

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
ESCOLA DE INFORMÁTICA APLICADA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

APLICAÇÃO DE PROCESSO DE DESENVOLVIMENTO COM MODEL DRIVEN
ARCHITECTURE

BRUNO TARANTA ARRUDA
DIEGO FERNANDES
RAPHAEL CRERIE DA SILVA CASTRO

Prof. Márcio de Oliveira Barros

RIO DE JANEIRO

2006

APLICAÇÃO DE PROCESSO DE DESENVOLVIMENTO COM MODEL DRIVEN
ARCHITECTURE

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

BRUNO TARANTA ARRUDA
DIEGO FERNANDES
RAPHAEL CRERIE DA SILVA CASTRO

Orientador: Prof. Márcio de Oliveira Barros

RIO DE JANEIRO

2006

Fonte: ARRUDA, Bruno Taranta. FERNANDES, Diego. CASTRO, Raphael Cerie da Silva. *Aplicação de Processo de Desenvolvimento com Model Driven Architecture*. Rio de Janeiro, 2006. Monografia (Bacharelado em Sistemas de Informação) – Escola de Informática Aplicada. Universidade Federal do Estado do Rio de Janeiro. Rio de Janeiro. 2006.

APLICAÇÃO DE PROCESSO DE DESENVOLVIMENTO COM MODEL DRIVEN
ARCHITECTURE

Aprovado em ____/_____/____

BANCA EXAMINADORA

Nome e Assinatura do (a) professor (a) orientador (a)

Names e Assinaturas dos demais Membros da Banca

O(s) autor(es) deste Projeto autoriza(m) a ESCOLA DE INFORMÁTICA
APLICADA da UNIRIO a divulgá-lo, no todo ou em parte, resguardados os direitos
autorais conforme legislação vigente.

Rio de Janeiro, ____ de ____ de ____.

Nome(s) e Assinatura(s) do(s) aluno(s)

A minha família pelo amor, a meus mestres pelo conhecimento, a meus amigos pela compreensão e a Deus por dar sentido a tudo isto.

B.A.

Agradeço a minha família pela paciência e suporte, aos amigos por entender os finais de semana perdidos e ao professor Márcio pela ajuda a terminar esse trabalho.

D.F.

Agradeço especialmente a Daniele pelo apoio e paciência nos finais de semana e nas noites de trabalho, ao Pai do Bruno por aturar a gente trabalhando em grupo em sua casa ao profº Márcio que nos apoiou e tornou possível a finalização deste projeto.

R.C.

SUMÁRIO

| | |
|---|-----------|
| Resumo ----- | 7 |
| 1 – Capítulo Primeiro – Introdução ----- | 8 |
| 1.1 – <u>Motivação</u> ----- | 8 |
| 1.2 – <u>Objetivos</u> ----- | 8 |
| 2 – Capítulo Segundo – Processo, Tecnologia e Padrões Utilizados ----- | 10 |
| 2.1 – <u>Aplicação do Processo de Desenvolvimento</u> ----- | 11 |
| 2.1.1 – <u>Concepção</u> ----- | 12 |
| 2.1.1.a – <u>Levantamento de Requisitos</u> ----- | 12 |
| 2.1.1.b – <u>Glossário</u> ----- | 13 |
| 2.1.1.c – <u>Visão</u> ----- | 13 |
| 2.1.1.d – <u>Pasta de Desenvolvimento</u> ----- | 13 |
| 2.1.1.e – <u>Casos de Uso</u> ----- | 13 |
| 2.1.1.f – <u>Modelo de Domínio</u> ----- | 14 |
| 2.1.2 – <u>Elaboração</u> ----- | 14 |
| 2.1.2.a – <u>Documento de Arquitetura</u> ----- | 14 |
| 2.1.2.b – <u>Modelo de Projeto</u> ----- | 14 |
| 2.1.3 – <u>Construção</u> ----- | 14 |
| 2.1.3.a – <u>Codificação</u> ----- | 15 |
| 2.1.3.b – <u>Testes</u> ----- | 15 |
| 2.2 – <u>Arquitetura e Model Driven Architecture – MDA</u> ----- | 15 |
| 2.3 – <u>O Sistema e sua estrutura</u> ----- | 16 |
| 2.3.1 – <u>Arquitetura em camadas</u> ----- | 17 |
| 2.4 – <u>Padrões Utilizados</u> ----- | 19 |
| 2.4.1 – <u>Data Transfer Object</u> ----- | 19 |
| 2.4.2 – <u>Service Locator</u> ----- | 20 |

| | |
|---|-----------|
| 2.4.3 – Data Access Object ----- | 21 |
| 2.4.4 – Application Service ----- | 21 |
| 2.4.5 – Business Objects ----- | 22 |
| 2.4.6 – Session Façade ----- | 22 |
| 2.4.7 – Business Delegate ----- | 23 |
| 3 – Capítulo Terceiro – Ferramentas Utilizadas e o Sistema ----- | 26 |
| 3.1 – Sistema de Compras Web ----- | 26 |
| 3.2 – Criando a Aplicação ----- | 33 |
| 4 – Capítulo Quarto – Conclusão ----- | 37 |
| Apêndice A – Mini-mundo ----- | 38 |
| Apêndice B – Glossário ----- | 39 |
| Apêndice C – Visão ----- | 40 |
| Apêndice D – Casos de Uso ----- | 41 |
| Apêndice E – Modelo de Domínio ----- | 48 |
| Apêndice F – Modelo de Projeto ----- | 49 |
| Bibliografia ----- | 56 |

RESUMO

Neste presente trabalho foi demonstrada a execução de todo o processo de criação de um software, partindo do levantamento de requisitos e atingindo a geração do código final da aplicação, utilizando para isso, uma série de padrões de projeto e tecnologias recentes para o desenvolvimento de aplicações corporativas. Baseados no Processo Unificado [JBR99], foi criado um processo de desenvolvimento adaptado às necessidades e ao foco do projeto, que é a criação de um carrinho de compras web. Utilizando a *Model Driven Architecture* (MDA) [OMG06], através da ferramenta AndromDA, foi obtida a geração de grande parte do código necessário para a infraestrutura da aplicação a partir de modelos independentes de plataforma. A base tecnológica utilizada foi a plataforma *JAVA 2 Enterprise Edition* (J2EE) [J2EEP06] e o *framework* Struts [Struts05].

PALAVRAS-CHAVE: Processo Unificado (PU), Model Driven Architecture (MDA), J2EE, Padrões de Projeto.

Capítulo Primeiro

INTRODUÇÃO

Ao longo do curso de Sistemas de Informação, foram estudados os passos necessários para a construção de *software*. A proposta para este trabalho foi o alinhamento da execução de um processo de desenvolvimento aos conhecimentos adquiridos durante a graduação, focando também na tecnologia utilizada para o desenvolvimento e na arquitetura do sistema, de forma a criarmos uma aplicação mais portátil, escalonável, adaptável e com maior facilidade de manutenção.

Para alcançar estes objetivos, foram utilizadas modernas tecnologias e técnicas, tendo como base a construção de sistemas através de modelos independentes de plataforma, conforme rege a *Model Driven Architecture* (MDA) [OMG06]. Para seguir com este projeto, foi resolvido implementar um sistema de carrinho de compras, chamado Sistema Compras Web, que tem como objetivo proporcionar recursos para uma loja vender seus produtos via *Web* de forma simplificada.

1.1. MOTIVAÇÃO

Fazer uso da plataforma *JAVA 2 Enterprise Edition* (J2EE) [J2EEP06], juntamente com diversos padrões de projeto e um processo de desenvolvimento (derivado do Processo Unificado [JBR99]). Criar uma arquitetura portátil e adaptável, que viabilize um software de fácil manutenção, customização e implantação.

1.2. OBJETIVOS

Criação de um carrinho de compras com arquitetura bem definida utilizando a *Model Driven Architecture* (MDA) [OMG06] como base, juntamente com a adaptação e

aplicação de um Processo de Desenvolvimento conforme as características do projeto, gerando assim, uma documentação confiável e produtiva.

A aplicação será chamada de Sistemas de Compras Web, e é dividida em duas interfaces, uma para o administrador e outra para o cliente. A interface desenvolvida para o administrador é responsável pelo cadastro de produtos, marcas e grupos, além da localização, edição e remoção dos mesmos. Já a interface do cliente é responsável pela localização dos produtos, adição destes no carrinho, remoção dos produtos incluídos no carrinho e fechamento do pedido.

Capítulo Segundo

PROCESSO, TECNOLOGIA E PADRÕES UTILIZADOS

Um processo de desenvolvimento de software é um conjunto de regras e padrões que deve guiar as etapas necessárias para a criação e manutenção de um software em uma ordem cronológica.

Quando bem definido, um processo de desenvolvimento auxilia não só a gerência mas também a construção do sistema. Com relação à gerência, o processo permite uma definição mais transparente dos papéis e das responsabilidades dos elementos da equipe. Com relação à construção, o processo estabelece um caminho a ser seguido para se atingir o objetivo final, gerando maior confiança e permitindo uma melhoria gradativa na eficiência da equipe.

Não existe um processo único ideal para todos os tipos de projeto, visto que cada projeto possui suas limitações e necessidades específicas. Logo, o processo deve suprir estas peculiaridades, adaptando-se a elas. O processo deve ser passível de mudanças e melhorias, porém deve se basear em conhecimentos e experiências adquiridos anteriormente.

O Processo Unificado [JBR99] surgiu como um processo popular para o desenvolvimento de software visando à construção de sistemas orientados a objetos. O processo se baseia na utilização da *Unified Modeling Language* (UML), é centrado na arquitetura, iterativo e incremental, e fora refinado para dar origem ao famoso *Rational Unified Process* (RUP) [Kruchten00].

Dentre as práticas que o Processo Unificado [JBR99] promove, a que mais se destaca é o desenvolvimento iterativo, do qual cada ciclo possui 4 fases: Concepção, Elaboração, Construção e Transição, todas essas divididas em iterações ou etapas.

gerados, o que pode ser visto nos passos 2 a 6 da fase. Na última fase – Transição – o projeto é implantado.

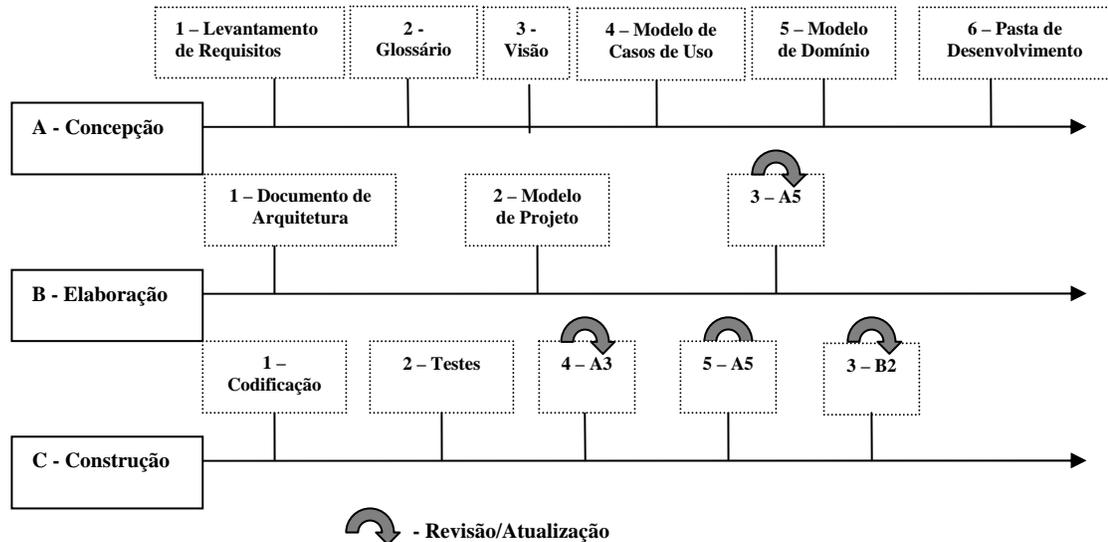


Figura 2.2 – Modelo representativo do Processo Adaptado

A figura 2.2 ilustra o processo adaptado para o projeto descrito neste documento, sendo um refinamento do Processo Unificado. Este tipo de refinamento ou adaptação é importante para que não haja retrabalho ou tarefas que não trarão benefícios ao projeto. Abaixo, são descritas as fases e as etapas que foram adotadas neste processo.

2.1.1. CONCEPÇÃO

A fase da Concepção tem como objetivo o entendimento do problema para que possa ser avaliada a viabilidade de sua construção e para que forme uma estimativa de tempo, mesmo que ainda não confiável, mas que possa ser considerada como um indicador ou uma ordem de grandeza do esforço necessário para o desenvolvimento.

2.1.1.a. Levantamento de Requisitos

O objetivo desta etapa é a compreensão do problema através de seus conceitos básicos. Realizando pesquisas em diversos sítios de comércio eletrônico, foram

definidos os principais requisitos e uma visão geral do domínio abordado pelo projeto. O mini-mundo apresentado no Apêndice A foi produzido como resultado desta etapa.

2.1.1.b. Glossário

Esta etapa tem o objetivo de padronizar nomenclaturas e conceitos relacionados a atividade fim do sistema, para que não haja interpretações ambíguas destes termos. O glossário apresentado no Apêndice B foi produzido como resultado desta etapa.

2.1.1.c. Visão

A finalidade desta etapa é coletar, analisar, definir e documentar uma visão de alto nível das funcionalidades necessárias ao sistema. Ela se concentra nas funcionalidades requisitadas pelos envolvidos no projeto e nas razões que levam a essas necessidades. O documento de visão apresentado no Apêndice C foi produzido como resultado desta etapa.

2.1.1.d. Pasta de Desenvolvimento

Consiste na criação de um repositório estruturado para armazenar a documentação do projeto de forma organizada, oferecendo apoio para a gerência de configuração do projeto. A pasta de desenvolvimento do projeto foi produzida como resultado desta etapa e utilizada para organizar os arquivos que o compõem.

