IMPLEMENTAÇÃO DE UMA ARQUITETURA PARA O DESENVOLVIMENTO DE SISTEMAS CORPORATIVOS WEB BASEADA NO PARADIGMA DE LINHAS DE PRODUTO DE SOFTWARE

Jorge Luís Fernandes da Costa Duarte – Graduando
Departamento Acadêmico de Tecnologia da Informação – CEFET-RN
Av. Salgado Filho, 1159 Morro Branco CEP 59.000-000 Natal-RN
E-mail: jlfcosta@yahoo.com.br

Alessandro José de Souza - M.Sc.
Departamento Acadêmico de Tecnologia da Informação e Indústria– CEFET-RN/UNED-Zona Norte
Av. Salgado Filho, 1159 Morro Branco CEP 59.000-000 Natal-RN
E-mail: ajdsouza@cefetrn.br

Leonardo Ataíde Minora - M.Sc.
Departamento Acadêmico de Tecnologia da Informação – CEFET-RN
Av. Salgado Filho, 1159 Morro Branco CEP 59.000-000 Natal-RN
E-mail: minora@cefetrn.br

RESUMO

Atualmente o desenvolvimento de Software encontra-se em um contexto onde existe uma grande demanda de produção. Devido a isso estão sendo realizados incentivos a iniciativas de formas de produção de Software para que requisitos de qualidade e produtividade sejam economicamente viáveis. Iniciativas de utilização de componentes em conjunto com o desenvolvimento de Software baseado no paradigma orientação a objetos fizeram com que o reuso se tornasse uma prática fundamental contribuindo assim para atendimento destes requisitos. Com o desenvolvimento dessas iniciativas surge uma nova abordagem no desenvolvimento Software, a linha de produto de Software. Esta nova abordagem tem como conceito a observação de características comum entre Softwares de uma mesma família, fazendo com que esses pontos se tornem mecanismos de reutilização de código através da utilização de componentes.

Dentro deste conceito de desenvolvimento de linha de produto, a arquitetura de Software é considerada o artefato de maior importância, sendo a base da linha de produtos em conjunto com a observação de pontos de variação de cada Software.

Observando este conceito nosso trabalho propõe o desenvolvimento de uma arquitetura de linha de produto de Software que tem como escopo as aplicações corporativas para a *Web* utilizando a linguagem Java.

A arquitetura foi desenvolvida utilizando padrões de projetos e *frameworks* dispostos de maneira a reduzir o impacto da utilização das tecnologias, contribuindo não somente com o fator reutilização, mas fazendo com que o desenvolvedor mantenha seu foco na codificação do Software e por conseqüência contribuindo também para o aumento da produção.

PALAVRAS-CHAVE: Arquitetura de Software; Linha de Produto de Software; Reuso de Software, Padrões de Projeto.

1. INTRODUÇÃO

Na década de 80, com o desenvolvimento de Software baseado no paradigma da orientação a objetos despertou-se nos desenvolvedores de Software as vantagens do reuso de componentes. Hoje o Software se encontra em uma situação ubíqua, e o mercado cada vez mais demanda por sua produção. Como forma de atender essa demanda e como resultado do desenvolvimento de Softwares baseado em componentes, surge um novo modelo no desenvolvimento de Software. Este modelo consiste na substituição da abordagem do desenvolvimento "projeto a projeto" por esforços para a criação e manutenção de uma linha base de produção, onde são consideradas características em comum em um determinando domínio. A este novo modelo de desenvolvimento dar-se o nome de linha de produto de Software.

Dentro deste modelo um componente de fundamental importância para o sucesso da adoção dessa abordagem é a arquitetura da linha de produção. O presente trabalho tem como objetivo apresentar uma arquitetura base para a utilização no desenvolvimento de sistemas *Web* corporativos para ser utilizada na abordagem da linha de produtos de Software, e está assim constituído: na segunda e terceira seção conceitos sobre Linhas de Produto de Software e Padrões de Projeto. Na quarta seção é apresentada a arquitetura desenvolvida e na quinta e ultima é apresentada a conclusão.

2. LINHAS DE PRODUTO DE SOFTWARE

Há algum tempo pesquisadores e tecnólogos vêem amadurecendo a idéia de que para o desenvolvimento de um produto de software de alta qualidade e economicamente viável é necessário um conjunto sistematizado de processos, técnicas e ferramentas. Neste contexto o conceito de reuso está entre as técnicas mais relevantes deste conjunto. Uma vez que a arquitetura pode ser definida como um conjunto de componentes computacionais (ou subsistemas) e o relacionamento entre eles, o reuso é considerado uma característica muito forte no que diz respeito a tempo e custo de desenvolvimento do sistema. Se a arquitetura estiver bem organizada e estruturada, cada componente computacional ou parte dele pode ser construído com vistas para o reuso. O modelo de desenvolvimento baseado nas linhas de produto de Software observa as características em comum sobre um conjunto de Softwares pertencentes a um mesmo domínio. Essas características em comum são identificadas pelo modelo de linhas de produto de Software como pontos de reutilização de Software.

