UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANDRÉ KRZYK TARAS

BRUNO VICHINHESKI

MODELAGEM DE UMA FERRAMENTA PARA GERAÇÃO AUTOMÁTICA DE E-COMMERCE B2B E B2C USANDO DDD

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2016

ANDRÉ KRZYK TARAS

BRUNO VICHINHESKI

MODELAGEM DE UMA FERRAMENTA PARA GERAÇÃO AUTOMÁTICA DE E-COMMERCE B2B E B2C USANDO DDD

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná.

Orientadora: Prof(a). Dra. Simone Nasser Matos

PONTA GROSSA

2016

TERMO DE APROVAÇÃO

Modelagem de uma Ferramenta para Geração Automática de *e-Commerce* B2B e B2C Usando DDD

por

ANDRÉ KRZYK TARAS
BRUNO VICHINHESKI

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 20 de Maio de 2016 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Profª Drª Simone Nasser Matos
Orientadora

_____
Prof. MSc. Vinicius Camargo Andrade
Membro titular

_____
Prof. Dr. Augusto Foronda
Responsável pelos Trabalhos
de Conclusão de Curso

_____
Prof. MSc Alessandro Luiz Stamatto
Membro titular

_____
Prof. Dr Erikson Freitas de Morais
Coordenador do curso

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

À nossa professora, pela paciência na orientação e pelo incentivo ao longo da realização deste trabalho.

AGRADECIMENTOS

A esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbramos um horizonte superior.

Agradecemos à nossa orientadora Profª. Drª. Simone Nasser Matos, pela sabedoria com que nos guiou nesta trajetória.

Aos colegas de sala.

Aos amigos companheiros de trabalhos e irmãos na amizade que fizeram parte de nossa formação e que continuarão presentes em nossas vidas com certeza.

Aos pais, pelo amor, incentivo e apoio incondicional.

Enfim, a todos os que por algum motivo contribuíram para a realização deste trabalho.

Entusiasmo, apoiado por bom senso e
persistência é a qualidade que mais
frequentemente contribui para o sucesso.
(CARNEGIE, Dale)

RESUMO

TARAS, André krzyk. VICHINHESKI, Bruno. **Modelagem de uma Ferramenta para Geração Automática de *e-Commerce* B2B e B2C Usando DDD**. 2016. 95 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2016.

O comércio eletrônico (*e-commerce*) está sendo utilizado por milhões de pessoas e podem atender categorias, tais como: empresa para empresa (B2B), consumidores (B2C) ou governos (G2G). Existem plataformas para criação de *e-commerce*, porém atendem apenas uma categoria. Este trabalho propõe a modelagem dos módulos fundamentais que compõem uma plataforma *e-commerce* nas categorias B2B (*Business To Business*) e B2C (*Business To Customers*) utilizando o conceito de *Domain-Drive Design* (DDD), como também a implementação dos módulos de cadastro e login de usuários e catálogo de produtos. O padrão de desenvolvimento DDD foi escolhido pela necessidade da compreensão do problema como um todo, pela alta manutenabilidade e pela capacidade de solucionar problemas intrínsecos nos sistemas de *e-commerce* atuais. O modelo proposto não se limita a uma categoria, proporcionando uma melhor reutilização de código e também faz com que cada etapa desenvolvida não seja dependente de outros recursos existentes no sistema.

Palavras-chave: B2B. B2C. DDD. Modelagem. Plataforma *e-commerce*.

ABSTRACT

TARAS, André krzyk. VICHINHESKI, Bruno. Modeling a tool for automatic generation of e-commerce B2B and B2C Using DDD. 2016. 95 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2016.

E-commerce is being used by millions of people and it is divided by categories: Business To Business (B2B), Business To Customers (B2C) or Government To Government (G2G). There are platforms used to create e-commerce, however it's only destined to one category. This final paper proposes the modeling of main modules that compose an e-commerce in the B2B and B2C categories using the DDD concept; as well as the code development for the user's register and login, and products catalog. The DDD pattern was chose to understand the complexity problem, maintainability and ability to solve specific problems in the current e-commerce. The suggested model is not limited to a category, providing a better code reuse and each developed step is not dependent on the system resources.

Keywords: B2B. B2C. DDD. Modeling. E-commerce.

LISTA DE ILUSTRAÇÕES

Figura 1 - Componentes de um Sistema *E-commerce* 16
Figura 2 - Mapa conceitual das categorias de *e-commerce* 17
Figura 3 - Características do B2B 19
Figura 4 - Características B2C 20
Figura 5 - Especificações do Produto IBM 23
Figura 6 - Cadastro de usuários no *website* da IBM 23
Figura 7 - Cadastro / Login de Usuário – Categoria B2B 24
Figura 8 - Relacionamento Cliente e Empresa – Categoria B2B 25
Figura 9 - Catálogo de Produtos Para Cada Cliente – Categoria B2B 26
Figura 10 - Transação entre Cliente/Empresa – Categoria B2B 27
Figura 11 - Processo de Entrega – Categoria B2B 28
Figura 12 - Especificações do Produto Ponto Frio 29
Figura 13 - Cadastro de Usuário Ponto Frio 30
Figura 14 - Cadastro / Login de Usuário – Categoria B2C 31
Figura 15 - Relacionamento Cliente e Empresa – Categoria B2C 31
Figura 16 - Catálogo de Produtos Apresentados aos Clientes 32
Figura 17 - Transação entre Cliente/Empresa – Categoria B2C 33
Figura 18 - Processo de Entrega – Categoria B2C 33
Figura 19 - Faturamento Anual *e-commerce* Brasileiro em bilhões de Reais 34
Figura 20 - Faturamento Anual *e-commerce* Mundial em bilhões de Dólares 35
Figura 21 - Modelo inicial cálculo de frete 40
Figura 22 - Código das operações de frete 40
Figura 23 - Modelo com aplicação do padrão *Strategy* para o cálculo do frete 42
Figura 24 - Código após a aplicação do padrão *Strategy* para o cálculo do frete 42
Figura 25 - Camada de Domínio 45
Figura 26 - *Breakthrough* 48
Figura 27 - Mapa de Contextos 50
Figura 28 - Núcleo Compartilhado 51
Figura 29 - Camada Anti-corrupção 52
Figura 30 - Primeiros elementos identificados no cadastro e *login* de usuários 56
Figura 31 - Primeiros elementos identificados do catálogo de produtos 57
Figura 32 - Criando um projeto *Symfony* 59
Figura 33 - Instalação do Composer 59
Figura 34 - Execução do *Symfony* no *localhost:8000/* 60
Figura 35 - Estrutura do framework *Symfony* depois de instalado 60
Figura 36 - Biblioteca doctrine e twig 61
Figura 37 - Arquivos da camada interface em *html.twig* 62
Figura 38 - Arquivos da camada interface em PHP 63
Figura 39 - Arquivo TWIG 63
Figura 40 - UserType 64
Figura 41 - Arquivos da camada de aplicação 65