2.1.1.e. Casos de Uso

A criação de casos de uso foi adotada para especificar a interação entre os clientes e o sistema. Servem para expor e documentar detalhes sobre cada operação que é relevante para o projeto. Os Casos de Uso apresentados no Apêndice D foram produzidos como resultado desta etapa.

2.1.1.f. Modelo de Domínio

Apesar de aparecer no Processo Unificado apenas na fase de Construção, foi decidido embutir esta etapa na fase de Concepção para que fique disponível, desde o início do projeto, um diagrama simples que forneça uma visão geral dos elementos envolvidos no sistema. Este modelo foi produzido através do cruzamento de todas as informações obtidas nas etapas anteriores. O Modelo de Domínio apresentado no Apêndice E foi produzido como resultado desta etapa.

2.1.2. ELABORAÇÃO

Nesta fase se executa a maior parte do levantamento de requisitos e é criada a arquitetura do software. Após a execução destas etapas, é possível gerar estimativas referentes a custo e prazo de execução do projeto com maior precisão.

2.1.2.a. Documento de Arquitetura

Esta etapa tem o objetivo de relacionar as soluções para os problemas identificados com as tecnologias utilizadas, fornecendo uma estrutura a ser respeitada no modelo de projeto. A sessão 2.3 deste trabalho é o documento de arquitetura do sistema.

2.1.2.b. Modelo de Projeto

Etapa onde são especificadas as interações entre os módulos do sistema, juntamente com suas classes realizadoras e as interações entre elas. A partir deste modelo, passamos a trabalhar com o AndroMDA [AndroMDA06] para a construção de nossa aplicação. O Modelo de Projeto apresentado no Apêndice F foi produzido como resultado desta etapa.

2.1.3. CONSTRUÇÃO

Na Construção o sistema é codificado, os testes são estruturados e são executadas algumas revisões em documentos anteriormente criados, de acordo com as necessidades ou problemas encontrados.

2.1.3.a. Codificação

Nesta etapa o sistema é codificado de acordo com o projeto gerado anteriormente. Utilizando o AndroMDA [AndroMDA06], é gerado o código relacionado a infra-estrutura da aplicação. O código necessário para a realização das regras de negócio é produzido de forma convencional e integrado ao restante da aplicação. O código fonte foi produzido como resultado desta etapa.

2.1.3.b. Testes

Os testes unitários e de integração são executados durante esta fase, com o objetivo de verificar a coerência do que foi documentado com o que foi desenvolvido. A identificação de problemas durante os testes implica na correção do código e revisão da documentação anteriormente gerada. Os testes unitários foram produzidos como resultado desta etapa.

2.2. ARQUITETURA E MODEL DRIVEN ARCHITECTURE – MDA

No cenário da Tecnologia da Informação atual, alguns problemas se tornaram recorrentes e cada vez mais graves, dentre eles pode-se destacar: implementações de sistemas distantes das reais necessidades dos clientes; alta rotatividade tecnológica, acarretando grandes custos de migração; e a dessincronização da documentação, dificultando muito a manutenção dos sistemas construídos e a inserção de novos integrantes nas equipes de desenvolvimento. Diante a estes problemas, a Object Management Group(OMG), criadora das especificações da UML e CORBA, propôs a Model Driven Architecture(MDA), uma abordagem de desenvolvimento de software onde tem-se o modelo como centro de trabalho. A MDA não pode ser considerada uma “bala de prata” para resolução dos problemas recorrentes da área de desenvolvimento, mas sim, um passo natural para a evolução da construção de *software* como uma disciplina de engenharia. Dentre seus principais objetivos, se destacam: independência de plataformas (interoperabilidade, reusabilidade e portabilidade); melhoria na

qualidade(foco no problema, não na solução); e ganho de produtividade e diminuição da possibilidade de “falha humana” (a partir da automação de processos); .

Dois conceitos fundamentais da MDA são:

Platform Independent Model(PIM) que representa um modelo relacionado ao problema totalmente independente de tecnologia;

Platform Specific Model(PSM) que traduz a solução do problema para uma tecnologia específica.

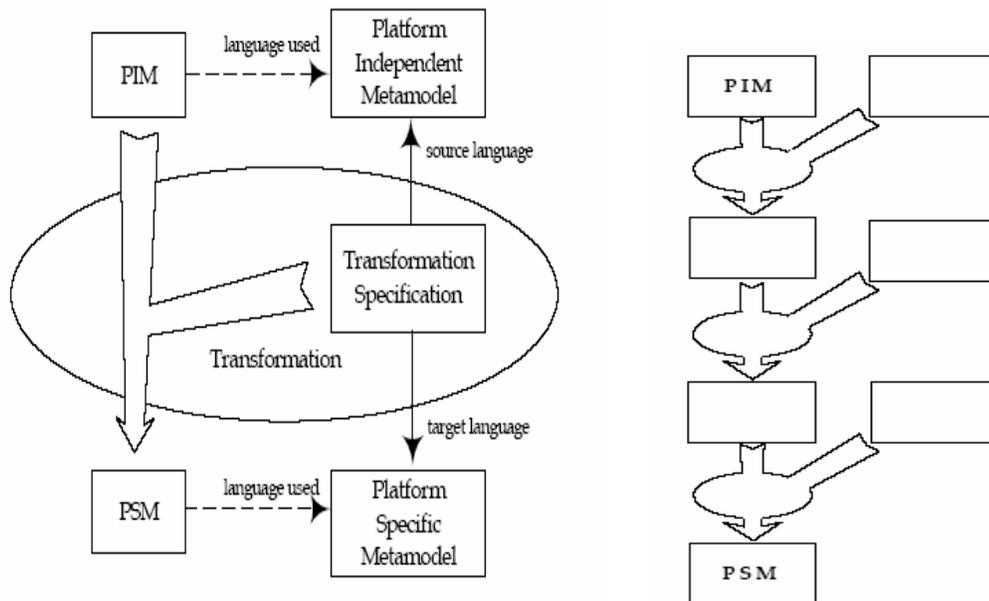


Figura 2.3 – MDA

Conforme a figura 2.3, o desenvolvedor deve criar um PIM contendo as regras de negócio específicas do domínio, submetê-lo juntamente com as regras de transformações específicas de plataforma para uma ferramenta que efetuará a transformação. Podemos

observar na figura 2.3, que a MDA fornece suporte ao desenvolvimento iterativo, pois possibilita a efetuação de diversas transformações com abstrações diferentes em um mesmo modelo.

As transformações entre os modelos são baseadas em meta-modelos definidos seguindo o *framework Meta Object Facility*(MOF), que define regras para criação de meta-modelos, por exemplo a *Unified Modeling Language*(UML) e a *Common Warehouse Metadata Interchange*(CWMI).

Ao avaliar a MDA pela primeira vez, é comum a comparação com as famosas ferramentas CASE dos anos 80. Porém, existem muitas características que apontam para o sucesso da MDA frente aos erros cometidos por estas ferramentas, como por exemplo: na MDA a arquitetura inteira do projeto pode ser gerada, no entanto, fica a cargo do desenvolvedor decidir o quanto gerar; MDA permite a sincronização com o código, principalmente porque o modelo faz parte do código, enquanto as ferramentas CASE geralmente possuem uma amarração muito ineficaz dificultando a manutenção do código gerado, além de possuírem grande complexidade e falhas na configuração de seus geradores de código.

Por utilizar o padrão *XML Metadata Interchange*(XMI), MDA não requer nenhuma ferramenta comercial específica, podendo utilizar os modelos gerados por qualquer ferramenta que implemente este padrão.

A ferramenta AndroMDA implementa a MDA de uma forma diferente da maioria das ferramentas comerciais. Enquanto muitas executam a transformação do PIM para o PSM e o tornam visualmente editáveis, o AndroMDA executa a transformação para PSM em memória, que serve de insumo para outras automatizações, como a geração de código. A ferramenta AndroMDA na sua versão 3.0 possui a limitação de não possibilitar transformações entre PSM's, funcionalidade prometida para a versão 4.0.

2.3. O SISTEMA E SUA ESTRUTURA

O Sistema de Compras Web é um sistema de *e-commerce* simples que foi desenvolvido para ilustrar a criação de um sistema seguindo todas as etapas do processo de desenvolvimento apresentado neste trabalho. A proposta desta aplicação é a realização de buscas de produtos, compras de produtos e a manutenção de produtos, grupos e marcas. Durante o processo de análise, o sistema foi subdividido em 4 módulos para melhor organização do projeto, conforme pode ser visto na figura 2.3.

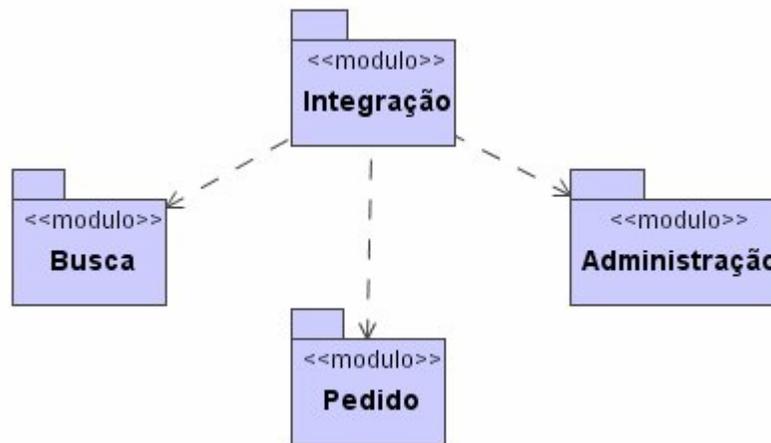


Figura 2.3 – Módulos do Sistema de Compras Web

- Módulo de Busca: tem como objetivo centralizar todos os métodos de pesquisa da aplicação;
- Módulo de Administração: é constituído pelas classes responsáveis pela inclusão, alteração e exclusão de produtos;
- Módulo de Pedidos: é responsável pela manutenção do Carrinho de Compras e fechamento do pedido;
- Módulo de Integração: este módulo integra os outros módulos da aplicação, fornecendo também uma interface ao usuário final.

2.3.1. Arquitetura em camadas

A arquitetura do projeto foi realizada em camadas de responsabilidades. Esta abordagem possui muito benefícios, dentre eles se destacam: maior facilidade de

manutenção, devido ao isolamento provido entre as camadas; maior facilidade na divisão do trabalho na equipe de desenvolvimento; possibilidade de utilização de tecnologias diversas, em consequência do fraco acoplamento entre as camadas; implantação fragmentada do sistema, devido a componentização; e um entendimento facilitado, tendo em vista que esta abordagem é amplamente utilizada atualmente. É importante notar que as camadas inferiores devem ser independentes das superiores, delegando a responsabilidade da interação a camada imediatamente superior. A nomenclatura das camadas não possui um padrão estabelecido, permanecendo sempre aquela que facilita o entendimento da mesma. Um resumo da arquitetura e as interações entre as camadas encontram-se na figura 2.4.

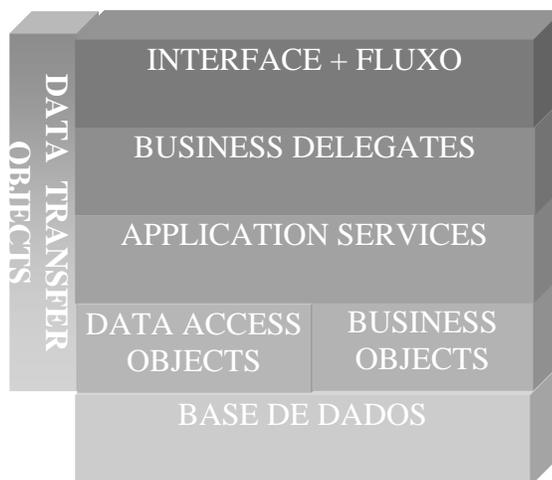


Figura 2.4 – Camadas da Aplicação

- Interface + Fluxo: esta camada é o ponto de interação do usuário com o sistema. É sua responsabilidade garantir uma interface agradável e com boa usabilidade. Esta camada também é responsável pelo fluxo da aplicação, quando a convergência de diversos passos for necessária para a utilização de alguma funcionalidade;
- Business Delegates: esta camada tem a responsabilidade de centralizar o acesso aos serviços oferecidos pela aplicação, escondendo detalhes de implementação das camadas cliente;

- Application Services: esta camada expõe os serviços oferecidos ao sistema em relação a suas funcionalidades de negócio. A tecnologia *Enterprise Java Beans* (EJB) é utilizada para isto, através de *Session Beans*;
- Data Access Objects: esta camada tem o objetivo de acesso de leitura aos dados da aplicação. Seu objetivo é prover acesso customizado e com bom desempenho;
- Business Objects: esta camada tem como objetivo mapear os objetos de domínio e de realizar regras específicas do negócio. A tecnologia *Enterprise Java Beans* (EJB) é utilizada para isto, através de *Entity Beans*.
- Base de Dados: Esta camada possui o único objetivo de armazenar os dados que devem ser persistidos. Utilizamos um banco de dados relacional como tecnologia de suporte.
- Data Transfer Objects: Esta camada “vertical” possui o objetivo de transportar as informações de uma forma eficiente entre as demais camadas, com exceção da Base de Dados. Utilizamos POJOs (Plain Old Java Objects) para isto.

2.4. PADRÕES UTILIZADOS

“Cada padrão é uma regra de três partes, que expressa uma relação entre um certo contexto, um problema e uma solução.” - Christopher Alexander

Foram utilizados diversos padrões no decorrer deste projeto, pois são reflexo da experiência e conhecimento de desenvolvedores, são altamente reusáveis e flexíveis em relação a diversos tipos de problemas e definem soluções a problemas recorrentes. Mesmo que o uso de padrões não garanta o sucesso da aplicação, o aprendizado do uso devido desses padrões aumenta potencialmente a produtividade do desenvolvimento.