Entende-se por família de produtos de Software como sendo um conjunto de sistemas que usam Software intensivamente, compartilhando em comum um conjunto de características que satisfazem a uma específica necessidade de mercado, segmento ou missão, sendo desenvolvidos sobre um conjunto de artefatos utilizando uma estratégia de reuso (Clements, 2002). Ou seja, aplicações similares que pertencem a um mesmo domínio, compartilhando de características comuns que podem ser desenvolvidas sobre uma arquitetura genérica e um conjunto de componentes que povoam essa arquitetura (Itana, 2002).

Um conceito importante na linha de produtos de Software é o da variabilidade sendo representada pelas variações ou melhor, as diferenças identificadas entre os produtos de uma mesma família. Essa identificação pode ocorrer em qualquer fase do desenvolvimento da linha de produto. Dentre as formas existentes de representação das variabilidades, podemos citar a representação através das *features*. A *feature* é uma característica importante de um produto considerada pelos seus usuários e clientes, descrevendo e distinguindo os membros de uma família de produtos (Griss, 2000). O modelo de *features* consiste basicamente na preparação de uma base para a reutilização a partir de uma organização dos aspectos comuns e variáveis, observando apenas as características importantes ao domínio.

Outra forma de obtenção das *features* é discutida por Jacobson (Jacobson, 1997), onde ele as define como um caso de uso ou uma extensão dele.

2.1 - Elementos de uma linha de produto de Software

Algumas atividades são definidas para o desenvolvimento de uma linha de produto de Software consistindo em um ciclo de vida que é dividido em duas práticas mais o gerenciamento da linha de produto. Na primeira fase do ciclo ocorre o desenvolvimento do núcleo de artefatos, sendo reconhecida como Engenharia de Domínio. Essa fase é constituída da análise de domínio, desenvolvimento da arquitetura e desenvolvimento dos componentes reutilizáveis. A segunda fase do ciclo de vida é chamada de desenvolvimento do produto. Também chamada de Engenharia da Aplicação e consiste na instanciação dos artefatos aplicados ao desenvolvimento do produto (Clements, 2002).