Figura 42 - LoginPessoaController 65
Figura 43 - Arquivos de entidades de domínio 66
Figura 44 - Entidade Pessoa 67
Figura 45 - Serviço ProdutoManager 67
Figura 46 - Modelo responsável por gerar a aplicação *e-commerce* nas categorias B2B ou B2C 68
Figura 47 - Repositório *ProdutoRepository* 69
Figura 48 - Assinatura de um Repositório em uma Entidade 69
Figura 49 - Modelo Inicial Cadastro e *Login* de Usuários 71
Figura 50 - Modelo Final Cadastro e *Login* de Usuários 71
Figura 51 - Modelo Inicial Catálogo de Produtos 73
Figura 52 - Modelo Final Catálogo de Produtos 73
Figura 53 - Página Inicial da ferramenta 75
Figura 54 - Página para escolha dos módulos para o e-commerce 76
Figura 55 - Página de Cadastro de Usuário 77
Figura 56 - Página de Cadastro Pessoa Física 78
Figura 57 - Página Formulário de *Login* 79
Figura 58 - Página do Cadastro de Produtos 80
Figura 59 - Visualização do Catálogo de Produtos B2B 81
Figura 60 - Visualização do Catálogo de Produtos B2C 82
Figura 61 - Visualização da Busca B2B 82
Figura 62 - Visualização da Busca B2C 83
Figura 63 - Modelo Relacionamento Cliente/Empresa 92
Figura 64 - Modelo Transação e Entrega 93

LISTA DE QUADROS

Quadro 1 - Comparativo elementos B2B vs B2C 22
Quadro 2 - Ranking de Plataformas *e-commerce* mais usadas 36
Quadro 3 - Módulos e Funções Desempenhadas para B2B e B2C 55
Quadro 4 - Módulos e Refatorações 74
Quadro 5 - Comparação de funcionalidades 84

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
B2B	*Business-to-Business*
B2C	*Business-to-Customer*
B2G	*Business-to-Government*
C2C	*Customer-to-Customer*
C2G	*Customer-to-Government*
CEP	Código de Endereçamento Postal
CLTV	*Customer Lifetime Value*
DDD	*Domain-Driven Design*
G2G	*Government-to-Government*
GRIS	Taxa de Gerenciamento de Risco
HTML	*HyperText Markup Language*
IBM	*International Business Machines*
ICMS	Imposto sobre Operações relativas à Circulação de Mercadorias e Prestação de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação
IPI	Imposto sobre Produtos Industrializados
SEBRAE	Serviço de Apoio às Micro e Pequenas Empresas
ST	Substituição Tributária
TAS	Taxa de Administração da Secretária da Fazenda
TRT	Taxa de Restrição de Trânsito
TV	Televisão
UML	*Unified Modeling Language*

LISTA DE ACRÔNIMOS

E-commerce	Electronic Commerce
Mercosul	Mercado Comum do Sul
PagSeguro	Pagamento Seguro

SUMÁRIO

1 INTRODUÇÃO 14
1.1 OBJETIVOS 15
1.2 ORGANIZAÇÃO DO TRABALHO 15
2 *E-COMMERCE*: UMA VISÃO GERAL 16
2.1 DEFINIÇÃO E CATEGORIAS DE *E-COMMERCE* 16
2.2 BUSINESS TO BUSINESS (B2B) 19
2.3 BUSINESS TO CUSTOMERS (B2C) 20
2.4 APLICAÇÃO E FLUXO DE PROCESSO B2B 22
2.5 APLICAÇÕES E FLUXO DE PROCESSOS B2C 28
2.6 ANÁLISE DO CENÁRIO DE B2B E B2C A NÍVEL BRASILEIRO E MUNDIAL 34
2.7 FERRAMENTAS DE DESENVOLVIMENTO *E-COMMERCE* 35
3 DOMAIN-DRIVE DESIGN (DDD) 38
3.1 VISÃO GERAL SOBRE O DDD 38
3.2 TRABALHAR O MODELO DO DOMÍNIO 39
3.2.1 Design Consistente 39
3.2.2 Modelo Baseado em Comunicação 43
3.3 CONSTRUIR O MODELO BASEADO EM DOMÍNIO 44
3.3.1 Camada de Domínio 46
3.4 REFATORAR O MODELO 47
3.5 PROJETO ESTRATÉGICO 49
4 MODELAGEM DE UMA FERRAMENTA PARA GERAR *WEBSITE E-COMMERCE* B2B E B2C BASEADA EM DDD 54
4.1 TRABALHAR O MODELO DO DOMÍNIO 54
4.2 CONSTRUIR O MODELO BASEADO EM DOMÍNIO 57
4.2.1 Camada de Infraestrutura 58
4.2.2 Camada de Interface 62
4.2.3 Camada de Aplicação 64
4.2.4 Camada de Domínio 66
4.3 REFATORAR O MODELO 70
5 RESULTADOS 75
5.1 IMPLEMENTAÇÃO DOS MÓDULOS: CADASTRO E LOGIN DE USUÁRIOS E CATÁLOGO DE PRODUTOS 75
5.1.1 Cadastro de Usuários 76
5.1.2 Login de Usuários 79
5.1.3 Cadastro de Produtos 79
5.1.4 Catálogo de Produtos 81
5.2 UMA BREVE COMPARAÇÃO DO MODELO PROPOSTO E TRABALHOS RELACIONADOS 83
5.3 VANTAGENS NO USO DO DDD 84

6 CONCLUSÃO.....86
6.1 TRABALHOS FUTUROS.....86
REFERÊNCIAS.....88
APÊNDICE A: MODELO RELACIONAMENTO CLIENTE/EMPRESA 92
APÊNDICE B: MODELO TRANSAÇÃO E ENTREGA.....93

1 INTRODUÇÃO

Uma plataforma *e-commerce* é responsável por realizar transações de compra e venda de produtos e serviços por meio da Internet. Essa plataforma possibilita a interação entre compradores e vendedores de diferentes distâncias (ASCENSÃO, 2015).

O sistema *e-commerce* possui um alto crescimento a cada ano; no Brasil o faturamento desse comércio eletrônico chegou a R$48,5 bilhões em 2015 e para 2016 a estimativa é que alcance R$56,8 bilhões (DOTSTORE, 2016).

A rápida evolução da tecnologia e do comércio eletrônico, fez com que muitas empresas desenvolvessem plataformas *e-commerce* para seus clientes aumentarem a competitividade de venda de seus produtos. Porém, muitas das plataformas *e-commerce* existentes são desenvolvidas sem grande planejamento. Essa falta pode causar uma alta demanda em manutenção e com isso a empresa sofre um grande prejuízo para a correção de tais defeitos.

A utilização de plataformas *e-commerce* faz com que apareça a necessidade de criar um produto com baixo custo de manutenção, que possa acompanhar as constantes novidades das tecnologias empregadas no seu desenvolvimento de forma eficaz e com isso possa proporcionar um preço compatível com a necessidade de cada empresa.

Para abranger essas finalidades este trabalho estudou os tipos de plataformas *e-commerce* existentes, as quais são divididas em cinco categorias: B2B (*Business To Business*), B2C (*Business To Customers*), B2G (*Business To Governments*), G2G (*Governments To Governments*) e C2C (*Customers To Customers*), para criação de um modelo que possa atender tais categorias. Neste trabalho foram contempladas as categorias mais usadas: B2B e B2C (QIN, 2009).

O modelo proposto foi desenvolvido usando a abordagem *Domain-Drive Design* (DDD) proposto por Evans, a qual possui foco no domínio de um sistema e tem por objetivo conceitualizá-lo em forma de um modelo representando o conhecimento do funcionamento de um negócio (EVANS, 2003).

Foram criados os modelos para os módulos cadastro e *login* de usuários; relacionamento existente entre o cliente e a empresa; listagem do catálogo de produtos; transações e entrega das categorias B2B e B2C. Para validar o modelo foram implementados um módulo que apresenta características comuns entre as

categorias (*cadastro e login de usuários*) e outro que contém alguma similaridade (*catalogo de produtos*).

