador de Código. O Sistema de Gestão utiliza os quatro padrões de projeto:

- Formulários: responsáveis pela inserção de dados pelos usuários da aplicação instanciada;
- Atributos: elementos independentes que podem ser utilizados nos formulários da aplicação instanciada;
- Grupos de Usuários: grupos passíveis de iteração com a aplicação instanciada; e
- Módulos: definem os *workflows* de submissão e gerenciamento de propostas eletrônicas. Mais especificamente os módulos são responsáveis por integrar de forma interativa Grupos de Usuários e Formulários em diferentes momentos da aplicação instanciada.

Assim, para gerenciar estes padrões, o Sistema de Gestão é composto por quatro gestores, por meio dos quais o engenheiro da aplicação poderá definir e configurar todo o sistema gerado: Gestor de Usuários, Gestor de Módulos, Gestor de Formulários e Gestor de Atributos. Estes gestores podem ser considerados como ferramentas de configuração dos *features* da LPS. O Sistema de Segurança é responsável pela camada de autenticação do sistema gerador e o Gerador de Código é responsável por gerar, de modo interativo, o código do sistema gerado. Cada um destes sistemas e gestores é apresentado nas próximas seções.

Em relação as tecnologias utilizadas, a arquitetura do Fênix é dividida em três camadas, seguindo os conceitos definidos no modelo MVC (*Model-View-Controller*) [88]. A camada de apresentação, responsável pela interface gráfica com o usuário, foi implementada usando JSP. Essa camada é responsável pela entrada e saída dos dados e comunica-se com a camada de negócio para a obtenção dos dados a serem apresentados na interface, devolvendo informações e realizando as solicitações processadas pela camada de negócio. A camada de negócio é formada por *beans* e classes de ação que manipulam a criação de objetos. As classes de ação foram implementadas a partir do *framework* Struts [58]. Essa camada comunica-se com a camada de persistência quando torna-se necessário armazenar um objeto permanentemente, listar ou alterar objetos persistentes. A camada de persistência utiliza o *framework* Hibernate [8] para realizar toda a interação com a base de dados. Utiliza-se o sistema de banco de dados PostgreSQL [89], entretanto uma das características do Hibernate é oferecer portabilidade para diversos SGBDs (Sistemas de Gerência de Banco de Dados).

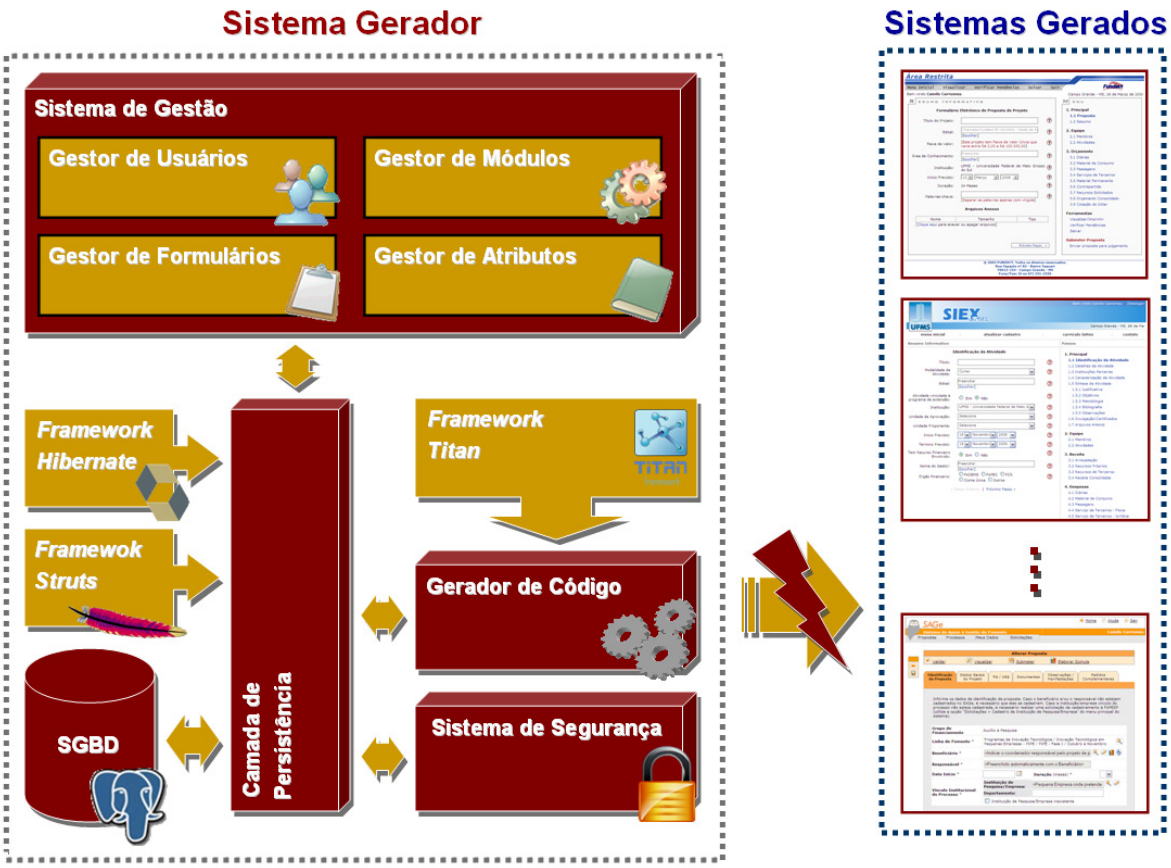


Figura 4.8: Arquitetura do Fênix.

4.4.2 Sistema de Segurança

O Sistema de Segurança é responsável por garantir a segurança das informações e funcionalidades da ferramenta Fênix. O mecanismo de segurança é realizado por um subsistema de controle de acesso, que gerencia as permissões de grupos de usuários sobre as funcionalidades e informações do sistema; e por um subsistema de autenticação, que é responsável por autenticar os usuários do sistema.

O cadastro de novos usuários no sistema é realizado por usuários associados a grupos que tenham tal permissão. Existe um grupo de usuários inicial e permanente chamado Administrador. Usuários pertencentes ao grupo Administrador possuem todas as permissões possíveis sobre todas as funcionalidades e informações do Fênix.

A autenticação do usuário é realizada por meio de um identificador (login) e uma senha, ambos definidos durante o cadastro do usuário. A senha é criptografada e armazenada no banco de dados. Assim, durante o processo de autenticação a senha fornecida pelo usuário deve ser criptografada antes de ser comparada com a informação armazenada no banco de dados. A criptografia utilizada no sistema Fênix é a SHA-1 (*Secure Hash Standard*) [95].

Além do cadastro de usuários, o Sistema de Segurança do Fênix também gerencia

grupos de usuários. As permissões primárias do sistema podem ser atribuídas somente a grupos, e os usuários herdam tais permissões dos grupos que pertencem. Estas permissões estão diretamente relacionadas a funcionalidades específicas do sistema e quando são atribuídas a um grupo pode-se ou não indicar qual aplicação específica a permissão é concedida. Quando não especificada uma aplicação, a funcionalidade não envolve nenhuma ou é válida para todas, dependendo do contexto utilizado.

Cada usuário cadastrado deve ser vinculado a um ou mais grupos para que seja possível herdar as permissões. A lista de permissões possíveis varia de acordo com as funcionalidades como, por exemplo, criar aplicações, cadastrar usuários e criar atributos. Enfim, o sistema de segurança é responsável pelo controle de acessos no Fênix, ou seja, por verificar as permissões de um usuário a cada requisição ao sistema, validando ou não a ação requisitada. Um exemplo do diagrama de casos de uso do Sistema de Segurança do Fênix é ilustrado na Figura 4.9.

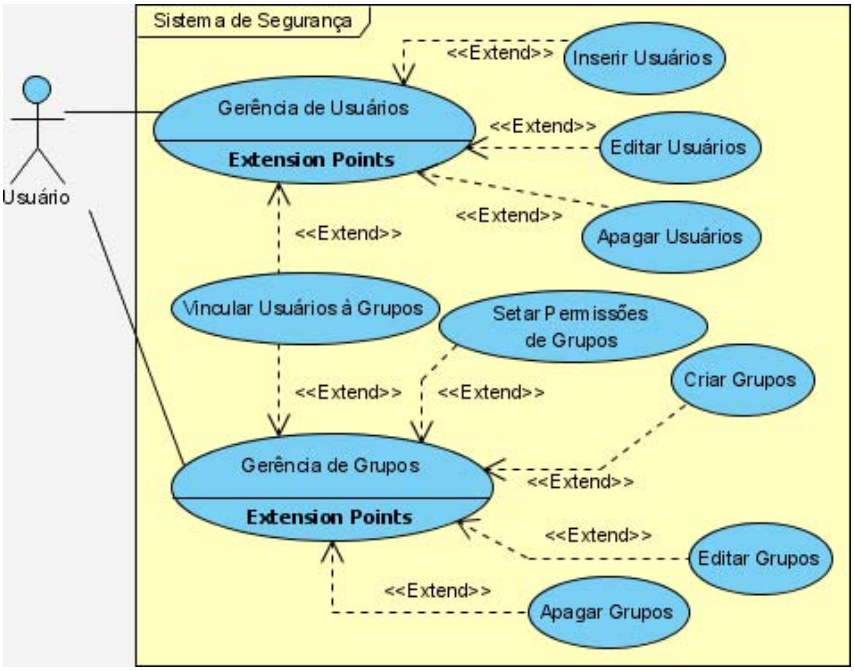


Figura 4.9: Diagrama de casos de uso do Sistema de Segurança do Fênix.

4.4.3 Sistema de Gestão

O Sistema de Gestão do Fênix define toda a infra-estrutura para permitir a configuração das *features* da LPS. O primeiro passo no processo de instanciação do Fênix é a criação de uma nova aplicação. Uma aplicação dentro da ferramenta é uma futura instância do *framework* e pode ser criada por meio da interface apresentada na Figura 4.10.

O segundo passo do processo de instanciação é a configuração da aplicação, que consiste em definir os módulos da aplicação, seus formulários (interfaces gráficas com o usuário), conjunto de atributos necessários a cada interface e os grupos de usuários da aplicação. Na Figura 4.11 é apresentada a interface do Fênix, que contém um menu que direciona o

Figura 4.10: Interface para gerência de aplicações.

usuário a realizar todas essas configurações, ou seja, criar, editar e apagar novos Módulos, Grupos de Usuários, Atributos e Formulários para uma aplicação a ser instanciada. Dessa forma, o sistema de Gestão do Fênix gerencia o funcionamento de quatro outros subsistemas gestores, conforme descritos a seguir: Gestor de Usuários, Gestor de Atributos, Gestor de Formulários e Gestor de Módulos. O diagrama de casos de uso para o Sistema de Gestão é apresentado na Figura 4.12.

