UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PAULO EDUARDO BOEIRA CAPELLER
VINÍCIUS CAMARGO ANDRADE

USO DO PROCESSO DIRIGIDO A RESPONSABILIDADES NO DESENVOLVIMENTO DA ARQUITETURA E MODELAGEM DO FRAMEWORK DE PREÇO DE VENDA

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA
2010

PAULO EDUARDO BOEIRA CAPELLER
VINÍCIUS CAMARGO ANDRADE

USO DO PROCESSO DIRIGIDO A RESPONSABILIDADES NO DESENVOLVIMENTO DA ARQUITETURA E MODELAGEM DO FRAMEWORK DE PREÇO DE VENDA

Trabalho de conclusão de curso de graduação, apresentado à disciplina Trabalho de Diplomação, do curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Coordenação de Análise e Desenvolvimento de Sistemas – COADS – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Tecnólogo.

Orientadora: Prof.ª Dr.ª Simone Nasser Matos

PONTA GROSSA
2010

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS PONTA GROSSA
GERÊNCIA DE ENSINO E PESQUISA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA DE TRABALHO DE DIPLOMAÇÃO

TERMO DE APROVAÇÃO

USO DO PROCESSO DIRIGIDO A RESPONSABILIDADES NO DESENVOLVIMENTO DA ARQUITETURA E MODELAGEM DO FRAMEWORK DE PREÇO DE VENDA

Paulo Eduardo Boeira Capeller

Vinícius Camargo Andrade

Este Trabalho de Diplomação foi considerado adequado como cumprimento das exigências legais do currículo do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas e aprovado em sua forma final pela Coordenação de Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná – Campus de Ponta Grossa.

_____
Prof.ª Dr.ª SIMONE NASSER MATOS
Orientador

_____
Profª. MSc. HELYANE BRONOSKI BORGES
Responsável pelo Trabalho de Diplomação

_____
Prof. Dr. ANDRE KOSCIANSKI
Coordenador do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Banca Examinadora:

_____
Prof. Dr. ANDREY RICARDO PIMENTEL

_____
Prof. Dr. GLEIFER VAZ ALVES

AGRADECIMENTOS

Agradecemos primeiramente a Deus pela saúde e sabedoria concedidas para a realização deste trabalho.

A nossos pais pelas nossas vidas e pelo apoio que dispuseram em nosso favor nas horas mais difíceis.

Especialmente a Prof.ª Dr.ª Simone Nasser Matos, que nos orientou com extremo profissionalismo e dedicação em todas as etapas deste grande desafio.

A Universidade Tecnológica Federal do Paraná, Campus Ponta Grossa, representado por sua diretoria, por nos proporcionar este curso de tão alto nível.

A todos os demais familiares, amigos e professores, que sempre mentalizaram positivamente em nosso favor e acreditam em nosso sucesso pessoal e profissional.

RESUMO

CAPELLER, Paulo E. B.; ANDRADE, Vinícius C. **Uso do Processo Dirigido a Responsabilidades no Desenvolvimento da Arquitetura e Modelagem do Framework de Preço de Venda**. 2010, 172f. Trabalho de Conclusão de Curso – Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2010.

Os aplicativos utilizados para formação de preço de venda implementam um único método de custeio, dificultando ao gestor encontrar o preço ideal para um produto ou serviço pois para realizar uma simulação deverá utilizar vários softwares. Além disso, não foi encontrado na literatura um *framework* para este fim. Este trabalho propõe a arquitetura de um *framework* no domínio de preço de venda, contendo três métodos, a saber, Custeio Baseado em Atividade, Sebrae e Custo Pleno, os quais são utilizados para determinação do preço de venda. A modelagem do *framework* foi desenvolvida com base no processo dirigido a responsabilidades que permitiu a identificação e a modelagem dos pontos de estabilidade e flexibilidade e dos subframeworks para este domínio. Relata também a implementação de alguns dos subframeworks, tais como: Gerenciador Atributos, Entradas e de Cálculo. Com o framework proposto o gestor pode calcular o preço de venda utilizando cada método implementado. Além disso, o desenvolvedor terá uma arquitetura mais flexível a qual pode ser reusada ou estendida para um novo método de formação de preço de venda.

Palavras-chave: *Framework* de Domínio. Preço de Venda. *Subframework*.

ABSTRACT

CAPELLER, Paulo E. B; ANDRADE, Vinícius C. **Use of Driven Responsibilities Process in Architecture and Modeling Development of Sales Prices Framework**. 2010, 172p. TD – Systems Analysis and Development Technology, Federal University of Technology - Paraná. Ponta Grossa, 2010.

The applications used for sales price formation implement a single costing method, making it difficult for the manager to find the ideal price for a product or service because to perform a simulation he will have to use several systems. Besides, a framework for this purpose was not found in the literature. This work proposes the architecture of a framework in the sales price domain, containing three methods: Activity-Based Costing (ABC), Sebrae, and Full Cost, which are used for sales price determination. The modeling of the framework was developed based on the Responsibility-Driven Process that allowed the identification and modeling of the stability and flexibility points and the sub frameworks for this domain. It also reports the implementation of some of the sub frameworks such as Attributes, Inputs, and Calculation Manager. With the proposed framework the manager can calculate the sales price using each implemented method. Furthermore, the developer will have a more flexible architecture that can be reused or extended for a new sales price formation method.

Keywords: Domain Framework. Sale Price. Sub Framework.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE EQUAÇÕES

LISTA DE SIGLAS

ABC	*Activity-Based Costing*
AM	Aplicação de Metapadrões
C	Componente
Ca	Custo de Aquisição
CA	Comum Ator
CCU	Comum Caso de Uso
CIF	Custos Indiretos de Fabricação
CORBA	*Common Object Request Broker Architecture*
DCOM	*Distributed Component Object Model*
Df	Despesas Fixas
EA	Específico Ator
ECU	Específico Caso de Uso
EIAC	Eliminar Inconsistência, Ambiguidade e Contradição
FA	*Framework Adaptable*
FCE	*Framework* Classe Estendida
FD	Fase de Desenvolvimento
Fp	Faturamento Previsto
F-UML	*Framework Unified Modeling Language*

ICMS	Imposto Sobre Operações Relativas à Circulação de Mercadorias e Prestação de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação
IPTU	Imposto Sobre Propriedade Predial e Territorial Urbano
It	Investimento Total
Ld	Lucro Desejável
MAP-F	Modelo de Arquitetura Preliminar Para o *Framework*
MRF	Modelo de Requisitos Para o *Framework*
MVC	*Model-View-Controller*
ORB	*Object Request Broker*
PDR-DFD	Processo Dirigido a Responsabilidade Aplicada ao Desenvolvimento de *Framework* de Domínio
PI	Processo Iterativo
SEBRAE	Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
Sf	*Subframework*
Su	Subsistemas
TH	*Template Hook*
UML	*Unified Modeling Language*
UPP	Utilização de Padrão de Projeto

SUMÁRIO

1 INTRODUÇÃO

Para se manterem no mercado, as empresas atualmente enfrentam diversos problemas, o principal deles é a formação de preço de venda. Durante anos, diversos autores desenvolveram métodos para calcular o preço de venda de um produto ou serviço, sendo que cada método é composto por suas próprias características. Este trabalho analisou os seguintes métodos: Método de Custeio Baseado em Atividades (*Activity-Based Costing*) (COGAN, 1999; SILVESTRE, 2002); Método Sebrae-PR (SEBRAE, 2009); Método Baseado no Custo Pleno (MARTINS, 2003).

A princípio os cálculos eram feitos manualmente, porém a dificuldade era identificar o método que fornecesse o preço ideal para o produto ou serviço. Atualmente existem software, tal como: Hipercusto (NATSAM CONSULTORIA 2008), que implementam métodos de formação de preço de venda que auxiliam o gestor, porém estes softwares são pagos ou implementam um método apenas. Além disso, atualmente na literatura não se encontrou um *framework* no domínio de preço de venda.

Neste contexto, este trabalho apresenta um estudo sobre os métodos de formação de preço de venda, juntamente com uma comparação sobre as metodologias para o desenvolvimento de *framework*.

Após a análise desta comparação (ANDRADE; CAPELLER; MATOS, 2010) foi aplicado o Processo Dirigido a Responsabilidades (MATOS; FERNANDES, 2008) para o desenvolvimento da arquitetura geral do *framework* de formação de preço de venda e a implementação dos *subframeworks* Gerenciador Atributos (denominado de *Attributes)*, Entradas (*FoodSystem*) e de Cálculo (*Calculation*). Estes *subframeworks* contêm as características comuns e específicas de cada método analisado possibilitando a criação de um aplicativo.

A arquitetura proposta atinge, do ponto de vista do desenvolvedor, as características do desenvolvimento de *frameworks* tais como: reúso, extensão, flexibilidade, manutenibilidade, entre outras. Do ponto de vista do gestor, abrange vários métodos de formação de preço de venda, proporcionando a facilidade e a agilidade no estabelecimento do preço de custeio, pois ele pode realizar o cálculo várias vezes e em cada iteração pode usar um dos métodos.

1.1 PROBLEMA

Os softwares de cálculo de preço de venda implementam geralmente um único método de custeio, dificultando a análise do gestor pois para haver um comparativo de preço ele deve executar vários sistemas diferentes, ou seja, não existe um *framework* neste domínio.

Um *framework* nesta área permite contemplar vários métodos de custeio. Com isso, o gestor pode utilizar somente um software para análise do preço ideal de venda para os produtos ou serviços.

A arquitetura do *framework* proposto facilita a sua extensão quando houver da necessidade de uma adição de um novo método.

1.2 OBJETIVOS

O objetivo geral e os específicos serão descritos a seguir.

1.2.1 Objetivo Geral

Desenvolver e modelar a arquitetura do *framework* de preço de venda, proporcionando a criação de um aplicativo que implementa vários métodos de custeio, trazendo as vantagens do desenvolvimento baseado em *framework* tais como: reusabilidade, manutenibilidade e flexibilidade. O aplicativo também permite ao gestor simular o preço de venda ou serviço utilizado um método em cada interação.

1.2.2 Objetivos Específicos

- Compreender os métodos que formam o *framework* para análise, elaboração e otimização de preço de venda;
- Levantar os pontos de estabilidade e flexibilidade dos *subframeworks* através da abordagem PDR-DFD (Processo Dirigido a Responsabilidade aplicado ao Desenvolvimento de *Framework* de Domínio);
- Desenvolver a modelagem dos *subframeworks*;
- Implementar 3 (três) *subframeworks.*

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está divido em cinco capítulos. O Capítulo 2 apresenta as definições e classificações de um *Framework.* Também relata o estudo sobre os métodos de formação de preço de venda e aborda as metodologias de desenvolvimento de *framework* de domínio. O Capítulo 3 descreve o desenvolvimento da arquitetura geral do *framework* de preço de venda e a modelagem dos *subframeworks*. O Capítulo 4 exibe os resultados obtidos por este trabalho. E, por fim, o Capítulo 5 apresenta as conclusões do trabalho e faz propostas para trabalhos futuros.

2 FRAMEWORK DE DOMÍNIO

Este Capítulo aborda um panorama sobre os *frameworks* de domínio e os métodos da literatura destinados ao estabelecimento de preço de venda. A Seção 2.1 apresenta as definições e classificações de um *framework*. A Seção 2.2 descreve a definição de preço de venda juntamente com os métodos estudados. E, por fim, a Seção 2.3 aborda as metodologias de desenvolvimento de *framework* de domínio.

2.1 FRAMEWORKS: UMA VISÃO GERAL

Segundo Johnson e Foote (1988), “um *framework* é um conjunto de classes que personifica um projeto abstrato de uma solução para uma família de problemas relacionados”. Ou seja, *framework* é um conjunto de classes que podem ser reusadas em projetos diferentes, promovendo soluções para uma família de problemas relacionados (JOHNSON; FOOTE, 1988).

O desenvolvimento de *framework* possui como sua principal característica a generabilidade sobre o domínio de aplicações, com isso possibilita a especialização da estrutura para a produção de diferentes aplicações. Além disso, é indispensável apresentar em sua estrutura, flexibilidade e extensibilidade.

Segundo Rogers (1997), um *framework* objetiva gerar diferentes aplicações para um domínio que pode ser bem específico ou mais abrangente. Quanto à abrangência existem duas categorias: horizontal e vertical. A horizontal não depende do domínio da aplicação, ou seja, pode ou não ser utilizado em diferentes domínios. Por fim, a categoria vertical resolve os problemas de um determinado domínio.

A base do *framework* corresponde aos aspectos que são comuns para o domínio das aplicações, ou seja, dada as aplicações-exemplo A, B e C, se estas apresentarem características comuns – interseção não vazia – pode-se criar um *framework* em que as características iguais irão pertencer à base. Estes aspectos são também conhecidos como *frozen spots*, enquanto os flexíveis, de variabilidade ou *hot spots* representam aspectos que são específicos para alguns produtos de software em um domínio (PREE, 1999).

Segundo Matos (2008), “um *framework* é um conjunto de subsistemas, *subframeworks* e componentes interagindo e colaborando de forma a produzir um projeto geral para um domínio particular”. A Figura 1 representa esta definição graficamente.

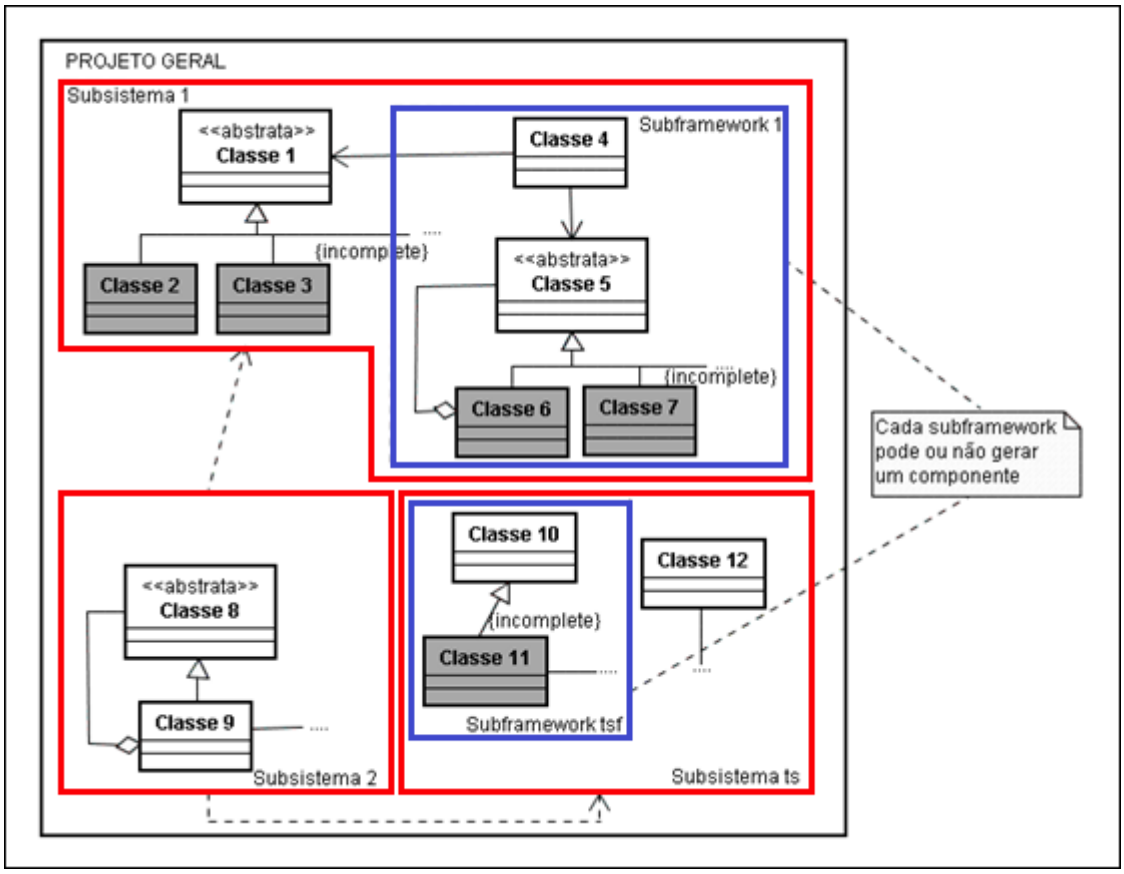


Figura 1 - Representação gráfica de um *framework*
Fonte: Matos (2008, p.43).

Um subsistema é um conjunto de classes interagindo e colaborando para executar uma função. Um *subframework* “representa uma parte reutilizável de um subsistema ou um conjunto deles que pode ser reusada em uma nova aplicação-exemplo” (MATOS, 2008). Por sua vez, um componente é o resultado da transformação de um *subframework* ou de um subsistema em um projeto executável, permitindo ao desenvolvedor alterar somente o que for flexível.

2.1.1 Classificação

Segundo Silva (2000), a classificação de *frameworks* se dá a partir do modo de como ele será empregado, sendo voltado para dados ou para a arquitetura. Se

um *framework* refere-se à dados, a construção de aplicações é feita pelas várias maneiras de se combinar a instanciação das classes já existentes. Já os relacionados à arquitetura possuem subclasses construídas tendo como base as classes pré-existentes do *framework*.

Um *framework* quanto ao seu propósito ou escopo pode ser classificado como (TALIGENT, 1994):

I. Aplicação ou de Infraestrutura: cobre funcionalidades aplicáveis a vários domínios, como por exemplo, *frameworks* para sistemas operacionais, comunicação, redes, construção de interfaces, entre outros;
II. Domínio: refere-se às funcionalidades aplicáveis há um domínio específico, tais como: *frameworks* para telecomunicações, manufatura, controle de produção, multimídia, engenharia financeira, otimização de preço de venda, entre outros;
III. Suporte ou de Integração de *Middleware*: oferecem serviços de baixo nível, normalmente utilizados na integração de aplicações distribuídas e seus componentes, como por exemplo, *frameworks* CORBA ORB, DCOM, entre outros.

Um *framework* pode ser classificado também quanto ao processo de reúso de suas classes durante a sua extensão como sendo: caixa branca, caixa preta ou caixa cinza (YASSIN; FAYAD, 2000), sendo que:

I. Caixa Branca (*White Box*): está fortemente ligada às características da linguagem orientada a objetos, sendo o reúso e a extensão das funcionalidades desenvolvidas por meio da herança de classes e da implementação de métodos;
II. Caixa Preta (*Black Box*): possibilita a extensão por meio de interfaces para componentes que podem ser “plugados” ao *framework*, tendo o reúso das funcionalidades aplicadas por meio da composição ou da definição de interfaces para os componentes;
III. Caixa Cinza (*Gray Box*): esse tipo de *framework* possui flexibilidade e extensibilidade, além de esconder informações que não são necessárias para os desenvolvedores de aplicações.

Neste trabalho é abordado o *framework* de domínio, pois os métodos estudados pertencem ao mesmo contexto, ou seja, apesar dos métodos possuírem diferenças entre si, todos tem como principal função o estabelecimento de preço de venda.

2.2 ANÁLISE DE DOMÍNIO DO FRAMEWORK DE DOMÍNIO PROPOSTO

Segundo Bornia (2002), preço de venda é o valor atribuído a um produto ou serviço, cujo valor deve ser suficiente para cobrir todas as despesas (fixas e variáveis) e o custo do que está sendo vendido, gerando ainda assim, uma margem de lucro para investimentos posteriores.

Uma empresa possui despesas fixas e variáveis (BORNIA, 2002). Despesas fixas, ou também chamadas de custos fixos, são aqueles custos que independentemente da quantidade produzida ou vendida serão sempre as mesmas, como por exemplo, salários administrativos, aluguel, IPTU.

Despesas variáveis ou custos variáveis são obtidos em função das vendas, ou seja, quanto maior o nível de produção e de trabalho em uma empresa, maior será a despesa. Normalmente, as despesas variáveis se caracterizam por um porcentual sobre o valor das vendas efetivadas, tal como: comissões.

Para uma empresa obter sucesso, vários aspectos importantes devem ser levados em consideração, entre eles está o estabelecimento do preço de venda, uma vez que o mercado está cada vez mais competitivo devido às tarifações e impostos, da necessidade de novos investimentos e do surgimento de novas despesas (SILVA, 2001).

Com o crescimento da concorrência e do mercado consumidor, as empresas se obrigaram a procurar novas formas de analisar e tomar decisões que atendam as suas necessidades. Além de essas decisões serem satisfatórias, elas precisam ser tomadas com rapidez, pois o mercado é variável e o preço de venda precisa ser revisto a todo o momento.

2.2.1 Métodos de Preço de Venda

Nesta Subseção são abordados todos os métodos estudados para a elaboração do *framework* de domínio. Ela está subdividida em 3 (três) subseções. A Subseção 2.2.1.1 aborda a análise do Método de Custeio Baseado em Atividades (*Activity-Based Costing*). A Subseção 2.2.1.2 descreve a análise do Método Sebrae-PR. Por fim, a Subseção 2.2.1.3 destaca a análise do Método Baseado no Custo Pleno.

2.2.1.1 Método de Custeio Baseado em Atividades (*Activity-Based Costing*)

Segundo Nakagawa (1995), o Custeio Baseado em Atividades trata-se de um método desenvolvido para facilitar a análise estratégica de custos relacionados com as atividades que mais impactam o consumo de recursos de uma empresa.

Os recursos são classificados como diretos e indiretos. Os recursos diretos são os custos relacionados diretamente com a produção de um produto. Por exemplo, custo de mão-de-obra direta, horas de mão-de-obra direta, matéria-prima e a quantidade de horas de utilização de máquinas na produção.

Já os recursos indiretos, também chamados de custos indiretos de fabricação (CIF) ou *overhead*, não estão relacionados diretamente aos custos dos produtos, por esse motivo, o método *Activity-Based Costing* (ABC) leva grande vantagem em relação aos sistemas tradicionais de custeio (COGAN, 1995).

Diferentemente dos demais métodos, o Custeio Baseado em Atividades leva em consideração as atividades envolvidas na produção, como por exemplo: preparação de máquinas, inspeções de qualidade, recebimento e movimentação de materiais, entre outros.

O Custeio Baseado em Atividades pressupõe que estas consomem recursos, gerando custos, e que os produtos utilizam tais atividades, absorvendo seus custos. Assim, os procedimentos do ABC consistem em dividir a organização em atividades relacionadas à produção, determinada pelo gestor de finanças. Após a compreensão e comportamento destas atividades, identificam-se as causas dos custos, em seguida, aplica-se o procedimento da alocação dos custos aos produtos de acordo com as intensidades de uso (BORNIA, 2002).

Segundo Bornia (2002), as seguintes etapas são dadas para o estabelecimento do preço de venda seguindo o método de Custeio Baseado em Atividades:

I. Mapear e detalhar as atividades;
II. Alocar custos às atividades;
III. Redistribuir os custos das atividades indiretas até as diretas;
IV. Calcular os custos dos produtos.

2.2.1.1.1 Exemplo de Aplicação do Método

Para o melhor entendimento do método ABC, o exemplo a seguir é aplicado em uma atividade agrícola (CRAZUSKI *et al.*, 2008). Nesta atividade, o método ABC localiza indicadores representados não somente por fatores de custo (máquinas, mão-de-obra, materiais, etc.), mas também que permitam compreender todo o processo de cada atividade desenvolvida na empresa.

A Figura 2 representa o ciclo para a produção de uma cultura agrícola.

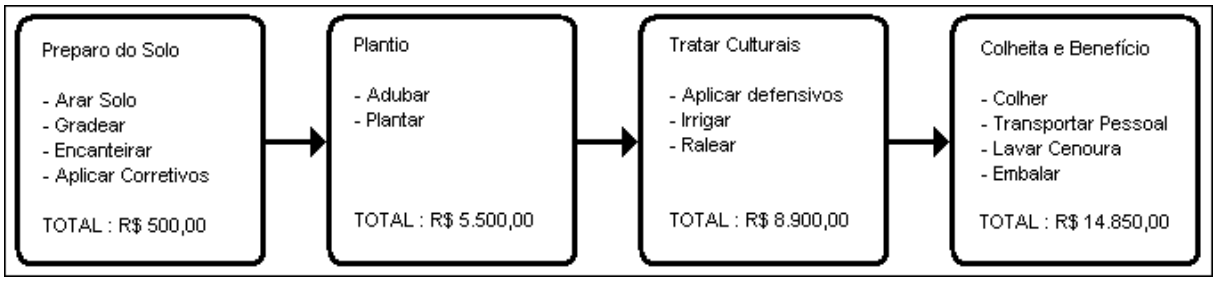


Figura 2 - Modelo de ciclo produtivo agrícola
Fonte: Adaptado de Crazuski *et al.* (2008).

Utilizando os dados da Figura 2, a Tabela 1 representa a apuração de custo com base em volume, e a Tabela 2 mostra os mesmos valores, porém adotando o método ABC.

Tabela 1 - Apuração de custo com base em volume

Fator	Custo (R$)
Custo com base em Volume	
Mão-de-obra direta	1.000,00
Custos indiretos	950,50
Insumos	3.500,00

Fatores de Custo	
Máquinas	2.100,00
Serviços de terceiros	850,00
Custo total	**8.400,00**

Fonte: Pimenta; Rocha; Lemes (2007).

Tabela 2 – Apuração de custo com base em atividades

Atividades	Custo (R$)
Preparo do Solo	450,00
Aplicação de Corretivos	80,00
Plantio	4.250,00
Aplicação de defensivos	2.568,00
Irrigação	548,00
Colheita	250,00
Transporte	254,00
Custo total	**8.400,00**

Fonte: Pimenta; Rocha; Lemes (2007).

Nas duas tabelas foram utilizados custos totais idênticos a fim de mostrar o agrupamento das atividades e seus respectivos custos de acordo com o método empregado.

Uma característica particular da área agrícola é sazonalidade da ocorrência das atividades. Como as pulverizações, fertilizações e outras atividades são executadas sem função de algumas variáveis, como as infestações de pragas e os índices pluviométricos, não há execução de todas as atividades durante o ano todo, nem capacidade de se prever com precisão quando elas ocorrerão. Percebe-se, dessa forma, que um sistema de gestão de custos para a área agrícola não pode ser o mesmo que é utilizado nos ambientes industriais, onde os processos de fabricação se repetem nos vários meses do ano (DOMENICO; LIMA, 1995).

2.2.1.2 Método Sebrae-PR

Para o melhor entendimento do método Sebrae-PR, é necessário conhecer alguns conceitos utilizados pelo mesmo. São eles:

I. Custo de Aquisição: somatória dos custos verificados entre a requisição de produtos através da emissão de ordens de fabricação ou compra, até o efetivo recebimento dos mesmos na área de recebimento;
II. Margem de Contribuição: permite conhecer quais produtos são mais lucrativos, evitando assim um investimento em produtos menos rentáveis (CRAZUSKI *et al.*, 2008);
III. Ponto de Equilíbrio: é o valor ou a quantidade que a empresa precisa vender para cobrir o custo das mercadorias vendidas, as despesas variáveis e as despesas fixas;
IV. Lucro Desejável - representa a remuneração do capital investido na empresa. Normalmente, é calculado com percentual superior às aplicações financeiras disponíveis no mercado de baixo risco;
V. Lucro Líquido: resultado do faturamento total decrementado da soma de todas as despesas e custos. Tem por objetivo remunerar o investimento feito na empresa. Se não for distribuído, o valor do patrimônio líquido é aumentado.

2.2.1.2.1 Etapas de Aplicação do Método

Primeiramente é necessário retirar o valor do ICMS do custo de aquisição do produto. Com isso, é feito um levantamento do percentual das despesas fixas e variáveis da empresa, como ilustrado em Crazuski *et al.* (2008).

É necessário o estabelecimento da porcentagem de lucro desejável, isso se dá pela obtenção do faturamento previsto pela empresa e seu investimento total.

A precisão do valor final de venda do produto depende da precisão do levantamento efetuado para o cálculo. Depois disso, chega-se a um valor que ainda deve ser analisado, pois deve-se, ao fim do processo, verificar se este preço calculado está compatível com os processos que o mercado pratica (SEBRAE, 2009).

2.2.1.2.2 Exemplo de Aplicação do Método

Tendo como exemplo uma empresa fictícia que possua um Faturamento previsto (Fp) no valor de R$ 90.000,00, uma despesa fixa (Df) de R$ 1.000,00, possui como despesas variáveis (%Dv) , um percentual de 3% sobre o faturamento previsto para comissões, além de 20% para ICMS e 6% para a tributação simples, um investimento total (It) de R$110.000,00, um Lucro Desejável (%Ld) de 4% ao mês e um produto cujo Custo de Aquisição(Ca) foi de R$18,00, tem-se:

Sendo: *Fp*= 90.000,00
It = 110.000,00
Ca = R$18,00

- Percentual de Despesas Fixas

$$\%Df = \frac{Df}{Fp} = \frac{10.000,00}{90.000,00} = 11,11\% \quad (1)$$

- Percentual de Despesas Variáveis

$$\%Dv = \sum \%Dv = 20\% + 6\% + 3\% = 29\% \quad (2)$$

- Lucro Desejável

$$Ld = \frac{It}{\%Ld} = 110.000,00\ x\ \frac{4}{100} = 4.400,00 \quad (3)$$

- Lucro Desejável Sobre Faturamento Previsto

$$Ldf = \frac{Ld}{Fp} = \frac{4.400,00}{90.000,00} = 4,889\% \quad (4)$$

- Custo Unitário de Aquisição da Mercadoria Desconsiderando o ICMS

$$Ca = 18,00 - \frac{20}{100} = 14,40 \quad (5)$$

- Preço de Venda

$$Pe = \frac{Ca}{100\% - (\%Df + \%Dv + \%Ll)}$$

$$Pe = \frac{14{,}40}{100\% - (11{,}11\% + 29\% + 4{,}889\%)} \quad (6)$$

$$Pe = \frac{14{,}40}{1 - (0.1111 + 0.29 + 0.04889)}$$

$$Pe = \frac{14{,}40}{0.55001} = 26{,}18$$

2.2.1.3 Método Baseado no Custo Pleno

Segundo Nascimento; Vartaniam (1999, p.34), “método de custeio pleno é aquele em que todos os custos e despesas de uma entidade são levados aos objetos de custeio, normalmente unidades de produtos e/ou ordens de serviços”.

O método de custo pleno leva em consideração todos os gastos de uma organização, sem exceção. Porém, não faz distinção entre custos fixos e variáveis, o que pode levar a uma empresa a vender um produto e não cobrir as despesas do mesmo.

O seu funcionamento consiste na soma de todos os custos e despesas do produto ou serviço, envolvendo matérias-primas, mão-de-obra direta, custos indiretos de produção, custos de transformação, custos de produção e despesas de vendas e administração.

Após realizada esta soma, é acrescentada uma margem de lucro que é definida de acordo com a necessidade de venda e sua fórmula está descrita no trabalho de Crazuski *et al.* (2008).

2.2.1.3.1 Etapas de Aplicação do Método

Inicialmente, define-se o custo envolvido em matérias-primas, mão-de-obra direta e custos indiretos de produção, então soma-se gerando o valor dos Custos de Produção. A esse resultado acrescenta-se o valor das Despesas de vendas e

administração, gerando o valor do Custo de Produção e Venda. Por fim, insere-se uma margem de lucro e se estabelece o preço de Venda (COGAN, 1999).

A Tabela 3 exemplifica o uso da aplicação do método baseado no custo pleno.

Tabela 3 - Exemplo da aplicação do Método baseado no Custo Pleno

Variável	Valor
Matérias-primas	15,00
Mão-de-obra direta	4,00
Custos indiretos de produção	9,00
Custos de Produção	**28,00**
Despesas de vendas e administração	6,00
Custo de Produção e Venda	**34,00**
Margem de lucro (35% do custo de produção e venda)	11,90
Preço de Venda	**45,90**

Fonte: Cogan (1999).

Na Tabela 3, aplica-se uma margem de lucro de 35%, sobre os custos de produção e de venda, pois nesse caso os custos das matérias-primas são elevados em relação aos custos de transformação (constituído pelos custos da mão-de-obra e pelos custos indiretos de produção), com isso, o preço de venda é influenciado pelo Custo de Produção e Venda.

Há outro caso, que os custos de transformação são bem maiores em relação ao de material direto, com isso, a margem de lucro é influenciada pelos custos de transformação, como ilustrado na Tabela 4.

Tabela 4 – Exemplo da aplicação do Método baseado no Custo Pleno

Variável	Valor
Matérias-primas	5,00
Custos de transformação	40,00
Custos de Produção	**45,00**
Despesas de vendas e administração	10,00
Custo de Produção e Venda	**55,00**
Margem de lucro (70% dos custos de transformação)	28,00
Preço de Venda	**83,00**

Fonte: Cogan (1999).

Caso a margem de lucro da Tabela 4 fosse a mesma adotada da Tabela 3 (35% dos custos de produção e vendas), o que equivaleria a R$ 19,25, o preço de venda proposto então seria de R$ 74,25.

2.3 ABORDAGENS PARA O DESENVOLVIMENTO DE FRAMEWORK DE DOMÍNIO

Para a realização deste trabalho, levantaram-se na literatura as abordagens que contemplavam um processo de definição arquitetural para o desenvolvimento de *framework*s e *subframework*s. Dentre elas, destacam-se as seguintes: Johnson (1993), Taligent (1994), Landin e Niklasson (1995), Pree (1999), Fayad *et al.* (1999), Mattson (2000), Butler e Xu (2001), Braga (2002), e por fim, Matos e Fernandes (2008a).

Após o estudo das abordagens, realizou-se uma análise qualitativa entre elas, considerando-se os seguintes critérios:

I. Criação de subsistemas (SU), *subframework*s (SF) e componentes (C): possibilita a definição de uma arquitetura que facilita o entendimento do funcionamento do sistema, bem como seu reúso;
II. Aplicação de padrões de projeto e metapadrões: permite aumentar a reusabilidade na fase de projeto;
III. Análise arquitetural: é um processo iterativo que permite que sejam feitos refinamentos no modelo arquitetural que inicialmente foi escolhido de forma a contribuir para a criação de uma arquitetura que contemple algumas características julgadas interessantes, tais como: integridade, reusabilidade e flexibilidade. Para Butler e Xu (2001), esse processo é denominado de refatoração arquitetural;
IV. Estilo Arquitetural: reduz o esforço para compreender o sistema desenvolvido por outra pessoa e, portanto, diminui a quantidade de informações que serão assimiladas em um novo projeto, permitindo facilitar o reúso (BASS *et al.*, 2003);
V. Reúso Arquitetural: oferece benefícios que incluem redução de custo de construção e do tempo de produção;

VI. Definição dos *Hot Spots* nos *subframework*s: representam os pontos que são específicos para uma aplicação-exemplo e constituem a parte central do projeto de *framework* (PREE, 1999).

No Quadro 1, relacionam-se as abordagens citadas e os critérios. As linhas onde aparecem o “X” significa a presença do critério, caso contrário, a sua ausência.

Abordagens	Características							
	Criação de:			Aplicação de Padrões de Projeto e Metapadrões	Análise Arquitetural	Estilo Arquitetural	Reúso Arquitetural	Definição dos *hot spots* no *subframework*
	SU	SF	C					
Johnson (1993)				X				
Taligent (1994)	X	X		X	X		X	
Landin e Niklasson (1995)	X			X	X			
Pree (1995)				X				
Mattson (1996)	X	X		X	X	X		
Fayad *et al.* (1999)				X	X	X	X	
Butler e Xu (2001)	X		X	X	X		X	
Braga (2003)				X	X	X	X	
Matos e Fernandes (2008a)	X	X	X	X	X	X	X	X

Quadro 1 - Critérios contemplados pelas abordagens da literatura
Fonte: Autoria Própria.

Observou-se que todas as abordagens contemplam a utilização de padrões de projeto e metapadrões durante a definição da arquitetura de *framework*. Com relação à Criação de Subsistemas, *Subframework*s e Componentes verificou-se que somente a abordagem de Matos e Fernandes (2008a) abrange mais detalhadamente seu processo de identificação e construção. Além disso, essa abordagem explica como a definição dos *hot spots* deve ser feita durante a construção dos *subframework*s na fase inicial do seu processo de construção. Isso, de acordo com Tang *et al.* (2004), pode contribuir para a confecção de um sistema

com um maior grau de flexibilidade e portabilidade, impactando sob os seus aspectos de desempenho, custo e escalabilidade.

Constatou-se também que o critério de Análise Arquitetural é contemplado na maioria das abordagens e é considerado um fator importante no processo de definição da arquitetura do *framework*. Menos da metade das abordagens não contemplam a definição de um estilo arquitetural e que, segundo Paris (2004), a sua escolha implica em um aumento de flexibilidade do sistema, sendo assim, um critério importante na definição da arquitetura. Por fim, o Reúso Arquitetural é contemplado por mais da metade das abordagens possibilitando o reúso de requisitos, projeto, métodos e de componentes de software.

Após a análise das abordagens, verificou-se que a de Matos e Fernandes (2008a) é a ideal para ser aplicada no desenvolvimento do *framework* de preço de venda, pois oferece o processo detalhado para sua construção além de contemplar a criação de *subframeworks*, um dos pontos pesquisados neste trabalho. Além disso, contempla também os benefícios fornecidos pelas outras abordagens, dentre eles, podem-se citar: estilo arquitetural, aplicação de padrões de projeto e metapadrões.

2.3.1 Processo Dirigido por Responsabilidades Aplicado ao Desenvolvimento de Frameworks de Domínio (PDR-DFD): Metodologia Adotada

O processo de desenvolvimento de *framework*s de domínio corresponde a uma evolução iterativa de sua estrutura de classes, que envolve atividades como identificação de classes, modelagem de cenários, identificação de estados de objetos, entre outros.

Uma abordagem que pode ser utilizada para o desenvolvimento de *framework*s de domínio é a abordagem PDR-DFD (Processo Dirigido a Responsabilidade aplicado ao Desenvolvimento de *Framework* de Domínio) (MATOS; FERNANDES, 2006). Na PDR-DFD as responsabilidades são declarações gerais sobre as ações que um objeto executa e mantém, e sobre as decisões principais produzidas por um objeto que afeta os outros.

Essas responsabilidades são encontradas a partir das ações que um sistema deverá executar e nas informações que necessitam ser mantidas ou criadas (WIRFS-BROCK; MCKEAN, 2003). Além disso, o PDR-DFD foi construído com o

objetivo de dar um contexto adequado de aplicação do método de identificação antecipada de pontos de estabilidade e de flexibilidade, proposto por Matos e Fernandes (2008a).

2.3.1.1 Descrição das Fases da Abordagem PDR-DFD

As fases da abordagem proposta por Matos e Fernandes (2007) são as seguintes:

I. Definir o Domínio: o projeto deve ser iniciado pelo conhecimento do domínio das aplicações-exemplo. Neste trabalho, as aplicações-exemplo no domínio de estabelecimento de preço de venda são os métodos ABC, Sebrae e Custo Pleno;
II. Projetar *Framework* de Domínio: desenvolve-se o *framework* propriamente dito, ou seja, geram-se os *frameworks* base e de aplicação. O primeiro representa um conjunto de classes comuns entre as aplicações-exemplo, já o segundo as classes específicas para cada aplicação-exemplo. Esta fase é composta por subconjuntos de subfases que serão descritas na Subseção 2.3.1.2;
III. Implementar o *Framework*: nesta etapa, traduz-se a especificação das classes para a linguagem escolhida, de acordo com a especificação da fase anterior;
IV. Instanciar o *Framework*: cada classe ou grupo de classes constituintes do *framework* será instanciado com a finalidade que se gera a aplicação-exemplo para um ou mais exemplos concretos;
V. Validar o *Framework*: esta etapa é utilizada para validar o *framework*, confrontando o que foi desenvolvido com o que foi especificado.

2.3.1.2 Fase Projetar Framework de Domínio

A fase “Projetar *Framework* de Domínio” (Figura 3) é composta por duas subfases: “Compreender Aplicação-Exemplo” e “Definir *Framework* Base e de Aplicação”.

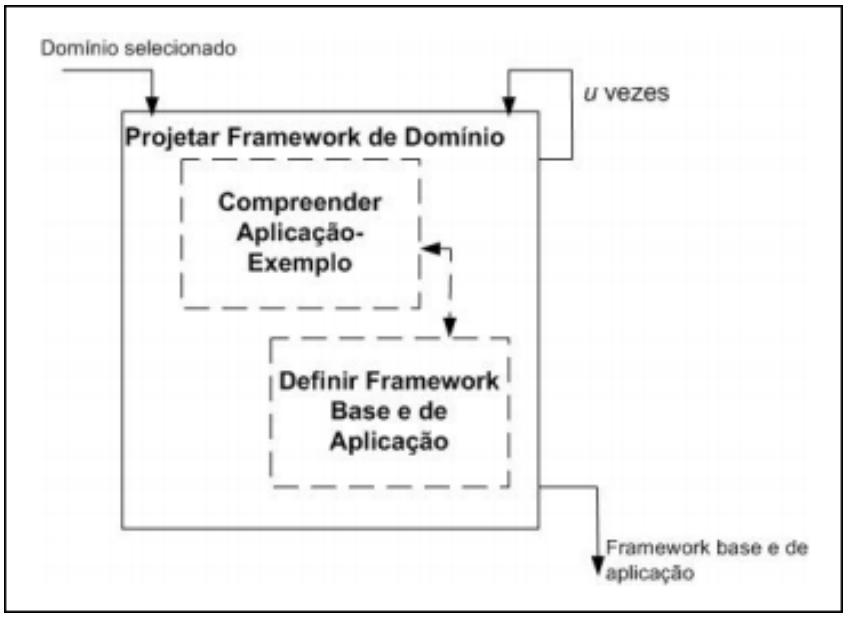


Figura 3 – Subfases da fase "Projetar *Framework* de Domínio"
Fonte: Matos; Fernandes (2007).

A subfase de Compreender Aplicação-Exemplo já foi desenvolvida para os métodos de formação de preço de venda e os artefatos gerados podem ser encontrados em Crazuski *et al.* (2008).

A subfase de Definir *framework* Base e de Aplicação é o foco deste trabalho. Nesta etapa é importante obter o conhecimento dos modelos de arquiteturais, dos requisitos e também do diagrama de classes de cada aplicação-exemplo, pois é nesta fase que será elaborada a modelagem do *framework*, definindo as classes que formarão o *framework* base e de aplicação.

A Figura 4 ilustra a subfase Definir *framework* Base e de Aplicação, que é formada pelas atividades: Definir a Arquitetura do Framework, Selecionar e Modelar Requisitos das Aplicações-Exemplo, Refinar os Requisitos, Refinar Classes e Definir *Subframeworks*, e Definir *Framework* Base e de Aplicação.

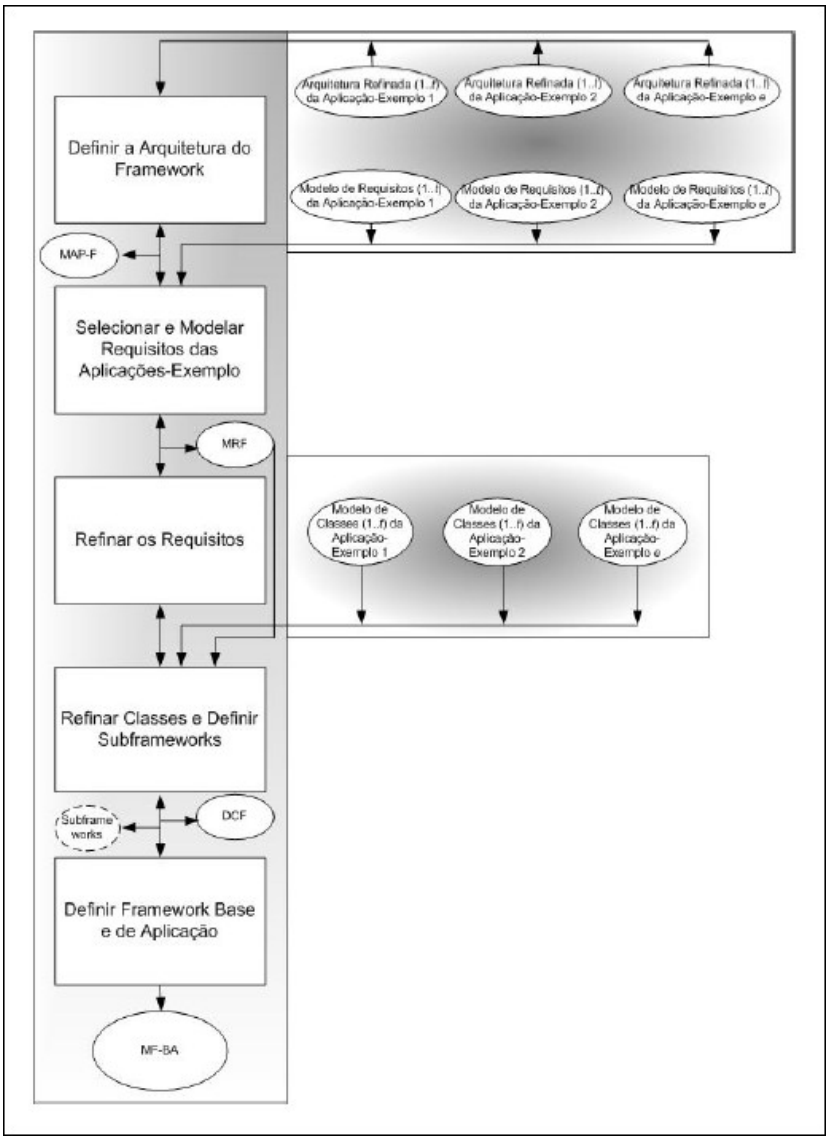


Figura 4 - Execução da subfase "Definir *Framework* Base e de Aplicação".
Fonte: Matos; Fernandes (2007).

A seguir, as atividades que compõem a subfase Definir Framework Base e de Aplicação.

Definir a Arquitetura do Framework

É neste momento que o projetista decide um estilo de arquitetura de alto nível satisfatória para o *framework*. Também é nesta atividade que há a separação entre aspectos funcionais, aspectos de interação entre objetos e aspectos não funcionais.

As ações internas que um objeto deve executar ou sofrer é descrita pela funcionalidade, enquanto as visões externas deste mesmo objeto e o controle de

atualizações via teclado entre visões e modelos são descritas pela interação. Por sua vez, os requisitos não funcionais dizem respeito às propriedades do sistema, como confiabilidade, tempo de resposta, manutenibilidade, entre outros.

Ao final desta fase, obtém-se o Modelo de Arquitetura Preliminar para o *Framework* (MAP-F).

Selecionar e Modelar Requisitos das Aplicações-Exemplo

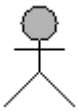
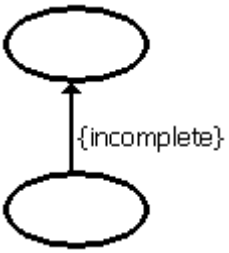
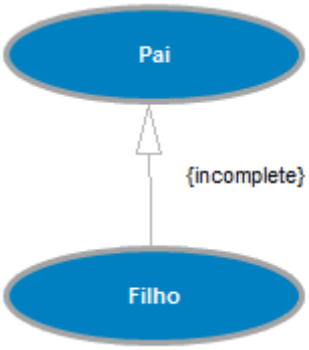
A partir de cada Modelo de Requisitos gerado na subfase “Compreender Aplicação-Exemplo”, identifica-se os requisitos iguais, similares e os diferentes em cada subsistema.

Para um auxílio na seleção dos requisitos iguais, similares e diferentes, aplica-se o método dirigido a responsabilidades, descrito em Matos (2008).

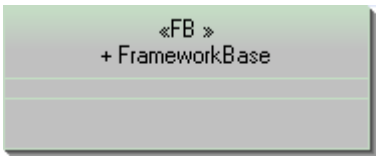
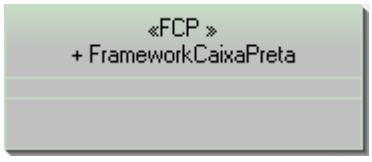
Deve-se elaborar um diagrama de Caso de Uso, notação UML-F (FONTOURA *et al.*, 2000), para esses requisitos. Os pontos de estabilidade e de flexibilidade identificados são facilmente compreendidos devido à utilização da notação UML-F. Esses diagramas constituem o Modelo de Requisitos para o *Framework* (MRF).

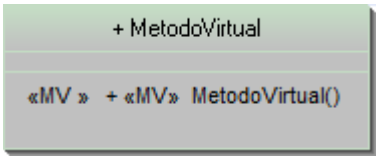
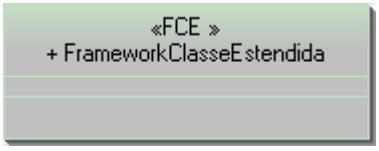
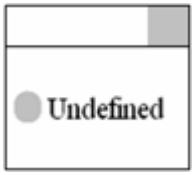
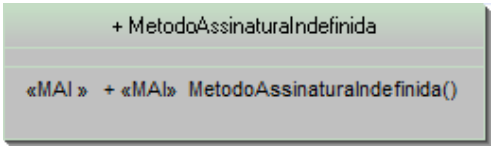
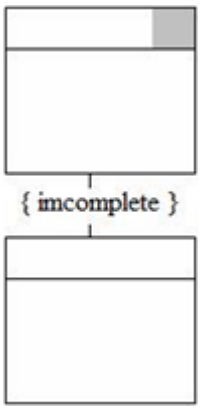
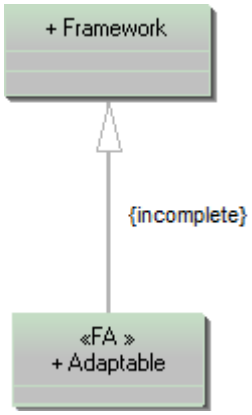
Neste trabalho, como não se encontrou na literatura, até a presente data, uma ferramenta de modelagem utilizando a UML-F, adaptaram-se as simbologias desta linguagem para serem utilizadas em uma ferramenta gratuita, denominada de OMONDO (CAPELLER, ANDRADE, SILVA, MATOS, 2010). Os Quadro 2 e 3 ilustram respectivamente os símbolos da UML-F adaptados para os diagramas de Caso de Uso e de Classes.

Notação Gráfica	Explicação	Objetivos	Notação Gráfica Omondo
	Borda do use case em negrito	Utilizados em casos que o aspecto é comum entre as aplicações concretas do domínio	
	Borda do ator em negrito	Ator comum entre as aplicações concretas do domínio	

	Caso de uso preenchido com a cor cinza	Utilizado em casos que o aspecto é diferente entre as aplicações concretas do domínio	
	Ator preenchido com a cor cinza	Ator diferente entre as aplicações concretas do domínio	
	A tag *incomplete* entre dois casos de uso	Torná-lo mais adaptável	

Quadro 2 - Explicação símbolos
Fonte: Adaptado de Gura e Carmo (2009, p. 32); Fontoura *et al.* (2000).