2.4.1. Data Transfer Object

“[...]classes Java simples chamadas objetos de transferência de dados, que contém e encapsulam grande quantidade de dados em um lote transportável pela rede.”

[Marinescu04]

Baseado no padrão Transfer Object [J2EEP06], este padrão tem como objetivo fazer a comunicação entre as camadas de um sistema de informação. A opção pela utilização deste padrão foi devida a estrutura do sistema ser dividida em camadas e ser necessário um meio de transporte entre elas. Usamos dois tipos de Data Transfer Object, o Custom Data Transfer Object e o Domain Data Transfer Object.

Um Custom Data Transfer Object é um objeto de transferência de dados que não mapeia nenhuma estrutura de dados do servidor, sendo objetos projetados de acordo com a necessidade do cliente. A utilização deste padrão está principalmente concentrada nas camadas “superiores”, onde nem sempre existe a intenção ou a necessidade de uma réplica exata dos objetos de domínio. O fluxo de interface por exemplo, utiliza bastante este padrão para o transporte de informações entre seus passos.

Os objetos de transferência de dados do domínio fornecem aos clientes cópias de objetos de domínio localizados no servidor. Este padrão foi utilizado não apenas para suprir eventuais *overheads* de performance causados pelos *Entity Beans*, mas também por fornecer uma visão do modelo de domínio com um menor grau de hiato para os clientes das camadas superiores.

2.4.2. Service Locator

“O Service Locator encapsula os serviços de pesquisa da API (nomeação), as dependências do fornecedor, as complexidades da pesquisa e a criação do objeto de negócios, além de fornecer uma interface simples para os clientes. Isso reduz a complexidade do cliente e aumenta a reutilização.”

“Use um Service Locator para implementar e encapsular o serviço e a pesquisa de componente. Um Service Locator oculta os detalhes de implementação do mecanismo de pesquisa e encapsula as dependências relacionadas.

Os clientes da aplicação podem reutilizar o Service Locator para reduzir a complexidade do código, fornecer um único ponto de controle e melhorar o desempenho fornecendo uma facilidade de cache. Você normalmente precisa de um único Service Locator para a aplicação inteira. Entretanto, é comum ver duas implementações do Service Locator em uma aplicação, uma para a camada de apresentação e uma para a

camada de negócios. Um Service Locator reduz a dependência do cliente em relação à infra-estrutura de pesquisa subjacente e otimiza a pesquisa com uso intensivo de recursos e as operações de criação.

Um Service Locator normalmente é implementado como um Singleton. Entretanto, as aplicações J2EE executam em um contêiner J2EE que usa diversos carregadores de classe e JVMs, tornando impossível ter um verdadeiro Singleton com apenas uma instância. Portanto, você talvez precise de várias instâncias do Service Locator em um ambiente distribuído. Entretanto, como o Service Locator não armazena em cache qualquer dado que precise ser acessado simultaneamente você não precisa replicar e sincronizar as alterações por meio dessas várias instâncias .“ [ACM04]

O motivo da utilização deste padrão foi principalmente pela intenção de isolar a complexidade na localização e busca dos serviços relacionados a plataforma J2EE que possui codificação e tratamento de exceção bem específicos. Ao utilizar este padrão facilita-se também a implantação do projeto em um ambiente distribuído e suavizamos as possíveis trocas de regras e codificação numa eventual troca de plataforma.

2.4.3. Data Access Object

“[...] Use um Data Access Object para abstrair encapsular todo acesso ao armazenamento persistente. O Data Access Object gerencia a conexão com a fonte de dados para obter e armazenar dados.[...] O Data Access Object (também conhecido simplesmente como DAO) implementa o mecanismo de acesso necessário para trabalhar com a fonte de dados. Independente do tipo de fonte de dados usado, o DAO sempre fornece uma API uniforme para seus clientes .“ [ACM04]

Este padrão foi utilizado principalmente para o contorno do problema de performance gerado na consulta de grandes listas de dados à partir de *Business Objects* implementados como *Entity Beans*. Seu uso restringe-se apenas a consulta de dados e

não deve, em hipótese alguma, ter acesso de escrita aos repositórios de dados. Este padrão é fortemente acoplado ao padrão *Transfer Object* no projeto.

2.4.4. Application Service

“O ApplicationService encapsula a lógica de negócios para fornecer um serviço específico. Ele pode encapsular vários tipos de lógica de negócio, como a lógica comum em diversos BusinessObjects, usar lógica de negócio específica de caso ou uma lógica específica para um tipo particular de cliente ou canal. Um ApplicationService pode chamar um método de negócio em um BusinessObject ou outro ApplicationService. A implementação dos Application Services como POJOs é mais comum porque esta manobra a capacidade de reutilização da lógica de controle em diferentes clientes e facilita a criação de camadas dos Application Services.” [ACM04]

Este padrão é importante para qualquer projeto bem estruturado, pois como se pode observar na definição do mesmo, são permitidos diversos níveis de abstração e disponibilização de funcionalidades. Este padrão foi utilizado juntamente com o Session Façade para a disponibilização dos serviços básicos dos módulos estruturais ao encapsular o acesso aos *Business Objects*.

2.4.5. Business Objects

“Os Business Objects encapsulam e gerenciam os dados de negócios, comportamento e persistência. Os Business Objects ajudam a separar a lógica de persistência da lógica de negócios. Os Business Objects mantêm os dados de negócios principais e implementam o comportamento que é comum a toda aplicação ou domínio.” [ACM04]

Decidimos pela utilização deste padrão principalmente por possibilitar uma estrutura que permita uma melhor reutilização dos componentes de negócio. Outros benefícios importantes são o impedimento de acesso direto para escrita nos repositórios

de dados e uma manutenção mais suave e centralizada das principais regras de negócio. Os EJB *Entity Beans* implementaram este padrão no projeto do Sistema de Compras Web.

2.4.6. Session Façade

“Uma Session Façade é implementada como um Session Bean e interage com os componentes de negócios, como Business Objects e Application Services. Uma Session Façade fornece uma camada de serviço remoto que expõe apenas as interfaces usadas pelos clientes.

Uma Session Façade funciona melhor quando contém pouca ou nenhuma lógica de negócios. Quando a lógica de negócios está presente, ela deve ser colocada no Application Service, que é chamado pela Session Façade.” [ACM04]

Este padrão é um dos mais utilizados no mercado. Baseado no Padrão *Façade* [GOF94] sua idéia é praticamente a mesma: centralizar chamadas co-relacionadas em um ou mais “canais” de comunicação. A grande diferença de *Session Façade*[ACM04] para o padrão *Façade* é que o primeiro indica um *Session Bean Stateless* para servir como “canal” de comunicação. Este padrão foi bastante utilizado principalmente nas integrações entre os módulos, pois a utilização do mesmo diminui a necessidade de conhecimento estrutural do serviço por parte do cliente.

2.4.7. Business Delegate

“Um Business Delegate atua como uma abstração de negócios no cliente: ele abstrai e oculta os detalhes de implementação dos serviços de negócios. O uso de um Business Delegate reduz o acoplamento entre o cliente e os serviços de negócios do sistema. Dependendo da estratégia de implementação, o Business Delegate protegerá os clientes de uma possível volatilidade na implementação da API de serviço de negócios. Potencialmente, isso reduz o número de alterações que devem ser feitas no código do cliente quando a API de serviço de negócios ou sua implementação subjacente sofrem alterações.” [ACM04]

Apesar deste padrão ser indicado para equipes maiores, ele foi adotado por outros motivos como: reduzir significativamente a complexidade dos códigos clientes, facilitar uma futura migração de plataforma para a especificação EJB 3.0 através da centralização dos códigos de acesso aos EJB's e uma melhor estruturação do projeto.

FERRAMENTAS UTILIZADAS E O SISTEMA

O JBoss foi escolhido como servidor de aplicação para este sistema. Atualmente ele é considerado um dos mais robustos no mercado, é amplamente utilizado e é *Open Source*. A base de dados foi criada no Hypersonic, sistema de gerenciamento de banco de dados embutido no JBoss.

Para a construção de todos os diagramas do Sistema de Compras Web, foi utilizado o Magic Draw. Nesta ferramenta *case* foram gerados os diagramas que serviram de insumo para o AndroMDA realizar a geração do código de infra-estrutura da aplicação. A ferramenta AndroMDA utiliza as classes produzidas e exportadas para o formato XMI para esta tarefa, seguindo os princípios da Model Driven Architecture.

O resultado alcançado pela geração de código foi importado no Eclipse, que foi a IDE utilizada, e onde foram implementados os métodos de negócio e desenvolvida a camada de apresentação com a utilização do plugin Exadel Studio, que fornece suporte facilitado ao framework Struts.

Além destas ferramentas, foram utilizados também o Maven, ferramenta de gerência de projeto e configuração, e o Ant, para a automação de scripts de empacotamento e implantação da aplicação.

3.1. SISTEMA DE COMPRAS WEB

O Sistema de Compras Web, em sua apresentação é composto por duas partes: uma interface para o usuário administrador, denominada tela de administração, e outra interface para o usuário cliente, denominada tela de compras. Nas figuras 3.1 e 3.2 mostram as respectivas telas:



Figura 3.1 – Tela administração



Figura 3.2 – Tela de compras

O usuário administrador tem 6 opções de seleção em seu ambiente, sendo 3 de listagem e 3 de cadastro. Ao iniciar uma loja, primeiramente o administrador deverá cadastrar um ou mais grupos, clicando na opção “*add group*” do sistema. A Figura 3.3 mostra o cadastro a ser preenchido pelo administrador para a criação de um grupo.

SISTEMA DE COMPRAS

t groups add group list brands add brand list products add product

* **Nome:** Livraria

* **Descricao:** Livros e Revistas

Voltar Limpar Enviar Dados

Figura 3.3 – Tela de cadastro de grupo

Depois de criados os grupos, o administrador poderá listar todos os grupos registrados no sistema selecionando a opção “*list groups*”. Nesta tela de listagem de grupos, mostrada na figura 3.4, o administrador pode optar por alterar um grupo já existente (ao clicar no link Editar) ou excluí-lo (ao clicar em Excluir). A tela de alteração de grupo é a mesma que a de cadastro, apresentada na figura 3.3.

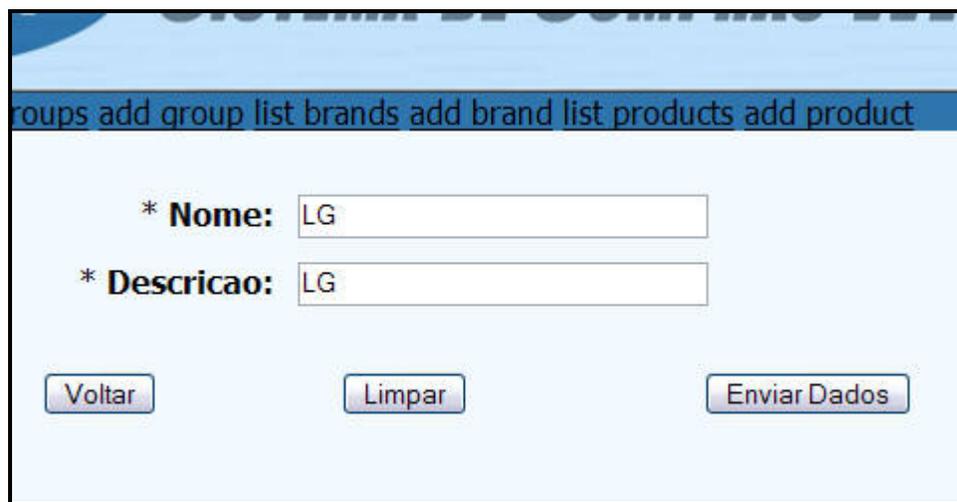
list groups add group list brands add brand list

4 items found, displaying all items.1

| Name | |
|-------------|--|
| Informatica | Editar Remover |
| Video | Editar Remover |
| Audio | Editar Remover |
| Brinquedos | Editar Remover |

Figura 3.4 – Listagem dos grupos

Para prosseguir com a inclusão de dados na loja, o usuário deverá criar as marcas que estarão disponíveis no sistema. Para isso, o administrador deverá clicar na opção “*add brand*” localizada na tela de administração. A tela de cadastro que deve ser preenchida para a criação da marca está apresentada na Figura 3.5.



The image shows a web application interface for brand registration. At the top, there is a blue navigation bar with the following links: "groups", "add group", "list brands", "add brand", "list products", and "add product". Below the navigation bar, the main content area is light blue and contains two text input fields. The first field is labeled "* Nome:" and contains the text "LG". The second field is labeled "* Descricao:" and also contains the text "LG". Below the input fields, there are three buttons: "Voltar" (Back), "Limpar" (Clear), and "Enviar Dados" (Send Data).

Figura 3.5 – Tela de cadastro da marca

Da mesma forma que foi feito com os grupos, o administrador pode listar as marcas na opção “*list brands*” da tela de administração. O resultado gerado é mostrado na figura 3.6, onde se pode visualizar que as opções de exclusão e alteração também estão disponíveis. A tela de alteração é a mesma de inclusão (figura 3.5).



Figura 3.6 – Tela de listagem de marcas

Para prosseguir na aplicação o administrador deve efetuar o cadastro de um ou mais produtos. A tela de cadastro de produtos pode ser acessada a partir da tela de administração, no link “*add product*”, e pode ser visualizada na figura 3.7.