Gestor de Usuários

O Gestor de Usuários é responsável pela criação e configuração de todo o sistema de segurança, autenticação e controle de acesso para as aplicações instanciadas a partir do Fênix. Este sistema não deve ser confundido com a funcionalidade responsável pela segurança da ferramenta Fênix. O Gestor de Usuários tem como foco a aplicação instanciada, enquanto o Sistema de Segurança tem como alvo o próprio Fênix.

Este gestor permite criar grupos de usuários, aos quais são atribuídas permissões de acesso à aplicação instanciada. Uma lista pré-determinada de permissões pode ser combinada em cada grupo de usuários a partir da função e dos módulos que esses grupos podem ter acesso e do nível de acesso. O nível de acesso em um módulo pode ser medido pelas permissões que um usuário possui sobre os formulários pertencentes a este módulo. A combinação de um formulário, grupo de usuários e um estado do módulo formam uma "visão", que será definida pelo Gestor de Módulos.

Para gerenciar os usuários de um grupo específico é criado um formulário no Gestor de Formulários que pode ser de três tipos:

Figura 4.11: Menu de seleção de gestores de configuração.

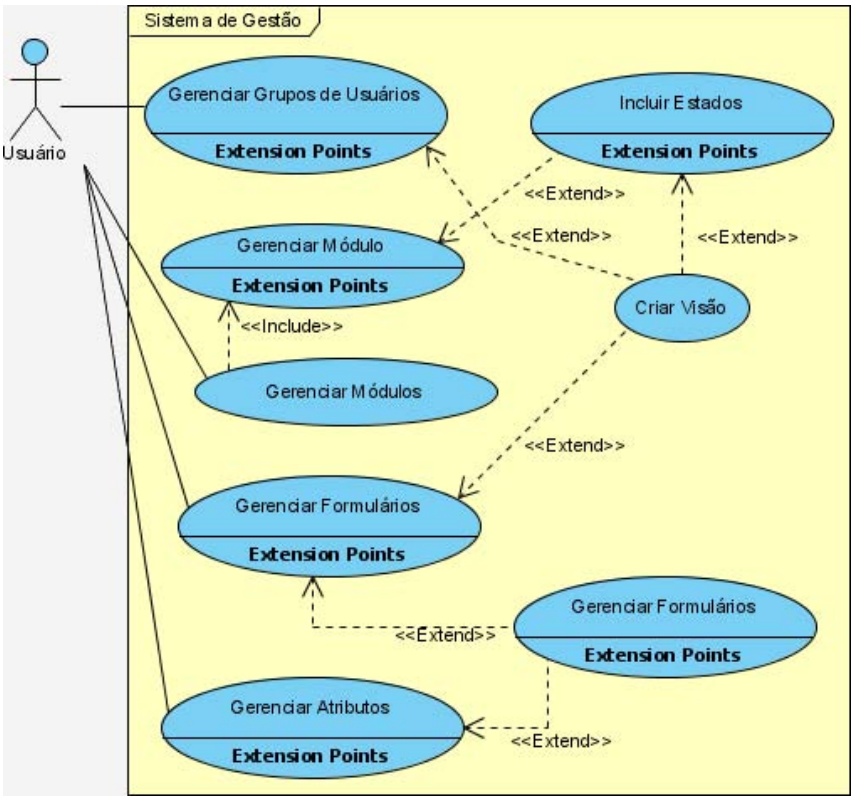


Figura 4.12: Diagrama de casos de uso para o Sistema de Gestão do Fênix.

- Formulário de cadastro público (*public*): o usuário faz cadastro no sistema a partir de um formulário público e pode acessar instantaneamente o sistema a partir de seu

login e senha recém cadastrados;

- Formulário de cadastro protegido (*protected*): o usuário faz cadastro no sistema a partir de um formulário público, mas somente um usuário autenticado pode "aprovar"o cadastro e, desta forma, torná-lo válido ao sistema de segurança;
- Formulário de cadastro privado (*private*): somente usuários autenticados no sistema podem cadastrar o novo usuário naquele grupo.

Assim, o Gestor de Usuários permite criar grupos de usuários como, por exemplo, Pesquisadores, Consultores e Administradores que têm permissões de acesso em diferentes seções[1] de um projeto em andamento tais como: Área Restrita, Julgamento e Administração. Para exemplificar o uso dos diferentes tipos de formulários de cadastro considere o cadastro de Administradores privado, o de Pesquisadores público e o de Consultores protegido. Logo, qualquer pessoa pode se cadastrar como Pesquisador e efetuar login instantaneamente na área restrita de Pesquisador; Administradores só podem ser cadastrados por outros Administradores; e, Consultores podem se cadastrar publicamente, mas seu cadastro deve ser aprovado por outro grupo (como, por exemplo, Administradores).

Gestor de Módulos

Um módulo representa as diferentes modalidades de apoio que são gerenciadas pela aplicação gerada, por exemplo, propostas de projetos de pesquisa submetidos à avaliação de uma agência de fomento. Todo o trâmite, desde a submissão da proposta, sua avaliação, seu acompanhamento técnico e administrativo deve ser gerenciado por meio de uma série de passos, ou seja, os estados desta modalidade de apoio.

Como apresentado na Figura 4.13, a arquitetura genérica de um módulo envolve três abstrações: estado (Ei), transição de estados e visão do sistema. Cada módulo possui um conjunto finito de estados (E0, E1, ..., Ef-1, Ef onde E0 indica o estado inicial e Ef o estado final) que representa o fluxo de execução das tarefas de um módulo (*workflow* linear da vida do módulo). A transição de um estado Ei para Ei+1 é dada pela ocorrência de um evento (gatilho), no qual alguns requisitos devem ser atendidos. Uma transição de estados é representada, na Figura 4.13, por uma seta partindo do estado anterior e apontando para o estado seguinte. Um gatilho pode ser uma data específica, um botão em um formulário ou simplesmente um conjunto de requisitos atendidos. Em todos os casos o sistema verifica se os requisitos pré-determinados foram atendidos (exceto quando o cumprimento de tais requisitos definem o próprio gatilho). No caso de uma submissão de proposta de projeto, a transição de estados determina todo o ciclo de vida da proposta, desde o momento da submissão até o completo encerramento do projeto (ou não-aprovação do mesmo).

Cada estado do *workflow* de um Módulo agrupa um conjunto de "Visões"do sistema. Cada Visão constitui um relacionamento entre um Grupo de Usuários e um Formulário,

[1]Uma seção representa uma área restrita a um Grupo de Usuários, cujo acesso é feito mediante autenticação de um usuário deste grupo.

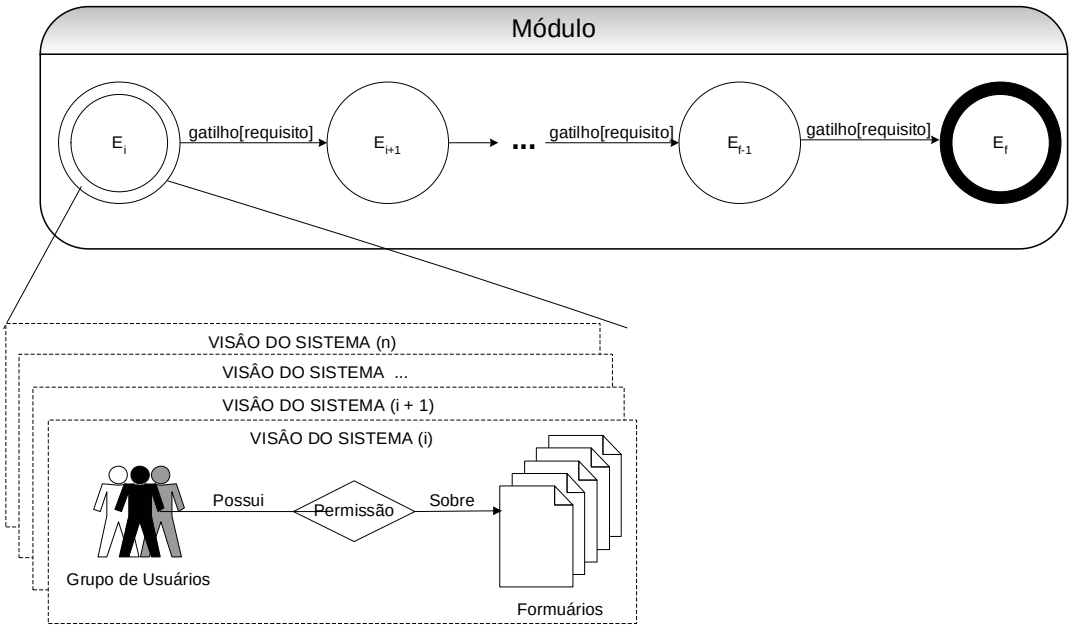


Figura 4.13: Estrutura de um Módulo.

ou seja, representa as permissões que cada Grupo de Usuários têm sobre um determinado Formulário. Um mesmo grupo de usuários pode estar associado a diferentes formulários, formando diferentes Visões no mesmo estado do *workflow*.

O Gestor de Módulos foi desenvolvido de forma a tornar o desenvolvimento do *workflow* transparente para o usuário. Para tal optou-se pelo uso da tecnologia SVG (*Scalable Vector Graphics*) [38], pois a tecnologia permite desenvolver uma interface gráfica onde o desenvolvedor pode manipular de forma simples os estados e as visões de um *workflow* de módulo, aumentando a usabilidade da ferramenta Fênix.

Gestor de Atributos

Uma aplicação desenvolvida no Fênix possui uma série de atributos. Cada atributo é uma entidade que pode ser utilizada em diferentes momentos dentro da aplicação e pode ser de vários tipos dependendo do seu papel na aplicação. São definidos os seguintes tipos primários de atributos: Integer, Date, String, Float e Boolean. Existem tipos estendidos de atributos, que são variações dos tipos primários com funções mais específicas, tais como: CPF, CNPJ, Coin, Password e Text. Os tipos estendidos podem ser criados de acordo com a necessidade, entretanto, isto somente pode ser feito por meio de novas implementações.

Os atributos são independentes das funcionalidades do sistema e isto torna possível a não restrição do seu uso. Assim, um atributo CPF do pesquisador, por exemplo, pode tanto ser utilizado no formulário de cadastro de pesquisador quanto no formulário de autenticação. Apesar de serem dois campos diferentes de dois formulários diferentes o atributo é o mesmo, ou seja, o mesmo valor cadastrado é utilizado para fazer autenticação

no sistema. Ainda assim, um atributo deve estar vinculado a um formulário principal (*form main*) que mapeia a transformação deste atributo em uma coluna no banco de dados relacional. Atributos podem ser atribuídos na forma de campos de formulários e na forma de variáveis de controle. Qualquer campo de formulário deve estar associado a um atributo (que tem seu valor modificado ou visualizado). Na forma de variáveis de controle, os atributos podem ser utilizados para verificação de requisitos de mudança de estados em Módulos e na validação de campos de formulários através de equações lógicas.