A implementação dos módulos mostrou que é possível criar um modelo genérico que atenda as categorias B2B e B2C, desta forma disponibilizar aos usuários uma plataforma para geração automatizada de *e-commerce*, não importando qual o tipo de categoria escolhida.

1.1 OBJETIVOS

O objetivo geral deste trabalho é modelar as categorias de *e-commerce* B2B e B2C utilizando a abordagem *Domain Drive Design* (DDD) gerando um modelo adaptável e flexível neste domínio. Os objetivos específicos são:

- Identificar as características fundamentais que diferem as categorias B2B e B2C.
- Desenvolver a modelagem dos processos das categorias B2B e B2C.
- Implementar dois módulos das categorias B2B e B2C.
- Elencar *framework* que pode ajudar na implementação do modelo.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em cinco capítulos. O capítulo 2 apresenta as principais categorias de *e-commerce,* porém com foco maior nas categorias B2B e B2C.

O capítulo 3 apresenta a definição de DDD, bem como suas etapas de desenvolvimento e as camadas que dividem o DDD; também é mostrado cada um dos padrões que compõem esta camada.

O capítulo 4 descreve a aplicação dos conceitos vistos nos capítulos 2 e 3 para a modelagem e implementação da plataforma *e-commerce* proposta.

O capítulo 5 apresenta os resultados obtidos e também é feita uma breve comparação da ferramenta criada com outras já existentes. Por fim, o capítulo 6 relata as considerações finais deste trabalho e os trabalhos futuros que podem ser realizados a partir desta pesquisa.

2 *E-COMMERCE*: UMA VISÃO GERAL

Este capítulo apresenta os conceitos importantes sobre *e-commerce* para o desenvolvimento desta pesquisa. A seção 2.1 relata o que é um sistema *e-commerce*, seus elementos e suas categorias. Nas seções 2.2 e 2.3 é descrito, respectivamente, as principais diferenças entre as categorias de *e-commerce* B2B (*Business To Business*) e B2C (*Business To Customers*). As seções 2.4 e 2.5 apresentam os principais fluxos de processo para B2B e B2C, bem como exemplo de sua aplicação.

2.1 DEFINIÇÃO E CATEGORIAS DE *E-COMMERCE*

O comércio eletrônico (*e-commerce*) expande o comércio que era realizado sem o auxílio da tecnologia da informação e é responsável por movimentar parte da economia mundial (ALBERTIN; MOURA, 2002).

Este sistema propõe uma maneira fácil de se realizar transações entre clientes e empresas por meio de um dispositivo eletrônico; porém cada empresa possui uma definição diferente para o termo *e-commerce*, mas todas as categorias têm o mesmo propósito, que é o comércio realizado por meio de uma rede (QIN, 2009). Os componentes de um *e-commerce* são ilustrados na Figura 1.

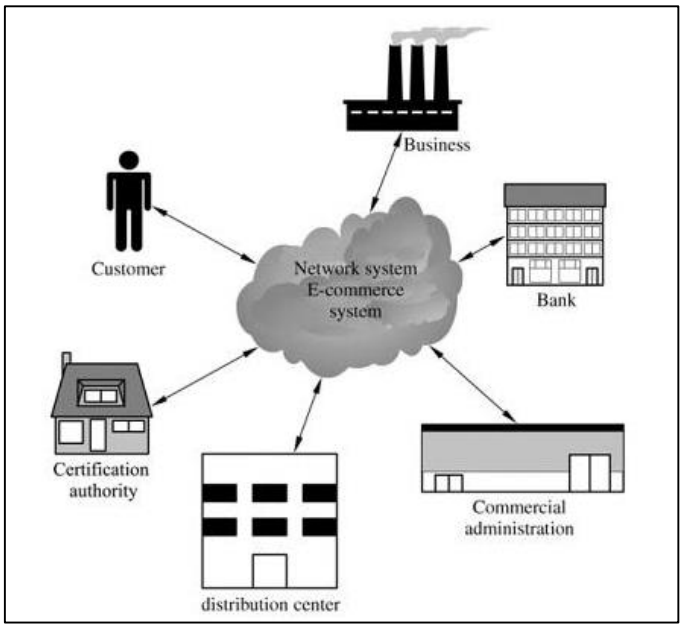

Figura 1 - Componentes de um Sistema *E-commerce*
Fonte: Qin (2009)

A finalidade de cada elemento é (QIN, 2009):

- *Network system E-commerce system:* é a rede responsável por fazer a ligação de todos os elementos e por tratar das transações comerciais.
- *Customer* e *Business*: é o usuário *e-commerce*, ou seja, é o cliente que está conectado à Internet disposto a realizar uma transação.
- *Certification authority*: é a organização responsável por verificar a integridade de um *website*.
- *Distribution center*: é a empresa responsável pela entrega do produto, quando o mesmo é algo físico.
- *Bank*: responsável por permitir meios de pagamento entre compradores e vendedores.
- *Commercial Administration*: responsável por concretizar a negociação, provendo informações, envio e atendimento ao cliente.

O *e-commerce* pode ser dividido em cinco diferentes categorias quando se considera o tipo de transação (QIN, 2009): B2B (*Business To Business*), B2C (*Business To Customers*), B2G (*Business To Governments*), G2G (*Governments To Governments*) e C2C (*Customers To Customers*), conforme é ilustrado na Figura 2.

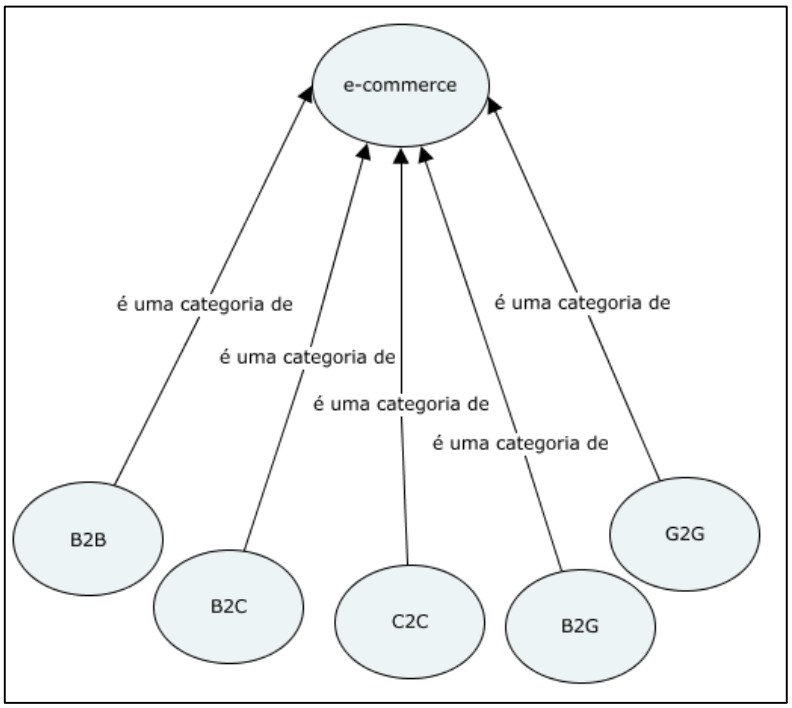

Figura 2 – Mapa conceitual das categorias de *e-commerce*
Fonte: Autoria própria

B2B (*Business To Business*) é voltado para transações entre empresas. As características desta categoria são: vendas diretas e suporte comercial, como por exemplo, a empresa Cisco no qual é vendido um serviço e disponibilizado *downloads* e suporte *online*; compradores com objetivo de negociar preços por serviços e *websites* de informação responsáveis por mostrar características de determinada empresa (TASSABEHJI, 2003).