Notação Gráfica	Explicação	Objetivos	Notação Gráfica Omondo
	Borda da classe em negrito	Para mostrar o núcleo (base) do *framework*	
	Um quadrado preenchido de preto no canto superior direito da classe	Para mostrar *hot-spot* do *framework* caixa-preta	
	Um quadrado preenchido de cinza no canto superior direito da classe	Para mostrar que essa classe com todas as suas classes herdadas são um *hot-spot* do *framework* caixa-branca	

	Um círculo preenchido de cinza antes do nome do método	Para mostrar um método virtual. O método virtual indica que o método pode ter sua implementação redefinida	
	A tag *extensible* na classe	Para mostrar que a classe pode ser adaptada por meio da adição e/ou remoção de atributos e/ou métodos	
	A tag *undefined* para métodos com assinatura variada	Para mostrar métodos com assinatura indefinida	
	A tag *incomplete* na relação de agregação, associação e generalização	Para mostrar que *framework* pode ser adaptado adicionando outras classes relacionadas	

Quadro 3 - Explicação símbolos
Fonte: Adaptado de Gura e Carmo (2009, p. 32); Fontoura *et al.* (2000).

Refinar Requisitos

Deve-se refinar a descrição sobre os requisitos de cada subsistema de cada aplicação-exemplo, com o objetivo de compreender melhor seu funcionamento para o *framework.* Este processo é realizado tanto para os requisitos de flexibilidade quanto para os de estabilidade e consiste em uma elaboração de uma descrição dos casos de uso, ilustrado pelo Quadro 4.

Nome do Caso de Uso	
Estabilidade ☐	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
1.	**2.**
3.	**4.**
Exceções	

Quadro 4 - Descrição do caso de uso para o *framework*
Fonte: Matos; Fernandes (2007).

Refinar Classes e Definir *Subframeworks*

É possível compreender quais requisitos são classificados como de estabilidade e de flexibilidade por meio dos diagramas UML-F gerados na fase "Selecionar e Modelar Requisitos das Aplicações-Exemplo".

Com base em tal análise, identificam-se os pontos de estabilidade e de flexibilidade em nível de classes.

Definir o *Framework* Base e de Aplicação

Com as classes geradas anteriormente, separam-se as classes que farão parte do *Framework* Base e de Aplicação. Sendo que o primeiro representa os pontos de estabilidade, enquanto o segundo os pontos de flexibilidade do *framework*.

Os artefatos gerados nesta fase constituem-se em entrada para a fase: "Implementar *Framework*".

3 ARQUITETURA DO FRAMEWORK PROPOSTO

Este Capítulo aborda o desenvolvimento da arquitetura geral do *framework* proposto. A Seção 3.1 apresenta como a arquitetura do *framework* foi construída. A Seção 3.2 relata a aplicação da subfase *Definir a Arquitetura do Framework* para o domínio de preço de venda. A Seção 3.3 apresenta a modelagem da subfase *Selecionar e Modelar Requisitos das Aplicações-Exemplo*. A Seção 3.4 relata como foi o desenvolvimento da subfase *Refinar Requisitos*. A Seção 3.5 descreve a modelagem da subfase *Refinar Classes e Definir Subframeworks*. A Seção 3.6 apresenta a modelagem da subfase *Definir Framework Base e de Aplicação*.

3.1 CONCEPÇÃO DA ARQUITETURA DO FRAMEWORK

A arquitetura de software permite projetar a estrutura geral do sistema. Questões estruturais envolvem: (i) organização e estrutura geral de controle; (ii) protocolos de comunicação, sincronização; (iii) atribuição de funcionalidade a componentes de projeto; (iv) escalabilidade e desempenho; e (v) seleção de alternativas de projeto (MENDES, 2002).

Em uma visão e estratégia de reúso de software, tem-se enfatizado a importância de reúso centrado na arquitetura para o desenvolvimento de sistemas durante todo o seu ciclo de vida. A arquitetura de software serve como uma estrutura que permite o entendimento de componentes de um sistema e seus inter-relacionamentos, especialmente aqueles atributos que são consistentes ao longo do tempo e implementações.

Um dos pontos principais no desenvolvimento da arquitetura de um *framework* é a identificação dos pontos comuns (*frozen spots*) e flexíveis (*hot spots*). Um ponto comum caracteriza-se pelas funcionalidades iguais a todas as aplicações-exemplo do domínio. Por sua vez, os pontos flexíveis são as funcionalidades específicas de cada aplicação-exemplo. Neste trabalho, as aplicações-exemplo são os métodos da literatura usados para formação de preço de venda, tais como: Método de Custeio Baseado em Atividades (*Activity-Based Costing*) (COGAN, 1999; SILVESTRE, 2002); Método Sebrae-PR (SEBRAE, 2009); Método Baseado no Custo Pleno (MARTINS, 2003), os quais foram descritos na Subseção 2.2.

Após a realização da análise de cada método descrita em Crazuski *et al.* (2008), e a partir de artefatos gerados por este trabalho, a saber, Modelo de Requisitos e de Classes, foi possível aplicar a fase de “Projetar *Framework* de Domínio” descrito em 2.3.1.2.

3.2 SUBFASE 1 - DEFINIR A ARQUITETURA DO FRAMEWORK

Ao analisar cada arquitetura das aplicações-exemplo descritas no trabalho de Crazuski *et al.* (2008), composta por um conjunto de pacotes descritos no Quadro 5, pode-se iniciar a elaboração da arquitetura do *framework* proposto ilustrado na Figura 5.

ABC	SEBRAE	CUSTO PLENO
Cálculo de Preço de Venda	Calcular Preço de Venda	Cadastro de Atributos
Gerenciar Linha de Produção	Cadastrar Atributos	Conexão
Gerenciar Produto	Conexão	Cálculo do Preço de Venda
Conexão	Alimentar Sistema	
Gerenciar Atividade		
Gerenciar Atributo		
Gerenciar Dados do Atributo		
Gerenciar Dados da Atividade		

Quadro 5 – Elementos da Arquitetura dos Métodos de Preço de Venda
Fonte: Autoria Própria.

A arquitetura geral do *framework*, ilustrada na Figura 5, é composta pelos seguintes subsistemas: *ProductionLine, Attributes, Product, Activity, FoodSystem* e *Calculation*. O módulo *ProductionLine* equivale ao módulo Gerenciar Linha de Produção do método ABC.

O módulo *Attributes* corresponde aos módulos: Gerenciar Atributo (ABC), Cadastrar Atributos (Sebrae) e Cadastro de Atributos (Custo Pleno).

O módulo *Product* se relaciona ao módulo Gerenciar Produto do método ABC. O módulo *Activity* equivale ao módulo Gerenciar Atividade do método ABC.

O módulo *FoodSystem* corresponde aos módulos: Gerenciar Dados do Atributo (ABC), Gerenciar Dados da Atividade (ABC), Alimentar Sistema (Sebrae), Calcular Preço de Venda (Custo Pleno) – este foi colocado dentro deste subsistema pois além de ter o cálculo contém também as entradas.

O módulo *Calculation* equivale aos módulos: Cálculo de Preço de Venda (ABC), Calcular Preço de Venda (Sebrae) e Cálculo do Preço de Venda (Custo Pleno). Por fim, o módulo de Conexão não foi colocado nesta arquitetura porque foi implementado na camada de persistência.

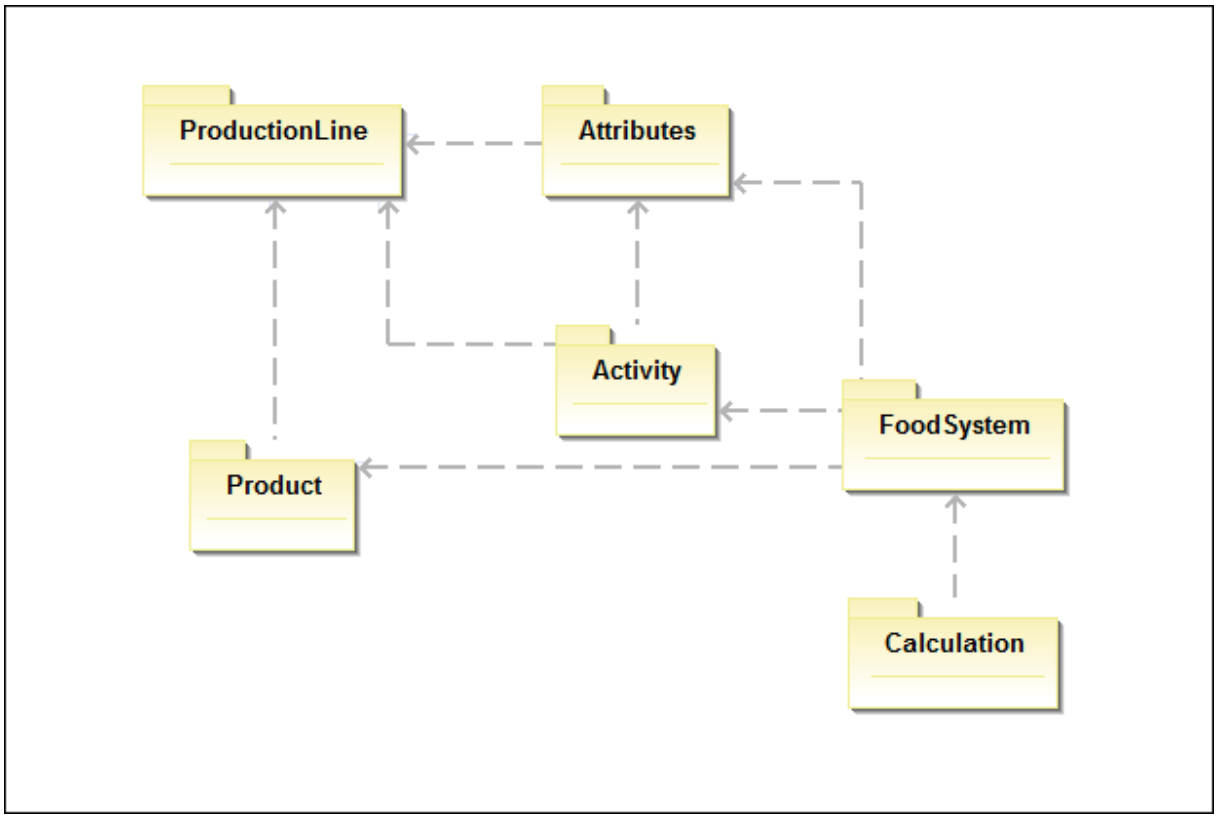


Figura 5 - Arquitetura Inicial para o *Framework*
Fonte: Autoria Própria.

Neste trabalho foi escolhido o desenvolvimento dos *Subframework FoodSystem, Attributes* e *Calculation*, pois verificou-se que possuem funcionalidades comuns entre os métodos de preço de venda estudados.

3.3 SUBFASE 2 – SELECIONAR E MODELAR REQUISITOS DAS APLICAÇÕES-EXEMPLO

Após o desenvolvimento do modelo arquitetural geral do *framework*, optou-se por implementar 3 (três) subsistemas: *Attributes; FoodSystem* e *Calculation*.

3.3.1 Modelo de requisitos

Tomando como base os modelos de requisitos criados para os métodos ABC, Sebrae e Pleno (CRAZUSKI *et al.*, 2008) elaborou-se um refinamento destes modelos com o objetivo de melhorar a compreensão das funcionalidades. Os novos modelos estão ilustrados nos Apêndices A, B e C.

A partir dos novos diagramas, identificaram-se os casos de uso que eram iguais, similares ou diferentes entre as aplicações-exemplo (MATOS, 2008). Um caso de uso é classificado como igual quando o conjunto de responsabilidades está contido em outro das aplicações-exemplo analisadas. O caso de uso será classificado como diferente quando não existe nenhuma responsabilidade igual entre as aplicações-exemplo, ou seja, a interseção entre eles é igual a 0 (zero). Caso contrário, se a interseção for diferente de 0 (zero) será classificado como similar. Para identificação utilizou-se os seguintes passos.

PASSO 1 – Análise das aplicações-exemplo

Analisou-se cada aplicação-exemplo descritas no trabalho de Crazuski *et al.* (2008) a partir do artefato gerado tal como o modelo de requisitos.

PASSO 2 – Refinamento do modelo das aplicações-exemplo

Elaborou-se uma planilha com os seguintes dados: *Actor, Perspective, UseCase, Responsability* e *Examplename*, onde:

- *Actor:* Corresponde ao usuário ou sistema que interage com a aplicação;
- *Perspective:* Equivale ao subsistema de onde o caso de uso pertence, tais como: gerenciar produto, cadastro de atributos, validação, etc.;
- *UseCase:* Ações a qual o ator pode executar;
- *Examplename:* Nome da aplicação-exemplo o qual a perspectiva ou subsistema pertence, por exemplo, para o domínio de formação de

preço de venda se tem as seguintes aplicações-exemplo: ABC, SEBRAE e CUSTO PLENO.

- *Responsability:* Provê a descrição textual do caso de uso. Por exemplo: o Quadro 6 representa a descrição textual do caso de uso “Abrir Tela de Cálculo”.

Ator	Sistema
1. Usuário solicita abertura da Tela de Cálculo	2. Exibir Tela de Cálculo

Quadro 6 - Descrição textual do Caso de Uso “Abrir Tela de Cálculo”
Fonte: Adaptado de Crazuski *et al.* (2008).

Neste caso, a responsabilidade do caso de uso “Abrir Tela de Cálculo”, será “Exibir Tela de Cálculo”. Importante ressaltar que o caso de uso não possui necessariamente apenas uma responsabilidade.

A Figura 6 ilustra apenas uma parte da planilha gerada para os métodos ABC, Sebrae e Custo Pleno. A planilha em sua totalidade é apresentada no Quadro 13, Apêndice D.

Actor	Perspective	UseCase	Responsability	Examplename
Usuário	Gerenciar Linha de Produção	Consultar Linha	Exibir Tela de Consulta	ABC
Usuário	Gerenciar Linha de Produção	Consultar Linha	Exibir todas as linhas	ABC
Usuário	Gerenciar Linha de Produção	Consultar Linha	Verificar opção selecionada	ABC
Usuário	Gerenciar Linha de Produção	Consultar Linha	Exibir mensagem de erro	ABC
Usuário	Gerenciar Linha de Produção	Editar Linha	Exibe Tela de Cadastro	ABC
Usuário	Gerenciar Linha de Produção	Editar Linha	Exibe uma linha	ABC
Usuário	Gerenciar Linha de Produção	Editar Linha	Verificar opção selecionada	ABC
Usuário	Gerenciar Linha de Produção	Editar Linha	Exibir mensagem de erro	ABC
Usuário	Gerenciar Linha de Produção	Excluir Linha	Exibe mensagem de confirmação	ABC
Usuário	Gerenciar Linha de Produção	Excluir Linha	Verificar opção selecionada	ABC
Usuário	Gerenciar Linha de Produção	Excluir Linha	Excluir Linha	ABC
Usuário	Gerenciar Linha de Produção	Excluir Linha	Exibir mensagem de erro	ABC
Usuário	Gerenciar Linha de Produção	Cadastrar Linha	Exibir Tela de Cadastro	ABC
Usuário	Gerenciar Linha de Produção	Cadastrar Linha	Verificar opção selecionada	ABC
Usuário	Gerenciar Linha de Produção	Cadastrar Linha	Inserir Linha	ABC
Usuário	Gerenciar Linha de Produção	Cadastrar Linha	Exibir mensagem de erro	ABC
Usuário	Gerenciar Linha de Produção	Cadastrar Linha	Limpar campos	ABC
Usuário	Gerenciar Produto	Consultar Produto	Exibir Tela de Consulta	ABC
Usuário	Gerenciar Produto	Consultar Produto	Exibir Todos os Produtos	ABC
Usuário	Gerenciar Produto	Consultar Produto	Verificar opção selecionada	ABC
Usuário	Gerenciar Produto	Consultar Produto	Exibir mensagem de erro	ABC

Figura 6 – Tabela de Caso de Uso UML de cada aplicação-exemplo
Fonte: Autoria Própria.

O Quadro 7, mostra os números totais de subsistemas de cada aplicação-exemplo e responsabilidade que foram analisados.

Examplename	*Perspective*	*Responsability*
ABC	8	201
SEBRAE	5	
CUSTO PLENO	3	

Quadro 7 – Número total de aplicação-exemplo e reponsabilidades
Fonte: Autoria Própria.

PASSO 3 – Análise semântica

Após realizar o estudo das aplicações-exemplo descritas no trabalho de Crazuski *et al.* (2008), elaborou-se a planilha com todos os modelos de requisitos, em que se realizou uma análise semântica dos nomes das responsabilidades.

Das 201 responsabilidades dos modelos de requisitos, houve alteração de nomes em 21 delas, representando um total de aproximadamente 10,5% de problema semântico. Um problema semântico ocorre quando nas aplicações-exemplo têm-se responsabilidades com nomes diferentes, mas semanticamente possuem a mesma finalidade (MATOS, 2008). A Figura 7 ilustra essa planilha em que consta o nome da responsabilidade antiga (coluna Responsabilidade), seu novo nome e baseado em qual método de formação de preço de venda foi estabelecido esse nome.

Quantidade	Método	Responsabilidade	Novo Nome	Baseado no Método
1	ABC	Exibir todos dados dos atributos	Exibir todos os atributos	ABC
2	ABC	Exibir Tela de Cadastro	Exibir Tela de Cadastro de Atributos	SEBRAE
3	ABC	Exibir o preço de venda	Exibir resultado na Tela de Cálculo	SEBRAE
4	ABC	Consultar valores	Buscar valores dos atributos cadastrados no Banco de Dados	SEBRAE
5	ABC	Verificar opção selecionada	Verificar ação escolhida	PLENO
6	ABC	Exibir todos os atributos	Mostrar os atributos cadastrados no Banco de Dados	PLENO
7	ABC	Exibir um atributo	Mostrar os dados do Atributo selecionado	PLENO
8	SEBRAE	Listar os valores já persistidos anteriormente no Banco de Dados	Exibir todos dados dos atributos	ABC
9	SEBRAE	Exibir Tela de Alimentação de Valores	Exibir Tela de Cadastro	ABC
10	SEBRAE	Emitir mensagem de confirmação de exclusão	Exibe mensagem de confirmação	ABC
11	SEBRAE	Efetuar Cálculo do Preço de Venda	Calcular o preço de venda baseado nos valores informados	PLENO
12	SEBRAE	Verificar preenchimento de valores	Validar dados	PLENO
13	SEBRAE	Verificar qual ação executar	Verificar ação escolhida	PLENO
14	SEBRAE	Listar os valores já persistidos anteriormente no Banco de Dados	Mostrar os atributos cadastrados no Banco de Dados	PLENO
15	SEBRAE	Buscar dados do registro no Banco de Dados	Mostrar os dados do Atributo selecionado	PLENO
16	PLENO	Mostrar a tela de Consulta de Atributos	Exibir Tela de Consulta	ABC
17	PLENO	Mostrar a tela de Consulta de Atributos	Exibir Tela de Consulta	ABC
18	PLENO	Confirmar com o Usuário se ele realmente deseja excluir o atributo selecionado	Exibe mensagem de confirmação	ABC
19	PLENO	Calcular o preço de venda baseado nos valores informados	Efetuar Cálculo do Preço de Venda	SEBRAE
20	PLENO	Mostrar o preço de venda calculado	Exibir resultado na Tela de Cálculo	SEBRAE
21	PLENO	Mostrar a Tela de Cadastro de Atributos	Exibir Tela de Cadastro de Atributos	SEBRAE

Figura 7 – Responsabilidades que sofreram alterações em seus nomes
Fonte: Autoria Própria.

PASSO 4 – Modelo de Requisitos de cada Subsistema

A seguir será descrito o modelo de requisitos para cada subsistema que foi ilustrado na Figura 5.

a) Modelo de Requisitos: *Attributes*

Considerando o modelo de requisitos elaborado nos passos anteriores, definiram-se os seguintes requisitos comuns para o subsistema *Attributes.*

- Cadastrar Atributo;
- Consultar Atributo;
- Editar Atributo;
- Excluir Atributo;
- Conectar ao Banco de Dados;
- Desconectar do Banco de Dados.

Também foram definidos os requisitos específicos:

- Atributo Variável (Custo Pleno);
- Despesas Fixas (Sebrae-PR);
- Filtrar Pesquisa (Custo Pleno).

Com os requisitos já definidos, elaborou-se um diagrama de caso de uso UML-F para o subsistema *Attributes,* ilustrado na Figura 8.

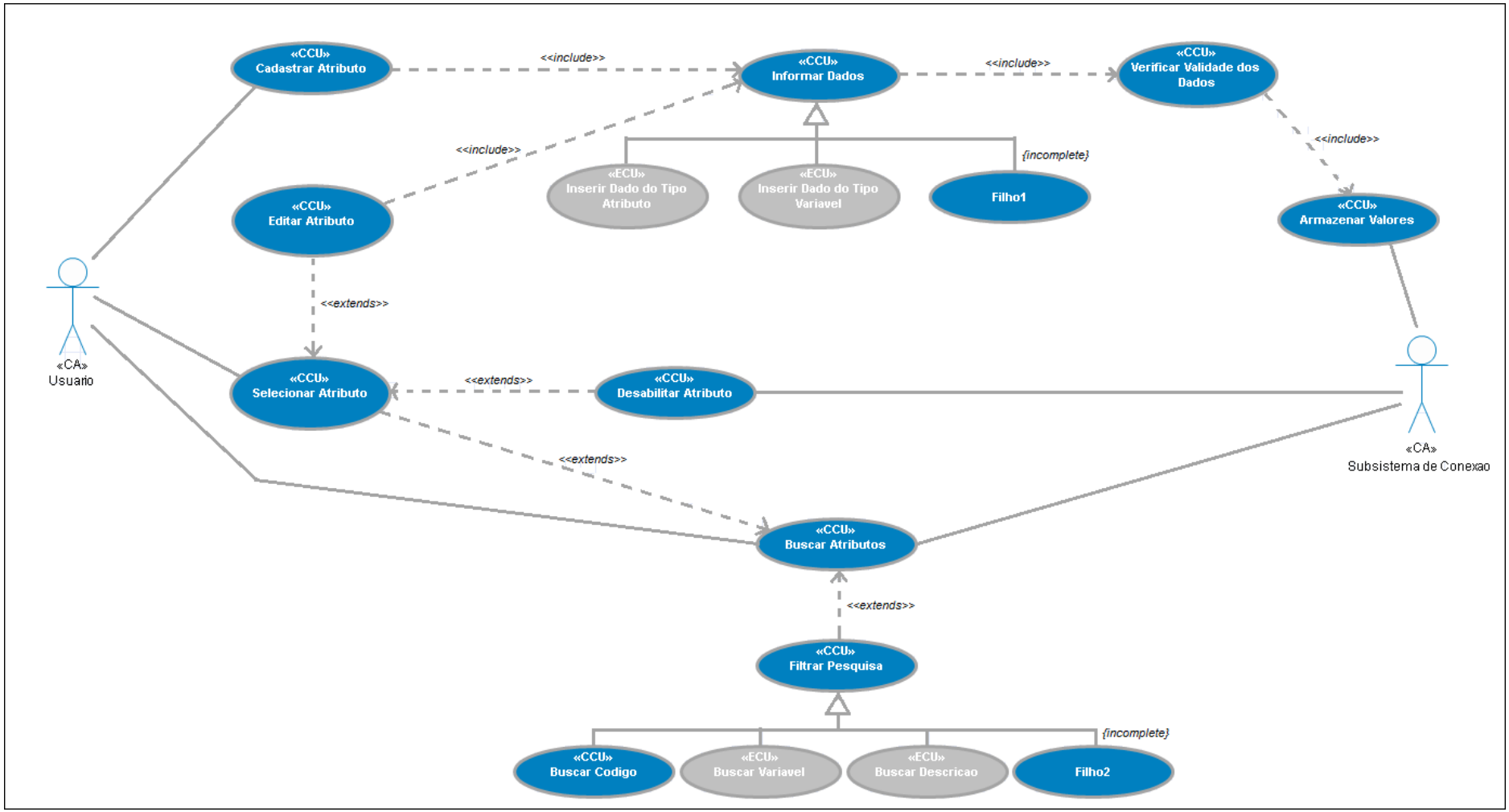


Figura 8 – Diagrama de Caso de Uso UML-F para o subsistema *Attributes*
Fonte: Autoria Própria.

As notações utilizadas na Figura 8 são: caso de uso base (*frozen spot*), caso de uso específico ou flexível (*hot spot*), ator base e uma herança entre casos de uso em uma aplicação reusando o *framework*.

Os casos de uso base estão todos representados pelo estereótipo "*CCU*" antecedendo o nome do caso de uso, por serem aspectos comuns entre as aplicações-exemplo do domínio, os casos de uso específicos seguem a mesma regra, porém são representados pelo estereótipo "*ECU*" antecedendo o nome do caso de uso.

Ambos os atores são atores bases, ou seja, atores comuns entre as aplicações-exemplo do domínio, e são representados pelo estereótipo "*CA*" antes do seu nome (por exemplo: <<*CA*>> *Usuario*).

Outra notação utilizada neste diagrama é o conceito de generalização entre casos de uso, representado por dois casos de uso interligados por um "*{incomplete}*", o que significa que poderá futuramente ter novas generalizações/especializações, deixando assim o *framework* mais adaptável e flexível.

b) Modelo de Requisitos: *FoodSystem*

Ao analisar os modelos e diagramas desenvolvidos no trabalho de Crazuski *et al.* (2008), foi elaborado um diagrama de caso de uso UML-F para o subsistema *FoodSystem,* que está ilustrado na Figura 9. O processo de identificação dos pontos de estabilidade e de flexibilidade usado foi o mesmo descrito para o *Attributes*.

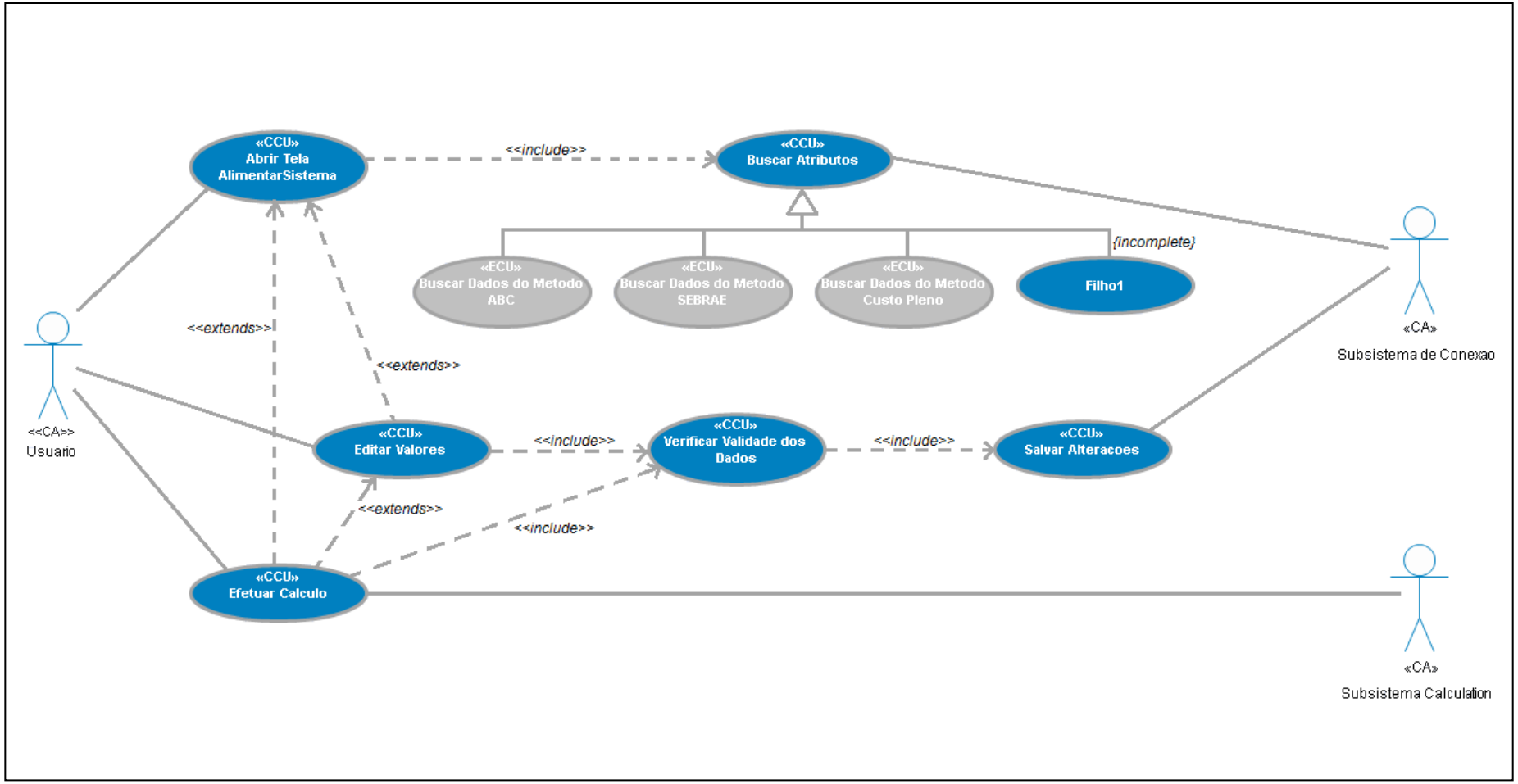


Figura 9 – Diagrama de Caso de Uso UML-F para o subsistema *FoodSystem*
Fonte: Autoria Própria.

As notações utilizadas para modelar o subsistema *FoodSystem* seguem o mesmo padrão do subsistema *Attributes* (Figura 8), no qual são utilizados casos de uso base *(frozen spot)*, caso de uso específico *(hot spot)*, ator base e uma herança entre casos de uso em uma aplicação reusando o *framework.*

Os estereótipos "*CCU"* e *"ECU"* representam os casos de uso base e específicos, respectivamente, entre as aplicações-exemplo do domínio.

Os atores existentes nestes subsistemas correspondem aos atores base, em que são representados pelo estereótipo *"CA"* antes de seu nome.

Também neste subsistema é possível a utilização do conceito de generalização/especialização, representado pelos casos de uso interligados por um *"{incomplete}"*.

c) Modelo de Requisitos: *Calculation*

A Figura 10 ilustra o diagrama de caso de uso UML-F para o subsistema *Calculation*.

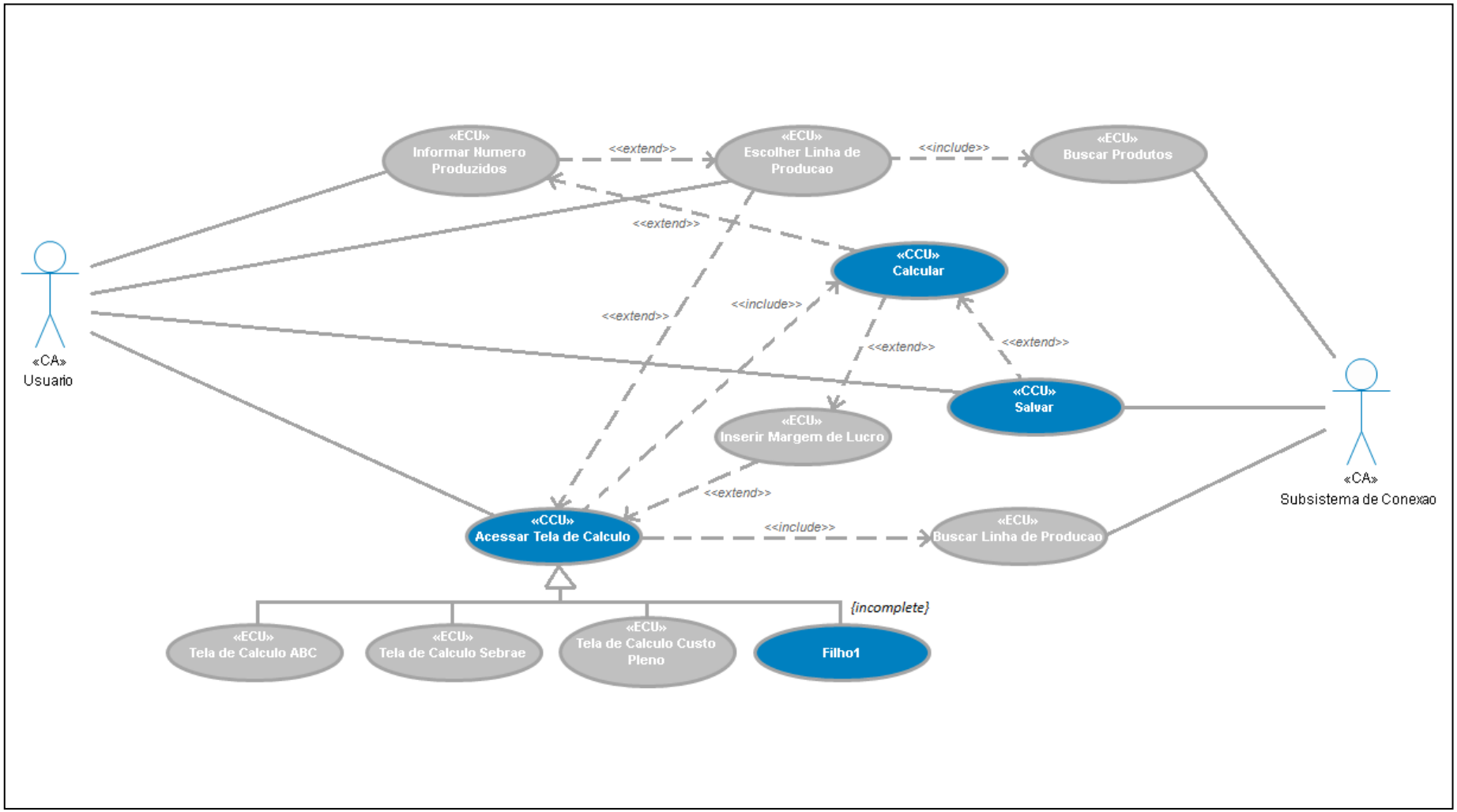


Figura 10 – Diagrama de Caso de Uso UML-F para o subsistema *Calculation*
Fonte: Autoria Própria.

Os estereótipos contidos nos casos de uso e nos atores identificam os casos de uso e atores base e específicos entre as aplicações–exemplo do domínio.

As notações usadas seguem o mesmo padrão dos diagramas de caso de uso UML-F dos subsistemas *Attributes* e *FoodSystem.*

3.4 SUBFASE 3 – REFINAR OS REQUISITOS

Nesta subfase são refinados os requisitos do sistema a fim de melhorar o entendimento e consequentemente deixar o *framework* mais flexível a novos métodos que futuramente venham ser acrescentados. Esta Subseção está dividida da seguinte forma. A Subseção 3.4.1 aborda o refinamento de requisitos para o Subsistema *Attributes*. A Subseção 3.4.2 descreve o refinamento de requisitos para o Subsistema *FoodSystem*. E por fim, a Subseção 3.4.3 destaca o refinamento de requisitos para o Subsistema *Calculation.*

3.4.1 Refinamento de requisitos para o Subsistema Attributes

Após a análise dos diagramas de caso de uso desenvolvidos por Crazuski *et al.* (2008), percebeu-se a necessidade de alterá-los, deixando-os mais adaptáveis e flexíveis aos novos métodos que formarão o *framework.*

Como relatado anteriormente, os diagramas de caso de uso desenvolvidos por Crazuski *et al.* (2008), foram expandidos e consequentemente sofreram alterações em suas descrições de caso de uso. Outro detalhe importante, foi a alteração dos quadros de descrições utilizados por Crazuski *et al.* (2008) para os quadros propostos por Matos e Fernandes (2007), pois nesses há possibilidade de assinalar se o caso de uso é comum (Estabilidade) ou específico (Flexibilidade).

Para exemplificar a mudança ocorrida no diagrama e na descrição dos casos de uso, os Quadros 8 e 9 ilustram respectivamente a descrição textual do Caso de Uso “Cadastrar Atributo”, desenvolvido por Crazuski *et al.* (2008), e a descrição do Caso de Uso “Cadastrar Atributo” do Diagrama de Caso de Uso UML-F para o subsistema *Attributes,* desenvolvido neste trabalho.

Ator	**Sistema**
1. Usuário acessa Tela de Cadastro 3. Usuário informa valores 4. Usuário escolhe opção	2. Exibir Tela de Cadastro 5. Verificar opção selecionada
Fluxo alternativo 5. Verificar opção selecionada: 5.1. Caso seja “Salvar”, tem a responsabilidade de realizar “Inserir Atributo”. a) Uma mensagem é retornada sobre o sucesso ou insucesso do cadastro do atributo 5.2. Caso seja “Cancelar” tem a responsabilidade de limpar os campos.	

Quadro 8 - Descrição textual do Caso de Uso “Cadastrar Atributo”
Fonte: Crazuski *et al.* (2008).

Nome do Caso de Uso	Cadastrar Atributo
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
1. Usuário acessa a Tela de Cadastro de Atributos	
	2. Exibir Tela de Cadastro de Atributos
	3. Requisitar o caso de uso “Informar Dados”
Exceções	

Quadro 9 - Descrição textual do caso de uso “Informar Dados”
Fonte: Autoria Própria.

As descrições dos Casos de Uso UML-F para o subsistema proposto encontram-se em sua totalidade no Apêndice E.

3.4.2 Refinamento de requisitos para o Subsistema FoodSystem

Para o refinamento de requisitos do Subsistema *FoodSystem* também foram elaboradas as descrições dos Casos de Uso UML-F seguindo o modelo de quadros propostos por Matos e Fernandes (2007). O Quadro 10 ilustra a descrição textual do caso de uso “Abrir Tela Alimentar Sistema”, referente ao Diagrama de Caso de Uso UML-F para o Subsistema *FoodSystem* (Figura 9).

Nome do Caso de Uso	Abrir Tela Alimentar Sistema
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
1. Usuário acessa a Tela Alimentar Sistema	
	2. Exibir Tela Alimentar Sistema
	3. Requisitar o caso de uso “Buscar Atributos”
Exceções	

Quadro 10 - Descrição textual do caso de uso “Abrir Tela Alimentar Sistema”
Fonte: Autoria Própria.

As demais descrições textuais do Diagrama de Caso de Uso do Subsistema *FoodSystem* encontram-se no Apêndice F.

3.4.3 Refinamento de requisitos para o Subsistema Calculation

O modelo de quadros propostos por Matos e Fernandes (2007) também foi utilizado para o refinamento de requisitos do Subsistema *Calculation* na elaboração das descrições dos Casos de Uso UML-F. O Quadro 11 ilustra a descrição textual do Caso de Uso “Acessar Tela de Calculo”, referente ao Diagrama de Caso de Uso UML-F para o Subsistema *Calculation* (Figura 10).

Nome do Caso de Uso	Acessar Tela de Calculo
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
1. Usuário acessa a Tela de Cálculo	
	2. Verificar método
Exceções	
2. Verificar método	
2.1. Caso seja o método ABC, requisitar o caso de uso “Tela de Calculo ABC”.	
2.2. Caso seja o método Sebrae, requisitar o caso de uso “Tela de Calculo Sebrae”	
2.3. Caso seja o método Custo Pleno, requisitar o caso de uso “Tela de Calculo Custo Pleno”	

Quadro 11 - Descrição textual do caso de uso “Acessar Tela de Calculo”
Fonte: Autoria Própria.

As descrições textuais do Diagrama de Caso de Uso do Subsistema *Calculation* podem ser encontradas na íntegra no Apêndice G.

3.5 SUBFASE 4 – REFINAR CLASSES E DEFINIR SUBFRAMEWORKS

Após a análise dos subsistemas identificou-se que os mesmo podem ser desenvolvidos obedecendo às características de um *framework* e por isso podem ser classificados como *subframeworks*. O Quadro 12 ilustra os *subframeworks* identificados relatando as características que levam ser classificados como tal.

SUBFRAMEWORKS	**CARACTERÍSTICAS**
Activity	Ao analisar o subsistema Gerenciar Atividade identificou-se que o mesmo é composto de vários atributos tais como: Nome da Atividade, Linha de Produção e Custo Total, necessários para calcular o preço de venda. Considerando um sistema industrial em que se tem uma linha de produção este subsistema pode ser utilizado para calcular o tempo e o custo de cada processo. Desta forma, pode se acrescentar o atributo tempo para atender as necessidades da indústria, ou seja, este pode ser utilizado em aplicações diferentes, tanto para preço de venda quanto para gerenciamento de linha de produção.

Attributes	Analisando o subsistema Gerenciar Atributos, notou-se que o mesmo necessita de atributos para realizar o cálculo de suas despesas fixas e variáveis. Considerando um sistema empresarial em que se tem muitos gastos, este subsistema pode ser utilizado para o cadastramento de novos atributos, atendendo a necessidades desta empresa.
Calculation	Ao analisar o subsistema Cálculo do Preço de Venda, identificou-se que o mesmo é composto de várias fórmulas necessárias para calcular o preço de venda de um determinado produto, porém o mesmo pode ser utilizado para calcular notas de um sistema acadêmico.
FoodSystem	Levando em consideração um software para controle de campeonato de futebol, identificou-se que o mesmo necessita de uma entrada de dados, como por exemplo, os resultados de cada jogo. Desta maneira, verificou-se que o subsistema *FoodSystem* atende as exigências tanto para as entradas de dados dos métodos de preço de venda como para outros softwares que necessitem de alguma forma uma entrada de dados.
Product	Nota-se no subsistema Gerenciar Produtos que o mesmo é formado pelos seguintes atributos: Nome do Produto e Linha de Produção. Considerando um exemplo de uma empresa de tecidos em que se tem uma linha de produção, este subsistema pode ser utilizado para calcular o tempo e o custo de cada processo. Desta maneira, pode acrescentar um produto de acordo com a necessidade da empresa.
ProductionLine	Analisando o subsistema Gerenciar Linha de Produção, percebeu-se que o mesmo é formado apenas pelo Nome da Linha de Produção. Considerando o mesmo exemplo de um sistema industrial, este subsistema pode ser utilizado para calcular o tempo e o custo de cada processo.

Quadro 12 – Identificação dos *Subframeworks* e suas características
Fonte: Autoria Própria.

Esta Seção está dividida da seguinte forma. A Subseção 3.5.1 aborda o refinamento de Classes para o *Subframework Attributes.* A Subseção 3.5.2 destaca o refinamento de Classes para o *Subframework FoodSystem*. E por fim, a Subseção 3.5.3 descreve o refinamento de Classes para o *Subframework Calculation.*

3.5.1 Refinamento de Classes para o Subframework Attributes

Ao analisar os Diagramas de Classes UML descritos no trabalho de Crazuski *et al.* (2008), notou-se que todos os Diagramas de Classes para os subsistemas Gerenciar Atributo (ABC), Cadastrar Atributo (Sebrae-PR) e Cadastro de Atributos (Custo Pleno), são formados por 5 (cinco) classes cada um e são representados respectivamente nas Figuras 90, 91 e 92 que encontram-se no Anexo A e que não haviam sido construídas utilizando padrões de projeto (GAMMA *et al.*, 1994).

Por isso, para o desenvolvimento dos subsistemas propostos, foi necessário compreender alguns padrões, entre eles o "*Decorator"*, "*Abstract Factory"*, "*FactoryMethod"*, *"DAOFactory"* e "*Model-View-Controller (MVC)"*.

Após o estudo dos padrões de projeto e uma análise sobre o projeto a ser desenvolvido, foi observado que no Subsistema *Attributes* precisaria de 3 (três) telas de busca e 3 (três) telas de cadastro, porém a janela de busca era comum para os 3 (três) métodos de preço de venda, e as telas de cadastros tinham algo em comum, como por exemplo, um campo em que o usuário insere o nome do atributo para cadastrá-lo e também os botões "Salvar" e "Fechar". Foi então aplicado o padrão de projeto *"Decorator"* para agregar responsabilidades adicionais a um objeto dinamicamente. Classes decoradoras oferecem uma alternativa flexível ao uso de herança para estender uma funcionalidade (SILVA, 2010).

Quanto à parte lógica, foi desenvolvida seguindo o conceito do padrão de projeto MVC (*Model-view-controller*), o qual visa separar a lógica de negócio da lógica de apresentação, permitindo o desenvolvimento, teste e manutenção isolado de ambos.

Este padrão divide o sistema em 3 (três) camadas: *model* (modelo)*, view* (apresentação) e *controller* (controle)*.* No *model*, encontram-se as classes que compõem o domínio do sistema, sendo o núcleo da aplicação. Na *view*, encontram-se as classes que provêm as interfaces do sistema por meio das quais os usuários irão interagir com o *software.* No *controller*, encontram-se as classes que definem o modo como às interfaces do usuário reagem às entradas do mesmo, ou seja, o *controller* é responsável por gerenciar a interação entre a camada *view* e *model.*

Com o intuito de abstrair a camada de persistência, a fim de tornar possível a utilização de qualquer repositório de dados, foi utilizado o padrão *DAOFactory*. Com esse padrão, foi centralizado o serviço de persistência de objetos em um

conjunto de classes. A Figura 11 ilustra as camadas do projeto seguindo os padrões citados.

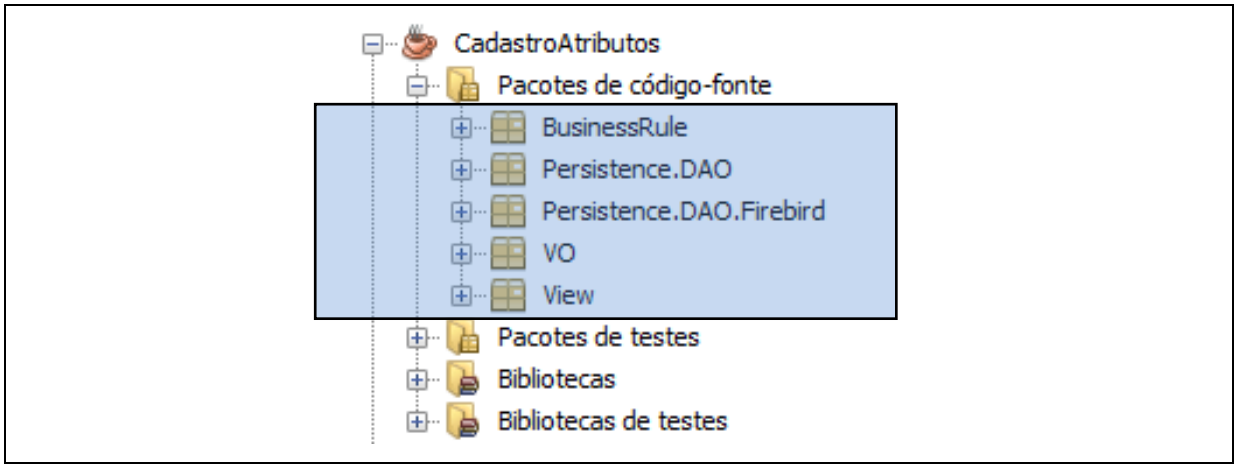


Figura 11 – Aplicação dos Padrões no Subsistema *Attributes*
Fonte: Autoria Própria.

Conforme mencionado anteriormente, percebeu-se que existiam algo em comum entre as telas, possibilitando assim o uso do padrão de projeto *"Decorator"*. As Figuras 12 e 13 ilustram os diagramas de classes das telas de cadastro e de busca, respectivamente. A Figura 12 contém a classe *WindowAdd* a qual define a *interface* para objetos que podem ter responsabilidades acrescentadas a eles dinamicamente, *CommonWindowAdd* define um objeto para o qual as responsabilidades adicionais podem ser atribuídas, *DecoratorAdd* mantém referência para o objeto *WindowAdd*. Por fim, *WindowAddAbc, WindowAddSebrae* e *WindowAddFullCost* acrescentam responsabilidades específicas a cada método.

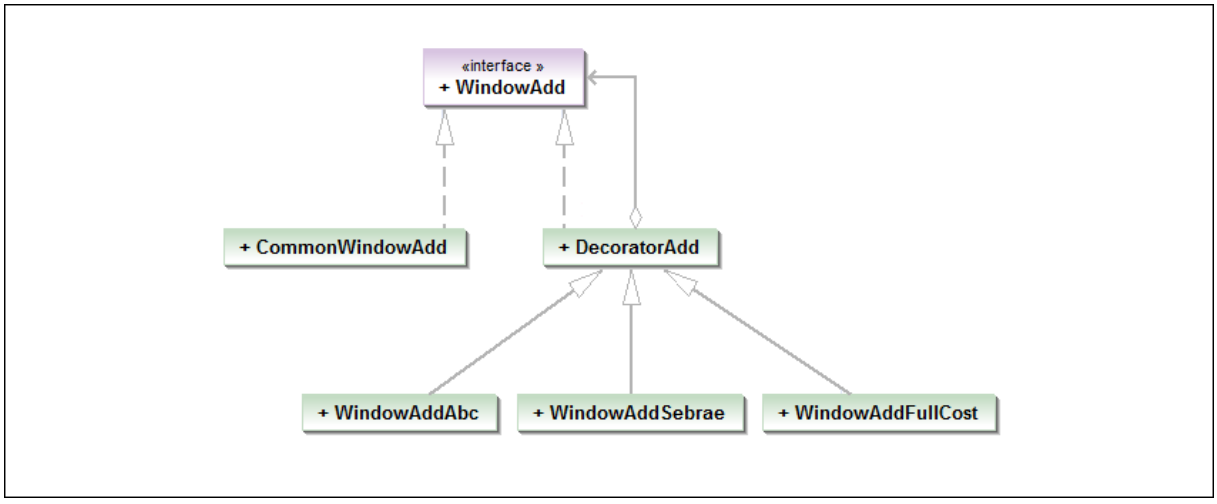


Figura 12 - Diagrama de Classe das Telas de Cadastro de Atributos
Fonte: Autoria Própria.

A Figura 13 utiliza a mesma concepção implementada no cadastro de atributos (Figura 12).

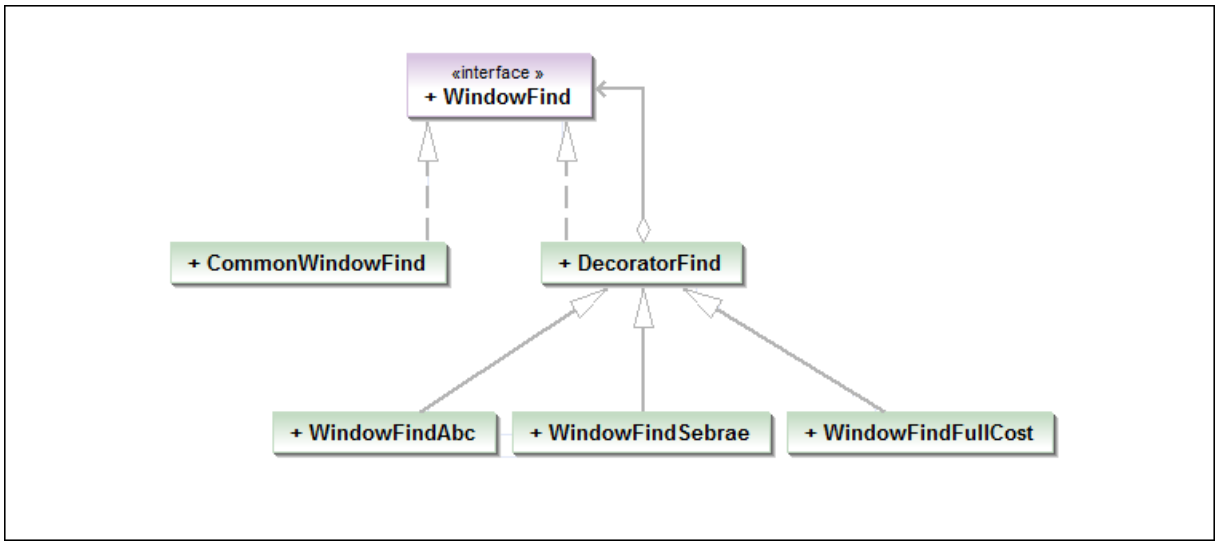


Figura 13 – Diagrama de Classe das Telas de Busca de Atributos
Fonte: Autoria Própria.