* **Nome:** bicicleta js

* **Descricao:** bicicleta js special

* **Preço:** 99

* **Tamanho:** ARO 20

* **Peso:** 12

* **Imagem:** bicicleta.jpg

* **Marca:** Caloi

* **Grupo:** Brinquedos

Voltar Limpar Enviar Dados

Figura 3.7 – Tela de cadastro de produtos

Todos os produtos cadastrados podem ser listados ao se clicar na opção “*list products*” na tela de administração que pode ser visualizada na figura 3.8. Nela ficam disponíveis as opções de alteração e exclusão de cada produto e, como nos cadastros anteriores, a tela de alteração é a mesma de inclusão (figura 3.7).

| list groups add group list brands add brand list products add product | | | |
|---|--------------------------------|-------|--|
| 10 items found, displaying all items.1 | | | |
| Name | Description | Price | |
| Triciclo Tabajara | Triciclo Tabajara | 29 | Editar Remover |
| Joystick | Joystick | 12 | Editar Remover |
| Impressora Genérica | Impressora Genérica | 499 | Editar Remover |
| Gravador de CD Piratao | Grava de tudo! | 121 | Editar Remover |
| DVD Player Sony | DVD Player Sony | 299 | Editar Remover |
| Discman Piratao | Discman que toca de tudo mesmo | 123 | Editar Remover |

Figura 3.8 – Tela de listagem de produtos

Após a efetuação do cadastro dos dados básicos na tela de administração do sistema, a tela de compras já possui os dados necessários para que seja acessada pelos clientes. Nesta tela, já apresentada na figura 3.2, o usuário cliente pode clicar nos produtos que deseja adquirir, enviando-os para o seu carrinho de compras. O carrinho pode ser visualizado ao se clicar no botão “*Minhas Compras*” ou “*Concluir Compras*”. Na tela de visualização de carrinho, que pode ser vista na figura 3.9, são listados os produtos que estão no carrinho de compras do cliente e pode ser alterada a quantidade de cada produto que será comprado. Nesta tela devem ser preenchidos os dados do cliente para seu registro.

| | | | |
|----------|--------------------------------|----|-------------------------------------|
| Joystick | <input type="text" value="2"/> | 12 | retirar do carrinho |
| Cabo USB | <input type="text" value="1"/> | 15 | retirar do carrinho |

* **Nome:**

* **Endereço:**

* **Cidade:**

* **Estado:**

* **País:**

* **Email:**

* **Telefone:**

Figura 3.9 – Tela de visualização do pedido

Além disso, o cliente pode fazer pesquisas de produtos de duas formas. Clicando nos links dos grupos existentes, localizados no lado esquerdo da tela inicial, o cliente poderá buscar os produtos deste grupo. Digitando o nome ou parte do nome do produto procurado na área de “*Busca de produtos*”, também localizada na área esquerda da tela, o cliente estará realizando uma busca por parte do nome. O resultado dessa pesquisa é mostrado como na figura 3.10. A partir desta lista, é possível incluir os produtos encontrados no carrinho de compras.



Figura 3.10 – Tela de resultado de uma pesquisa

É possível também ao cliente executar uma busca avançada, onde, além do nome do produto, é possível limitar a busca por uma marca ou grupo existente no sistema. Para acessar esta busca o cliente deve clicar no botão “Busca Avançada”. A figura 3.11 ilustra esta opção.



Figura 3.11 – Tela de busca avançada

3.2. CRIANDO A APLICAÇÃO

É importante destacar os passos seguidos para a criação de uma aplicação usando as tecnologias usada na aplicação para demonstrar o ganho gerado. O primeiro

passo é a criação de um projeto usando o AndroMDA. Para isso é digitado um comando no prompt de comando (figura 3.12) que iniciara o AndroMDA e criará o projeto baseado em uma série de perguntas básicas. Uma estrutura de arquivos é gerada com o nome do projeto. É importante destacar que nessa estrutura de arquivos temos pastas “Sources”, que é a fonte para o AndroMDA, e pastas “Targets”, em que o AndroMDA gera a aplicação usando como base o que esta na source.

```
C:\>maven andromdapp:generate
|-----|
|  /  \  | Apache
| /    \ | ~ intelligent projects ~
|_|    | | v. 1.0.2
Please enter your first and last name (i.e. Chad Brandon):
```

Figura 3.12 – Geração de projetos no AndroMDA

Após a criação do projeto, usa-se a ferramenta case, sendo usado aqui o Magic Draw, para a criação dos diagramas de classes que servem de base para gerar o código base para a aplicação. Nos diagramas (Apêndice F) são usados estereótipos para que o androMDA consiga gerar o objeto a que se deseja alcançar. O diagrama deve ser criado no arquivo xmi encontrado na pasta mda do projeto gerado. A figura 3.13 ilustra a criação dos diagramas.

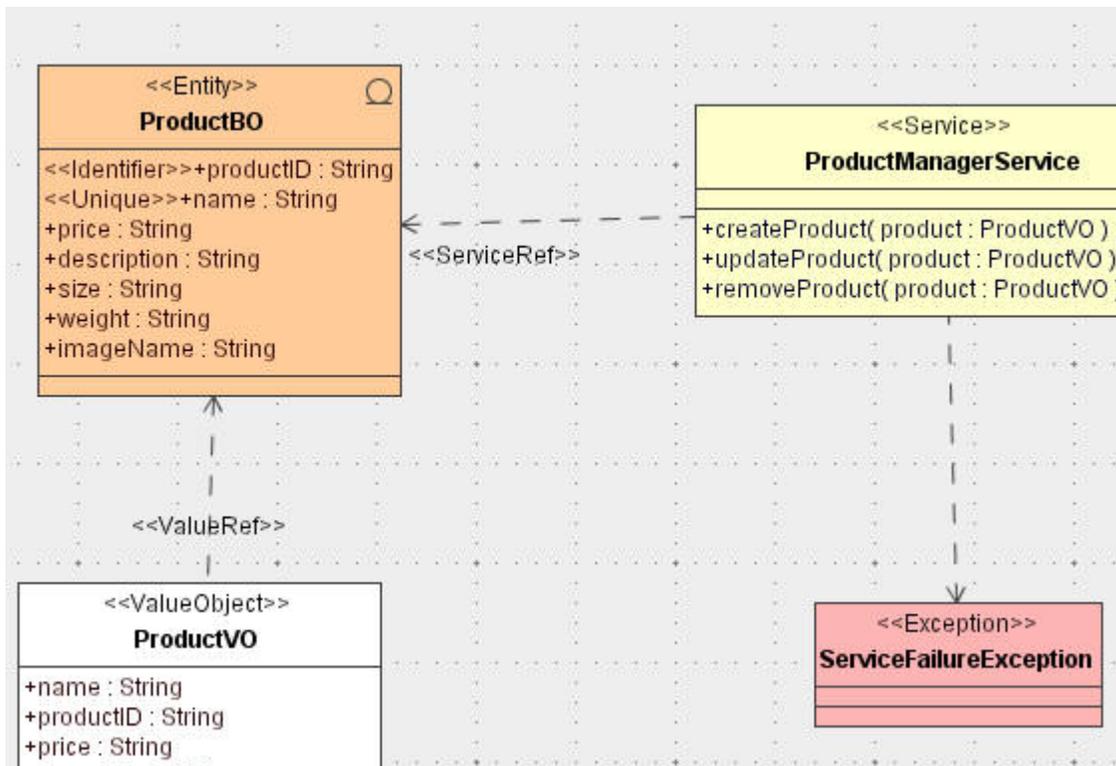


Figura 3.13 – Diagrama de classes

Após os diagramas, segue o primeiro build na aplicação, que irá gerar todos os objetos sem as respectivas implementações das regras de negócio. O comando maven deve ser dado no prompt de comando na pasta do projeto para que se inicie o build. Após a geração a implementação das regras de negócio devem ser feitas na IDE, lembrando-se de usar os arquivos da pasta source. Após a implementação um novo build deverá ser dado para que o AndroMDA junte a implementação no target. Assim a aplicação está pronta e deve ser integrada. O Build e a implementação estão ilustradas nas Figuras 3.14 e 3.15.

```

Memory: 4M/5M
-----
Starting the reactor...
Your processing order:
unicart MDA
unicart Common
unicart Core Business Tier
unicart Application
-----
Executing multiproject:install-callback unicast MDA
Memory: 3M/5M
-----

multiproject:goal:
build:start:

multiproject:install-callback:
[echo] Running pom:install for unicast MDA
pom:install:
andromda:run:
[echo] +-----+
[echo] |           R u n n i n g   A n d r o M D A           |
[echo] +-----+

```

Figura 3.14 – O build no maven

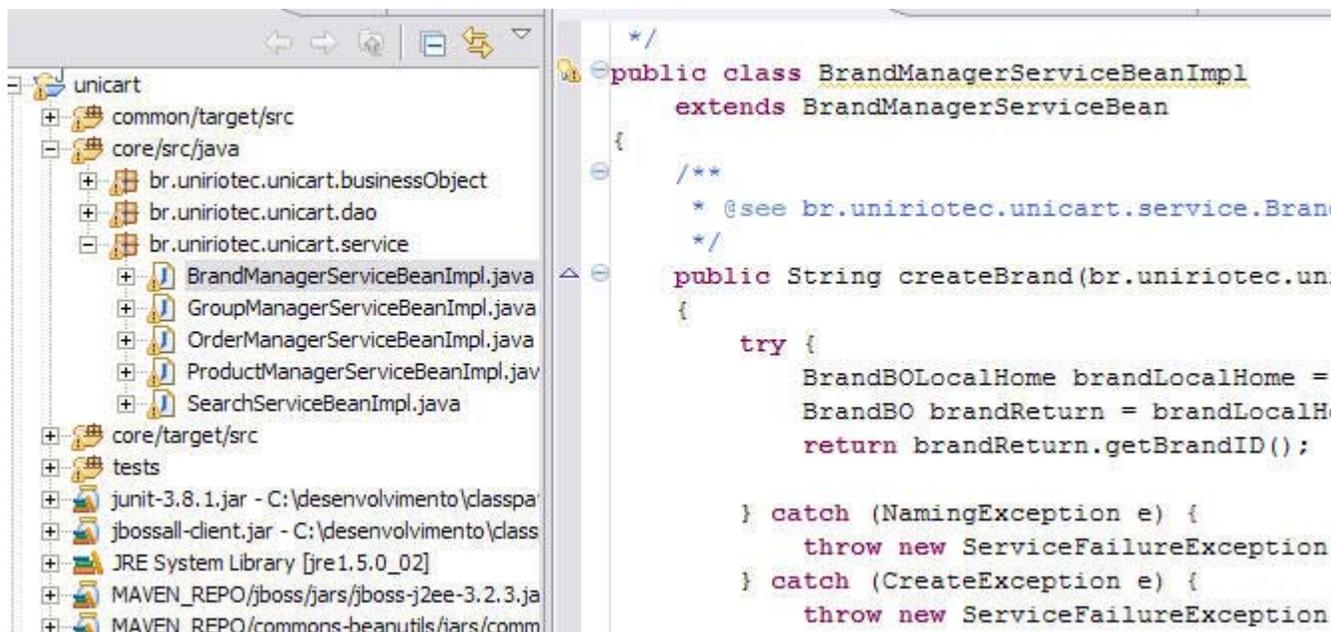


Figura 3.15 – Implementação no Eclipse

Com isso resta o projeto Web, que irá integrar todos os objetos gerados até então. O projeto web foi um projeto a parte e é baseado no Struts. Para o uso do struts

foi utilizado o plugin Exadel que mantém o controle de todo o fluxo da aplicação de maneira simples e intuitiva, como pode ser visto na figura 3.16.

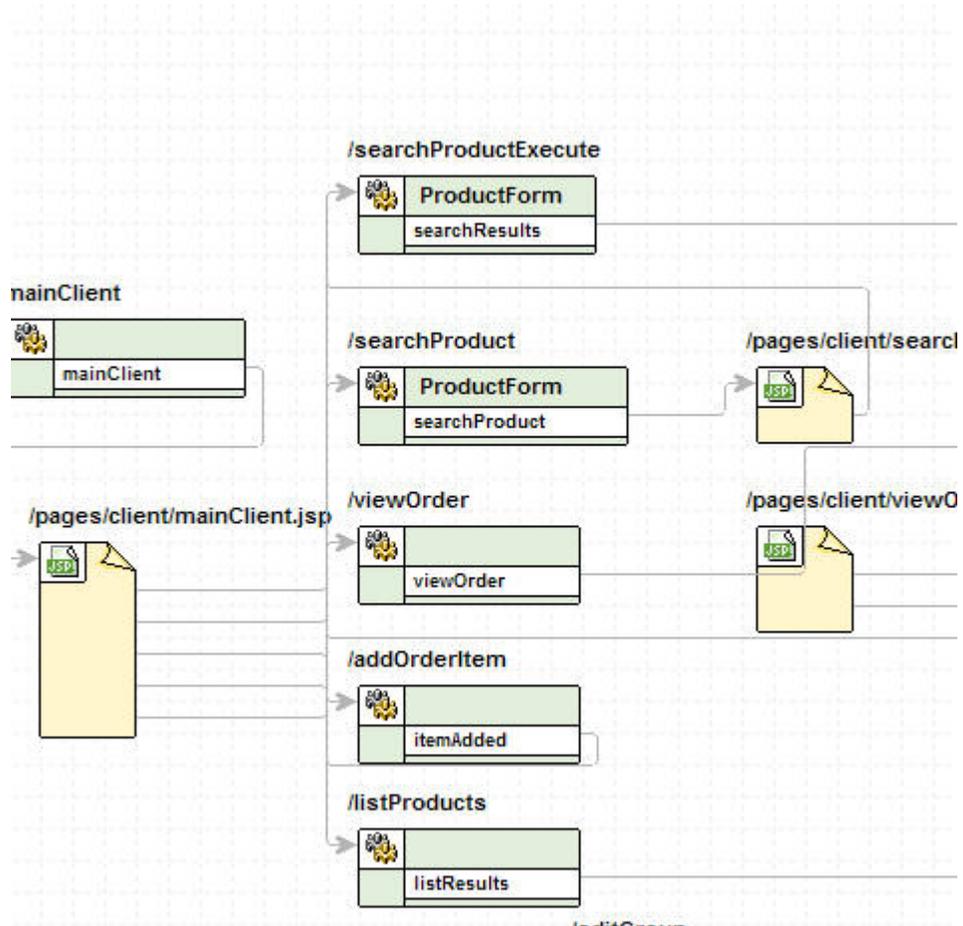


Figura 3.16 – Controle do fluxo com o Exadel

CONCLUSÃO

Durante a execução deste projeto, foi utilizado um processo de desenvolvimento coeso e bem elaborado, juntamente com diferentes técnicas e padrões de projeto e uma metodologia de desenvolvimento alinhada com a *Model Driven Architecture* (MDA) [OMG06], facilitando bastante as atividades de desenvolvimento e manutenção do *software*.

