UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MAICON AZEVEDO DA LUZ

UMA LINHA DE PRODUTO DE SOFTWARE PARA APLICAÇÕES WEB

São Leopoldo
2013

Maicon Azevedo da Luz

UMA LINHA DE PRODUTO DE SOFTWARE PARA APLICAÇÕES WEB

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pelo Curso de Análise e Desenvolvimento de Sistemas da Universidade do Vale do Rio dos Sinos – UNISINOS

Orientador: Kleinner Silva Farias de Oliveira

São Leopoldo

2013

*Agradeço a **Oxalá** e todas as entidades que me iluminaram nessa caminhada. Agradeço a minha **família** por me apoiar e me incentivar em todos os momentos.*

AGRADECIMENTOS

Agradeço ao professor Kleinner Silva Farias de Oliveira por ter me orientado nesse trabalho, por sua dedicação, por sua parceria e por ter compartilhado o seu conhecimento comigo.

RESUMO

Nos últimos anos, as empresas de desenvolvimento de aplicações Web têm enfrentando uma crescente demanda por produtos de alta qualidade, com baixo custo e com tempo de produção cada vez menor. Diante deste contexto, as empresas buscam metodologias para potencializar o reuso sistemático. Devido à ineficiência das metodologias tradicionais, Linha de Produto de Software surge como uma abordagem promissora para o desenvolvimento de aplicações Web, permitindo o alinhamento estratégico do negócio das empresas com o paradigma de desenvolvimento de software. A Linha de Produto permite ter um ganho de produtividade por potencializar o reuso dos artefatos de software de forma sistemática. Esse trabalho, portanto, apresenta a análise dos requisitos e o projeto para o desenvolvimento de uma Linha de Produto de Software para aplicações Web.

Palavras-Chave: Java. Framework. Linha de Produto de Software.

ABSTRACT

In the last years, companies in developing Web applications are facing a growing demand for high quality products with low cost and production time getting smaller. In this context, companies seek methodologies to enhance the systematic reuse. Due to the inefficiency of traditional methodologies, Software Product Line emerges as a promising approach for the development of Web applications, allowing the strategic alignment of business enterprises with the paradigm of software development. The Product Line improves the productivity by enhancing the reuse of software artifacts in a systematic way. This work, therefore, presents a requirements analysis and design for the development of a Software Product Line for Web applications.

Keywords: Java. Framework. Software Product Line.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API	Application Programming Interface
CDI	Contexts and Dependency Injection for Java EE
CSS	Cascading Style Sheets
DAO	Data Access Object
DML	Data Manipulation Language
DTO	Data Transfer Object
EJB	Enterprise JavaBeans
FDL	Feature Domain Language
FODA	Feature-Oriented Domain Analysis
GWT	Google Web Toolkitc
HTML	HyperText Markup Language
JAX-RS	Java API for RESTful Services
JCA	Java EE Connector Architecture
JMS	Java Message Service
JPA	Java Persistence API
JSF	JavaServer Face
JSP	JavaServer Pages
JTA	Java Transaction API
LPS	Linha de Produto de Software
MVC	Model–View–Controller
PLA	Product Line Architecture
SEI	Software Engineering Institute
SMS	Short Message Service
SWT	Standard Widget Toolkit
UML	Unified Modeling Language
UNISNIOS	Universidade do Vale do Rio dos Sinos

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, empresas de desenvolvimento de aplicações Web tem enfrentado o desafio de gerar produtos com alta qualidade, com baixo custo e com tempo de produção cada vez menor. Tais empresas têm como objetivos primários a busca constante por: maior qualidade dos produtos de software gerados, domínio do mercado de atuação, agilidade na sua produção, alinhamento dos produtos da empresa para promover redução de custo e aumento de produtividade, reduzir o custo de desenvolvimento e manutenção de produtos, e tecnologias que facilitem a customização dos produtos. Atingir esses objetivos é algo decisivo para tais empresas, visto que terá um impacto direto no lucro e, consequentemente, na sobrevivência das mesmas.

Diante deste contexto de crescente necessidade de aumento de produtividade a baixo custo, as empresas que desenvolvem e comercializam software buscam metodologias que potencializem a eficiência e produtividade ao longo do processo de desenvolvimento. Mais especificamente, essas empresas buscam adotar metodologias que potencializem o *reuso sistemático* de módulos de aplicações Web, que permitam construir módulos *intercambiáveis* entre aplicações, e favoreçam o gerenciamento do ciclo de vida dos seus produtos a serem disponibilizados na Web (KANG et al., 2010). Consequentemente, espera-se que o desafio da construção de produtos de *alta qualidade* em um *menor espaço* de tempo e com *menor custo* seja superado ou, ao menos, minimizado. Exemplos de empresas que buscam tais resultados seriam Nokia (NORTHROP, 2008), Microsoft (NORTHROP, 2008), dentre outras.

Atualmente, para enfrentar esta problemática, as empresas têm utilizado metodologias tradicionais de desenvolvimento, incluindo desenvolvimento baseado em componentes, em serviços e baseado no paradigma de orientação a objetos. Embora essas metodologias tenham sido amplamente utilizadas, quando comparadas com Linha de Produto de Software (LPS), elas têm se mostrado não tão competitivas, ou mesmo produtivas, para o desenvolvimento de software de alta qualidade a baixo custo e em um curto espaço de tempo, principalmente quando aplicadas sem uma *rastreabilidade* entre os artefatos de software criados. De fato, tem-se observado que tais metodologias estão estritamente relacionadas às questões de implementação como, por exemplo, definição de sub-rotinas, módulos, objetos, componentes e serviços, ao contrário de promoverem um alinhamento consistente entre os objetivos de negócio das empresas e os produtos de software oferecidos pelas mesmas.

Dada à popularidade da internet, esse desalinhamento tem sido crítico para empresas que desenvolvem e comercializam aplicações Web, incluindo sistemas Web de gestão de empresas e redes sociais. Neste sentido, o desafio atacado neste trabalho é utilizar metodologias que promovam o reuso sistemático, ao contrário de *ad hoc,* de aplicações Web. Proporcionando, desse modo, um aumento de produtividade através do reuso sistemático.

Com isso em mente, a academia e a indústria têm investido esforços para entender as características de tais aplicações Web a fim de sistematizar o processo de desenvolvimento. Estudos recentes (POHL et al., 2005) mostram evidências que as empresas produzem e mantem famílias de produtos que compartilham características em comum e que possuem algumas particularidades que os diferenciam. Diante da ineficiência das *metodologias tradicionais*, a metodologia de LPS surge como uma abordagem considerada promissora para o desenvolvimento Web, visto que permite o alinhamento entre as estratégias de negócio das empresas e o paradigma de desenvolvimento de software, permitindo um ganho de produtividade por potencializar a rastreabilidade e, consequentemente, o reuso dos artefatos de software.

Em (NORTHROP, 2008), Northrop mostra que o uso de LPS foi decisivo para o desenvolvimento de produtos de software em grandes empresas (tais como Nokia e Microsoft), nas áreas de sistemas de comando e controle de naves espaciais terrestres, sistemas de controle de navio de guerra (conhecido como *Celsiustech*), e sistemas Web para o mercado de ações. Por exemplo, a Nokia tem usado LPS com sucesso para gerenciar o núcleo comum entre seus produtos, bem como as suas particularidades. O uso de LPS tem permitido lidar de forma sistemática com: (1) diferentes tipos de teclas e tamanhos de telas; (2) recursos de localização aplicados em diferentes regiões no mundo; (3) a internacionalização dos celulares através do suporte a múltiplos idiomas; (4) a interoperabilidade dos celulares através do suporte de múltiplos protocolos de redes, (5) a manutenabilidade da compatibilidade entre as diferentes versões dos produtos da empresa; (6) o suporte a recursos configuráveis dos dispositivos; e (7) a necessidade de gerenciamento das funcionalidades dos produtos durante a fase de desenvolvimento e durante a fase de manutenção e evolução dos mesmos. Em (KANG et al., 2010), os autores demonstram que LPS viabiliza o desenvolvimento de produtos que compartilham de uma mesma arquitetura e, consequentemente, dos componentes que configuram esta arquitetura; sendo a diferença entre os mesmos representada pelas suas variabilidades.

Embora cada vez mais as aplicações sejam desenvolvidas e disponibilizadas na web, como *e-commerce* e aplicações governamentais, o seu desenvolvimento ainda é custoso e

frequentemente extrapola os prazos estabelecidos. De fato, o estado da prática em desenvolvimento de aplicações Web mostra que boa parte dos projetos extrapolam os cronogramas estabelecidos. Dito isso, fica caracterizado a necessidade de estudos que tentem resolver, ou mesmo mitigar, o desafio de desenvolver aplicações Web de alta qualidade com prazos curtos e com orçamento reduzidos. Portanto, a grande contribuição deste trabalho é usar LPS para desenvolver famílias de aplicações Web fundamentada na rastreabilidade e reuso sistemático de seus módulos, conseguindo produzir novos produtos mais rapidamente através do gerenciamento efetivo de suas partes variantes.

Sendo assim, a Seção 1.1 apresenta a formulação do problema atacado neste trabalho. A Seção 1.2 especifica os objetivos gerais e específicos que se deseja atingir. A Seção 1.3.3 descreve brevemente quais métodos de estudo serão utilizados para atingir os objetivos traçados, bem como define a maneira como tais estudos serão executados. Visando posicionar o trabalho na comunidade científica, a Seção 1.4 caracteriza as contribuições científicas do trabalho e descreve o impacto da pesquisa no estado da prática de desenvolvimento de software de aplicações Web. Por fim, a Seção 1.5 descreve como cada Capítulo do trabalho está organizado.

1.1 Formulação do problema

Para superar o desafio de desenvolver aplicações Web de alta qualidade em um menor espaço de tempo e com menor custo, é necessário resolver três problemas centrais descritos a seguir.

Representação inadequada das features de aplicações Web.

Dado que as aplicações Web usualmente compartilham de um núcleo comum, sendo essas diferenciadas por algumas particularidades, os desenvolvedores precisam compreender de forma adequada quais são as *features* das aplicações e como a variabilidade entre as aplicações Web se manifestam. Porém, na prática, os desenvolvedores não compreendem as *features,* bem como a variabilidade entre os produtos das aplicações. Isso pode ser explicado por alguns motivos: (1) a inadequada representação das *features* das aplicações e das dependências entre elas; (2) a não especificação das diferenças entre os produtos, bem como das suas variabilidades; e (3) a indefinição de quais *features* são necessárias para configurar

um determinado produto. Portanto, tem-se, de fato, uma inadequada representação das *features* que compõem os produtos, bem como das suas variabilidades.

Imprecisão na rastreabilidade entre features, artefatos e módulos de aplicações Web.

O ganho de produtividade na manutenção e evolução de aplicações Web é alcançado quando os desenvolvedores entendem como a implementação das *features* é distribuída através dos artefatos e módulos das aplicações. Porém, esse mapeamento raramente é feito e mantido. Consequentemente, é particularmente desafiante para os desenvolvedores promover um reuso sistemático das *features*, e dos módulos que as implementam; dificultando as atividades de evolução e manutenção das aplicações.

A falta de entendimento pode ser explicada por algumas razões: (1) as *features* que formam a família de produtos são raramente elicitadas e mapeadas de forma precisa para os artefatos de software. Isso implica em dizer que os desenvolvedores comumente não têm acesso às especificações de casos de uso e diagramas estruturais e comportamentais da Unified Modeling Language (UML); e (2) os desenvolvedores são incapazes de navegar dos artefatos que especificam os requisitos, para os artefatos que definem o projeto e a implementação das features, e vice-versa. Portanto, tem-se uma imprecisão na rastreabilidade entre as features, seus artefatos e módulos que as implementam.

Desconhecimento sobre os reais ganhos do uso de LPS para aplicações Web.

Como previamente mencionado, é aconselhável utilizar LPS para superar o desafio de desenvolver aplicações Web de alta qualidade em um menor espaço de tempo e com menor custo. Porém, a literatura atual falha ao não fornecer detalhes e evidências que comprovem se o uso de LPS pode trazer reais benefícios, incluindo maior reusabilidade e melhor modularidade dos produtos gerados. Pode ser, por exemplo, que a natureza do domínio das aplicações Web comprometa os reais ganhos esperados com LPS. Portanto, existe um desconhecimento sobre os reais ganhos do uso de LPS para o desenvolvimento de aplicações Web.

1.2 Objetivos

O objetivo geral deste trabalho é *desenvolver* uma LPS de aplicações Web e *avaliar* a modularidade dos produtos derivados da mesma. Os objetivos específicos são descritos brevemente a seguir:

- Fazer um estudo detalhado da literatura sobre LPS de aplicações web;
- Analisar, projetar e implementar uma LPS de aplicações web;
- Desenvolver uma arquitetura para LPS de aplicações web;
- Estudar métodos de avaliação de LPS;
- Realizar estudos de caso derivando produtos da LPS de aplicações web elaborada;
- Fazer uma análise comparativa entre os produtos derivados da LPS com produtos não derivados da linha.

1.3 Metodologia do estudo

O trabalho será realizado em seis etapas que serão executadas seguindo a sua ordem de apresentação. As etapas 1, 2 e 3 já foram executadas e os resultados são apresentados neste trabalho. As etapas 4, 5 e 6 serão executadas em um trabalho futuro, o qual também consiste de requisito básico para a colação de grau da Universidade do Vale do Rio dos Sinos (UNISINOS). As etapas do estudo são apresentadas a seguir:

- *Etapa 1***:** esta etapa foca na execução de atividades visando fomentar o entendimento dos aspectos teóricos do trabalho, do estado da arte, e na caracterização do problema. Em particular, nessa etapa foram executadas as seguintes atividades: (1) estudo dos aspectos teóricos do trabalho; (2) estudo do estado-da-arte; (3) formulação do problema estudado; e (4) definição dos objetivos de pesquisa.

- *Etapa 2***:** baseado no entendimento adquirido na Etapa 1, a segunda etapa se concentra na análise e no projeto da linha de produto de software. Em particular, nessa etapa foram executadas as seguintes atividades: (5) análise e projeto da linha de produto de software; e (6) estudos dos métodos de avalição que serão utilizados para avaliar a linha de produto projetada.

- *Etapa 3***:** esta etapa se concentrou na elaboração da monografia. Ela se caracteriza como uma etapa que ocorreu em paralelo as etapas 1 e 2. Mais especificamente, as seguintes atividades foram executadas: (7) redação da monografia do TC 1; e (8) revisão e entrega da monografia.

- *Etapa 4***:** o foco desta etapa consiste no desenvolvimento da linha de produto projetada. Desse modo, a atividade a ser realizada nesta etapa é a (9)

implementação da linha de produto, incluindo o desenvolvimento do framework de suporte e as variabilidades da linha de produto.

- *Etapa 5***:** esta etapa se concentra na avaliação da linha de produto desenvolvida. Para isto, duas atividades serão executadas: (10) realização de um estudo de caso exploratório; e (11) avaliação dos produtos derivados da LPS implementada em termos de modularidade e custo-benefício.

- *Etapa 6***:** o foco desta etapa é na elaboração do artigo científico e será transversal às Etapas 4 e 5. À medida que as atividades 9, 10 e 11 forem finalizadas, os conteúdos gerados serão inseridos no artigo. Em particular, uma macro-atividade será executada: (12) escrita, entrega e submissão do artigo científico.

Por fim, é importante destacar que todos os métodos de pesquisa utilizados são baseados no estado-da-arte na área de Engenharia de Software Experimental (WOHLIN et al., 2012).

1.4 Contribuições

As contribuições deste trabalho são geradas a partir da execução das etapas desta pesquisa descrita anteriormente. Sendo assim, é possível elencar as seguintes contribuições: (1) uma análise do estado da arte na área de linhas de produto de aplicações Web; (2) análise e projeto de uma linha de produto de aplicações Web; (3) análise de domínio de LPS de aplicações Web; (4) conhecimento empírico sobre os reais benefícios do uso de linha de produto para resolver a inadequada representação de features de aplicações Web e para reduzir a imprecisão na rastreabilidade entre as features, os artefatos e os módulos das aplicações Web.