A Figura 14 ilustra o pacote *View* do Subsistema *Attributes*, no qual é identifica-se as *interfaces* pelos seus estereótipos como já citadas nas Figuras 12 e 13. Além das classes pertencentes a este pacote, também encontram-se neste diagrama algumas classes pertencentes a outro pacote, tais como: *Attribute, AbcAttribute, FullCostAttribute, SebraeAttribute, Validates, ValidatesAbc, ValidatesSebrae, ValidatesFullCost, AbcAttributeRN, SebraeAttributeRN, FullCostAttributeRN e Persistence.DAO.* Estas classes estão ilustradas neste diagrama para mostrar as dependências das classes do pacote *View*.

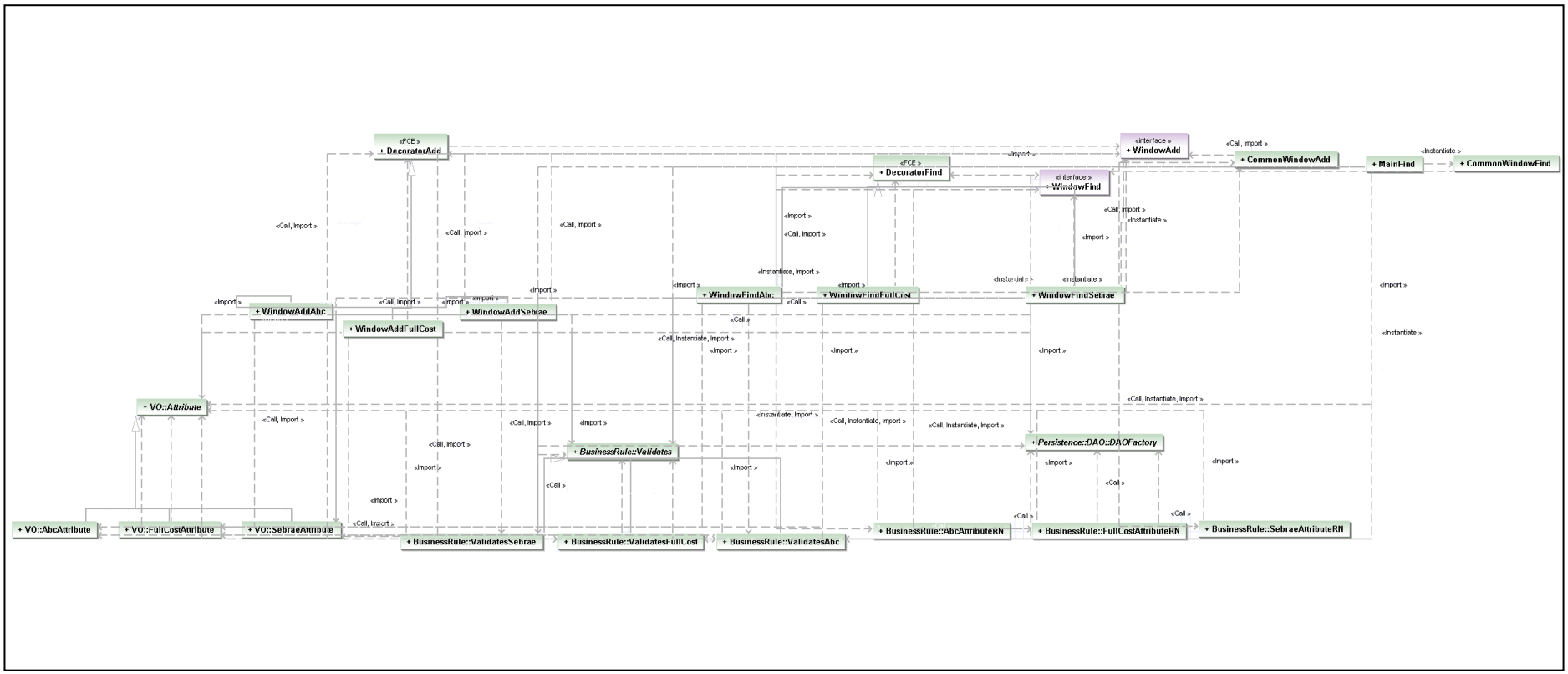


Figura 14 – Pacote *View* do Subsistema *Attributes*
Fonte: Autoria Própria.

No pacote VO existe uma herança entre as classes *Attribute, AttributeAbc, AttributeSebrae* e *AttributeFullCost,* em que a classe *Attribute* terá todos os atributos e métodos comuns entres classes e cada subclasse terá apenas os atributos e métodos específicos. O Diagrama de Classe é ilustrado pela Figura 15.

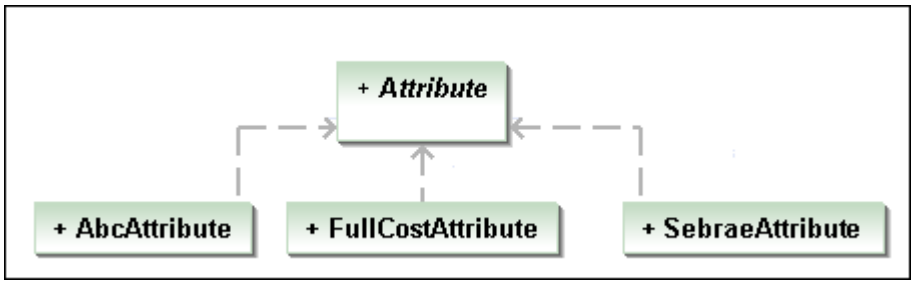


Figura 15 – Pacote VO do Subsistema *Attributes*
Fonte: Autoria Própria.

No pacote *Persistence.DAO* localiza-se a classe *DAOFactory*, a qual tem a responsabilidade de instanciar a classe correta para cada método requisitado. Também neste pacote situam-se as *interfaces FullCostAttributeDAO, AbcAttributeDAO* e *SebraeAttributeDAO,* estas recebem as assinaturas dos métodos necessários, tais como: cadastro, edição, remoção, entre outros. A Figura 16 ilustra o pacote *Persistence.DAO*.

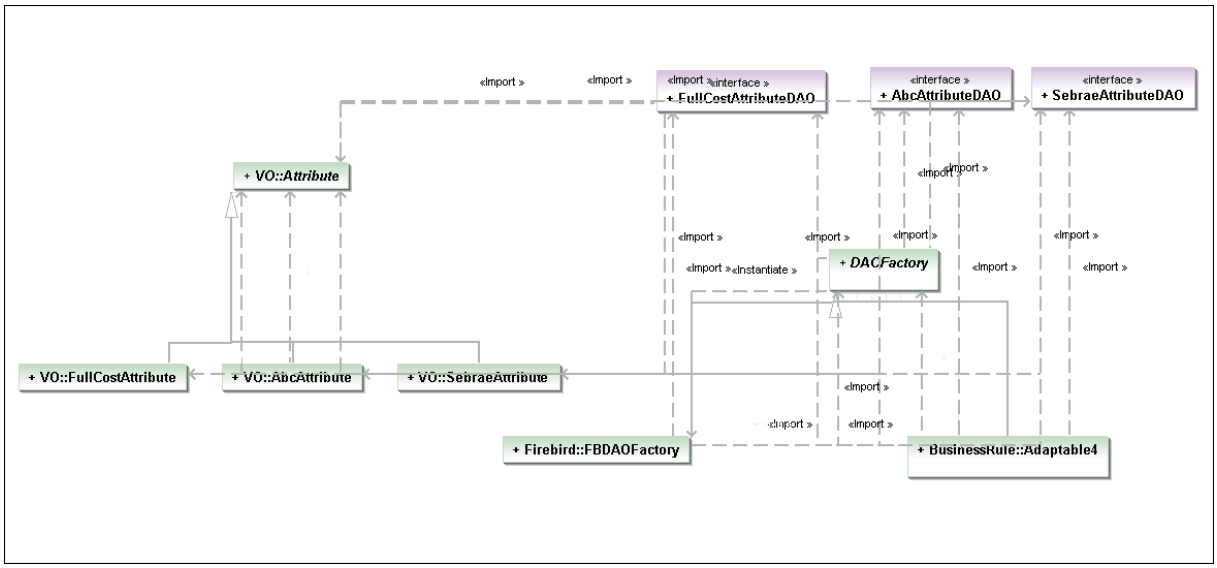


Figura 16 – Pacote Persistence.DAO do Subsistema *Attributes*
Fonte: Autoria Própria.

A Figura 17 ilustra o pacote *BusinessRule*, em que as classes *ValidatesSebrae, ValidatesFullCost* e *ValidatesAbc* herdam as características da classe *Validates* e tem como função validar os dados dos métodos Sebrae, Custo Pleno e ABC, respectivamente. Situam-se também neste pacote as classes *AbcAttributeRN, SebraeAttributeRN* e *FullCostAttributeRN,* as quais são responsáveis pelas regras de negócio de cada método de preço de venda.

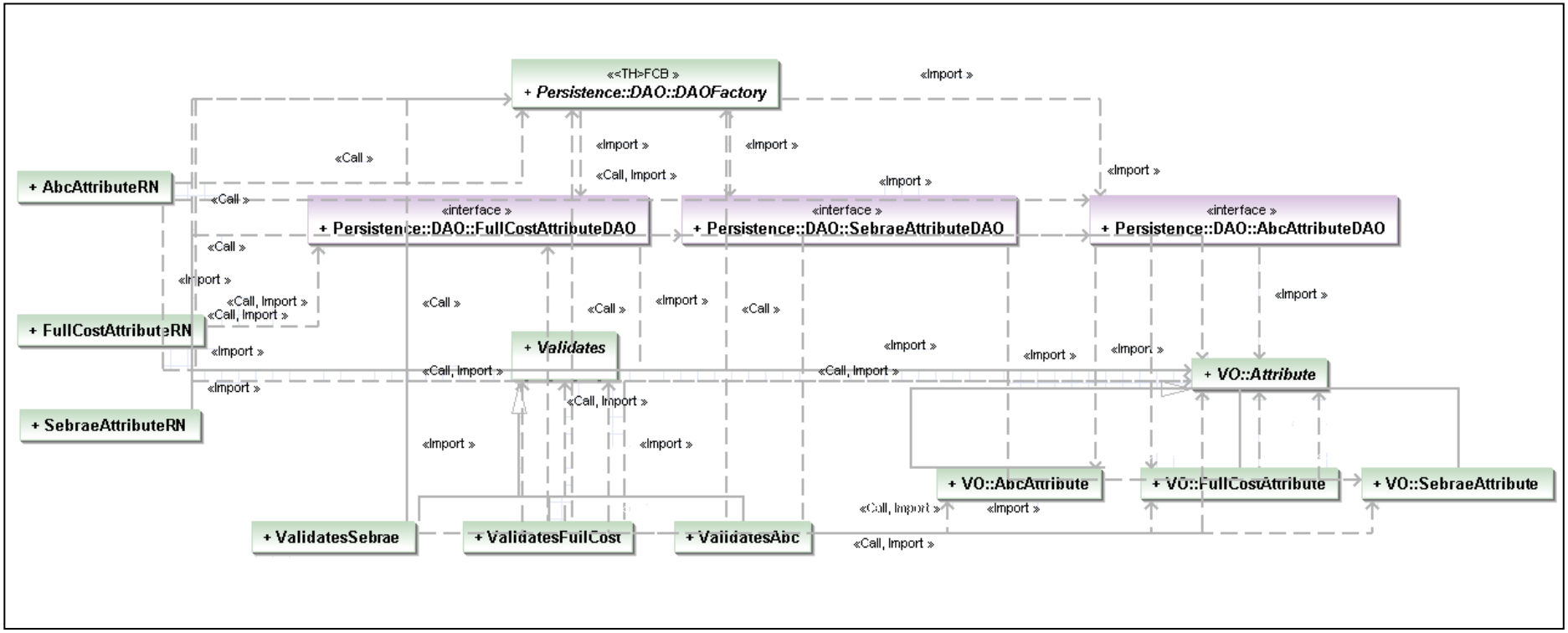


Figura 17 – Pacote BusinessRule do Subsistema *Attributes*
Fonte: Autoria Própria.

O pacote *Persistence.DAO.Firebird,* ilustrado pela Figura 18, contém a classe *FBDAOFactory*, a qual tem a responsabilidade de fazer a conexão ao Banco de Dados. As classes *AbcAttributeDAOFirebird, FullCostAttributeDAOFirebird* e *SebraeAttributeDAOFirebird* implementam todos os métodos das *interfaces* ilustrados pelas classes com estereótipos *<<interface>>*. As *interfaces* e as classes *DAOFactory, Attribute, AbcAttribute, SebraeAttribute* e *FullCostAttribute* não pertencem a este pacote, mas estão ilustradas no mesmo para mostrar as dependências existentes com as classes existentes.

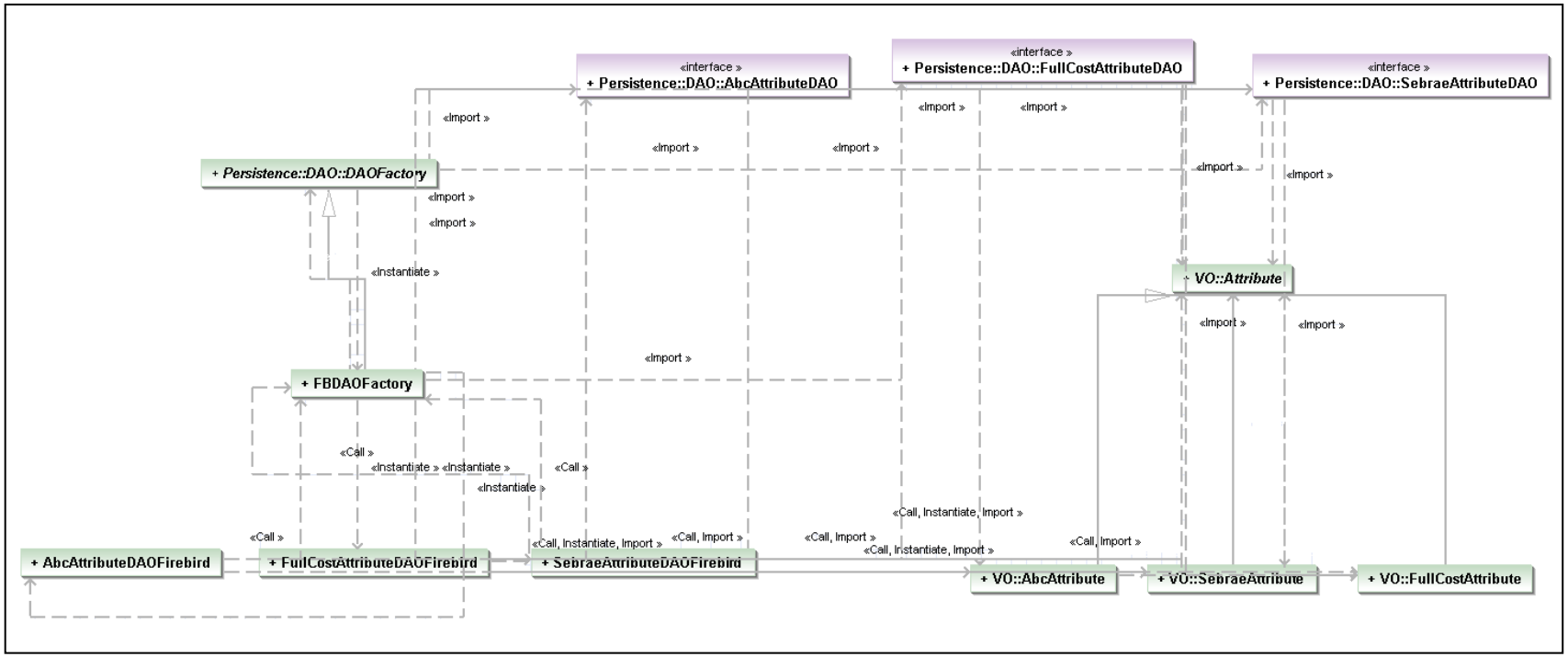


Figura 18 – Pacote *Persistence.DAO.Firebird* do Subsistema *Attributes*
Fonte: Autoria Própria.

3.5.2 Refinamento de Classes para o Subframework FoodSystem

Após a análise dos Diagramas de Classes UML descritas no trabalho de Crazuski *et al.* (2008), notou-se que apenas o método Sebrae, obtinha um alimentar sistemas, em todos os outros métodos, o sistema era alimentado durante o cadastro ou no momento de realizar o cálculo de preço de venda.

Observou-se então que o Diagrama de Classes do método Sebrae era formado apenas por 4 (quatro) classes e é ilustrado na Figura 93 do Anexo B. Partindo da análise realizada neste diagrama de classes, foi elaborado o Subsistema *FoodSystem* não só para o método Sebrae, como no trabalho de Crazuski *et al.* (2008), mas para todos os métodos contidos no *framework*.

Para o desenvolvimento do mesmo aplicou-se alguns padrões de projeto, tais como: *Decorator*, *Abstract Factory*, *FactoryMethod*, *DAOFactory* e *MVC (Model-view-controller)* com o intuito de deixar o sistema mais flexível e, consequentemente, obter uma melhora na questão do reúso do mesmo, facilitando também a manutenibilidade.

Em primeira análise, notou-se que o Subsistema *FoodSystem*, necessitaria de 3 (três) telas de entrada de valores para cumprir a sua função, cada janela correspondia a um método específico (ABC, Sebrae e Custo Pleno). Porém, observou-se que continham algo em comum entre eles, podendo assim aplicar o padrão de projeto "*Decorator*".

Como no Subsistema *Attributes*, desenvolveu-se a parte lógica do Subsistema *FoodSystem* seguindo o padrão de projeto MVC (*Model-view-controller*), separando a lógica de negócio da lógica de apresentação. Além disso, utilizou-se o padrão de projeto *DAOFactory* na camada de persistência. A Figura 19 ilustra as camadas do Subsistema *FoodSystem.*

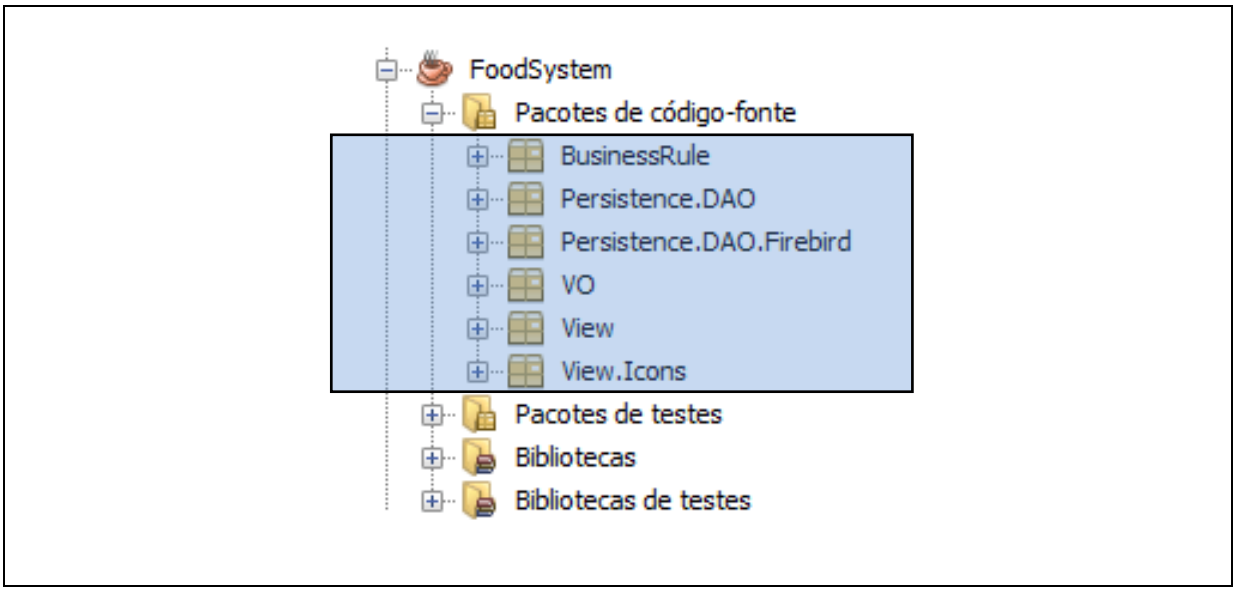


Figura 19 – Aplicação dos Padrões de Projeto no Subsistema *FoodSystem*
Fonte: Autoria Própria.

A Figura 20 ilustra o Diagrama de Classes das Telas *FoodSystem*, na qual observou-se que havia algo em comum entre as telas de cada método e foi possível a utilização do padrão de projeto *Decorator*.

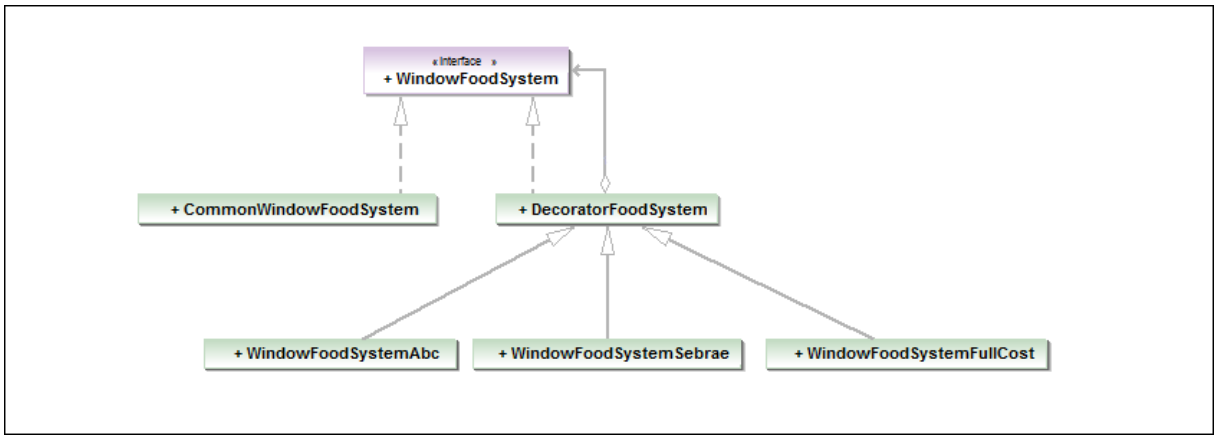


Figura 20 – Diagrama de Classe das Telas de Alimentação do Sistema
Fonte: Autoria Própria.

A Figura 21 ilustra o pacote *View* do Subsistema *Calculation*, o qual contém as classes *WindowFoodSystem, CommonWndowFoodSystem, DecoratorFoodSystem, WindowFoodSystemAbc, WindowFoodSystemSebrae* e *WindowFoodSystemFullCost*. A classe *WindowFoodSystem* define a interface para objetos que pode ter responsabilidades acrescentadas dinamicamente. A classe *CommonWindowFoodSystem* define um objeto para o qual as reponsabilidades adicionais podem ser atribuídas, a classe *DecoratorFoodSystem* mantém referência

ao objeto *WindowFoodSystem* e, por fim, as classes *WindowFoodSystemAbc*, *WindowFoodSystemSebrae* e *WindowFoodSystemFullCost* acrescentam responsabilidades específicas a cada método.

Outras classes existentes neste diagrama apenas apresentam as dependências com as classes que pertençam ao mesmo.

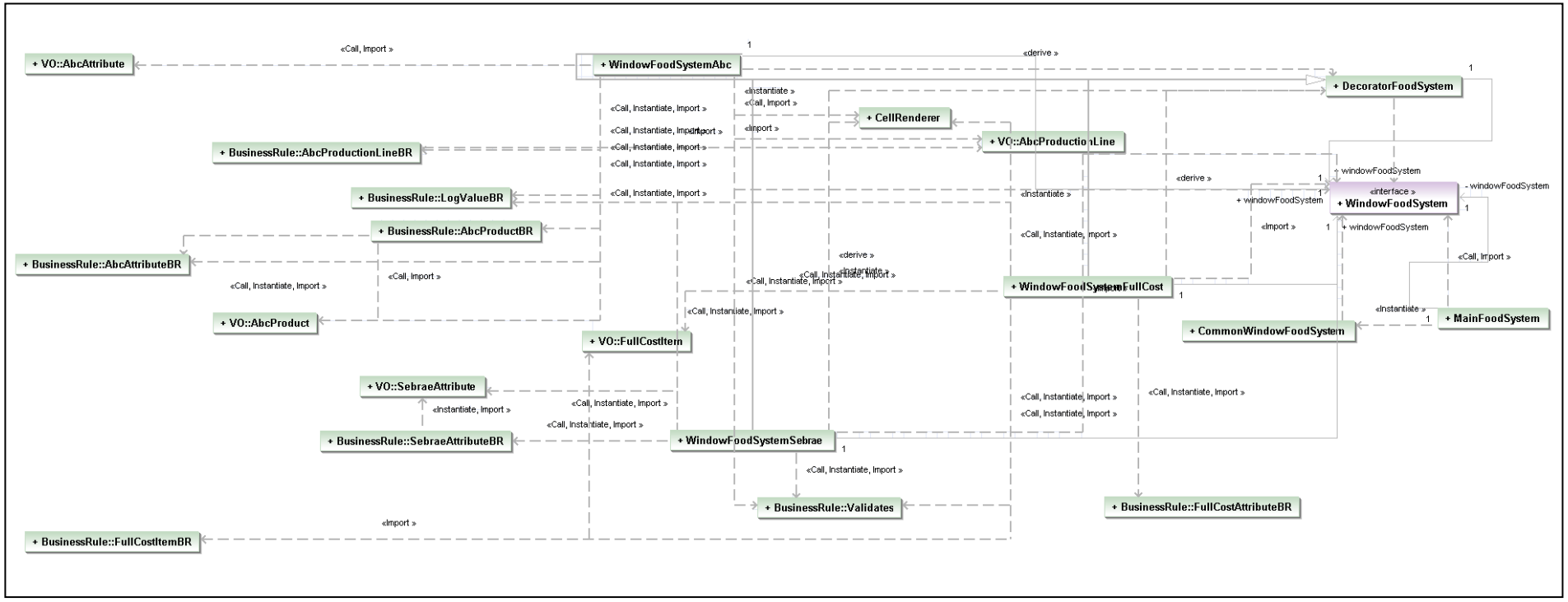


Figura 21 – Pacote *View* do Subsistema *FoodSystem*
Fonte: Autoria Própria.

O pacote VO, ilustrado pela Figura 22, apresenta as sete classes deste pacote, das quais apenas três delas apresentam ligações, são elas: *AbcAttribute*, *SebraeAttribute* e *Attribute*, esta última mostra os atributos e os métodos comuns das duas primeiras. As classes *AbcProductionLine, FullCostItem, AbcProduct* e *LogValue* referem-se aos subsistemas *Production Line* (Linha de Produção) e *Product* (Produto). Estas classes não possuem ligações, pois não possuem dependências com as demais classes, pois representam as classes que estão no pacote VO. Ressalta-se que outras classes dependem delas, porém, elas não dependem de ninguém.

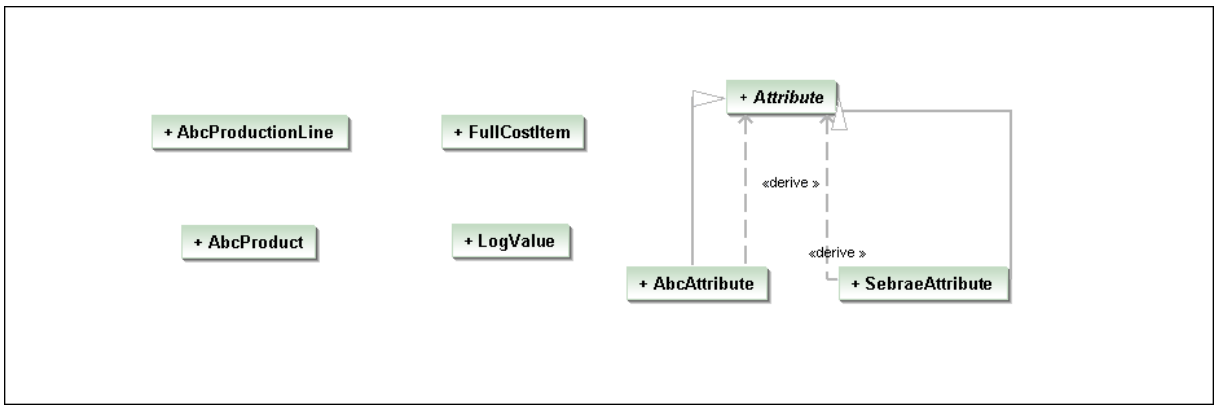


Figura 22 – Pacote VO do Subsistema *FoodSystem*
Fonte: Autoria Própria.

A Figura 23 ilustra o pacote *Persistence.DAO* do subsistema *FoodSystem*, o qual possui as *interfaces AbcAttributeDAO, SebraeAttributeDAO, FullCostAttributeDAO, LogValuesDAO, FullCostItem, AbcProductDAO* e *AbcProductionLineDAO,* todas estas identificadas pelo seu estereótipo. Também neste pacote situa-se a classe *DAOFactory* a qual quando requisitada é responsável pela instanciação da classe correta. As classes *AbcProductionLine, FullCostItem, SebraeAttribute* encontram-se ilustradas neste pacote apenas pela questão de dependência entre elas e as classes citadas anteriormente.

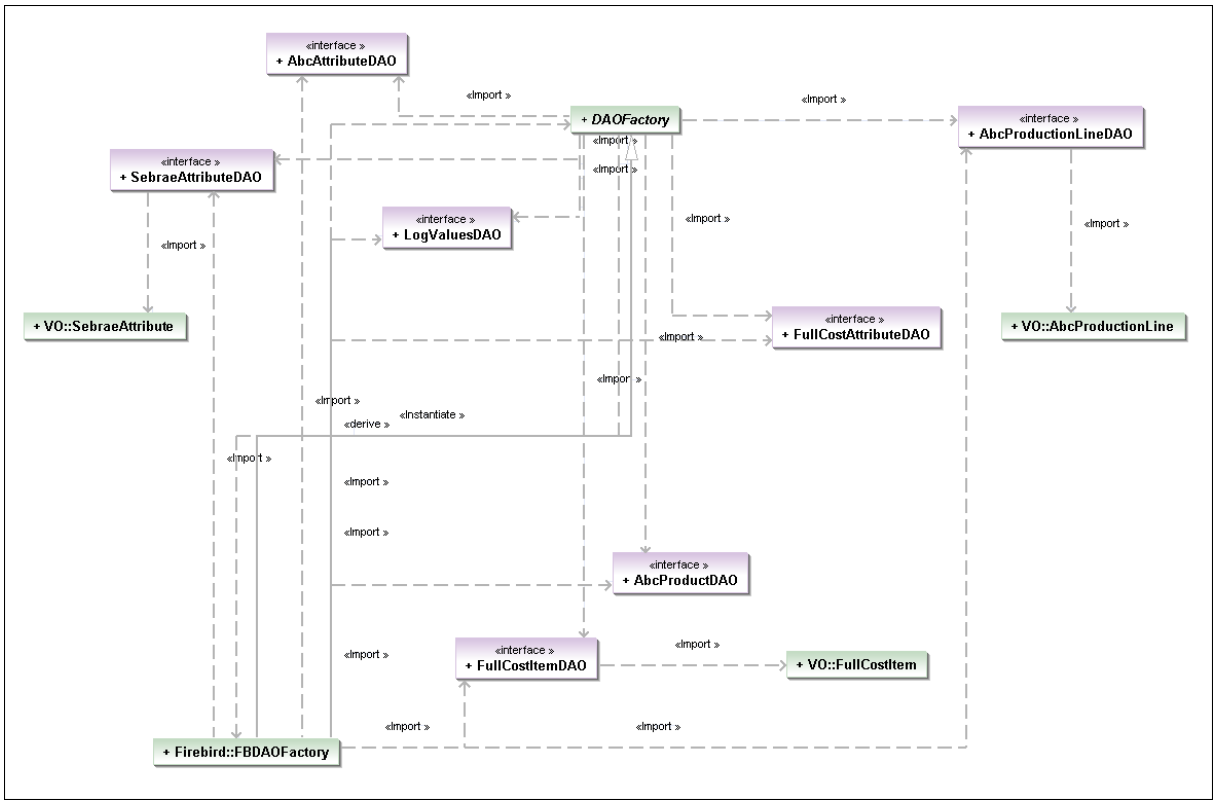


Figura 23 – Pacote *Persistence.DAO* do Subsistema *FoodSystem*
Fonte: Autoria Própria.

As classes *AbcActivityBR, AbcAttributeBR, AbcProductBR, AbcProductionLineBR, FullCostAttributeBR, FullCostItemBR, LogValueBR, SebraeAttributeBR* e *Validates* pertencem ao pacote *BusinessRule*. Estas classes têm a função de interligar a parte de interação do usuário com o banco de dados, ou seja, ocupam-se com a parte da regra de negócio do Subsistema *FoodSystem*. Este pacote é ilustrado na Figura 24.

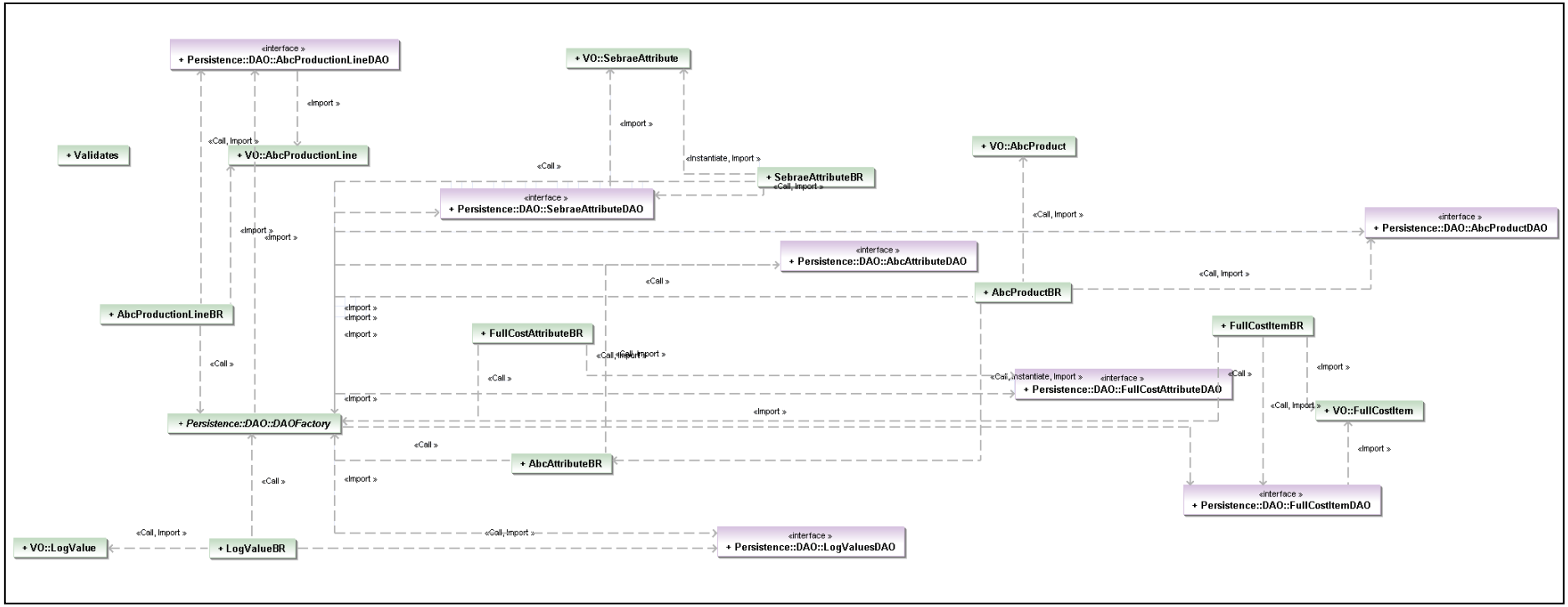


Figura 24 – Pacote *BusinessRule* do Subsistema *FoodSystem*
Fonte: Autoria Própria.

O Diagrama de Classes ilustrado pela Figura 25, ilustra o pacote *Persistence.DAO.Firebird*, o qual contém as classes *AbcActivityDAOFirebird, AbcAttributeDAOFirebird, AbcProductDAOFirebird, AbcProductionLineDAOFirebird, FullCostAttributeDAOFirebird, FullCostItemDAOFirebird, LogValuesDAOFirebird, SebraeAttributeDAOFirebird* e, a mais importante delas, *FBDAOFactory*, pois é responsável pela conexão ao Banco de Dados.

Todas estas classes com exceção da classe *FBDAOFactory* implementam a *interface* correspondente a cada classe do pacote *Persistence.DAO,* e tem alguma dependência com outra classe de outro pacote, ilustrada também no Diagrama.

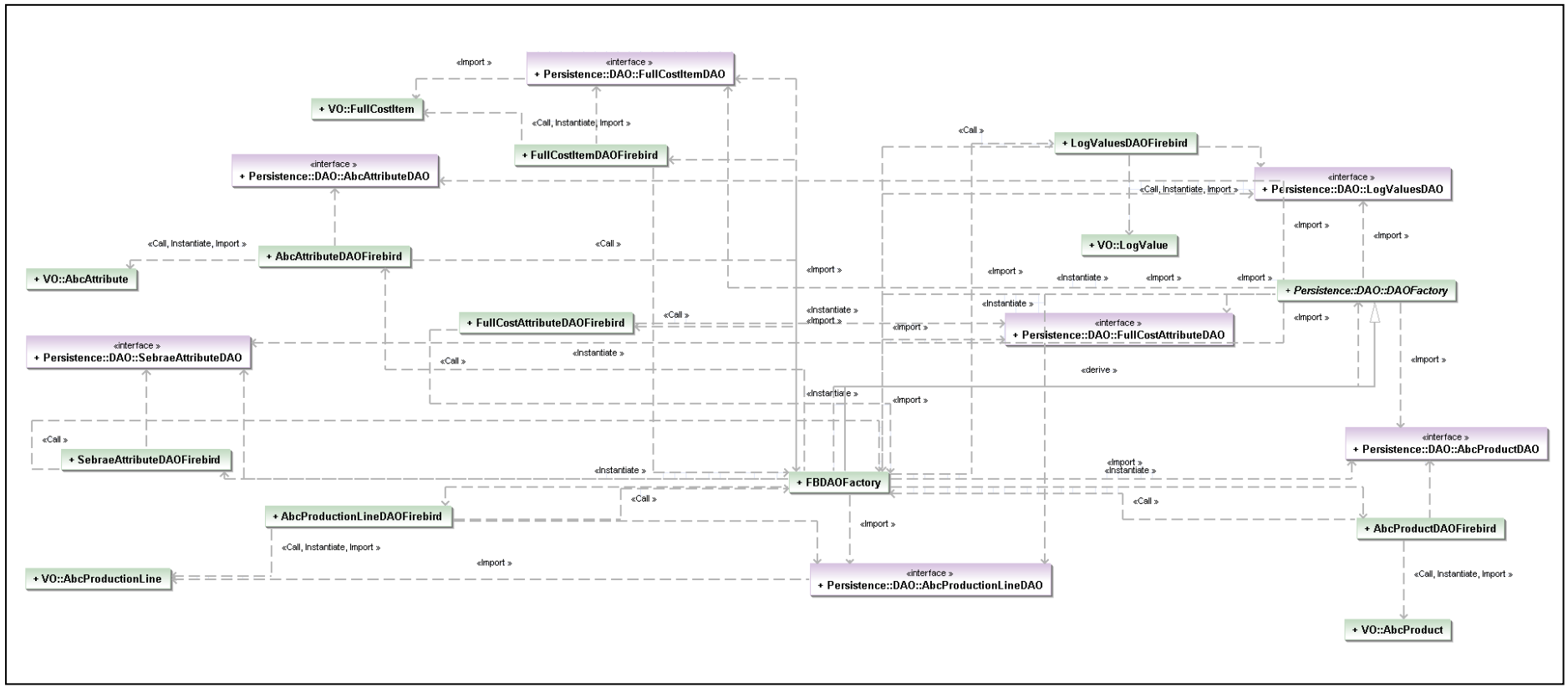


Figura 25 – Pacote *Persistence.DAO.Firebird* do Subsistema *FoodSystem*
Fonte: Autoria Própria.

3.5.3 Refinamento de Classes para o Subframework Calculation

Para o refinamento de Classes do Subsistema *Calculation*, primeiramente analisou-se os Diagramas de Classes UML do trabalho de Crazuski *et al.* (2008) e constatou-se que o Subsistema Cálculo do Preço de Venda (ABC) é formado por 2 (duas) classes, o Subsistema Calcular Preço de Venda (Sebrae-PR) é formado por 4 (quatro) classes e o Subsistema Cálculo do Preço de Venda (Custo Pleno) é formado por 2 (duas) classes. As Figuras 94, 95 e 96, contidas no Anexo C, representam cada um destes, respectivamente.

Notou-se que poderia ser aplicado o padrão de projeto *Decorator*, pois as telas dos 3 (três) métodos apresentavam algo em comum, tais como: campo exibindo a fórmula do método, campo para exibição do resultado final do cálculo e botão “Gravar”.

O conceito do padrão de projeto MVC (*Model-view-controller*) foi aplicado à parte lógica, para melhorar a questão de desenvolvimento, teste e manutenção.

Para a conexão ao banco de dados, foi aplicado o padrão de projeto *DAOFactory*, com o intuito de centralizar o serviço de persistência de objetos em um pequeno conjunto de classes. Com isso, o subsistema ficou dividido em 4 (quatro) pacotes: *View, VO, BusinessRule* e *Persistence.* A Figura 26 ilustra os pacotes do projeto.

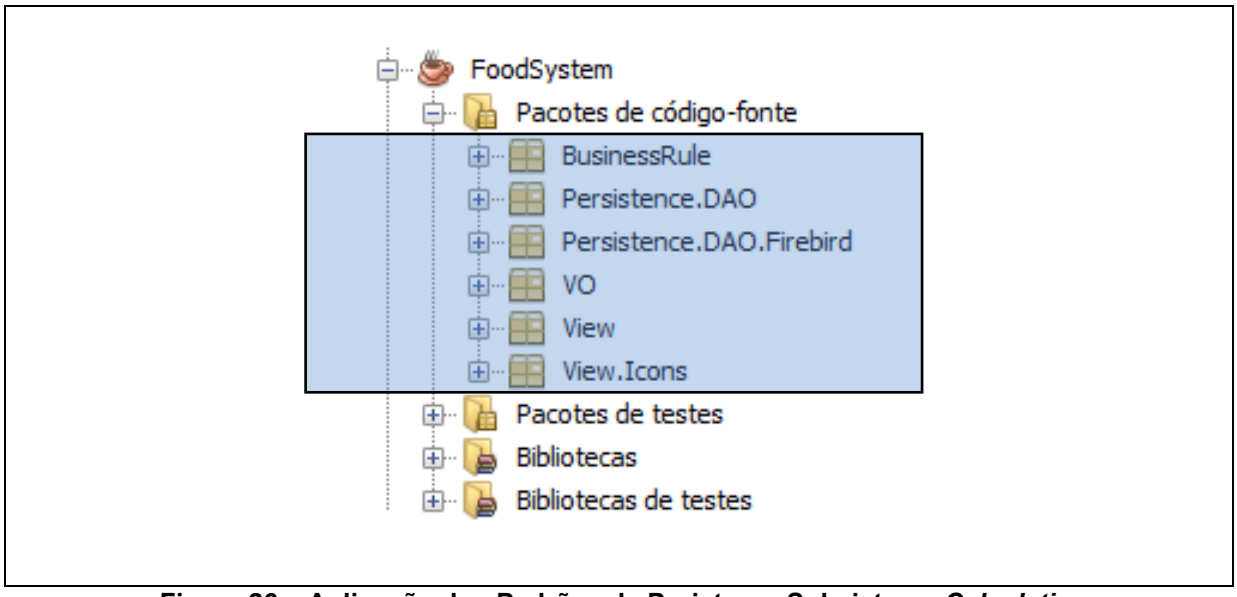


Figura 26 – Aplicação dos Padrões de Projeto no Subsistema *Calculation*
Fonte: Autoria Própria.

A Figura 27 ilustra o Diagrama de Classes das telas de Cálculo, no qual foi utilizado o padrão de projeto *Decorator*.

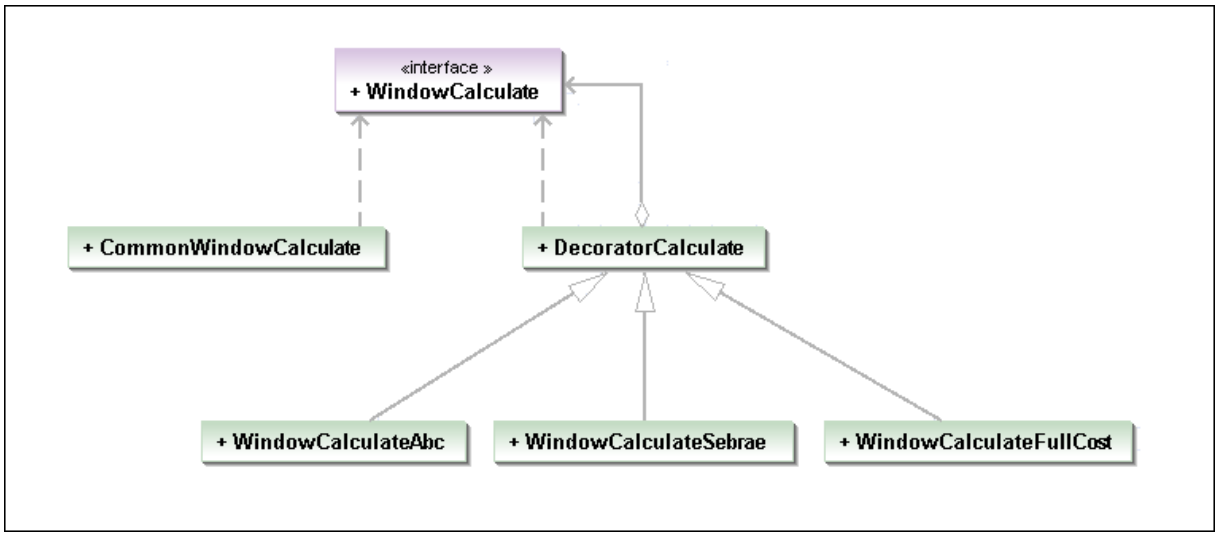


Figura 27 – Diagrama de Classe das Telas de Cálculo
Fonte: Autoria Própria.

A Figura 28 ilustra o pacote VO o qual possui as seguintes classes: *Attribute, AbcAttribute, SebraeAttribute, FullCostAttributeValue, AbcProductionLine, FullCostItem, LogValue, AbcProduct.* Das quais, *AbcAttribute* e *SebraeAttribute* herdam as características de *Attribute*. As demais classes não contem ligações, pois não possuem dependências com outras classes nem com outros pacotes.

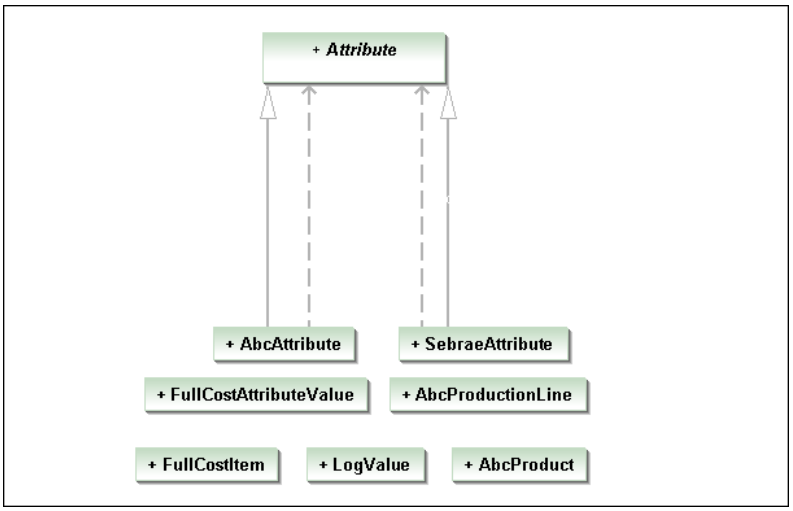


Figura 28 – Pacote VO do Subsistema *Calculation*
Fonte: Autoria Própria.

As classes *DecoratorCalculate, CommonWindowCalculate, WindowCalculateAbc, WindowCalculateFullCost, WindowCalculateSebrae, CellRenderer*, e a *interface WindowCalculate,* identificada pelo estereótipo *<<interface>>,* pertencem ao pacote *View* ,ilustrado pela Figura 29. Estas classes foram implementadas seguindo o padrão de projeto *Decorator* e suas descrições seguem as mesmas já relatadas no Subsistema *Attributes* e *FoodSystem.*

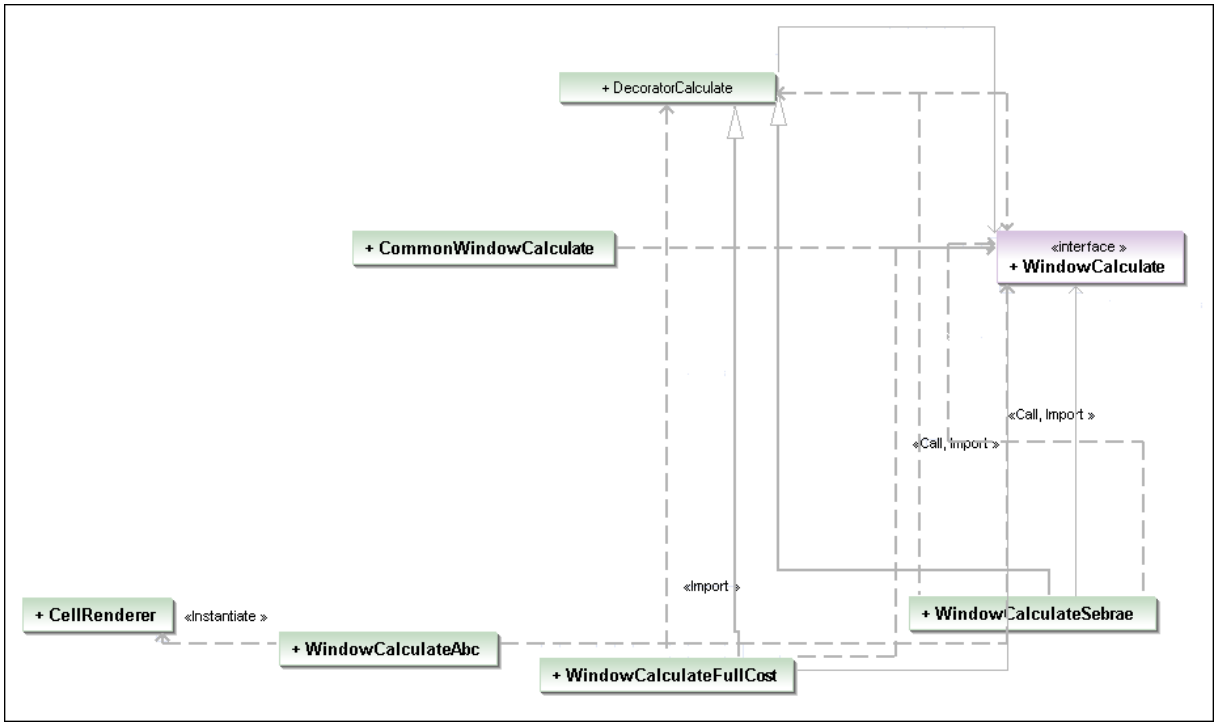


Figura 29 – Pacote View do Subsistema *Calculation*
Fonte: Autoria Própria.