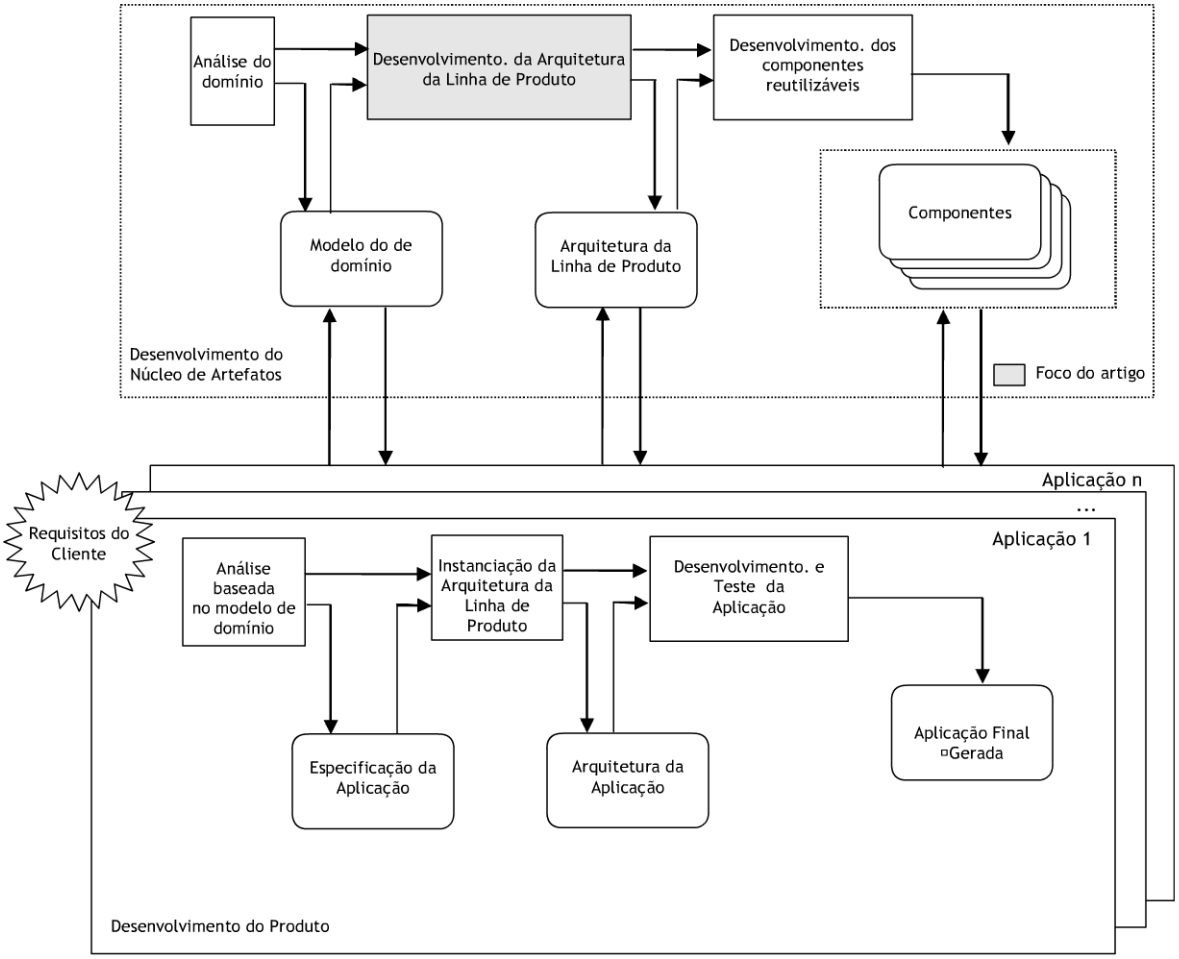

Figura 1: Ciclo de vida de desenvolvimento de softwares por linha de produto (Clements, 2002)

2.1.1 - Desenvolvimento do Núcleo de Artefatos

Antes de iniciarmos a contextualização do Desenvolvimento do Núcleo de Artefatos é importante conceituarmos artefato. Artefato é um componente reutilizável gerado para compor o núcleo de elementos no conceito de uma linha de produto de Software (Larman, 2000).

Arquitetura de uma linha de produto de Software

Dentre todos os artefatos existentes em uma linha de produto de Software a arquitetura é o mais importante deles. Deverá ser estabelecida sobre os aspectos invariáveis da linha de produto onde o seu sucesso dependerá da determinação do seu escopo, do que irá variar e o que será fixo. Durante esta fase a arquitetura se caracteriza por uma base ainda não aplicada, ou seja, não instanciada para participar da composição do desenvolvimento de um produto em específico.

A arquitetura de um sistema engloba a definição de suas estruturas gerais, descrevendo os elementos que compõem os sistemas e as interações entre estes (Bass, 2002). A arquitetura também é responsável por questões como: a organização do sistema, composição de componentes, as estruturas de controle globais, os protocolos de comunicação, a composição dos elementos do projeto e a designação da funcionalidade dos componentes do projeto. Alguns requisitos importantes do projeto são (Clements, 2002):

- Ser estável, persistindo por toda a vida da linha de produto;

- Ser extensível, permitindo a agregação de novas funcionalidades, permitindo também as variações impostas pelo mercado;
- Permitir a representação explícita de variações de modo a maximizar a reutilização;
- Fornecer mecanismos para implementar variações;
- Permitir as variações inerentes da família de aplicação.

Componentes da linha de produto

Um componente é uma unidade de Software independente que encapsula dentro de si seu projeto e implementação oferecendo interfaces bem definidas determinando pontos de interconexão expondo as funcionalidades (interfaces fornecidas) ou requisitando funcionalidades (interfaces requisitadas). Para utilizarmos essas funcionalidades é importante que sua especificação esteja clara, assim como seus argumentos de pré e pós condições (Itana, 2002).

2.2.2. - Desenvolvimento do Produto

Constitui a segunda fase do ciclo de vida do desenvolvimento de uma Linha de Produto de Software. Nesta fase ocorrerá a instanciação dos artefatos para o desenvolvimento de um produto em específico.

Durante o desenvolvimento do produto será utilizado um documento que descreverá as necessidades do cliente em conjunto com as necessidades a nível de arquitetura e componentes, o modelo de domínio. Outro artefato existente em conjunto ao modelo de domínio é o modelo de decisão que define os principais aspectos a serem considerados durante o desenvolvimento do produto, identificando ajustes necessários aos componentes. Durante a análise do modelo de domínio poderão surgir novos requisitos ainda não cobertos que podem implicar na aquisição de novos componentes para que seja realizada a extensão da arquitetura de linha de produto (Itana, 2002).

Outro artefato do desenvolvimento do produto é a arquitetura que a partir de sua versão inicial com pontos de variações e questões não resolvidas é especializada para o produto transformando-se na arquitetura específica do produto. A sua construção se fará de forma iterativa (Itana, 2002).

Como última prática temos o povoamento da arquitetura com os componentes, dando se atenção aos requisitos específicos da aplicação. O povoamento da arquitetura ocorrerá através da aquisição de novos componentes, através do repositório da Linha de Produtos, adquiridos de terceiros, dentre outros (Itana, 2002).

A fase de desenvolvimento do produto se caracteriza pela obtenção do produto a partir da instanciação dos artefatos em observação a um domínio em específico, havendo a especialização da arquitetura com a população pelos componentes. Como podemos concluir é nessa fase onde será construído um membro da linha de produtos.