1.5 Organização do trabalho

Este trabalho é organizado da seguinte forma. O Capítulo **Erro! Fonte de referência não encontrada.** apresenta a fundamentação teórica do trabalho descrevendo os principais conceitos que contribuem para o bom entendimento dos próximos capítulos. O Capítulo 0 descreve a análise e o projeto da linha de produto para as aplicações Web. O Capítulo **Erro! Fonte de referência não encontrada.** apresenta os trabalhos relacionados, bem como uma análise comparativa entre os mesmos. O Capítulo 0 descreve as considerações finais e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Após apresentar a formulação do problema atacado neste trabalho, descrever os objetivos do estudo e a metodologia de estudo, bem como as contribuições esperadas com este trabalho, este Capítulo tem como objetivo central descrever os principais conceitos para o bom entendimento do projeto de uma linha de produto de software de aplicações Web.

2.1 Linha de Produto de Software (LPS)

A abordagem de linha de produto teve origem na indústria automobilística, permitindo que vários modelos pudessem ser customizados. A primeira proposta de desenvolvimento de componentes em escala surgiu em 1969 e com o passar dos anos foi aprimorado por diversos pesquisadores até que o Instituto de Engenharia de Software[1] (SEI) refinasse o conceito de linha de produto e disponibilizasse um framework descrevendo os aspectos que devem ser observados na construção de uma LPS.

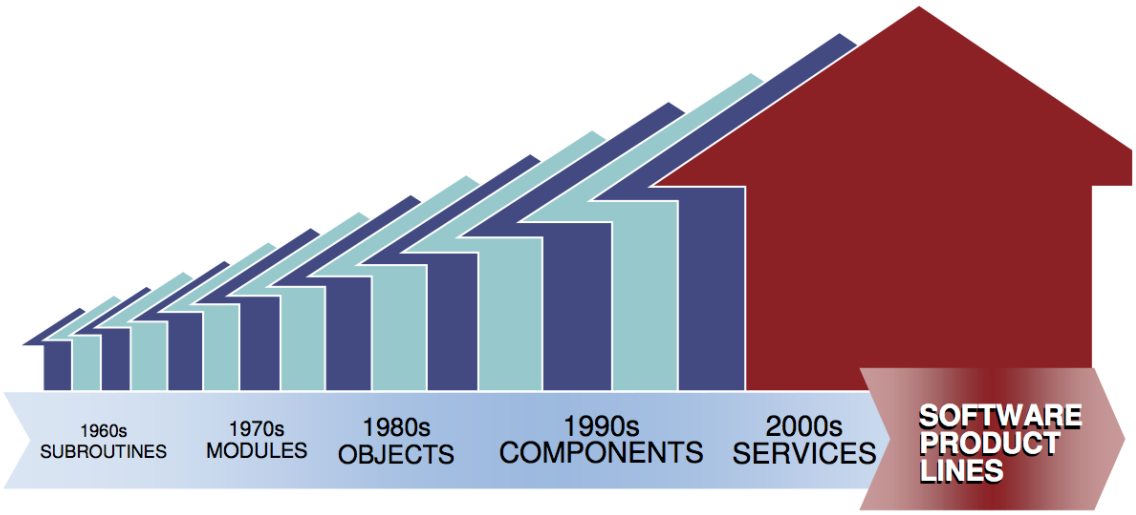
Linha de produto de software representa uma nova metodologia de desenvolvimento de software que busca definir técnicas para aumentar a produtividade das organizações através do reuso sistemático dos ativos de software. Essa nova metodologia visa substituir o reuso *ad-hoc* praticado até então por uma abordagem de reuso sistemático, aquela que potencializa e gerencia o reuso dos ativos das organizações. Em (CLEMENTS; NORTHROP, 2001), o autor define LPS como "sendo um conjunto de sistemas de software que compartilham um conjunto comum e gerenciado de características[2], as quais satisfazem necessidades específicas de um segmento de mercado, e que são desenvolvidos de uma forma pré-definida a partir de um conjunto comum de ativos. Em (CZARNECKI; EISENECKER, 2000), o autor define característica como sendo "uma propriedade de um sistema que é relevante para um cliente e que é usada para capturar os aspectos comuns e diferentes entre produtos de uma linha". Tipicamente, essas características são agrupadas em um modelo, conhecido como modelo de features, e são classificadas como, obrigatórias, opcionais e alternativas. Além disso, o reuso sistemático só é alcançado quando é possível compreender como uma família de sistemas de software compartilham características; e (2) *entender* como

[1] Do inglês *Software Engineering Institute*

[2] Do inglês *feature*

as diferenças entre tais sistemas se manifestam. A Figura 1 mostra a evolução do uso de metodologias para aumentar o grau de reuso.

Figura 1 - Evolução das metodologias de reuso: de ad hoc para abordagens sistemáticas.

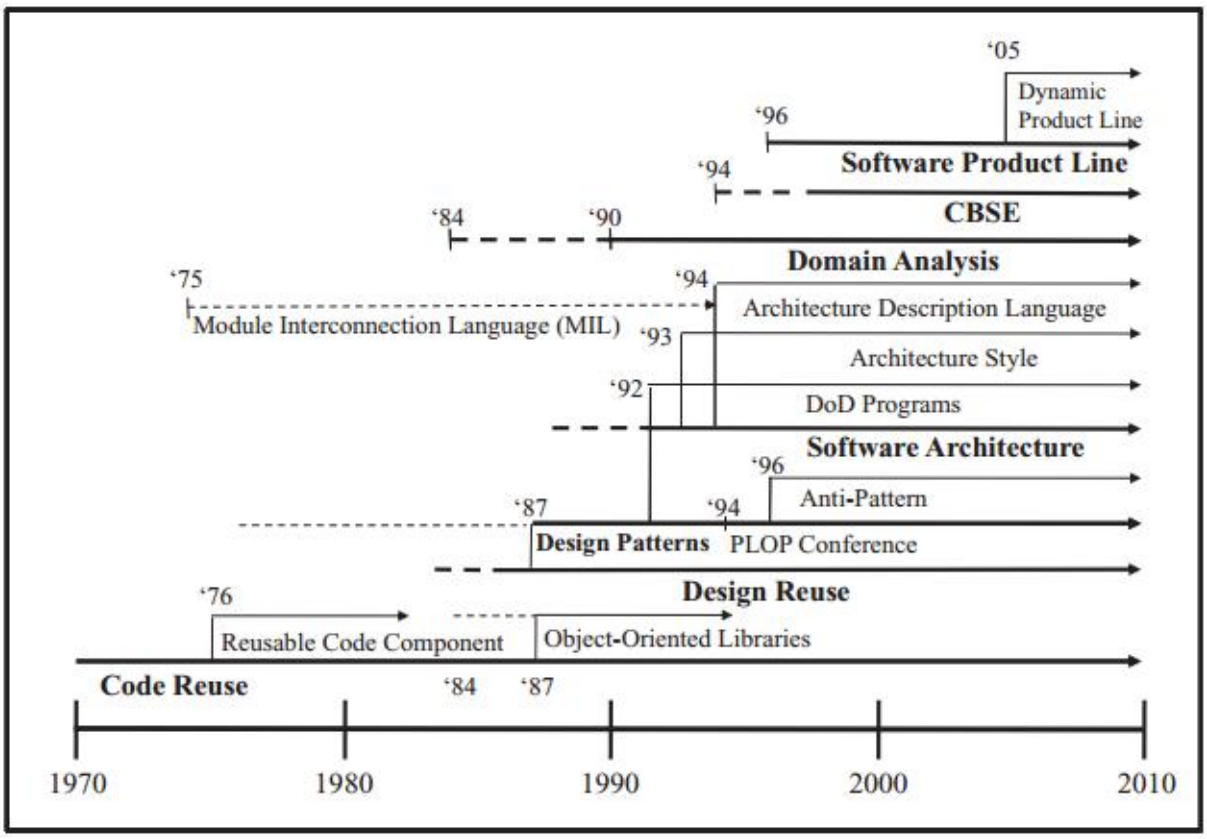


Fonte: (NORTHROP, 2008)

O conceito de LPS oferece uma nova estratégia para permitir o avanço no ganho de produtividade das equipes de desenvolvimento nas organizações. Essa estratégia tende a potencializar o reuso identificando as partes comuns e diferentes entre os produtos de tal forma que cada produto possa ser elaborado através do reuso sistemático das partes comum/diferentes entre os produtos. O uso de LPS é motivado pela ajuda que as empresas passam a ter para superar o desafio de elaborar produtos de alta qualidade a baixo custo. Ao implementar LPS, as empresas podem: (1) aumentar a produtividade e a qualidade dos produtos de software; (2) reduzir o custo de desenvolvimento em curto espaço de tempo; (3) diminuir o retrabalho e o tempo de entrega dos produtos; e (4) potencializar a adaptação das empresas para atingir novos mercados ao conseguir desenvolver novos produtos de uma forma mais rápida e com qualidade.

A primeira LPS conhecida foi desenvolvida no Japão em 1977 e ficou conhecida como *Toshiba Software Factory* (KANG et al., 2010). Abaixo é possível observar na Figura 2 a evolução das abordagens utilizadas para reutilizar software ao longo dos anos.

Figura 2 - Evolução do reuso em LPS



Fonte: (KANG et al., 2010)

2.1.1 Análise de domínio orientado a *feature*

O desenvolvimento de software usando a metodologia de LPS pode ser dividido em duas etapas: (1) *engenharia de domínio*, onde os requisitos da LPS são elicitados e especificados. A definição da estrutura utilizada na linha de produtos ocorre na engenharia de domínio, definindo os pontos que serão comuns aos produtos bem como os pontos de variabilidade; e (2) *engenharia da aplicação*, onde os produtos da linha são analisados, projetados e implementados.

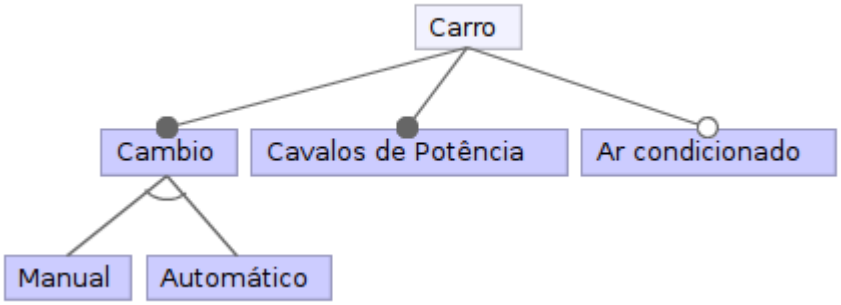
O conceito de análise orientada a *feature*[3] (FODA) surgiu em 1990 em (KANG et al., 1990). A modelagem orientada a *feature* busca identificar e analisar os requisitos ou características que são comuns e a variabilidade em uma linha de produto. A modelagem de features é a atividade de identificar as características externas visíveis na linha de produto e organizar essas features em um modelo. As features também são uma maneira de identificar os requisitos da linha de produto, os quais são requisitos funcionais ou não funcionais. A

3 Do inglês Feature-Oriented Domain Analysis

variabilidade dessas features pode ser dividida em obrigatória, opcional e alternativa, a saber: (1) *obrigatória* são features que são comuns a diferentes produtos; (2) *opcional* são *features* que podem ou não estar presentes; e (3) *alternativa*, quando uma *feature* é composta por um conjunto de features. Nesse caso, é possível especificar se é permitido escolher somente uma ou mais, ou se a escolha de uma *feature* exclui outras.

Ao desenvolver o diagrama de features, é possível identificar três tipos de relacionamento entre features: composição, generalização ou especialização e implementado por, a saber: (1) *composição* é o relacionamento que indica uma dependência mútua ou uma dependência mutua exclusiva, utilizado para restringir a seleção de features opcionais ou alternativas; (2) *generalização/especialização*, utilizado para indicar entre features genéricas e funcionalidades genéricas; (3) *implementado por*, indica quem implementa a *feature*.

Figura 3 - Exemplo de diagrama de *features*



Fonte: Adaptado de (KANG et al., 1990)

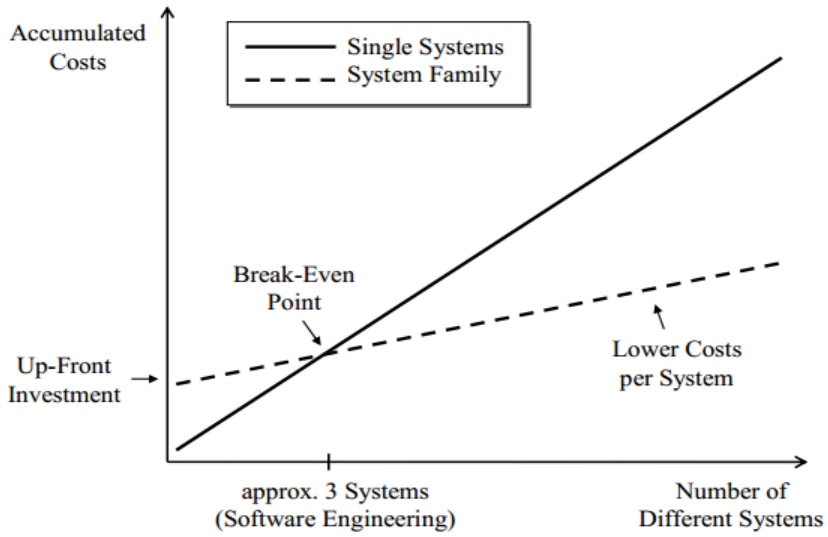
2.1.2 Vantagem do Uso de LPS

A LPS foi concebida para realizar o reuso e permitir que softwares possam ser customizados em massa. A seguir são listadas as principais vantagens de acordo com (POHL et al., 2005):

- *Redução do custo de desenvolvimento*: Umas das principais justificativas para o uso de LPS é a redução de custo. Quando os artefatos da plataforma são reutilizados em diferentes sistemas o custo de cada sistema reduz. Para realizar a criação desses artefatos é necessário que a plataforma seja planejada e desenvolvida, para isso será necessário que seja realizado um investimento inicial, permitindo a criação desses artefatos para posterior reuso em outros sistemas. A Figura 4 mostra que o custo de uma LPS tem o seu *break-even*

entre três e quatro sistemas, a partir desse ponto é possível observar a redução de custos providos pela LPS. O *break-even* pode variar dependendo das características da organização e do mercado, tais como a base de clientes, domínio de negócio e variabilidade de produtos.

Figura 4 - Custo de uma LPS



Fonte: (Pohl, Böckle, & Van der Linden, 2005)

Melhora da qualidade: os artefatos produzidos são revisados e testados em muitos produtos. Dessa forma, a possibilidade de encontrar um defeito aumenta, o que possibilita uma rápida correção do artefato para todos os produtos.

- *Redução do time-to-market*: um produto utilizando LPS tem o potencial de ter o seu tempo de desenvolvimento reduzido, uma vez que os principais artefatos já foram desenvolvidos.
- *Evolução planejada e organizada*: à medida que novos produtos são desenvolvidos, novos artefatos poderão ser incorporados à plataforma.
- *Satisfação do cliente*: pois o prazo de entrega e custo é menor.

2.1.3 Desvantagem do Uso de LPS

A implantação de uma linha de produto exige uma mudança de cultura da empresa e uma mudança na forma de desenvolvimento dos produtos, podendo ocasionar uma resistência por parte das pessoas envolvidas. Com o aumento das pessoas nas organizações percebesse

que a LPS não é escalável, pois é necessário reorganizar as equipes e criar uma unidade específica para esse fim, quando existem diferentes unidades é preciso tomar cuidado para evitar conflitos com a unidade de negócios, pois com o aumento de pessoas envolvidas a comunicação pode ficar prejudicada.