Sobre o processo de desenvolvimento que foi utilizado, foi alcançada uma visão positiva devido à organização de atividades e aos resultados que foram gerados ao final de cada uma destas. A adaptação deste processo foi fundamental para um desenvolvimento organizado e um resultado final bem documentado. A MDA contribuiu bastante com a aplicação do processo devido à sincronização dos modelos com o código.

A utilização de *Model Driven Architecture* (MDA) [OMG06] também foi algo muito produtivo ao projeto. Isto implicou diretamente em uma maior atenção à modelagem das classes do sistema e numa conseqüente diminuição no esforço de desenvolvimento, devido à geração de código. Outra grande conseqüência ligada ao uso desta tecnologia é a atualização constante dos modelos do sistema, pois é a partir deles que ocorrem as gerações de código.

Assim, acreditamos que foi alcançado o resultado desejado neste projeto: um software com o código de manutenção facilitado e bem documentado. O sucesso obtido neste projeto nos faz crer que este tipo de abordagem poderá ser de grande importância no futuro próximo da Engenharia de Software.

APÊNDICE A – MINI-MUNDO

Sistema de Compras Web

O sistema servirá para que possamos vender nossos produtos pela web e é dividido em duas partes.

<ADMINISTRAÇÃO>É onde devemos poder cadastrar nossos PRODUTOS e cada um deles deve estar vinculado a uma MARCA e a um GRUPO. O PRODUTO deve conter: Nome, descrição, preço, tamanho, peso, imagem, marca e grupo. Além do cadastro, deverá ser possível a listagem destes PRODUTOS cadastrados, com a opção de edição e remoção dos mesmos. Deverá também ser possível o cadastro de MARCAS, que deverá conter um nome e uma descrição e o cadastro de GRUPOS que contém os mesmos dados. Ambos devem ter métodos para edição e remoção iguais aos de PRODUTOS, além da listagem dos mesmos.

<CLIENTE>O Cliente poderá fazer pesquisas dos PRODUTOS da loja seja só por parte do nome (Busca Simples), ou filtrar a busca por GRUPOS e MARCAS (Busca Avançada). Ao localizar PRODUTOS no site, o usuário poderá incluir estes PRODUTOS no carrinho. A qualquer momento o cliente pode visualizar o carrinho vendo todos os PRODUTOS incluídos até então, e ter a opção de finalizar o PEDIDO. Um PEDIDO é o resultado das informações dos PRODUTOS que o cliente quer comprar. Ao requisitar o fim do PEDIDO, o sistema deverá mostrar o carrinho e exibir uma tela para cadastro dos dados de entrega e para os dados financeiros. Se tudo for corretamente preenchido o PEDIDO é cadastrado no sistema e finalizado.

APÊNDICE B – GLOSSÁRIO

Definições

- Produto – item negociável em uma loja virtual;
- Loja virtual – A utilização do sistema para a realização de comercialização de produtos on-line;
- Comprador – usuário que tem a intenção de adquirir produtos à venda na loja virtual;
- Pedido – Quando um comprador solicita a negociação de um ou mais itens (produtos);
- Compra – Processo de negociação de um pedido (Pedido + definição da entrega e pagamento).
- Marca – A Marca da qual um ou mais produtos estarão vinculados
- Grupo – Grupo de produtos

Atores

- Cliente – Usuário que deseja efetuar uma compra na loja virtual;
- Gerente – Usuário que tem acesso aos dados relativos a B.I.;
- Administrador – Usuário que alimenta o sistema;

APÊNDICE C – VISÃO

<ComprasWeb>
Visão Básica
Versão <3.0>

Histórico da Revisão

| Data | Versão | Descrição | Autor |
|-------------|---------------------------|--|--------------|
| 07/09/2005 | 1.0 - Minuta da Concepção | Primeira Versão seguindo RUP | Grupo |
| 12/03/2006 | 2.0 - Versão Incremental | Revisão e adição de conteúdo baseado na Arquitetura definida, projeto e redefinição de escopo. | Grupo |
| 07/05/2006 | 3.0 – Versão Final | Revisão final do documento | Grupo |
| | | | |

Índice Analítico

[1. Introdução](#)

[1.1 Referências](#)

[2. Posicionamento](#)

[2.1 Descrição do Problema](#)

[2.2 Sentença de Posição do Produto](#)

[3. Descrições dos Envolvidos e dos Usuários](#)

[3.1 Resumo dos Envolvidos](#)

[3.2 Resumo dos Usuários](#)

[3.3 Ambiente do Usuário](#)

[3.4 Resumo das Principais Necessidades dos Envolvidos ou dos Usuários](#)

[3.5 Alternativas e Concorrência](#)

[4. Visão Geral do Produto](#)

[4.1 Perspectiva do Produto](#)

[4.2 Suposições e Dependências](#)

[5. Recursos do Produto](#)

[6. Outros Requisitos do Produto](#)

Visão

1. Introdução

A finalidade deste documento é coletar, analisar e definir necessidades e recursos de nível superior do Sistema Compras Web . Ele se concentra nos recursos necessários aos envolvidos e aos usuários-alvo e nas razões que levam a essas necessidades. Os detalhes de como o Sistema Compras Web satisfaz essas necessidades são descritos no caso de uso e nas especificações suplementares.

1.1 Referências

- Modelo RUP;
- Utilizando Padrões e UML;

2. Posicionamento

2.1 Descrição do Problema

A principal motivação para o desenvolvimento deste sistema foi a idealização de uma ferramenta portátil, com alta escalabilidade e de fácil manutenção e adaptação. Estas características são decorrentes da arquitetura e do processo de desenvolvimento bem definido com base em padrões já existentes que se adequam a solução mencionada. Em geral os sistemas de informação de médio e grande porte que não são projetados e construídos com o devido tratamento destas questões importantes sofrem sérios problemas quando são implantados ou quando necessitam de customizações.

O Sistema Compras Web procura viabilizar uma forma de compra e venda de produtos pela Internet. Engloba a visualização por parte dos possíveis clientes e a seleção dos produtos que os interessam, e a efetuação da compra também por parte dos clientes. Na prática, um sistema destes facilita tanto a vida do cliente, que pode efetuar compras na comodidade de seu lar ou ambiente de trabalho, quanto a do comerciante, que tem a possibilidade de aumentar muito as suas vendas sem ter que gastar com espaço físico e pessoal(quando gasta a quantia é muito menor do que se gasta para abrir e manter uma loja).

2.2 Sentença de Posição do Produto

Este produto é voltado para empresas ou instituições que desejem efetuar algum tipo de venda através da internet e que exijam (ou não) necessidades específicas. O Compras Web foi planejado com o objetivo de se adaptar da forma mais rápida e fácil às necessidades que um cliente possa ter.

3. Descrições dos Envolvidos e dos Usuários

Cliente – Usuários da grande rede de computadores que irão acessar o sistema para pesquisar preços e efetuar uma compra.

Administradores – Funcionários da empresa que adquiriu o Compras Web que tem a função de alimentar o sistema, de uma forma geral.

3.1 Resumo dos Usuários

| Nome | Descrição | Responsabilidades |
|---------------|---|--|
| Cliente | Usuário que acessa o sistema para efetuar compras | - Efetua buscas e compras no sistema - Seleciona produtos que deseja comprar - Finaliza pedido informando seus dados |
| Administrador | Funcionários que alimentam o sistema | - alimenta o sistema com dados básicos e fundamentais |

3.2 Ambiente do Usuário

- Usuário-alvo do Sistema: Compradores e funcionários da empresa que utilizará o sistema;
- Por ser um sistema web deve suportar uma grande carga de acesso simultâneo;
- **Ciclos de sistema e período:**

| Ciclos | Atividade | Tempo estimado | Entrada | Saída |
|--------|-------------------------------|----------------|---|-----------------------|
| 1 | Pesquisa e Seleção de produto | 6 min. | - | Produtos Selecionados |
| 2 | Requisita Compra | 3 min. | Usuário deve revisar valor total e informar seus dados para entrega | Registro de Pedido |

- O Compras Web foi projetado sobre uma plataforma portátil e poderá rodar em servidores com diferentes Sistemas operacionais. A especificação mínima para a instalação da aplicação deve ser avaliada caso a caso de acordo com as especificações do Servidor de Aplicação escolhido e do Sistema de Gerenciamento de Banco de Dados e com o nível de utilização do Sistema.
- A arquitetura utilizado no projeto do Compras Web foi adotada com o intuito de agilizar a interligação deste com outros sistemas, além das alterações internas necessárias.

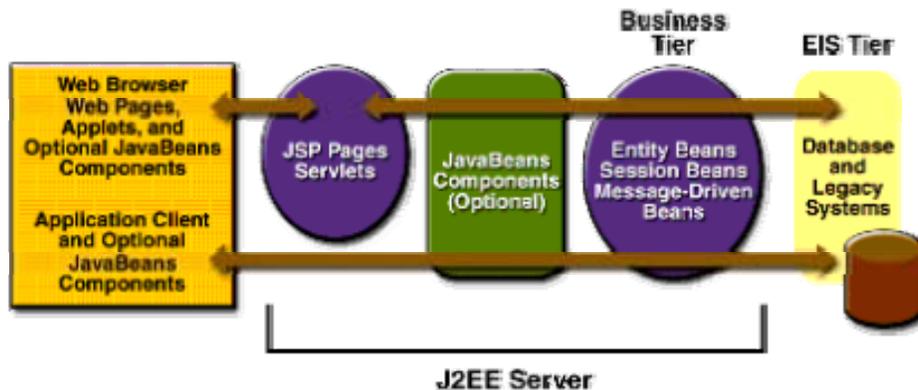
3.4 Principais Necessidades dos Usuários ou dos Envolvidos

As principais necessidades de quem recorre a um produto voltado para a venda na Web são a simplificação da execução de vendas, a ampliação dos negócios e a diminuição de custo de venda do produto. Para isso uma ferramenta de vendas para deve ser confiável e robusta. Outro ponto muito importante é a diminuição do custo de implantação e de manutenção

4. Visão Geral do Produto

4.1 Perspectiva do Produto

A aplicação Compras Web foi construída para rodar em um Servidor de Aplicação J2EE. No desenvolvimento foi utilizado o JBOSS versão 4.0.2 e a JDK na versão 1.5.x. O Sistema de Gerenciamento de Banco de Dados utilizado foi o Hypersonic embutido no JBOSS. A arquitetura utilizada para a aplicação não a torna dependente destes softwares, no entanto no caso de alteração de ambiente serão necessárias algumas alterações e testes de homologação.



5. Recursos do Produto

A finalidade do Compras Web é viabilizar a venda de produtos através da Internet. Abaixo segue um resumo das atividades que serão executadas no sistema:

Cadastros – Deve ser possível cadastrar grupos (nome, descrição), Marca (nome, descrição), Produto (nome, descrição, preço, tamanho, grupo, marca). Após o produto ser vinculado a um pedido este não pode ser excluído, assim como a marca e o grupo também não poderão ser excluídos quando vinculados a um produto. Cada item destes deve poder ser pesquisado .

Pesquisa – Um cliente irá acessar a página inicial de nosso sistema e poderá efetuar pesquisa por alguns critérios como nome do produto ou grupo. Após selecionar um produto ele pode adicioná-lo ao seu Carrinho, informando a quantidade.

Carrinho – Esta é uma estrutura para armazenar as informações da compra que está sendo efetuada. Cada cliente que acessa o site possuirá seu Carrinho individual que contém os produtos e a quantidade que ele deseja comprar. Ele deve poder ser visualizado e alterado (produtos excluídos).

Pedido – Ao final de sua pesquisa, o cliente deve informar que deseja finalizar o pedido. Neste momento ele precisa inserir seus dados pessoais e após confirmação o pedido é registrado.

6. Outros Requisitos do Produto

- Disponibilidade: O sistema deve estar disponível continuamente;
- O sistema deve ter a capacidade de processar informações de um número alto de usuários concorrentemente. O número de usuários deve crescer sem comprometer a performance do sistema.
- Escalabilidade: O sistema deve ter a capacidade de aumentar o desempenho sob carga com a melhoria de recursos.

APÊNDICE D – CASOS DE USO

Casos do módulo "Administração"

Gerenciar Marca

1. Geral

| | |
|----------------------------|---|
| ID do Caso de Uso | 101 |
| Nome do Caso de Uso | Gerenciar Marca |
| Objetivo | Inclui/Altera/Deleta uma marca no sistema |
| Descrição | Administrador entra com os dados de uma marca no sistema para gerenciar |
| Atores | Administrador |
| Pré-condições | |
| Pós-condições | A marca é cadastrada/Alterada/deletada |
| Inclusões/Extensões | Listar Marcas |

2. Fluxo de eventos

- 1 – Administrador entra na interface de administração.
- 2 - O administrador seleciona a ação relacionada a marca.
- 3 - O sistema verifica os dados enviados e realiza a função.

3. Extensões

2a – Subfluxo - Administrador quer alterar marca

1. O administrador lista as marcas (INCLUDE Listar Marcas)
2. O administrador seleciona a opção de editar marca.
3. O administrador entra com os novos dados (Nome, descrição) para a marca.
4. Sistema verifica os dados enviados e registra a alteração.