Gestor de Formulários

Todos os dados necessários para a avaliação de uma modalidade de apoio durante o seu ciclo de vida são inseridos por meio de formulários. Desde a submissão por meio de pesquisadores, as alterações e observações de funcionários da FAP, o julgamento de consultores, as revisões da equipe financeira e qualquer outras informações relevantes que devem ser inseridas, modificadas ou visualizadas utilizam formulários. Os formulários são responsáveis pela inserção de usuários no sistema gerado (cadastros públicos, privados e protegidos).

O Gestor de Formulários possibilita a criação e edição de formulários para a aplicação. Na criação de um formulário devem, primeiramente, ser definidos os dados básicos do formulário (nome e descrição). Em seguida são definidos os três elementos básicos que compõem a estrutura do formulário:

- Passos: cada formulário é composto por um ou mais passos. Os passos dividem o preenchimento do formulário em páginas a fim de facilitar o preenchimento e aumentar a segurança (já que cada passo é salvo independentemente);
- Grupos: a cada passo podem ser atribuídos um ou mais grupos organizadores. Cada grupo é, basicamente, um elemento organizador utilizado apenas para separar campos em determinados contextos; e
- Campos: um campo é a unidade mais elementar de um formulário. Cada grupo pode agregar um ou mais campos. Um campo tem a função de instanciar um atributo e, desta forma, permitir sua visualização ou edição. Durante a criação do campo o usuário pode definir uma série de características, tais como: nome, descrição, atributo que será instanciado e tipo do campo (visualização ou edição).

Um formulário pode ser associado a uma visão de um determinado estado de algum módulo ou a um grupo de usuários na forma de formulário de cadastro ou edição de dados cadastrais. Em uma visão, o formulário forma uma tríplice juntamente com um grupo de usuários e um estado do *workflow* de módulo.

Um formulário pode ser criado de duas formas: é um formulário *main* ou é uma instância de um formulário *main*. Um formulário *main* é importante pois se tornará uma tabela no banco de dados relacional durante o mapeamento realizado na geração de código da aplicação a ser instanciada. Uma instância de um formulário *main* é uma "cópia"ou

uma "sombra"deste formulário. Este tipo de formulário se tornará uma referência à tabela do formulário *main* do qual foi instanciado, na aplicação gerada.

4.4.4 Gerador de Código

O Gerador de Código do Fênix é responsável por instanciar, de forma interativa, uma aplicação no domínio de SAGF. Basicamente este sistema recupera toda a informação armazenada pelo Sistema de Gestão e a utiliza na criação do banco de dados, arquivos de configuração, metadados e código fonte do sistema gerado.

No Fênix, todo o comportamento da aplicação a ser instanciada encontra-se devidamente armazenada. Cabe ao Gerador de Código recuperar estas informações e transformá-las em uma aplicação concreta com toda a estrutura lógica de regras de negócios e física que o usuário que a instanciou imaginou.

Este gerador está em fase de implementação em outro trabalho de mestrado do DCT-UFMS [96]. Para auxiliar o mapeamento da geração de código estão sendo desenvolvidos protótipos que seguem integralmente a arquitetura da LPS no domínio de SAGF. Um destes protótipos é apresentado no Capítulo 5 como estudo de caso desta pesquisa.

4.5 Tecnologias adotadas

Para o desenvolvimento dos artefatos de software citados neste Capítulo foram adotadas as seguintes tecnologias, tendo como base o paradigma de software livre:

- Java/J2EE/JSP [37]: A linguagem Java e seus derivados formam a base da ferramenta Fênix. Esta tecnologia foi escolhida pela sua portabilidade e pelo fato de ser uma tecnologia que segue o paradigma do software livre, ou seja, sua licença é gratuita e possui código aberto;
- *Framework* Hibernate [8]: Responsável pela camada de persistência do Fênix. Foi escolhido por tornar a aplicação independente de SGBD. Apesar de ter sido adotado o PostgreSQL, o Fênix é totalmente portável, sem a necessidade de implementação para outro SGBD;
- *Framework* Struts [58]: Responsável pela camada de negócios do Fênix. A escolha deste *framework* foi por ser extremamente difundido e estável dentre os vários *frameworks* em linguagem Java responsáveis pelo controle da camada de negócios das aplicações;
- PostgreSQL [89]: SGBD utilizado no desenvolvimento da ferramenta Fênix e do *framework* Titan. Foi escolhido por seguir o paradigma de software livre e se enquadrar como banco de dados robusto e estável; e

- SVG (*Scalable Vector Graphics*) [38]: Tecnologia utilizada no Gerenciador de Módulos do Fênix para criação visual de *workflows*. Esta tecnologia dispõe uma maneira eficiente de geração de interfaces gráficas para Web baseada em linguagem textual. Esta característica foi decisiva na sua escolha para implementação do Gestor de Módulos (Seção 4.4.3);
- XML (*Extensible Markup Language*) [16]: Linguagem de marcação derivada do SGML (*Standard Generalized Markup Language*) para descrição de tipos de dados. Os arquivos XML são gerados pelo Fênix para configuração de novas instâncias do *framework* Titan; e
- PHP [31]: A linguagem PHP está sendo utilizada para implementação de uma instância do *framework* Titan. Por ser livre e de código aberto foi escolhida para mostrar que o Fênix e o Titan, apesar de implementados em diferentes linguagens de programação, podem interagir normalmente utilizando uma linguagem genérica.

4.6 Considerações Finais

Neste Capítulo foi apresentado o ambiente de geração de aplicações para o domínio de SAGF. Com o uso desta plataforma, que envolve a utilização da ferramenta Fênix para gerar instâncias do *framework* Titan, é possível gerar de forma rápida e fácil novos membros para a FPS. O ambiente apresentado neste Capítulo automatiza a fase de Implementação Incremental da Engenharia da Aplicação, guiando o engenheiro da aplicação no processo de geração das aplicações, garantindo que os caminhos certos sejam seguidos, fazendo as devidas consistências para assegurar que os padrões e componentes aplicados sejam coerentes entre si. Portanto, o engenheiro tem condições de saber exatamente onde começar e onde terminar a instanciação. No próximo capítulo é apresentado um estudo de caso explorando o ambiente.

Capítulo 5

Estudo de Caso: Fundect OnLine

5.1 Considerações Iniciais

Neste Capítulo é apresentado um estudo de caso no domínio de SAGF para validar a aplicação do processo e as ferramentas implementadas. Optou-se por utilizar o ambiente gerador de aplicação em um membro da família de produtos já existente (Fundect OnLine - http://www.fundect.ms.gov.br) a fim de tornar possível destacar as dificuldades e as principais necessidades no domínio. Assim, são apresentadas a descrição do problema do sistema Fundect OnLine e a sua instanciação utilizando o ambiente de geração de aplicações.

5.2 Descrição do Problema

O Fundect OnLine é um projeto cujo objeto foi especificar, desenvolver e implementar um ambiente Web para a gestão de projetos de pesquisa e eventos científicos, além do acompanhamento dos processos resultantes, de tal forma que, tanto a FUNDECT quanto os gestores dos projetos tenham acesso restrito e qualificado aos seus dados. Trata-se portanto, de um novo meio de comunicação entre a FUNDECT e os gestores, essencial para a total transparência das informações geradas a partir dos processos gerenciados.

O ambiente Fundect OnLine e o website da FUNDECT foram implementados na plataforma Web utilizando técnicas de Engenharia de Software, Banco de Dados e Linguagem de Programação que permitem a geração dinâmica de páginas de informação. Na Figura 5.1 é apresentado o portal da FUNDECT que permite o acesso ao sistema Fundect OnLine. Para construir um estudo de caso que valide a presente pesquisa partiu-se desta aplicação, fazendo-se uma Engenharia Reversa de forma a modelar as funcionalidades existentes do sistema em um novo produto do domínio.

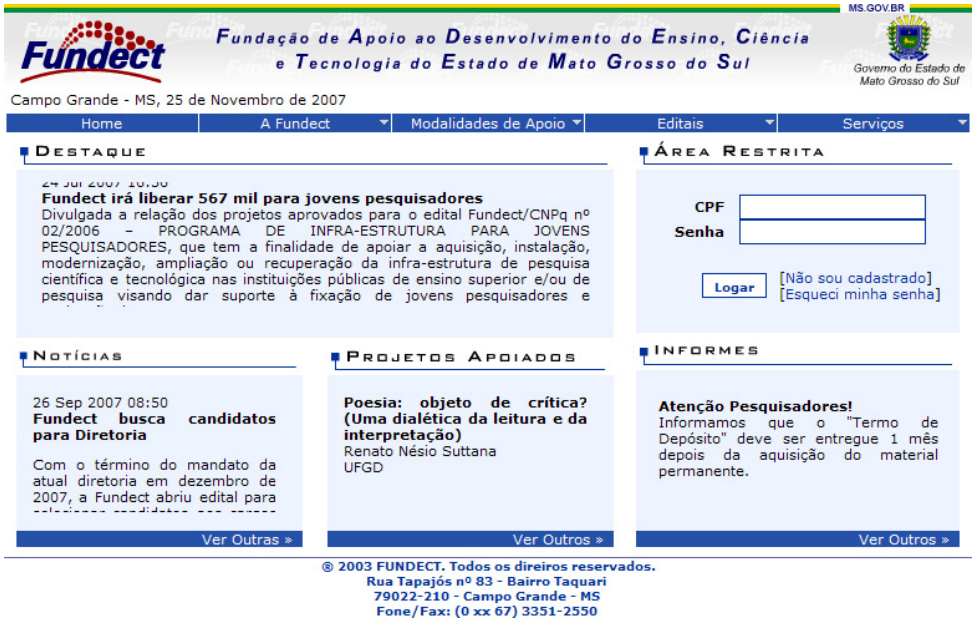


Figura 5.1: Website da FUNDECT-MS.

5.3 Geração da Aplicação

Para gerar a nova aplicação é utilizado o ambiente de geração de aplicações proposto que, por meio da ferramenta Fênix e do *framework* Titan, é configurado o membro da FPS. Como ainda não está finalizado o Gerador de Código do Fênix foi gerado manualmente o código. Assim, a partir de um roteiro passo-a-passo pretende-se exemplificar a geração de novos membros para a família de produtos da LPS no domínio de SAGF.

Na Figura 5.2 é ilustrado o passo 1.1 denominado "Proposta"do formulário eletrônico da modalidade "Projeto de Pesquisa"da Fundect OnLine. Para criar a mesma funcionalidade do ambiente de geração de aplicações a ferramenta Fênix oferece uma interface baseada em passos onde cada característica da funcionalidade pode ser configurada.

Na Figura 5.3 é exemplificada a seqüência de eventos para realizar todas as configurações necessárias.