B2C (*Business To Customers*) propõe o contato entre empresa e consumidor, por meio da Internet, *websites* colocam seus serviços a venda e consumidores realizam as transações por algum meio de pagamento estabelecido pelo *website*. B2C é a categoria que possui mais destaque e está em constante crescimento pois novas empresas e clientes procuram novas oportunidades nesse mercado (QIN, 2009).

B2G (*Business To Governments*) possui como características principais o espaço virtual destinado a empresa e o governo para realizarem um determinado projeto, coordenar reuniões *online* e gerenciar o seu progresso. Nesta categoria há também o aluguel de plataformas e de banco de dados com o propósito apenas de uso governamental (TASSABEHJI, 2003).

G2G (*Governments To Governments*) é uma categoria onde se têm relação apenas entre governos, os quais usam este meio para realizar transações com governos locais e internacionais, como por exemplo, a União Europeia que começou a desenvolver estratégias para conectar os diferentes sistemas nacionais (TASSABEHJI, 2003).

C2C (*Customers To Customers*) tem como objetivo o contato entre consumidores para que os mesmos façam negociações sem interferência direta de uma empresa. Consumidores trocam informações, compram e vendem produtos através de uma plataforma *e-commerce* específica. O *website Ebay* é um exemplo de *e-commerce* nessa categoria, em que qualquer pessoa está disposta a criar um cadastro e vender algum produto pelo preço que deseja (TASSABEHJI, 2003).

Este trabalho está focado nas duas principais categorias com maior destaque na literatura: *Business To Business* (B2B) e *Business To Customers* (B2C), que estão voltados na criação de *websites* para negócios e clientes, respectivamente; sendo a categoria B2B com uma receita de vendas maior que a B2C (MENDES, 2015). Estas categorias serão mais detalhadas nas próximas seções.

2.2 BUSINESS TO BUSINESS (B2B)

Plataformas B2B possuem um número de clientes menor quando comparado com a B2C; pois sua finalidade é detalhar seu produto de venda e a relação com que tem com a empresa que fará negócio (SILVA, 2013). Podem-se citar como exemplos de plataformas B2B aquelas que possuem produtos específicos para determinadas áreas e são vendidos em atacado, como por exemplo, equipamentos médicos, químicos, industriais, tecnológicos (SILVA, 2013); os quais serão negociados para consumo da empresa, revenda ou insumo na sua produção (BONIFACIO, 2015).

Apesar desta categoria possuir poucos clientes, esta possui a maior quantidade de vendas. Suas vendas são estimadas em no mínimo o dobro quando comparado com a categoria B2C (BONIFACIO, 2015). A Figura 3 apresenta os detalhes fundamentais sobre o que um *website* B2B objetiva.

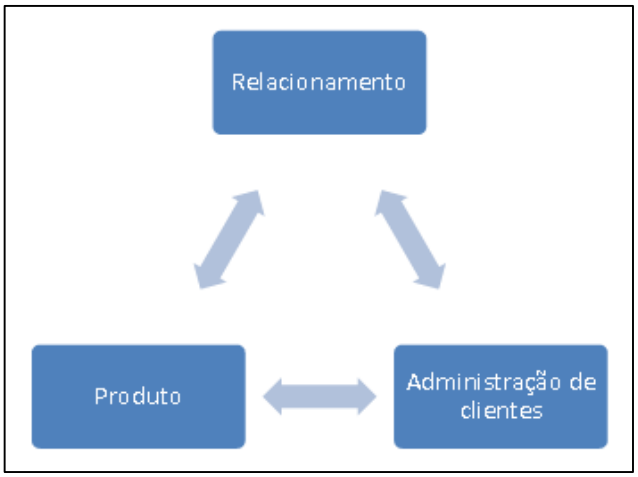

Figura 3- Características do B2B
Fonte: Silva (2013)

O relacionamento é um forte aspecto na relação de vendas entre as empresas. Representantes e executivos possuem uma boa interação para que negócios sejam concretizados. Os produtos possuem descrições técnicas responsáveis por responder todas as questões que a empresa solicitante da compra precisa. Neste caso, não há uma vasta quantidade de clientes, porém uma vez efetivado o negócio, a empresa possui uma forte ligação com o cliente, prestando suporte disponível para a aquisição de novos produtos. Na maioria das vezes, as vendas realizadas em *website* B2B são seguidos de contratos, então a valorização do cliente se torna significativa para futuras negociações.

Operações B2B são aquelas com foco em clientes que sejam outras empresas, como por exemplo, revenda para distribuidor e lojista, produtos para consumo da empresa (material de limpeza, papel), bens para outra empresa (móveis, carros, manutenção, reparo) (BONIFACIO, 2015).

Na categoria B2B a análise *Customer Life Time Value* (CLTV) é a principal característica para que as negociações possam ocorrer, pois o vendedor quer ter um relacionamento de longo prazo com a empresa cliente (BONIFACIO, 2015).

Nesta categoria técnicas de *marketing*, como *banners* em *websites* populares, mídias e TV não são aplicadas. O objetivo é encontrar os clientes certos e não uma grande quantidade (BONIFACIO, 2015).

O preço final de determinada venda B2B depende do produto em negociação, a quantidade que está sendo adquirida e também do relacionamento com o cliente atual. Empresas B2B possuem tabelas de custos para seus produtos e a margem de desconto dependendo do tipo de negociação realizada. Além disso, o valor final do produto é calculado sobre impostos com a base de cálculo no ICMS, Substituição Tributária (ST) e IPI (BONIFACIO, 2015).

2.3 BUSINESS TO CUSTOMERS (B2C)

Plataformas B2C são destinadas a um grande número de clientes com características diferentes (SILVA, 2013). Empresas trabalham com o número de clientes elevado para que haja maior quantidade de lucro. As vendas ocorrem para o cliente final, na maioria dos casos para pessoas físicas (BONIFACIO, 2015). A Figura 4 apresenta os detalhes fundamentais sobre o que um *website* B2C objetiva.

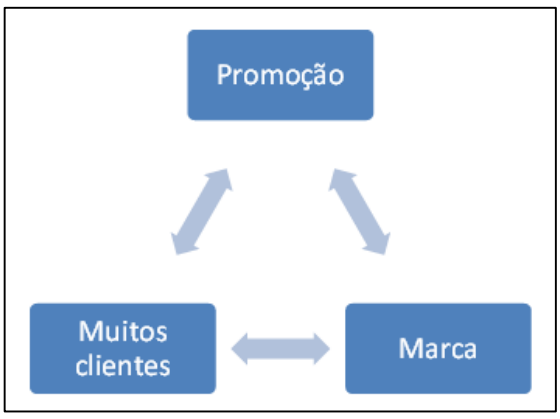

Figura 4 - Características B2C
Fonte: Silva (2013)

Nesta categoria o preço e a divulgação do produto estão fortemente ligados ao número de transações que esta plataforma irá realizar. Devem existir diversas marcas para que haja uma maior seleção perante o consumidor. A cada dia surgem novos clientes e também clientes antigos voltam a comprar quando tiveram uma boa experiência passada.

Operações B2C são aquelas que têm como foco o consumidor final, como por exemplo, venda de eletrônicos, produtos pessoais, serviços para uso do cliente, como ingressos para *shows*, cinema, teatro, passagens, pacotes turísticos, entre outros (BONIFACIO, 2015).

Empresas B2C usam técnicas de *marketing* para atrair maior número de clientes, pois isto faz com que uma maior quantidade de pessoas visite determinado *website* e ocorra uma maior quantidade de vendas (BONIFACIO, 2015).