O pacote *BusinessRule*, ilustrado pela Figura 30, possui as seguintes classes: *AbcAttributeBR, FullCostAttributeBR, SebraeAttributeBR, AbcProductionLine, Validates, FullCostItemBR, LogValueBR, AbcActivityBR, AbcProductBR*, as quais possuem uma relação de dependência com algumas classes do pacote *Persistence*. Para evitar a “poluição” deste diagrama, optou-se por demonstrar as classes externas apenas ilustrando o pacote *Persistence*.

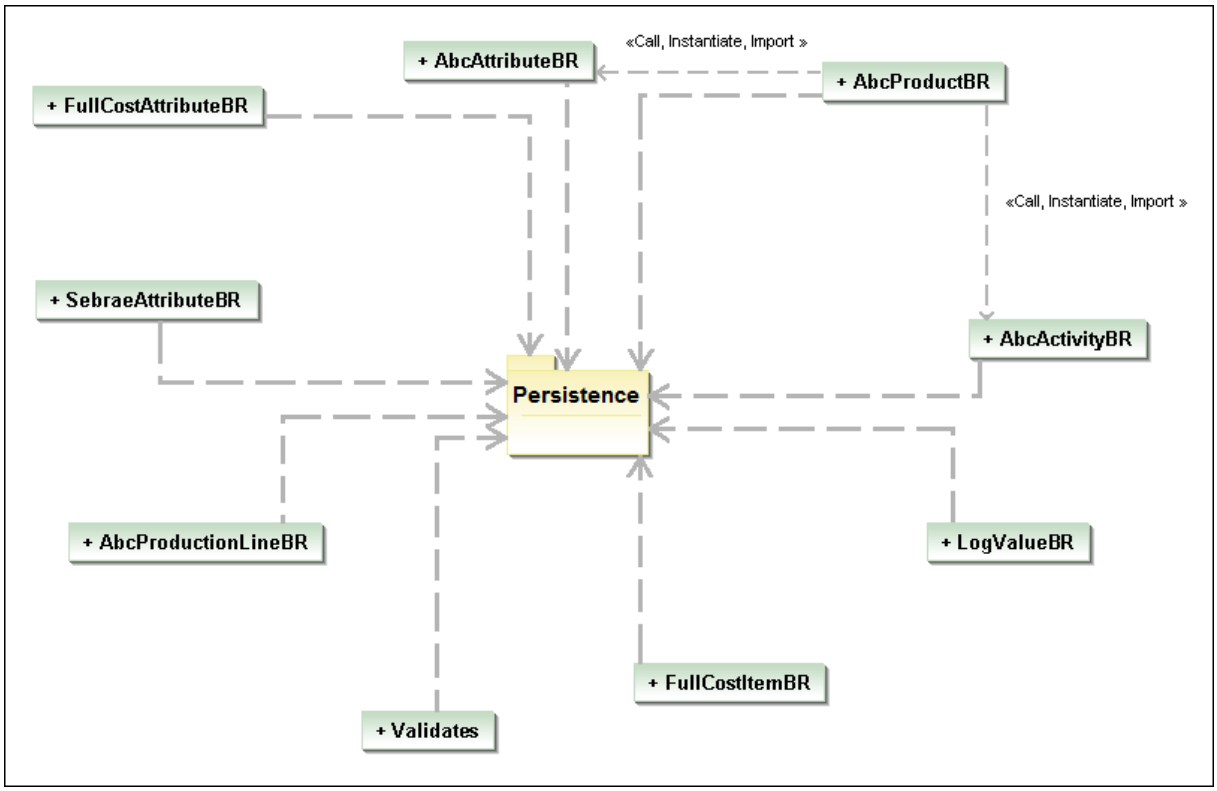


Figura 30 – Pacote *BusinessRule* do Subsistema *Calculation*
Fonte: Autoria Própria.

A Figura 31 ilustra o pacote *Persistence.DAO.* Este pacote é formado apenas por *interfaces*, com exceção da classe *DAOFactory.* As *interfaces* presentes neste diagrama possuem assinaturas dos métodos necessários para a realização de buscas, cadastros, edições e outras ações para o Subsistema *Calculation* envolvidas com Banco de Dados.

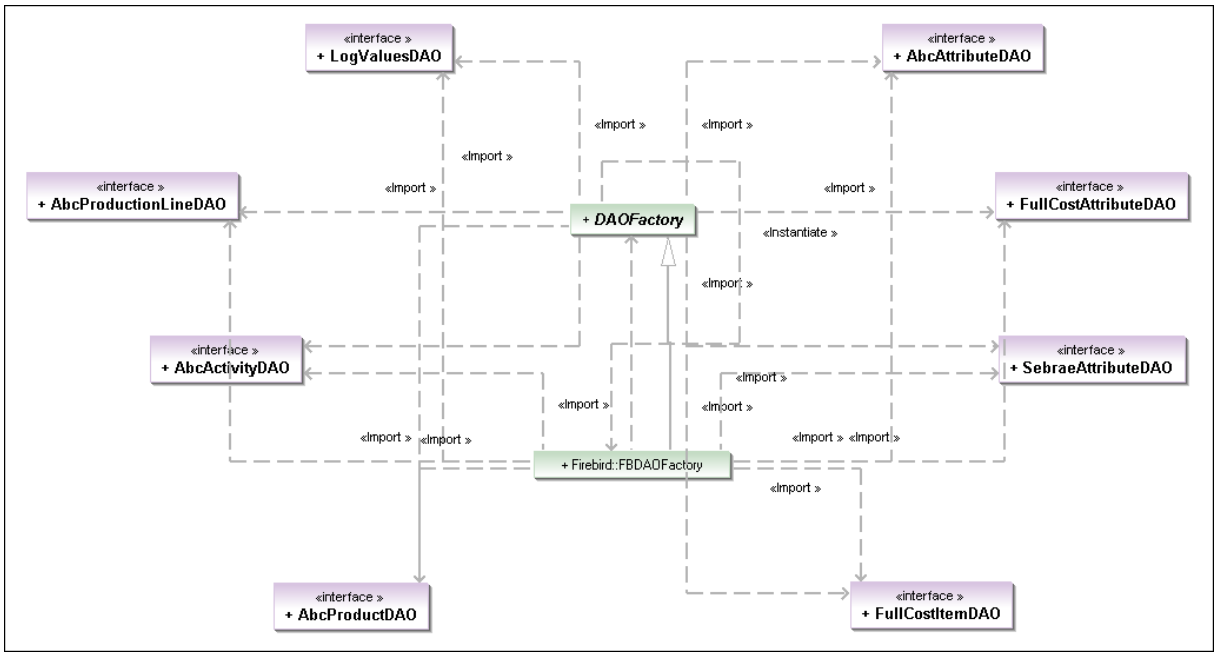


Figura 31 – Pacote *Persistence.DAO* do Subsistema *Calculation*
Fonte: Autoria Própria.

As classes *AbcActivityDAOFirebird, AbcProductionLineDAOFirebird, AbcProductDAOFirebird, AbcAttributeDAOFirebird, FullCostAttributeDAOFirebird, SebraeAttributeDAOFirebird, FullCostItemDAOFirebird* e *LogValuesDAOFirebird* contidas no pacote *Persistence.DAO.Firebird,* ilustrado na Figura 32, implementam as *interfaces* contidas no pacote *Persistence.DAO* (Figura 31), por este motivo as *interfaces* também estão demonstradas neste mesmo diagrama. A classe *FBDAOFirebird*, encontrada no pacote *Persistence.DAO.Firebird* tem a responsabilidade de efetuar a conexão ao Banco de Dados.

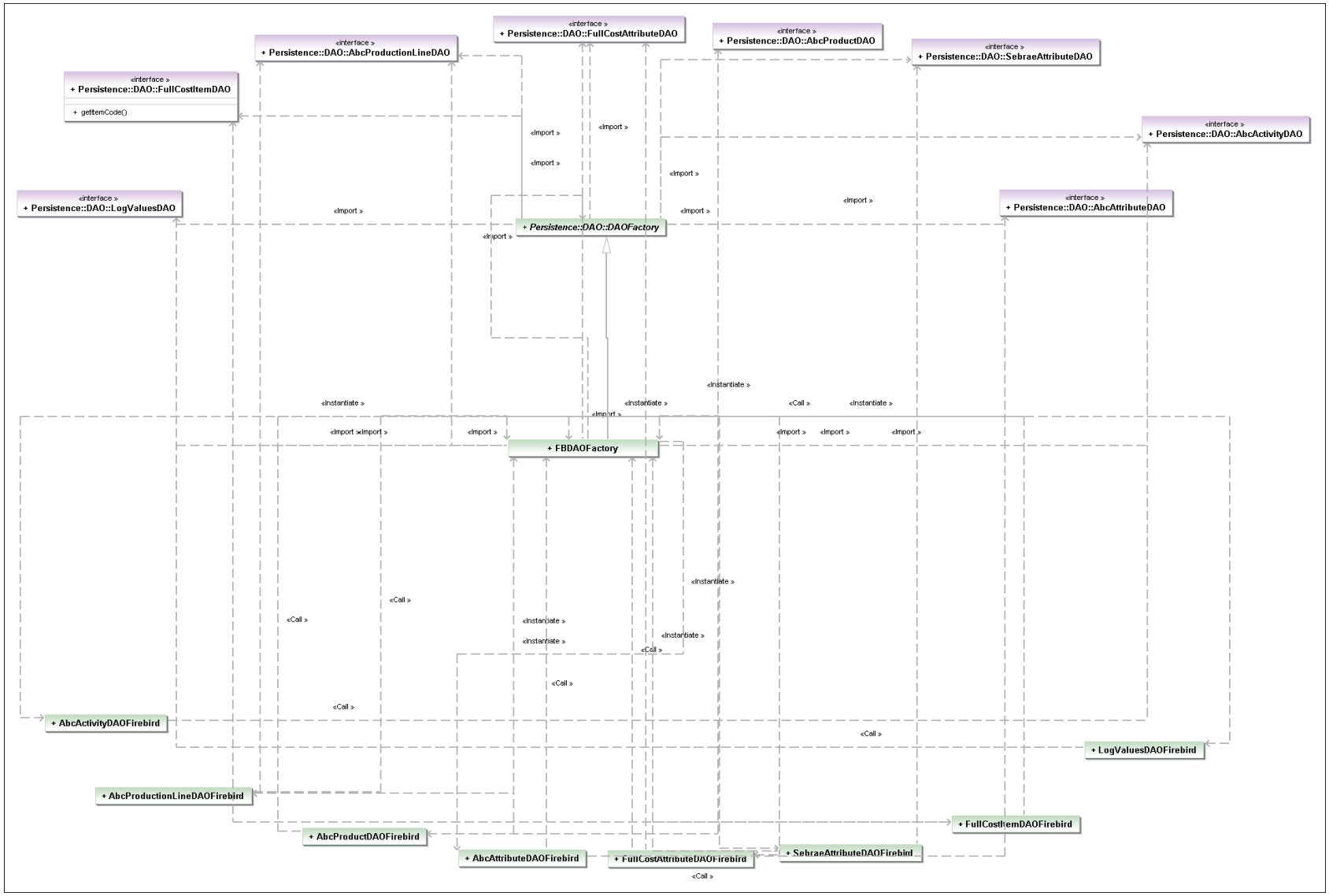


Figura 32 – Pacote *Persistence.DAO.Firebird* do Subsistema *Calculation*
Fonte: Autoria Própria.

3.6 SUBFASE 5 – DEFINIR FRAMEWORK BASE E DE APLICAÇÃO

Segundo Pree (1999), para o desenvolvimento de um *framework* de domínio é necessário identificar os pontos de estabilidade e flexibilidade. O primeiro refere-se aos aspectos comuns a todas as aplicações do domínio, já o segundo são os aspectos específicos a cada aplicação-exemplo. Mais informações sobre o método que é utilizado para identificação destes pontos pode ser obtidas no trabalho de Matos (2008).

Após a análise das responsabilidades de cada um dos casos de uso em todos os métodos de formação de preço de venda, a arquitetura geral do *framework* (Figura 33) foi dividida em dois pacotes principais: *Base* e *Specific*, os quais contêm as classes que são comuns – *frozen spots* – e específicas – *hot spots* –, respectivamente.

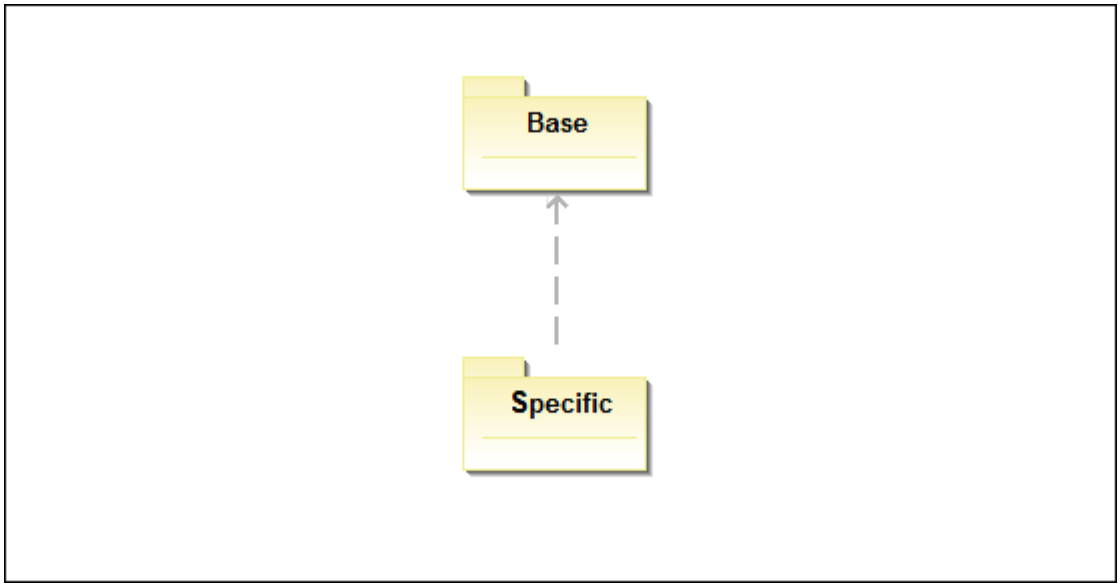


Figura 33 – Arquitetura geral do *framework*
Fonte: Autoria Própria.

Há uma relação de dependência entre o pacote "*Specific*" e o pacote "*Base*", pois todos os pontos específicos do *framework* necessitam dos pontos comuns para seu correto funcionamento.

No interior do pacote "*Base*", partes comuns do *framework* – *frozen spots* –, verificou-se a necessidade de outros 2 (dois) novos pacotes: *Attributes* e *FoodSystem* (Figura 34). O pacote *Attributes* é responsável por todo o gerenciamento dos atributos utilizados nos métodos de custeio e o pacote

FoodSystem tem como função gerenciar a entrada de dados que alimenta o sistema a fim de efetuar os cálculos necessários para estabelecer o preço de venda do produto.

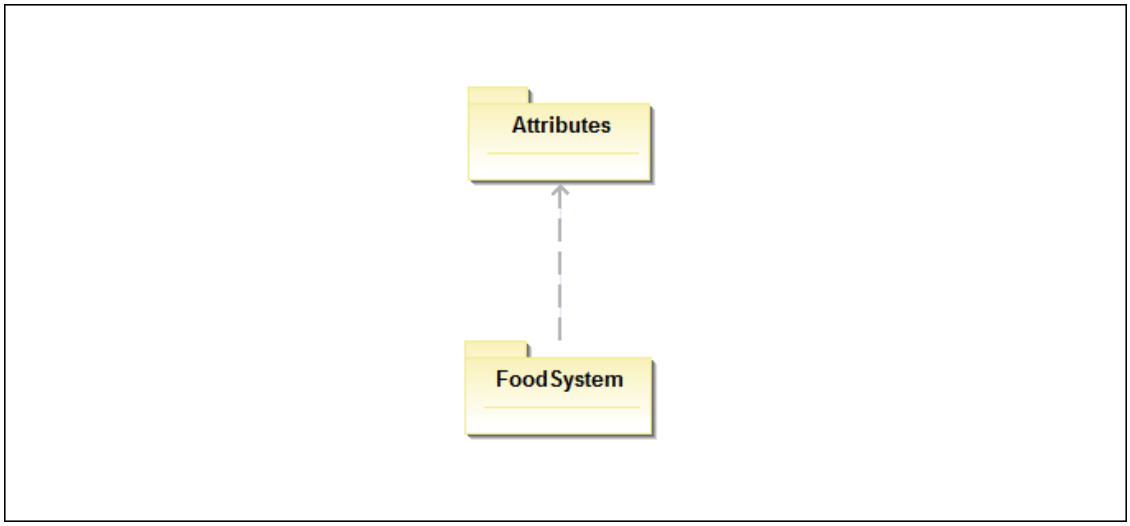


Figura 34 – Pacote *Base*
Fonte: Autoria Própria.

Para alimentar o sistema, ou seja, atribuir valores aos atributos, é necessário haver atributos cadastrados, por este motivo há uma relação de dependência entre o pacote *FoodSystem* e o pacote *Attributes.*

As partes específicas, *hot spots*, de todos os *subframeworks* que compõem o *framework* de otimização de Preço de Venda permanecem no interior do pacote *Specific* e ilustrado na Figura 35.

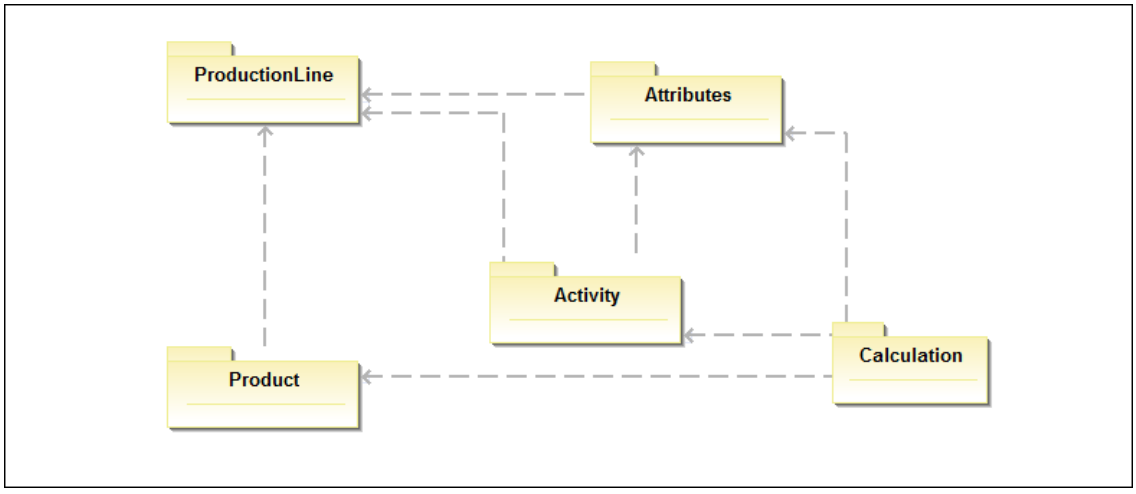


Figura 35 - Pacote *Specific*
Fonte: Autoria Própria.

Alguns *subframeworks* são comuns entre os métodos de formação de preço de venda, como por exemplo, o *Attributes* e *Calculation*, porém a implementação de ambos são diferentes para cada método a ser empregado. O *subframework Attributes*, não contém os mesmos atributos nos 3 (três) métodos estudados, como o *Calculation* também não possui a mesma fórmula para calcular o preço de venda de um determinado produto.

O pacote *ProductionLine* conterá a implementação do *subframework ProductionLine*, que tem como principal função exibir todas as linhas de produção, podendo o usuário cadastrar novas linhas, editar ou desabilitar linhas já existentes.

No pacote *Product* será implementado o *subframework Product*, o qual será responsável por exibir todos os produtos cadastrados, podendo o usuário efetuar o cadastro de um novo produto, editá-lo ou desabilitá-lo.

No pacote *Activity* será implementado o *subframework Activity*, que terá como função principal exibir todas as atividades existentes, podendo o usuário efetuar o cadastro de uma nova atividade, editá-la ou desabilitá-la.

No pacote *Attributes* foi implementado o *subframework Attributes*, sua função é exibir todos os atributos existentes, dando ao usuário a opção de cadastrar, editar ou desabilitar um atributo.

Com exceção do pacote *Attributes*, todos os outros *subframeworks* contidos no pacote *Specific* são em sua totalidade específicos do método ABC.

Há relações de dependência entre os *subframeworks*. O *subframework ProductionLine* é o único que não depende de nenhum outro, pois para efetuar o cadastro de uma nova linha de produção é necessário apenas o nome desta nova linha. Por outro lado, o *subframework Product* necessita o nome do novo produto e também da linha de produção que ele pertence. O mesmo acontece com o *subframework Attributes*, que precisa do nome do novo atributo a ser cadastrado juntamente com a linha de produção que este produto esteja relacionado. O *subframework Activity* além de necessitar do nome da nova atividade, também se relaciona com um atributo e com uma linha de produção. Por fim, o *subframework Calculation* é responsável pelo cálculo do preço de venda do produto, porém, para obter êxito em sua função, deve estar com todos os atributos e valores disponíveis.

3.6.1 Definição dos pontos comuns e específicos do Subframework *Attributes*

O pacote *Attributes* foi modelado e implementado usando o desenvolvimento em camadas – *View*, *VO, BusinessRule*, *Persistence* – e dentro destas foram aplicados alguns padrões de projeto, entre eles: *Decorator*, *Abstract Factory*, *FactoryMethod*, *DAOFactory* e *MVC* (*Model-view-controller*). A arquitetura deste pacote está ilustrada na Figura 36.

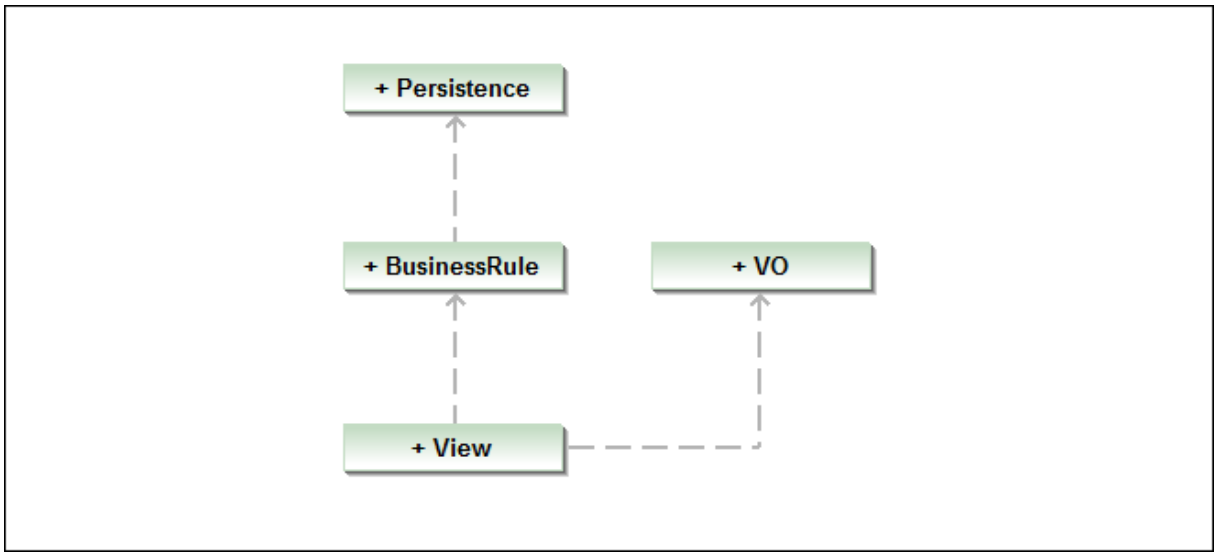


Figura 36 – Pacote *Attributes*
Fonte: Autoria Própria.

Definidas as dependências de cada pacote, as Figuras 37, 38, 39 e 40 ilustram respectivamente os pacotes *View, VO, BusinessRule*, *Persistence*, contidos no *subframework Attributes*.

No pacote *View*, encontram-se as classes que provêm as interfaces do sistema por meio das quais o usuário irá interagir com o *software.* No pacote *Persistence* encontram-se as classes responsáveis pela conexão ao Banco de Dados.

Nas classes *VO* e *BusinessRule,* encontram-se as classes que definem o modo como as interfaces do usuário reagem às entradas do mesmo, ou seja, ambos os pacotes são responsáveis pelo gerenciamento da interação entre os pacotes *View* e *Persistence.*

Os estereótipos contidos nas figuras são: <<*Import*>> o qual refere-se à importação de classes, <<*Instantiate*>> refere-se à instanciação, <<*Call*>> refere-se

a uma chamada a outra classe, *{incomplete}* o qual refere-se ao ponto de flexibilidade do *framework, <<FCE>>* indica que essa classe pode ser adaptada por meio da adição e/ou remoção de atributos e/ou métodos e o estereótipo <<FA>> indica que a classe é adaptável.

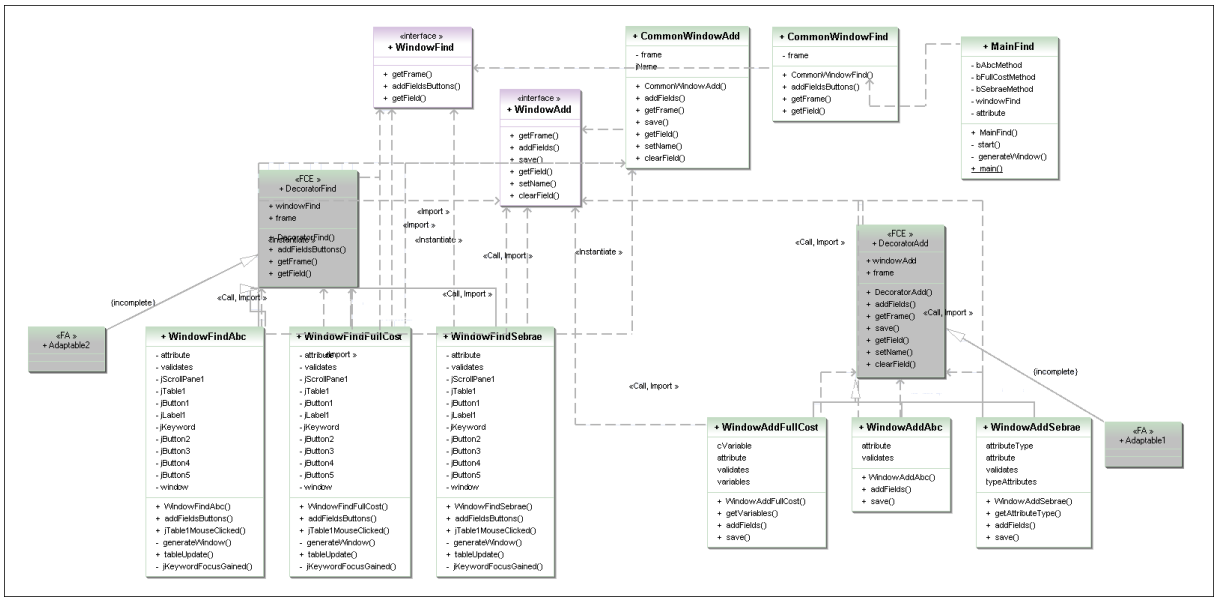


Figura 37 – Pacote *View* pertencente ao *Subframework Attributes*
Fonte: Autoria Própria.

Neste pacote, encontram-se classes comuns – *frozen spots* – identificadas pelas classes de cor cinza e pelas *interfaces* identificadas pelo estereótipo <<*interface*>>, e específicas – *hot spots* –, identificadas pelas classes de cor branca sem estereótipos.

As classes comuns, frozen spots, permanecem no interior do pacote *Attributes,* ilustrado na Figura 34. Em contrapartida, as classes específicas, *hot spots*, situam-se no interior do pacote *Attributes* na parte específica do *framework* (Figura 35).

As classes *Adaptable1* e *Adaptable2*, referentes às heranças das classes *DecoratorAdd* e *DecoratorFind,* respectivamente, e identificam justamente o ponto de flexibilidade do *framework* proposto, no qual poderão ser adicionadas mais telas referentes aos novos métodos acrescentados ao mesmo.

A Figura 38 representa o pacote VO, no qual se encontram classes comuns e específicas. Como todos os diagramas seguem um padrão, as classes comuns são identificadas pelas classes de cor cinza enquanto as classes específicas identificadas de cor branca.

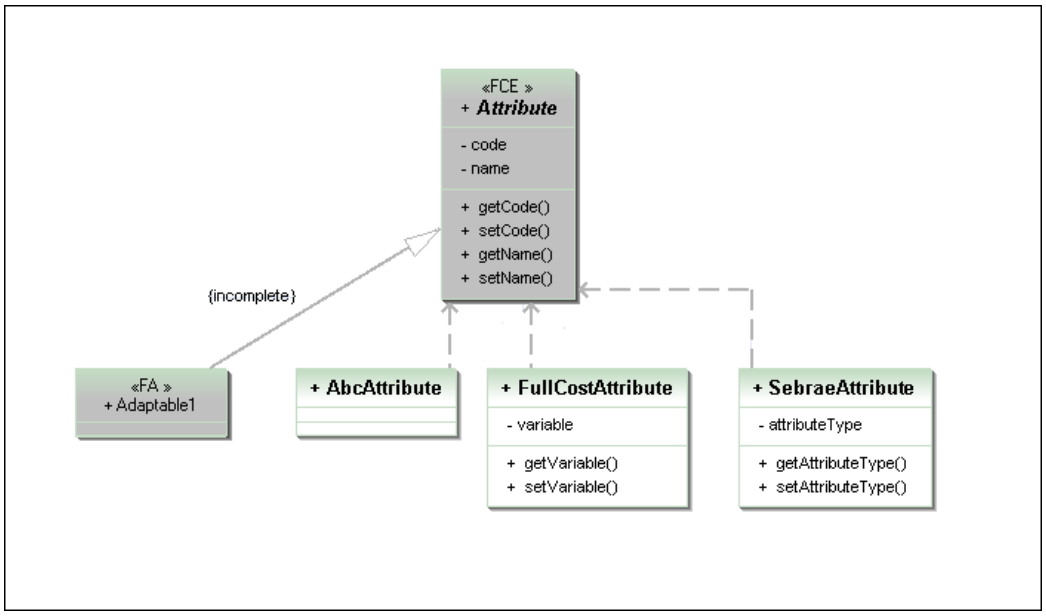


Figura 38 – Pacote VO pertencente ao *Subframework Attributes*
Fonte: Autoria Própria.

A classe *Adaptable1* presente no diagrama do pacote VO (Figura 38) identifica o ponto de flexibilidade do *framework,* ou seja, onde futuramente poderão ser adicionados novos métodos ao *framework.*

A Figura 39 ilustra o pacote *BusinessRule*. As notações presente nesta imagem, como *frozen spots, hot spots* e *adaptable,* seguem o mesmo padrão citado anteriormente.

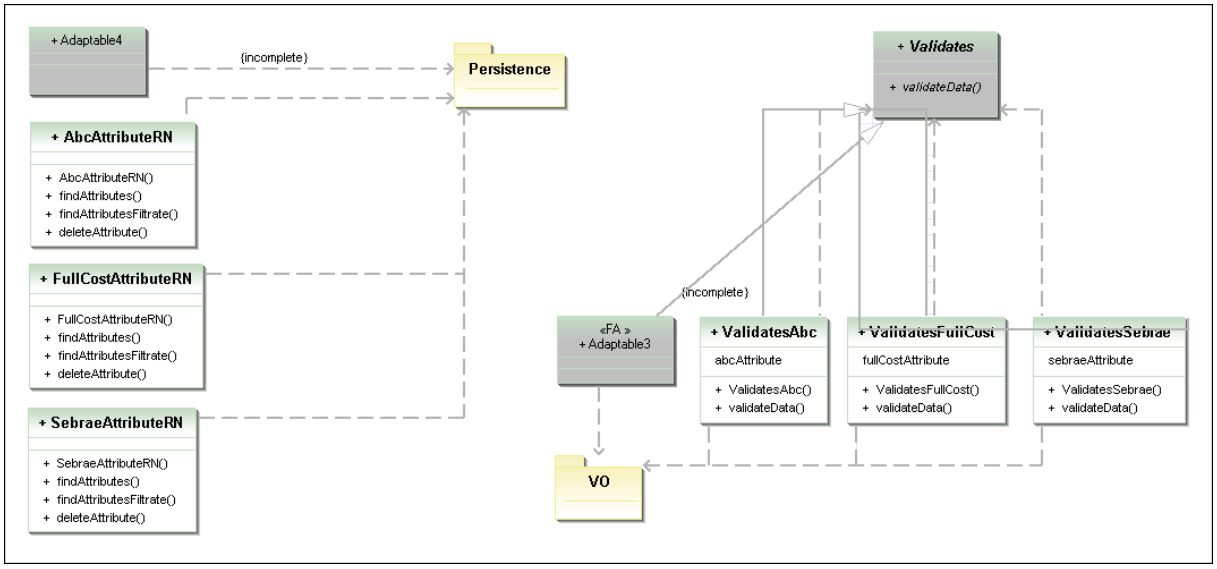


Figura 39 – Pacote *BusinessRule* pertencente ao *Subframework Attributes*
Fonte: Autoria Própria.

No pacote *Persistence*, foi aplicado o padrão de projeto *DAOFactory* e, por este motivo, possui em seu interior um pacote chamado *DAO,* onde se encontram a classe *DAOFactory* e as interfaces que compõem o *subframework Attributes* podendo ser extensível caso seja necessário adicionar um novo método para este domínio. Contém também o pacote *Firebird,* que é composto pela classe *FBDAOFirebird,* a qual é responsável pela conexão ao Banco de Dados e as classes referentes a cada método de estabelecimento de preço de venda que implementam as *interfaces* do pacote *DAO*. A Figura 40 ilustra o pacote *Persistence.*

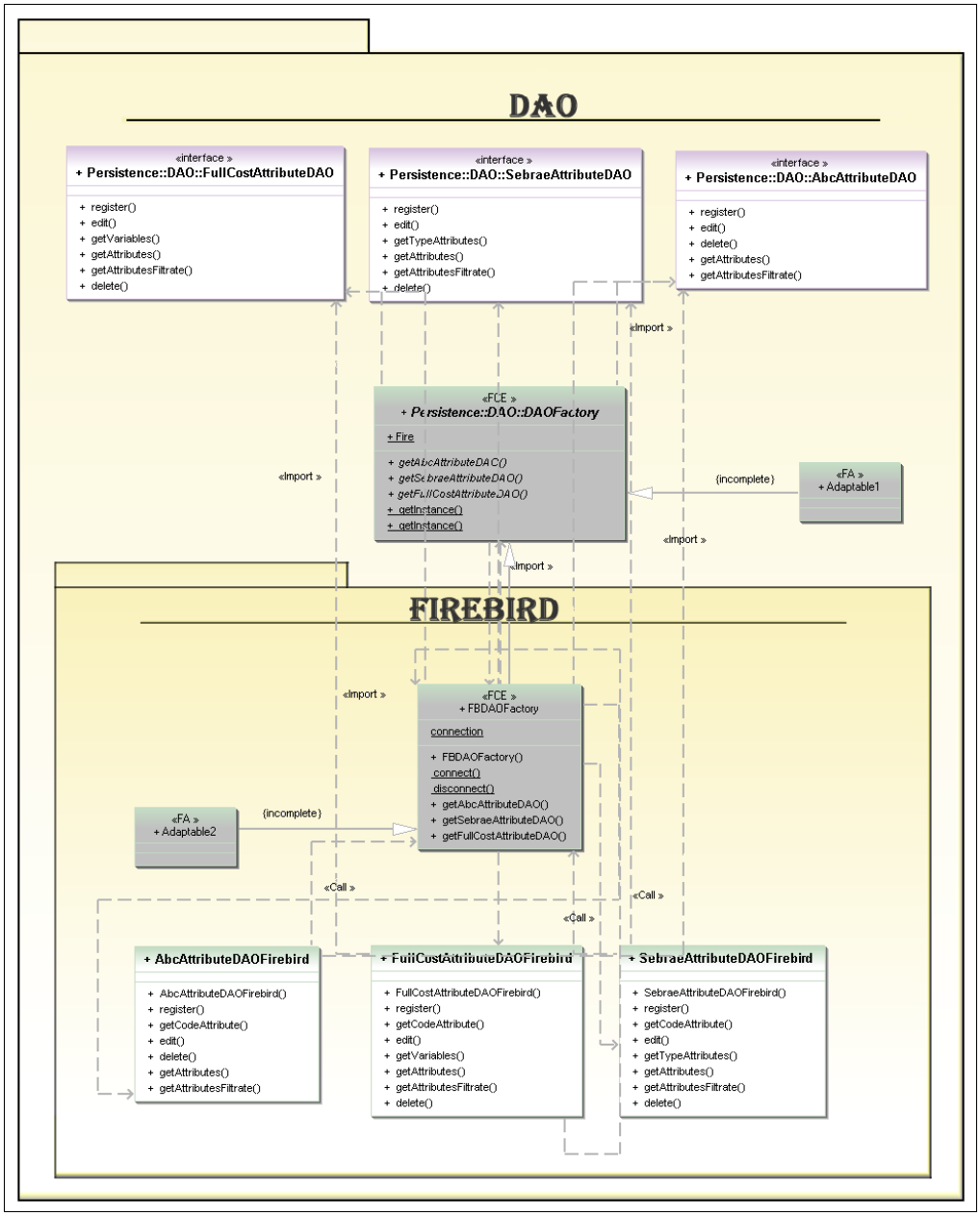


Figura 40 – Pacote *Persistence* pertencente ao *Subframework Attributes*
Fonte: Autoria Própria.

3.6.2 Definição dos pontos comuns e específicos do Subframework FoodSystem

A Figura 41 ilustra a arquitetura do pacote *FoodSystem*, o qual está implementado utilizando o desenvolvimento em camadas, além de serem aplicados os padrões de projeto: *Decorator*, *Abstract Factory*, *FactoryMethod*, *DAOFactory* e *MVC* (*Model-view-controller*).

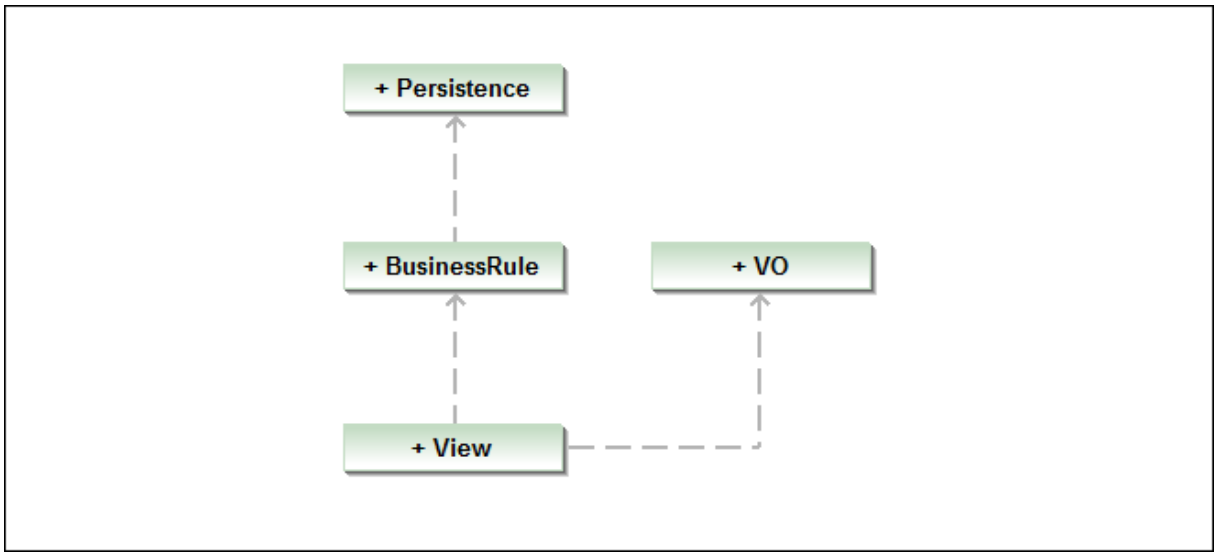


Figura 41 – Pacote *FoodSystem*
Fonte: Autoria Própria.

O pacote *View*, encontram-se as classes as quais o usuário irá interagir com o sistema, ou seja, encontram-se neste pacote as classes referentes às interfaces do software. No pacote *Persistence* estão todas as classes responsáveis pela conexão ao Banco de Dados. E por fim, os pacotes *VO* e *BusinessRule*, estão todas as classes responsáveis pelo gerenciamento da interação entre os pacotes *View* e *Persistence.*

As Figuras 42, 43, 44 e 45, ilustram a implementação de cada um destes pacotes.

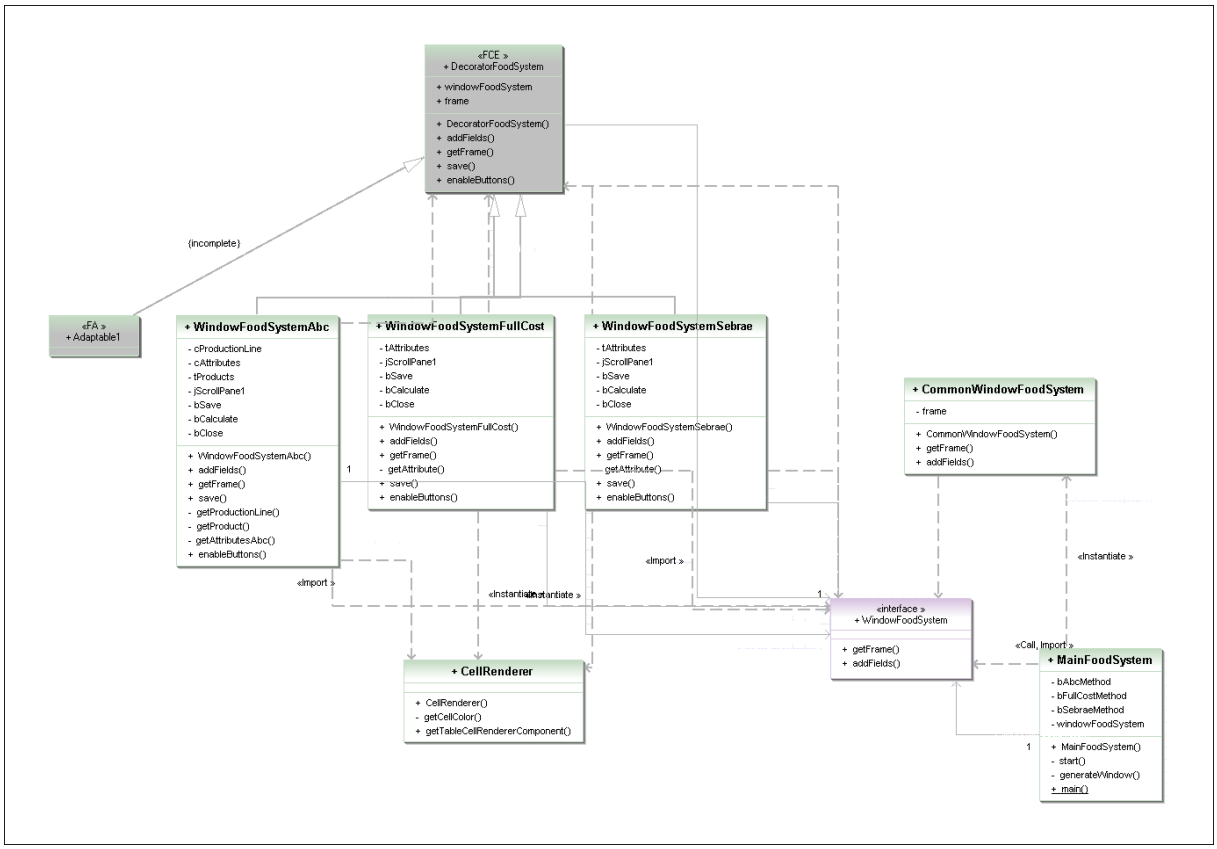


Figura 42 – Pacote *View* pertencente ao *subframework FoodSystem*
Fonte: Autoria Própria.

A Figura 43 representa o pacote VO. Neste pacote são apresentadas as classes comuns e específicas, representadas pelas classes de cor branca e cinza e pelos referentes estereótipos. A classe *Adaptable1* representa o ponto de flexibilidade do *framework.*

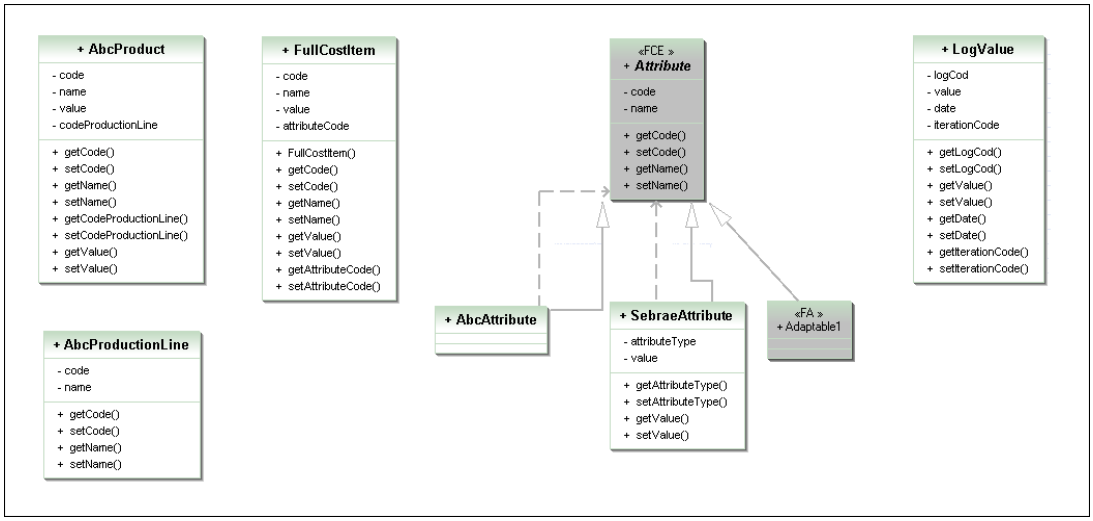


Figura 43 - Pacote *VO* pertencente ao *subframework FoodSystem*
Fonte: Autoria Própria.

O pacote *BusinessRule,* ilustrado na Figura 44, contempla as notações citadas anteriormente além de representar a dependência das classes aos pacotes *Persistence* e *VO*.

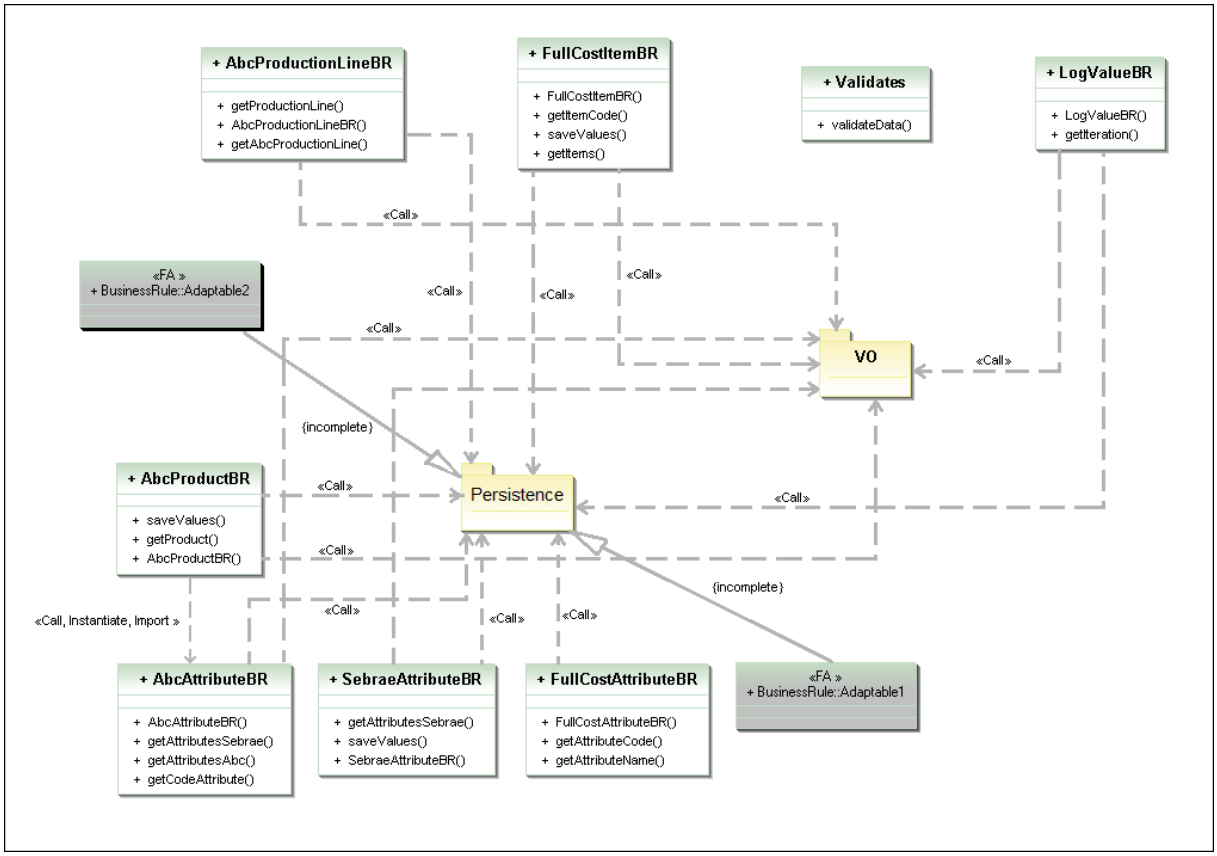


Figura 44 - Pacote *BusinessRule* pertencente ao *subframework FoodSystem*
Fonte: Autoria Própria.

A Figura 45 ilustra o pacote *Persistence*, no qual aplicou-se o padrão de projeto *DAOFactory*.