2.2.3 - Gerência do Produto

Esta atividade está presente durante toda a vida da linha de produto. Clements (2002) diz que o sucesso para uma linha de produto depende do compromisso da organização em construí-la e mantê-la. A organização deve estabelecer um plano de adoção do desenvolvimento baseado em linha de produto descrevendo os objetivos desejados e estratégias para atingí-los.

A gerência do produto deverá garantir que as práticas para a linha de produto de Software sejam coordenadas e supervisionadas, devendo haver uma interação síncrona entre a equipe que desenvolve os artefatos e a que desenvolve os produtos (Itana, 2002).

3. PADRÕES DE PROJETO

Em 1990 a Engenharia de Software em conjunto com a programação orientada a objetos passa a utilizar padrões de projeto com o objetivo de encontrar soluções para problemas recorrentes em uma determinada situação. O padrão de projeto consiste não somente da solução, mas de toda a discussão sobre o contexto que levou a utilização daquela alternativa. Tendo como objetivo capturar questões centrais de um problema e oferecer uma solução que funcione em termos práticos bem como teóricos, os padrões são sugeridos como solução para problemas focados em três elementos: problema, solução e contexto.

No estudo de caso foram utilizados padrões do catálogo do GoF[1] e Java EE. Os padrões de projeto do GoF são classificados em três categorias: criacionais, estruturais e comportamental (Tabela I). Os criacionais tratam de como os objetos são criados, geralmente ocultando os detalhes da classe concreta através da utilização de uma classe abstrata ou interface. Os padrões estruturais lidam com a composição das classes ou objetos como elas herdam uma das outras e como elas são compostas a partir de outras classes. Os comportamentais tratam sobre a responsabilidade e interação dos objetos, tornando um comportamento complexo gerenciável especificando as responsabilidades dos objetos e as maneiras como eles se comunicam entre si.

Tabela I: Padrões Clássicos

Padrões de Projeto do GoF		
Criacional	Estrutural	Comportamental
Abstract Factory Builder Factory Method Prototype Singleton	Adapter Bridge Composite Decorator Facade Proxy	Chain of responsability Command Flyweight Interpreter Iterator Mediator Memento Observer State Strategy Template Method Visitor

Os padrões de projeto pertencentes ao catálogo Java EE são padrões que descrevem as melhores práticas relacionadas à arquitetura e *design* de aplicações Java EE. Consistem em um catálogo originado das experiências entre os arquitetos da *Sun Microsystems* e seus clientes. A partir dessas experiências foram surgindo documentações que passaram a registrar soluções para situações recorrentes. Cada padrão é classificado dentro de um contexto que trata do seu papel com relação à camada a que pertence na aplicação (Tabela II).

Tabela II: Padrões de Projeto Java EE

Padrões de Projeto Java EE	
Camada	**Nome do Padrão**
Apresentação	Intercepting Filter Front Controller Context Object Application Controller View Helper Composite View Service to Worker Dispatcher View
Negócio	Bussiness Delegate Service Locator Session Façade Application Service Business Object Composite Entity Transfer Object Transfer Object Assembler Value List Handler

[1] GoF – Gang of Four: Acrônimo que se refere aos criadores do catálogo de padrões de projeto (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides).

Integração	Data Access Object Service Activator Domain Store Web Service Broker

Para a implementação do estudo de caso deste trabalho os padrões utilizados foram,: *Singleton*, *Factory Method*, *Command (EJB-Command)* e o *DAO – Data Access Object*.

Singleton - Padrão que tem como função garantir uma única instância de uma classe, provendo um ponto global de acesso a ela. O *Singleton* permite um controle sobre a instância do objeto, isso ocorre devido ao encapsulamento dessa instância, permitindo também um controle sobre de que forma e quem irá acessar essa instancia (Gamma, 1995)..

Factory Method - Define uma interface para a criação de um objeto, deixando a decisão de instanciação para as subclasses. Elimina a necessidade do conhecimento dos detalhes das implementações das classes específicas as aplicações, lidando apenas com a inteface ou classes abstratas fazendo com que os detalhes relacionados a implementação de uma determinada classe sejam abstraídos (Gamma, 1995)..

Command - Tem como característica o encapsulamento de uma requisição em um objeto permitindo que os clientes parametrizem diversas requisições. O *Command* tem como característica instanciar e executar processadores através de comandos encapsulados em beans resultantes de uma requisição. O *Command* se destaca pela sua parametrização dos objetos, assim como gera ao desenvolvimento uma modularização estruturando assim as operações (Gamma, 1995).

Data Access Object - DAO - Pertencente ao catálogo de padrões para aplicações corporativas desenvolvidas em Java (Java EE), tendo como objetivo desacoplar o acesso da camada de negócios ao repositório de dados. Dessa forma abstrai o acesso aos dados e sua implementação. O DAO é implementado como parte da camada de persistência atuando como uma camada de integração (Deepak, 2004).