2b – Subfluxo - Administrador quer excluir uma marca

1. O administrador lista as marcas (INCLUDE Listar Marcas)
2. O administrador seleciona a opção de excluir marca.
3. Sistema verifica os dados e realiza a exclusão

2c – Subfluxo - Administrador quer incluir uma marca

1. Administrador requisita a inclusão de uma nova marca.
2. Administrador envia o nome e a descrição da marca a ser incluída.
3. Sistema verifica os dados enviados e inclui a marca.

2c.a – Marca já existe

1. O sistema retorna um erro.

Gerenciar Grupo

1. Geral

| | |
|----------------------------|--|
| ID do Caso de Uso | 102 |
| Nome do Caso de Uso | Gerenciar Grupo |
| Objetivo | Incluir/Alterar/Deletar um grupo no sistema |
| Descrição | Administrador entra com os dados de um grupo no sistema e o gerencia |
| Atores | Administrador |
| Pré-condições | |
| Pós-condições | O grupo é cadastrado/alterado/deletado |
| Inclusões/Extensões | Listar Grupos |

2. Fluxo de eventos

- 1 – Administrador entra na interface de administração.
- 2 - O administrador seleciona a ação relacionada ao grupo.
- 3 - O sistema verifica os dados enviados e realiza a função.

3. Extensões

2a – Subfluxo - Administrador quer alterar grupo

1. O administrador lista os grupos (INCLUDE Listar Grupos)
2. O administrador seleciona a opção de editar grupo.
3. O administrador entra com os novos dados (Nome, descrição) para o grupo.
4. Sistema verifica os dados enviados e registra a alteração.

2b – Subfluxo - Administrador quer excluir um grupo

1. O administrador lista os grupos (INCLUDE Listar Grupos)
2. O administrador seleciona a opção de excluir grupo.
3. Sistema verifica os dados e realiza a exclusão

2c – Subfluxo - Administrador quer incluir um grupo

1. Administrador requisita a inclusão de um novo grupo.
2. Administrador envia o nome e a descrição do grupo a ser incluído.
3. Sistema verifica os dados enviados e inclui o grupo.

2c.a – Grupo já existe

1. O sistema retorna um erro.

Gerenciar produto

1. Geral

| | |
|----------------------------|--|
| ID do Caso de Uso | 103 |
| Nome do Caso de Uso | Gerenciar produto |
| Objetivo | Incluir/Excluir/Alterar uma produto no sistema |
| Descrição | Administrador entra com os dados de um produto no sistema e o gerencia |
| Atores | Administrador |
| Pré-condições | Ter grupos e marcas já registradas |
| Pós-condições | O produto é cadastrado/excluído/alterado |
| Inclusões/Extensões | Listar Produtos |

2. Fluxo de eventos

- 1 – Administrador entra na interface de administração.
- 2 - O administrador seleciona a ação relacionada ao produto.
- 3 - O sistema verifica os dados enviados e realiza a função.

3. Extensões

2a – Subfluxo - Administrador quer alterar produto

1. O administrador lista os produtos (INCLUDE Listar Produtos)
2. O administrador seleciona a opção de editar produto.
3. O administrador entra com os novos dados (Nome, descrição, preço, tamanho, peso, imagem, marca e grupo) para o produto.
4. Sistema verifica os dados enviados e registra a alteração.

2b – Subfluxo - Administrador quer excluir um produto

1. O administrador lista os grupos (INCLUDE Listar Produtos)
2. O administrador seleciona a opção de excluir produto.
3. Sistema verifica os dados e realiza a exclusão

2c – Subfluxo - Administrador quer incluir um produto

1. Administrador requisita a inclusão de um novo produto.
2. Administrador envia o nome, a descrição, o preço, o tamanho, o peso, a imagem, a marca e o grupo do produto a ser incluído.
3. Sistema verifica os dados enviados e inclui o produto.

2c.a – Produto já existe

1. O sistema retorna um erro.

Casos do módulo "Busca"

Pesquisar Produto

1. Geral

| | |
|----------------------------|---|
| ID do Caso de Uso | 201 |
| Nome do Caso de Uso | Pesquisar Produto |
| Objetivo | Localizar produtos e visualizar suas informações. |
| Descrição | Cliente entra com dados do produto e visualiza suas informações |
| Atores | Cliente |
| Pré-condições | Produtos cadastrados |
| Pós-condições | Exibe lista de produtos |
| Inclusões/Extensões | |

2. Fluxo de eventos

- 1 – Cliente requer uma pesquisa de produto.
- 2 - O Cliente entra com os dados do produto (nome, grupo, marca) a ser pesquisado.
- 3 - O Sistema verifica a existência dos produtos.
- 4 – O sistema exibe a lista com o nome, imagem, preço e a opção de incluir no carrinho os produtos encontrados.

3. Extensões

- 3a – Sistema não encontra produtos
 - Sistema retorna mensagem de erro de não encontrado

Listar Marcas

1. Geral

| | |
|----------------------------|--|
| ID do Caso de Uso | 202 |
| Nome do Caso de Uso | Listar Marca |
| Objetivo | Lista as marcas cadastradas no sistema |
| Descrição | Administrador pede uma lista de marcas no sistema e o sistema a provê. |
| Atores | Administrador |
| Pré-condições | Existirem marcas cadastradas |
| Pós-condições | Exibir lista de Marcas |
| Inclusões/Extensões | |

2. Fluxo de eventos

- 1 – O administrador requer uma lista de marcas cadastradas no sistema.
- 2 – O sistema verifica as marcas existentes.
- 3- O sistema exibe lista com os nomes das marcas existentes com as opções de edição e remoção das mesmas.

Listar Grupos

1. Geral

| | |
|----------------------------|---|
| ID do Caso de Uso | 203 |
| Nome do Caso de Uso | Listar Grupos |
| Objetivo | Listar grupos cadastrados no sistema |
| Descrição | Administrador pede uma lista de grupos no sistema e o sistema o provê |
| Atores | Administrador |
| Pré-condições | Existirem grupos cadastrados |
| Pós-condições | Exibir lista de grupos |
| Inclusões/Extensões | |

2. Fluxo de eventos

- 1 – O administrador requisita uma lista de grupos cadastrados no sistema
- 2 – O sistema verifica os grupos existentes
- 3 - O sistema exibe lista com o nome dos grupos encontrados além das opções de edição e remoção dos mesmos.

Listar Produtos

1. Geral

| | |
|----------------------------|---|
| ID do Caso de Uso | 204 |
| Nome do Caso de Uso | Listar Produtos |
| Objetivo | Listar produtos cadastrados no sistema |
| Descrição | Administrador pede uma lista de produtos no sistema e o sistema o provê |
| Atores | Administrador |
| Pré-condições | Existirem produtos cadastrados |
| Pós-condições | Exibir lista de produtos |
| Inclusões/Extensões | |

2. Fluxo de eventos

- 1 – O administrador requisita uma lista de produtos cadastrados no sistema
- 2 – O sistema verifica os produtos existentes
- 3 - O sistema exibe lista com o nome, descrição e preço dos produtos encontrados além das opções de edição e remoção dos mesmos.

Casos do módulo "Pedidos"

Gerenciar Carrinho

1. Geral

| | |
|----------------------------|--|
| ID do Caso de Uso | 301 |
| Nome do Caso de Uso | Incluir Produto no Carrinho |
| Objetivo | Incluir/Exclui um produto no carrinho ou visualiza os produtos do carrinho |
| Descrição | Cliente gerencia seu carrinho |
| Atores | Cliente |
| Pré-condições | Produtos cadastrados |
| Pós-condições | Produto incluído/excluído no carrinho |
| Inclusões/Extensões | Pesquisar produtos |

2. Fluxo de eventos

- 1 – O Cliente pesquisa um produto (INCLUDE Pesquisar Produtos)
- 2 - O Cliente seleciona os produtos retornados na lista que deseja adicionar no carrinho.
- 3 – O Sistema registra os produtos no carrinho.

3. Extensões

1a. Uso Alternativo - Cliente deseja visualizar Carrinho

1. Cliente requisita a visualização do carrinho.
2. O sistema retorna as informações (nome, quantidade, preço, e opção de retirar o produto do carrinho) dos produtos que estão no carrinho.

1b. Uso Alternativo - Cliente deseja excluir produto do carrinho

1. Cliente requisita a visualização do carrinho
2. O sistema retorna as informações (nome, quantidade, preço, e opção de retirar o produto do carrinho) dos produtos que estão no carrinho.
3. Cliente seleciona a exclusão de um dos produtos.
4. Produtos são excluídos

Finalizar Pedido

1. Geral

| | |
|----------------------------|---|
| ID do Caso de Uso | 302 |
| Nome do Caso de Uso | Finalizar pedido |
| Objetivo | Finalizar o pedido no sistema |
| Descrição | Cliente visualiza seu carrinho, confirma os dados e o sistema registra o pedido |
| Atores | Cliente |
| Pré-condições | Existem produtos no carrinho |
| Pós-condições | O Pedido é registrado |
| Inclusões/Extensões | Visualizar Carrinho |

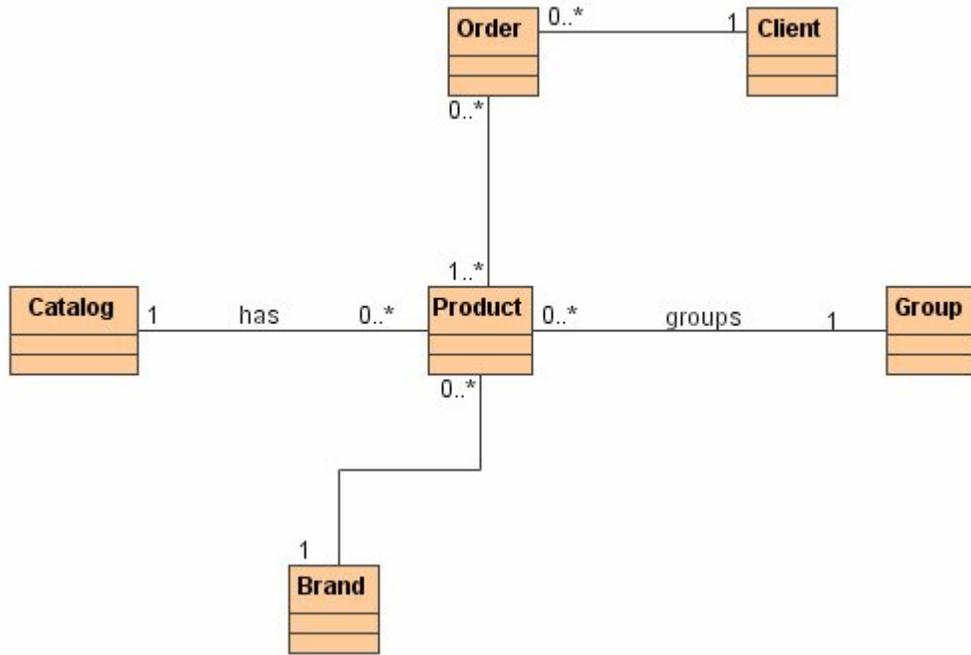
2. Fluxo de eventos

- 1 - O Cliente requisita o fechamento do pedido.
- 2 - O sistema exibe os dados do carrinho. (INCLUDE Visualizar carrinho)
- 3 - O sistema exibe tela de dados de entrega.
- 4 - O Cliente preenche os dados de entrega e os confirma.
- 5 - O sistema confirma os dados e exibe dados de forma de pagamento.
- 6 - O sistema registra o pedido e finaliza o carrinho.

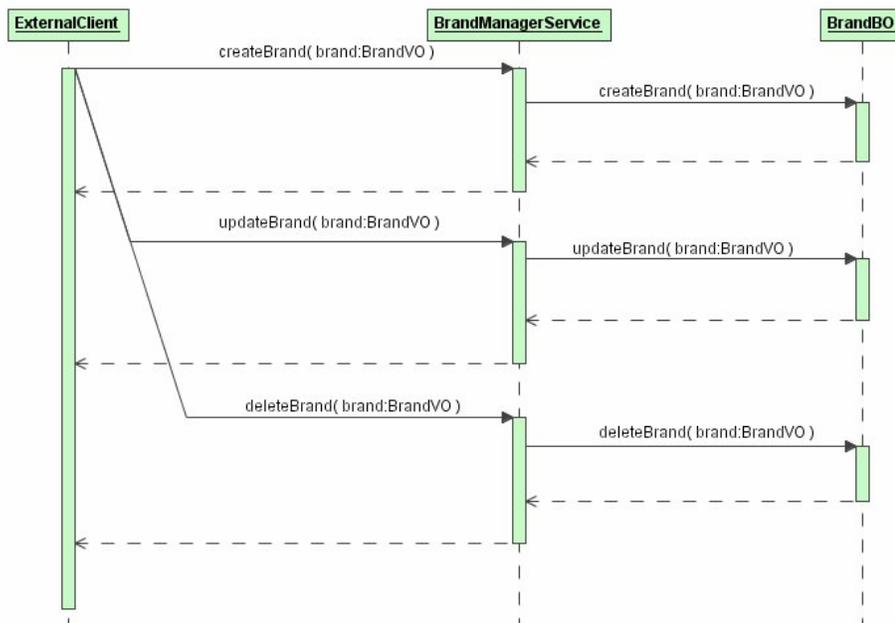
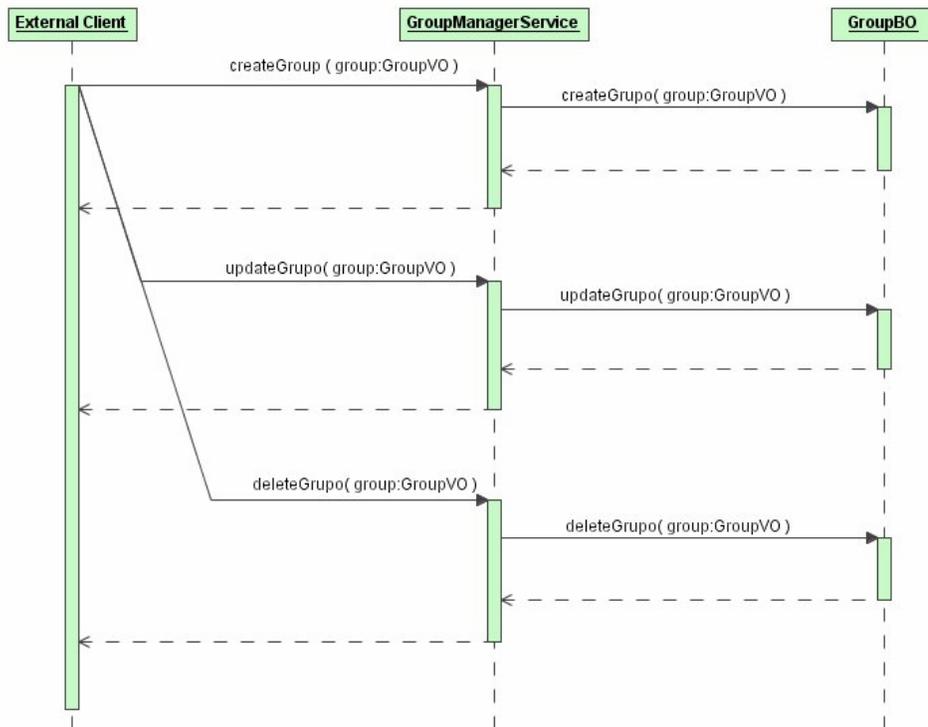
3. Extensões