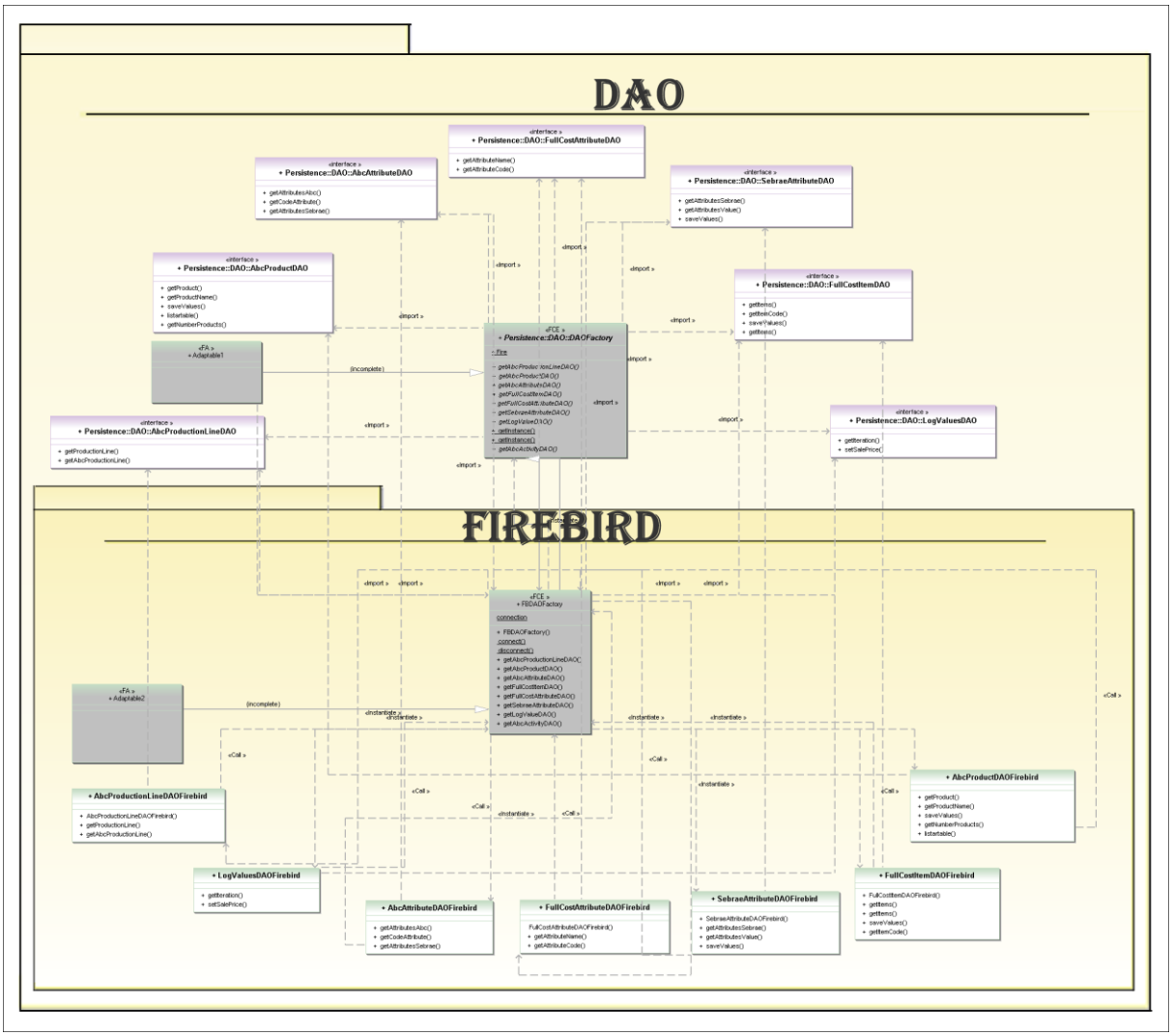


Figura 45 - Pacote *Persistence* pertencente ao *subframework FoodSystem*
Fonte: Autoria Própria.

3.6.3 Definição dos pontos comuns e específicos do Subframework Calculation

Semelhante aos demais *subframeworks,* o *Subframework Calculation* foi desenvolvido utilizando padrões de projeto e desenvolvimento em camadas. A Figura 46 ilustra a arquitetura do pacote *Calculation*.

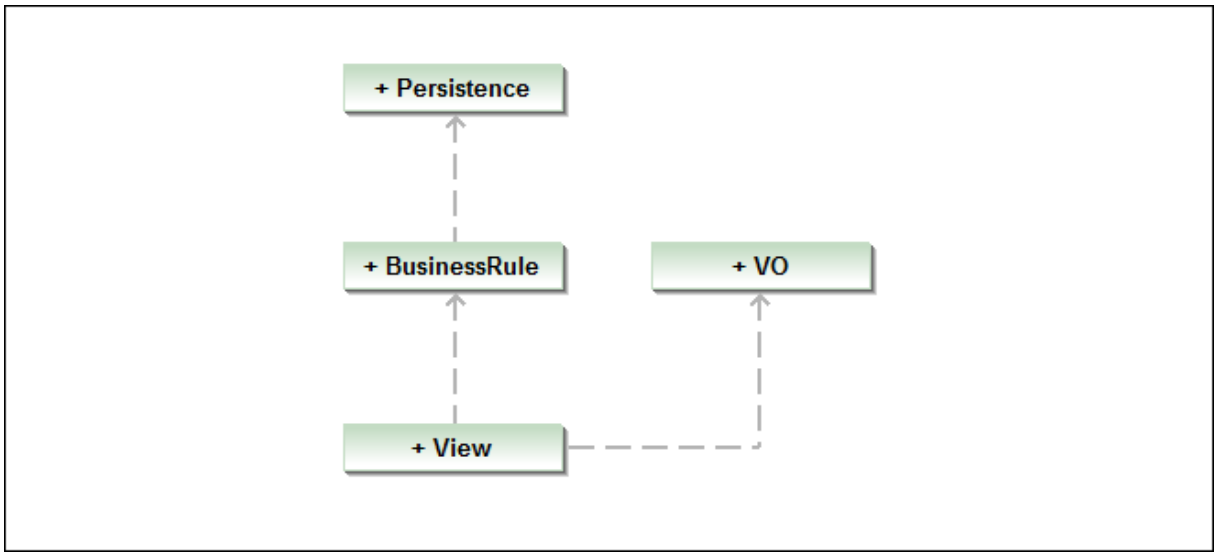


Figura 46 – Pacote *Calculation*
Fonte: Autoria Própria.

A Figura 47 ilustra o Diagrama de Classes do Pacote VO, o qual foi possível identificar os pontos de flexibilidade do mesmo.

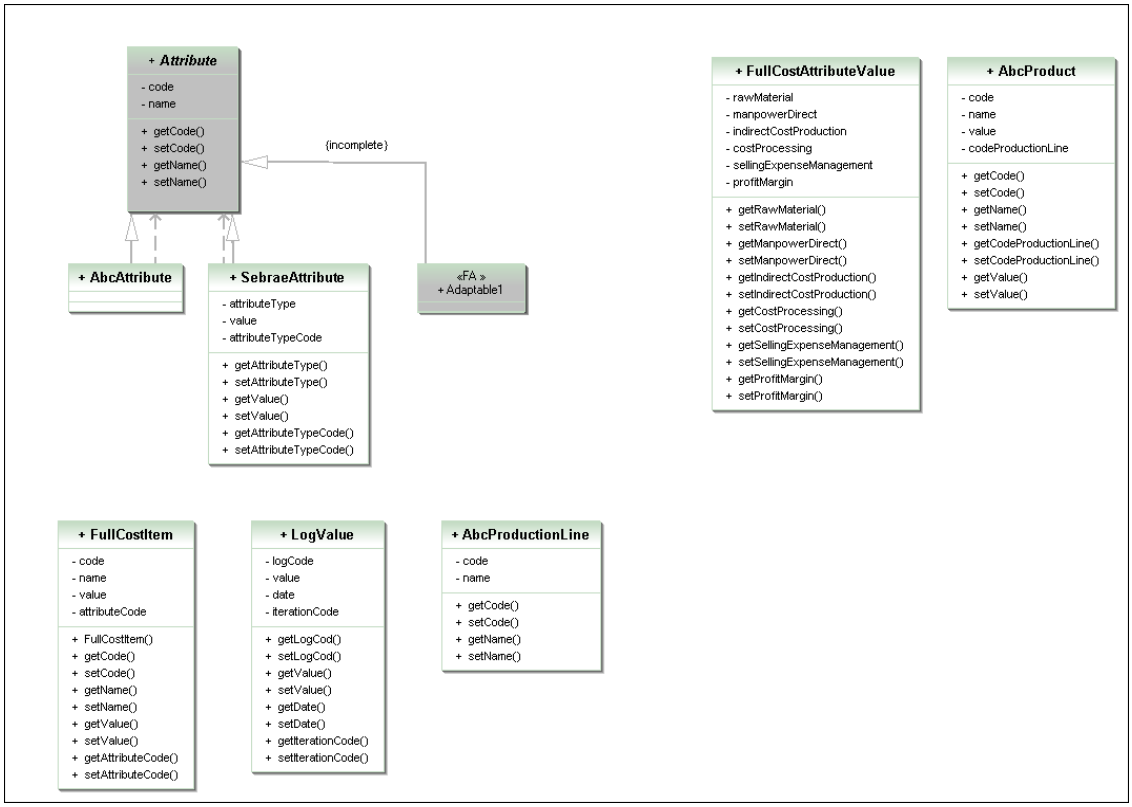


Figura 47 - Pacote VO pertencente ao *subframework Calculation*
Fonte: Autoria Própria.

A Figura 48 refere-se ao pacote *View.* Encontra-se neste pacote uma *interface*, a qual é identificada pelo seu estereótipo <<*interface*>> sendo comum entre as aplicações-exemplo como as classes identificadas pela cor branca.

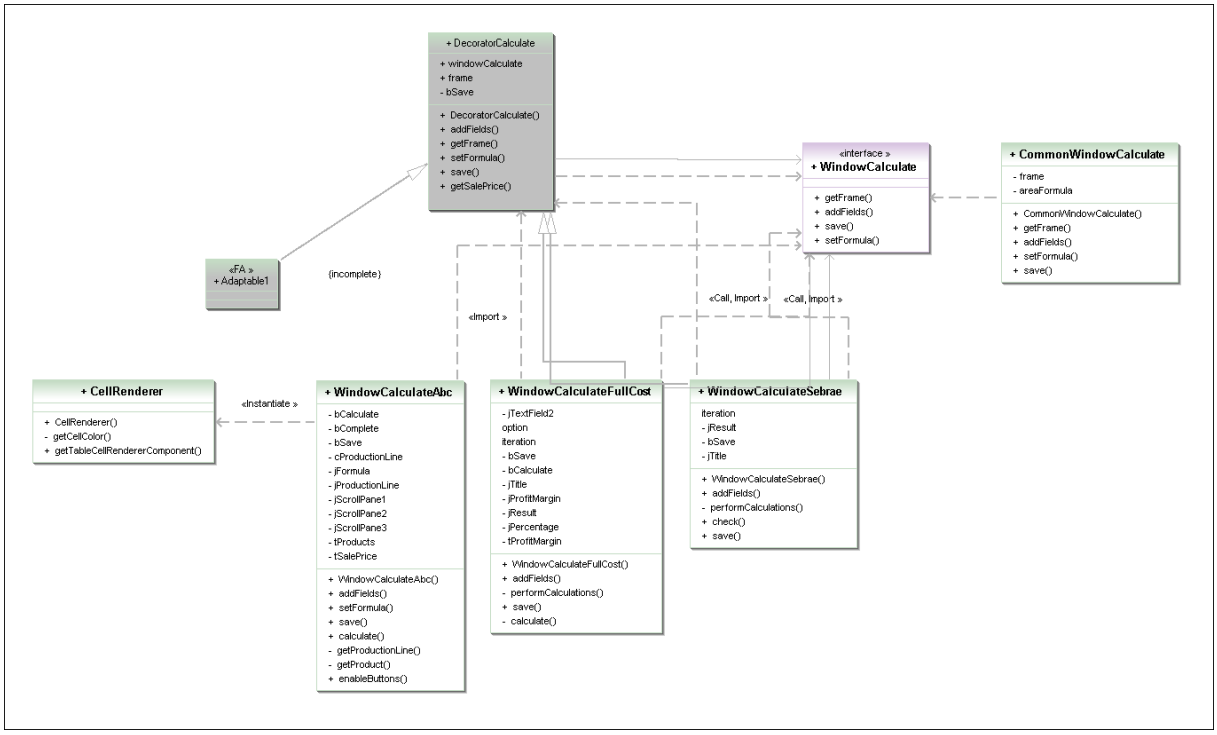


Figura 48 - Pacote *View* pertencente ao *subframework Calculation*
Fonte: Autoria Própria.

Assim como nos diagramas anteriores, a classes *Adaptable1* representa o ponto de flexibilidade do *framework* proposto. Esta ilustração também é observada na Figura 49, a qual explica o pacote *BusinessRule.*

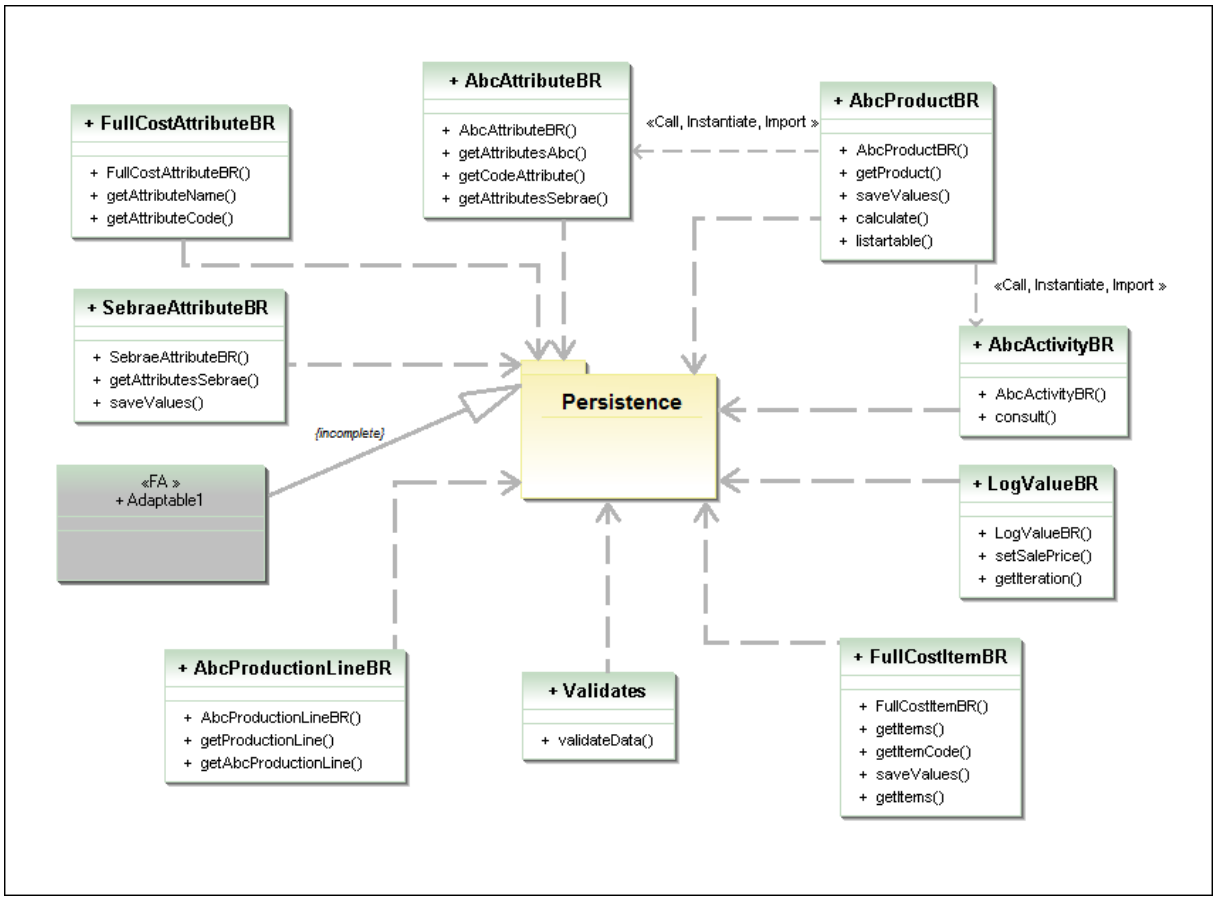


Figura 49 - Pacote *BusinessRule* pertencente ao *subframework Calculation*
Fonte: Autoria Própria.

Para evitar um “poluição” do Diagrama, foi utilizado o Pacote *FoodSystem* para explicar as dependências das classes contidas no pacote *BusinessRule*, facilitando assim o entendimento do mesmo.

Assim como nos outros *subframeworks*, neste também foi aplicado o padrão de projeto *DAOFactory*. A Figura 50 ilustra o pacote *Persistence* onde em seu interior localiza-se o pacote *DAO*, este, por sua vez, possui o pacote *Firebird,* as *interfaces* que fazem parte do *Subframework Calculation,* e a classe *DAOFactory*.

A conexão ao Banco de Dados é realizada por meio da classe *FBDAOFirebird* que se encontra no interior do pacote *Firebird,* juntamente com as classes que implementam as *interfaces*.

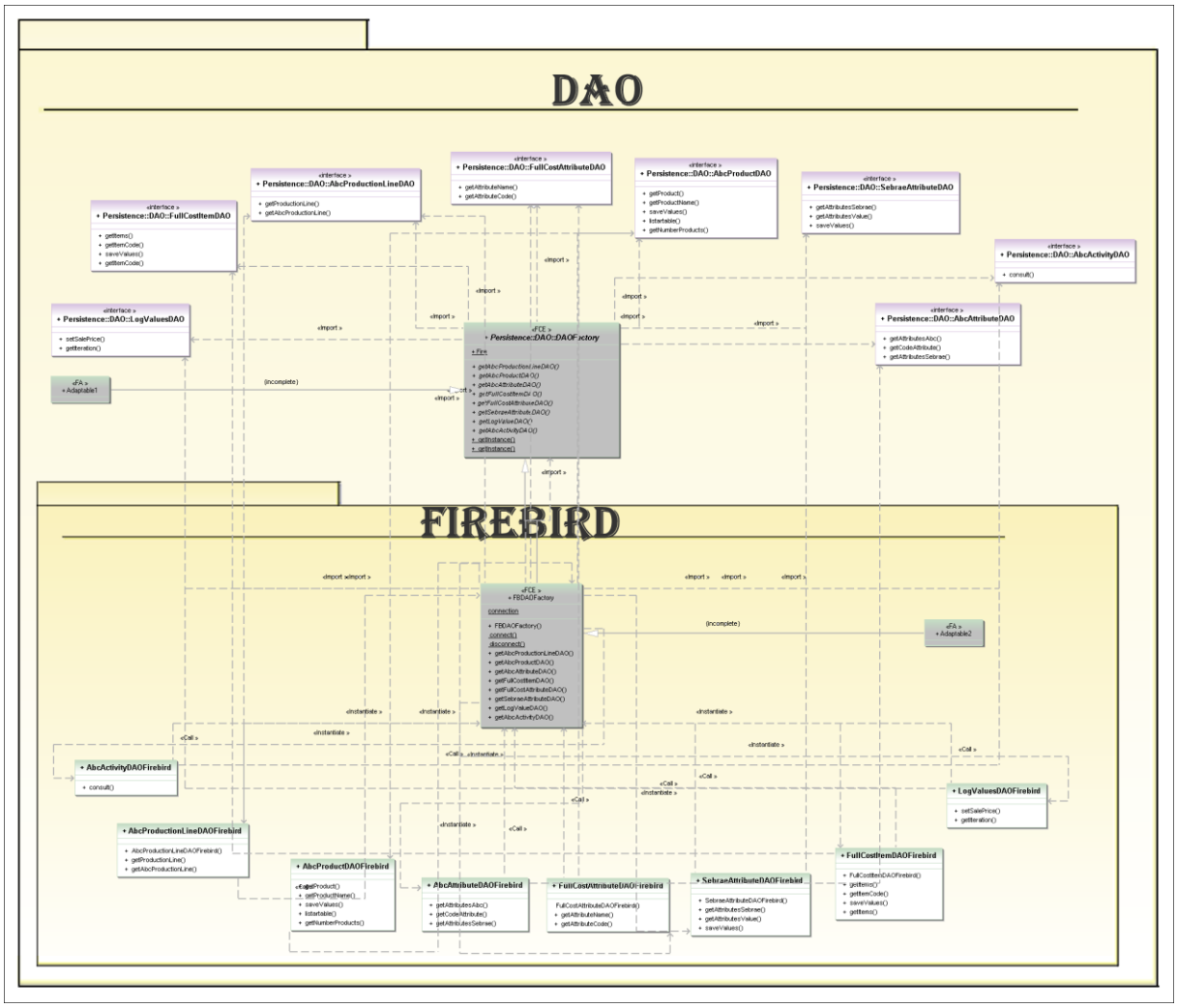


Figura 50 - Pacote *Persistence* pertencente ao *subframework Calculation*
Fonte: Autoria Própria.

4 RESULTADOS

Este Capítulo aborda os resultados obtidos com o desenvolvimento da arquitetura geral do *framework* e a implementação dos *Subframeworks Attributes, FoodSystem* e *Calculation*. A Seção 4.1 apresenta os protótipos dos *Subframeworks* desenvolvidos. A Seção 4.2 descreve os passos necessários para uma futura extensão dos métodos de formação de preço de venda. Por fim, A Seção 4.3 apresenta as vantagens e desvantagens obtidas com o desenvolvimento de *framework*.

4.1 PROTÓTIPOS DOS SUBFRAMEWORKS

A seguir será apresentado os resultados da implementação da modelagem dos *subframeworks* relatados no capítulo anterior. A Seção 4.1 apresenta os protótipos do *Subframework Attributes*. A Seção 4.2 exibe os protótipos do *Subframework FoodSystem*. E por fim, a Seção 4.3 aborda os protótipos do *Subframework Calculation.*

4.1.1 Subframework Attributes

Ao acessar o *Subframework Attributes*, primeiramente o usuário depara-se com a Tela de Busca de Atributos do Método específico a que ele está efetuando o cálculo de preço de venda.

Observa-se que as Telas de Busca dos três métodos são semelhantes, porém, notam-se algumas diferenças, como por exemplo, a Tela de Busca de Atributos do Método ABC (Figura 51) tem apenas dois campos na tabela – Código e Descrição –, por outro lado, a Tela de Busca de Atributos do Método Sebrae (Figura 52), apresenta 3 colunas na tabela – Código, Descrição e Tipo de Atributo –, por fim, a Tela de Busca de Atributos do Método Custo Pleno (Figura 53) também apresenta 3 colunas – Código, Descrição e Variável.

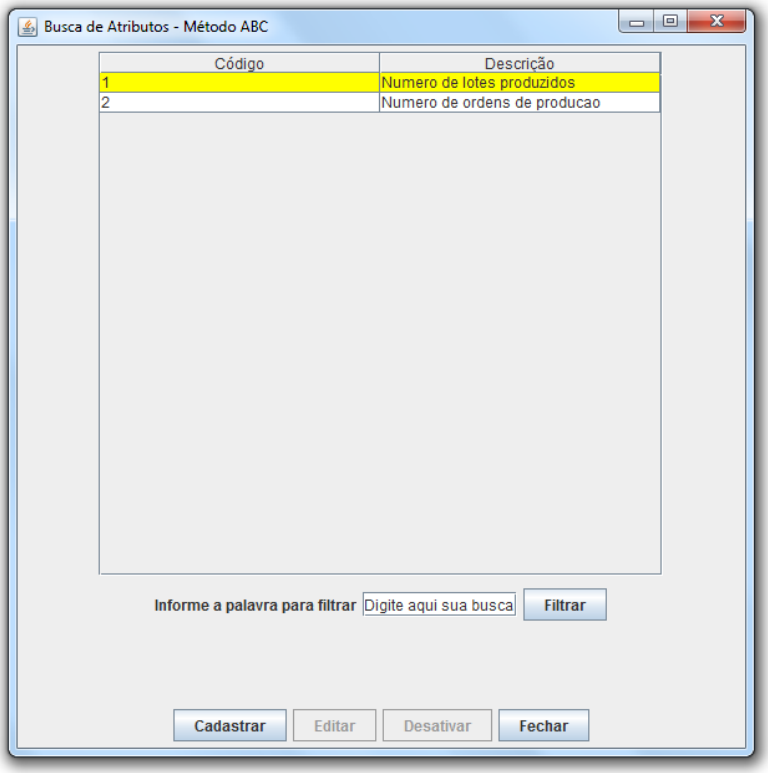


Figura 51 – Tela de Busca do Método ABC
Fonte: Autoria Própria.

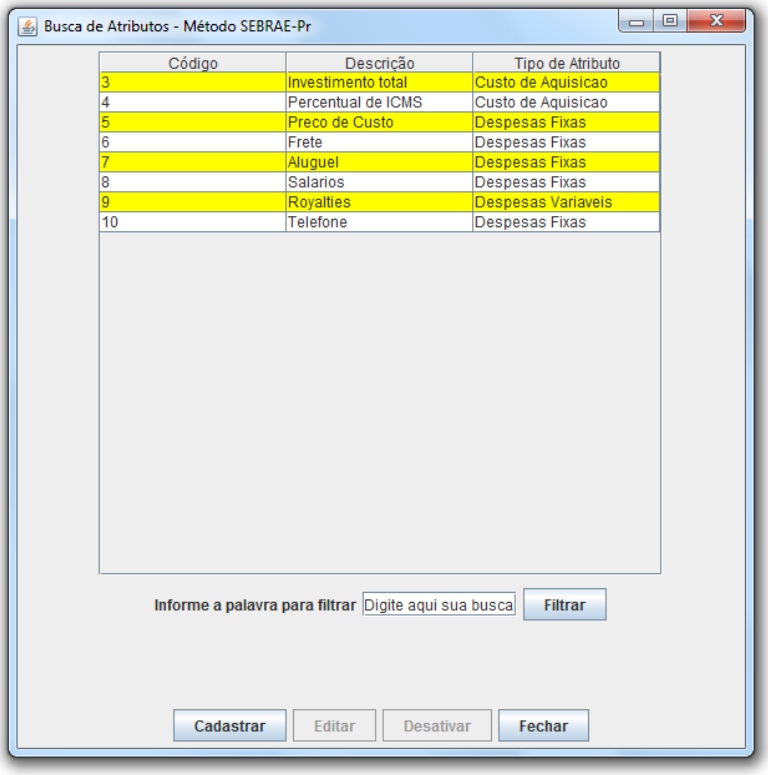


Figura 52 – Tela de Busca do Método Sebrae
Fonte: Autoria Própria.

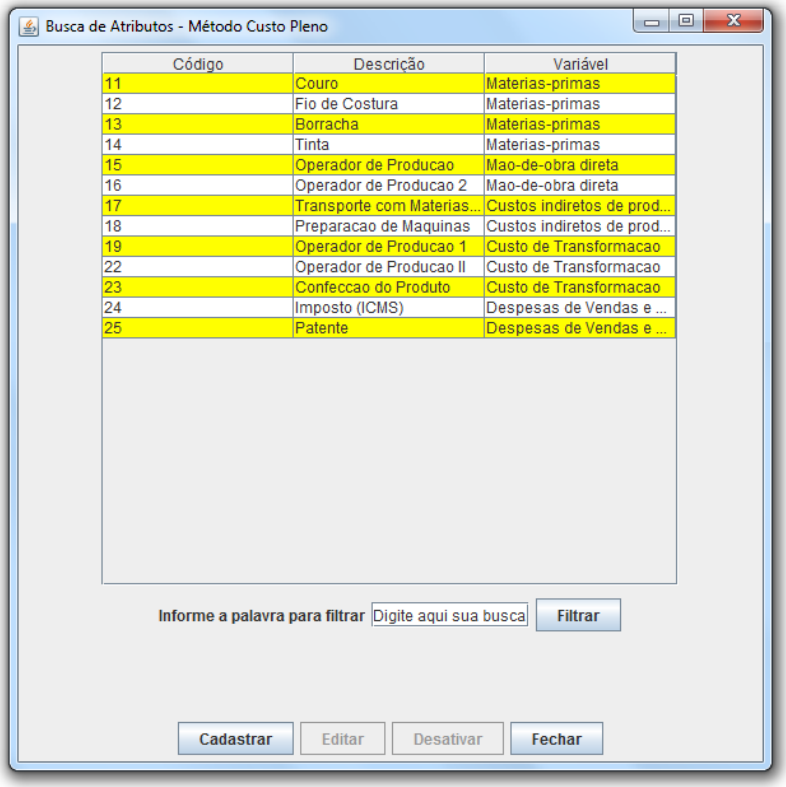


Figura 53 – Tela de Busca do Método Custo Pleno
Fonte: Autoria Própria.

Nas três telas o usuário tem a opção de cadastrar um novo atributo, editar ou desabilitar um atributo existente e filtrar a pesquisa. Ao escolher um método, o sistema localiza todos os objetos que devem ser instanciados para ele.

Para filtrar a pesquisa dos atributos, basta digitar qualquer caractere ou palavra no campo de filtro e pressionar a tecla “Filtrar”, a qual fará a pesquisa e exibirá na tabela apenas atributos relacionados à pesquisa.

As telas de cadastro e edição de um novo atributo são iguais para cada Método, a diferença entre elas é o preenchimento automático do campo do nome do atributo em caso de edição. As Figuras 54, 55 e 56 ilustram respectivamente as telas de cadastro do Método ABC, Sebrae e Custo Pleno.

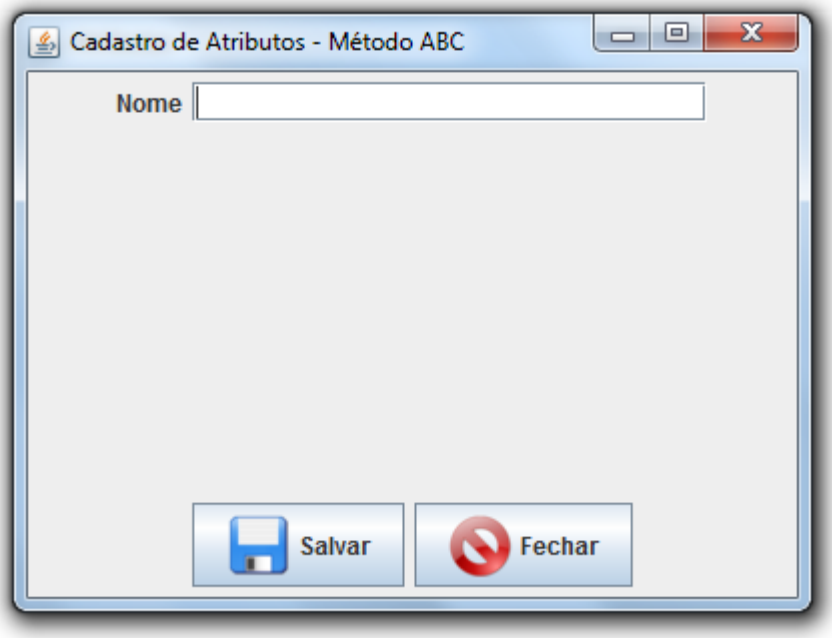


Figura 54 – Tela de Cadastro de Atributos do Método ABC
Fonte: Autoria Própria.

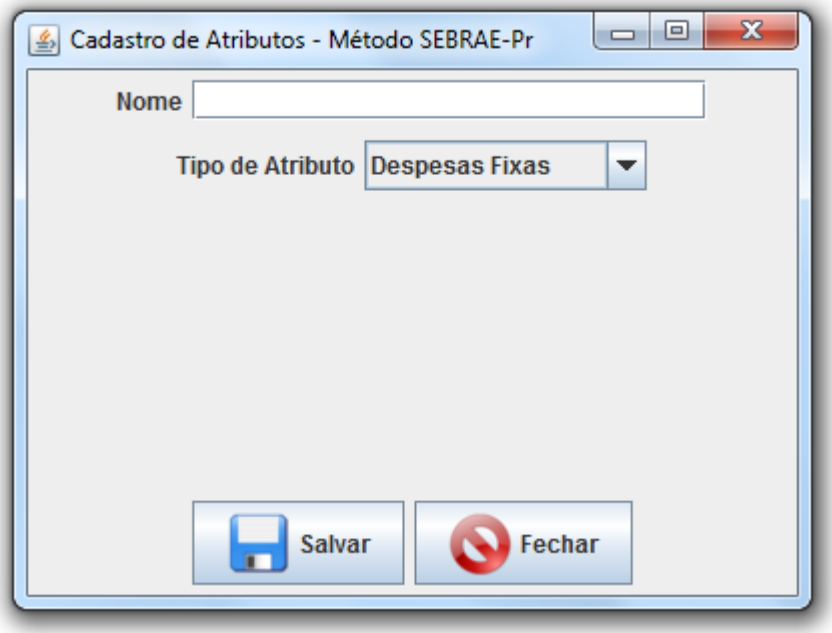


Figura 55 – Tela de Cadastro de Atributos do Método Sebrae
Fonte: Autoria Própria.

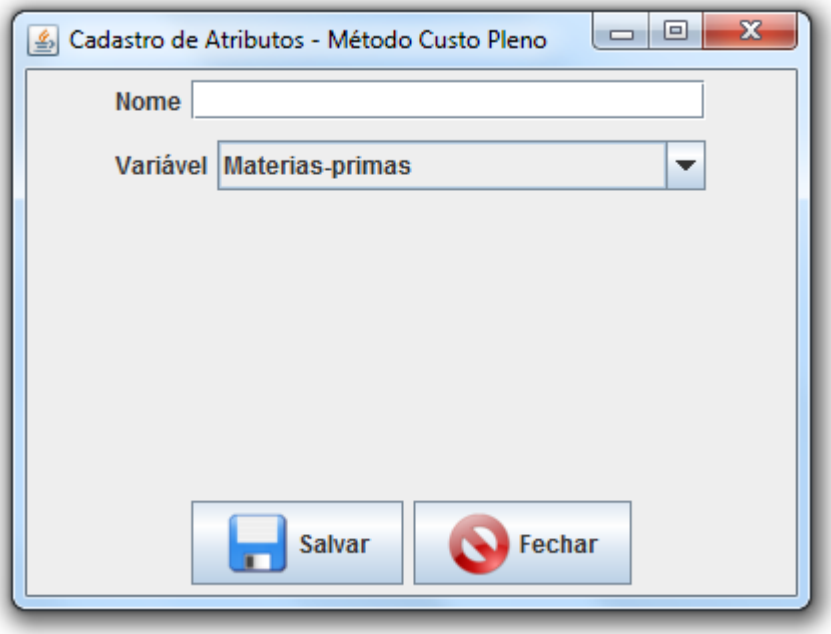


Figura 56 – Tela de Cadastro de Atributos do Método Custo Pleno
Fonte: Autoria Própria.

Para efetuar o cadastro de um novo atributo, o usuário deve informar o nome do atributo. Se este atributo for do Método Sebrae-PR é necessário também selecionar o Tipo de Atributo. Se o atributo que está cadastrando for do Método Custo Pleno, é necessário selecionar a Variável que este atributo pertence.

Após preenchidos todos os dados e pressionado o botão “Salvar”, o sistema faz uma validação dos dados informados. Esta validação é feita no pacote *“BusinessRule”*. Se os dados estiverem todos corretos, os mesmos são enviados a camada de persistência, onde é realizada a conexão com o Banco de Dados e o novo atributo é armazenado, exibindo uma mensagem de confirmação ao usuário (Figura 57). Se algo na validação estiver errado, uma mensagem de erro é exibida (Figura 58) e o processo de cadastro não é concluído.

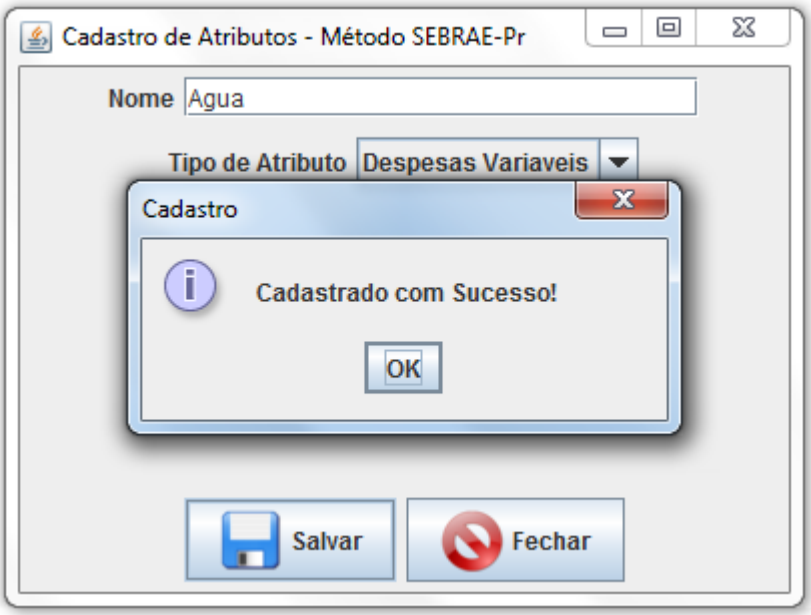


Figura 57 – Mensagem de Confirmação do Cadastro de Atributos
Fonte: Autoria Própria.

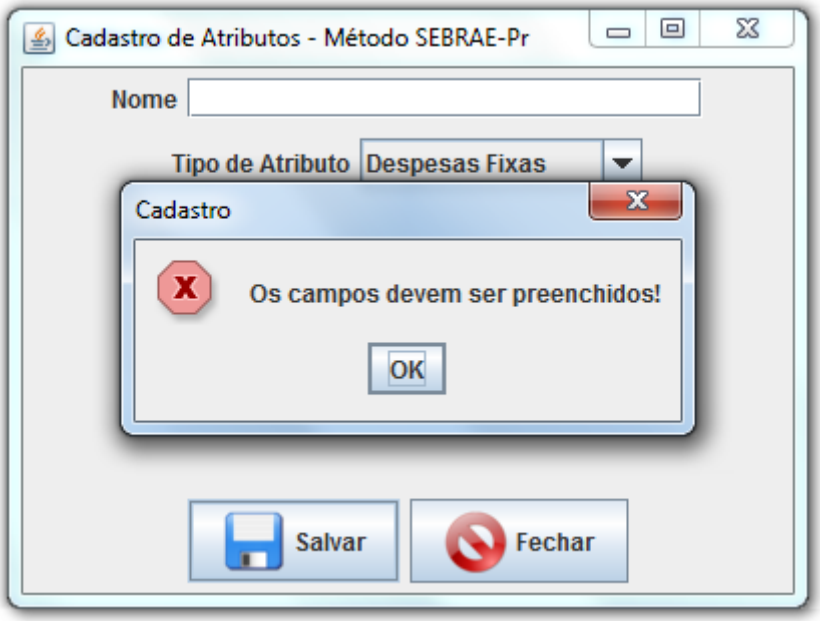


Figura 58 – Mensagem de erro ao tentar efetuar o cadastro de atributos
Fonte: Autoria Própria.

4.1.2 Subframework FoodSystem

A Tela Alimentar Sistema tem a função de atribuir valores para as variáveis, tais como: produtos, itens e atributos, para posteriormente o *Subframework Calculation* efetuar o cálculo.

A Tela Alimentar Sistema para o Método ABC é ilustrada na Figura 59.

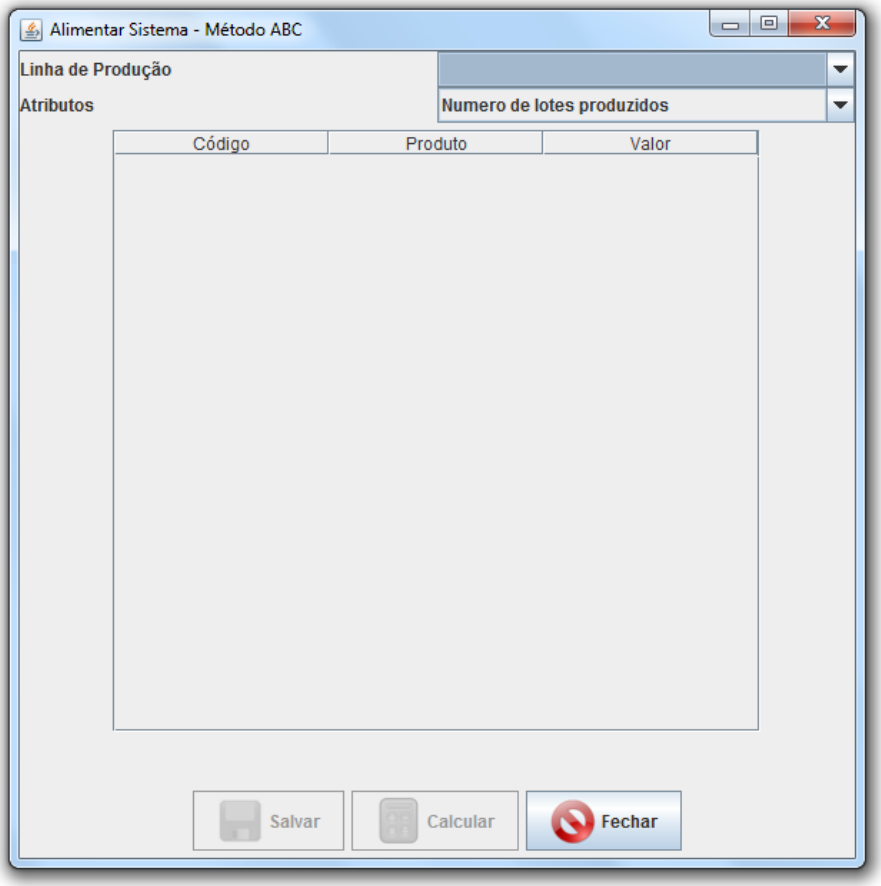


Figura 59 – Tela Alimentar Sistema do Método ABC
Fonte: Autoria Própria.

Inicialmente os botões são desabilitados, pois não há nenhum produto na tabela para efetuar o cálculo de preço de venda. Deve-se então escolher uma linha de produção a qual preencherá a tabela com os produtos relacionados à mesma. A Figura 60 ilustra a escolha da linha de produção Pizza e seus respectivos produtos.

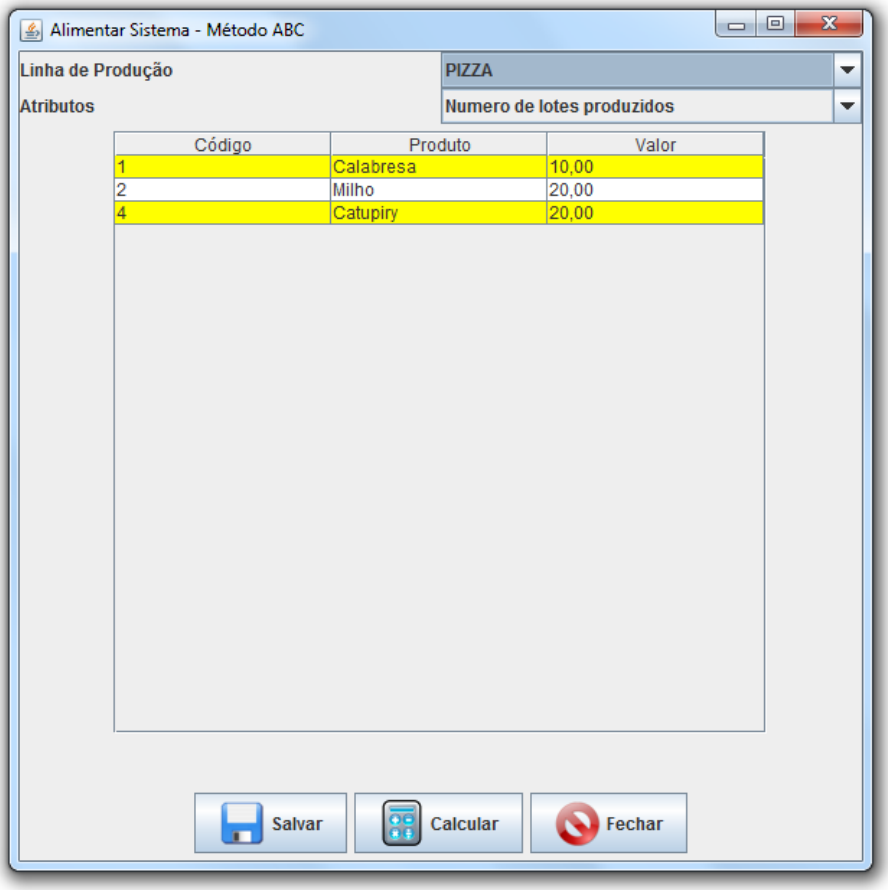


Figura 60 – Atualização da Tela Alimentar Sistema do Método ABC
Fonte: Autoria Própria.

Após selecionar uma linha de produção, poderá ser efetuada uma atualização dos valores dos produtos contidos na tabela. Posteriormente, poderá ser escolhida a tarefa a realizar. Se pressionado o botão “Salvar”, o sistema apenas atualiza os valores dos produtos, retornando uma mensagem de sucesso ou insucesso da operação, ilustrados nas Figuras 61 e 62, respectivamente. Se a opção for “Calcular”, o sistema realiza a atualização dos valores, assim como o botão “Salvar”, porém após esta atualização é chamada a Tela de Cálculo do Método (*Subframework Calculation*).

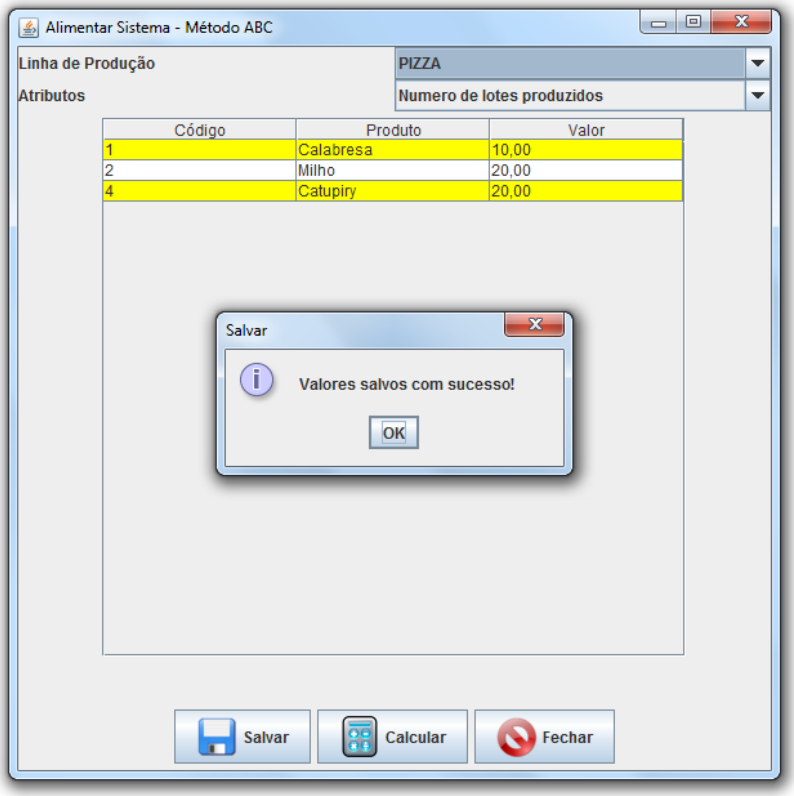


Figura 61 – Mensagem de sucesso ao atualizar valores do Método ABC
Fonte: Autoria Própria.

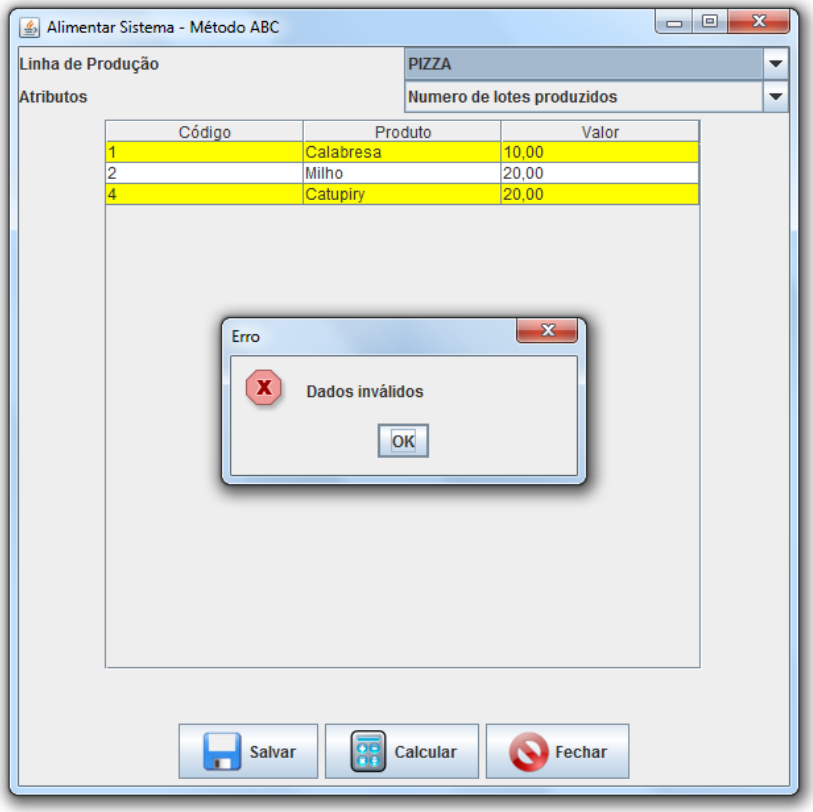


Figura 62 – Mensagem de erro ao tentar atualizar valores do Método ABC
Fonte: Autoria Própria.

A Figura 63 ilustra a Tela Alimentar Sistema do Método Custo Pleno. Apesar da semelhança desta tela e a Tela Alimentar Sistema do Método ABC (Figura 60), no método ABC necessita-se dos campos: linha de produção e atributos, e não possui um campo *RadioButton*. Campo este necessário para efetuar o cálculo de preço de venda do Método Custo Pleno, a qual difere pela fórmula. Na tabela também nota-se a troca da coluna Produtos (Método ABC) por Itens (Método Sebrae).

As funcionalidades permanecem as mesmas, podendo atualizar os valores na tabela e escolher entre os botões “Salvar” e “Calcular” explicados anteriormente.