1. No passo 1 o usuário cria a aplicação denominada "Fundect OnLine";
2. No passo 2 cria os formulários principais (formulário *main*) Eventos, Projetos de Pesquisa e Publicações;
3. No passo 3 cria atributos vinculados ao formulário "Projetos de Pesquisa". Como este formulário é um formulário principal ele é mapeado em uma tabela no banco de dados relacional e cada atributo torna-se uma coluna;
4. No passo 4 cria os passos do formulário "Projetos de Pesquisa". Em destaque está o passo "Proposta";

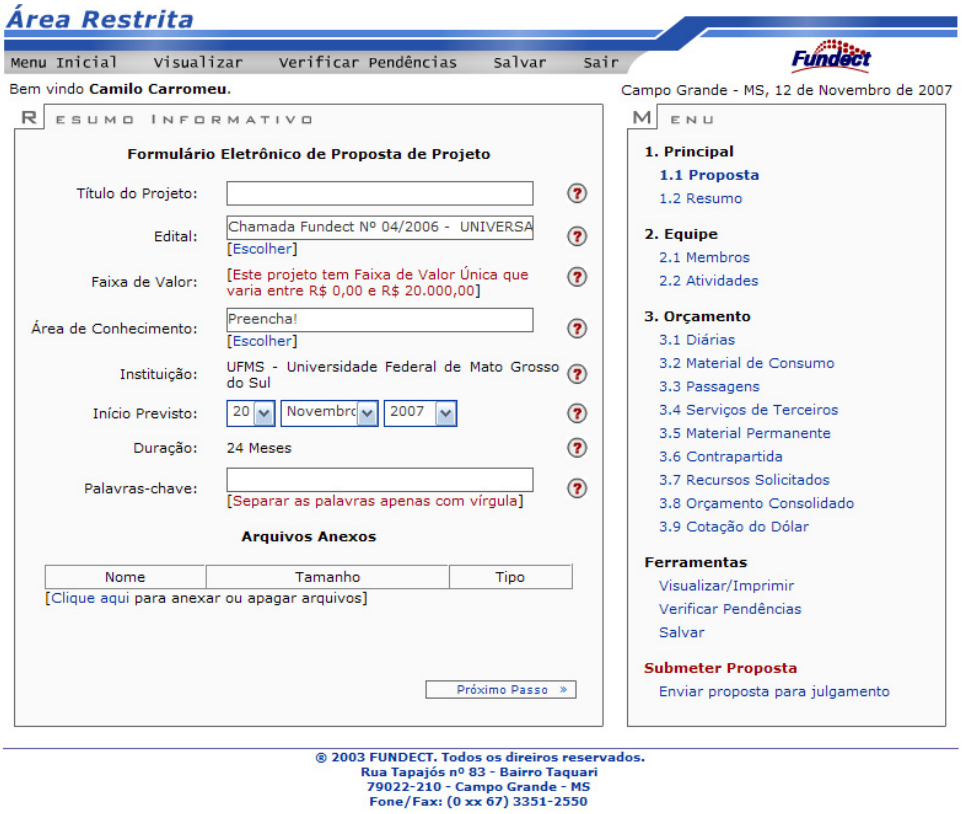


Figura 5.2: "Passo 1.1: Proposta"do formulário eletrônico de Projeto de Pesquisa da Fundect OnLine.

5. No passo 5 cria os grupos aos quais os campos são vinculados. Seguindo a interface que está sendo replicada foram criados os grupos Dados Principais e Arquivos Anexos; e

6. No passo 6 o usuário cria os campos, vinculando os atributos criados no passo 3 aos grupos e atribuindo informações visuais, tais como o rótulo do campo.

As configurações efetuadas no Fênix são salvas no Banco de Dados e, na geração de código são encaminhadas diretamente para os arquivos de configuração do *framework* Titan e para o *script* de geração do Banco de Dados da instância. Na Figura 5.4 é ilustrado o diagrama entidade-relacionamento da estrutura gerada pelo Fênix para a funcionalidade configurada. O arquivo XML que mapeia estas entidades, gerado no exemplo, pode ser visto abaixo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<form table="fenix.projeto_de_pesquisa" primary="_id">
<go-to flag="success-next" action="step_1" />
<go-to flag="fail" action="[same]" />
<go-to flag="success-step_0" action="step_0" />
<go-to flag="success-step_1" action="step_1" />
<go-to flag="success-step_2" action="step_2" />
<go-to flag="success-step_3" action="step_3" />
<go-to flag="success-step_4" action="step_4" />
<go-to flag="success-step_5" action="step_5" />
```

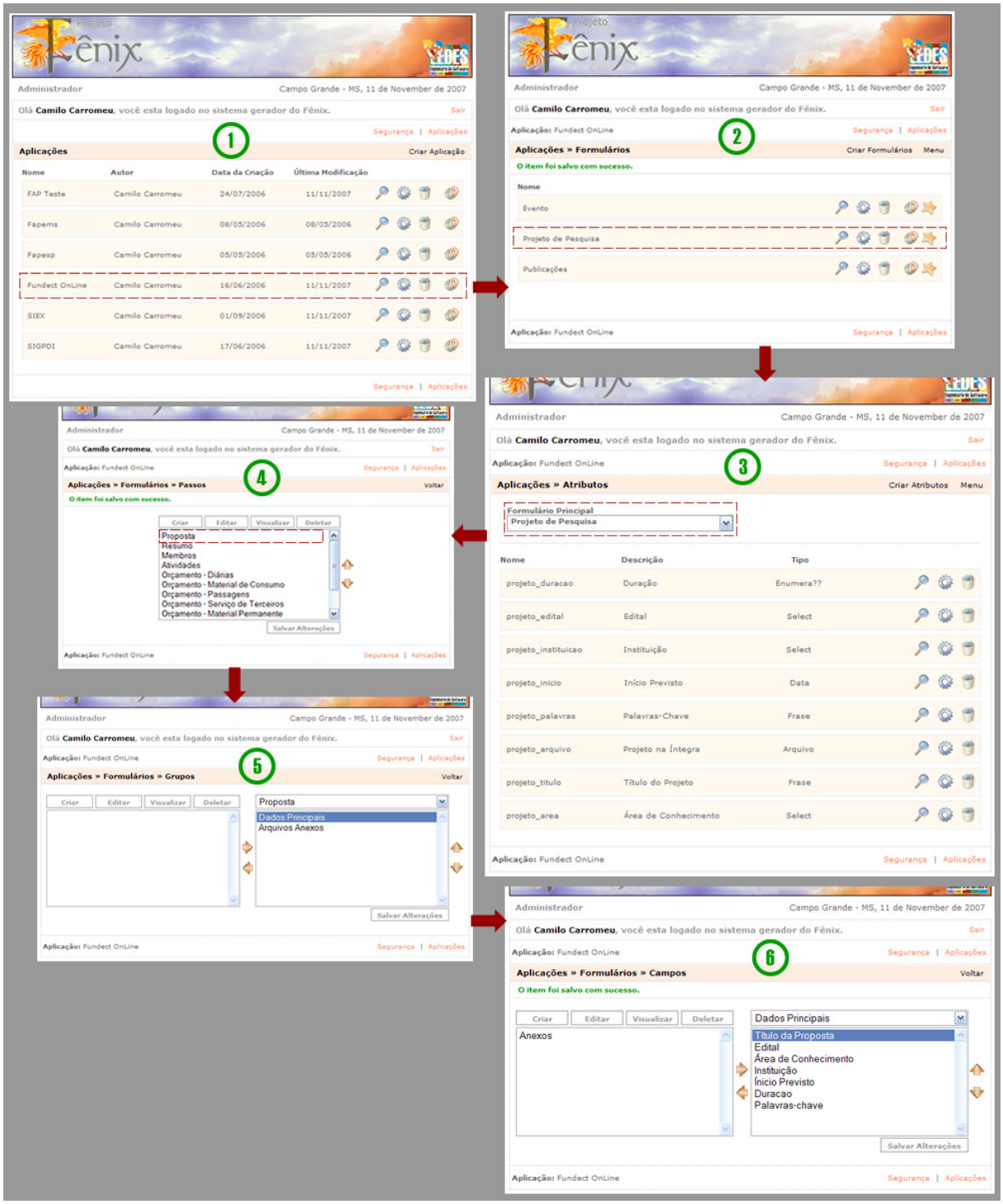


Figura 5.3: Criação de formulário eletrônico para a modalidade "Projeto de Pesquisa"e configuração do primeiro passo.

```
<go-to flag="success-step_6" action="step_6" />
<go-to flag="success-step_7" action="step_7" />
<go-to flag="success-step_8" action="step_8" />
<go-to flag="success-step_9" action="step_9" />
<go-to flag="success-step_10" action="step_10" />
<go-to flag="success-step_11" action="step_11" />
```

```
<group label="Dados Principais">
<field type="String" column="projeto_titulo" label="Título" help="Insira um título para sua proposta." max-leght="256" />
<field type="Select" column="projeto_edital" label="Edital" link-table="edital" link-column="_id"
link-view="edital_titulo" />
<field type="Select" column="projeto_area" label="Área de Conhecimento" link-table="area_de_conhecimento"
link-column="_id" link-view="area_nome" />
<field type="Select" column="projeto_instituicao" label="Instituição" link-table="instituicao" link-column="_id"
link-view="insituicao_nome" />
<field type="Date" column="projeto_inicio" label="Data de Ínicio" help="Data de ínicio da execução do projeto." />
<field type="Enum" column="projeto_duracao" label="Duração" help="Duração do período de execução.">
<enum-mapping value="6"  label="6 meses" />
<enum-mapping value="12" label="12 meses" />
<enum-mapping value="18" label="18 meses" />
<enum-mapping value="24" label="24 meses" default="true" />
<enum-mapping value="30" label="30 meses" />
<enum-mapping value="36" label="36 meses" />
</field>
<field type="String" column="projeto_palavras" label="Palavras-chave" max-lenght="512" />
</group>
<group label="Arquivos Anexos">
<field type="File" column="projeto_arquivo" label="Anexo" />
</group>
</form>
```

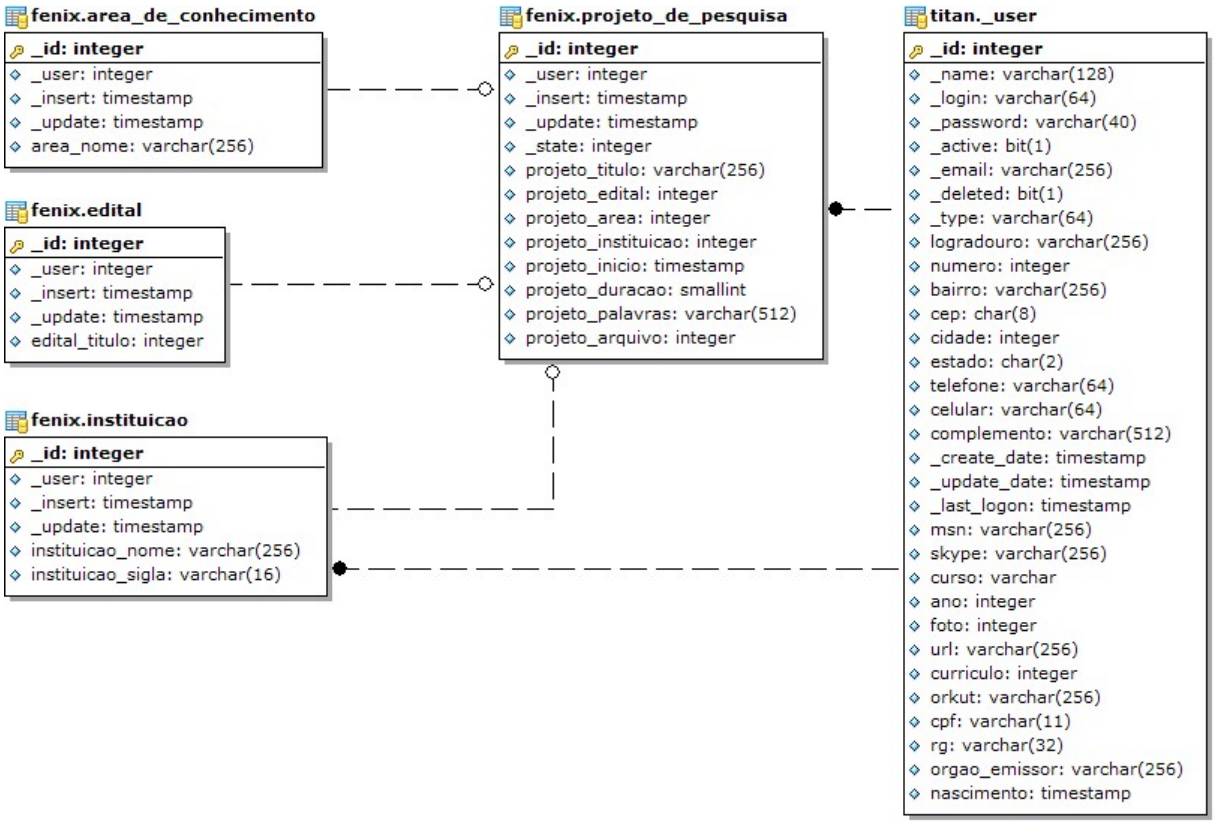


Figura 5.4: Diagrama entidade-relacionamento gerado para o formulário eletrônico da modalidade “Projeto de Pesquisa”.

No próximo passo, a ferramenta Fênix adiciona o passo configurado como uma nova ação no arquivo de configuração desta seção (config.inc.xml), conforme ilustrada abaixo:

```
<action
```

```
name="step_0"
engine="create"
xml-path="step_0.xml"
label="I-a. Informações Gerais do Projeto [1/12]"
description="">
<menu function="submenu">
<item function="save" label="&lt;b&gt;1.1&lt;/b&gt; Informações Gerais do Projeto" go-to="success-step_0" />
<item function="save" label="&lt;b&gt;1.2&lt;/b&gt; Resumo" go-to="success-step_1" />
<item function="save" label="&lt;b&gt;2.1&lt;/b&gt; Membros" go-to="success-step_2" />
<item function="save" label="&lt;b&gt;2.2&lt;/b&gt; Atividades" go-to="success-step_3" />
<item function="save" label="&lt;b&gt;3.1&lt;/b&gt; Orçamento: Diárias" go-to="success-step_4" />
<item function="save" label="&lt;b&gt;3.2&lt;/b&gt; Orçamento: Material de Consumo" go-to="success-step_5" />
<item function="save" label="&lt;b&gt;3.3&lt;/b&gt; Orçamento: Passagens" go-to="success-step_6" />
<item function="save" label="&lt;b&gt;3.4&lt;/b&gt; Orçamento: Serviço de Terceiros" go-to="success-step_7" />
<item function="save" label="&lt;b&gt;3.5&lt;/b&gt; Orçamento: Material Permanente" go-to="success-step_8" />
<item function="save" label="&lt;b&gt;4.1&lt;/b&gt; Contrapartida" go-to="success-step_9" />
<item function="save" label="&lt;b&gt;4.2&lt;/b&gt; Recursos Solicitados" go-to="success-step_10" />
<item function="save" label="&lt;b&gt;4.3&lt;/b&gt; Orçamento Consolidado" go-to="success-step_11" />
</menu>
<menu function="js" image="save.png" label="Salvar Formulário" js="saveSteps ()" />
<menu function="save" image="forward.png" label="Próximo Passo" go-to="success-next" />
<menu function="save" image="rewind.png" label="Passo Anterior" go-to="success-previous" />
<menu action="list" image="list.png" label="Listar Formulários" />
</action>
```

O núcleo do Titan processa os arquivos XML modificados, e o resultado é ilustrado na Figura 5.5. Têm-se, portanto, a funcionalidade equivalente ao passo da Fundect OnLine. No menu de passos, aberto na imagem para visualização, é possível observar os demais passos do formulário de Projetos de Pesquisa.

Após terminar a configuração do formulário, têm-se um formulário *main* com passos, grupos e campos, e uma estrutura de atributos bem definida para a modalidade de projetos de pesquisa. Entretanto, não existem ainda configurações relacionadas ao acesso deste formulário por usuários da instância e não foi definido nenhum *workflow* para o trâmite das entidades (projetos de pesquisa) geradas pelo formulário. Como próximo passo, é necessário gerar um tipo de usuário com permissão de acesso para criação de novas entidades deste formulário denominado Pesquisador e, também, gerar outro tipo de usuário com permissão para julgar as propostas submetidas pelo primeiro tipo, denominado "Gestor".

Para a criação de tipos de usuários utiliza-se a opção "Usuários"do menu de configuração de aplicações. Através desta interface do Fênix é possível criar tipos de usuários para a aplicação especificando se o cadastro do usuário é público, privado ou protegido e vinculando um formulário *main* ao tipo. Depois de criado o novo tipo de usuário, deve-se escolher um formulário, que pode ser o mesmo formulário *main* escolhido durante a criação do tipo ou uma de suas instâncias, para cadastro, outro para modificação dos dados pelo próprio usuário e outro para edição dos dados por meio de uma seção gerada na instância somente para gestão deste tipo.

Assim, antes de criar este tipo, volta-se ao gestor de formulários e cria-se um formulário *main* denominado "Pesquisador". No gestor de atributos criam-se os atributos vinculados a este formulário *main* e, novamente no gestor de formulários, pode-se gerar as instâncias do formulário "Pesquisador"para serem utilizadas no cadastro de novos pesquisadores, modificação dos dados pessoais pelos pesquisadores e gestão de pesquisadores. Para cada uma destas instâncias são vinculados passos, grupos e campos, onde, estes últimos, são

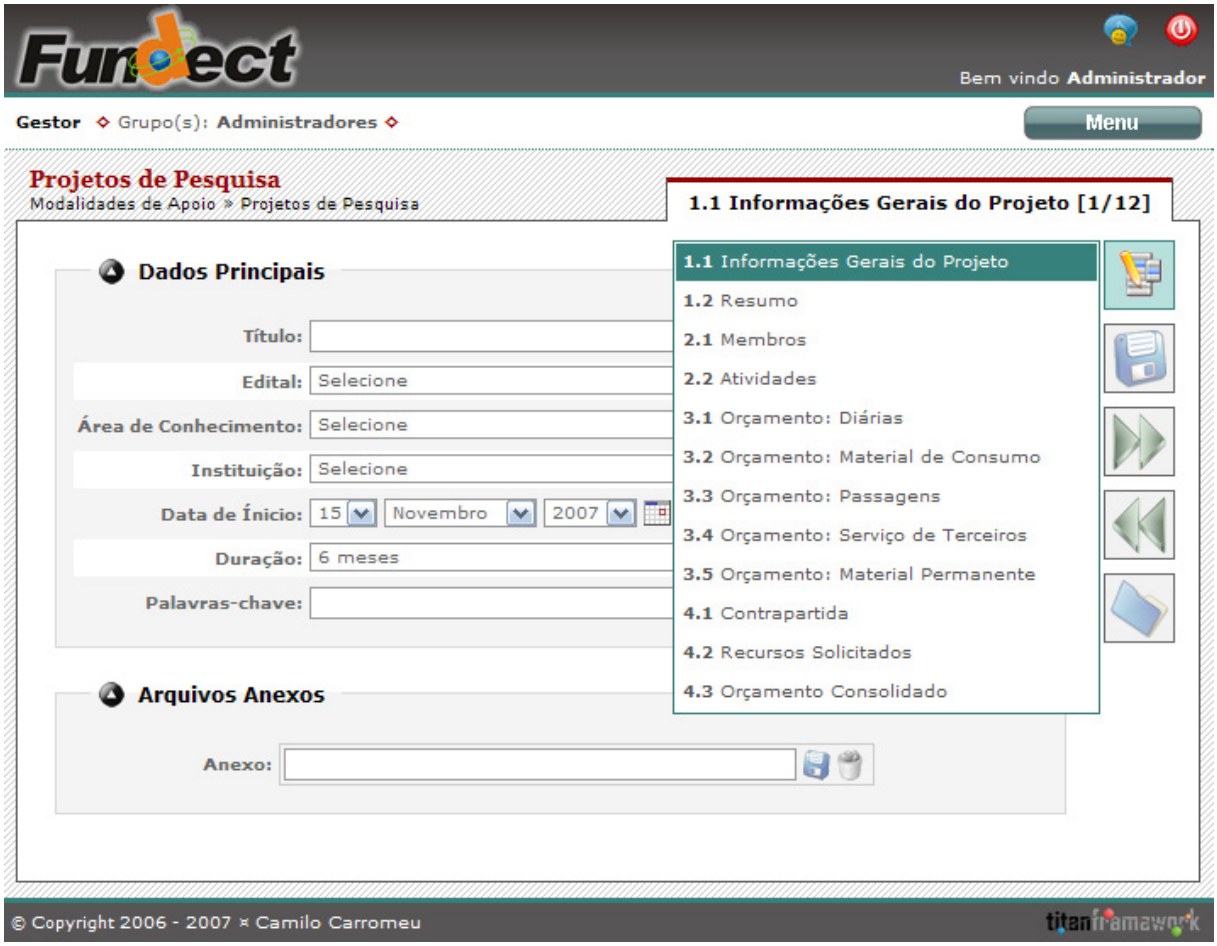


Figura 5.5: Passo 1.1 do formulário eletrônico para a modalidade "Projeto de Pesquisa" da instância gerada.

associados aos atributos criados anteriormente.

O mesmo procedimento repete-se para o tipo de usuário "Gestor", partindo da criação de um formulário *main* homônimo. Cada um destes tipos de usuários tem seu próprio conjunto de atributos e instâncias de seus formulários *main*. Não é descrito cada campo destes formulários, mas vale citar que o tipo de usuário "Gestor" é criado com o cadastro privado e o tipo "Pesquisador" com o cadastro público. Na geração de código para o *framework* são criadas duas novas seções para gestão destes usuários e as configurações são gravadas no arquivo *security.xml* da instância do Titan, responsável por identificar o tipo de usuário no momento do logon. O arquivo *security.xml* gerado para este exemplo pode ser observado abaixo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<security-mapping>
<user-type
label="Gestor"
name="gestor"
description="Usuários gestores."
type="private"
form-modify="modify.xml"
/>
```

```
<user-type
label="Pesquisador"
name="pesquisador"
description="Usuários pesquisadores."
type="public"
form-modify="modify.xml"
form-register="register.xml"
/>
</security-mapping>
```

O Titan gera formulários públicos que podem ser disponibilizados dentro de portais seguindo o conceito de *mashups* [67]. Desta forma, pode-se substituir o formulário de cadastro de pesquisadores atual do portal Fundect OnLine pelo formulário gerado pelo Titan para cadastro do usuário do tipo "Pesquisador". Na Figura 5.6 é apresentada esta substituição.