4. ARQUITETURA PROPOSTA

A necessidade de um ponto inicial bem definido para que iniciássemos o desenvolvimento dos sistemas, foi o fator que motivou a adoção de uma arquitetura base. Sempre que o desenvolvimento de um sistema era iniciado havia todo um ritual a ser cumprido: que tecnologias utilizar, que arquitetura construir. O sistema seria um sistema puramente *Web* ou utilizaria algum outro tipo de formato. Então foi observado que questões desse tipo causavam problemas e nem sempre as opções selecionadas eram corretas ou se adequavam para o tipo de aplicação a ser desenvolvida.

A partir de uma prática adotada recentemente no departamento de desenvolvimento de sistemas da instituição da qual faço parte da equipe, tive a atenção voltada para uma solução adotada no desenvolvimento dos sistemas daquele setor. A prática consistiu na criação de uma arquitetura base para o desenvolvimento das aplicações objetivando prover aos desenvolvedores uma forma de se produzir Softwares onde se tivesse o máximo de reutilização.

4.1. Delimitação do trabalho

O presente trabalho é parte do trabalho de conclusão de curso do aluno Jorge Luís Fernandes da Costa Duarte, que teve seu inicio em trabalhos de disciplinas do curso de Tecnologia em Desenvolvimento de Software. Neste sentido, foi realizado um levantamento das necessidades de requisitos não funcionais que influenciam na arquitetura e na definição da linha de produto a ser construída. A partir destes requisitos, foi modelado uma arquitetura proposta, descrita nas seções a seguir.

Para a validação da arquitetura do estudo de caso foi utilizado um sistema proposto na disciplina de Prática em Desenvolvimento de Software II. A proposta consistiu da implementação de um prontuário eletrônico do paciente que tinha como objetivo disponibilizar um sistema para o gerenciamento de prontuários tendo como foco a triagem e a consulta do paciente. A idealização do sistema surgiu do Prof. Alessandro José de Souza que observou uma lacuna existente no serviço público do Rio Grande do Norte propondo a implementação de um protótipo de um sistema de prontuário eletrônico para atender aos postos de saúde das comunidades desse Estado. A motivação da implementação de um sistema desse tipo foi a melhoria em termos de organização e gerência da informação no serviço de saúde comunitária do Rio Grande do Norte.

A proposta do sistema consistiu em que fosse possível permitir o acesso simultâneo a partir de vários lugares diferentes, garantir níveis de acesso aos usuários e manter centralizada as informações solucionando o problema de duplicação e/ou discrepância nas informações dos pacientes. Quanto às funcionalidades o sistema deveria contemplar a abertura do prontuário (cadastro do paciente), controle de triagem, controle da consulta (incluindo agendamento da consulta, lançamento dos diagnósticos e exames), encaminhamento do paciente e controle das unidades de saúde que fossem integrar o sistema.

Durante a implementação do sistema foi observada a necessidade de realização de ajustes na arquitetura, contudo foi observada uma significativa melhora em níveis de organização de código, reutilização e produtividade. As funcionalidades implementadas do sistema foram: abertura do prontuário; realização da triagem; agendamento e realização da consulta; e lançamento de exames e diagnósticos.

Em continuidade a este trabalho e como complemento do trabalho de final de curso, será testada a arquitetura proposta em aplicativos reais.

4.2 - Visão da Arquitetura

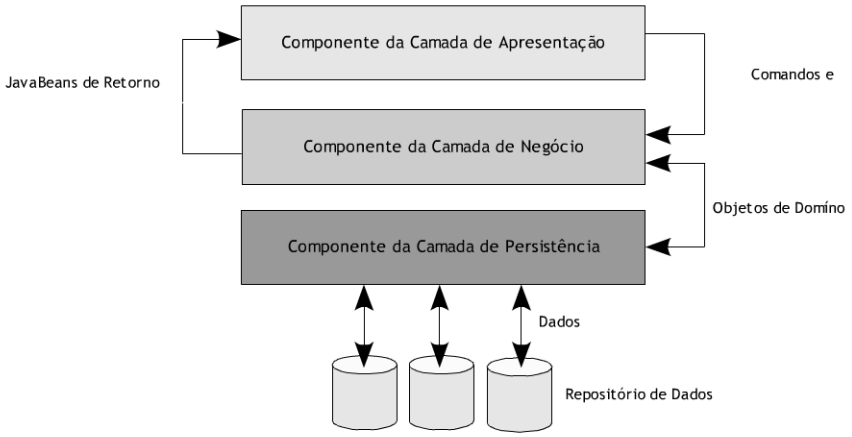

Figura 2: Visão Geral da Arquitetura

A arquitetura implementado é composta de três componentes, distribuídos em três camadas (Figura 2): apresentação, negócio e persistência. A implementação da arquitetura foi realizada utilizando alguns padrões de projeto como o *Command*, *Factory Method*, *Singleton* e *Data Access Object – DAO* - em conjunto com *frameworks* como o JavaServer Faces, Tiles e Hibernate.