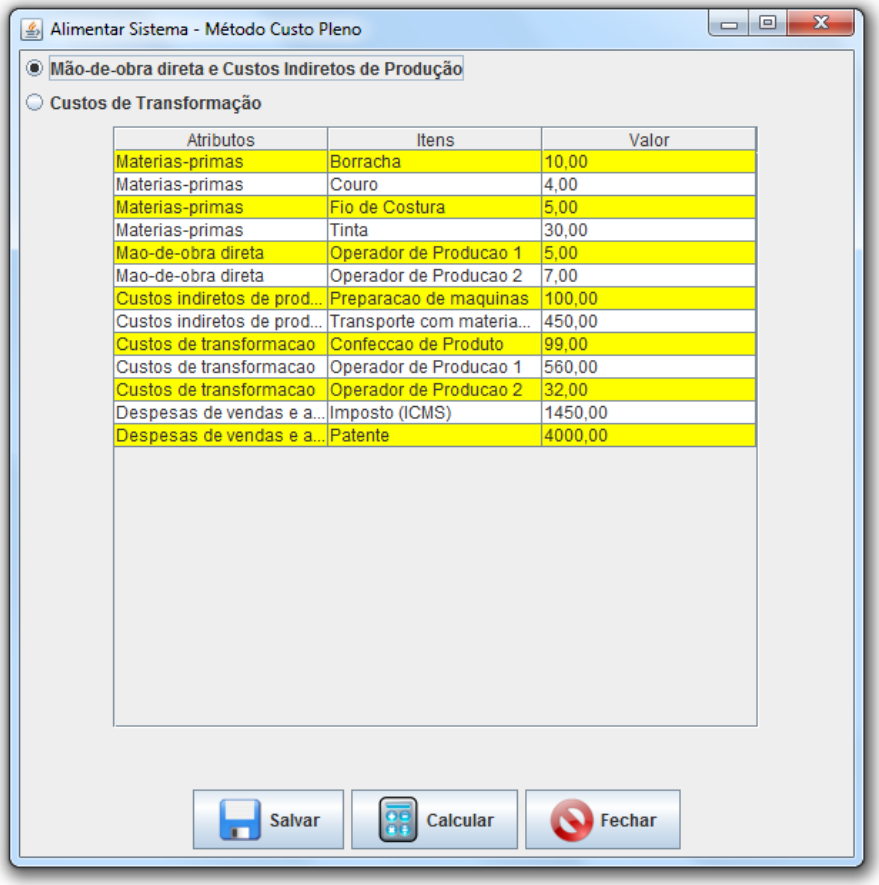


Figura 63 – Tela Alimentar Sistema do Método Custo Pleno
Fonte: Autoria Própria.

A Tela Alimentar Sistema do Método Sebrae é semelhante à Tela Alimentar Sistema dos outros métodos já citados, porém possui algumas particularidades. Os botões “Salvar” e “Calcular” permanecem neste Método com as mesmas funções, porém, nesta tela há ausência das opções contidas nas outras telas (linha de produção, atributo e fórmulas para obter o preço de venda) podendo apenas realizar

alterações nos valores da tabela. A Tela Alimentar Sistema do Método Sebrae é ilustrada pela Figura 64.

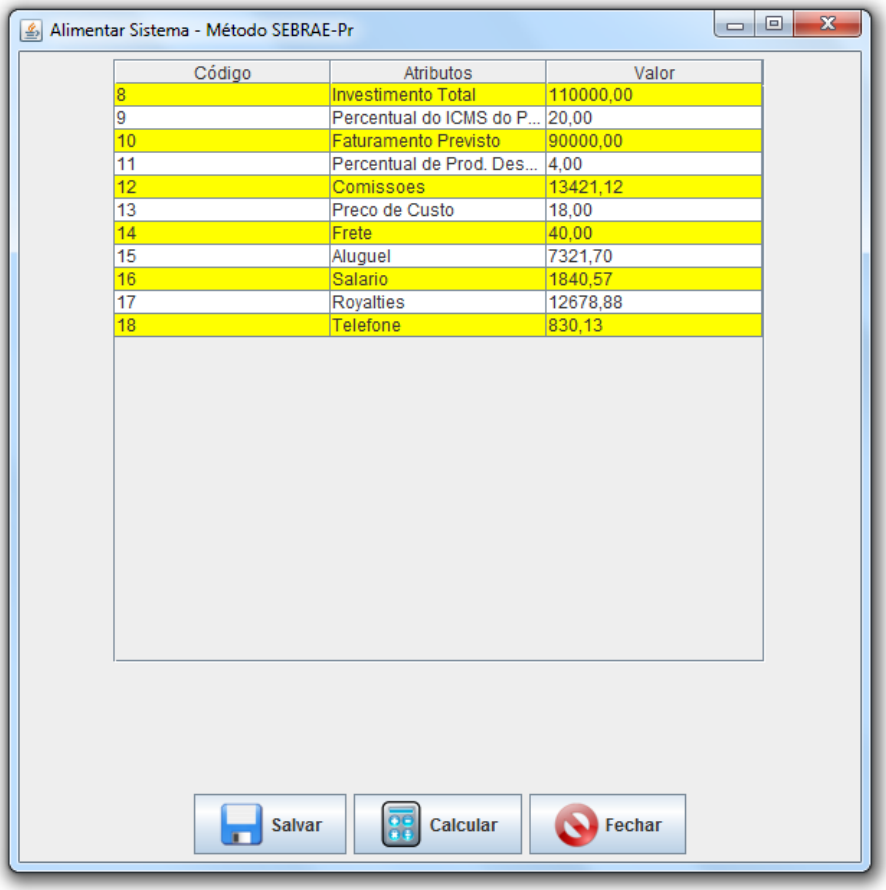


Figura 64 – Tela Alimentar Sistema do Método Sebrae
Fonte: Autoria Própria.

4.1.3 Subframework Calculation

Os três Métodos estudados possuem fórmulas diferentes para a realização do cálculo de preço de venda, por este motivo as telas são diferentes.

A Tela de Cálculo do Método ABC é ilustrada na Figura 65.

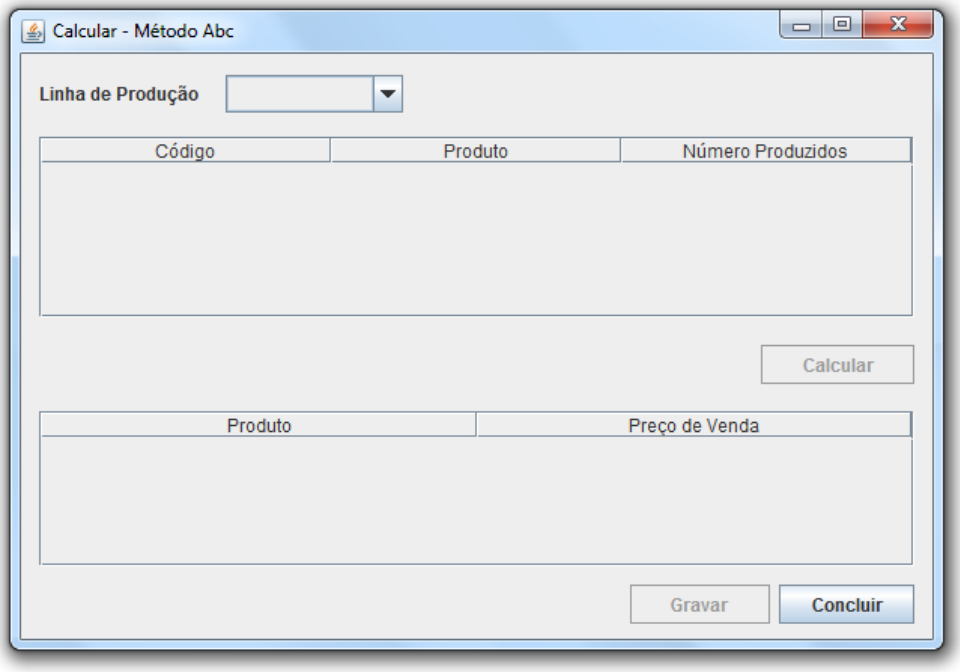


Figura 65 – Tela Cálculo do Método ABC
Fonte: Autoria Própria.

É necessário selecionar uma linha de produção para que o sistema preencha a tabela com os produtos condizentes com a mesma. Concluída esta etapa, deve-se então completar a coluna “Números Produzidos”, para depois pressionar o botão “Calcular” e efetuar o cálculo. Esta etapa é ilustrada na Figura 66.

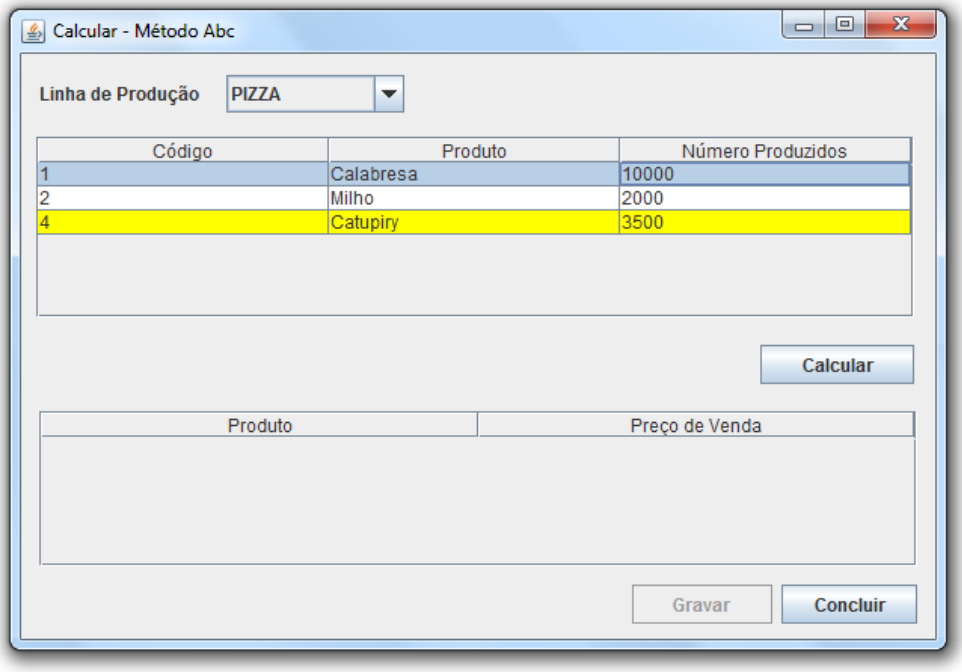


Figura 66 – Preenchimento da Coluna Número Produzidos
Fonte: Autoria Própria.

Ao pressionar o botão “Calcular”, o sistema realiza o cálculo e preenche a tabela inferior com o produto e seu preço de venda. Esta fase é ilustrada na Figura 67.

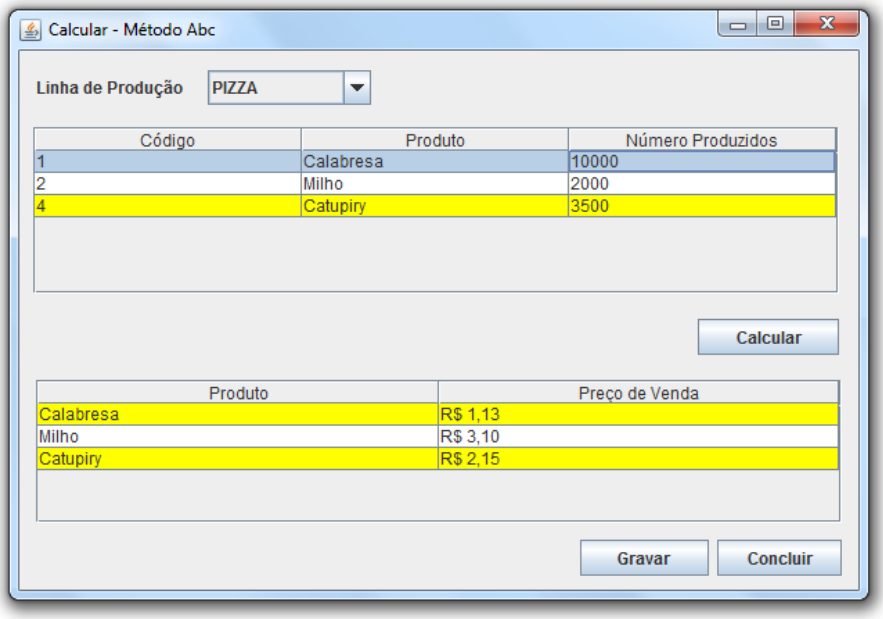


Figura 67 – Efetuação do cálculo pelo Método ABC
Fonte: Autoria Própria.

Após realizado o cálculo, é possível armazená-lo pressionando o botão “Gravar”. O sistema retornará uma mensagem com o sucesso ou insucesso da operação, ilustrado na Figura 68.

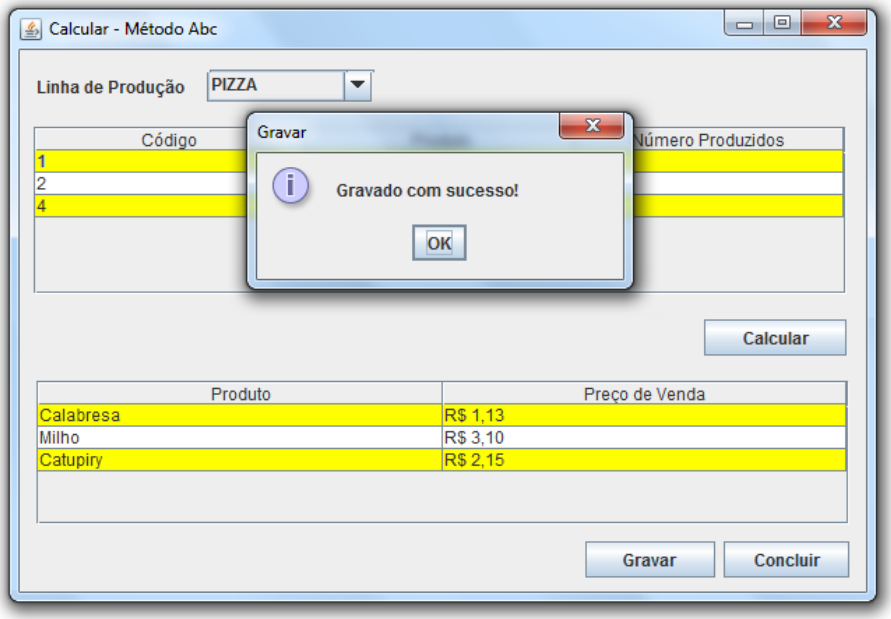


Figura 68 – Mensagem de sucesso ao gravar - Método ABC
Fonte: Autoria Própria.

O processo de cálculo da Tela de Cálculo do Método Custo Pleno (Figura 69) é bem mais simples, pois é necessário apenas informar a porcentagem da margem de lucro que deseja obter no preço final.

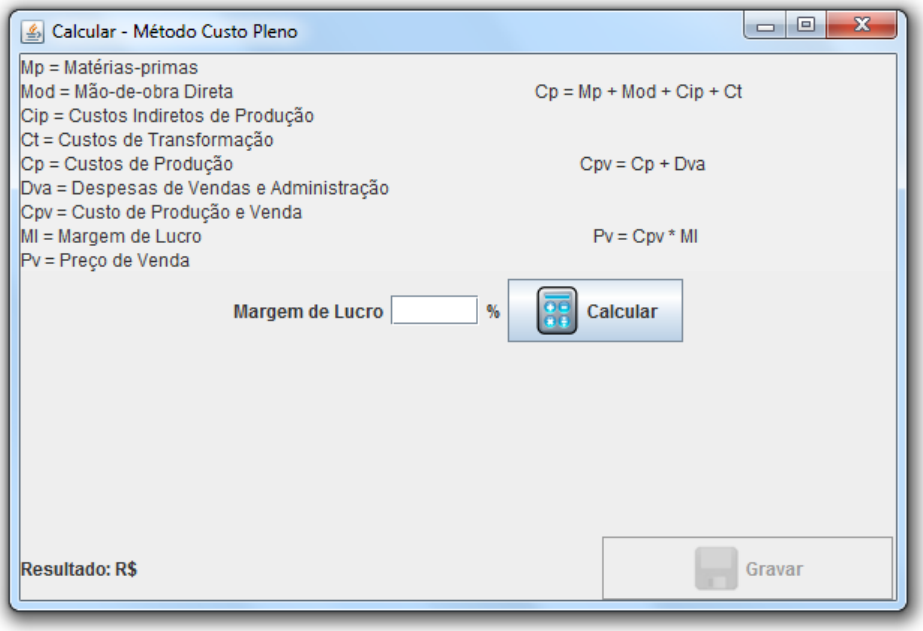


Figura 69 – Tela Cálculo do Método Custo Pleno
Fonte: Autoria Própria.

As informações na parte superior da Tela exibem a fórmula utilizada para efetuar o cálculo. Na parte central da tela há um campo onde é necessário o preenchimento com a margem de lucro e um botão “Calcular” o qual efetua o cálculo do Método Custo Pleno. Caso o valor informado não seja válido, uma mensagem de erro é exibida não deixando completar o processo (Figura 70).

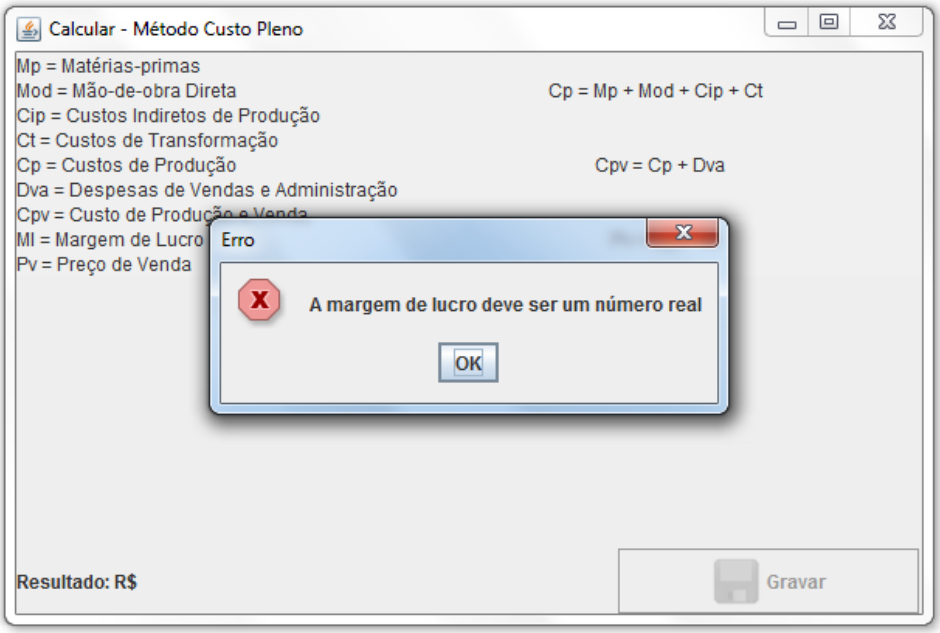


Figura 70 – Mensagem de Erro ao Tentar Efetuar o Cálculo- Método Custo Pleno
Fonte: Autoria Própria.

A Figura 71 ilustra o cálculo levando em consideração uma margem de lucro de 10%.

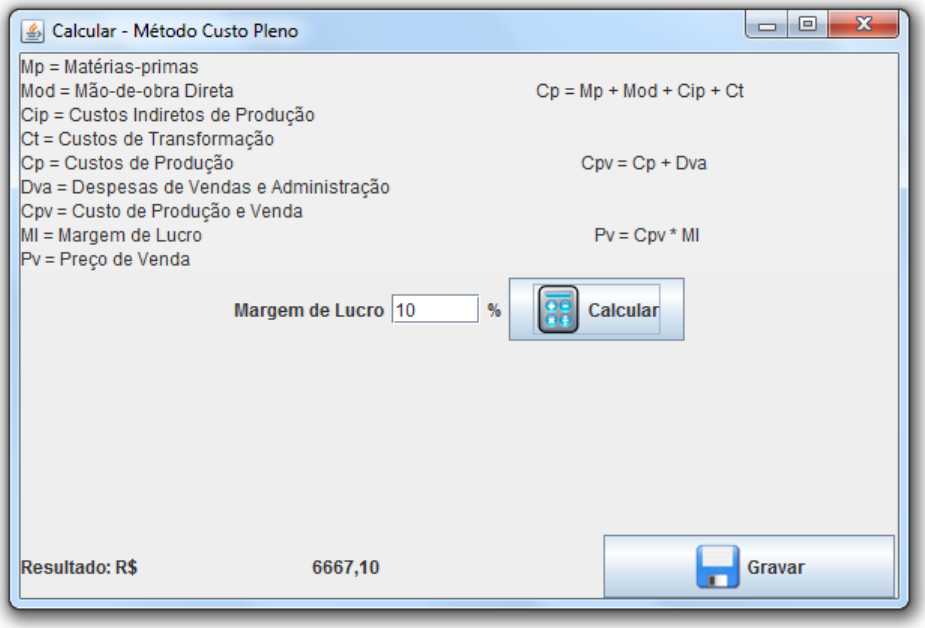


Figura 71 – Cálculo efetuado com 10% de Margem de Lucro
Fonte: Autoria Própria.

Após efetuar o cálculo, pode-se realizar novamente mudando a porcentagem da margem de lucro ou salvar o resultado do preço de venda pressionando o botão

“Gravar”, o qual armazenará o resultado final e retornará uma mensagem de confirmação da operação. Esta mensagem é ilustrada na Figura 72.

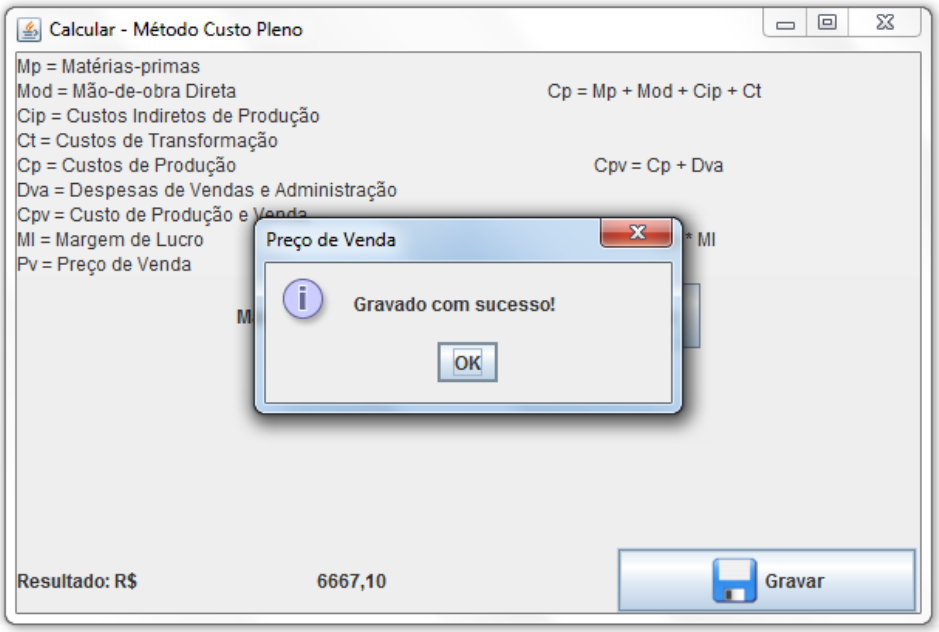


Figura 72 – Mensagem de sucesso ao gravar - Método Custo Pleno
Fonte: Autoria Própria.

A Tela de Cálculo do Método Sebrae, ilustrada na Figura 73, apresenta em sua parte superior as fórmulas para o estabelecimento do preço de venda e na parte inferior o resultado final podendo ser armazenado ao pressionar o botão “Gravar”.

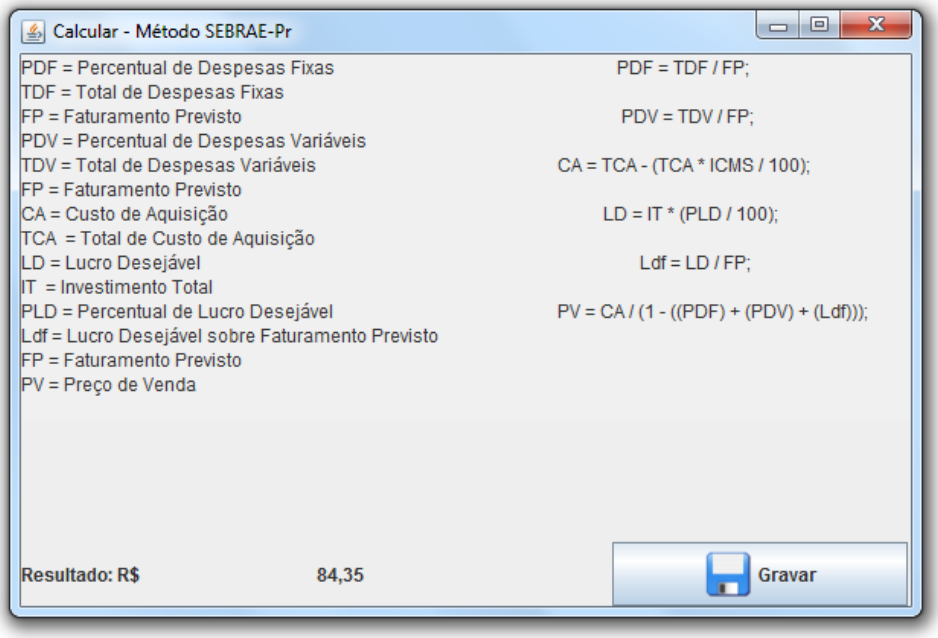


Figura 73 – Tela de Cálculo do Método Sebrae
Fonte: Autoria Própria.

Ao pressionar o botão “Gravar” o sistema irá armazenar o resultado final para uma eventual pesquisa futura. Uma mensagem é exibida com o resultado da operação. Esta mensagem é ilustrada pela Figura 74.

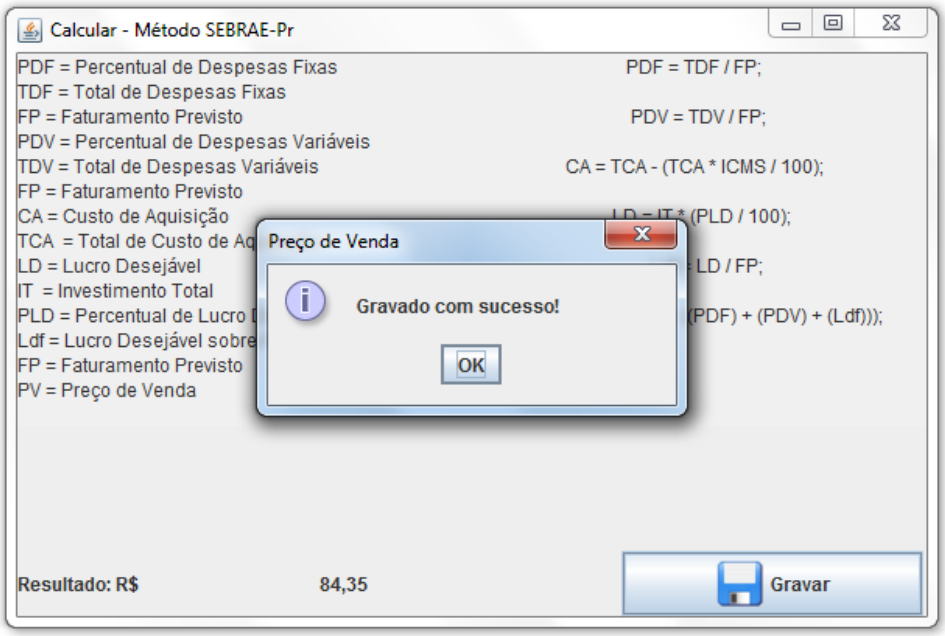


Figura 74 – Mensagem de sucesso ao gravar - Método Sebrae
Fonte: Autoria Própria.

4.2 EXTENSÃO DOS MODELOS

Esta Seção exibe os pontos que devem ser alterados para a inclusão de um novo método para o cálculo de preço de venda. Como o sistema foi desenvolvido em pacotes, esta Seção ilustra os diagramas divididos pelos mesmos. A Subseção 4.2.1 exibe as alterações que devem ser efetuadas no *Subframework Attributes*. A Subseção 4.2.2 indica as modificações para a inserção de um novo método no *Subframework FoodSystem*. E por fim, a Subseção 4.2.3 indica as adequações no *Subframework Calculation*.

4.2.1 Subframework Attributes

Esta Subseção apresenta as alterações que devem ser realizadas no *Subframework Attributes* para acrescentar um novo método de preço de venda e está dividida em 4 (quatro) Subseções. A Subseção 4.2.1.1 descreve as alterações

necessárias no pacote *View*. A Subseção 4.2.1.2 relata as alterações que devem ser realizadas no pacote VO. A Subseção 4.2.1.3 apresenta as alterações no pacote *BusinessRule.* Por fim, a Subseção 4.2.1.4 aborda as mudanças que devem ocorrer no pacote *Persistence.*

4.2.1.1 Pacote *View*

A Figura 75 ilustra as classes que devem ser inseridas do pacote *View* para que um novo método de formação de preço de venda seja adicionado. A classe *Adaptable1,* contendo o estereótipo <<FA>> e conectada a classe *DecoratorAdd*, e a classe *Adaptable2,* contendo o estereótipo <<FA>> e conectada na classe *DecoratorFind,* são as classes que representam o local em que as classes referentes ao novo método a ser inserido devem ser colocadas. Por exemplo, se o método de formação de preço de venda a ser inserido for o Custo Marginal, tem-se a inserção das classes *WindowAddMarginal* e *WindowFindMarginal*.

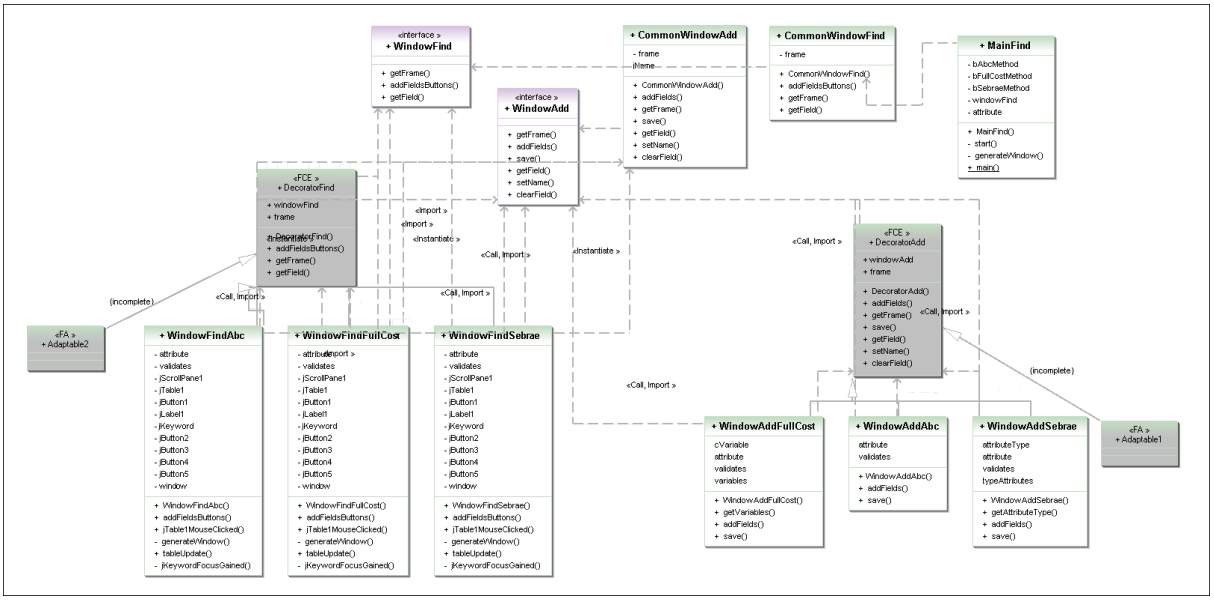


Figura 75 – Classes necessárias no pacote *View* do *Subframework Attributes* para a inclusão de um novo método
Fonte: Autoria Própria.

4.2.1.2 Pacote VO

A Figura 76 exibe a classe *Adaptable1* contendo o estereótipo <<FA>> conectada à classe *Attribute* com o estereótipo <<FCE>>, exibindo assim o que é

necessário modificar no pacote *VO* para a inserção de um novo método. Para a implantação de, por exemplo, o Custo Marginal, a classe *MarginalAttribute* será implementada no local onde se encontra a classe *Adaptable1*.

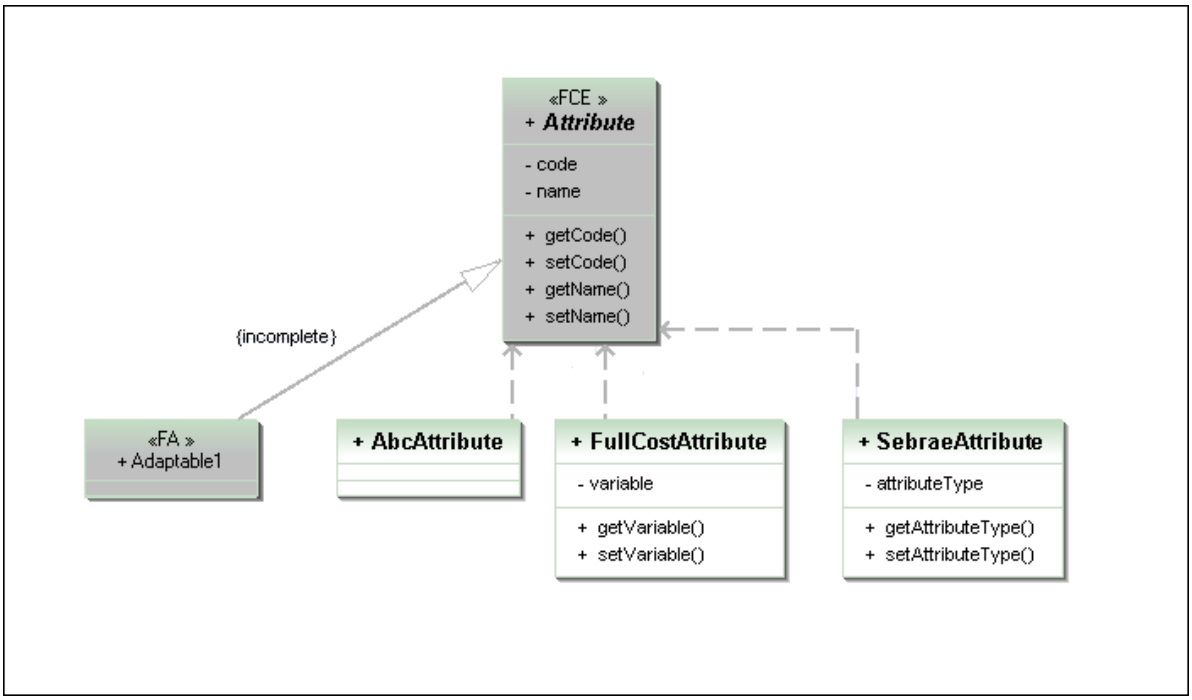


Figura 76 – Classe necessária no pacote VO do *Subframework Attributes* para a inclusão de um novo método
Fonte: Autoria Própria.

4.2.1.3 Pacote *BusinessRule*

A Figura 77 ilustra as classes *Adaptable3* e *Adaptable4* que devem ser inseridas no pacote *BusinessRule* para a inserção de um novo método. Essas classes se encontram com uma cor diferenciada das demais para facilitar sua visualização e podem ser facilmente identificada no diagrama pela sua cor acinzentada.

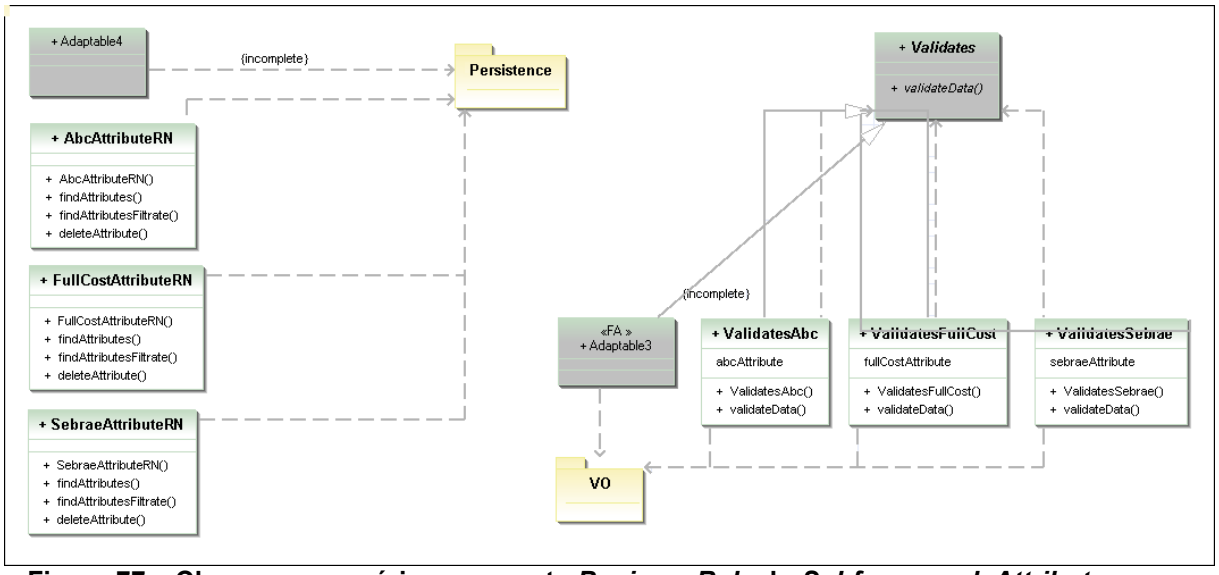


Figura 77 – Classes necessárias no pacote *BusinessRule* do *Subframework Attribute* para a inclusão de um novo método
Fonte: Autoria Própria.

4.2.1.4 Pacote *Persistence*

A Figura 78 exibe as classes necessárias, tanto no pacote *DAO* quanto no sub pacote *DAO.FIREBIRD*, para adicionar um novo método. No pacote *DAO* a classe *Adaptable1* que possui o estereótipo <<FA>> representa onde deve ser inserida uma nova classe e, no pacote *DAO.FIREBIRD,* a classe *Adaptable2* com o estereótipo <<FA>> representa a inserção que deve ocorrer na classe *FBDAOFactory*. Para adicionar, por exemplo, o método Custo Marginal, a classe *Adaptable1* deve ser substituída pela *interface MaginalAttributeDAO* e a classe *Adaptable2* será substituída pela *MarginalAttributeDAOFirebird.*

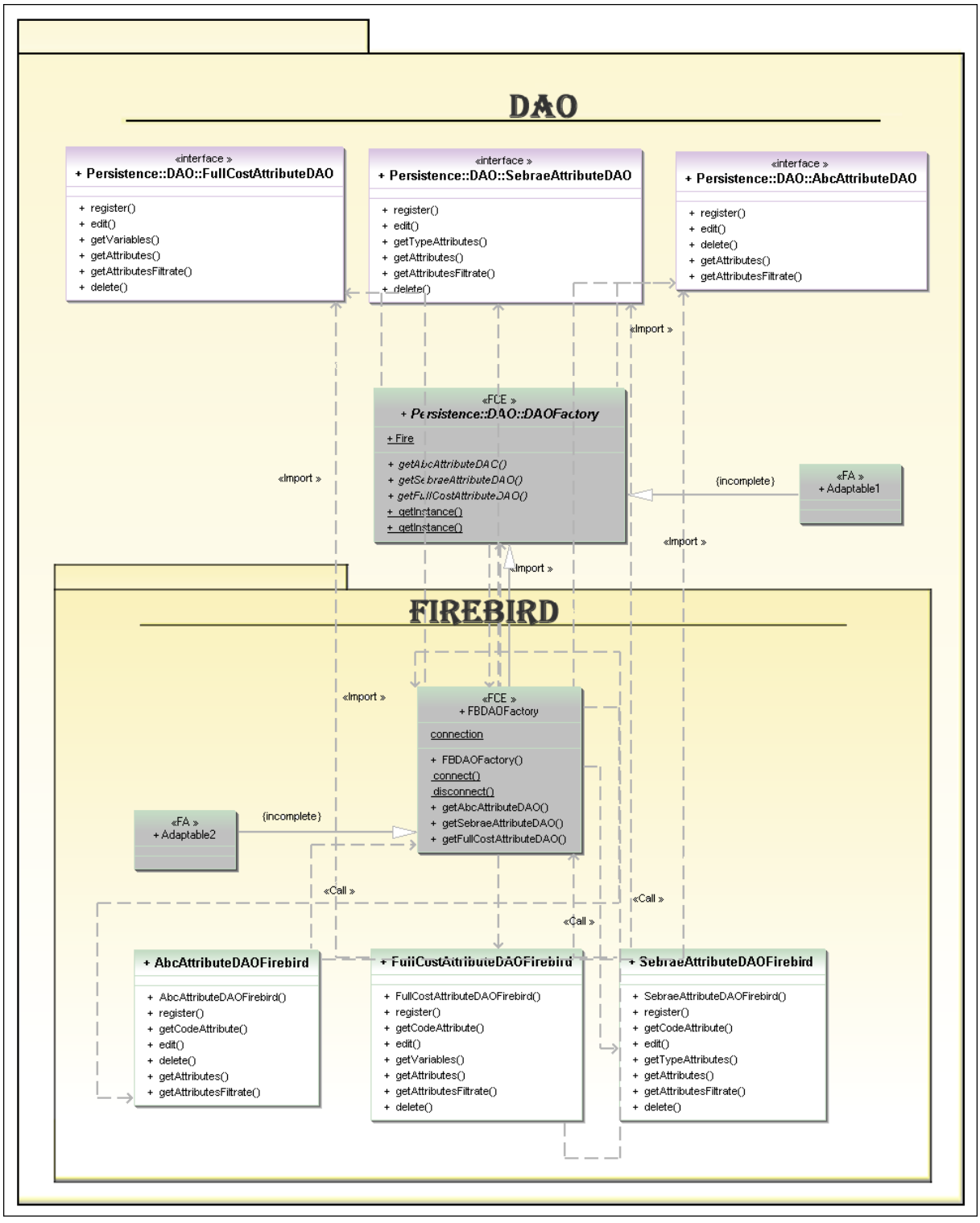


Figura 78 – Classes necessárias no pacote *Persistence* do *Subframework Attributes* para a inclusão de um novo método
Fonte: Autoria Própria.

4.2.2 Subframework FoodSystem

Esta Subseção apresenta as alterações que devem ser realizadas no *Subframework FoodSystem* para acrescentar um novo método de preço de venda e está dividida em 4 (quatro). A Subseção 4.2.2.1 descreve as alterações necessárias no pacote *View*. A Subseção 4.2.2.2 relata as alterações que devem ser realizadas no pacote VO. A Subseção 4.2.2.3 apresenta as alterações no pacote *BusinessRule.* Por fim, a Subseção 4.2.2.4 aborda as mudanças que devem ocorrer no pacote *Persistence.*

4.2.2.1 Pacote View

A Figura 79 exibe a classe *Adaptable1* com o estereótipo <<FA>> a qual é a classe necessária no pacote *View* para a inserção de um novo método no *Subframeworks FoodSystem*. Neste caso, por exemplo, da inserção do Custo Marginal a classe *adaptable1* será substituída pela classe *WindowFoodSystemMarginal*.

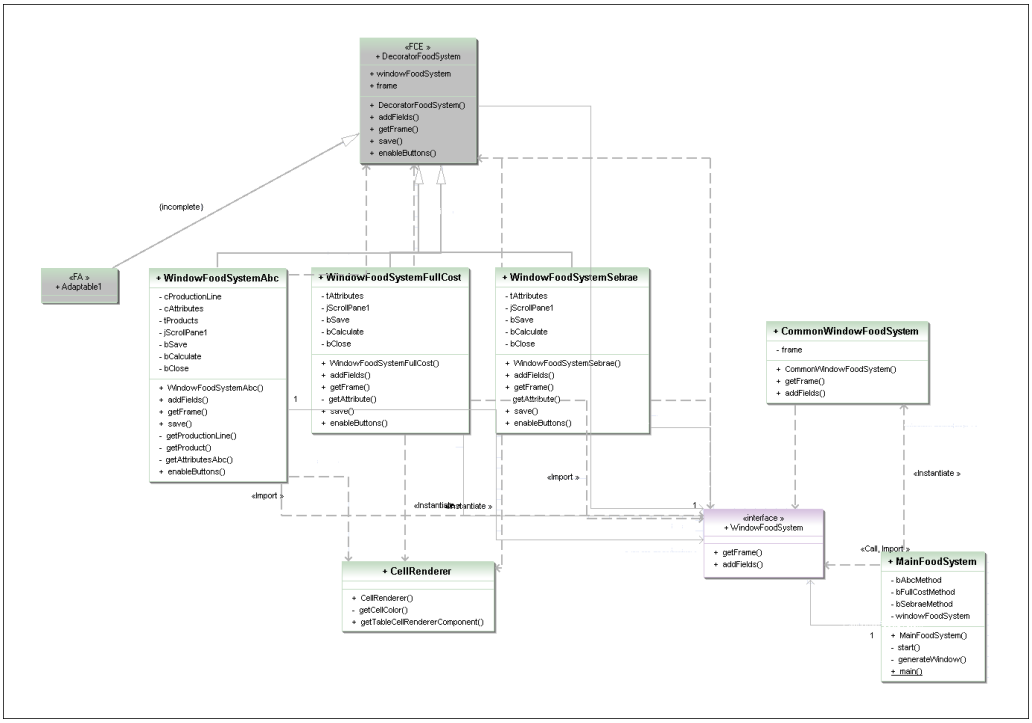


Figura 79 – Classe necessária no pacote *View* do *Subframework FoodSystem* para a inclusão de um novo método
Fonte: Autoria Própria.

4.2.2.2 Pacote VO

A Figura 80 ilustra a classe *Adaptable1* na cor cinza se destacando no modelo, essa classe no pacote VO indica onde supostamente será adicionada uma nova classe, caso mais um método seja adicionado.

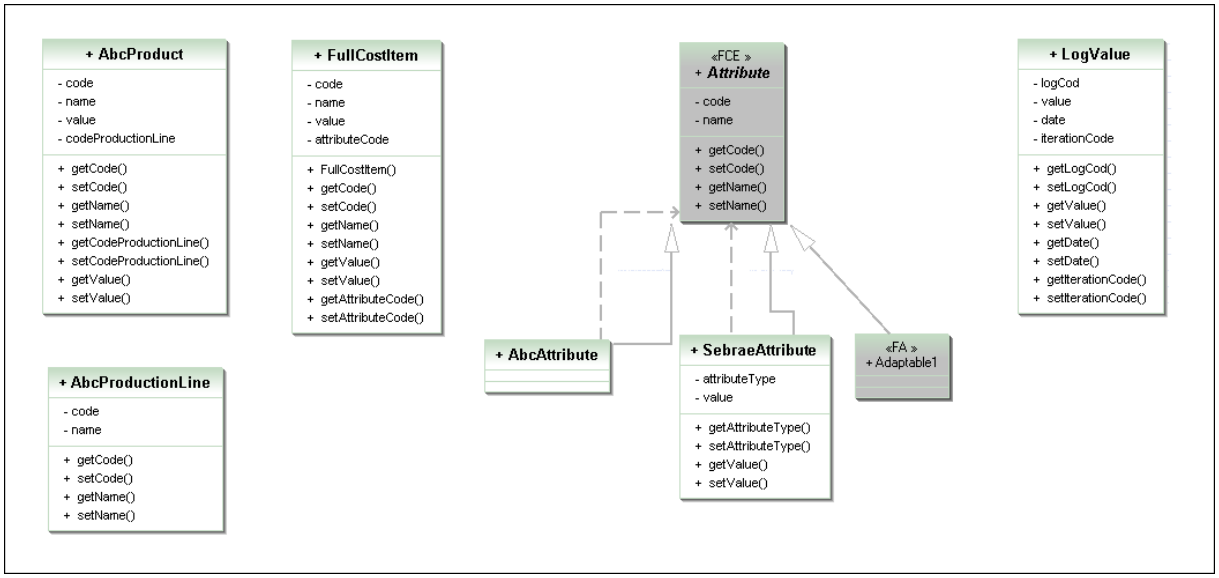


Figura 80 - Classe necessária no pacote *VO* do *Subframework FoodSystem* para a inclusão de um novo método
Fonte: Autoria Própria.

4.2.2.3 Pacote BusinessRule

A Figura 81 exibe o local para inserção de novas classes no pacote *BusinessRule* caso novos métodos sejam adicionados, são as classes: *Adaptable1* e *Adaptable2,* que são representadas pelo estereótipo <<FA>>.

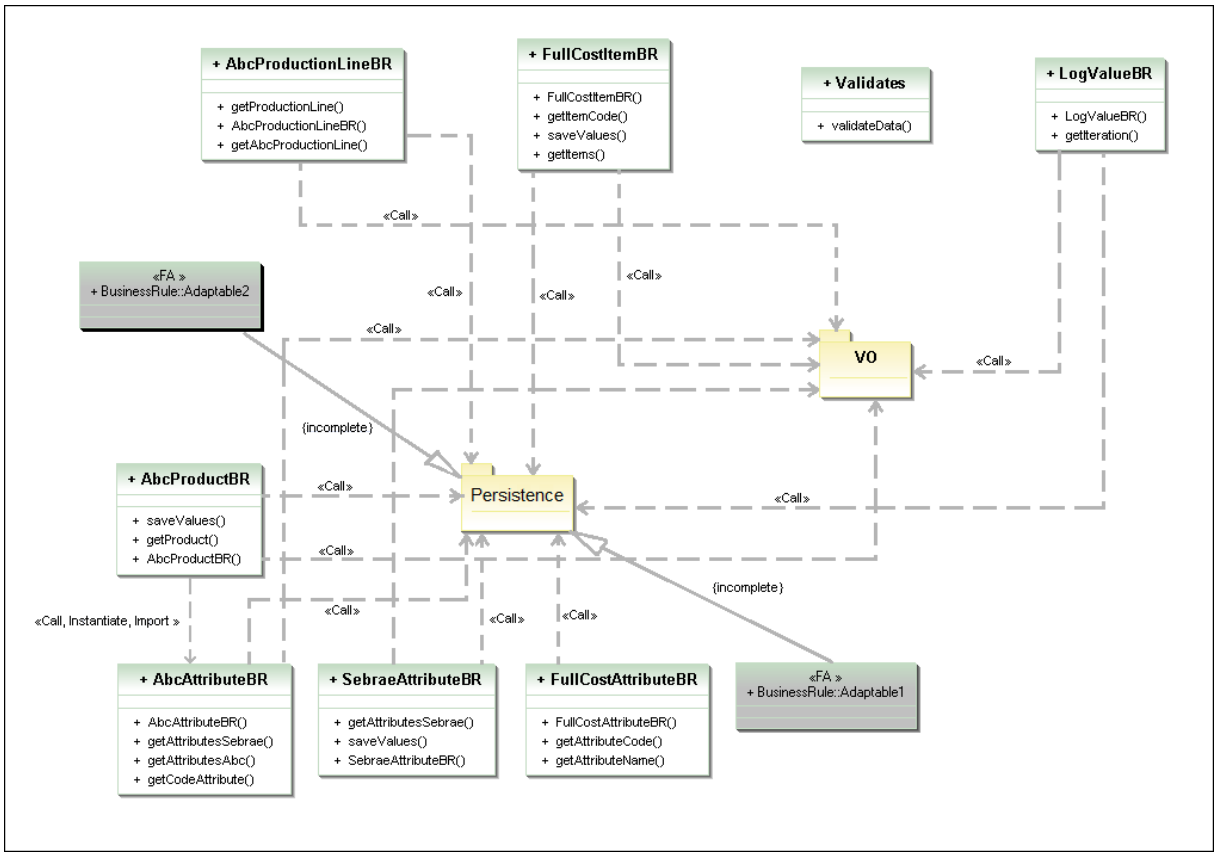


Figura 81 - Classes necessárias no pacote *BusinessRule* do *Subframework FoodSystem* para a inclusão de um novo método
Fonte: Autoria Própria.