No desenvolvimento da LPS, ocorre um aumento natural do foco no domínio da aplicação não dando foco na criação de funcionalidades que podem ser compartilhadas. Essa é uma das principais causas da erosão arquitetural dos componentes da família de produtos. Para permitir a evolução da linha de produto, é necessário que haja uma cultura e um compromisso das pessoas envolvidas na construção da família de produtos (BOSCH, 2001; HEYMANS; TRIGAUX, 2003).

Uma desvantagem que a linha de produtos traz é a agilidade para realizar mudanças, se um componente da linha de produto deve ser alterado de forma ágil para se adequar a alguma mudança não prevista – acompanhar um concorrente ou uma nova oportunidade de mercado – um esforço de sincronização é exigido para realizar a alteração e ainda contemplar os sistemas que já utilizam o componente, nem que para isso seja necessário lançar versões diferentes do mesmo componente (BOSCH, 2001). Muitos engenheiros de software não possuem uma visão global da arquitetura. Consequentemente, talvez seja necessário um especialista do domínio da linha de produto. Uma pessoa com mais experiência e responsabilidade tende a entender melhor os conceitos da LPS aplicados aquele domínio (CLEMENTS; NORTHROP, 2001).

Por fim, projetar linhas de produtos não é fácil, pois não existe uma diretiva, técnica ou ferramenta que valide a arquitetura de referência, ocasionando riscos técnicos e organizacionais. Sendo assim, é necessário um planejamento de longo prazo, uma vez que a organização faz um investimento e precisa produzir produtos utilizando a LPS para recuperar o seu investimento (HEYMANS; TRIGAUX, 2003).

2.2 Variabilidade

A variabilidade é essencial para a linha de produtos e está associada com a capacidade de se adequar a mudanças, mais precisamente com as mudanças planejadas. Sendo introduzida durante o sub-processo de gestão do produto, quando *features* comuns e variáveis são identificadas na linha de produtos.

Os requisitos de domínio detalham as features definidas no gerenciamento do produto, dessa forma as features exercem forte influência não só nos requisitos de domínio bem como

no design, desenvolvimento e testes. É possível utilizar diferentes níveis de abstração, a cada novo nível de abstração o nível anterior é refinado e novas features são adicionadas, resultando em refinamento. Para isso é necessário introduzir a variabilidade desde a arquitetura para que os componentes possam ser compatíveis com diferentes versões da linha de produtos (POHL et al., 2005).

Dentro desse contexto temos os pontos de variação, um ponto de variação representa uma oportunidade de variação dentro do domínio da LPS. Os pontos de variação são modelados para permitir a customização das aplicações utilizando o reuso definido, permitindo que as customizações possam se ajustar para atender as necessidades propostas.

2.3 Aplicações Web

As aplicações web permitem que qualquer pessoa com um browser possa utilizá-las. Com o aumento da popularidade das aplicações web é possível realizar o acesso não só de computadores comuns, mas também de dispositivos móveis como *tablets* e smartphones. Cada vez mais aplicações web são desenvolvidas e disponibilizadas. Nesse cenário é importante notar que o reuso possui um papel importante. LPS têm se tornado um fator chave para obter sucesso no desenvolvimento de aplicações web, porém é possível observar a falta de tecnologias específicas e a falta de estratégica aplicada ao domínio de aplicação web (ZHOU et al., 2010).

As aplicações web tem se mostrado uma excelente área para a aplicação de LPS, pois torna o desenvolvimento mais ágil, segundo (BALZERANI et al., 2005) as aplicações web podem ser consideradas produtos de software derivados de uma infraestrutura comum, onde o core é a abstração do domínio, por exemplo, carrinho de compras, *login*, registro de usuários e sistema de varejo.

2.4 Arquitetura da Plataforma

Uma dos ativos mais importantes de uma linha de produto é a arquitetura[4] (PLA), também conhecida como plataforma. Através de uma plataforma comum os produtos podem ser customizados, onde é possível produzir diferentes produtos baseado em uma plataforma. A arquitetura deve explicitar os pontos comuns e a variabilidade, permitindo compartilhar os

[4] Do inglês *Product Line Architecture*

elementos comuns da LPS, facilitando a customização dos produtos, nela estão definidos os principais ativos da LPS que foram identificados durante a análise do domínio: requisitos, design, implementação, testes, etc. É um ponto estratégico para a organização que visa adotar uma abordagem de linha de produto, a criação ou retirada de uma plataforma exerce uma forte influência no sucesso empresarial. (POHL et al., 2005).

A construção da plataforma exige um investimento inicial da organização, pois será necessário realizar um esforço inicial, somente após a sua conclusão será desenvolvido os produtos. A remoção ou adição de novas features na plataforma afeta todos os produtos que a utilizam, dessa forma é possível manter e evoluir a plataforma permitindo que organização se adapte as necessidades do mercado. Outros requisitos que devem ser atendidos pela plataforma são a segurança e desempenho, a plataforma deve abstrair para os produtos essas particularidades.

2.5 Métricas de Software

A métrica de software é a forma de mensurar um software. Utilizada como ferramenta para realizar a análise do software produzido, fornecendo informações que podem auxiliar no desenvolvimento do mesmo. As métricas de software podem ser divididas em dois grupos: métricas do produto de software e métricas do processo de software. Enquanto as métricas do produto de software são as métricas extraídas do código fonte ou do design de código, por exemplo, as métricas do processo de software são as métricas extraídas do processo utilizado para o desenvolvimento do software.

A utilização de métricas é importante para guiar a organização e auxiliar na tomada de decisões. É recomendado que a extração de métricas seja realizada desde as primeiras fases do projeto (LI, 2000). A extração de métricas da PLA possui dois objetivos: avaliar a qualidade e servir como base para analisar a gestão e o valor econômico da linha de produto. A análise do impacto de variabilidade pode determinar o valor agregado de uma linha de produto para a organização. A aplicação de métricas na PLA fornece indicadores eficientes para avaliar se a PLA está cumprindo com o seu papel dentro da linha de produto da qual faz parte (OLIVEIRA JUNIOR et al., 2010).

3 LINHA DE PRODUTO PROPOSTA

A literatura atual em Engenharia de Software propõe um conjunto de metodologias para o desenvolvimento de LPS, as quais especificam um conjunto de atividades e artefatos de software. Porém, tais metodologias são imprecisas na definição de quais atividades devem ser executadas, bem como os artefatos gerados, durante o projeto de uma LPS para aplicações Web. Sendo assim, o projeto da LPS proposta nesta monografia terá duas etapas. A primeira, chamada de Engenharia de Domínio, foca na elicitação dos requisitos e na especificação das configurações dos produtos. A segunda, chamada de Engenharia de Aplicação, descreve o projeto de LPS ao apresentar a arquitetura da linha, os diagramas de classes para cada *feature* e o mapeamento das features em classes; tudo baseado nos requisitos e configurações de produtos apresentados na etapa anterior. Esse alinhamento entre as suas etapas visa potencializar o reuso dos artefatos de software, algo central em LPS.

3.1 Engenharia de domínio

A etapa de Engenharia de Domínio foca na elicitação dos requisitos, na especificação das variabilidades e definição da configuração dos produtos da linha. Sendo assim, um conjunto de atividades são executadas, incluindo (1) a descrição dos requisitos da LPS proposta, bem como as possíveis versões da linha; (2) a identificação da variabilidade da LPS e especificação do diagrama de *feature*; (3) a descrição das configurações dos produtos que poderão ser derivados da LPS; e, por fim, (4) a criação do diagrama de casos de uso, bem como a especificação dos casos de uso.

3.1.1 Descrição dos Requisitos

O desenvolvimento da linha de produtos proposta utiliza uma abordagem proativa, o que segundo (KANG et al., 2010) é indicado para domínios maduros e estáveis, permitindo que as características da linha de produto possam ser planejadas. A construção da linha de produtos foi dividida em versões, permitindo agregar novas funcionalidades e melhorando as funcionalidades existentes à medida que cada nova versão é lançada. A utilização de versões também tem como foco a modularidade, permitindo que sejam lançadas versões estáveis e funcionais da linha de produto a cada nova versão, facilitando a manutenção e evolução das funcionalidades que são desenvolvidas.

A primeira versão da linha de produtos tem como base algumas características comuns em uma aplicação web, a capacidade de gerenciar e manter as informações manipuladas pelas aplicações web é uma característica básica, e serve como base para as demais funcionalidades. As outras funcionalidades dessa versão fornecem subsídio para gerenciar os usuários da aplicação, bem como o controle de autenticação e autorização dos mesmos. Para facilitar a visualização das versões o Quadro 1 resume as funcionalidades de cada versão.

Quadro 1 - Versões da linha de produto de software

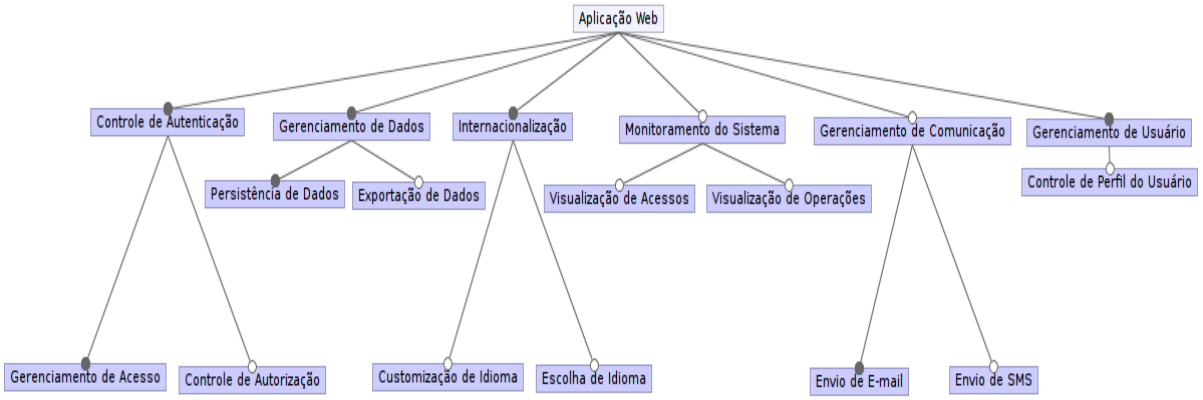
Versão	**Descrição**
Versão 1	*Features* obrigatórias e mais importantes para a linha de produto de aplicações web
Versão 2	Adição das *features* obrigatórias restantes, relacionadas a internacionalização. Refatoração das funcionalidades da versão 1 para uso das novas funcionalidades
Versão 3	Adição de *features* opcionais que acrescentam a capacidade de monitorar as ações realizadas na aplicação web
Versão 4	Adição de *features* opcionais que acrescentam a capacidade da aplicação web interagir com o usuário
Versão 5	Adição da *feature* opcional para exportação de dados

Fonte: Elaborado pelo autor.

3.1.2 Diagrama de Features

A construção da linha de produtos desenvolvida nessa monografia tem como foco o domínio de aplicações web. É possível observar em diversas aplicações que várias features se repetem, mesmo que a finalidade de cada sistema seja totalmente diferente. Com base na notação FODA, as features da linha de produtos são apresentadas na Figura 5.

Figura 5 - Diagrama de features da LPS



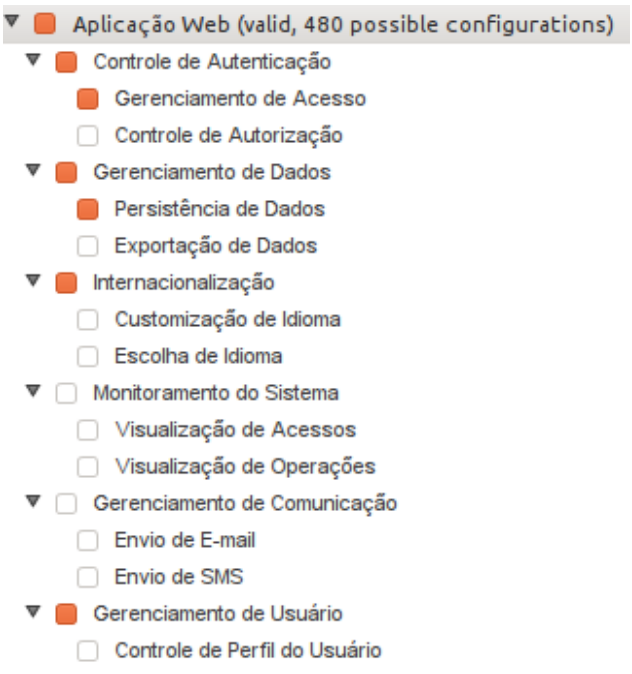
Fonte: Elaborado pele autor

3.1.3 Configuração dos produtos da linha de produto

A configuração da linha de produtos tem como objetivo central definir a configuração de um produto. Isto é, as *features* que são necessárias para gerar um produto. Baseado no diagrama de *features* apresentado anteriormente são apresentadas duas configurações de dois produtos. Para isso, a ferramenta *FeatureIDE* foi utilizada para especificar o produto a ser gerado.

- *Configuração de um produto com as features obrigatórias.*

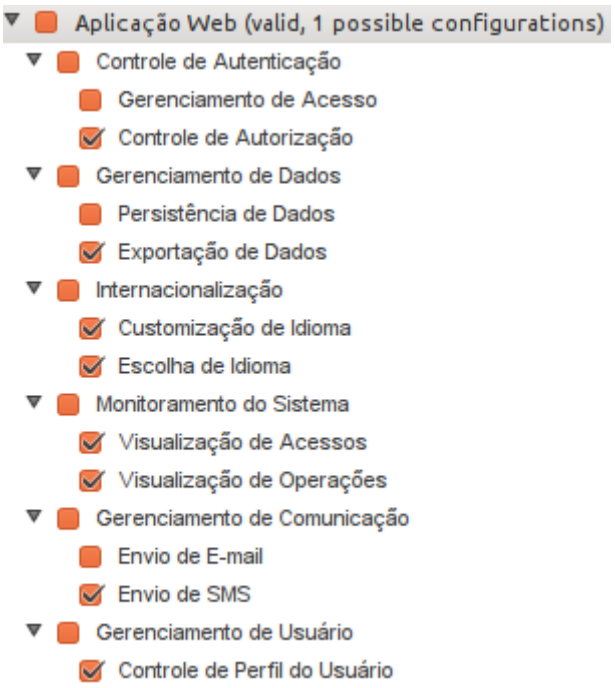
Figura 6 - Configuração de um produto com as features obrigatórias



Fonte: Elaborado pelo autor

- *Configuração utilizando todas as features disponíveis.*

Figura 7 - Configuração de um produto com todas as features

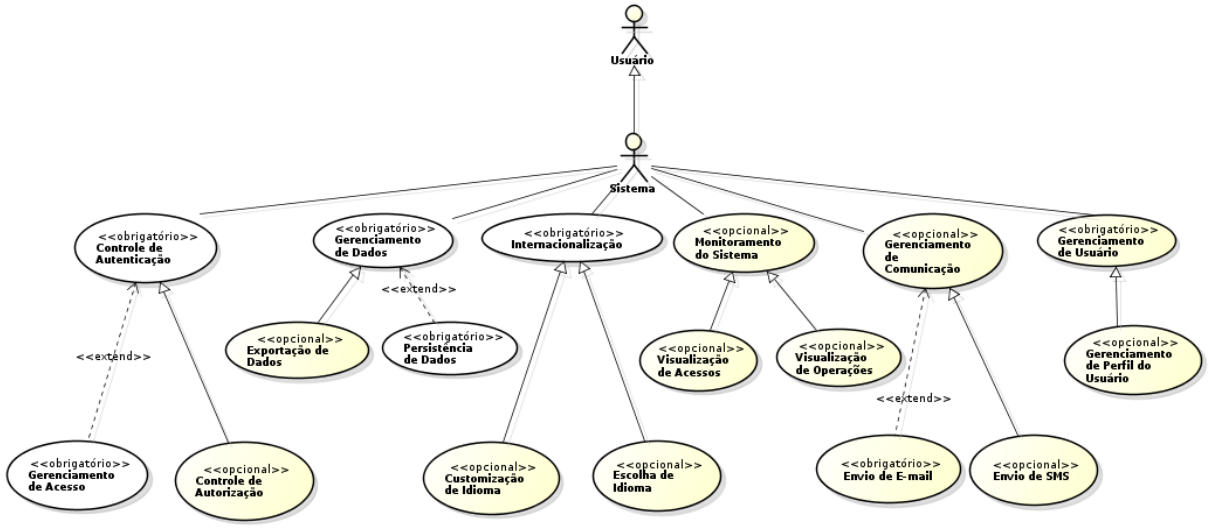


Fonte: Elaborado pelo autor

3.1.4 Diagrama de casos de uso

Baseado na descrição dos requisitos e no diagrama de *feature* é apresentado o diagrama de casos de uso da LPS. Para cada *feature* foi criado um caso de uso visando potencializar o reuso e melhorar a rastreabilidade entre as *features* e os casos de uso.