Nessa categoria o cliente final tem acesso ao mesmo preço informado no *website* sem acréscimo de impostos. Os modelos de pagamento são: cartão de crédito, boleto bancário, débito em conta, *paypal* (maneira facilitada de realizar pagamentos pela Internet, onde foca em garantir pela segurança durante a compra (PAYPAL, 2015)), *pagseguro* (meios de pagamentos mais convenientes disponibilizados pela empresa UOL, a qual facilita a negociação entre compradores e vendedores (PAGSEGURO, 2015)), etc. O produto é despachado apenas depois do pagamento.

O Quadro 1 mostra alguns dos principais critérios apresentados por diferentes autores, os quais foram selecionados e usados para comparar as duas categorias de *e-commerce*, B2B e B2C. Os critérios são: qual o tipo de cliente para cada categoria, com qual frequência as operações são realizadas, o relacionamento que o cliente possui com a empresa, o que a categoria maior proporciona a respeito de seu produto, para o cliente qual a quantidade de itens comprados, se a categoria possui foco em divulgar seus produtos (*marketing*) e como funciona o preço do produto final.

Critério	**B2B**	**B2C**
Cliente	Empresa	Consumidor Final
Operações	Contrato	Ocasional
Relacionamento Empresa/Cliente	Muito	Pouco
Produto	Garantia de Fornecimento, Prazo, Confiança	Marketing e preço
Compra	Maior Quantidade	Produtos Isolados
Marketing	Menor	Maior
Preço Produto	Preço + Imposto – Desconto	Preço apresentado

Quadro 1 - Comparativo elementos B2B vs B2C
Fonte: Autoria própria

O quadro 1 compara algumas das principais diferenças entre as categorias de *e-commerce* B2B e B2C e mostra de forma sintetizada suas respectivas características.

2.4 APLICAÇÃO E FLUXO DE PROCESSO B2B

As dez melhores empresas de vendas *online* no Brasil na categoria B2B presentes atualmente são: B2B Magazine, BahiaExpport, ComprasNet, Dell, IBM Brasil, Martins, Mercado Eletrônico, Mercosul Search, Ofiicenet e SEBRAE (PORTALDACOMUNICACAO, 2015).

Websites na categoria B2B focam em mostrar as especificações dos produtos, sendo que para mais detalhes referentes ao preço, entrega e suporte é necessário entrar em contato com a empresa. A Figura 5 exibe como exemplo um produto do web*site* da IBM. A IBM ficou em primeiro lugar na categoria de *e-commerce* B2B devido sua alta capacidade de gerenciamento de pedidos e seu atendimento aos clientes (EVIGO.COM, 2015).

Figura 5 - Especificações do Produto IBM
Fonte: IBM (2015)

No *website* da IBM, página do produto, observa-se que apresenta apenas algumas especificações de seus produtos e ao lado segue as opções de contato para maiores informações.

Quando o usuário deseja fazer alguma transação com determinada empresa, ele deve fazer o *login* na sua conta. Caso não o possua, deve realizar seu cadastro. Em *website* na categoria B2B há possibilidade do usuário se cadastrar como pessoa física ou jurídica. A Figura 6 exibe um exemplo de cadastro de usuário no *website* da IBM, os campos sobre as informações jurídicas do usuário não são obrigatórios.

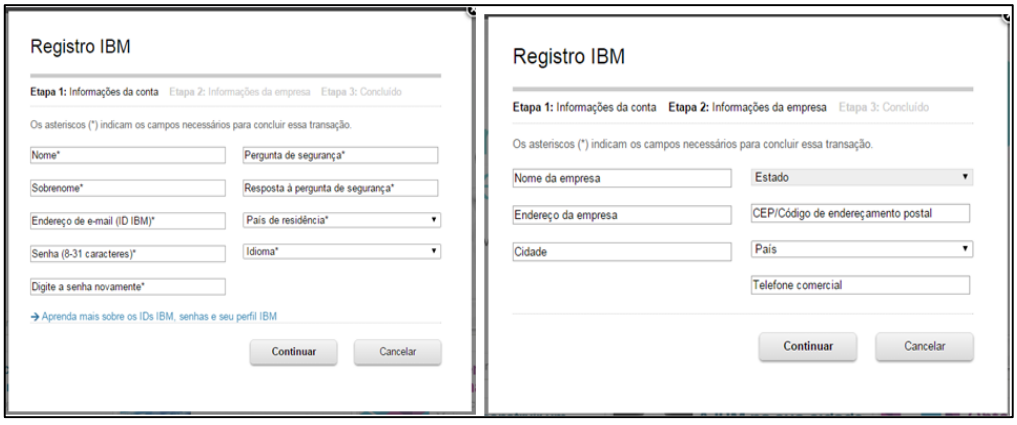

Figura 6 - Cadastro de usuários no *website* da IBM
Fonte: IBM (2015)

Tomando como base o *website* da IBM, Dell, Martins, Mercado Eletrônico, entre outros, pode-se identificar que os principais processos que ocorrem no *website* da categoria B2B são:

a) cadastro e *login* de usuários;
b) relacionamento existente entre o cliente e a empresa;
c) listagem do catálogo de produtos;
d) transações;
e) entrega.

Detalha-se cada um dos processos identificados usando como fonte de referências o conhecimento adquirido na área durante o estágio supervisionado realizado em uma empresa de desenvolvimento *e-commerce* da região e *websites,* tais como: Dell, IBM Brasil, Martins, Mercado Eletrônico, entre outros.

a) Cadastro e login de usuários

Depois de realizado o cadastro, o usuário é colocado em um determinado grupo de usuários. Esses grupos recebem benefícios perante seu histórico de transações com a empresa (BONIFACIO, 2015), como por exemplo, oferecer descontos na compra de alguns produtos.

A Figura 7 mostra o fluxo de processo para cadastro e validação de *login* de usuário.

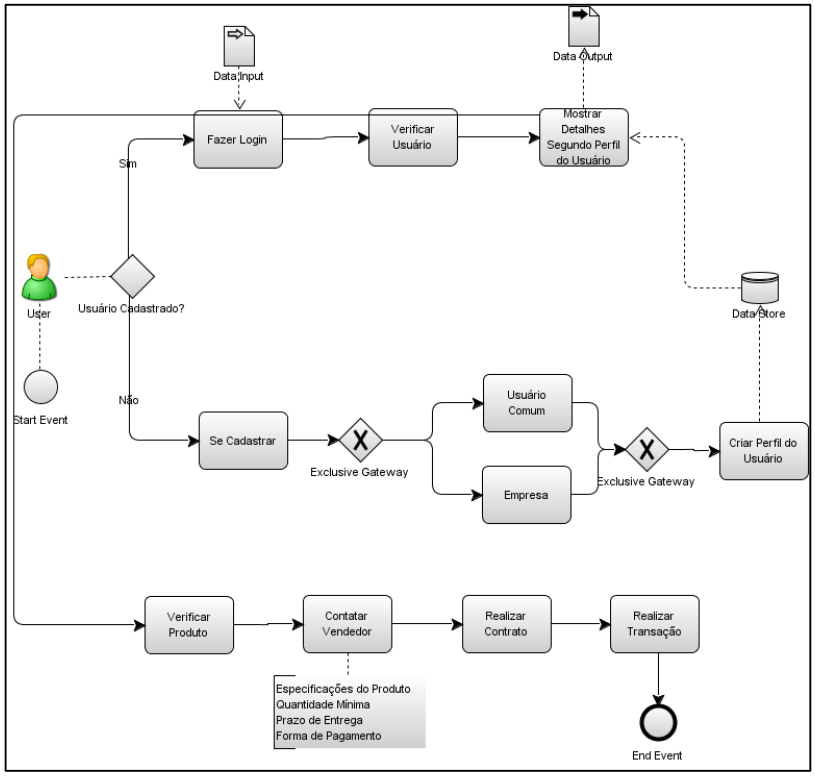

Figura 7 - Cadastro / Login de Usuário – Categoria B2B
Fonte: Autoria própria

O fluxo de processo explica quando um usuário acessa o *website* B2B, no qual ele possui duas opções: fazer *login* em sua conta ou fazer o seu cadastro. Na situação em que o usuário não possui o cadastro, este deve realizá-lo podendo ser: usuário comum ou empresa. Usuário comum é aquele que deseja comprar determinado produto sem ter uma empresa, já no cadastro como empresa, o usuário que está comprando o produto tem como propósito seu uso na instituição.