4.2.2.4 Pacote Persistence

As classes que devem ser inseridas no pacote *Persistence* no *Subframework FoodSystem* estão presentes na Figura 82 com o estereótipo <<FA>> e na coloração acinzentada, são elas: *Adaptable1* e *Adaptable2.* Essas classes representam as supostas localizações de futuras classes adicionadas para a inserção de novos métodos, como o Custo Marginal.

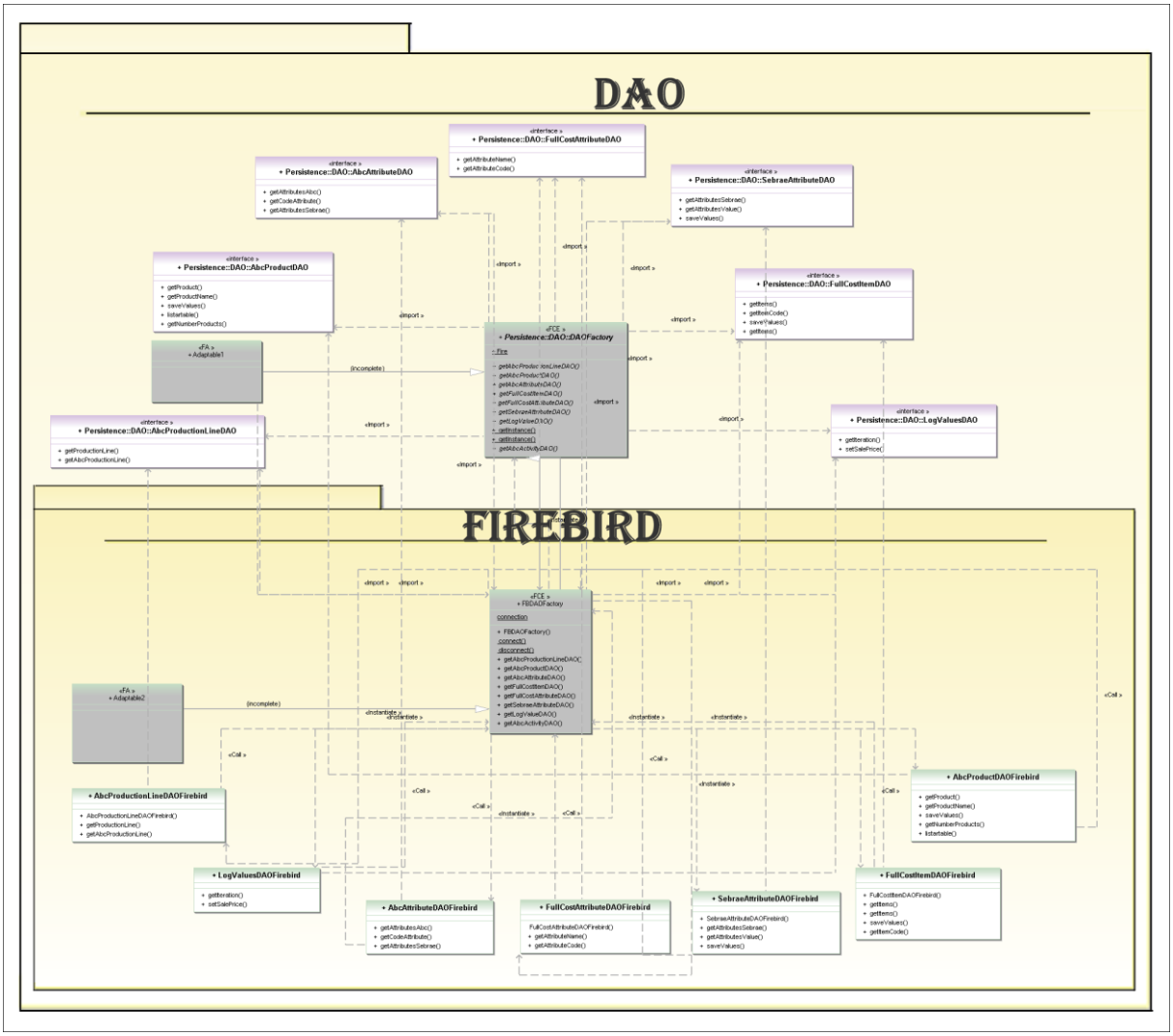


Figura 82 - Classes necessárias no pacote *Persistence* do *Subframework FoodSystem* para a inclusão de um novo método
Fonte: Autoria Própria.

4.2.3 Subframework Calculation

Esta Subseção apresenta as alterações que devem ser realizadas no *Subframework Calculation* para acrescentar um novo método de preço de venda e está dividida em 4 (quatro) Subseções. A Subseção 4.2.3.1 descreve as alterações necessárias no pacote *View*. A Subseção 4.2.3.2 relata as alterações que devem ser realizadas no pacote VO. A Subseção 4.2.3.3 apresenta as alterações no pacote *BusinessRule.* Por fim, a Subseção 4.2.3.4 aborda as mudanças que devem ocorrer no pacote *Persistence.*

4.2.3.1 Pacote View

A Figura 83 exibe a classe *Adaptable1* com o estereótipo <<FA>> a qual é a classe necessária no pacote *View* para a inserção de um novo método no *Subframework Calculation*. Neste caso, por exemplo, da inserção do Custo Marginal a classe *adaptable1* será substituída pela classe *WindowCalculateMarginal*.

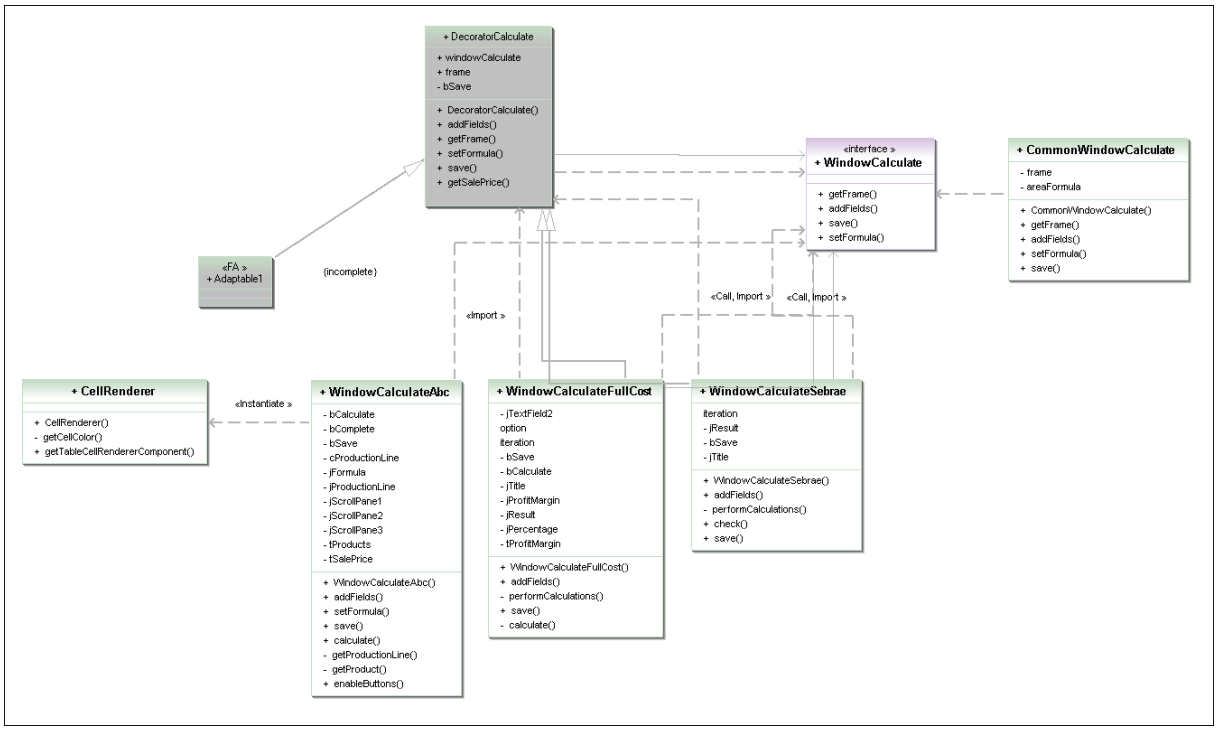


Figura 83 - Classe necessária no pacote *View* do *Subframework Calculation* para a inclusão de um novo método
Fonte: Autoria Própria.

4.2.3.2 Pacote VO

A Figura 84 exibe a classe *Adaptable1* com o estereótipo <<FA>> e com coloração acinzentada representando a localização de uma futura classe caso um novo método seja adicionado, como no exemplo, da inserção do Custo Marginal, onde a classe *Adaptable1* será substituída pela classe *MarginalAttribute*.

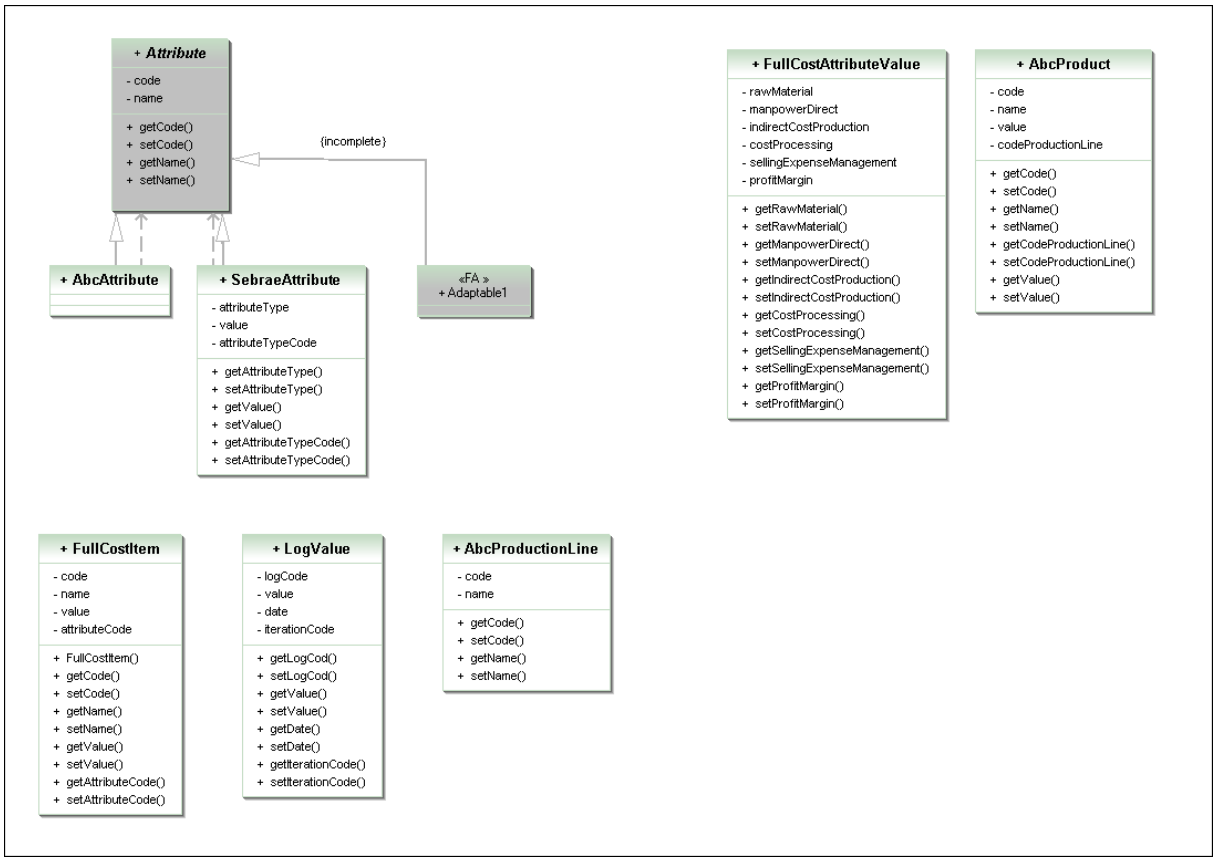


Figura 84 - Classe necessária no pacote VO do *Subframework Calculation* para a inclusão de um novo método
Fonte: Autoria Própria.

4.2.3.3 Pacote BusinessRule

A Figura 85 exibe a classe *Adaptable1* que deve ser inserida no pacote *BusinessRule* para a inserção de um novo método. Essa classe se encontra com uma cor diferenciada das demais para facilitar sua visualização, pode ser facilmente identificada no diagrama pela sua cor acinzentada. Sua posição indica onde uma classe deve ser inserida caso um novo método seja adicionado.

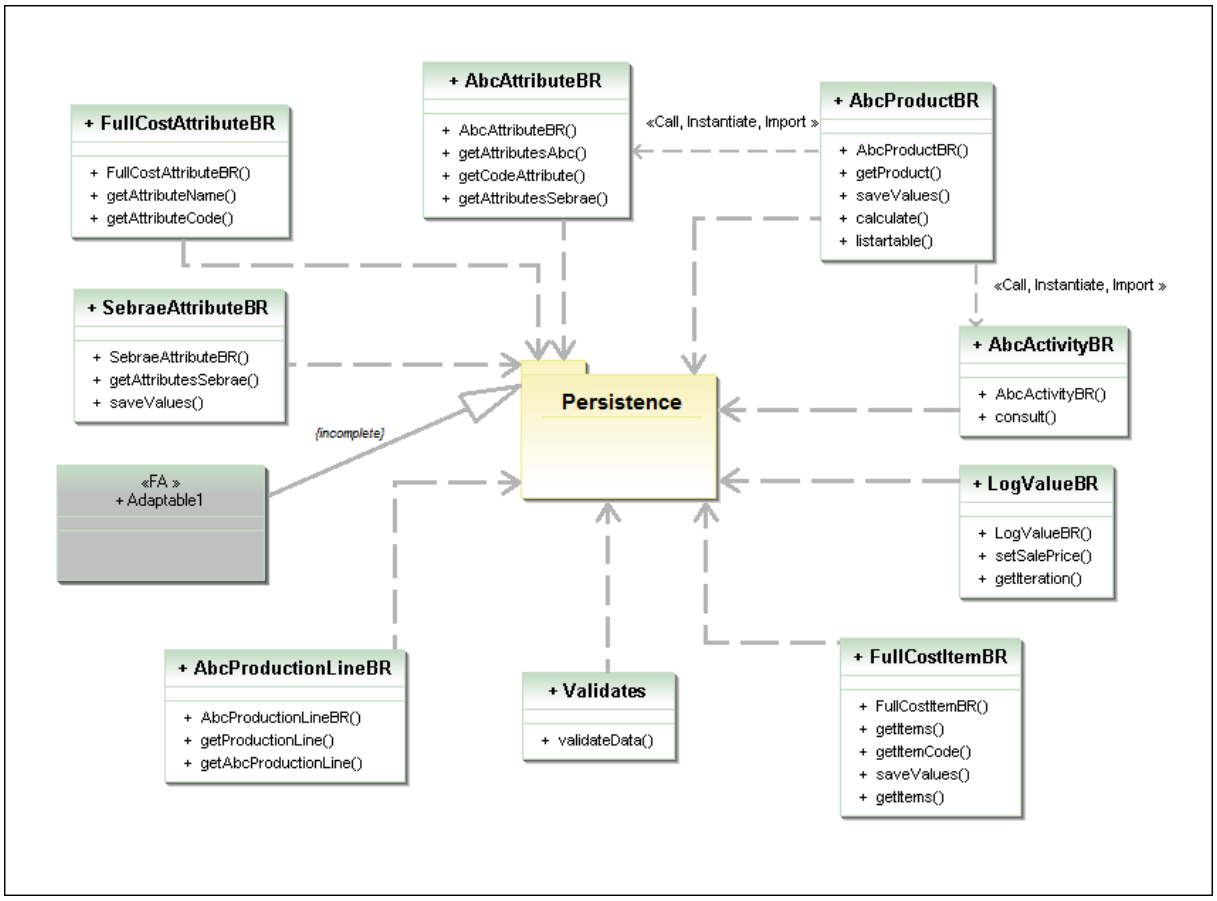


Figura 85 - Classe necessária no pacote *BusinessRule* do *Subframework Calculation* para a inclusão de um novo método
Fonte: Autoria Própria.

4.2.3.4 Pacote Persistence

A Figura 86 exibe as classes necessárias para adicionar um novo método no pacote *Persistence*. No pacote *DAO* é representada pela classe *Adaptable1* que possui o estereótipo <<FA>> e no pacote *DAO.FIREBIRD* é representada pela classe *Adaptable2* com o estereótipo <<FA>> que está conectada a classe *FBDAOFactory*.

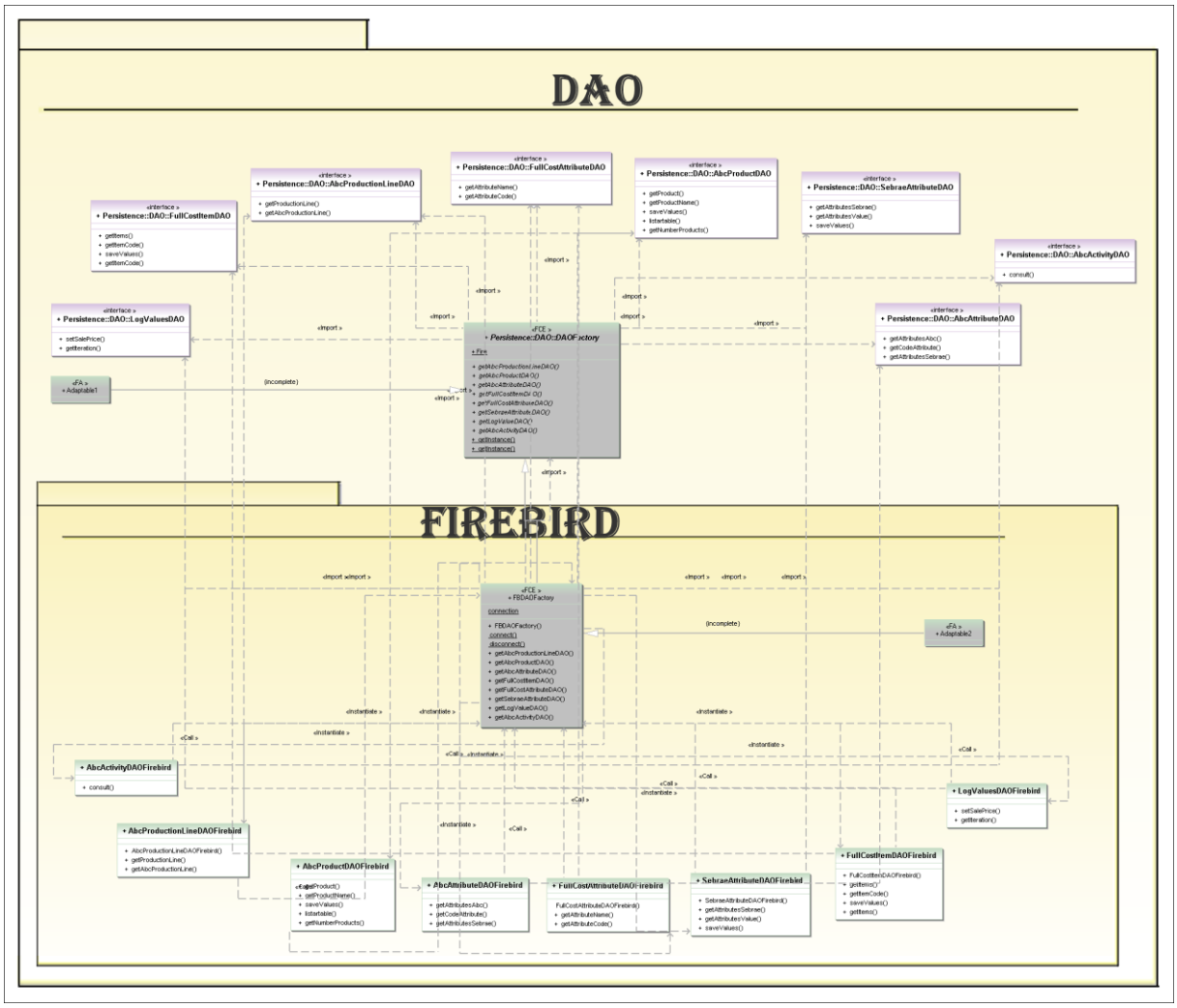


Figura 86 - Classes necessárias no pacote *Persistence* do *Subframework Calculation* para a inclusão de um novo método
Fonte: Autoria Própria.

4.3 VANTAGENS E DESVANTAGENS DO DESENVOLVIMENTO DE FRAMEWORKS

Umas das maiores e melhores vantagens concretizadas nessa técnica é a facilidade na reutilização de código, sem dúvida complexo de se desenvolver no início, porém, uma vez implementado, esse código é reutilizado inúmeras vezes, compensando assim a dificuldade inicial, como a implementação de padrões de projeto, técnica essa que exige um conhecimento específico e um trabalho mais elaborado. Após a primeira instância programada, os próximos passos se limitam em melhorar o que já foi desenvolvido, acrescentando qualidades positivas no trabalho, contudo, um bom planejamento deve ser levantado para que todas as suas

vantagens possam ser alcançadas, garantindo assim também um alto nível de estabilidade. Profissionais que utilizam técnicas como essa, adquirem um conhecimento que pode ser utilizado também em outras técnicas, aprimorando assim, cada vez mais o conhecimento e qualidade do profissional.

5 CONCLUSÃO

O *framework* proposto atende os requisitos no domínio de preço de venda, pois foi construído contemplando três métodos de custeio. Contempla também as necessidades do ponto de vista do gestor, pois o *framework* contém vários métodos de formação de preço de venda, podendo utilizar o sistema para calcular o preço de venda de um serviço ou produto em métodos diferentes.

A arquitetura geral do *framework* foi desenvolvida procurando separar os pontos de estabilidade e de flexibilidade, os quais ficaram agrupados na base – pontos comuns entre os métodos – e os específicos – características específicas de cada método –. Esta separação foi possível porque aplicou-se o Processo Dirigido por Responsabilidades para Desenvolvimento de *Framework* de Domínio (PDR-DFD).

Durante a modelagem foram identificados seis *subframeworks*: *Production Line, Activity, Product, Attributes, Food System* e *Calculation*, dos quais foram implementados os três últimos.

Mostrou-se também que, devido ao fato da modelagem ter sido desenvolvida baseado nos critérios de *framework*, a extensão do modelo para um novo método foi facilitada pois os elementos que devem ser adicionados, implementados ou alterados estão explicitados nos diagramas, conforme relatado no Capítulo 4.

5.1 TRABALHOS FUTUROS

A seguir são apresentadas algumas sugestões para trabalhos futuros:

- Implementar os *subframeworks Production Line, Activity* e *Product.*
- Adicionar novos métodos ao modelo: o preço de venda pode ser determinado também pelos métodos: Marginal e por Retorno por Investimento. Por isso, esses métodos podem ser acrescentados na estrutura do *framework* proposto, que atualmente implementou os métodos Custeio Baseado em Atividades (ABC), Sebrae e Custo Pleno;
- Refatorar a camada *View* para web: o *framework* desenvolvido por esse trabalho foi implementado utilizando o pacote Swing do Java. Para que o mesmo possa ser disponibilizado em um ambiente *WEB* é necessário

refatorar as classes Swing para componentes cliente-servidor como, por exemplo, *Struts*.

- Testar os *Subframeworks:* aplicar técnicas de testes de software para validar os *Subframeworks* implementados.
- Aplicar o Metapadrão *Unification* na Classe *DAOFactory*: para aumentar a flexibilidade do framework proposto é necessário que alguns metapadrões sejam implementados ao modelo.

REFERÊNCIAS

ANDRADE, V. C.; CAPELLER, P. E. B; MATOS, S. N. **Identificação dos Pontos de Estabilidade e Flexibilidade no Modelo de Requisitos dos Métodos de Preço de Venda**. In: Congresso Sul Brasileiro de Computação, 2010, Criciúma. V SULCOMP, 2010.

BASS, L. AND CLEMENTS, P. AND KAZMAN, R. **Software Architecture in Practice**, Addison Wesley.2003.

BORNIA, A. C. **Análise gerencial de custos – aplicação em empresas modernas**. Porto Alegre: Bookman, 2002.

BRAGA, R. T. V. And Masiero, P. C. **A Process for Framework Construction Based on a Pattern Language.** In: Annual International CSAC, 26., 2002. Oxford: IEEE. p. 615-620.

BUTLER, G. AND XU, L. **Cascaded Refactoring for Framework Evolution**, In: Proc. SSR'01, May, Toronto, Ontario, Canada. ACM 1-58113-358-8/01/0005, p. 51-57. 2001.

CAPELLER, P. E. B; ANDRADE, V. C.; SILVA, R. A.; MATOS, S. N. **Uma Adaptação da Linguagem F-UML em uma Ferramenta Gratuita de Modelagem**. In: Congresso Sul Brasileiro de Computação, 2010, Criciúma. V SULCOMP, 2010.

COGAN, S. **Activy-Based Costing (ABC) – a ponderosa estratégia empresarial.** São Paulo: Pioneira, 1995.

COGAN, S. **Custos e preços: formação e análise.** São Paulo: Pioneira, 1999.

CRAZUSKI, A; FEITOSA, L. B; CORDEIRO, T, L**. Identificação dos pontos de estabilidade e de flexibilidade dos métodos para o estabelecimento de preço de venda.** Ponta Grossa, PR, 2008. 66 f. Trabalho de Conclusão de Curso (Graduação) - UTFPR. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Ponta Grossa, 2008.

DOMENICO, G.B. Di; LIMA, P.C. **Gestão de custos baseada em atividades em um ambiente agrícola.** IV Congresso Internacional de Custos - II Congresso Brasileiro de Gestão Estratégica de Custos Local: Campinas. 1995.

FAYAD, M., SCHMIDT, D. AND JOHNSON, R. **Building Application Frameworks – Object-Oriented Foundations of Framework Design**, Wiley, p.688. 1999.

FONTOURA, M.; PREE, W.; RUMPE, B. UML-F: **A modeling language for object-oriented frameworks.** In: European Conference on Object-Oriented Programming, 2000, Sophia Antipolis and Cannes, France. Proceedings ECOOP - Lecture Notes in Computer Science 1850 Springer-Verlag, 2000, p.63-82.

GAMMA, E. HELM, R. JOHNSON, R. VLISSIDES, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Reading, MA: Addison-Wesley, 1994.

GURA, E. R; CARMO, E. R. **Contextualização de Metapadrões no Desenvolvimento de Aplicações Desktop.** Ponta Grossa, PR, 2009. 85 f. Trabalho de Conclusão de Curso (Graduação) - UTFPR. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Ponta Grossa, 2009.

JOHNSON, R. E. **How to design frameworks,** In: Object-Oriented Programming Systems, Languages and Applications Conference – OOPSLA, Washington proceedings, 1993.

JOHNSON, R. E., FOOTE, B. **Designing reusable classes.** Journal of Object-Oriented Programming, [S.l.], p.22-35, June/July 1988.

LANDIN, N.; NIKLASSON, A. **Development of Object-Oriented Frameworks**, Department of Communication System.Lund Institute of Technology, Lund University.Lund, Sweden, 1995.

MARTINS, E. **Contabilidade de Custo**. 6ª Ed., São Paulo: Atlas, 2003. P.220

MATOS, S. N. AND FERNANDES, C. T. **Defining the architectural design of frameworks through a group of subframeworks created from frozen and hot spots.** In: International Conference on Software Engineering Advances, 2006, Tahiti: IEEE Computer Society Press.

MATOS, S. N. AND FERNANDES, C. T. **Using Responsibilities for Early Identification of Hot Spot Reused in Framework Modeling.** In: IEEE International Workshop on Security, Trust, and Privacy for Software Applications. 3. Turku: IEEE Computer Society Press. 2008a.

MATOS, S. N. **Um Método Dirigido por Responsabilidades para Obtenção Antecipada de Pontos de Estabilidade e de Flexibilidade no Desenvolvimento de Frameworks de Domínio.** 274f. Tese de Doutorado, ITA, 2008, São José dos Campos.

MATOS, S. N.; FERNANDES, C. T. **Abordagem dirigida por responsabilidades voltada ao desenvolvimento de frameworks de domínio.** São José dos Campos: ITA, 2007b. p.29. (CTA/ITA-IEC/RP-002/2007)

MATTSSON, M. **Evolution and composition of object-oriented frameworks.**224f.PhD Thesis– UniversityofKarlskrona/Ronneby, Sweden, 2000.

MENDES, A. **Arquitetura de software: desenvolvimento orientado a arquitetura.** Rio de Janeiro: Campus, 2002. 212p.

NAKAGAWA, M. ABC – **Custeio baseado em atividades.** São Paulo: Atlas, 1995.

NASCIMENTO, D. T. do; VARTANIAN, G. H. **O método pleno – uma abordagem conceitual.** Revista CRC-SP nº 09 Set/1999.

NATSAM CONSULTARIA (Brasil). **Hipercusto.** Disponível em: <www.natsam.com.br/html/hipercusto>. Acesso em: 25 nov. 2008. http://www.labiutil.inf.ufsc.br/ergolist/rec.htm

PADOVEZE, C. L. **Contabilidade Gerencial: um enfoque em sistema de informação contábil.** São Paulo: Atlas, 1997. p.46.

PARIS, M. **Reuse-based Layering: a Strategy for Architectural Framework for Learning Technologies,** In: IEEE Conferece on Advanced Learning Technologies (ICALT'04). 2004.

PIMENTA, M. L.; ROCHA, M.; LEMES, S. **Aplicação do método ABC no cultivo de hortaliças na região do Alto Paranaíba.** Custos e @gronegócio Online, v. 3, p. 2-21, 2007. Disponível em:: http://www.custoseagronegocioonline.com.br; Série: 2; ISSN/ISBN: 18082882.

PREE, W**. Hot-spot-driven development.** In: Fayad, M., Johnson, R. and Schmidt, D. "Building application frameworks: object-oriented foundations of framework design". NY: John Wiley and Sons. p. 379-393. 1999.

ROGERS, G. F. **Framework - Based Software Development in C++**. New Jersey: Prentice - Hall, 1997.

SÁ, C. A. O **Método de Custeio por Absorção e o Método de Custeio Variável**. Disponível em <http://carlosalexandresa.com.br>. Acessado em: 22 out de 2009.

SEBRAE. **Cálculo do preço de venda.** Disponível em <http://www.sebraepr.com.br>. Acessado em: 23 out de 2009.

SILVA, C. L. da. **Competitividade: mais que um objetivo, uma necessidade.** In: Revista FAE BUSINESS, Blumenau, nº1, Nov 2001.

SILVA, R. P. **Suporte ao desenvolvimento e uso de frameworks e componentes.** Tese de doutorado. Porto Alegre: UFRGS/II/PPGC, mar. 2000. 262p.

SILVA, Ricardo P. e, PRICE, R. T. **A busca de generalidade, flexibilidade e extensibilidade no processo de desenvolvimento de frameworks orientados a objetos.** In: Proceedingsof Workshop Iberoamericano de Engenharia de Requisitos e Ambientes de Software (IDEAS'98). Torres: apr. 1998. v.2, p.298-309.

SILVA, V. T. **Padrões de Design.** Disponível em: www.lia.ufc.br/~eti/menu/modulos/ Pattern/aulaDesignPatternsI4pp.pdf. Acesso em ago-2010.

SILVESTRE, W. C. **Sistema de custos ABC: uma visão avançada para tecnologia de informação e avaliação de desempenho.** São Paulo: Atlas, 2002.

TALIGENT. **Building object-oriented frameworks,** TaligentInc.white paper, 1994.

TANG, A., HAN, J. AND CHEN, P. **A Comparative Analysis of Architecture Frameworks,** In: Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC´04). 2004.

WIRFS-BROCK, A. **Designing Reusable Designs - Experiences Designing Object-Oriented Framework.** In: European Conference on Object-Oriented Programming and Conference on Object-Oriented Programming: Systems, Languages, and Applications, 1990, Ottawa. Proceedings of the ECOOP/OOPSLA, New York, NY: ACM Press, 1990, p.19-24.

WIRFS-BROCK, R. And MCKEAN, A. **Object Design: Roles, Responsabilities, and Collaborations,** Addison Wesley. 2003.

YASSIN, A., FAYAD, M. E. **Application frameworks: A survey**. In: FAYAD, M. E., JOHNSON, R. E. Domain-Specific Application Frameworks: Frameworks Experience by Industry. New York: John Wiley& Sons, 2000. Cap. 29 p.615-632.

APÊNDICE A - Modelagem do Subsistema “Gerenciar Atributo” do Método ABC

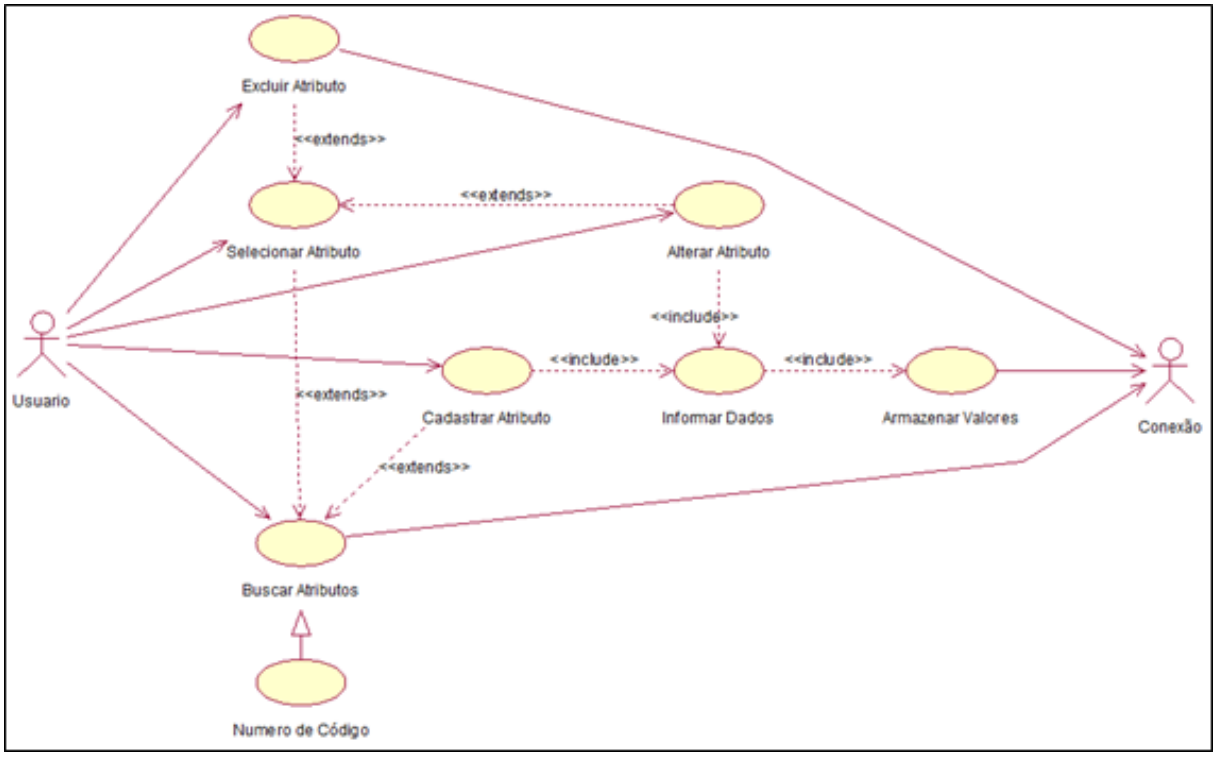


Figura 87 - Expansão do subsistema "Gerenciar Atributo" do método ABC
Fonte: Autoria Própria.

APÊNDICE B - Modelagem do Subsistema “Cadastrar Atributos” do Método Sebrae-PR

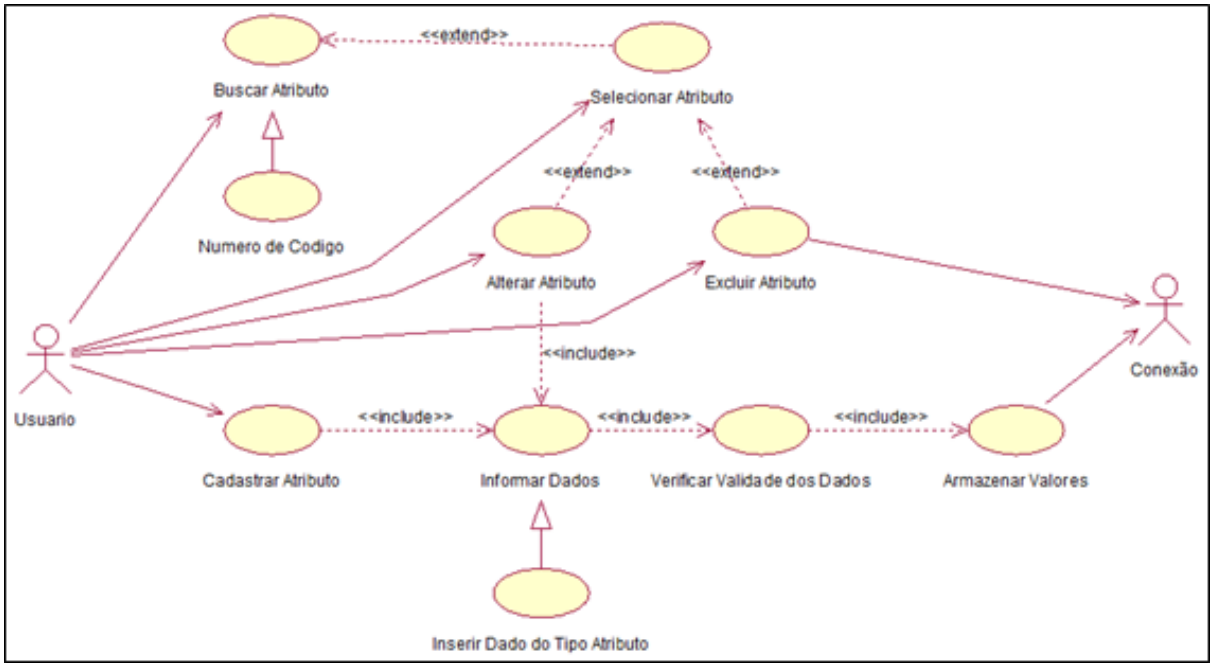


Figura 88 - Expansão do subsistema “Cadastrar Atributos” do método Sebrae-PR
Fonte: Autoria Própria.

APÊNDICE C - Modelagem do Subsistema “Cadastro de Atributos” do Método CUSTO PLENO

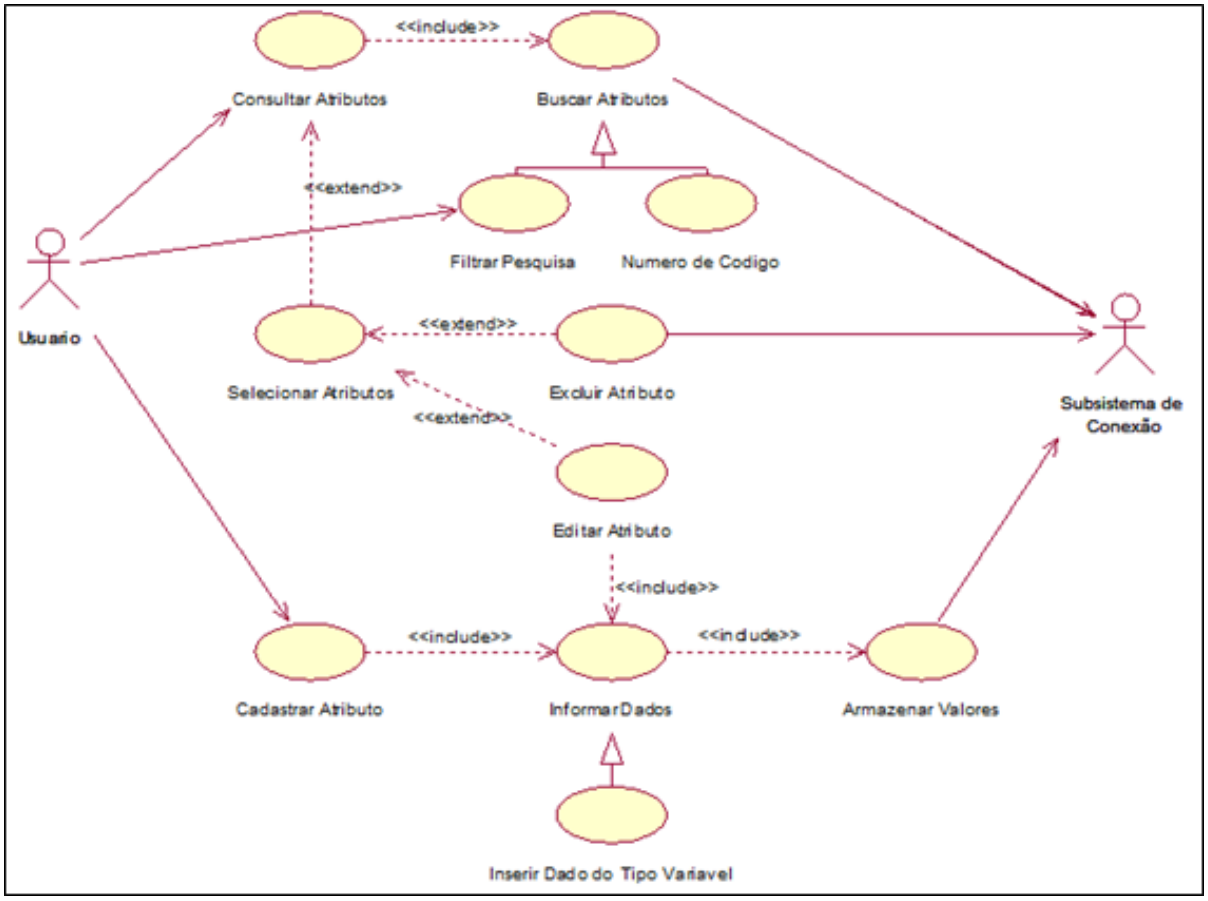


Figura 89 - Expansão do subsistema "Cadastro de Atributos" do método CUSTO PLENO
Fonte: Autoria Própria.

APÊNDICE D - Tabela de Caso de Uso UML de cada aplicação-exemplo

Actor	Perspective	UseCase	Responsability	Examplename
Usuário	Gerenciar Linha de Produção	Consultar Linha	Exibir Tela de Consulta	ABC
Usuário	Gerenciar Linha de Produção	Consultar Linha	Exibir todas as linhas	ABC
Usuário	Gerenciar Linha de Produção	Consultar Linha	Verificar opção selecionada	ABC
Usuário	Gerenciar Linha de Produção	Consultar Linha	Exibir mensagem de erro	ABC
Usuário	Gerenciar Linha de Produção	Editar Linha	Exibe Tela de Cadastro	ABC
Usuário	Gerenciar Linha de Produção	Editar Linha	Exibe uma linha	ABC
Usuário	Gerenciar Linha de Produção	Editar Linha	Verificar opção selecionada	ABC
Usuário	Gerenciar Linha de Produção	Editar Linha	Exibir mensagem de erro	ABC
Usuário	Gerenciar Linha de Produção	Excluir Linha	Exibe mensagem de confirmação	ABC
Usuário	Gerenciar Linha de Produção	Excluir Linha	Verificar opção selecionada	ABC
Usuário	Gerenciar Linha de Produção	Excluir Linha	Excluir Linha	ABC
Usuário	Gerenciar Linha de Produção	Excluir Linha	Exibir mensagem de erro	ABC
Usuário	Gerenciar Linha de Produção	Cadastrar Linha	Exibir Tela de Cadastro	ABC
Usuário	Gerenciar Linha de Produção	Cadastrar Linha	Verificar opção selecionada	ABC
Usuário	Gerenciar Linha de Produção	Cadastrar Linha	Inserir Linha	ABC
Usuário	Gerenciar Linha de Produção	Cadastrar Linha	Exibir mensagem de erro	ABC
Usuário	Gerenciar Linha de Produção	Cadastrar Linha	Limpar campos	ABC
Usuário	Gerenciar Produto	Consultar Produto	Exibir Tela de Consulta	ABC
Usuário	Gerenciar Produto	Consultar Produto	Exibir Todos os Produtos	ABC
Usuário	Gerenciar Produto	Consultar Produto	Verificar opção selecionada	ABC
Usuário	Gerenciar Produto	Consultar Produto	Exibir mensagem de erro	ABC
Usuário	Gerenciar Produto	Editar Produto	Exibe Tela de Cadastro	ABC
Usuário	Gerenciar Produto	Editar Produto	Exibe um Produto	ABC
Usuário	Gerenciar Produto	Editar Produto	Consultar linhas	ABC
Usuário	Gerenciar Produto	Editar Produto	Verificar opção selecionada	ABC
Usuário	Gerenciar Produto	Editar Produto	Exibir mensagem de erro	ABC
Usuário	Gerenciar Produto	Excluir Produto	Exibe mensagem de confirmação	ABC
Usuário	Gerenciar Produto	Excluir Produto	Verificar opção selecionada	ABC
Usuário	Gerenciar Produto	Excluir Produto	Exibir mensagem de erro	ABC

Usuário	Gerenciar Produto	Cadastrar Produto	Exibir Tela de Cadastro	ABC
Usuário	Gerenciar Produto	Cadastrar Produto	Consultar linhas	ABC
Usuário	Gerenciar Produto	Cadastrar Produto	Verificar opção selecionada	ABC
Usuário	Gerenciar Produto	Cadastrar Produto	Exibir mensagem de erro	ABC
Usuário	Gerenciar Atributo	Consultar Atributo	Exibir Tela de Consulta	ABC
Usuário	Gerenciar Atributo	Consultar Atributo	Exibir todos os atributos	ABC
Usuário	Gerenciar Atributo	Consultar Atributo	Verificar opção selecionada	ABC
Usuário	Gerenciar Atributo	Consultar Atributo	Exibir mensagem de erro	ABC
Usuário	Gerenciar Atributo	Editar Atributo	Exibe Tela de Cadastro	ABC
Usuário	Gerenciar Atributo	Editar Atributo	Exibir um atributo	ABC
Usuário	Gerenciar Atributo	Editar Atributo	Verificar opção selecionada	ABC
Usuário	Gerenciar Atributo	Editar Atributo	Exibir mensagem de erro	ABC
Usuário	Gerenciar Atributo	Excluir Atributo	Exibe mensagem de confirmação	ABC
Usuário	Gerenciar Atributo	Excluir Atributo	Verificar opção selecionada	ABC
Usuário	Gerenciar Atributo	Excluir Atributo	Exibir mensagem de erro	ABC
Usuário	Gerenciar Atributo	Cadastrar Atributo	Exibir Tela de Cadastro	ABC
Usuário	Gerenciar Atributo	Cadastrar Atributo	Verificar opção selecionada	ABC
Usuário	Gerenciar Atributo	Cadastrar Atributo	Exibir mensagem de erro	ABC
Usuário	Gerenciar dados do Atributo	Consultar dados Atributo	Exibir Tela de Consulta	ABC
Usuário	Gerenciar dados do Atributo	Consultar dados Atributo	Exibir todos dados dos atributos	ABC
Usuário	Gerenciar dados do Atributo	Consultar dados Atributo	Verificar opção selecionada	ABC
Usuário	Gerenciar dados do Atributo	Consultar dados Atributo	Exibir mensagem de erro	ABC
Usuário	Gerenciar dados do Atributo	Excluir dados Atributos	Exibe mensagem de confirmação	ABC
Usuário	Gerenciar dados do Atributo	Excluir dados Atributos	Verificar opção selecionada	ABC
Usuário	Gerenciar dados do Atributo	Excluir dados Atributos	Exibir mensagem de erro	ABC
Usuário	Gerenciar dados do Atributo	Cadastrar dados Atributos	Exibir Tela de Cadastro	ABC
Usuário	Gerenciar dados do Atributo	Cadastrar dados Atributos	Consultar produtos	ABC
Usuário	Gerenciar dados do Atributo	Cadastrar dados Atributos	Consultar atributos	ABC
Usuário	Gerenciar dados do Atributo	Cadastrar dados Atributos	Verificar opção selecionada	ABC
Usuário	Gerenciar dados do Atributo	Cadastrar dados Atributos	Exibir mensagem de erro	ABC

Usuário	Gerenciar dados da Atividade	Consultar dados Atividades	Exibir Tela de Consulta	ABC
Usuário	Gerenciar dados da Atividade	Consultar dados Atividades	Exibir todos dados das atividades	ABC
Usuário	Gerenciar dados da Atividade	Consultar dados Atividades	Verificar opção selecionada	ABC
Usuário	Gerenciar dados da Atividade	Consultar dados Atividades	Exibir mensagem de erro	ABC
Usuário	Gerenciar dados da Atividade	Excluir dados Atividades	Exibe mensagem de confirmação	ABC
Usuário	Gerenciar dados da Atividade	Excluir dados Atividades	Verificar opção selecionada	ABC
Usuário	Gerenciar dados da Atividade	Excluir dados Atividades	Exibir mensagem de erro	ABC
Usuário	Gerenciar dados da Atividade	Cadastrar dados Atividade	Exibir Tela de Cadastro	ABC
Usuário	Gerenciar dados da Atividade	Cadastrar dados Atividade	Consultar atividades	ABC
Usuário	Gerenciar dados da Atividade	Cadastrar dados Atividade	Consultar atributos	ABC
Usuário	Gerenciar dados da Atividade	Cadastrar dados Atividade	Verificar opção selecionada	ABC
Usuário	Gerenciar dados da Atividade	Cadastrar dados Atividade	Exibir mensagem de erro	ABC
Usuário	Cálculo do Preço de Venda	Informar dados	Exibir Tela de Calculo	ABC
Usuário	Cálculo do Preço de Venda	Informar dados	Consultar linhas	ABC
Usuário	Cálculo do Preço de Venda	Informar dados	Consultar produtos	ABC
Usuário	Cálculo do Preço de Venda	Informar dados	Exibir mensagem de erro	ABC
Usuário	Cálculo do Preço de Venda	Calcular preço	Consultar valores	ABC
Usuário	Cálculo do Preço de Venda	Calcular preço	Calcular o preço de venda baseado nos valores informados	ABC
Usuário	Cálculo do Preço de Venda	Calcular preço	Exibir o preço de venda	ABC
Usuário	Cálculo do Preço de Venda	Calcular preço	Exibir mensagem de erro	ABC
Subsistema	Conexão	Conectar ao Banco de Dados	Localiza Driver do Banco de Dados	ABC
Subsistema	Conexão	Conectar ao Banco de Dados	Estabelecer nova conexão	ABC
Subsistema	Conexão	Conectar ao Banco de Dados	Exibir mensagem de erro	ABC
Subsistema	Conexão	Desconectar do Banco de Dados	Localiza Driver do Banco de Dados	ABC
Subsistema	Conexão	Desconectar do Banco de Dados	Desconecta do banco de dados	ABC
Subsistema	Conexão	Desconectar do Banco de Dados	Exibir mensagem de erro	ABC
Usuário	Gerenciar Atividade	Consultar Atividade	Exibir Tela de Consulta	ABC
Usuário	Gerenciar Atividade	Consultar Atividade	Exibir todas as atividades	ABC
Usuário	Gerenciar Atividade	Consultar Atividade	Verificar opção selecionada	ABC
Usuário	Gerenciar Atividade	Consultar Atividade	Exibir mensagem de erro	ABC