Figura 8 - Diagrama de casos de uso



Fonte: Elaborado pelo autor

3.1.5 Especificação dos casos de uso

A especificação dos casos de uso tem como objetivo identificar e documentar os requisitos necessários para a linha de produto. Esta especificação apresenta os detalhes necessários para a implementação dos casos de uso representados de forma abstrata no Diagrama de casos de uso da Figura 8. Através da especificação dos requisitos é possível identificar os pontos de variabilidade, permitindo assim que o produto possa ser customizado de acordo com as necessidades. É importante destacar que cada *feature* foi mapeada para um caso de uso. Os casos de uso serão apresentados de forma sucinta a seguir (a especificação completa pode ser visualizada no APÊNDICE A).

Quadro 2 - Descrição dos casos de uso

Caso de uso	Feature	Descrição
Gerenciamento de Acesso	Gerenciamento de Acesso	Descreve o processo para manter o histórico dos acessos ao sistema
Gerenciamento de	Gerenciamento de	Descreve as regras e funcionamento do processo de

Autenticação	Autenticação	autenticação
Gerenciamento de Autorização	Gerenciamento de Autorização	Descreve as regras e funcionamento do processo de autorização a determinados recursos. Concedendo ou negando o acesso ao recurso
Gerenciamento de Dados	Gerenciamento de Dados	Descreve os requisitos necessários para manter os dados gerados pelo sistema
Exportação de Dados	Exportação de Dados	Descreve a processo para exportar o resultado das pesquisas realizadas no sistema
Persistência de Dados	Persistência de Dados	Descreve o processo necessário para realizar a persistência e busca das informações geradas pelo sistema
Internacionalização	Internacionalização	Descreve o processo de internacionalização da linha de produto
Customização de Idioma	Customização de Idioma	Descreve os requisitos necessários para permitir que os produtos possam customizar o idioma
Escolha de Idioma	Escolha de Idioma	Descreve os requisitos necessários para permitir que os produtos possam alternar entre diferentes idiomas
Monitoramento do Sistema	Monitoramento do Sistema	Descreve o processo necessário para que seja possível monitorar quais usuários estão conectados no sistema
Visualização de Acessos	Visualização de Acessos	Descreve os requisitos necessários para visualizar os acessos realizados ao sistema
Visualização de Operações	Visualização de Operações	Descreve os requisitos necessários para visualizar as operações de manipulação dos dados [5] (DML) realizadas por usuário no sistema
Gerenciamento de Comunicação	Gerenciamento de Comunicação	Descreve o processo necessário para que o sistema possa interagir com meios externos
Envio de E-mail	Envio de E-mail	Descreve os requisitos necessários para realizar o envio de e-mails
Envio de SMS	Envio de SMS	Descreve os requisitos necessários para realizar o envio de *Short Message Service* (SMS)
Gerenciamento de Usuário	Gerenciamento de Usuário	Descreve os requisitos necessários para realizar a criação ou atualização dos usuários que tem acesso ao sistema
Controle de Perfil de Usuário	Controle de Perfil de Usuário	Descreve os requisitos necessários para adicionar, atualizar ou remover os perfis dos usuários que tem acesso ao sistema

Fonte: Elaborado pelo autor

[5] Do inglês *Data Manipulation Language*

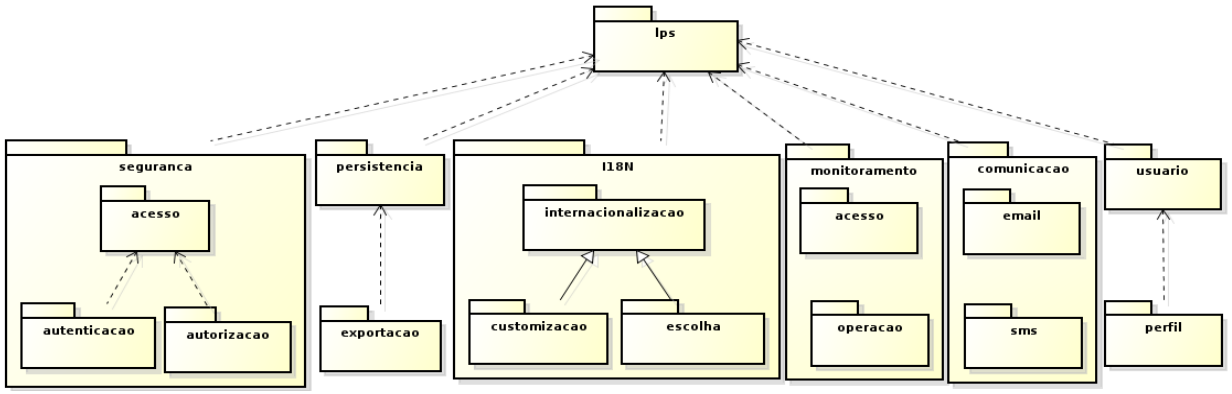
3.2 Engenharia de Aplicação

A etapa de Engenharia de Aplicação foca no projeto da linha de produto levando em consideração os artefatos apresentados anteriormente. Sendo assim, um conjunto de atividades são realizadas, incluindo (1) a definição da arquitetura da LPS proposta; (2) a criação do diagrama de classes; e, por fim, (3) o mapeamento das *features* em classes.

3.2.1 Arquitetura da LPS

Para um melhor entendimento as features foram separadas em pacotes, a Figura 9 mostra o diagrama de pacotes da linha de produtos.

Figura 9 - Diagrama de de pacotes

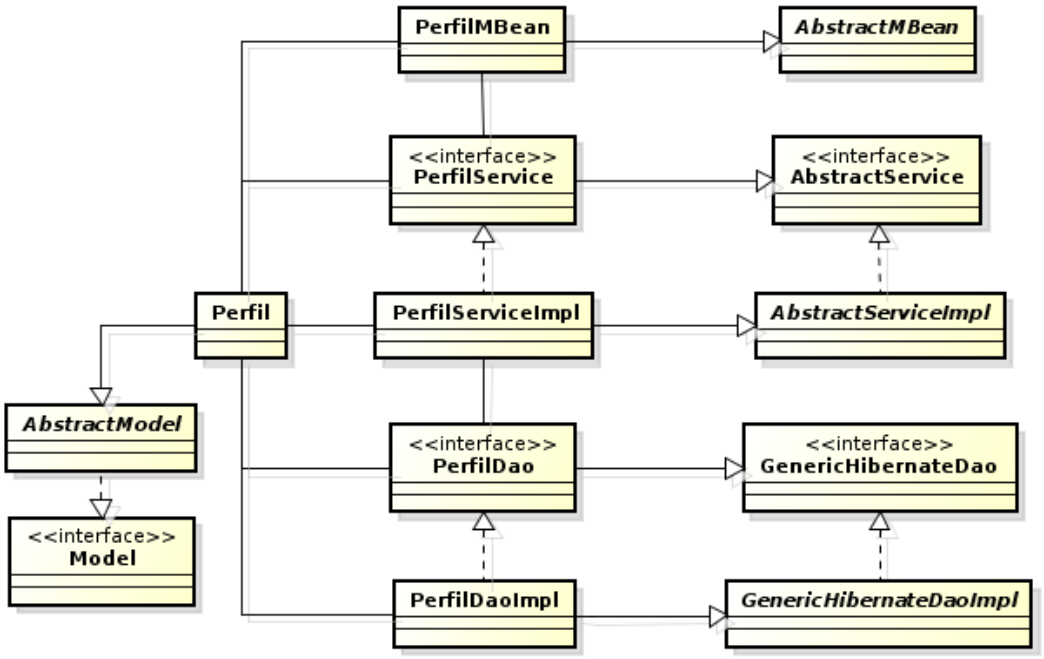


Fonte: Elaborado pelo autor.

3.2.2 Diagrama de classes

A modelagem das classes da linha de produto define a relação entre as classes bem como a estrutura. Durante a fase de modelagem é possível validar as dependências entre as *features*, identificando oportunidades de variabilidade nas classes. Para exemplificar, a Figura 10 mostra o diagrama de classes construído para a *feature* Controle de Perfil do Usuário.

Figura 10 - Exemplo do diagrama de classes da *feature* Controle de Perfil do Usuário



Fonte: Elaborado pelo autor.

Para cada *feature* da linha de produtos foi desenvolvido um diagrama de classes, buscando identificar as oportunidades de variabilidade e um maior grau de granularidade. Os diagramas de cada *feature* podem ser encontrados no APÊNDICE B desta monografia.

3.2.3 Mapeamento das *Features* em Classes

O mapeamento das classes e interfaces utilizadas em cada *feature* tem como objetivo identificar os objetos necessários para implementação. A seguir é apresentado o Quadro 3 detalhando as classes que participam de cada *feature*.

Quadro 3 - Mapeamento das classes de cada *feature*

Feature	**Classes**
Gerenciamento de Autenticação	UserDetailsServiceImpl, UsuarioService, AbstractService, PessoaService, UsuarioServiceImpl, AbstractServiceImpl, PessoaServiceImpl, UsuarioDao, GenericHibernateDao, PessoaDao, UsuarioDaoImpl, GenericHibernateDaoImpl, PessoaDaoImpl, AbstractUser, Usuario, Pessoa, Perfil, Modulo, Menu, AbstractModel, Model
Gerenciamento de Acesso	LoginSecurityListener, HistoricoLogin, LoginSecurityListenerImpl, HistoricoAcessoNegado, HistoricoLoginService, AbstractService HistoricoAcessoNegadoService, HistoricoLoginServiceImpl,

	AbstractServiceImpl, HistoricoAcessoNegadoServiceImpl, HistoricoLoginDao, GenericHibernateDao, HistoricoAcessoNegadoDao, HistoricoLoginDaoImpl, GenericHibernateDaoImpl, HistoricoAcessoNegadoDaoImpl, AbstractModel, Model
Gerenciamento de Autorização	FilterInvocationSecurityMetadataSourceImpl, SecureResourceCacheService, WebSecureResourceCacheServiceImpl MenuDao, GenericHibernateDao, MenuDaoImpl, GenericHibernateDaoImpl, Menu, AbstractModel, Model
Gerenciamento de Dados	GenericHibernateDao
Exportação de Dados	ExportacaoUtils, GenericHibernateDao, GeradorRelatorio
Persistência de Dados	GenericHibernateDaoImpl
Internacionalização	I18NProvider, ResourceBundleProvider
Customização de Idioma	ResourceBundleProvider, LocaleUtil
Escolha de Idioma	EscolhaIdiomaController, I18NProvider, BundleJSF
Monitoramento da Aplicação	MonitoramentoMBean, MonitoramentoService, MonitoramentoServiceImpl, SessionUtils, MonitoramentoUsuario
Visualização de Acessos	HistoricoAcessoMBean, AbstractMBean, HistoricoService, HistoricoLogin, HistoricoAcessoServiceImpl, HistoricoAcessoNegado, HistoricoLoginService, AbstractService, HistoricoAcessoNegadoService, HistoricoLoginServiceImpl, AbstractServiceImpl, HistoricoAcessoNegadoServiceImpl, HistoricoLoginDao, GenericHibernateDao, HistoricoAcessoNegadoDao, HistoricoLoginDaoImpl, GenericHibernateDaoImpl, HistoricoAcessoNegadoDaoImpl, HistoricoAcesso, AbstractModel, Model
Visualização de Operações	OperacaoMBean, AbstractMBean, OperacaoService, AbstractService, OperacaoServiceImpl, AbstractServiceImpl, OperacaoDao, GenericHibernateDao, OperacaoDaoImpl, GenericHibernateDaoImpl, Operacao, AbstractModel, Model
Gerenciamento de Comunicação	Comunicacao
Envio de E-mail	Comunicacao, MailUtils
Envio de SMS	Comunicacao, SmsUtils
Gerenciamento de Usuário	UsuarioMBean, AbstractMBean, UsuarioService, AbstractService, UsuarioServiceImpl, AbstractServiceImpl, UsuarioDao, GenericHibernateDao, UsuarioDaoImpl, GenericHibernateDaoImpl, Usuario, AbstractModel, Model
Controle de Perfil de Usuário	PerfilMBean, AbstractMBean, PerfilService, AbstractService, PerfilServiceImpl, AbstractServiceImpl, PerfilDao, GenericHibernateDao, PerfilDaoImpl, GenericHibernateDaoImpl, Perfil, AbstractModel, Model

Fonte: Elaborado pelo autor

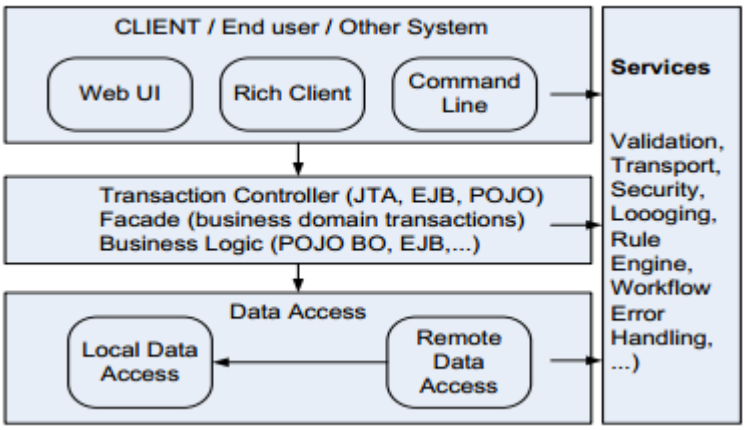
4 TRABALHOS RELACIONADOS

Este Capítulo tem como objetivo fazer uma análise comparativa do trabalho proposto com os trabalhos encontrados na literatura. A análise comparativa é apresenta nas próximas três Seções. Por fim, na Seção 4.4, é apresentada uma comparação entre os trabalhos relacionados e a LPS proposta.

4.1 Software Product Lines: Source Code Organization for 3-tier OLTP Architecture Systems

Em (ROŠKO, 2011), o autor apresenta uma arquitetura de linha de produto de software utilizando o conceito de camadas. A arquitetura proposta tem como objetivo agrupar todos os elementos que são comuns a uma linha de produto que utiliza frameworks de mercado como: Enterprise JavaBeans (EJB), Java Message Service (JMS), Java Transaction API (JTA), Java EE Connector Architecture (JCA), Spring, Struts, Hibernate e TopLink. Com a separação em camadas é possível desenvolver serviços que possuem interesses transversais como transação, segurança, *logging*, cache, tratamento de exceções, dentre outros. A arquitetura foi dividida em três camadas: (1) *apresentação*, podendo ser uma aplicação web, uma aplicação desktop ou através de linhas de comando; (2) *camada de lógica de negócio*, onde são processadas as regras de negócio; e (3) *camada de acesso a dados*, responsável por manipular os dados.