Após o cadastro, é criado um perfil para o usuário receber sua página de forma customizada de acordo com sua preferência.

b) Relacionamento existente entre o cliente e a empresa

A Figura 8 exibe o fluxo de processos sobre o relacionamento entre cliente e empresa.

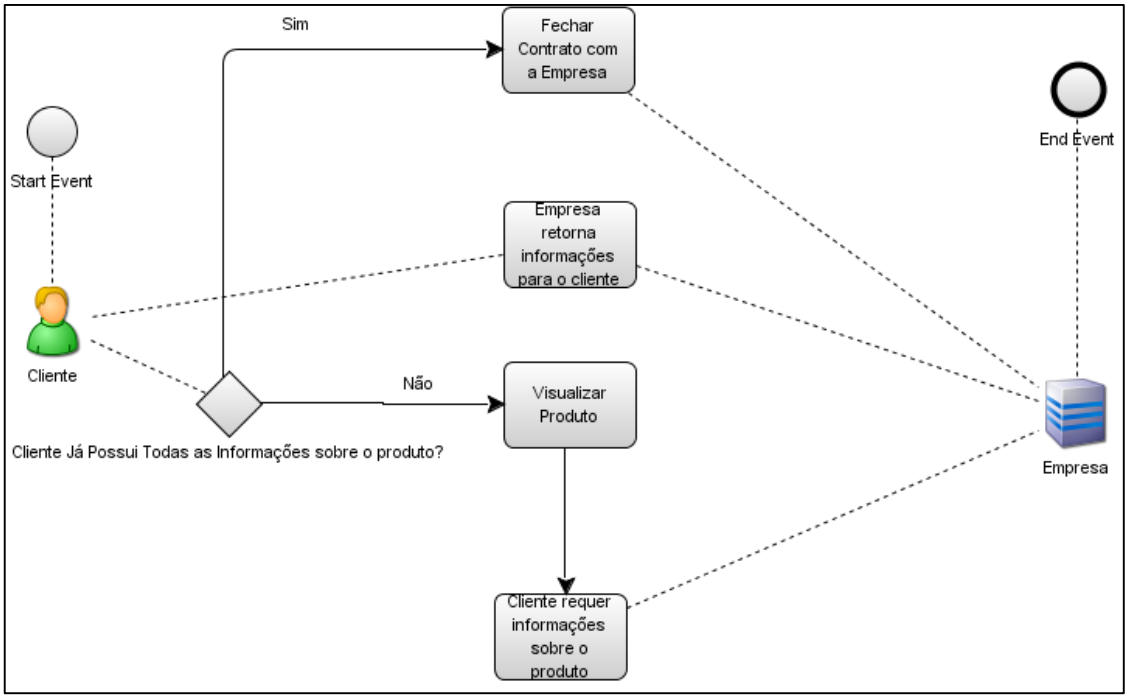

Figura 8 - Relacionamento Cliente e Empresa – Categoria B2B
Fonte: Autoria própria

Após o cliente já possuir um *login*, ele consegue entrar em contato com a empresa. Então, se determinado cliente tem interesse em algum produto, ele irá contatar a empresa para receber informações mais específicas, como por exemplo, quantidade mínima para o pedido, tempo de entrega, meios de pagamento (cartões de crédito, boleto, etc), suporte, entre outros. Este contato ocorre por meios que a

empresa informa no seu *website*, como *email*, telefone, *chat*. Com todas as questões respondidas pela empresa, o cliente pode fechar o contrato com a mesma.

c) Listagem do catálogo de produtos

O fluxo de processos sobre o catálogo de produtos apresentados ao cliente na categoria B2B é exibido na Figura 9.

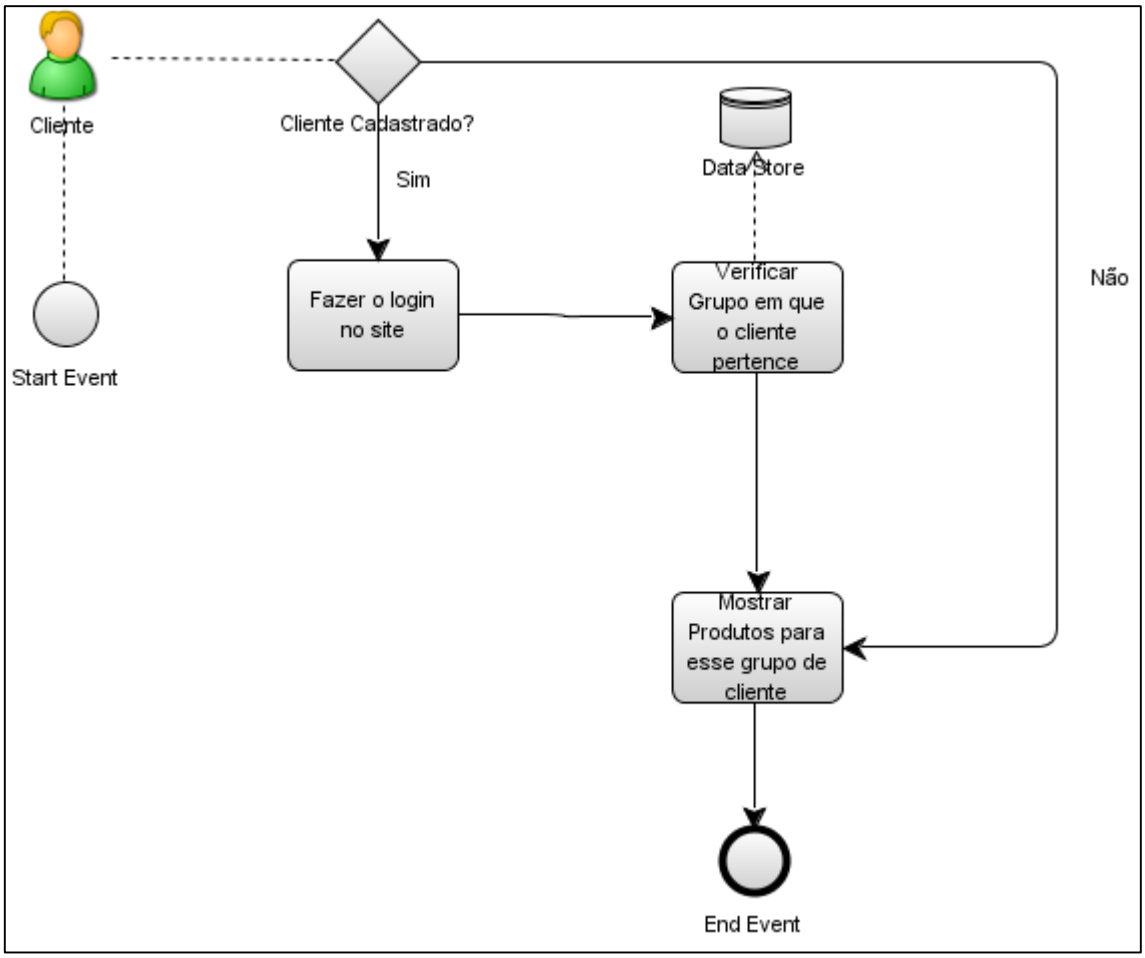

Figura 9 - Catálogo de Produtos Para Cada Cliente – Categoria B2B
Fonte: Autoria própria