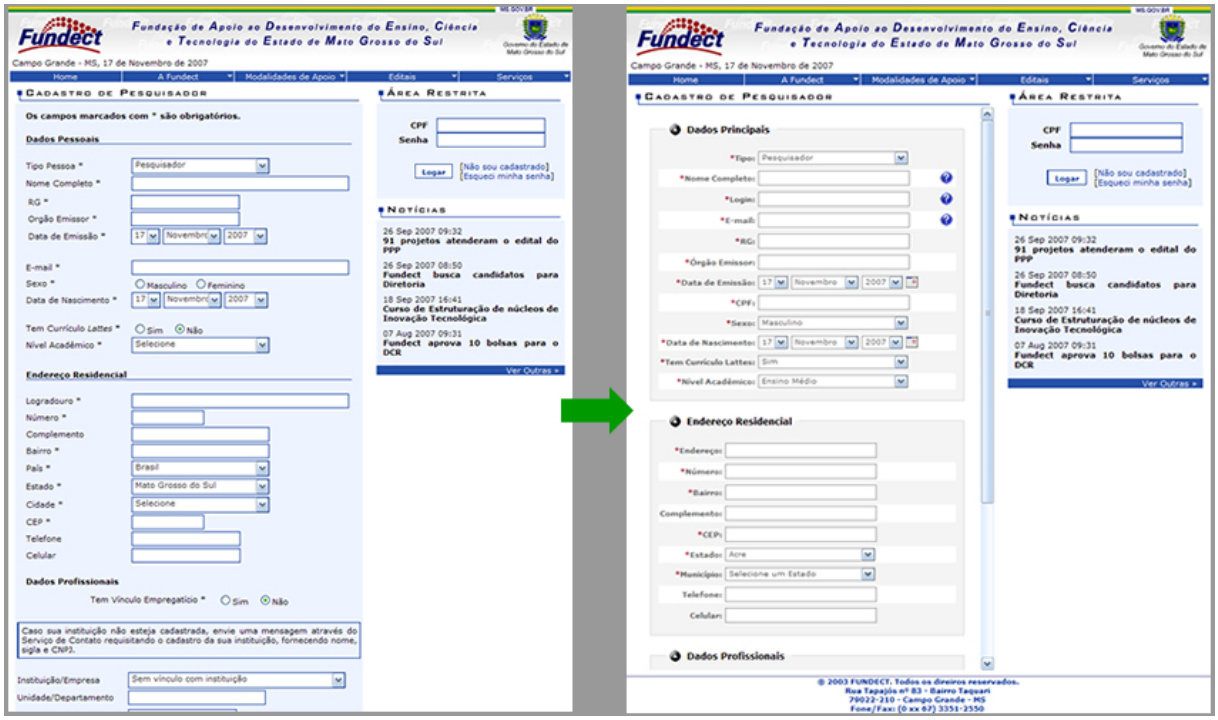


Figura 5.6: Formulário de cadastro de Pesquisador gerado pelo Titan embutido no Portal Fundect OnLine.

O próximo passo é configurar o trâmite de Projetos de Pesquisa na aplicação instanciada. Em resumo, deve ser configurado quais estados um projeto pode se tramitar, em que momento cada usuário tem acesso a este projeto e qual o nível deste acesso. A configuração deste *workflow* dá-se por meio do Gestor de Módulos do Fênix.

Para este exemplo, é criado um módulo denominado "Projeto de Pesquisa", que é definida como a modalidade de apoio da aplicação exemplo. Para configurá-la é utilizada ferramenta visual de criação de *workflow* do Gestor de Módulos do Fênix. Nela são criados três estados: um estado inicial, onde a nova entidade de projeto é criada pelo pesquisador; um segundo estado onde esta entidade pode ser visualizada pelo pesquisador e editada

pelos gestores; e, um terceiro estado que representa um projeto em execução, que pode ser apenas visualizado por pesquisadores e gestores.

Na criação de um módulo é necessário vincular um formulário "main". Na criação de visões em cada estado, este formulário, assim como suas instâncias, pode ser vinculado aos tipos de usuários a fim de suprir as regras indicadas. Desta forma, por meio da interface do Gestor de Módulos, mostrada na Figura 5.7, cria-se os três estados e, para cada um, criam-se visões que relacionam àquele estado a um grupo de usuários e a uma instância do formulário *main* "Projeto de Pesquisa". Em cada visão pode-se também declarar se o formulário é apenas para visualização (o que sobrecarrega a mesma regra individual de campos), se pode ser apagado, criado uma nova entidade e se a permissão em questão é apenas para o usuário autor da proposta. No estado 2, portanto, há duas visões: a primeira relaciona uma instância do formulário "Projeto de Pesquisa"ao grupo "Pesquisador"com permissão, apenas para o autor, de visualização da proposta; e, a segunda, relaciona outra instância deste formulário *main*, onde podem aparecer campos novos ao formulário como "Observações"ou "Mudança de Estado"(para um possível regresso do estado para revisão pelo pesquisador), ao grupo "Gestor"com direitos de edição (onde a regra de 'somente leitura' dos campos é válida).

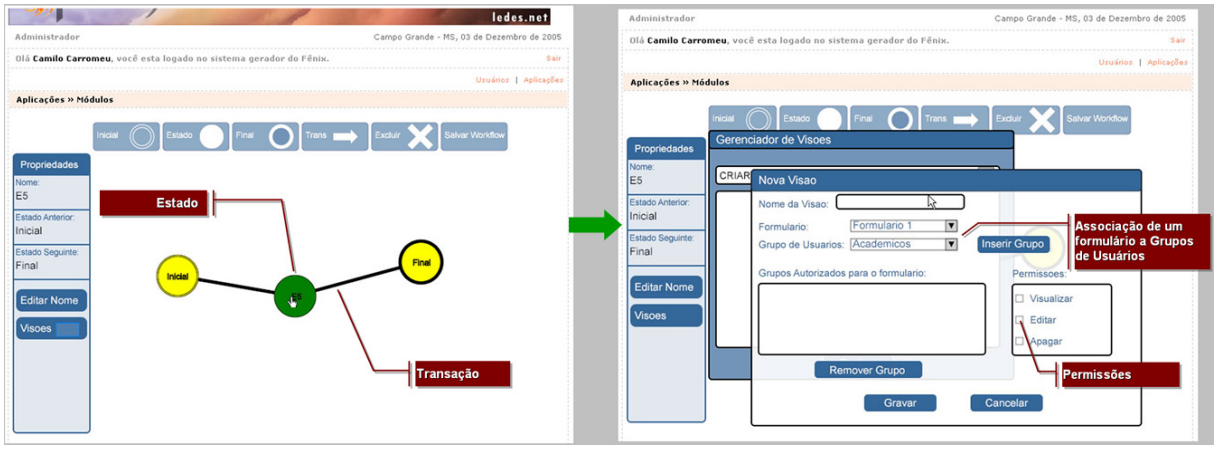


Figura 5.7: Criação do *workflow* para o trâmite de Projetos de Pesquisa.

Na geração de código a tabela do formulário *main* "Projeto de Pesquisa"recebe uma nova coluna para controle do estado de cada tupla. Além disso, as regras determinadas pelas visões são configuradas na instância no formato de permissões. Assim, o gerador de código irá criar uma ação de visualização destas tuplas pra cada estado existente, nomeando estas ações com o título do estado definido no Fênix. No exemplo em questão, o Fênix gera o seguinte arquivo *config.inc.xml* de configuração da seção:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<action-mapping>
<directive
name="NUMBER_OF_STATES"
value="3"
/>

<permission
name="ACCESS_ALL_state_0"
```

```
label="Pode listar e gerenciar todos os itens EM EDIÇÃO PELO PESQUISADOR."
/>
<permission
name="ACCESS_ALL_state_1"
label="Pode listar e gerenciar todos os itens EM JULGAMENTO."
/>
<permission
name="ACCESS_ALL_state_2"
label="Pode listar e gerenciar todos os itens EM EXECUÇÃO/ENCERRADOS."
/>

<directive
name="STATE_0_NAME"
value="state_0"
/>
<directive
name="STATE_0_LABEL"
value="EM EDIÇÃO PELO PESQUISADOR"
/>
<directive
name="STATE_1_NAME"
value="state_1"
/>
<directive
name="STATE_1_LABEL"
value="EM JULGAMENTO"
/>
<directive
name="STATE_2_NAME"
value="state_2"
/>
<directive
name="STATE_2_LABEL"
value="EM EXECUÇÃO/ENCERRADOS"
/>

<action
name="default"
label="Modalidade Projetos de Pesquisa"


<action
name="state_0"
engine="list"
label="Listar Itens EM EDIÇÃO PELO PESQUISADOR">
<menu function="search" />
<menu function="rss" />
<menu action="graph" />
</action>

<action
name="state_1"
engine="list"
label="Listar Itens EM JULGAMENTO">
<menu function="search" />
<menu function="rss" />
<menu action="graph" />
</action>