Figura 11 - Camadas da LPS proposta por (ROŠKO, 2011)



Fonte: (ROŠKO, 2011)

É possível observar que para realizar o desenvolvimento em camadas o autor do trabalho utiliza alguns padrões de projeto para auxiliar a construção da LPS como: Data Transfer Object (DTO), Facade, Proxy, Data Access Object (DAO) e Model–View–Controller (MVC).

Para separar melhor o código o autor divide a arquitetura proposta em três módulos:

- *Client*: módulo que contém todas as partes específicas que são necessárias para realizar a apresentação da aplicação para o usuário. Essa camada pode ser implementada utilizando Google Web Toolkit (GWT), *Servlet*, JavaServer Pages (JSP), JavaServer Faces (JSF), Standard Widget Toolkit (SWT) ou Swing. Será essa camada responsável por iniciar a comunicação com a camada de lógica de negócios.
- *Common*: nesse módulo estão as classes utilitárias da LPS (para tratamento de exceção, segurança, sessão, cache, DTO's, Proxy, Facades, e-mail, dentre outros) que abstraem os detalhes técnicos de desenvolvimento, permitindo a variabilidade na LPS se alguma dessas funcionalidades for necessária.
- *Server*: módulo de contém todas as classes Facades, BO's, e DAO's necessárias para o desenvolvimento da camada de lógica de negócios e de lógica de acesso a dados. Será através desse módulo que a escolha por algumas *features* centrais será realizada, como o tipo de transação - Plain Old Java Object (POJO) ou EJB - para processamento da lógica de negócios.

A arquitetura em três camadas proposta por (ROŠKO, 2011) atende a um número grande de domínios tais como, bancos, sistemas de telecomunicação, sistema de viagens, dentre outros.

4.2 Concepts and Implementation Techniques for Web Systems Product-Lines Using Existing Frameworks

Apresenta os conceitos, técnicas de implementação necessárias para permitir o reuso sistemático de sistemas web e algumas métricas de avaliação. De acordo com (MATHER, 2011), a forma mais predominante de documentação utilizada é a construção de diagramas da UML para demostrar conceitualmente as classes, adicionalmente é utilizada a construção de diagramas customizados e tabelas para representar graficamente os conceitos descritos em texto como decisões arquiteturais. Uma das formas mais utilizadas para avaliar os benefícios

de uma linha de produtos é a redução de linhas de código, uma métrica bem estabelecida e bastante utilizada, o que superficialmente é válido, mas é possível imaginar casos onde a redução de linhas de código pode aumentar a dificuldade de implementação. Dessa forma, se faz necessário a utilização de outras formas de avaliação.

Baseado na análise de domínio, no módulo principal e nos ativos de variabilidade a linha de produtos pode ter o seu design modelado através do diagrama UML de casos de uso, diagrama UML de sequência, protótipos de interface, diagrama UML de classes, e diagrama UML de componentes.

Na implementação da LPS, diferentes tecnologias podem ser utilizadas como, por exemplo, C#/ASP.NET para implementação do *SharePoint* ou Java/JSP para implementação do *Liferay*. Uma aplicação corporativa que interage com a lógica de negócios envolve uma arquitetura dividida em camadas utilizando padrões de projeto: mapeamento objeto relacional, DAO, DTO, BO, entre outros.

A pesquisa realizada pelo autor foi aplicada em uma pequena empresa para permitir uma abordagem extrativa e reativa mais ágil. A linha de produtos foi desenvolvida com base em uma aplicação inicial e, consequentemente, utiliza uma abordagem reativa que permite expandir a LPS para novas aplicações, novos domínios e novos clientes. Os artefatos para geração da LPS foram derivados de um modelo de variabilidade utilizando o modelo de features.

A aplicação final foi dividida em três camadas: camada de apresentação, camada de aplicação, e camada de acesso a dados. Foi possível observar que os conceitos de variabilidade podem estar presentes em uma camada particular ou podem se expandir entre as camadas. Para a implementação foi utilizada os princípios SOLID (*Single Responsability*, *Open-Closed*, *Liskov Substitution*, *Interface Segregation* e *Dependency Inversion*) que potencializam alta coesão e baixo acoplamento na aplicação. Consequentemente, será possível adicionar novas classes e novos componentes de forma mais fácil durante a abordagem reativa, sem a necessidade de modificar os componentes já existentes. A aplicação de diversos princípios de projeto e padrões de projeto facilita a construção de linhas de produtos.

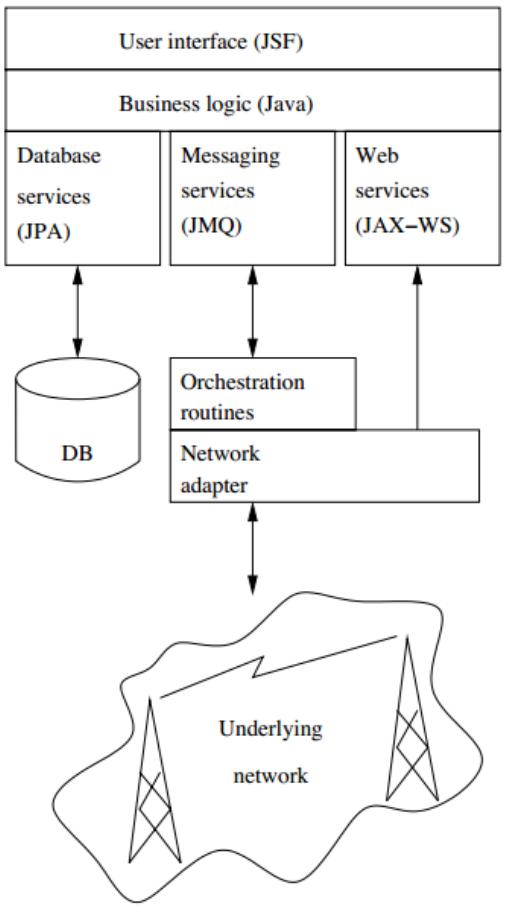
A separação de conceitos propicia um aumento de variabilidade, uma forma básica é separar os *estilos Cascading Style Sheets* (CSS) do conteúdo HyperText Markup Language (HTML), por exemplo, ou a separação das classes de modelo do domínio da camada de lógica de acesso a dados, utilizando uma ferramenta de mapeamento objeto relacional permitindo a utilização de diferentes bases de dados. A programação orientada a aspectos fornece uma

forma de implementar funcionalidades que tem preocupações transversais, é esperada segundo o autor que ela seja utilizada no desenvolvimento de linhas de produto.

4.3 Bottom-up modeling for a software product line

Este trabalho apresenta a experiência adquirida na construção de uma linha de produtos utilizando a modelagem de *features* para definir a combinação do serviço de telecomunicação entre requisições mobile. Segundo (POHJALAINEN, 2011) um dos objetivos desse trabalho é demonstrar que é possível desenvolver uma LPS utilizando modelagem de *features* combinado com a metodologia de desenvolvimento ágil utilizando uma abordagem *botton-up*.

Figura 12 - Arquitetura da LPS utilizada por (POHJALAINEN, 2011)



Fonte: (POHJALAINEN, 2011)

A imagem acima (Figura 12) mostra em alto nível a visão da arquitetura. As setas verticais representam a comunicação entre diferentes máquinas e as caixas representam os

componentes de software principais. A camada de visualização utiliza JSF e a camada de lógica de negócios que realiza o roteamento das requisições foi implementada utilizando puramente Java. O acesso ao banco de dados foi implementado utilizando Java *Persistence* API (JPA) como *Application Programming Interface* (API) e *Hibernate* como implementação, a entrega das requisições para rede são roteadas utilizando Java Message Service (JMS), a comunicação dos equipamentos de rede com o *front-end* utiliza Java API for RESTful Services (JAX-RS), permitindo que a cada mudança de status dos equipamentos a base de dados local de cada aplicativo fosse atualizada. Para a construção dos componentes de software e bibliotecas optou-se por utilizar a forma mais padronizada possível, utilizando o padrão de tecnologias fornecidas pela API do Java onde fosse possível.

À medida que os novos clientes foram aceitos a linha de produtos foi aprimorada, pois era necessário realizar modificações de código, adicionar novos serviços e os requisitos eram modificados. A utilização do modelo de features teve diversos benefícios, o primeiro deles, a validação para escolher onde um determinado conjunto de serviços deveria ser inserido e se era válido ou muito genérico. Em segundo lugar, permitiu analisar a rotina de validação utilizando o modelo e evitando a análise manual de código, segundo (POHJALAINEN, 2011) outro benefício que pode ser citado, dentre outros que podem ser enumerados, é a avaliação das configurações possíveis na linha de produtos evitando as configurações manuais que podem gerar erros. Combinar o desenvolvimento de linha de produtos com métodos ágeis foi um desafio, pois a linha de produtos tem como premissa realizar um planejamento mais aprofundado enquanto as práticas do princípio ágil possuem um foco maior na entrega evitando uma etapa de preparação mais demorada.

A modelagem orientada a objetos é maneira tradicional de modelar sistemas orientados a objetos, o MVC provê estrutura para transformar esses modelos em objetos, mas esse trabalho deve ser executado manualmente causando uma baixa produtividade. A abordagem utilizando *Model Driven Architecture* (MDA) não se mostrou a mais adequada por exigir a utilização de ferramentas que são de difícil introdução no processo ágil. Um dos maiores problemas no processo de modelagem formal é a definição de variabilidade em um projeto ágil, dessa forma foi necessária uma abordagem para entregar os resultados no final do Sprint. Com pouco tempo e apenas algumas horas destinadas para pensar na linha de produtos não foi possível integrar as ferramentas comumente utilizadas, a utilização de UML para simular a variabilidade também foi descartada pelos motivos já citados e por tornar o processo de desenvolvimento mais pesado.

Dessa forma foi utilizada uma configuração textual das features, utilizando *Feature Domain Language (FDL)*. É possível converter a anotação FODA para expressões regulares, dessa forma é possível validar as configurações dos produtos e realizar a análise do modelo.

O objetivo principal de realizar a análise do modelo é evitar a introdução de mudanças acidentais, durante os doze meses de desenvolvimento da linha de produtos nenhum processo formal de modelagem de features foi utilizado. As mudanças no modelo de features podem ser classificadas como: generalização, especialização, refatoração e mudanças arbitrárias. A análise de evolução do modelo mostra que 48% das edições foram generalizações da versão anterior, 22% foram de edições arbitrárias, 13% foram de especialização, 13% foram de refatoração e apenas 4% do modelo foram mantidos em sua versão inicial.

Segundo o autor esse trabalho mostra que o uso de expressões para a modelagem e análise de domínio é leve e atendeu as necessidades do projeto. E apesar de historicamente na área da Engenharia de Software diversas ferramentas prometerem aumentar a produtividade isso ainda não é uma realidade, por essa razão os desenvolvedores de software mais experientes são cautelosos para investir em ferramentas ou técnicas sensacionalistas. Muitas ferramentas ou técnicas são incluídas em um contexto especifico, são incompatíveis entre diferentes arquiteturas, são apenas jogadas de marketing ou são muito difíceis ou é impossível utilizá-las em outro contexto.

A modelagem *botton-up* tenta evitar esses problemas de contexto enfatizando o que é necessário para o contexto. Obviamente que essa abordagem não deve ser aplicada a todos os contextos, pelo contrário, a modelagem do autor foi utilizada para facilitar os sprints e continuar trabalhando com o código de forma tradicional. Ao aparecer um requisito novo que não se enquadre na abordagem utilizada é possível voltar à forma tradicional.

Segundo o autor é preciso priorizar a entrega das features e se não existir uma boa ferramenta por uma razão ou outra, você pode construir a sua própria e ir evoluindo ao longo das iterações se necessário. Deve ser evitado um apelo emocional com as ferramentas desenvolvidas pela equipe, é necessário procurar constantemente por oportunidades de troca dessa ferramenta por outra melhor. O modelo criado tem como objetivo atender a um modelo de comunicação de rede do governo e algumas organizações, e não uma rede de comunicação pública. Por esse motivo o número de serviços modelados foi baixo, permitindo que o modelo pudesse evoluir com a ferramenta desenvolvida internamente, caso contrário seria necessário investir em uma ferramenta de modelagem de features mais madura e considerar o desenvolvimento utilizando uma abordagem tradicional, não ágil.

4.4 Análise comparativa dos trabalhos relacionados e a linha de produtos proposta

A comparação entre os trabalhos relacionados tem como objetivo identificar as similaridades e as diferenças entre o trabalho proposto e os trabalhos previamente apresentados. Para realizar a análise foram estabelecidos alguns critérios para efeito de comparação, os critérios que foram escolhidos são:

- *Uso de camadas***:** permite a separação do sistema em camadas lógicas, facilitando o entendimento e a manutenção.
- *Uso de padrões de projeto*: A utilização de padrões aumenta a qualidade do código por aplicar soluções previamente validadas.
- *Uso de CDI:* o CDI é a API padrão do Java para gerenciamento de contexto e injeção de dependências. Permite o uso de contextos e injeção de dependências sem o uso de frameworks externos a API padrão como o *Spring*.
- *Tipo de construção*: a construção de linhas de produto pode ser do tipo: reativa, proativa e extrativa.
- *Linguagem*: a linguagem de programação utilizada para desenvolver a linha de produto.

O Quadro 4 apresenta o resumo da comparação entre as linhas de produto.

Quadro 4 - Comparação entre trabalhos relacionados e a linha de produtos proposta

	T1	**T2**	**T3**	**T4**
Uso de camadas	Sim	Sim	Sim	Sim
Uso de padrões de projeto	Sim	Sim	Sim	Sim
Uso de CDI	Não	Não	Não	Sim
Tipo de construção	Proativa	Reativa	Reativa	Proativa
Linguagem	Java	C#	Java	Java

T1: Source Code Organization for 3-tier OLTP Architecture Systems

T2: Concepts and Implementation Techniques for Web Systems Product-Lines Using Existing Frameworks

T3: Bottom-up modeling for a software product line

T4: Linha de produto proposta

5 CONSIDERAÇÕES FINAIS

A utilização de aplicações Web está crescendo para acompanhar a demanda por novos produtos. O uso de LPS pode ajudar as organizações a suportar esta demanda por potencializar o reuso de módulos dos produtos ao passo que potencializa a qualidade dos produtos gerados, reduz tempo e custo de desenvolvimento. Nesse cenário, a LPS exerce um papel muito importante, pois permite que as organizações aumentem a produtividade ao mesmo tempo em que oferecem produtos com preços competitivos.

Essa monografia, portanto, apresentou um estudo sobre LPS, análise e projeto de uma LPS para o domínio de aplicações Web. As *features* que formam a linha de produto proposta foram identificadas e documentadas, bem como foi apresentado um projeto que descreve como tais features serão implementadas. O projeto da LPS utilizou os artefatos preconizados pela UML, incluindo casos de uso, diagrama de classes, e pacotes, bem como artefatos essenciais para expressar a variabilidade da linha de produto, incluindo o diagrama de *features*. A LPS apresentada nesse estudo poderá ser utilizada para derivar novas aplicações Web, incentivando o reuso sistemático das *features* e permitindo com que os novos produtos possam ser desenvolvidos em um prazo menor.

Os trabalhos futuros serão concentrados na execução do projeto apresentado nesta monografia, bem como na realização de testes e a derivação de produtos da LPS. Através dos produtos derivados será possível, por exemplo, realizar uma análise mais aprofundada da LPS através da extração de métricas. Além disso, métricas serão utilizadas para comparar um produto extraído da LPS proposta com um produto que não foi derivado da mesma. Essa comparação permitirá identificar o real ganho do uso de LPS. Além disso, a comparação das métricas poderá indicar os pontos que precisarão ser melhorados na plataforma, bem como os benefícios providos pela LPS.