Quando um usuário não possui cadastro, este acessa o *website* e é tratado como um usuário comum e os produtos são mostrados de forma geral, sem um determinado filtro de listagem. Caso o usuário já possua um cadastro e realiza o *login* no *website*, os produtos são mostrados conforme a customização de visualizações que ele escolheu ou são exibidos produtos similares à sua última transação. Nessas visualizações de produtos, a empresa pode fornecer alguns valores referentes ao preço final do produto dependendo do grupo em que o usuário se enquadra.

d) Transações

A Figura 10 mostra um fluxo de processos sobre como as transações entre cliente e empresa são realizadas no B2B.

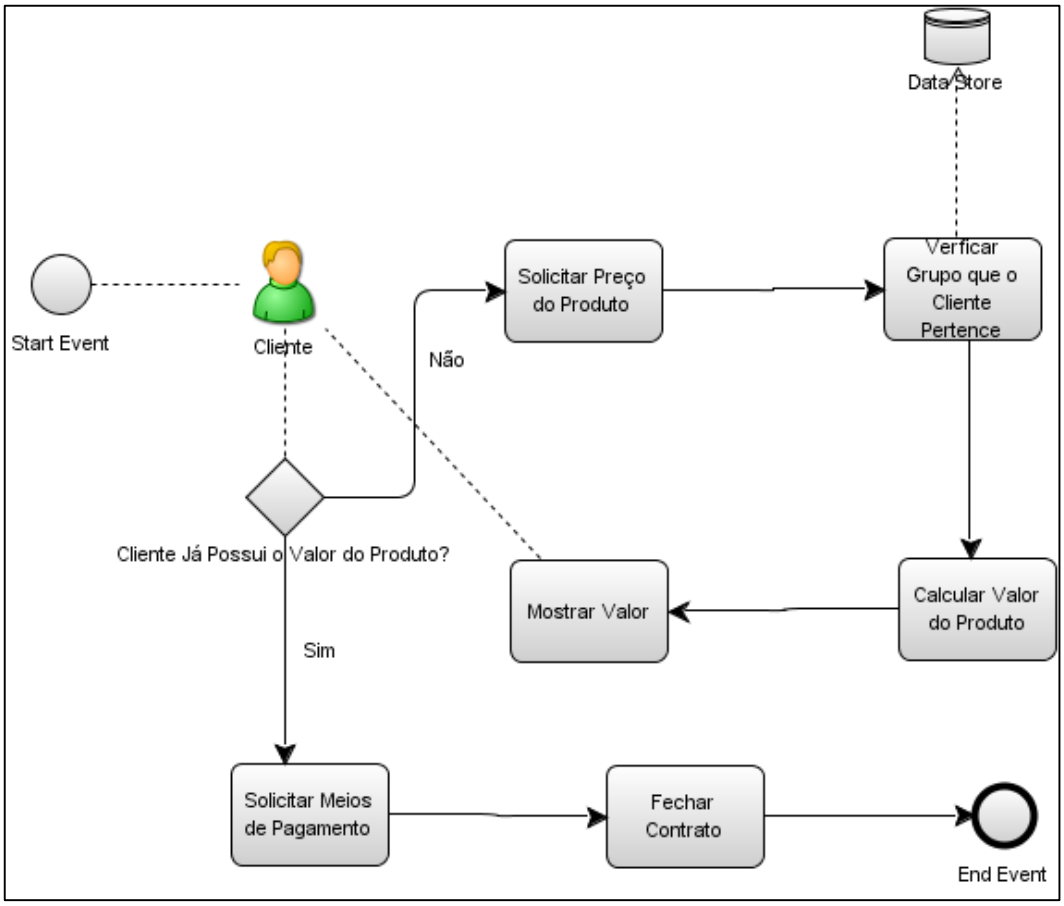

Figura 10 – Transação entre Cliente/Empresa – Categoria B2B
Fonte: Autoria própria

O cliente contata a empresa e solicita informações sobre o valor de determinado produto. Para o cálculo do valor do produto é necessário ver em que grupo o cliente está cadastrado. Verificado a qual grupo o cliente pertence, este pode ter algum benefício, tal como um desconto diferenciado. Então, esse valor é repassado para o cliente. Com o valor final do produto, o cliente solicita e escolhe a maneira com que irá realizar o pagamento. Em seguida, a empresa firma o contrato com o cliente para fechar o processo de transação.

e) *Entrega*

O processo de entrega na categoria B2B segue o fluxo de processo apresentado na Figura 11.

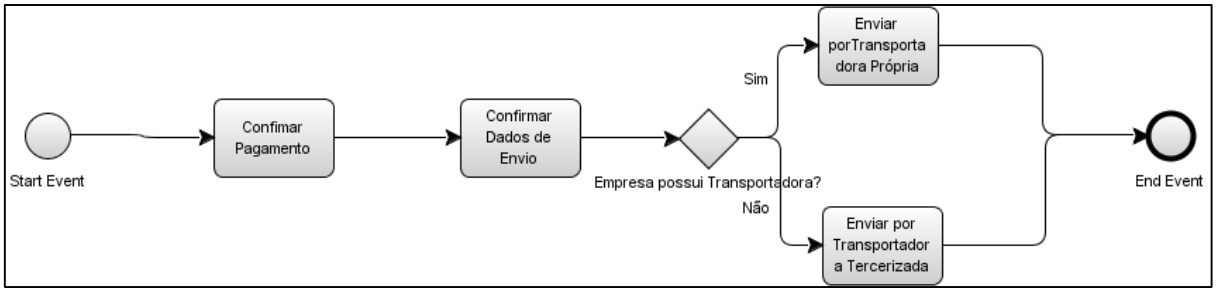

Figura 11 – Processo de Entrega – Categoria B2B
Fonte: Autoria própria

Após a confirmação de pagamento realizado através de um acordo presente no contrato, é confirmado o endereço e a data para envio. Muitas empresas B2B possuem transportadora própria, oferecendo melhor precisão referente ao tempo e qualidade para a entrega. Outras empresas apenas vendem o produto e pagam por uma transportadora especializada em realizar esse tipo de serviço.

O cálculo do valor da entrega do produto é realizado por meio da tabela Rota que possui faixas de *CEP* de origem e destino, bem como faixas de peso e o valor para cada um deles. Essa tabela se relaciona com a de *Transportadora* que irá mostrar ao cliente os tipos de entrega possíveis. Depois do cliente informar o *CEP* para a entrega, é realizado uma busca nessas tabelas de acordo com os itens e o peso do carrinho e será apresentado algumas transportadoras disponíveis, como por exemplo a transportadora que realiza a entrega em menor tempo, a que possui o frete mais barato, entre outras opções.

2.5 APLICAÇÕES E FLUXO DE PROCESSOS B2C

As dez melhores empresas de vendas *online* no Brasil na categoria B2C são: Americanas.com, Dell, Extra, Kalunga, Magazine Luiza, Mercado Livre, Pão de Açúcar, Ponto Frio, Submarino, Supermercado Zona Sul (PORTALDACOMUNICACAO, 2015).