Usuário	Gerenciar Atividade	Editar Atividade	Exibe Tela de Cadastro	ABC
Usuário	Gerenciar Atividade	Editar Atividade	Exibir uma atividade	ABC
Usuário	Gerenciar Atividade	Editar Atividade	Consultar linhas	ABC
Usuário	Gerenciar Atividade	Editar Atividade	Verificar opção selecionada	ABC
Usuário	Gerenciar Atividade	Editar Atividade	Exibir mensagem de erro	ABC
Usuário	Gerenciar Atividade	Excluir Atividade	Exibe mensagem de confirmação	ABC
Usuário	Gerenciar Atividade	Excluir Atividade	Verificar opção selecionada	ABC
Usuário	Gerenciar Atividade	Excluir Atividade	Exibir mensagem de erro	ABC
Usuário	Gerenciar Atividade	Cadastrar Atividade	Exibir Tela de Cadastro	ABC
Usuário	Gerenciar Atividade	Cadastrar Atividade	Consultar linhas	ABC
Usuário	Gerenciar Atividade	Cadastrar Atividade	Verificar opção selecionada	ABC
Usuário	Gerenciar Atividade	Cadastrar Atividade	Exibir mensagem de erro	ABC
Usuário	Calcular Preço de Venda	Abrir Tela de Cálculo	Exibir Tela de Cálculo	SEBRAE
Usuário	Calcular Preço de Venda	Solicitar Cálculo do Preço de Venda	Buscar valores dos atributos cadastrados no Banco de Dados	SEBRAE
Usuário	Calcular Preço de Venda	Solicitar Cálculo do Preço de Venda	Atribuir valores as variáveis	SEBRAE
Usuário	Calcular Preço de Venda	Solicitar Cálculo do Preço de Venda	Verificar preenchimento de valores	SEBRAE
Usuário	Calcular Preço de Venda	Solicitar Cálculo do Preço de Venda	Efetuar Cálculo do Preço de Venda	SEBRAE
Usuário	Calcular Preço de Venda	Solicitar Cálculo do Preço de Venda	Recuperar valores das variáveis	SEBRAE
Usuário	Calcular Preço de Venda	Solicitar Cálculo do Preço de Venda	Exibir resultado na Tela de Cálculo	SEBRAE
Usuário	Calcular Preço de Venda	Solicitar Cálculo do Preço de Venda	Exibir mensagem de erro	SEBRAE
Usuário	Calcular Preço de Venda	Fechar Tela de Cálculo	Fechar Tela de Cálculo	SEBRAE
Usuário	Cadastrar Atributos	Buscar Atributos	Listar os valores já persistidos anteriormente no Banco de Dados	SEBRAE
Usuário	Cadastrar Atributos	Buscar Atributos	Recuperar valores das variáveis	SEBRAE
Usuário	Cadastrar Atributos	Buscar Atributos	Verificar qual ação executar	SEBRAE
Usuário	Cadastrar Atributos	Buscar Atributos	Exibir mensagem de erro	SEBRAE
Usuário	Cadastrar Atributos	Buscar Atributos	Fechar Tela de Busca	SEBRAE
Usuário	Cadastrar Atributos	Cadastrar Atributos	Exibir Tela de Cadastro de Atributos	SEBRAE
Usuário	Cadastrar Atributos	Cadastrar Atributos	Atribuir valores as variáveis	SEBRAE

Usuário	Cadastrar Atributos	Cadastrar Atributos	Verificar inserção de valores	SEBRAE
Usuário	Cadastrar Atributos	Cadastrar Atributos	Inserir valores no Banco de Dados	SEBRAE
Usuário	Cadastrar Atributos	Cadastrar Atributos	Fechar Tela de Cadastro de Atributos	SEBRAE
Usuário	Cadastrar Atributos	Cadastrar Atributos	Exibir mensagem de erro	SEBRAE
Usuário	Cadastrar Atributos	Excluir Atributos	Emitir mensagem de confirmação de exclusão	SEBRAE
Usuário	Cadastrar Atributos	Excluir Atributos	Tratar exclusão	SEBRAE
Usuário	Cadastrar Atributos	Excluir Atributos	Excluir registro no Banco de Dados	SEBRAE
Usuário	Cadastrar Atributos	Excluir Atributos	Atualizar Tela de Busca de Atributos	SEBRAE
Usuário	Cadastrar Atributos	Excluir Atributos	Exibir mensagem de erro	SEBRAE
Usuário	Cadastrar Atributos	Alterar Atributos	Exibir Tela de Cadastro de Atributos	SEBRAE
Usuário	Cadastrar Atributos	Alterar Atributos	Buscar dados do registro no Banco de Dados	SEBRAE
Usuário	Cadastrar Atributos	Alterar Atributos	Recuperar valores das variáveis	SEBRAE
Usuário	Cadastrar Atributos	Alterar Atributos	Preencher os campos com os valores	SEBRAE
Usuário	Cadastrar Atributos	Alterar Atributos	Verificar valores preenchidos	SEBRAE
Usuário	Cadastrar Atributos	Alterar Atributos	Atribuir valores as variáveis	SEBRAE
Usuário	Cadastrar Atributos	Alterar Atributos	Verificar alteração dos valores	SEBRAE
Usuário	Cadastrar Atributos	Alterar Atributos	Alterar os valores no Banco de Dados	SEBRAE
Usuário	Cadastrar Atributos	Alterar Atributos	Fecha a Tela de Cadastro de Atributos	SEBRAE
Usuário	Cadastrar Atributos	Alterar Atributos	Exibir mensagem de erro	SEBRAE
Usuário	Cadastrar Atributos	Sair	Fechar Tela de Busca de Atributos	SEBRAE
Usuário	Alimentar Sistema	Abrir Tela de Alimentação de Valores	Listar os valores já persistidos anteriormente no Banco de Dados	SEBRAE
Usuário	Alimentar Sistema	Abrir Tela de Alimentação de Valores	Recuperar valores das variáveis	SEBRAE
Usuário	Alimentar Sistema	Abrir Tela de Alimentação de Valores	Preencher os campos com os valores	SEBRAE
Usuário	Alimentar Sistema	Abrir Tela de Alimentação de Valores	Exibir Tela de Alimentação de Valores	SEBRAE
Usuário	Alimentar Sistema	Abrir Tela de Alimentação de Valores	Exibir mensagem de erro	SEBRAE
Usuário	Alimentar Sistema	Fechar Tela de Alimentação de Valores	Fechar Tela de Alimentação de Valores	SEBRAE
Usuário	Alimentar Sistema	Inserir Valores dos Atributos	Verificar valores preenchidos	SEBRAE
Usuário	Alimentar Sistema	Inserir Valores dos Atributos	Atribuir valores as variáveis	SEBRAE

Usuário	Alimentar Sistema	Inserir Valores dos Atributos	Verificar inserção dos valores	SEBRAE
Usuário	Alimentar Sistema	Inserir Valores dos Atributos	Inserir valores no Banco de Dados	SEBRAE
Usuário	Alimentar Sistema	Inserir Valores dos Atributos	Exibir mensagem de erro	SEBRAE
Usuário	Alimentar Sistema	Solicitar Cálculo do Preço de Venda	Exibir Tela de Cálculo	SEBRAE
Usuário	Alimentar Sistema	Alterar Valores dos Atributos	Verificar valores preenchidos	SEBRAE
Usuário	Alimentar Sistema	Alterar Valores dos Atributos	Atribuir valores as variáveis	SEBRAE
Usuário	Alimentar Sistema	Alterar Valores dos Atributos	verificar alteração dos valores	SEBRAE
Usuário	Alimentar Sistema	Alterar Valores dos Atributos	Alterar os valores no Banco de Dados	SEBRAE
Usuário	Alimentar Sistema	Alterar Valores dos Atributos	Exibir mensagem de erro	SEBRAE
Usuário	Validação	Verificar tecla pressionada	Verificar validade do caractere para o campo	SEBRAE
Usuário	Validação	Verificar tecla pressionada	Retornar caractere ao campo	SEBRAE
Subsistema	Validação	Verificar valor da variável	Comparar conforme tipo de variável	SEBRAE
Subsistema	Validação	Verificar valor da variável	Retornar resultado da validação	SEBRAE
Subsistema	Conexão	Conectar ao Banco de Dados	Localizar Driver do Banco de Dados	SEBRAE
Subsistema	Conexão	Conectar ao Banco de Dados	Estabelecer nova conexão	SEBRAE
Subsistema	Conexão	Conectar ao Banco de Dados	Exibir mensagem de erro	SEBRAE
Subsistema	Conexão	Desconectar do Banco de Dados	Localizar Driver do Banco de Dados	SEBRAE
Subsistema	Conexão	Desconectar do Banco de Dados	Desconectar do banco de dados	SEBRAE
Subsistema	Conexão	Desconectar do Banco de Dados	Exibir mensagem de erro	SEBRAE
Subsistema	Conexão	Conectar ao Banco de Dados	Localizar Driver do Banco de Dados	Custo Pleno
Subsistema	Conexão	Conectar ao Banco de Dados	Estabelecer nova conexão	Custo Pleno
Subsistema	Conexão	Conectar ao Banco de Dados	Exibir mensagem de erro	Custo Pleno
Subsistema	Conexão	Desconectar do Banco de Dados	Localizar Driver do Banco de Dados	Custo Pleno
Subsistema	Conexão	Desconectar do Banco de Dados	Desconectar do banco de dados	Custo Pleno
Subsistema	Conexão	Desconectar do Banco de Dados	Exibir mensagem de erro	Custo Pleno
Usuário	Cálculo do Preço de Venda	Informar Dados	Mostrar a Tela de Cálculo	Custo Pleno
Usuário	Cálculo do Preço de Venda	Informar Dados	Chamar "Consultar Atributo"	Custo Pleno
Usuário	Cálculo do Preço de Venda	Informar Dados	Validar dados	Custo Pleno
Usuário	Cálculo do Preço de Venda	Informar Dados	Exibir mensagem de erro	Custo Pleno
Usuário	Cálculo do Preço de Venda	Calcular	Calcular o preço de venda baseado nos valores informados	Custo Pleno

Usuário	Cálculo do Preço de Venda	Calcular	Mostrar o preço de venda calculado	Custo Pleno
Usuário	Cálculo do Preço de Venda	Calcular	Exibir mensagem de erro	Custo Pleno
Usuário	Cadastro de Atributos	Consultar Atributo	Mostrar a Tela de Cadastro de Atributos	Custo Pleno
Usuário	Cadastro de Atributos	Consultar Atributo	Motrar a Tela de Consulta de Atributos	Custo Pleno
Usuário	Cadastro de Atributos	Consultar Atributo	Mostrar os atributos cadastrados no Banco de Dados	Custo Pleno
Usuário	Cadastro de Atributos	Consultar Atributo	Mostrar os dados do Atributo selecionado	Custo Pleno
Usuário	Cadastro de Atributos	Consultar Atributo	Exibir mensagem de erro	Custo Pleno
Usuário	Cadastro de Atributos	Cadastrar Atributo	Mostrar a tela Cadastro de Atributos	Custo Pleno
Usuário	Cadastro de Atributos	Cadastrar Atributo	Habilitar os campos para a inserção dos dados	Custo Pleno
Usuário	Cadastro de Atributos	Cadastrar Atributo	Chamar "Consultar Variável"	Custo Pleno
Usuário	Cadastro de Atributos	Cadastrar Atributo	Verificar ação escolhida	Custo Pleno
Usuário	Cadastro de Atributos	Cadastrar Atributo	Exibir mensagem de erro	Custo Pleno
Usuário	Cadastro de Atributos	Editar Atributo	Mostrar a tela Cadastro de Atributos	Custo Pleno
Usuário	Cadastro de Atributos	Editar Atributo	Mostrar a tela de Consulta de Atributos	Custo Pleno
Usuário	Cadastro de Atributos	Editar Atributo	Mostrar os atributos cadastrados no Banco de Dados	Custo Pleno
Usuário	Cadastro de Atributos	Editar Atributo	Mostrar os dados do Atributo selecionado	Custo Pleno
Usuário	Cadastro de Atributos	Editar Atributo	Habilitar os campos para a inserção dos dados	Custo Pleno
Usuário	Cadastro de Atributos	Editar Atributo	Chamar "Consultar Variável"	Custo Pleno
Usuário	Cadastro de Atributos	Editar Atributo	Verificar ação escolhida	Custo Pleno
Usuário	Cadastro de Atributos	Editar Atributo	Exibir mensagem de erro	Custo Pleno
Usuário	Cadastro de Atributos	Excluir Atributo	Mostrar a tela Cadastro de Atributos	Custo Pleno
Usuário	Cadastro de Atributos	Excluir Atributo	Mostrar a tela de Consulta de Atributos	Custo Pleno
Usuário	Cadastro de Atributos	Excluir Atributo	Mostrar os atributos cadastrados no Banco de Dados	Custo Pleno
Usuário	Cadastro de Atributos	Excluir Atributo	Mostrar os dados do Atributo selecionado	Custo Pleno
Usuário	Cadastro de Atributos	Excluir Atributo	Confirmar com o Usuário se ele realmente deseja excluir o atributo selecionado	Custo Pleno
Usuário	Cadastro de Atributos	Excluir Atributo	Exibir mensagem de erro	Custo Pleno

Quadro 13 –Caso de Uso UML de cada aplicação-exemplo
Fonte: Autoria Própria.

APÊNDICE E - Descrição Textual para os Casos de Uso do Subframework Attributes

Após a elaboração e a análise do Diagrama de Caso de Uso UML-F para o Subsistema *Attributes* (Figura 8) foram feitas as descrições textuais de cada caso de uso a fim de descrever as funcionalidades nele existentes. O Quadro 14 apresenta a descrição textual para o caso de uso “Informar Dados”, o Quadro 15 para “Inserir Dado do Tipo Atributo”, o Quadro 16 para “Inserir Dado do Tipo Variavel”, o Quadro 17 para “Verificar Validade dos Dados”, o Quadro 18 para “Armazenar Valores”, o Quadro 19 para “Buscar Atributo”, o Quadro 20 para “Selecionar Atributo”, o Quadro 21 para “Editar Atributo”, o Quadro 22 para “Desabilitar Atributo”, o Quadro 23 para “Buscar Codigo”, o Quadro 24 para “Filtrar Pesquisa”, o Quadro 25 para “Buscar Variavel” e, por fim, o Quadro 26 para “Buscar Descricao”.

Nome do Caso de Uso	Informar Dados
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
	1. Disponibilizar campos de entrada de dados
2. Informa os valores	
3. Usuário escolhe opção	
	4. Verificar opção selecionada
Exceções	
1. Disponibilizar campos de entrada de dados:	
1.1. Caso o método de preço de venda contenha o atributo tipo, requisitar o caso de uso “Inserir Dado do Tipo Atributo”.	
1.2. Caso o método de preço de venda contenha a variável tipo, requisitar o caso de uso “Inserir Dado do Tipo Variavel”.	
4. Verificar opção selecionada:	
4.1. Caso seja “Salvar”, requisita o caso de uso “Verificar Validade dos Dados”.	
4.2. Caso seja “Cancelar”, tem a responsabilidade de limpar os campos.	

Quadro 14 - Descrição textual do caso de uso “Informar Dados”
Fonte: Autoria Própria.

Nome do Caso de Uso	Inserir Dado do Tipo Atributo
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
1. Usuário insere dado do tipo atributo	
Exceções	

Quadro 15 - Descrição textual do caso de uso "Inserir Dado do Tipo Atributo"
Fonte: Autoria Própria.

Nome do Caso de Uso	Inserir Dado do Tipo Variavel
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
1. Usuário insere dado do tipo variável	
Exceções	

Quadro 16 - Descrição textual do caso de uso "Inserir Dado do Tipo Variavel"
Fonte: Autoria Própria.

Nome do Caso de Uso	Verificar Validade dos Dados
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
	1. Receber os dados
	2. Validar os dados
Exceções	
2. Validar os Dados:	
2.1. Caso os dados sejam válidos, requisita o caso de uso "Armazenar Valores".	
2.2. Caso haja algum dado informado incorretamente, tem a responsabilidade de limpar o campo.	
a) Uma mensagem é retornada sobre o erro que ocorrer.	

Quadro 17 - Descrição textual do caso de uso "Verificar Validade dos Dados"
Fonte: Autoria Própria.

Nome do Caso de Uso	Armazenar Valores
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
	1. Receber os valores
	2. Requisitar “Conectar ao Banco de Dados” do “Subsistema de Conexão”
	3. Gravar os valores no Banco de Dados
	4. Requisitar “Desconectar do Banco de Dados” do “Subsistema de Conexão”
Exceções	
3. Gravar os valores no Banco de Dados:	
a) Uma mensagem é retornada sobre o sucesso ou insucesso do cadastro do atributo.	

Quadro 18 - Descrição textual do caso de uso “Armazenar Valores”
Fonte: Autoria Própria.

Nome do Caso de Uso	**Buscar Atributo**
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
1. Usuário acessa a Tela de Consulta de Atributos	
	2. Exibir a Tela de Consulta de Atributos
	3. Requisitar “Conectar ao Banco de Dados” do “Subsistema de Conexão”
	4. Buscar atributos
	5. Requisitar “Desconectar do Banco de Dados” do “Subsistema de Conexão”
	6. Exibir todos os atributos
Exceções	
6. Exibir todos os atributos:	
6.1. Caso o usuário selecione um atributo, requisita o caso de uso “Selecionar Atributo”.	
6.2. Caso o usuário escolha a ação “Filtrar”, requisita o caso de uso “Filtrar Pesquisa”	

Quadro 19 - Descrição textual do caso de uso “Buscar Atributo”
Fonte: Autoria Própria.

Nome do Caso de Uso	Selecionar Atributo
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
1. Usuário seleciona um atributo	
2. Usuário escolhe opção	
	3. Verificar opção selecionada
Exceções	
3. Verificar opção selecionada:	
1.1. Caso seja “Editar”, requisita o caso de uso “Editar Atributo”.	
1.2. Caso seja “Desabilitar”, requisita o caso de uso “Desabilitar Atributo”.	

Quadro 20 - Descrição textual do caso de uso “Selecionar Atributo”
Fonte: Autoria Própria.

Nome do Caso de Uso	Editar Atributo
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
	1. Exibe Tela de Cadastro
	2. Exibe um atributo
3. Usuário edita atributo	
4. Usuário escolhe opção	
	5. Verificar opção selecionada
Exceções	
5. Verificar opção selecionada:	
5.1. Caso seja “Salvar”, tem a responsabilidade de realizar “Alterar Atributo”.	
a) Caso algum campo esteja vazio é retornada uma mensagem de erro, caso contrário uma mensagem é retornada sobre o sucesso ou insucesso do cadastro do atributo.	
5.2. Caso seja “Cancelar”, retorna-se a Tela de Consulta.	

Quadro 21 - Descrição textual do caso de uso “Editar Atributo”
Fonte: Autoria Própria.

Nome do Caso de Uso	Desabilitar Atributo
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
	1. Exibe mensagem de confirmação
2. Usuário escolhe opção	
	3. Verificar opção selecionada
Exceções	
3. Verificar opção selecionada:	
3.1. Caso o usuário confirme, o atributo é desabilitado.	
a) Uma mensagem é retornada sobre o sucesso ou insucesso da operação.	
3.2. Caso não confirme é cancelado a operação e retorna-se a Tela de Consulta.	

Quadro 22 - Descrição textual do caso de uso "Desabilitar Atributo"
Fonte: Autoria Própria.

Nome do Caso de Uso	Buscar Codigo
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
	1. Busca todos os atributos cadastrados no "Subsistema de Conexão" em que o código possui semelhança com código informado pelo usuário.
Exceções	
1. Busca todos os atributos cadastrados no "Subsistema de Conexão" em que o código possui semelhança com código informado pelo usuário:	
1.1. Caso nenhum atributo seja encontrado, uma mensagem será retornada ao caso de uso que o chamou.	

Quadro 23 - Descrição textual do caso de uso "Buscar Codigo"
Fonte: Autoria Própria.

Nome do Caso de Uso	Filtrar Pesquisa
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
1. Usuário informa a Variável ou a Descrição	
2. Usuário escolhe a ação “Filtrar”	
	3. Requisitar “Conectar ao Banco de Dados” do “Subsistema de Conexão”
	4. Verificar tipo de Pesquisa.
	5. Requisitar “Desconectar do Banco de Dados” do “Subsistema de Conexão”
	6. Mostrar dados
7. Usuário visualiza os atributos	
Exceções	
4. Verificar tipo de Pesquisa:	
4.1. Se a busca for realizada pelo código, requisita o caso de uso “Buscar Codigo”.	
4.2. Se a busca for realizada pela variável, requisita o caso de uso “Buscar Variavel”.	
4.3. Se a busca for realizada pela descrição, requisita o caso de uso “Buscar Descricao”.	

Quadro 24 - Descrição textual do caso de uso “Filtrar Pesquisa”
Fonte: Autoria Própria.

Nome do Caso de Uso	Buscar Variavel
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
	1. Busca todos os atributos cadastrados no “Subsistema de Conexão” em que a variável possui semelhança com a variável informada pelo usuário.
Exceções	
1. Busca todos os atributos cadastrados no “Subsistema de Conexão” em que a variável possui semelhança com a variável informada pelo usuário:	
1.1. Caso nenhum atributo seja encontrado, uma mensagem será retornada ao caso de uso que o chamou.	

Quadro 25 - Descrição textual do caso de uso “Buscar Variavel”
Fonte: Autoria Própria.

Nome do Caso de Uso	Buscar Descricao
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
	1. Busca todos os atributos cadastrados no “Subsistema de Conexão” em que a descrição possui semelhança com a semelhança informada pelo usuário.
Exceções	
1. Busca todos os atributos cadastrados no “Subsistema de Conexão” em que a descrição possui semelhança com a semelhança informada pelo usuário:	
1.1. Caso nenhum atributo seja encontrado, uma mensagem será retornada ao caso de uso que o chamou.	

Quadro 26 - Descrição textual do caso de uso “Buscar Descricao”
Fonte: Autoria Própria.

APÊNDICE F - Descrição Textual para os Casos de Uso do Subframework FoodSystem

A seguir encontram-se as descrições textuais do Diagrama de Caso de Uso UML-F para o Subsistema *FoodSystem* (Figura 9). O Quadro 27 apresenta a descrição textual para o caso de uso “Buscar Atributos”, o Quadro 28 para “Buscar Dados do Método ABC”, o Quadro 29 para “Buscar Dados do Método Sebrae”, o Quadro 30 para “Buscar Dados do Método Custo Pleno”, o Quadro 31 para “Editar Valores”, o Quadro 32 para “Verificar Validade dos Dados”, o Quadro 33 para “Salvar Alterações” e, por fim, o Quadro 34 para “Efetuar Calculo”.

Nome do Caso de Uso	Buscar Atributos
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
	1. Requisitar “Conectar ao Banco de Dados” do “Subsistema de Conexão”
	2. Verificar tipo de busca
	3. Requisitar “Desconectar do Banco de Dados” do “Subsistema de Conexão”
	4. Exibir dados
5. Usuário visualiza os dados	
Exceções	
2. Verificar tipo de busca:	
2.1. Caso o método a ser pesquisado seja o ABC, requisitar o caso de uso “Buscar Dados do Método ABC”.	
2.2. Caso o método a ser pesquisado seja o Sebrae, requisitar o caso de uso “Buscar Dados do Método Sebrae”.	
2.3. Caso o método a ser pesquisado seja o Custo Pleno, requisitar o caso de uso “Buscar Dados do Método Custo Pleno”.	

Quadro 27 - Descrição textual do caso de uso “Buscar Atributos”
Fonte: Autoria Própria.

Nome do Caso de Uso	Buscar Dados do Metodo ABC
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
	1. Buscar Atributos, Linha de Produção e Produtos relacionados ao Método ABC
Exceções	

Quadro 28 - Descrição textual do caso de uso "Buscar Dados do Metodo ABC"
Fonte: Autoria Própria.

Nome do Caso de Uso	Buscar Dados do Metodo Sebrae
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
	1. Buscar Atributos relacionados ao Método Sebrae
Exceções	

Quadro 29 - Descrição textual do caso de uso "Buscar Dados do Metodo Sebrae"
Fonte: Autoria Própria.

Nome do Caso de Uso	Buscar Dados do Metodo Custo Pleno
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
	1. Buscar Atributos e seus respectivos Itens relacionados ao Método Custo Pleno
Exceções	

Quadro 30 - Descrição textual do caso de uso "Buscar Dados do Metodo Custo Pleno"
Fonte: Autoria Própria.

Nome do Caso de Uso	Editar Valores
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
1. Usuário edita os valores	
2. Usuário escolhe opção	
	3. Verificar opção selecionada
Exceções	
3. Verificar opção selecionada:	
3.1. Caso seja “Salvar”, requisitar o caso de uso “Verificar Validade dos Dados”.	
a) Caso algum valor esteja preenchido incorretamente é retornada uma mensagem de erro, caso contrário uma mensagem é retornada sobre o sucesso ou insucesso da gravação no banco de dados.	
3.2. Caso seja “Calcular”, requisitar o caso de uso “Efetuar Calculo”.	
a) Caso algum valor esteja preenchido incorretamente é retornada uma mensagem de erro, caso contrário é requisitado o caso de uso “Efetuar Calculo”.	
3.3. Caso seja “Fechar”, retorna a Tela Principal.	

Quadro 31 - Descrição textual do caso de uso “Editar Valores”
Fonte: Autoria Própria.

Nome do Caso de Uso	Verificar Validade dos Dados
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
	1. Receber o valor
	2. Validar o valor
Exceções	
2. Validar o valor:	
2.1. Caso o valor seja válido, requisita o caso de uso ”Salvar Alteracoes”.	
2.2. Caso o valor não seja válido, retorna uma mensagem ao caso de uso que o requisitou.	

Quadro 32 - Descrição textual do caso de uso “Verificar Validade dos Dados”
Fonte: Autoria Própria.

Nome do Caso de Uso	Salvar Alteracoes
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
	1. Receber os valores
	2. Requisitar “Conectar ao Banco de Dados” do “Subsistema de Conexão”
	3. Gravar os valores no Banco de Dados
	4. Requisitar “Desconectar do Banco de Dados” do “Subsistema de Conexão”
Exceções	
3. Gravar os valores no Banco de Dados:	
a) Uma mensagem é retornada sobre o sucesso ou insucesso da gravação dos valores	

Quadro 33 - Descrição textual do caso de uso “Salvar Alteracoes”
Fonte: Autoria Própria

Nome do Caso de Uso	Efetuar Calculo
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
	1. Requisita o caso de uso “Verificar Validade dos Dados”
Exceções	
1. Requisita o caso de uso “Verificar Validade dos Dados”	
1.1. Caso os valores estejam todos corretos, requisita o subsistema “Calculation”	
a) Uma mensagem é mostrada caso algum valor esteja incorreto.	

Quadro 34 - Descrição textual do caso de uso “Efetuar Calculo”
Fonte: Autoria Própria.

APÊNDICE G - Descrição Textual para os Casos de Uso do Subframework Calculation

A seguir encontram-se as descrições textuais do Diagrama de Caso de Uso UML-F para o Subsistema *Calculation* (Figura 10). O Quadro 35 apresenta a descrição textual para o caso de uso “Informar Numero Produzidos”, o Quadro 36 para “Escolher Linha de Producao”, o Quadro 37 para “Buscar Produtos”, o Quadro 38 para “Calcular”, o Quadro 39 para “Inserir Margem de Lucro”, o Quadro 40 para “Salvar”, o Quadro 41 para “Tela de Calculo ABC”, o Quadro 42 para “Tela de Calculo Sebrae”, o Quadro 43 para “Tela de Calculo Custo Pleno” e, por fim, o Quadro 44 para “Buscar Linha de Producao”.

Nome do Caso de Uso	Informar Numero Produzidos
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
1. Usuário informa a quantidade produzida de um determinado produto	
2. Usuário escolhe opção	
	3. Verificar opção selecionada
Exceções	
3. Verificar opção selecionada	
3.1. Caso seja “Calcular”, requisita o caso de uso “Calcular”.	

Quadro 35 - Descrição textual do caso de uso “Informar Numero Produzidos”
Fonte: Autoria Própria.

Nome do Caso de Uso	Escolher Linha de Producao
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
1. Usuário seleciona a linha de produção	
	2. Requisitar o caso de uso “Buscar Produtos”
Exceções	

Quadro 36 - Descrição textual do caso de uso “Escolher Linha de Producao”
Fonte: Autoria Própria.

Nome do Caso de Uso	Buscar Produtos
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
	1. Requisitar “Conectar ao Banco de Dados” do “Subsistema de Conexão”
	2. Buscar Produtos
	3. Requisitar “Desconectar do Banco de Dados” do “Subsistema de Conexão”
	4. Exibir todos os produtos
Exceções	

Quadro 37 - Descrição textual do caso de uso “Buscar Produtos”
Fonte: Autoria Própria.

Nome do Caso de Uso	Calcular
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
	1. Efetuar o cálculo do preço de venda
	2. Exibir o resultado
3. Usuário verifica o resultado	
4. Usuário seleciona opção	
	5. Verificar opção selecionada
Exceções	
5. Verificar opção selecionada	
5.1. Caso seja “Gravar”, requisita o caso de uso “Salvar”.	

Quadro 38 - Descrição textual do caso de uso “Calcular”
Fonte: Autoria Própria.

Nome do Caso de Uso	Inserir Margem de Lucro
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
1. Usuário informa a porcentagem de lucro que deseja obter	
2. Usuário seleciona opção	
	3. Verificar opção selecionada
Exceções	
3. Verificar opção selecionada	
3.1. Caso seja “Calcular”, requisita o caso de uso “Calcular”.	

Quadro 39 - Descrição textual do caso de uso “Inserir Margem de Lucro”
Fonte: Autoria Própria.

Nome do Caso de Uso	Salvar
Estabilidade ☒	**Flexibilidade** ☐
Ações do Usuário	**Responsabilidades do Sistema**
	1. Receber os valores
	2. Requisitar “Conectar ao Banco de Dados” do “Subsistema de Conexão”
	3. Gravar os valores no Banco de Dados
	4. Requisitar “Desconectar do Banco de Dados” do “Subsistema de Conexão”
Exceções	
3. Gravar os valores no Banco de Dados	
a) Uma mensagem é retornada sobre o sucesso ou insucesso do cadastro do atributo.	

Quadro 40 - Descrição textual do caso de uso “Salvar”
Fonte: Autoria Própria.

Nome do Caso de Uso	Tela de Calculo ABC
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
	1. Requisita o caso de uso “Buscar Linha de Producao”
Exceções	

Quadro 41 - Descrição textual do caso de uso “Tela de Calculo ABC”
Fonte: Autoria Própria.

Nome do Caso de Uso	Tela de Calculo Sebrae
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
	1. Requisita o caso de uso “Calcular”
Exceções	

Quadro 42 - Descrição textual do caso de uso “Tela de Calculo Sebrae”
Fonte: Autoria Própria.

Nome do Caso de Uso	Tela de Calculo Custo Pleno
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
	1. Requisitar o caso de uso “Inserir Margem de Lucro”
Exceções	

Quadro 43 - Descrição textual do caso de uso “Tela de Calculo Custo Pleno”
Fonte: Autoria Própria.

Nome do Caso de Uso	Buscar Linha de Producao
Estabilidade ☐	**Flexibilidade** ☒
Ações do Usuário	**Responsabilidades do Sistema**
	1. Requisitar “Conectar ao Banco de Dados” do “Subsistema de Conexão”
	2. Buscar linha de producao
	3. Requisitar “Desconectar do Banco de Dados” do “Subsistema de Conexão”
	4. Exibir todas as linhas de produção
Exceções	

Quadro 44 - Descrição textual do caso de uso “Buscar Linha de Producao”
Fonte: Autoria Própria.

ANEXO A - Diagramas de Classes UML para os Métodos de Preço de Venda

As Figuras 90, 91 e 92, ilustram os Diagramas de Classes dos subsistemas Gerenciar Atributo (ABC), Cadastrar Atributo (Sebrae) e Cadastro de Atributos (Custo Pleno) desenvolvidas por Crazuski *et al.* (2008), respectivamente.

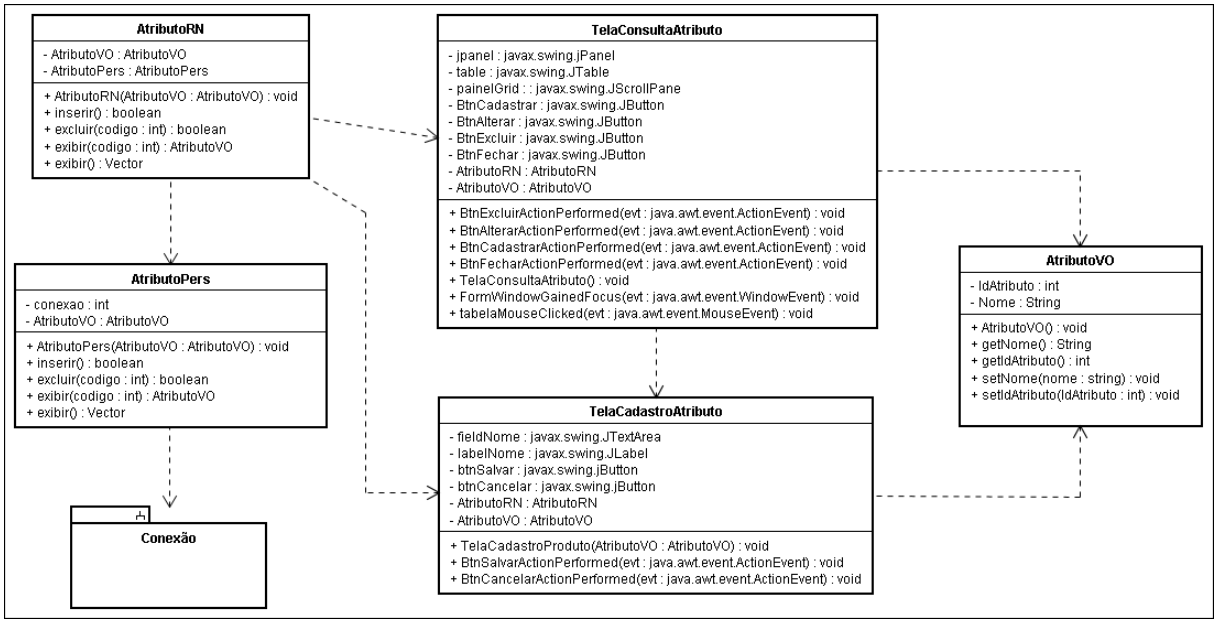


Figura 90 - Esboço do Diagrama de Classes para o subsistema ”Gerenciar Atributo”
Fonte: Crazuski et. al. (2008).

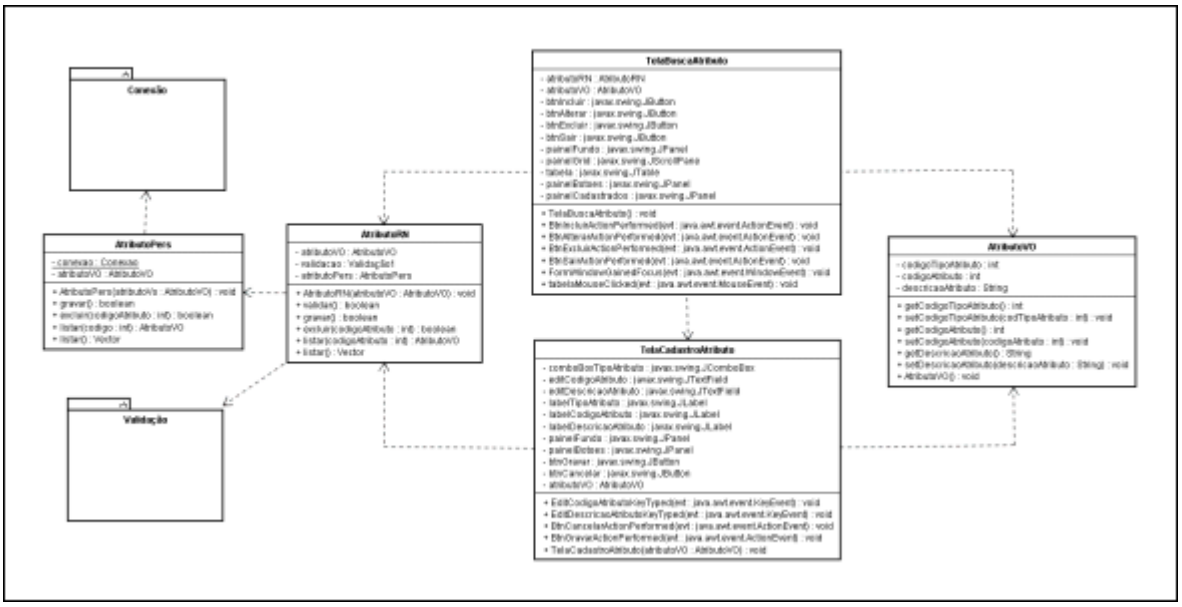


Figura 91 - Diagrama de Classes UML para o subsistema “Cadastrar Atributo”
Fonte: Crazuski *et al.* (2008).

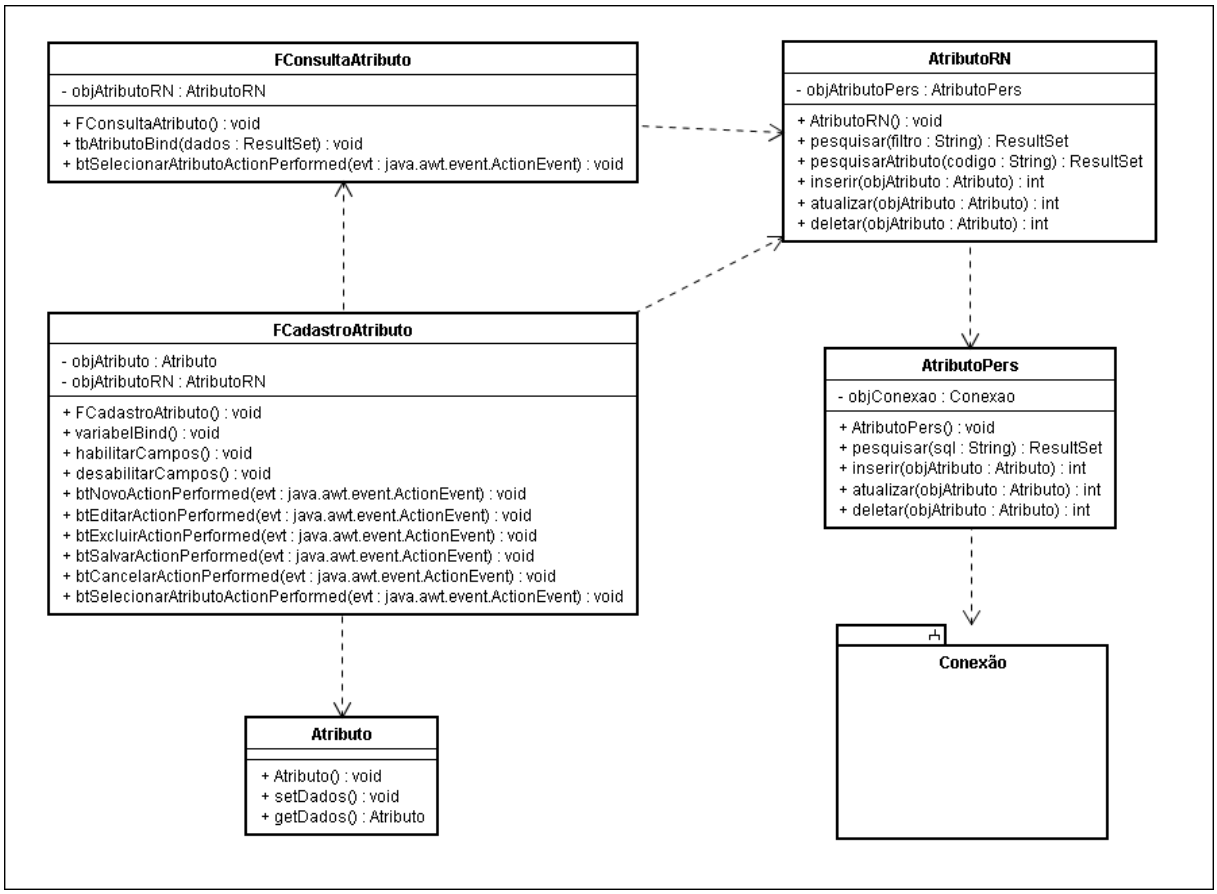


Figura 92 - Diagrama de Classes UML para o subsistema “Cadastro de Atributos”
Fonte: Crazuski *et al.* (2008).

ANEXO B - Diagramas de Classe UML do Subsistema Alimentar Sistema do Método Sebrae

A Figura 93, ilustra o Diagrama de Classe do subsistema Alimentar Sistema do método Sebrae desenvolvida por Crazuski *et al.* (2008).

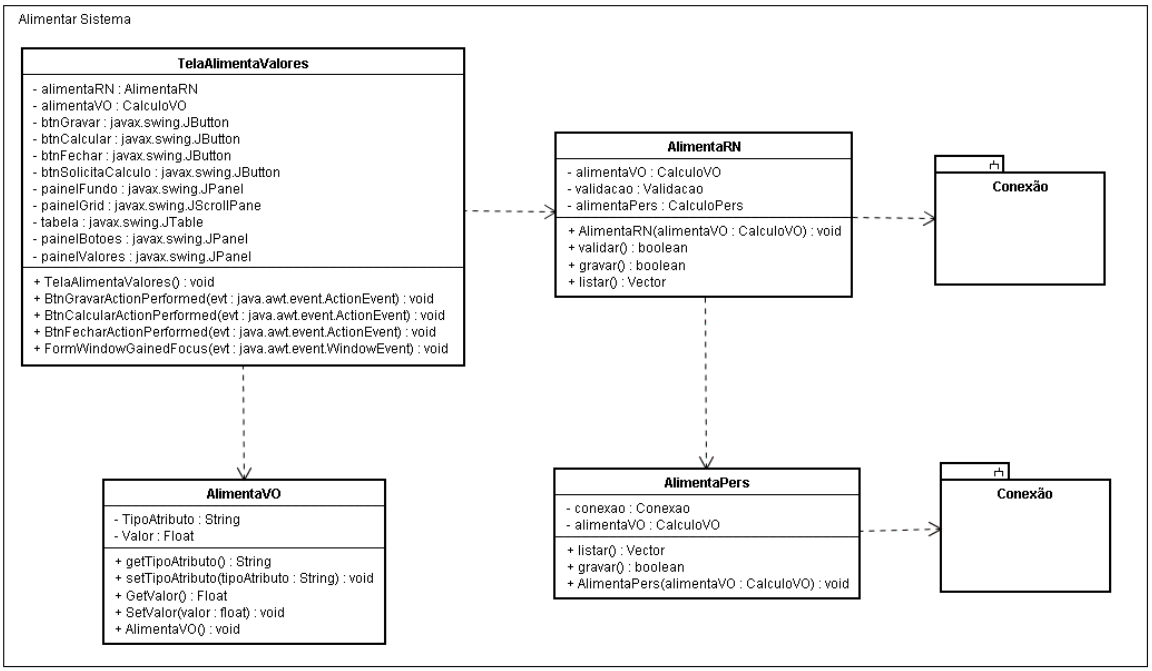


Figura 93 – Diagrama de Classes UML para o subsistema “Alimentar Sistema”
Fonte: Crazuski *et al.* (2008).

ANEXO C - Diagramas de Classe UML dos Subsistemas responsáveis pelo Cálculo do Preço de Venda

As Figuras 94, 95 e 96, ilustram os Diagramas de Classes dos subsistemas Cálculo do Preço de Venda (ABC), Calcular Preço de Venda (Sebrae) e Cálculo do Preço de Venda (Custo Pleno) desenvolvidas por Crazuski *et al.* (2008), respectivamente.

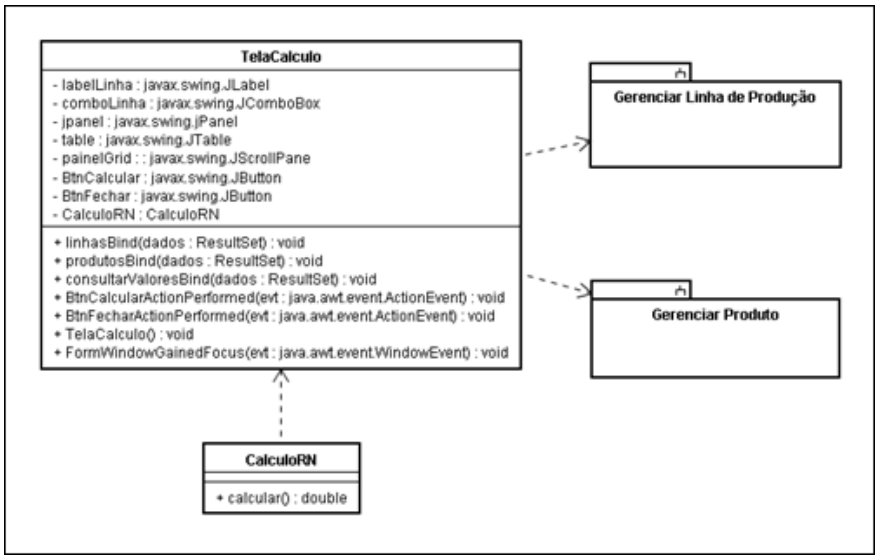


Figura 94 – Diagrama de Classes do subsistema "Cálculo do Preço de Venda" do método ABC
Fonte: Crazuski *et al.* (2008).

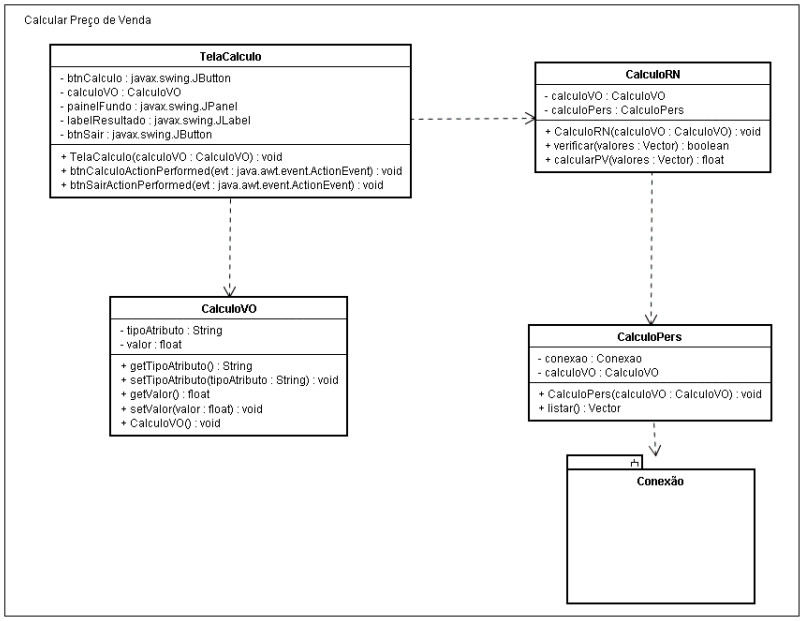


Figura 95 – Diagrama de Classes UML para o subsistema "Calcular Preço de Venda" do método Sebrae
Fonte: Crazuski *et al.* (2008).

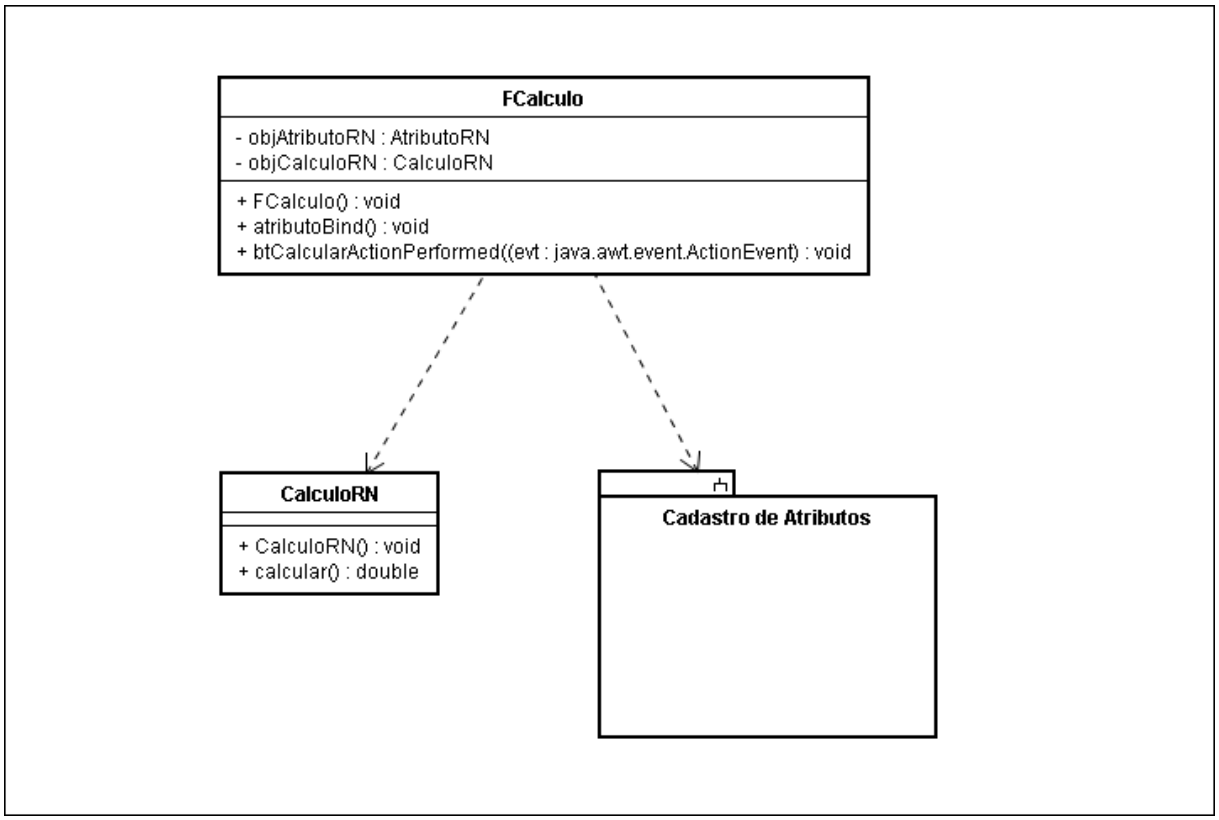


Figura 96 – Diagrama de Classes UML para o subsistema "Cálculo do Preço de Venda" do método Custo Pleno
Fonte: Crazuski *et al.* (2008).