REFERÊNCIAS

BALZERANI, L.; RUSCIO, D. DI; PIERANTONIO, A.; ANGELIS, G. DE. A product line architecture for web applications. Proceedings of the 2005 ACM symposium on Applied computing - SAC '05. **Anais...** p.1689, 2005. New York, New York, USA: ACM Press. Disponível em: <http://portal.acm.org/citation.cfm?doid=1066677.1067059>. Acesso em: 26/9/2013.

BOSCH, J. Software product lines: organizational alternatives. Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001. **Anais...** p.91–100, 2001. IEEE Comput. Soc. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=919084>. Acesso em: 17/9/2013.

CLEMENTS, P.; NORTHROP, L. **Software Product Lines: Practices and Patterns**. 3rd ed. Addison-Wesley Professional, 2001.

CZARNECKI, K.; EISENECKER, U. **Generative Programming: Methods, Tools, and Applications**. Addison-Wesley Professional, 2000.

HEYMANS, P.; TRIGAUX, J.-C. **Software Product Lines: State of the art**. Namur, 2003.

KANG, K.; COHEN, S.; HESS, J.; NOVAK, W.; PETERSON, S. **Feature-Oriented Domain Analysis (FODA) Feasibility Study**. 1990.

KANG, K.; SUGUMARAN, V.; PARK, S. **Applied software product line engineering**. Taylor and Francis Group, LLC, 2010.

LI, W. Software product metrics. **IEEE Potentials**, v. 18, n. 5, p. 24–27, 2000. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=807276>. Acesso em: 28/9/2013.

MATHER, N. Concepts and implementation techniques for web systems product-lines using existing frameworks. Proceedings of the 15th International Software Product Line Conference on - SPLC '11. **Anais...** p.1, 2011. New York, New York, USA: ACM Press. Disponível em: <http://dl.acm.org/citation.cfm?doid=2019136.2019188>. Acesso em: 23/10/2013.

NORTHROP, L. Software product lines essentials. **Software Engineering Institute, Carnegie Mel-Ion …**, 2008. Software Engineering Institute. Disponível em: <http://www.sei.cmu.edu/library/assets/spl-essentials.pdf>. Acesso em: 23/11/2013.

OLIVEIRA JUNIOR, E. A.; MALDONADO, J. C.; GIMENES, I. M. S. Empirical Validation of Complexity and Extensibility Metrics for Software Product Line Architectures. 2010 Fourth Brazilian Symposium on Software Components, Architectures and Reuse. **Anais...** p.31–40, 2010. IEEE. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5631689>. Acesso em: 28/9/2013.

POHJALAINEN, P. Bottom-up Modeling for a Software Product Line: An Experience Report on Agile Modeling of Governmental Mobile Networks. 2011 15th International Software Product Line Conference. **Anais...** p.323–332, 2011. IEEE. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6030075>. Acesso em: 24/10/2013.

POHL, K.; BÖCKLE, G.; LINDEN, F. J. VAN DER. **Software Product Line Engineering**. Springer, 2005.

ROŠKO, Z. Software Product Lines: Source Code Organization for 3-tier OLTP Architecture Systems. Proceedings of the 22nd Central European Conference on Information and Intelligent Systems. **Anais...** p.7, 2011. Varaždin. Disponível em: <http://www.ceciis.foi.hr/app/index.php/ceciis/2011/paper/viewFile/485/271>. .

WOHLIN, C.; RUNESON, P.; HÖST, M.; et al. **Experimentation in Software Engineering**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

ZHOU, J.; JI, Y.; ZHAO, D.; LIU, J. Platform Engineering in Enterprise Application Development. 2010 International Conference on E-Business and E-Government. **Anais...** p.112–115, 2010. IEEE. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5591387>. Acesso em: 10/9/2013.

APÊNDICE A - Especificações de Casos de Uso

Controle de Autenticação – Caso de Uso

Nome: Controle de Autenticação.

Descrição: Realizar o controle de autenticação do sistema, permitindo somente o acesso de usuários autorizados.

Atores: Usuário.

Pré-condição: Nenhuma.

Pós-condição: Ter acesso à página inicial do sistema.

Fluxo Principal

- Usuário informa *username* e senha.
- Usuário seleciona a opção de *login*.
- Sistema criptografa a senha.
- Sistema valida o usuário de acordo com a RN1.
- Sistema adiciona o usuário autenticado na sessão. (PE1)
- Sistema direciona o usuário para página inicial.
- Fim do fluxo principal.

Fluxo Alternativo (Usuário ou senha incorreto):

- Sistema permanece na tela de autenticação e apresenta mensagem: Usuário e/ou Senha incorretos.
- Fim do fluxo alternativo.

Regras de Negócio

(RN1) Criptografar a senha e confrontar username e senha criptografada com a tabela de usuário no banco de dados.

Pontos de Extensão (PE)

(PE1) Caso de Uso: Gerenciamento de Acesso

Gerenciamento de Acesso – Caso de Uso

Nome: Gerenciamento de Acesso.

Descrição: Registrar os acessos realizados ao sistema gerando histórico.

Atores: Usuário.

Pré-condição: Ter executado o caso de uso Controle de Autenticação.

Pós-condição:: Informação de acesso registrada.

Fluxo Principal

- Sistema recupera o usuário que realizou login com sucesso.
- Sistema registra o acesso do usuário contendo as seguintes informações:
    a. Id do usuário
    b. Identificador da sessão (jsessionid)
    c. IP de origem da requisição
    d. Data do login
- Fim do fluxo principal

Fluxo Alternativo (Usuário ou senha incorreto):

- Sistema recupera o usuário que teve o acesso negado ao sistema.
- Sistema registra a tentativa de acesso do usuário, contendo as seguintes informações:
    a. Identificador do usuário (se encontrar um usuário com o username igual na base de dados).
    b. Username.
    c. Identificador da sessão (jsessionid).
    d. IP de origem da requisição.
    e. Data da tentativa de acesso.
    f. Password utilizado criptografado.
- Fim do fluxo alternativo

Controle de Autorização – Caso de Uso

Nome: Controle de Autorização

Descrição: Realizar o controle de autorização as páginas do sistema.

Atores: Usuário.

Pré-condição: Nenhuma.

Pós-condição: Acesso a página requisitada.

Fluxo Principal

- Usuário seleciona um item de menu ou digita a o caminho da página manualmente e realiza a requisição.

- Sistema verifica se o usuário tem acesso à página requisitada de acordo com a RN1.
- Sistema redireciona o usuário para a página requisitada.
- Fim do fluxo principal

Fluxo Alternativo (Acesso negado):

- Sistema redireciona o usuário para página de acesso negado e apresenta a mensagem: Acesso Negado.
- Fim do fluxo alternativo.

Regras de negócio

(RN1) Verificar se existe um usuário autenticado na sessão e se o mesmo possui acesso à página, se a condição anterior não for atendida redirecionar usuário para página de acesso negado.

Gerenciamento de Dados – Caso de Uso

Nome: Gerenciamento de Dados.

Descrição: Gerenciar os dados do sistema.

Atores: Sistema.

Pré-condição: Nenhuma.

Pós-condição: Informação manipulada pelo sistema.

Fluxo Principal

- Sistema requisita operação no banco de dados.
- Sistema inicia transação (transação somente leitura ou transação de escrita).
- Sistema realiza a operação. (PE1)
- Sistema retorna resultado da operação.
- Fim do fluxo principal.

Pontos de Extensão (PE)

(PE1) Caso de Uso: Persistência de Dados.

Persistência de Dados – Caso de Uso

Nome: Persistência de Dados

Descrição: Persistir as informações manipuladas pelo sistema em banco de dados.

Atores: Sistema.

Pré-condição: Nenhuma.

Pós-condição: Informação persistida na base de dados.

Fluxo Principal

- Sistema recebe solicitação para persistir informações de acordo com a RN1.
- Sistema persiste informações no banco de dados.
- Sistema registra as operações realizadas para fins de auditoria de acordo com a RN4.
- Sistema retorna objeto atualizado de acordo com a RN3.
- Fim do fluxo principal

Fluxo Alternativo (Seleção de registros):

- Sistema recebe solicitação para realizar a busca de registros de acordo com a RN2.
- Sistema retorna o resultado encontrado de acordo com a RN3.
- Fim do fluxo alternativo.

Fluxo Alternativo (Erro ao realizar operação):

- Sistema registra em log o evento do erro.
- Sistema realiza rollback da transação.
- Sistema lança exceção para classe que solicitou operação.
- Fim do fluxo alternativo.

Regras de negócio

(RN1) O sistema deve suportar as três operações básicas de escrita:

a. Inserção
b. Remoção
c. Atualização

Deve ser iniciada uma transação - se não existir nenhuma transação ativa - antes da execução de cada operação. A operação deve ser capaz de participar de transações já iniciadas.

(RN2) A buscas pré-definidas devem ser do seguinte tipo:

a. Busca por id
b. Busca de todos os registros
c. Busca paginada de todos os registros
d. Busca genérica

(RN3) O retorno deve ser genérico, permitindo que as subclasses possam definir o tipo da classe em tempo de execução. Para retorno de um objeto o retorno deve ser compatível com o tipo da classe. Para os casos onde deve ser retornada uma listagem, a mesma deve ser do tipo de lista e deve ser genérica compatível com o tipo da classe.

(RN4) Registrar o usuário que realizou a operação, a operação, o identificador da sessão, o IP de origem, data e hora da operação.

Exportação de Dados – Caso de Uso

Nome: Exportação de Dados.

Descrição: Exportar os dados apresentado na pesquisa.

Atores: Sistema e Usuário.

Pré-condição: Nenhuma.

Pós-condição: Apresentação do relatório com as informações.

Fluxo Principal

- Usuário realiza uma pesquisa na tela de filtro.
- Usuário seleciona a opção exportar.
- Sistema recupera as mesmas informações do filtro.
- Sistema gera relatório com as informações.
- Sistema apresenta relatório para o usuário.
- Fim do fluxo principal.

Internacionalização – Caso de Uso

Nome: Internacionalização.

Descrição: Permitir que o sistema possa utilizar diferentes idiomas.

Atores: Sistema.

Pré-condição: Nenhuma.

Pós-condição: Sistema utiliza o idioma definido para exibir mensagens, fazer cálculos, formatar números, etc.

Fluxo Principal

- Sistema recebe solicitação para retornar uma mensagem.
- Sistema verifica o idioma definido conforme RN1.
- Sistema busca a mensagem.
- Sistema retorna mensagem.
- Fim do fluxo principal.

Regras de negócio

(RN1) Se nenhum idioma for definido o padrão utilizado será o português brasileiro (PT_BR).

Customização de idioma – Caso de Uso

Nome: Customização de idioma.

Descrição: Permitir que o sistema possa ser customizado para outro idioma.

Atores: Sistema e Usuário.

Pré-condição: Nenhuma.

Pós-condição: Sistema utiliza idioma diferente do padrão.

Fluxo Principal

- Usuário define a linguagem que será utilizada ou as mensagens que deseja sobrescrever do idioma padrão.
- Sistema carrega para o contexto do sistema as mensagens de acordo com o idioma definido.
- Fim do fluxo principal.

Fluxo Alternativo (Sistema não encontra mensagens com o idioma definido):

- Sistema carrega para o contexto da aplicação as mensagens definidas de acordo com o padrão.
- Fim do fluxo alternativo.

Escolha de idioma – Caso de Uso

Nome: Escolha de idioma.

Descrição: Permitir que o usuário possa escolher o idioma desejado.

Atores: Sistema e Usuário.

Pré-condição: Nenhuma.

Pós-condição: Sistema utiliza o idioma definido pelo usuário.

Fluxo Principal

- Sistema apresenta lista de idiomas disponíveis.
- Usuário seleciona o idioma desejado.
- Sistema utiliza o idioma definido pelo usuário.
- Fim do fluxo principal.

Monitoramento da Aplicação – Caso de Uso

Nome: Monitoramento da Aplicação.

Descrição: Permitir o monitoramento dos usuários autenticados no sistema em tempo real.

Atores: Sistema e Usuário.

Pré-condição: Nenhuma.

Pós-condição: Mostrar as informações encontradas no momento da solicitação.

Fluxo Principal

- Usuário solicita a funcionalidade de monitoramento do sistema.
- Sistema apresenta listagem paginada com os usuários autenticados no sistema conforme RN1.

Regras de negócio

(RN1) O sistema deve apresentar o resultado paginado e ordenado pelo usuário que está a mais tempo autenticado no sistema.

Visualização de Acesso – Caso de Uso

Nome: Visualização de Acesso.

Descrição: Visualizar as informações referentes ao acesso.

Atores: Sistema e Usuário.

Pré-condição: Estar autenticado no sistema.

Pós-condição: Exibir informações encontradas.

Fluxo Principal

- Usuário preenche os campos de filtro de acordo com a RN1.
- Usuário seleciona a opção de pesquisa.

- Sistema busca o histórico de acesso de acordo com a RN2.
- Sistema mostra os resultados de acordo com RN3.
- Fim do fluxo principal.

Fluxo Alternativo (Acesso não encontrado):

- Sistema exibe mensagem informando que nenhum acesso foi encontrado.
- Fim do fluxo alternativo

Regras de negócio

(RN1) Os campos de filtro são: *username*, identificador da sessão, IP de origem, intervalo de data e hora do acesso e status do *login* (Sucesso ou Falha). O campo status *login* é de preenchimento obrigatório, os demais são opcionais.

(RN2) O sistema deve realizar a busca exata ou que contenha a informação fornecida, se o campo estiver em branco deve ser desconsiderado da pesquisa.

(RN3) O sistema deve apresentar o resultado paginado e ordenado por *username*.

Visualização de Operações – Caso de Uso

Nome: Visualização de Operações.

Descrição: Visualizar as operações realizadas pelos usuários.

Atores: Sistema e Usuário.

Pré-condição: Estar autenticado no sistema.

Pós-condição: Exibir informações encontradas.

Fluxo Principal

- Usuário preenche os campos de filtro de acordo com a RN1.
- Usuário seleciona a opção de pesquisa.
- Sistema busca o histórico de operações realizadas de acordo com a RN2.
- Sistema mostra os resultados de acordo com RN3.
- Fim do fluxo principal.

Fluxo Alternativo (Operação não encontrada):

- Sistema exibe mensagem informando que nenhuma operação foi encontrada.
- Fim do fluxo alternativo.

Regras de negócio

(RN1) Os campos de filtro são: username, identificador da sessão, IP de origem, intervalo de data e hora de execução da operação.

(RN2) O sistema deve realizar a busca exata ou que contenha a informação fornecida, se o campo estiver em branco deve ser desconsiderado da pesquisa.

(RN3) O sistema deve apresentar o resultado paginado e ordenado por username.

Gerenciamento de Comunicação – Caso de Uso

Nome: Gerenciamento de Comunicação.

Descrição: Realizar interação com o usuário através do envio de mensagens.

Atores: Usuário.

Pré-condição: Nenhuma.

Pós-condição:: Envio da mensagem.

Fluxo Principal

- Sistema solicita envio da mensagem.
- Sistema envia mensagem (PE1).
- Fim do fluxo principal.

Pontos de Extensão (PE)

(PE1) Caso de Uso: Envio de e-mail.

Envio de e-mail – Caso de Uso

Nome: Envio de e-mail.

Descrição: Realizar o envio de e-mail para uma ou mais pessoas.

Atores: Sistema e Usuário.

Pré-condição: Nenhuma

Pós-condição: Envio do e-mail.

Fluxo Principal

- Usuário ou sistema solicita envio de e-mail.
- Sistema recebe solicitação de envio de e-mail de acordo com RN1.
- Sistema recupera as configurações do serviço de e-mail.
- Sistema envia mensagem de e-mail.
- Fim do fluxo principal.

Regras de negócio

(RN1) A solicitação deve receber como parâmetro o assunto, a lista de destinatários, o remetente e a lista de anexos que devem ser enviados.