<action
name="state_2"
engine="list"
label="Listar Itens EM EXECUÇÃO/ENCERRADOS">
<menu function="search" />
<menu function="rss" />
<menu action="graph" />
</action>
...
</action-mapping>
```

O componente denominado *fenix.module* do Titan se encarrega de ler as diretivas e buscar em cada ação apenas os projetos no estado relacionado. As permissões de acesso a cada ação são vinculadas aos grupos dos usuários em decorrência das configurações das visões. Além disso, somente os grupos de usuários cujo as visões não restrigem o acesso aos projetos da autoria do próprio usuário podem receber as permissões de acesso a todas as tuplas daquele estado, definidas nas tags <*permission* />.

A partir da realização destes passos têm-se implementada a modalidade de apoio "Projetos de Pesquisa"na instância. Os passos para gerar esta modalidade podem ser repetidos para "Eventos"e "Publicações", que são outras modalidades de apoio presentes na Fundect OnLine. Pode-se ainda aprimorar o sistema definindo um novo tipo de usuário denominado "Consultor *AdHoc*"e inserindo formulários eletrônicos de parecer.

5.4 Considerações Finais

Nesta seção foi discorrido um passo-a-passo para a geração de um novo membro para a família de produtos da LPS no domínio de SAGF. Partindo de uma aplicação existente e de sua arquitetura, foi utilizada a ferramenta Fênix para automatizar o processo LPS instanciando uma nova aplicação neste domínio. Analisando os arquivos gerados e, por fim, as telas renderizadas pelo *framework* é possível o uso do ambiente de geração de aplicações.

Capítulo 6

Conclusão

6.1 Resumo do Trabalho Realizado

A motivação deste trabalho teve início com a especificação e a implementação do sistema intitulado Fundect OnLine (http://www.fundect.ms.gov.br) da agência estadual FUNDECT-MS por um grupo de pesquisadores do LEDES-DCT/UFMS. A partir da análise do domínio e das necessidades de gestão da diretoria executiva da FUNDECT observou-se que vários desafios do domínio de gestão de fomento de projetos podem ser resolvidos com a utilização das tecnologias de informação e comunicação.

Para facilitar o desenvolvimento do sistema Fundect OnLine foi realizado um diagnóstico das agências de fomento estaduais (FAPs) e dos sistemas de informação que elas utilizam. Após o levantamento observou-se que somente existem 23 FAPs no Brasil e somente sete sistemas auxiliam a gestão das diferentes modalidades de apoio. Como conseqüência, o modelo de negócio de gestão de fomento de projetos das FAPs foi reconhecido como um possível domínio para utilizar LPS. Assim, iniciou-se a investigação da presente pesquisa que defende a adoção integrada de abordagens de LPS, *frameworks* e geradores de aplicação como técnicas de reutilização para a melhoria da qualidade das aplicações.

A partir da pesquisa das várias abordagens de LPS existentes para o projeto e o desenvolvimento de software baseado em reuso no domínio de SAGF a abordagem PLUS foi escolhida para se aplicar a LPS no domínio de SAGF, permitindo que os artefatos e os componentes do domínio evoluam por meio de várias iterações do ciclo de vida do processo ESPLEP (Engenharia da LPS e Engenharia da Aplicação). Após a especificação e a implementação dos artefatos reusáveis do domínio tornou-se necessária a implementação destes para gerar uma aplicação, nascendo o Ambiente de Geração de Aplicações.

Os aspectos enfoque metodológico, modificação e alcance do domínio apresentados na seção 2.2 foram empregados na especificação da arquitetura de LPS. O modelo do processo LPS aplicado no domínio deste trabalho visa atender no enfoque metodológico um desenvolvimento com reutilização, ou seja, um processo de construção de soluções a partir de padrões já especificados. Em relação ao aspecto de modificação considerado

neste processo compreende um tipo Adaptativo na maioria dos casos e Caixa-Preta em outros, pois propõe trabalhar sempre com domínios específicos. No aspecto alcance do domínio, o tipo é horizontal, ou seja, mesmo sendo o processo para uma FPS a reutilização de artefatos pode ser realizada de outros domínios que tenham correlação com este. Estes aspectos fazem desta uma proposta de reutilização relevante para projetos que procuram um reuso de produtos em um domínio específico.

O Ambiente de Geração de Aplicações especificado e implementado é do tipo compositor e é formado por duas ferramentas: o *framework* Titan, gerenciador de conteúdo do tipo caixa-branca, que permite a especificação e a implementação dos artefatos no Repositório no domínio de SAGF e a ferramenta **Fênix** que funciona como um *Wizard* para instanciação do Titan com base nos padrões existentes, gerando WebApps neste domínio. Além do uso neste projeto, o *framework* Titan está se destacando pela sua agilidade e facilidade de uso no processo de desenvolvimento de novas WebApps. Para testar, validar e avaliar a aplicação do processo LPS no domínio de SAGF, a ferramenta Fênix e o *framework* Titan foi elaborado um estudo de caso para implementar um novo sistema para a FUNDECT. A partir do uso das técnicas propostas foi possível destacar uma melhoria na qualidade do processo de desenvolvimento e da aplicação gerada quando comparado com o atual sistema Fundect OnLine, pois as características agregadas dos padrões e da instância do *framework* facilitam a adaptação e a manutenção.

Durante o desenvolvimento desta pesquisa cinco trabalhos ([21] [22] [24] [23] [97]) foram aceitos em congressos, o que ressalta a importância do tema.

6.2 Contribuições

As principais contribuições deste trabalho são:

1. Diagnóstico dos sistemas de informação existentes nas FAPS do Brasil em âmbito estadual;
2. Revisão bibliográfica e análise das principais abordagens de LPS na literatura, tais como, PLP, Synthesis/RSP, PULSE, FODA, FOOM, FAST, KobrA e PLUS;
3. Aplicação de um processo de desenvolvimento LPS baseado na abordagem PLUS para o domínio de SAGF integrando *frameworks* e geradores de aplicação;
4. Viabilidade da integração das técnicas de reuso: LPS, *frameworks* e geradores de aplicação;
5. Especificação da FPS para o domínio de SAGF em conjunto com o processo de desenvolvimento proposto. Foram analisados dois sistemas desenvolvidos pelo LEDES a fim de abstrair as similaridades e as variabilidades para a LPS: SIEX e Fundect OnLine. Além destes, também foram analisadas as variabilidades e similaridades de outros sistemas existentes neste domínio;

6. Teste e validação do *framework* Titan, que permite instanciar aplicações do domínio de gerência de conteúdo, e criação de novos componentes com a disponibilização do código para utilização livre e criação de projetos colaborativos, no domínio de SAGF;

7. Especificação, implementação e teste da ferramenta Fênix que automatiza o desenvolvimento de produtos de software no domínio de SAGF utilizando o processo de LPS. A ferramenta utiliza o *framework* Titan para a instanciação de produtos orientados à família;

8. Implementação de um estudo de caso de um sistema de apoio à gestão de fomento (baseado no sistema existente Fundect OnLine) a fim de validar e avaliar o processo de LPS no domínio de SAGF e as ferramentas desenvolvidas. Como resultado desta experiência, pôde-se concluir que, a partir da arquitetura de LPS e dos artefatos de software gerados, o processo de instanciação de novos sistemas torna-se fácil, intuitivo e rápido; e

9. Fortalecimento da linha de pesquisa em reutilização de software no LEDES-DCT/UFMS.

6.3 Limitações da Proposta

A principal limitação é que o subsistema Gerador de Código da ferramenta Fênix não está finalizado, prejudicando a geração automática de novos membros da FPS no domínio de SAGF. Este gerador está sendo desenvolvido por um outro trabalho de mestrado. As principais dificuldades desta pesquisa foram relacionadas a compreensão das diferentes abordagens existentes de LPS, assim como dos conceitos principais de FPS e LPS. A adaptação do *framework* Titan para o domínio de SAGF exigiu a reimplementação (*refactory*) de alguns componentes a fim de alcançar um padrão de estrutura de classes e linguagem de entrada que atendesse à grande quantidade variedade de gerenciadores de conteúdo.

6.4 Trabalhos Futuros

A partir dos resultados deste trabalho pode-se vislumbrar diversas pesquisas que podem ser exploradas :

1. Investigar e propor novos aspectos da taxonomia de técnicas de reutilização para facilitar os engenheiros na decisão de escolha de uma abordagem;

2. Modelar e implementar outros sistemas de apoio à gestão de fomento a fim de avaliar se os requisitos e padrões definidos são suficientes;

3. Reimplementar o Gestor de Atributos da ferramenta Fênix de forma a criar Tipos como componentes EJB (*Entreprise Java Beans*). Atualmente o suporte de novos tipos de atributos na ferramenta Fênix exige reimplementação. Essa limitação deu

origem a idéia de criar, como trabalho futuro, uma nova arquitetura do gestor de atributos baseada em engenharia de componentes. Indubitavelmente, isso aumenta a flexibilidade quanto as necessidades de diferentes tipos de campos de formulários que uma aplicação instanciada poderia utilizar; e

4. Estender a arquitetura da LPS para possibilitar a interoperabilidade com sistemas externos, por exemplo, currículo Lattes do CNPq e a Plataforma de Gestão de Talentos do Governo de Mato Grosso do Sul, a fim de fornecer informações para auxiliar na tomada de decisões dos gestores.

6.5 Trabalhos decorrentes desta pesquisa

Além dos resultados e contribuições apresentadas, encontra-se em fase final um trabalho de mestrado [96] que está implementando o Gerador de Código da ferramenta Fênix. A experiência adquirida neste projeto contribui para a consolidação do SIEX como sistema único adotado pela SESu/MEC para a gestão de ações de extensão nas universidades públicas brasileiras. O *framework* Titan, validado no domínio desta pesquisa, está sendo utilizado na implementação de novos sistemas Web e projetos de pesquisa, tais como: Projeto WebPIDE - Plataforma Aberta de Integração e Avaliação de Dados Educacionais - CAPES/INEP (http://webpide.ledes.net); CISM - Central de Indicadores e Serviços Municipais - FUNDECT (http://cism.ledes.net); portal do PAIR - Programa de Ações Integradas e Referênciais de Enfrentamento à Violência Sexual Infanto-Juvenil no Território Brasileiro - SEDH da Presidência da República (http://pair.ledes.net); e-SAPI / Portal da Carne - CNPq, Embrapa e MAPA (http://e-sapi.ledes.net); portal de Boas Práticas Agropecuárias - CNPq, Embrapa e MAPA (http://bpa.ledes.net); portal do LEDES (http://www.ledes.net); e, portais de departamentos e Pró-Reitorias da UFMS, por exemplo, do Departamento de Computação e Estatística (http://www.dct.ufms.br).

Referências Bibliográficas

[1] S. A. AJILA e P. J. TIERNEY. The foom method - modeling software product lines in industrial settings. *Proceedings of the 2002 International Conference on Software Engineering Research and Practice (SERP02), Las Vegas, Nevada, USA: CSREA Press*, 2002.

[2] C. ALEXANDER e ET AL. *A pattern language*. New York: Oxford University Press, 1977.

[3] C. R. ARAGÓN. *Processo de Desenvolvimento de uma Linha de Produtos para Sistemas de Gestão de Bibliotecas*. Tese de Mestrado, Universidade Federal do Mato Grosso do Sul, Campo Grande, MS, Brasil, 2004.