4.3 – Componente da Camada de Apresentação

Toda aplicação que possui interação com o usuário necessita de uma *interface*, seja ela gráfica ou não. Os sistemas desenvolvidos baseados na arquitetura proposta terão suas interfaces gráficas desenvolvidas no formato *Web*. Pensando neste tipo de situação os componentes desenvolvidos nesta camada tem o objetivo de fornecer ao desenvolvedor uma estrutura facilitadora para construção de *interfaces* com o usuário . Dentro do seu papel disponibiliza ao desenvolvedor as funcionalidades: manipulação da entrada e saída de dados do usuário de forma facilitada, exibição de mensagens (validação, erros e resultados das operações) e controle do *layout* da *interface* gráfica.

Observando o diagrama da Figura 3, podemos classificar as funcionalidades existentes nesta camada em dois tipos: as que tratam da execução das ações (interação usuário sistema) e as que tratam da configuração e exibição da interface gráfica. Sobre a interação do usuário com o sistema foram observadas pelo menos duas ações básicas: as de simples exibição de telas e as que realizam algum tipo de processamento retornando uma resposta ao usuário. Outras operações também disponibilizadas são as de manipulação de dados: inserção, atualização, exclusão e consultas simples. A disponibilização de operações para a manipulação de dados foi observada diante da necessidade da implementação de cadastros simples (cadastros que não necessitem de regras de negócio para sua realização).

Visando suprir as necessidades de criação das páginas que irão compor a interface com o usuário foi incorporado à arquitetura o *Framework Tiles,* que através da sua técnica de construção de páginas baseadas em templates possibilita o reuso. Então como podemos observar no diagrama da Figura 3, foram disponibilizadas ao desenvolvedor as operações de manipulação do *layout* em tempo de execução. Dessa forma o desenvolvedor irá manter seu foco apenas na criação das páginas necessárias, sem se preocupar com o restante do *layout* da interface. Além das operações de manipulação do *layout* dos sistemas, foram criadas operações que tratam da exibição de mensagens aos usuários (mensagens de informação ao usuário sobre o resultado das operações realizadas, como erros e operações de sucesso).

Além da utilização do Tiles para o gerenciamento dos layouts, esse componente também utilizou em sua implementação *Frameworks JavaServer Faces* que é desenvolvido sobre o padrão *Modelo, Vista, Controle – MVC*[2].

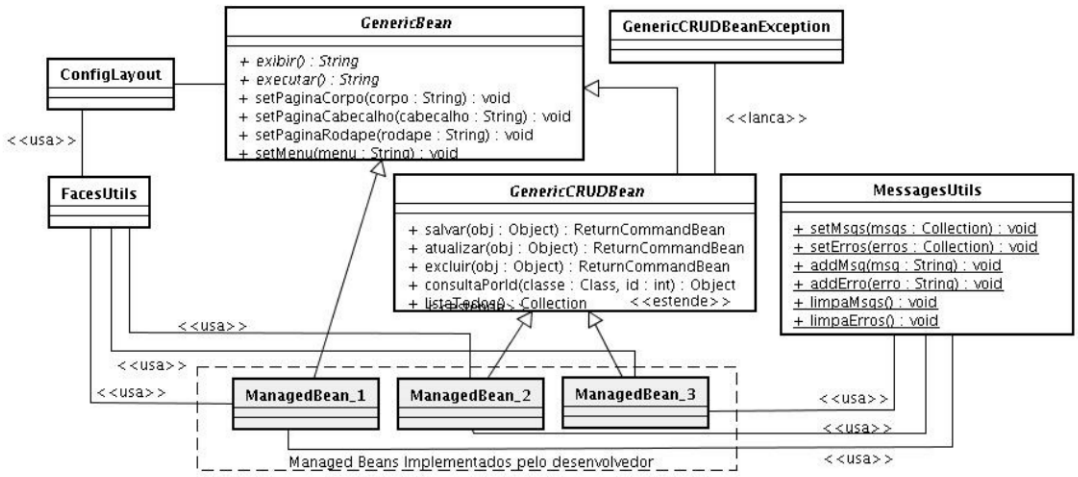

Figura 3: Componente da camada de apresentação

A utilização desse componente se dá através da implementação dos *Managed Beans* (Figura 3) que são parte integrante do *JSF*. Na arquitetura do *JSF* os *Managed Beans* correspondem ao controle, tratando as ações e eventos disparados pelos componentes da interface gráfica, fazendo também a correspondência das propriedades do formulário com as propriedades dos objetos de domínio. Sua utilização na arquitetura proposta deve ser feita realizando-se uma extensão de um dos `AbstractGenericBeans`. Os `AbstractGenericBeans` são classes que padronizam e disponibilizam operações de acordo com o propósito da sua utilização. Neste componente existem dois `AbstractGenericBeans`, um que disponibiliza as operações relativas a exibição de dados e execução de ações, e outro que disponibiliza as ações de manipulações de dados - as operações CRUD (*Create, Remove, Update*).