Envio de SMS – Caso de Uso

Nome: Envio de SMS.

Descrição: Realizar o envio de SMS.

Atores: Sistema e Usuário.

Pré-condição: Nenhuma.

Pós-condição: Envio do SMS.

Fluxo Principal

- Usuário solicita o envio de SMS.
- Sistema recebe solicitação de envio de SMS de acordo com RN1.
- Produto recupera as configurações do serviço de SMS.
- Produto envia mensagem.
- Fim do fluxo principal.

Regras de negócio

(RN1) A solicitação deve receber como parâmetro o número de destino e o texto da mensagem.

Gerenciamento de Usuário – Caso de Uso

Nome: Gerenciamento de Usuário.

Descrição: Permitir a listagem, inserção, atualização e inativação de usuários.

Atores: Sistema e Usuário.

Pré-condição: Estar autenticado no sistema.

Pós-condição: Informação do usuário atualizada na base de dados.

Fluxo Principal

- Usuário preenche os campos de filtro de acordo com a RN1.
- Usuário seleciona a opção de pesquisa.
- Sistema realiza a busca de usuário utilizando os critérios estabelecidos.

- Sistema mostra o resultado de acordo com a RN2.
- Fim do fluxo principal.

Fluxo Alternativo (Inserção):

- Usuário seleciona a opção inserir.
- Sistema direciona o usuário para o cadastro.
- Usuário preenche as informações.
- Usuário seleciona a opção inserir.
- Sistema valida os campos obrigatórios de acordo com RN3.
- Sistema valida os campos que devem ser únicos de acordo com o RN4.
- Sistema persiste as informações na base de dados.
- Sistema exibe mensagem de inserção realizada com sucesso.
- Fim do fluxo alternativo.

Fluxo Alternativo (Atualização):

- Usuário seleciona a opção atualizar.
- Sistema direciona o usuário para o cadastro de acordo com a RN5.
- Usuário atualiza as informações.
- Usuário seleciona a opção atualizar.
- Sistema valida os campos obrigatórios de acordo com RN3.
- Sistema valida os campos que devem ser únicos de acordo com o RN4.
- Sistema persiste as informações na base de dados.
- Sistema exibe mensagem de atualização realizada com sucesso.
- Fim do fluxo alternativo.

Fluxo Alternativo (Inativação):

- Usuário seleciona a opção inativar.
- Sistema solicita confirmação.
- Usuário confirma que deseja inativar.
- Sistema inativa o registro do usuário na base de dados.
- Sistema exibe mensagem de inativação realizada com sucesso.
- Fim do fluxo alternativo.

Fluxo Alternativo (Falha na validação):

- Sistema exibe mensagem informando os erros de validação que ocorreram.
- Fim do fluxo alternativo.

Fluxo Alternativo (Falha na verificação de unicidade):

- Sistema exibe mensagem informando qual informação já se encontra na base de dados.
- Fim do fluxo alternativo.

Fluxo Alternativo (Usuário não encontrado):

- Sistema exibe mensagem informando que nenhum usuário foi encontrado.
- Fim do fluxo alternativo

Regras de negócio

(RN1) Os campos de filtro são username e nome. O preenchimento dos campos é opcional.

(RN2) O resultado deve ser paginado e ordenado por nome.

(RN3) Os campos obrigatórios são: username, nome, CPF e e-mail.

(RN4) Os campos que devem ser únicos na base de dados são: username, CPF e e-mail.

(RN5) As informações referentes ao usuário devem ser preenchidas automaticamente de acordo com as informações encontradas na base de dados.

Gerenciamento de Perfil do Usuário – Caso de Uso

Nome: Gerenciamento de Perfil de Usuário.

Descrição: Permitir a adição e remoção de perfis do usuário.

Atores: Sistema e Usuário.

Pré-condição: Estar autenticado no sistema.

Pós-condição: Informação do perfil do usuário atualizada na base de dados.

Fluxo Principal

- Usuário preenche os campos de filtro de acordo com a RN1.
- Usuário seleciona a opção de pesquisa.
- Sistema realiza a busca de usuário utilizando os critérios estabelecidos.
- Sistema mostra o resultado de acordo com a RN2

- Usuário seleciona a opção atualizar perfil.
- Sistema direciona o usuário para o cadastro de acordo com a RN3.
- Usuário seleciona os perfis do usuário.
- Usuário seleciona a opção atualizar.
- Sistema exibe mensagem informando que o perfil do usuário foi atualizado com sucesso.
- Fim do fluxo principal.

Fluxo Alternativo (Usuário não encontrado):

- Sistema exibe mensagem informando que nenhum usuário foi encontrado.
- Fim do fluxo alternativo

Regras de negócio

(RN1) Os campos de filtro são username e nome. É necessário que pelo menos um dos campos seja preenchido para realização do filtro.

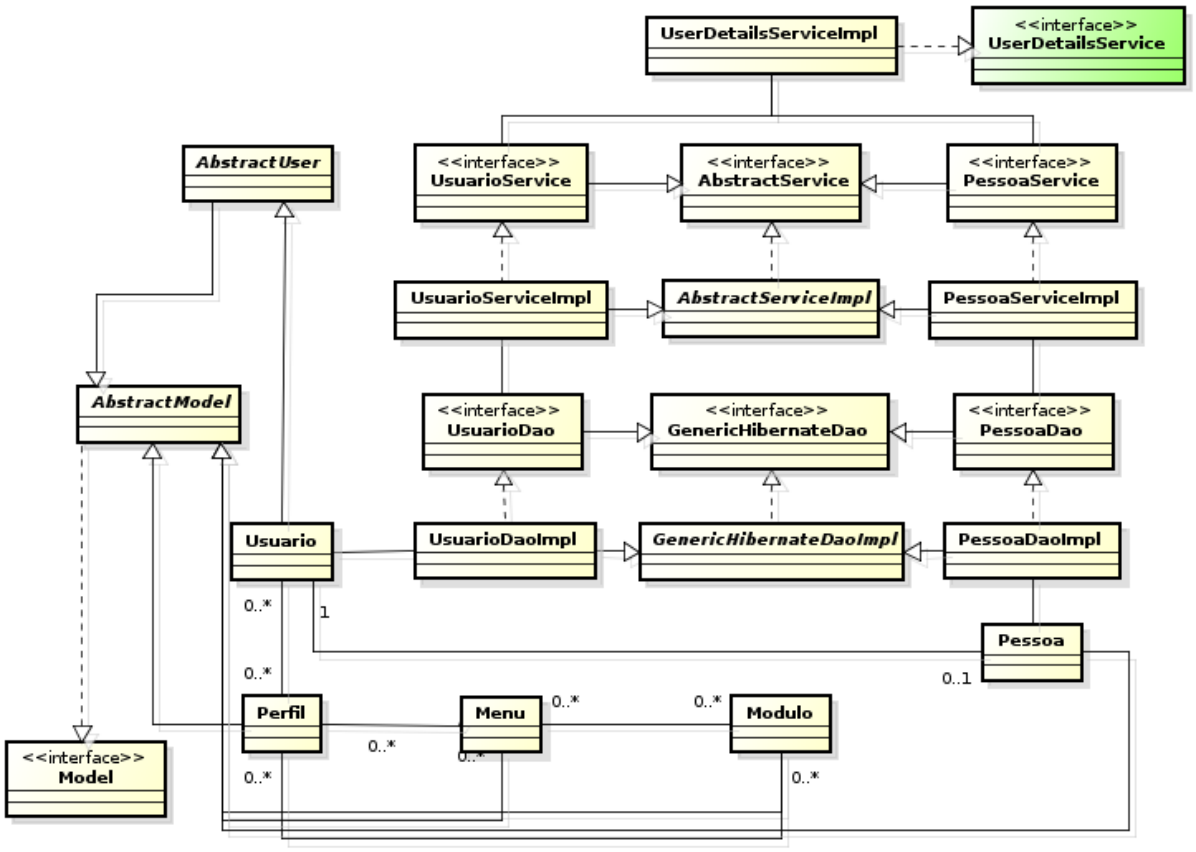
(RN2) O resultado deve ser paginado e ordenado por nome.

(RN3) As informações referentes ao usuário e o perfil devem ser preenchidas automaticamente de acordo com as informações encontradas na base de dados.

APÊNDICE B - Diagramas de Classes

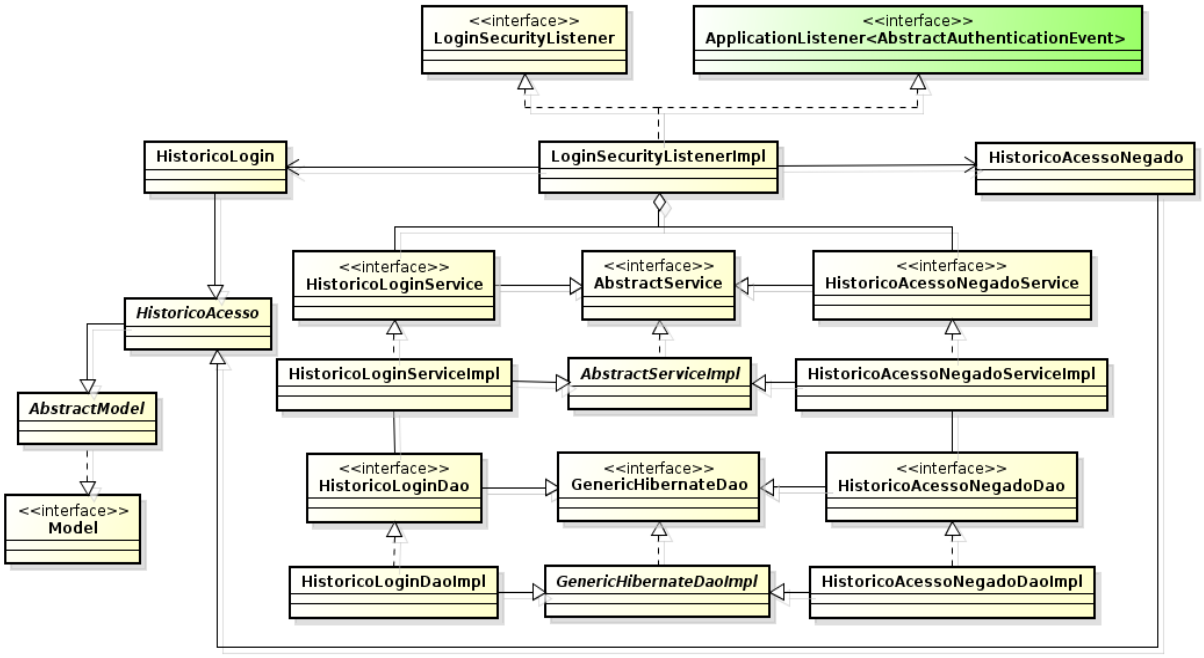
Abaixo será apresentado o diagrama de classes modelado para cada *feature*.