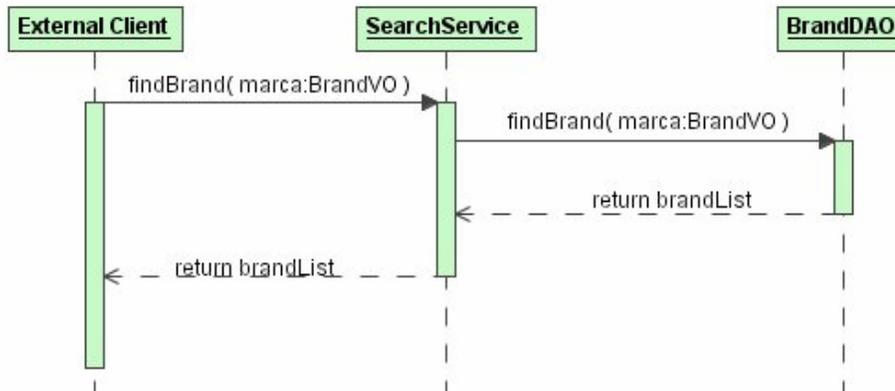
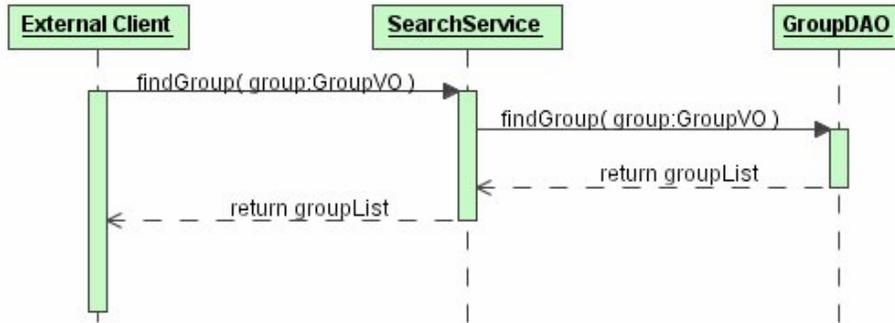
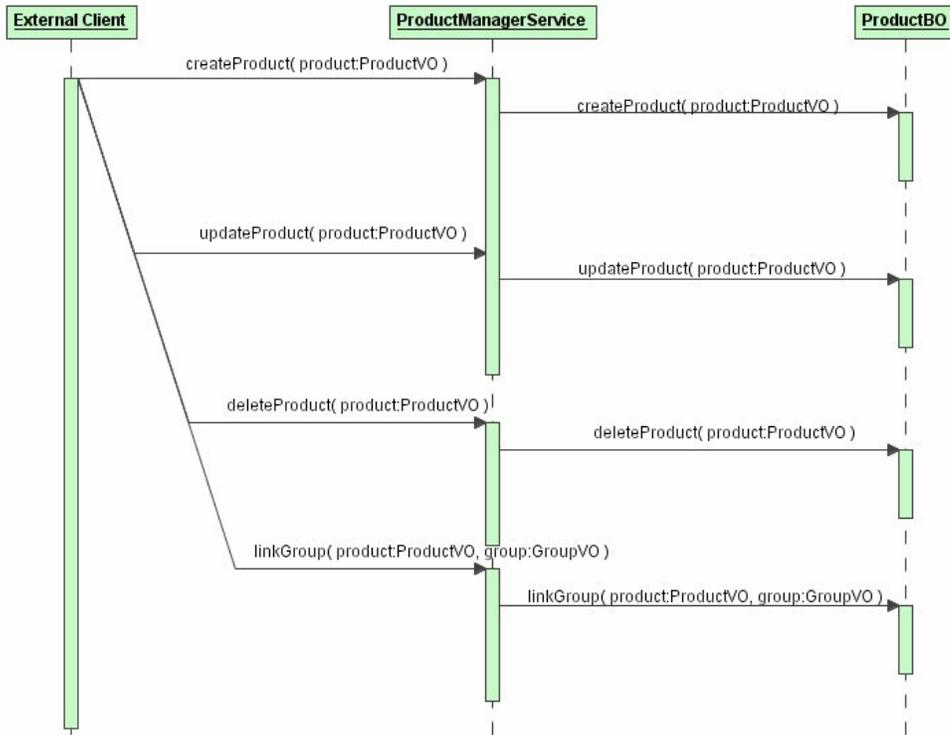
- 5a - Dados de entrega incompletos
- 1 . O sistema retorna uma mensagem de erro.
 - 2 . O sistema retorna a tela de dados de entrega.
- 6a - Dados de pagamento incompletos
- 1 . O sistema retorna uma mensagem de erro.

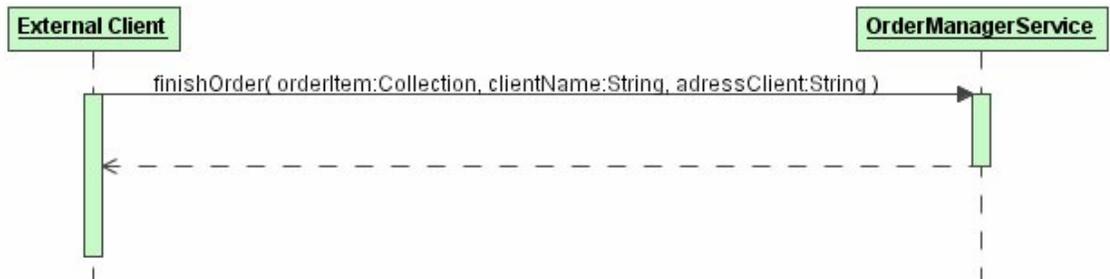
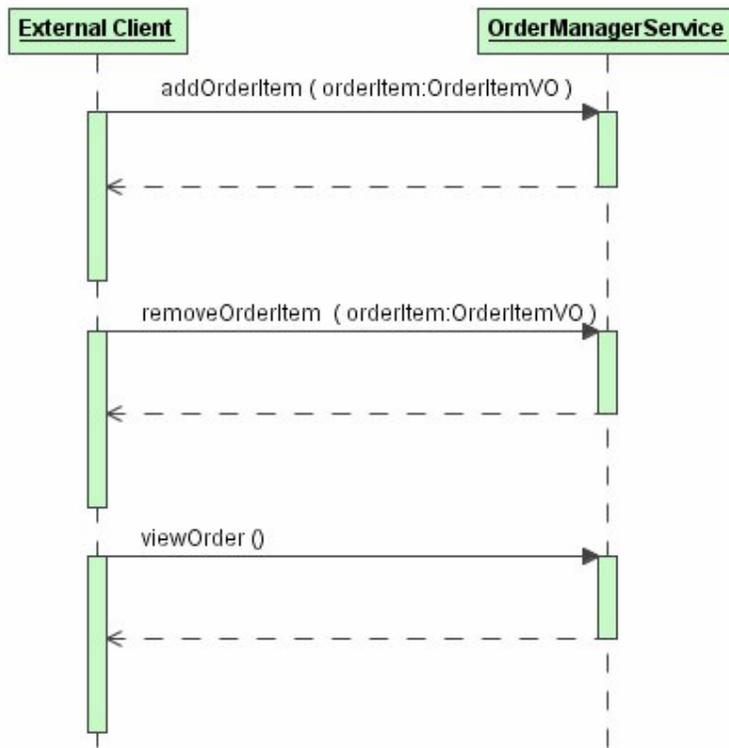
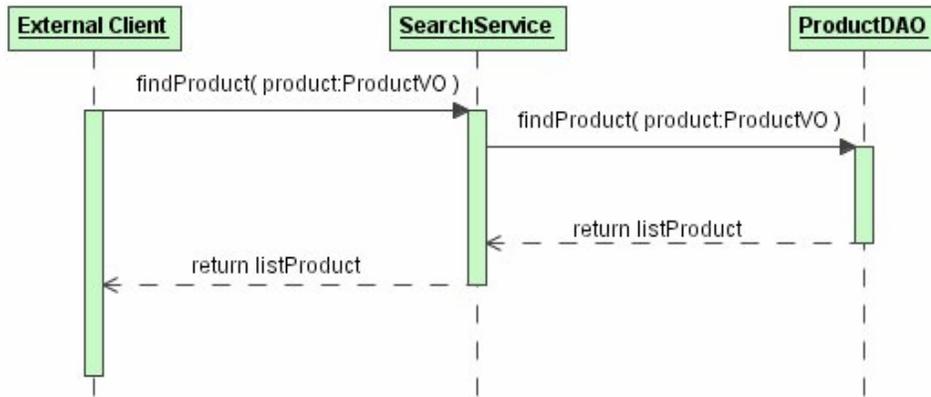
APÊNDICE E – MODELO DE DOMÍNIO

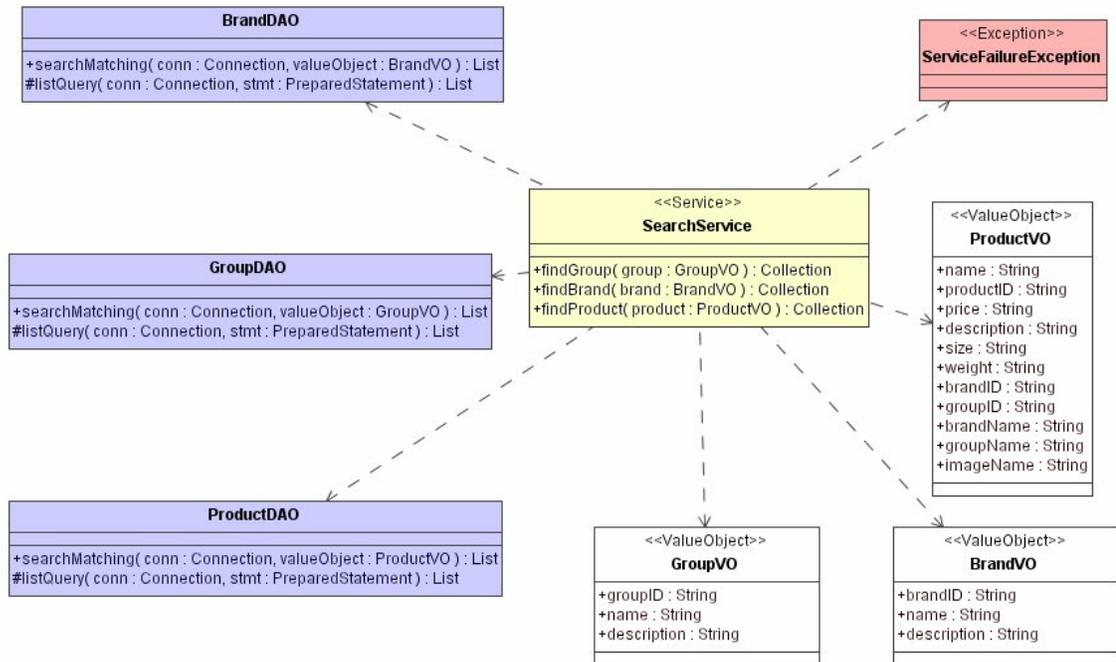
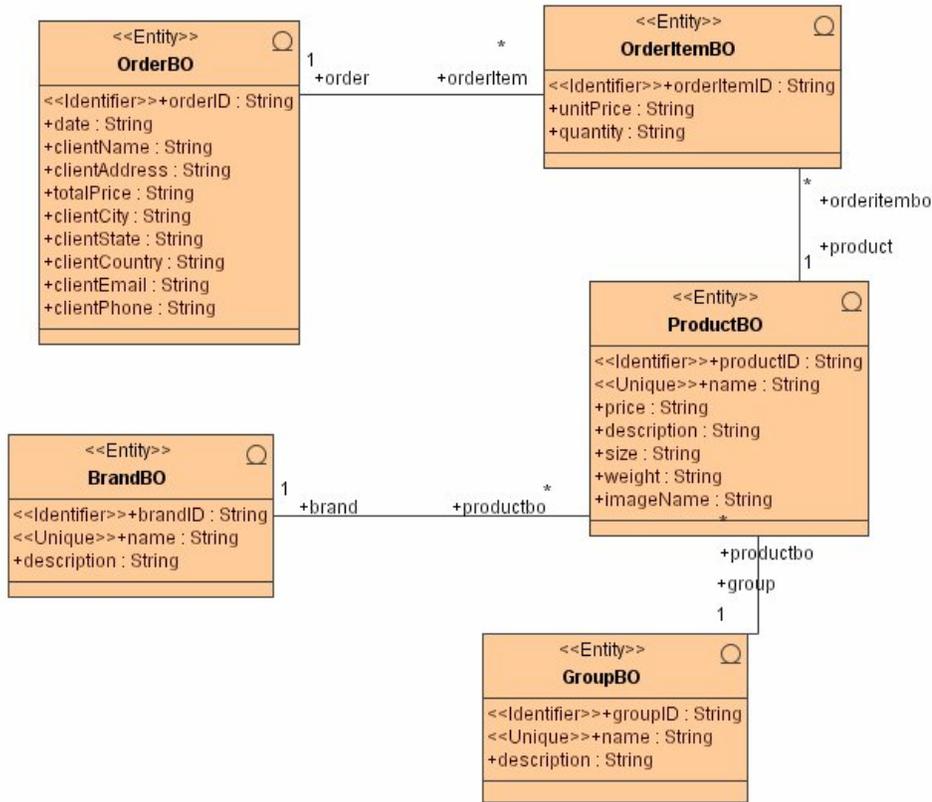