4.4 – Componente da Camada de Negócio

Este componente fornece ao desenvolvedor a estrutura para a implementação dos casos de uso dos sistemas. Sua implementação utiliza o padrão de projeto *EJB-Command* (extensão do *Command*) que tem a finalidade de prover o desenvolvimento modularizado e abstrair detalhes da tecnologia Java EE para o desenvolvedor.

A utilização do componente se dá pela implementação dos processadores (Figura 4) onde cada processador irá corresponder a um caso de uso. O processamento de uma operação nesse componente é realizado a partir de um comando e um objeto de domínio (correspondente ao caso de uso) que serão passados ao `ExecutorUtil`, sendo esse o responsável pela criação de um objeto encapsulador dos objetos recebidos. Dentro do contexto do EJB o objeto encapsulador será acessado identificando assim o processador a ser instanciado e executado. Nesse contexto uma

[2] Modelo, Vista, Controle - MVC: é um padrão de arquitetura onde a aplicação é dividida em modelo de dados, interface com o usuário e fluxo da aplicação possuindo como objetivo a separação lógica da aplicação.

transação envolverá todo o processo de execução garantindo a integridade dos dados. No caso do lançamento de alguma exceção as operações serão desfeitas através de um *rollback*[3].

Este componente fornece a implementação dos métodos básicos de manipulação de dados (inserção, atualização e exclusão) acessados pelas respectivas operações do componente da camada de apresentação.

Para o desenvolvimento deste componente foi utilizada a tecnologia Java EE por oferecer algumas APIs e que se justificam por retirar do desenvolvedor algumas preocupações, como por exemplo o gerenciamento de transações. Uma outra versão possível para esse componente pode ser implementada utilizando apenas classes Java simples através do padrão *Command*. Nessa versão questões abstraídas pela tecnologia Java EE deverão ser tratadas pelo desenvolvedor.

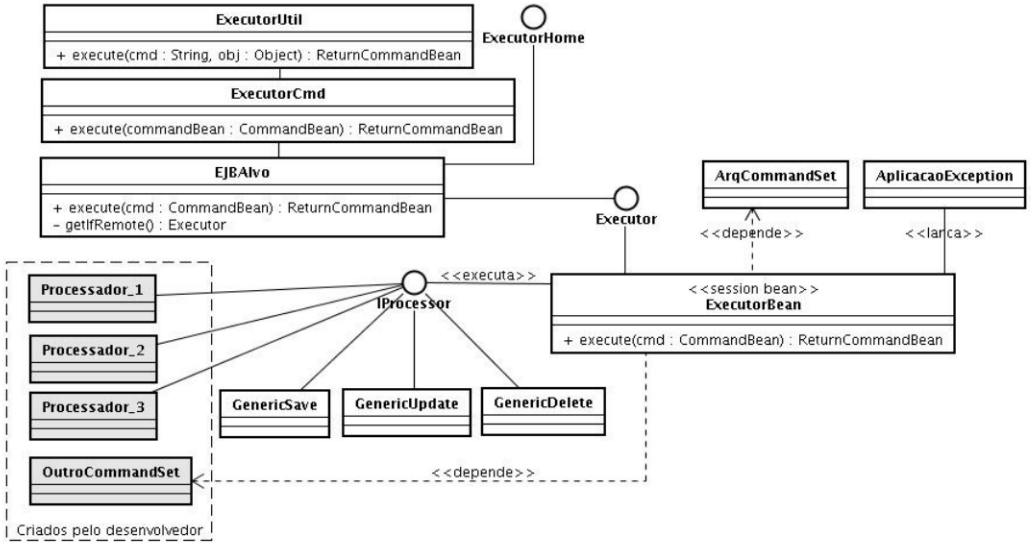

Figura 4: Componente da camada de negócio

4.5 – Componente da Camada de Persistência

Dentre os três componentes da arquitetura podemos dizer que este é o mais simples. Possui como objetivo desacoplar e servir como camada intermediária para o acesso ao repositório de dados. Sua implementação se constitui do padrão de projeto *DAO – Data Access Object*, utilizando também os padrões *Singleton* e *Factory Method*. O DAO oferece como vantagem abstração do tipo de implementação de acesso aos dados visando uma fácil manutenção caso venha a ser necessário modificar a forma de acesso ao repositório de dados, sendo essa uma de suas principais motivações.

Como pode ser visto no diagrama da Figura 5, esse componente disponibiliza através de uma interface genérica as operações para manipulação de dados. As consultas específicas devem ser implementadas pelo desenvolvedor e disponibilizadas através dos DAO's especializados.

Diante da necessidade de mudança no tipo de acesso a dados, deverá ser desenvolvido um `GenericDAO` específico ao tipo de acesso requerido. Esse DAO deverá implementar a interface `IGenericDAO` que fará com que o DAO a ser implementado siga o contrato da interface.

Nessa implementação todo o gerenciamento das transações foi deixado a cargo do contêiner, mais especificamente da *Java Transaction API – JTA*, embora esse controle possa ser feito programaticamente.