Figura 13 - Diagrama de classes da feature Controle de Autenticação



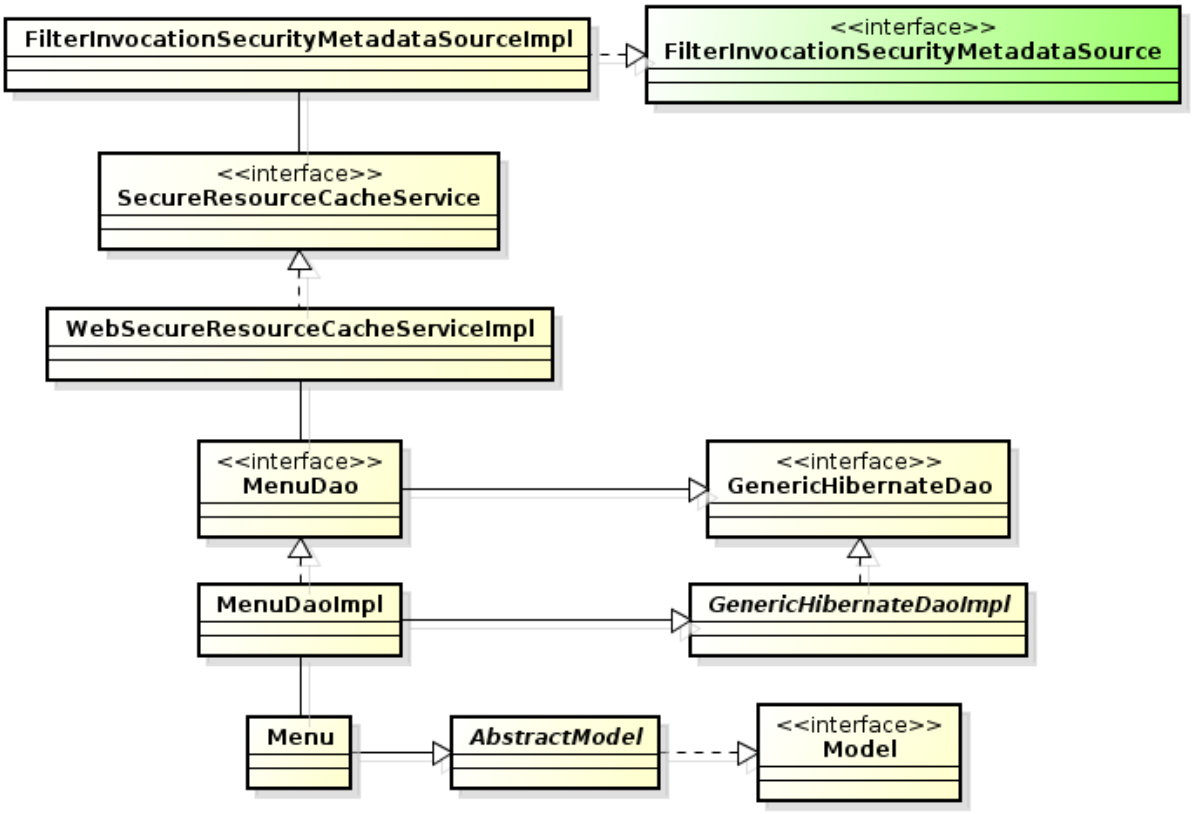
Fonte: Elaborado pelo autor

Figura 14 - Diagrama de classes da feature Gerenciamento de Acesso



Fonte: Elaborado pelo autor

Figura 15 - Diagrama de classes da feature Controle de Autorização

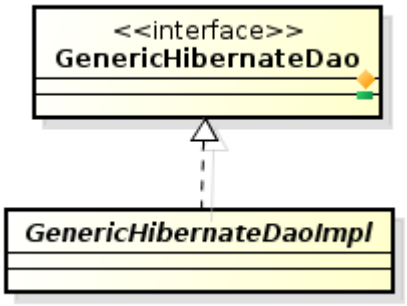


Fonte: Elaborado pelo autor

Figura 16 - Diagrama de classes da feature Gerenciamento de Dados

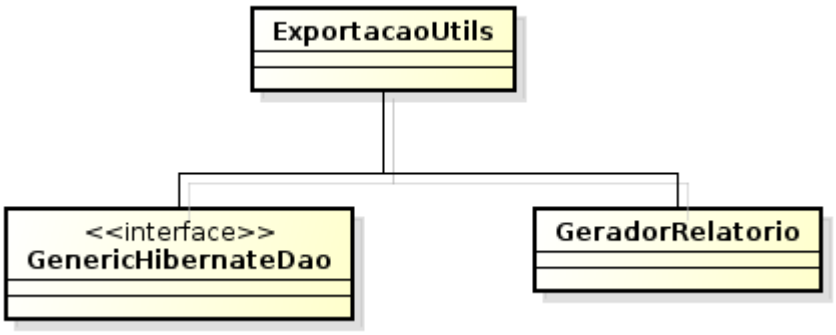
Fonte: Elaborado pelo autor

Figura 17 - Diagrama de classes da feature Peristência de Dados



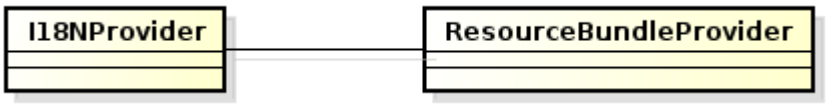
Fonte: Elaborado pelo autor

Figura 18 - Diagrama de classes da feature Exportação de Dados



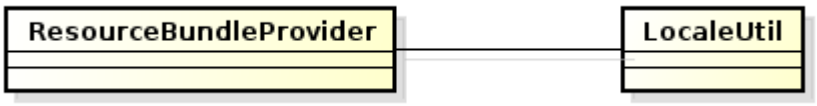
Fonte: Elaborado pelo autor

Figura 19 - Diagrama de classes da feature Interncionalização



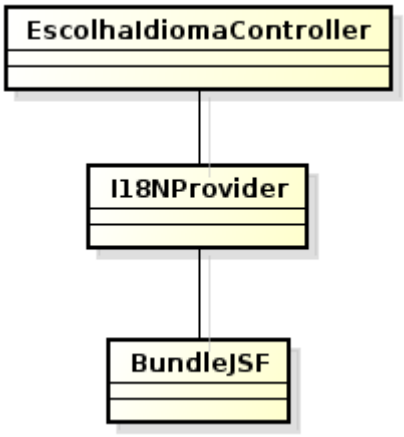
Fonte: Elaborado pelo autor

Figura 20 - Diagrama de classes da feature Customização de Idioma



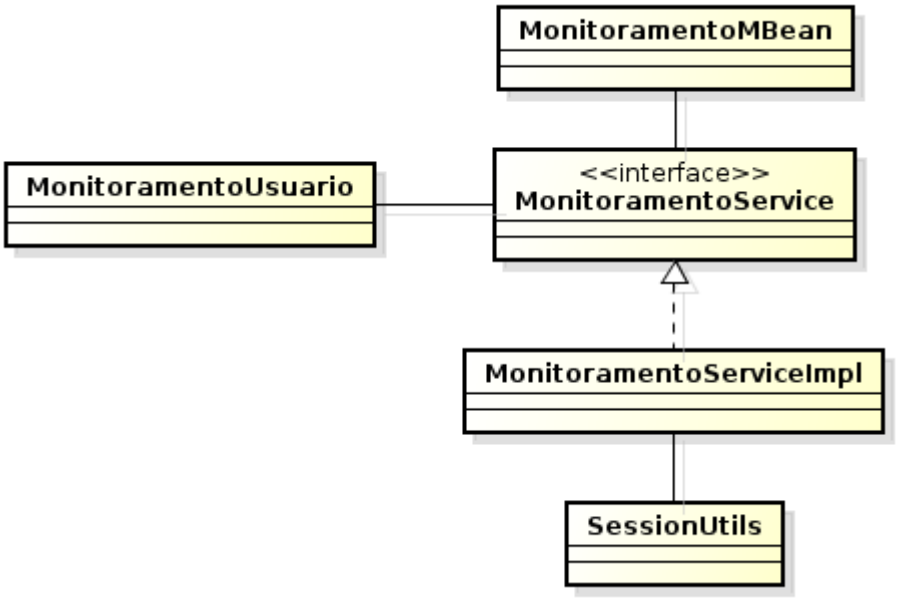
Fonte: Elaborado pelo autor

Figura 21 - Diagrama de classes da feature Escolha de Idioma



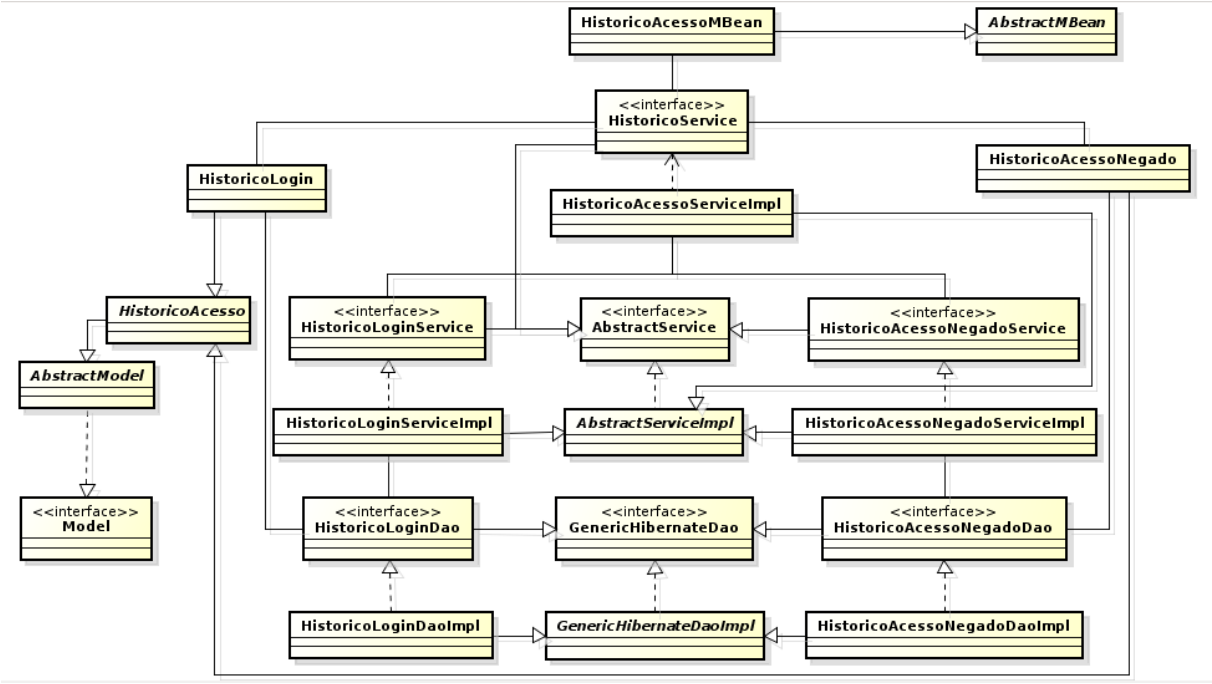
Fonte: Elaborado pelo autor

Figura 22 - Diagrama de classes da feature Monitoramento do Sistema



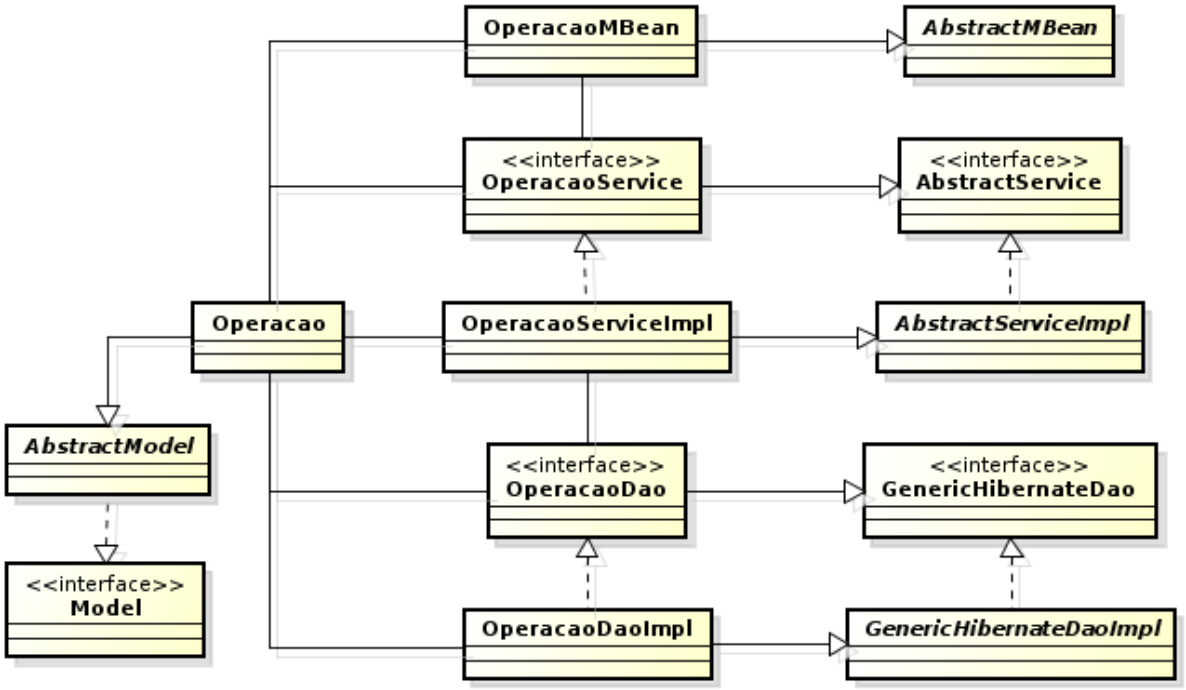
Fonte: Elaborado pelo autor

Figura 23 - Diagrama de classes da feature Visualização de Acessos



Fonte: Elaborado pelo autor

Figura 24 - Diagrama de classes da feature Visualização de Operações

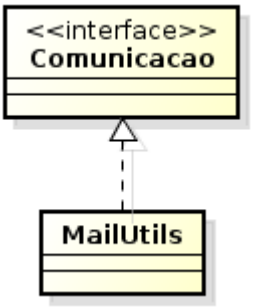


Fonte: Elaborado pelo autor

Figura 25 - Diagrama de classes da feature Monitoramento do Sistema

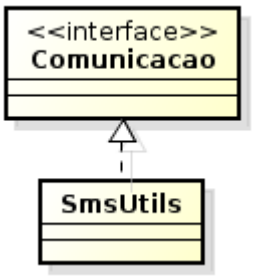
Fonte: Elaborado pelo autor

Figura 26 - Diagrama de classes da feature Envio de E-mail



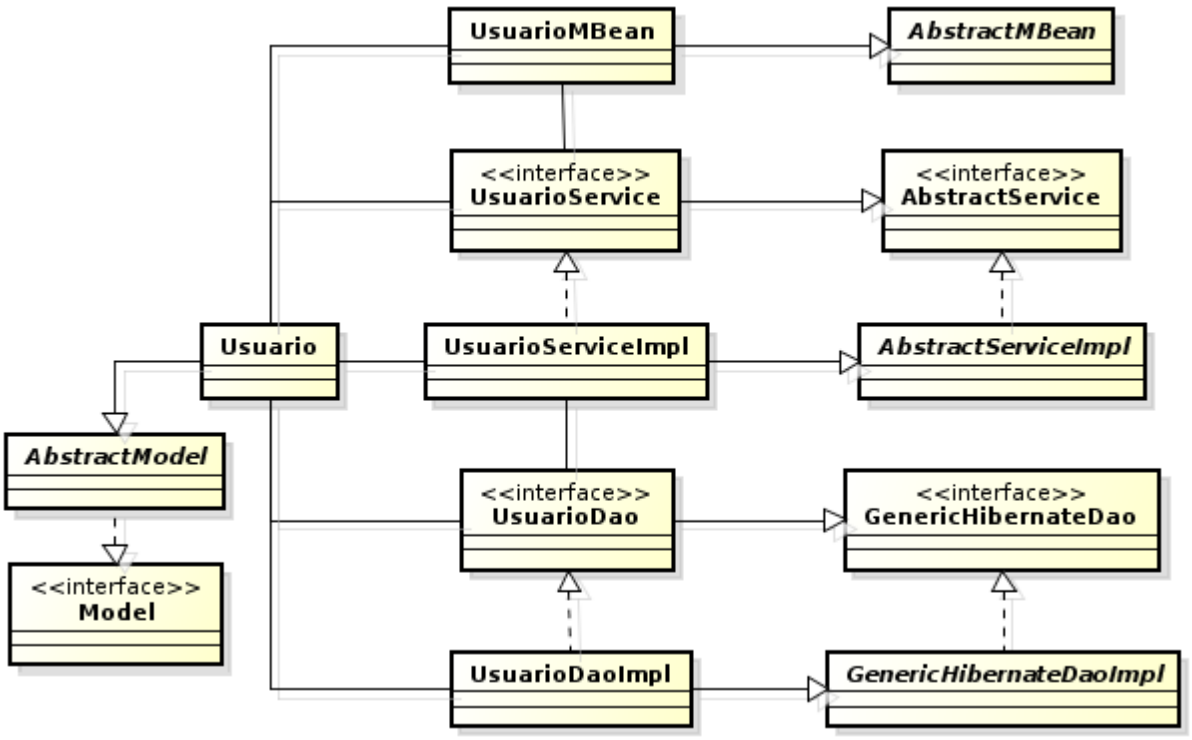
Fonte: Elaborado pelo autor

Figura 27 - Diagrama de classes da feature Envio de SMS



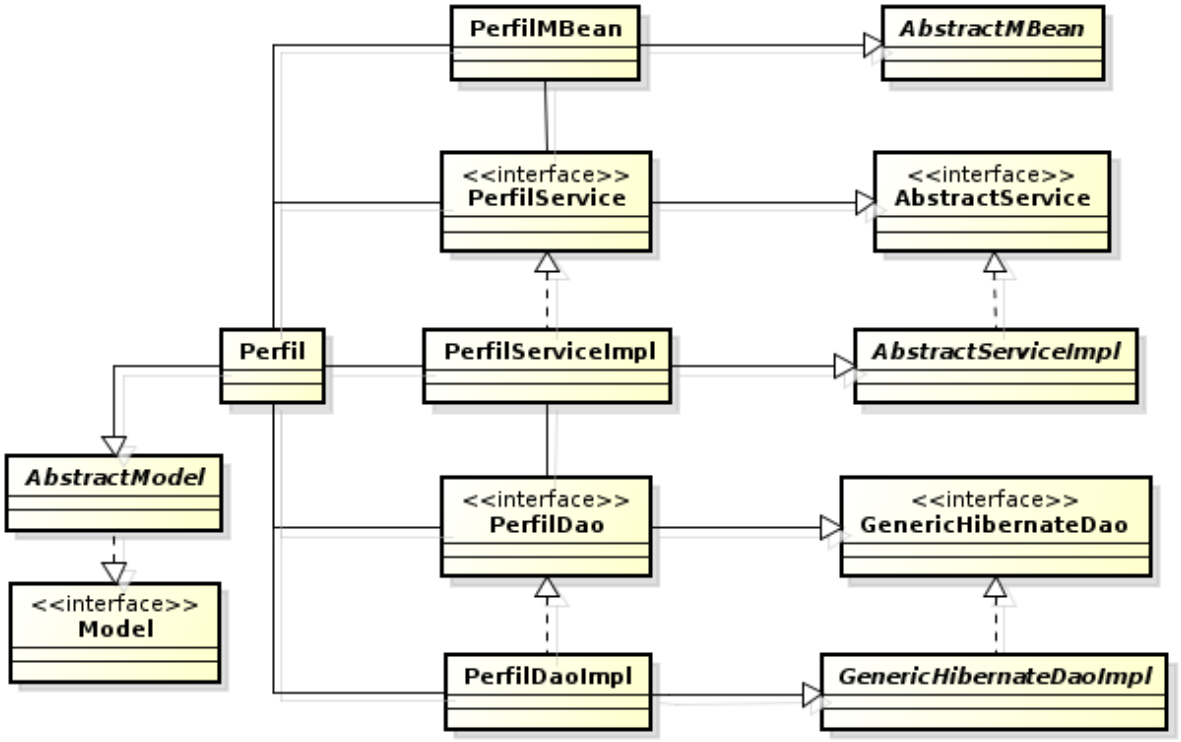
Fonte: Elaborado pelo autor

Figura 28 - Diagrama de classes da feature Gerenciamento de Usuário



Fonte: Elaborado pelo autor

Figura 29 - Diagrama de classes da feature Controle de Perfil do Usuário



Fonte: Elaborado pelo autor