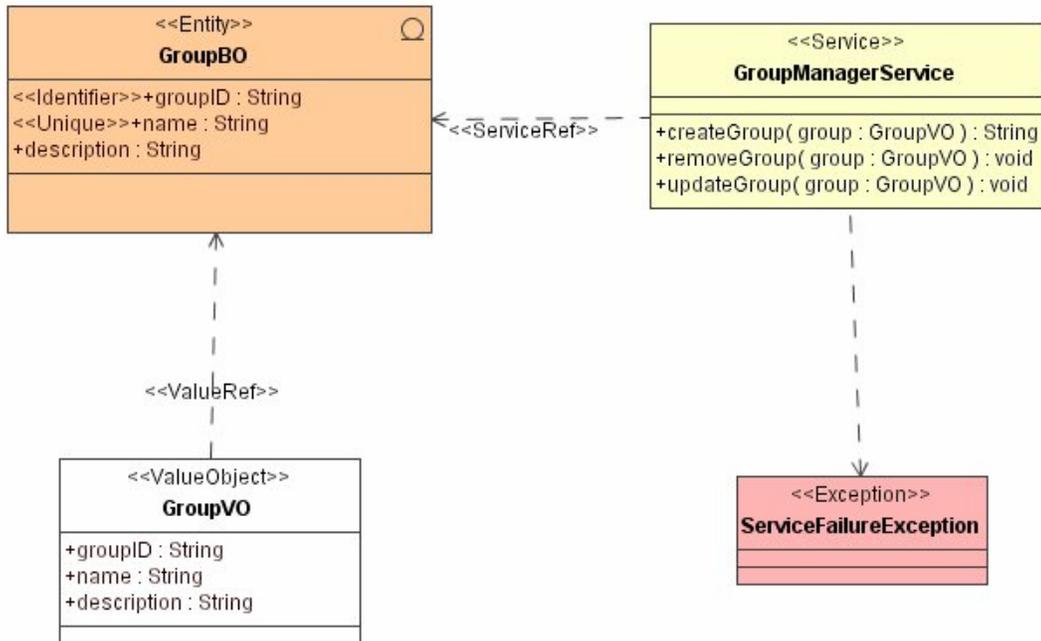
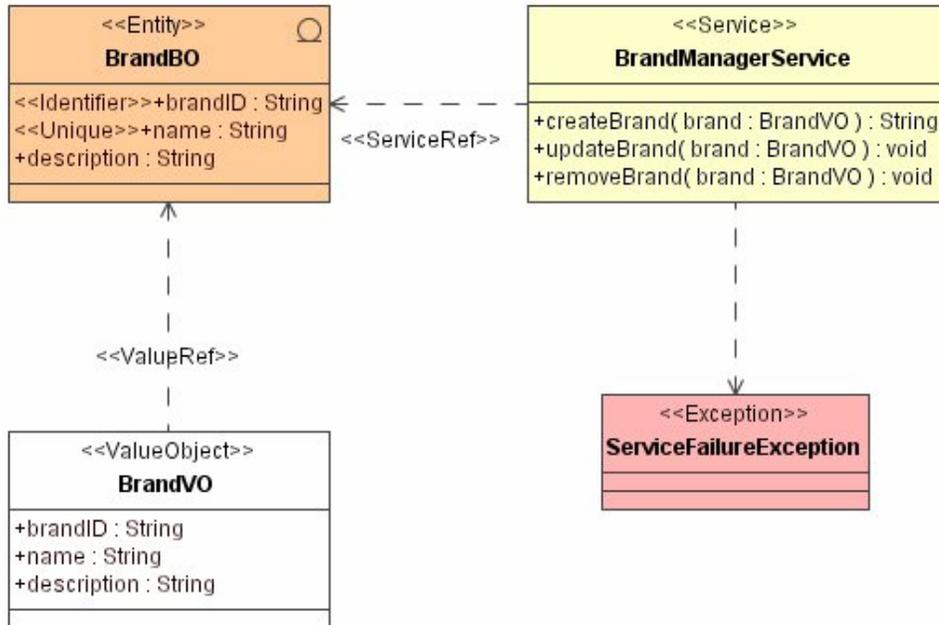
APÊNDICE F – MODELO DE PROJETO

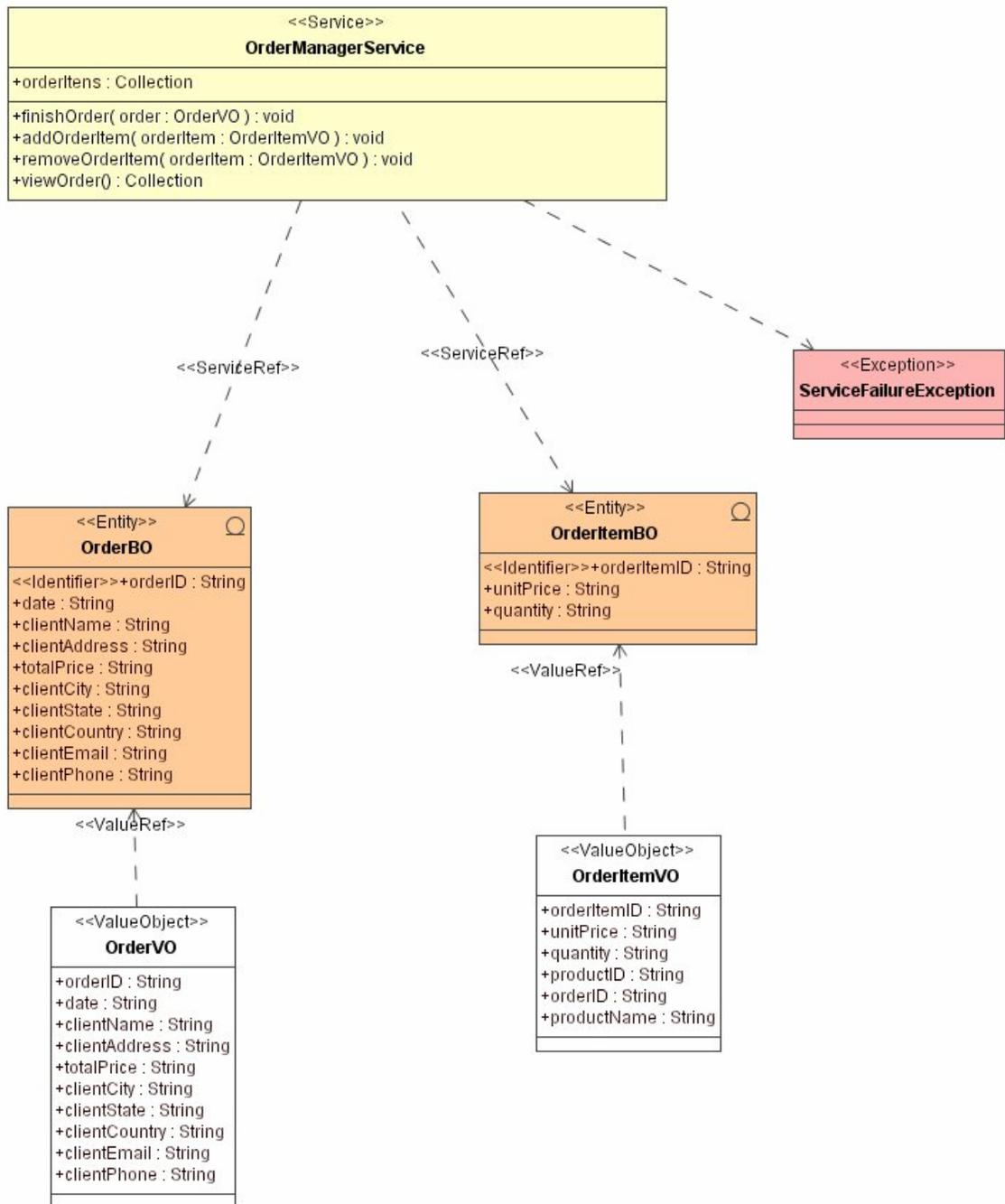


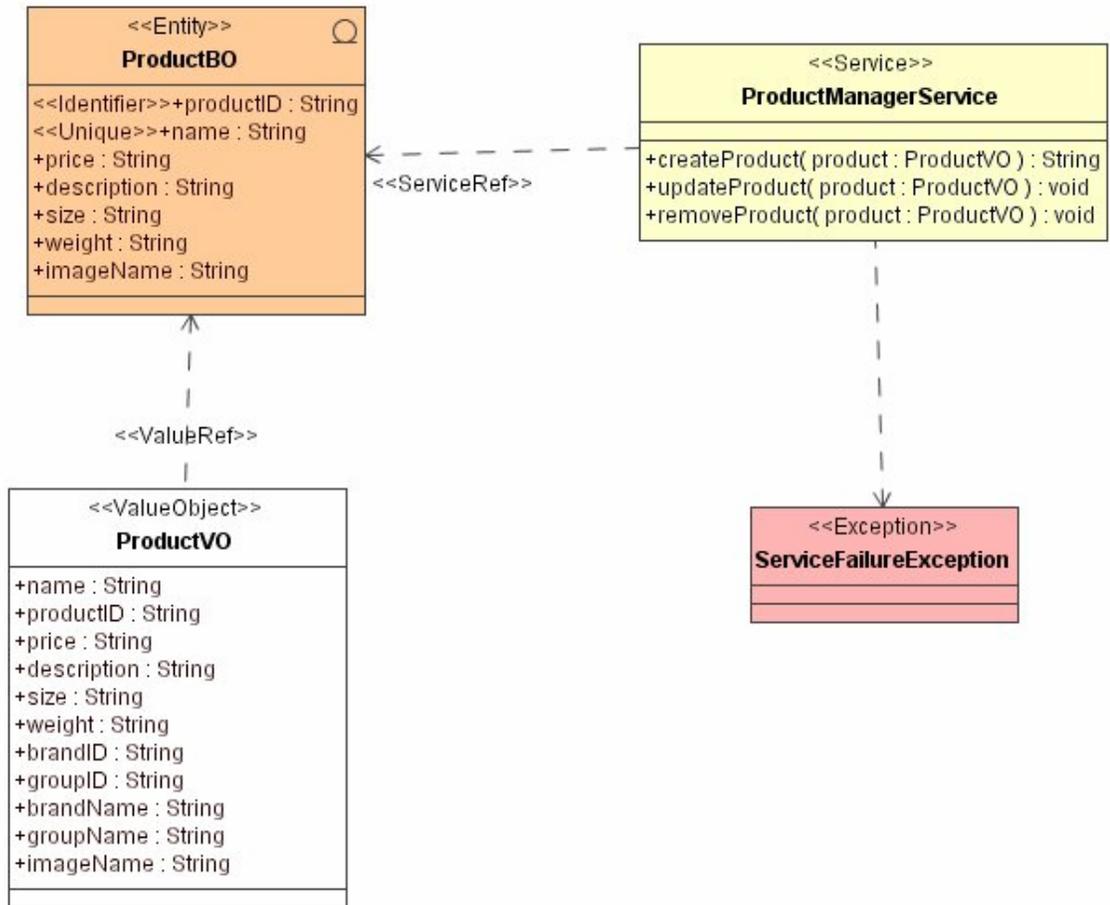












BIBLIOGRAFIA

- [AndroMDA06] AndroMDA, 2006. *AndroMDA(Andromeda)*. Disponível em <<http://andromda.org>> acesso em: fevereiro/março 2006.
- [OMG06] Object Management Group, 2006. *MDA – Model Driven Architecture*. Disponível em <<http://www.omg.org/mda>> acesso em: fevereiro/março 2006.
- [J2EEP06] Java Platform, Enterprise Edition (J2EE - Java EE). Disponível em <<http://java.sun.com/javaee/>> acesso em fevereiro/abril de 2006.
- [Struts05] Apache Struts Project – The Apache Software Foudation. Disponível em <<http://struts.apache.org/>> acesso em Janeiro/março de 2006.
- [Kruchten00] Kruchten, P. *The Rational Unified Process – Na Introduction*. 2nd Edition. Reading, MA.: Addison-Wesley, 2000.
- [Larman02] Larman, Craig. *Utilizando UML e Padrões – Uma introdução à análise e ao projeto orientados a objetos e ao Processo Unificado*. 2.ed. Bookman. 2002.

- [JBR99] Jacobson, I.,Booch, G., and Rumbaugh, J. The Unified Software Development Process. Reading, MA.: Addison-Wesley, 1999.
- [OMG01] Object Management Group, 2001. OMG Unified Modeling Language Specification. Disponível em <<http://www.omg.org>> acesso em: fevereiro/março 2006.
- [Marinescu04] Marinescu, Floyd. Trad, Acauan Fernandes. Padrões de e Projeto EJB. Porto Alegre: Bookman, 2004.
- [ACM04] Alur, D., Crupi, J., Malks, D. Core J2EE Paterns: as melhores práticas e estratégias de design. Rio de Janeiro: Elsevier(Campus), 2004.