[4] C. ATKINSON, J. BAYER, C. BUNSE, E. KAMSTIES, O. LAITENBERG, R. LAQUA, D. MUTHIG, B. PAECH, J. WUST, e J. ZETTEL. *Component-based Product Line Engineering with UML*. Addison-Wesley, 2001.

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

[6] L. BASS, P. CLEMENTS, S. COHEN, L. NORTHROP, e J. WITHEY. Product line practice workshop report. *Pittsburgh, Pennsylvania, USA*, 1997.

[7] D. BATORY e Y. SMARAGDAKIS. Application generators, 2003. Disponível em: http://www.cc.gatech.edu/ yannis [06/09/2006].

[8] C. BAUER e G. KING. *Hibernate in Action*. Manning Publications, 2005.

[9] 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.

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

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

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

[13] J. BOSCH, P. O. BENGTSSON, P. MOLIN, e M. MATTSSON. *Object oriented framework - problems & experiences*. Wiley & Sons, 1999.

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

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

[16] T. BRAY, J PAOLI, e C. M. SPERBERG-MCQUEEN. Extensible markup language (xml) 1.0. *W3C Recommendation*, 2000.

[17] S. N. BRISOLLA. Indicadores para apoio à tomada de decisão. Relatório Técnico v. 27, n. 2, p. 221-225, Ciência da Informação, Brasília, 1998.

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

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

[20] G. J. O. CAMPBELL, N. BURKHARD, e J. FACEMIRE. *Synthesis Guidebook*. Hemdon, Virginia, USA, 1991.

[21] C. CARROMEU e M. A. S. TURINE. Um framework de aplicações web de apoio a gestão de fomento. *Workshop de Teses e Dissertações do XI Simpósio Brasileiro de Sistemas Multimídia e Web - WebMedia 2005. Poços de Caldas, Minas Gerais, Brasil*, 2005.

[22] C. CARROMEU e M. A. S. TURINE. Fênix: Uma ferramenta para automatizar o processo de linha de produtos de software no domínio de gestão de fomento. *XXXIII Seminário Integrado de Software e Hardware - SEMISH 2006. Evento integrante do XXVI Congresso Brasileiro da Sociedade Brasileira de Computação. Campo Grande, MS, Brasil*, 2006.

[23] C. CARROMEU e M. A. S. TURINE. Uma ferramenta de desenvolvimento Ágil de sistemas web para a gestão de fomento de projetos para faps. *Workshop de Teses e Dissertações do XII Simpósio Brasileiro de Sistemas Multimídia e Web - WebMedia 2006. Natal, Rio Grande do Norte, Brasil*, 2006.

[24] C. CARROMEU, M. A. S. TURINE, T. BITENCOURT, N. OSHIRO, e J. E. SAVICKI. Processo de uma linha de produtos para sistemas web de apoio a gestão de fomentos. *XXXII Conferencia Latinoamericana de Informática - CLEI 2006. Santiago do Chile, Chile*, 2006.

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

[26] J. CHEESMAN e J. DANIELS. *Project Management: Strategic Design and Implementation*. Mc Graw-Hill Inc., 3th, 1999.

[27] J. CHEESMAN e J. DANIELS. *UML Components - A Simple Process for Specifying Component-based*. Addison-Wesley, 2001.

[28] J. C. CLEAVELAND. *Program Generator with XML and Java*. Prentice Hall, 2001.

[29] P. C. CLEMENTS e L. NORTHROP. *Software Product Lines: Practice and Patterns*. Addison-Wesley, 2001.

[30] S. G. COHEN, J. A. HESS, K. C. KANG, W. E. NOVAK, e A. S. PETERON. Feature-oriented domain analysis (foda) feasibility study. *Pittsburgh, Pennsylvania, USA*, 1990.

[31] T. CONVERSE e J. PARK. *PHP: a Bíblia*. Editora Campos, 2003.

[32] J. O. COPLIEN. Software design patterns: Common questions and answers. *The Patterns Handbook: Techniques, Strategies, and Applications. New York, NY, USA: Cambridge University Press*, 1998.

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

[34] K. CZARNECKI e U. W. EISENERCKER. *Generative Programming*. Addison-Wesley, 2002.

[35] D. F. D'SOUZA e A. W. WILLS. *Objects, components and frameworks with UML - the catalysis approach*. Addison-Wesley, 1999.

[36] W. R. DUNCAN. A guide to the project management body of knowledge. *Project Management Institute, Upper Darby, PA*, 1996.

[37] B. ECKEL. *Thinking in Java*. Prentice Hall, 2003.

[38] J. D. EISENBERG. *SVG Essentials*. O'Reilly, 2002.

[39] FAPEAM. Gipa - gerenciamento informatizado de programas e ações da fapeam, 2006. Disponível em: http://gipa.inpa.gov.br/ [06/09/2006].

[40] FAPESP. Sage - sistema de apoio a gestão, 2007. Disponível em: http://www.fapesp.br/sage/ [17/03/2007].

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

[42] M. E. FAYAD, R. E. JOHNSON, e D. C. SCHMIDT. *Building application frameworks: Object-Oriented foundations of framework design*. Wiley & Sons, 1999.

[43] M. E. FAYAD, D. C. SCHIMIDT, e R. E. JOHNSON. *Domain-Specific Application Frameworks*. Wiley Computer Publishing, 1999.

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

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

[46] W. FRAKES e C. TERRY. Software reuse: metrics and models. *ACM Computing Surveys (CSUR), ACM Press, v. 28, n. 2, p. 415-435*, 1996.

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

[48] J. FRYE e J. YODER. The hillside group, 2001. Disponível em: http://hillside.net [06/09/2006].

[49] FUNDECT. Fundação de apoio ao desenvolvimento do ensino, ciência e tecnologia do estado de mato grosso do sul, 2006. Disponível em: http://www.fundect.ms.gov.br/ [06/09/2006].

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

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

[52] J. J. GARRETT. Ajax: A new approach to web applications, 2005. Disponível em: http://www.adaptivepath.com/ideas/essays/archives/000385.php [16/11/2007].

[53] N. S. GILL e P. S. GROVER. Component-based measurement: few useful guidelines. *ACM SIGSOFT Software Engineering Notes, ACM Press, New York, NY, USA, v. 28, n. 6, p. 4-4*, 2003.

[54] I. M. S. GIMENES, F. R. LAZILHA, e R. T. PRICE. Uma proposta de arquitetura de linha de produtos para workflow management systems. *XVI Simpósio Brasileiro de Engenharia de Software. Gramado, RS, Brasil*, 2002.

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

[56] H. GOMAA. *Designing Software Product Lines with UML*. Addison-Wesley, 2005.

[57] R. GUIMARÃES. Avaliação e fomento de c&t no brasil: Propostas para os anos 90. *MCT/CNPq*, 1994.

[58] T. HUSTED, C. DUMOULIN, G. FRANCISCUS, e D. WINTERFELDT. *Struts in Action - Building web applications with the leading Java framework*. Manning Publications, 2003.

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

[60] R. E. JOHNSON. Documenting frameworks using patterns. *OOPSLA'92: Conference proceedings on Object-oriented programming systems, languages, and applications. Vancouver, British Columbia, Canadá: ACM Press*, 1992.

[61] R. E. JOHNSON. How to design frameworks. *Tutorial at OOPSLA'93*, 1993.

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

[63] H. KILOV. *Business Specifications: The Key to Successful Software Engineering*. Prentice Hall, 1998.

[64] P. KLINT e A. VAN DEURSEN. Domain-specific language design requires feature descriptions. *A. van Deursen and P. Klint. Domain-specific language design requires feature descriptions. Journal of Computing and Information Technology*, 2002.

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

[66] M. MATINLASSI. Comparison of software product line architecture design methods: Copa, fast, form, kobra and qada. *Proceedings of the 26th International Conference on Software Engineering. Washington, DC, USA: IEEE Computer Society*, 2004.

[67] D. MERRILL. Mashups: The new breed of webapp, 2006. Disponível em: http://www.ibm.com/developerworks/xml/library/x-mashups.html [17/11/2007].

[68] G. MESZAROS e J. DOBLE. *A pattern language for pattern writing*. Addison-Wesley, 1998.

[69] T. C. DE OLIVEIRA. *Uma Abordagem Sistemática para a Instanciação de Frameworks Orientados a Objetos*. Tese de Doutoramento, Pontifícia Universidade Católica do Rio de Janeiro, PUC-RJ, Rio de Janeiro, RJ, Brasil, 2001.

[70] T. O'REILLY. What is web 2.0: Design patterns and business models for the next generation of software, 2005. O'Reilly Network, disponível em: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html [05/11/2007].

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

[72] J. C. L. PEREIRA e M. P. BAX. Introdução à gestão de conteúdo. *Anais do III Workshop Brasileiro de Inteligência Competitiva e Gestão do Conhecimento, São Paulo*, 2002.

[73] PREAE. Sistema de informação de extensão da universidade federal de mato grosso do sul, 2006. Disponível em: http://preae.ufms.br/siex/ [06/09/2006].

[74] W. PREE. *Design patterns for object oriented software development*. Addison-Wesley, 1994.

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

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

[77] K. REED. Software engineering: a new millenium? *IEEE Software*, 2000.

[78] W. ROMÃO, C.A.P. NIEDERAUER, A. MARTINS, A. TCHOLAKIAN, R. C. S. PACHECO, e R.M. BARCIA. Extração de regras de associação em c&t: O algoritmo apriori. *XIX Encontro Nacional em Engenharia de Produção*, 1999.

[79] D. ROSS. Structured analysis: A language for communicating ideas. *IEEE Transmition Software Engineering, v. 3, n. 1*, 1977.

[80] SBC. Jems - journal and event management system, 2006. Disponível em: https://submissoes.sbc.org.br/ [06/09/2006].

[81] SEI. Software engineering institute, 2006. Disponível em: http://www.sei.cmu.edu/ [06/09/2006].

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

[83] R. P. SILVA. *Suporte ao desenvolvimento e uso de frameworks e componentes*. Tese de Doutoramento, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brasil, 2000.

[84] S. A. DA SILVEIRA e J. CASSINO. *Software Livre e Inclusão Digital*. Conrad Livros, 2003.

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

[86] I. SOMMERVILLE. *Software Engineering*. Addison-Wesley, 2001.

[87] E. SPINAK. Indicadores cienciométricos. *Ciência da Informação, Brasília, v. 27, n. 2, p. 121-148*, 1998.

[88] B. STEARNS, G. MURRAY, I. SINGH, J. INSCORE, L. DEMICHIEL, M. JOHNSON, N. KASSEM, R. SHARMA, R. ORTIGAS, R. MONZILLO, S. BRYDON, e V. RAMACHANDRAN. *Designing Enterprise Applications with the J2EE Platform*. 2th, Addison-Wesley, 2002.

[89] B. STINSON. *PostgreSQL Essential Reference*. Sams Publishing, 2001.

[90] SUN. Enterprise javabeans technology, 2006. Disponível em: http://java.sun.com/products/ejb/ [06/09/2006].

[91] T. SZMRECSÁNYI. Avaliação em ciência e tecnologia: Necessidade, critérios e procedimentos. *Revista de Administração*, 1987.

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

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

[94] J. VILJAMAA. Reverse engineering framework reuse interfaces. *ACM Press, Helsinki, Finland, p. 217-226*, 2003.

[95] W3C. Sha-1 - secure hash algorithm, 1998. Disponível em: http://www.w3.org/TR/1998/REC-DSig-label/SHA1-1_0 [06/09/2006].

[96] E. O. WESCHTER e M. A. S. TURINE. Uma proposta de arquitetura para o gerador de aplicação web no domínio de gestão de fomento de projetos. *Qualificação de Mestrado. Departamento de Computação e Estatística, UFMS. Campo Grande, Mato Grosso do Sul, Brasil*, 2006.

[97] E. O. WESCHTER, M. A. S. TURINE, e C. CARROMEU. Arquitetura de um gerador de aplicações web para agências de fomento no brasil. *Conferência Internacional em Educação em Engenharia e Computação - ICECE 2007. Santos, São Paulo, Brasil. p. 62*, 2007.

[98] A. YASSIN e M. E. FAYAD. *Application Frameworks: A Survey*. John Wiley & Sons, 1999.