O componente de persistência utilizou na sua implementação o *Framework* de mapeamento objeto relacional, *Hibernate*, onde foi implementado o seu respectivo DAO Genérico. A escolha por um *Framework* deste tipo se deu devido a suas funcionalidades e facilidades observadas a partir de sua utilização em outros projetos, porém outros

[3] Rollback: é o descarte da operação transacional devido a um erro ocorrido em uma ou mais operações envolvidas em uma transação.

mecanismos de persistência podem ser acoplados bastando que suas características específicas sejam implementadas e abstraídas pela implementação da interface `IGenericDAO`.

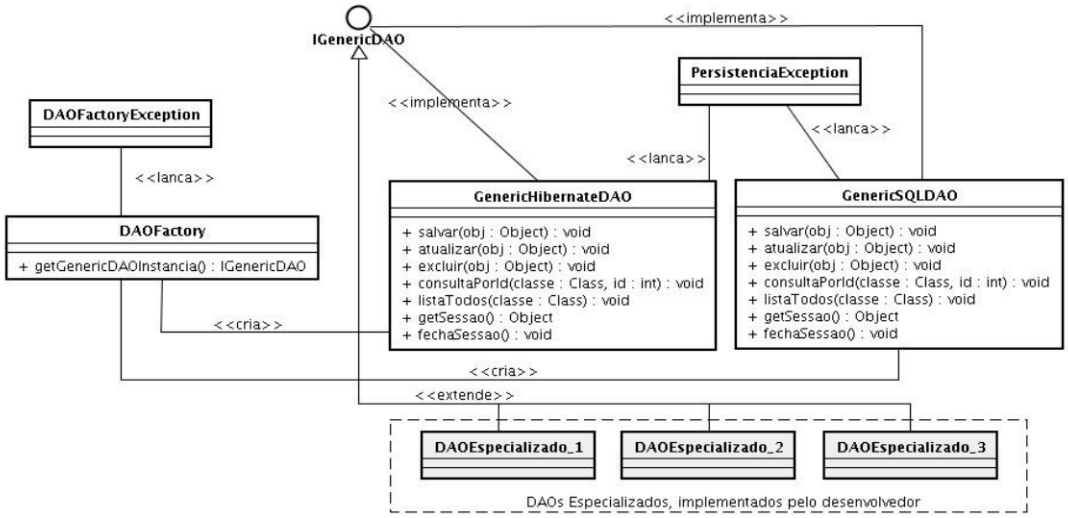

Figura 5: Componente da camada de persistência

5. CONCLUSÃO

O desenvolvimento baseado na linha de produto de software surge com uma evolução do desenvolvimento baseado em componentes. A sua idéia principal é a reutilização baseada nos pontos em comum de uma família de sistemas pertencentes a um mesmo domínio ou missão. O resultado da aplicação da arquitetura aqui proposta comprovou as vantagens defendidas pelo conceito de linha de produto de software, apesar da aplicação do conceito de linhas de produto ter ocorrido apenas em nível de instanciação da arquitetura. As vantagens aqui observadas foram a reutilização de uma forma organizada e estratégica incorrendo em uma produtividade considerável, não tendo sido alcançada antes. Foi observada também uma notável facilidade no desenvolvimento, uma vez que durante toda a implementação do sistema o foco foi mantido na codificação não tendo sido gasto tempo com configurações de *frameworks* e preocupações com *design* de interfaces do usuário. Por um outro lado é importante ressaltar que a arquitetura ainda se encontra num estágio inicial onde é observado seu potencial crescimento e amadurecimento com utilização de novos padrões e expansão de funcionalidades ou serviços, sendo necessário também uma validação mais efetiva em ambiente de produção.

6. REFERÊNCIAS BIBLIOGRÁFICAS

Bass, Len; Clements, Paul; Kazman, Rick. **Software Architecture in Practice**. Addison Wesley, 2002.

Clements, P.; Northrop, **L. Software Product Lines: Practices and Patterns**, SEI Series in Software Engineering, Addison Wesley, 2001.

Deepak, Alur; Crupi, John; Malks, Dan. Core J2EE Design Patterns, As Melhores Práticas e Estratégias de Design, Elsevier Editora, 2004.

Gamma, Erich; Helm, Richard; Jonhnson, Ralph; Vlissides, Jonh. **Design Patterns, Elements of Object-Oriented Software,** Addison Wesley, 1995.

Griss, M. Favarp. J. Alessandro, M. Interesting Feature Modeling with RSEB, 5th International Conference on Software Reuse, ACM/IEEE, Vienna, Austria, Junho 2000.

Gimenes, I.M.S. **O Enfoque de Linha de Produto para o desenvolvimento de Software**, XXI JAI - Livro Texto, Cap. 1, Florianópolis.

Jacobson, I.; Griss, M.; Jonhnson. **P. Software Reuse - Architecture Process and Organization for Business Sucess**, New York, Addison Wesley, 1997.

Larman. Graig. **Utilizando a UML e Padrões: Uma Introdução à Análise e Projetos Orientados a Objetos**. ED Bookman, 2000.