Website na categoria B2C focam em dar destaque no preço de venda de seus produtos, bem como as especificações, os meios de pagamento, a entrega, entre outros, ou seja, tudo o que é necessário para que o cliente compre sem precisar entrar em contato com a empresa. A Figura 12 mostra um exemplo de

página da categoria B2C Ponto Frio, o qual foi um dos destaques durante uma pesquisa de satisfação (E-BIT, 2013).

Figura 12 – Especificações do Produto Ponto Frio
Fonte: Ponto Frio (2015)

No *website* do Ponto Frio, página do produto, é possível observar que apresenta as especificações do mesmo, tais como: preço, quais os meios de pagamento e as formas de entrega disponíveis. Sendo assim, o cliente possui todas as informações necessárias para realizar a compra.

Para a categoria B2C também se identificou os seguintes fluxos de processos:

a) cadastro e *login* de usuários;
b) relacionamento existente entre o cliente e a empresa;
c) listagem do catálogo de produtos;
d) transações;
e) entrega.

Foram criados diferentes fluxos de processos básicos para explicar cada um dos processos usando como fonte de referências o conhecimento adquirido na área durante o estágio supervisionado e *websites*, tais como: Americanas, Dell, Extra, Ponto Frio, entre outros.

a) Cadastro e login de usuários

No momento em que o usuário realiza uma transação, é necessário fazer o *login* em sua conta ou, caso não a possua, deverá fazer um cadastro. Alguns *websites*, como por exemplo, Ponto Frio, Americanas, Dell, entre outros, B2C oferecem duas possibilidades de cadastro: como pessoa física ou jurídica. A Figura 13 ilustra um exemplo de tela de cadastro de usuário.

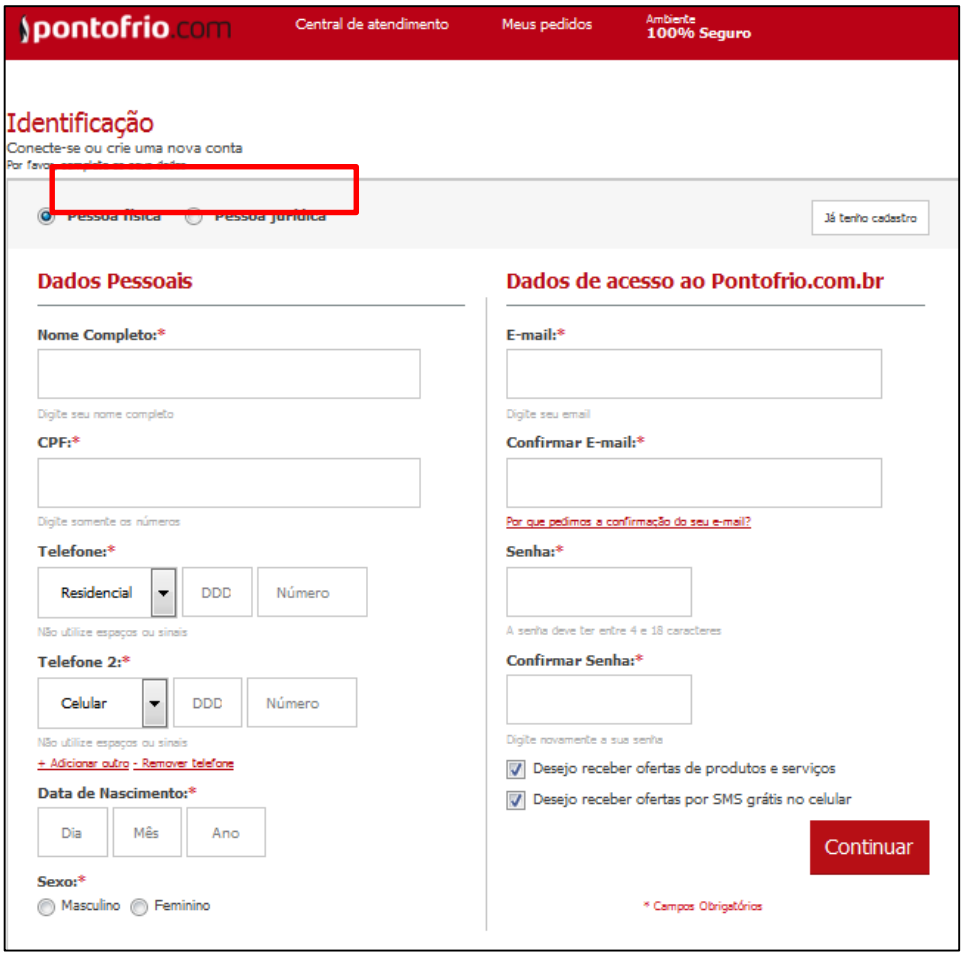

Figura 13 - Cadastro de Usuário Ponto Frio
Fonte: Ponto Frio (2015)

Após realizar o cadastro, os dados fornecidos pelo usuário são armazenados no banco de dados da empresa, para que posteriormente seja realizada uma consulta sobre o histórico dos produtos comprados. A Figura 14 mostra o fluxo de processo para o cadastro/*login* de usuário para a categoria B2C.

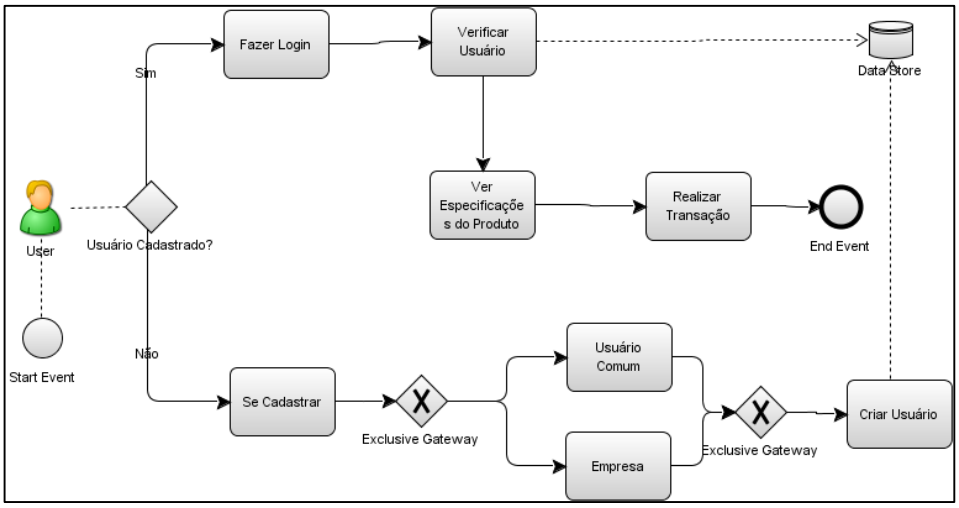

Figura 14 – Cadastro / Login de Usuário – Categoria B2C
Fonte: Autoria própria

A situação de cadastro funciona de forma parecida com a categoria B2B, no qual o usuário possui duas opções de cadastro: usuário comum ou empresa. Porém, depois de realizar o cadastro, é criado um perfil do usuário que não terá distinção, ou seja, não se cria grupos para se oferecer benefícios. Fazendo o *login* o usuário terá acesso a sua conta com as mesmas características dos outros usuários.

b) Relacionamento existente entre o cliente e a empresa

A Figura 15 mostra um fluxo de processos de relacionamento entre cliente e empresa.

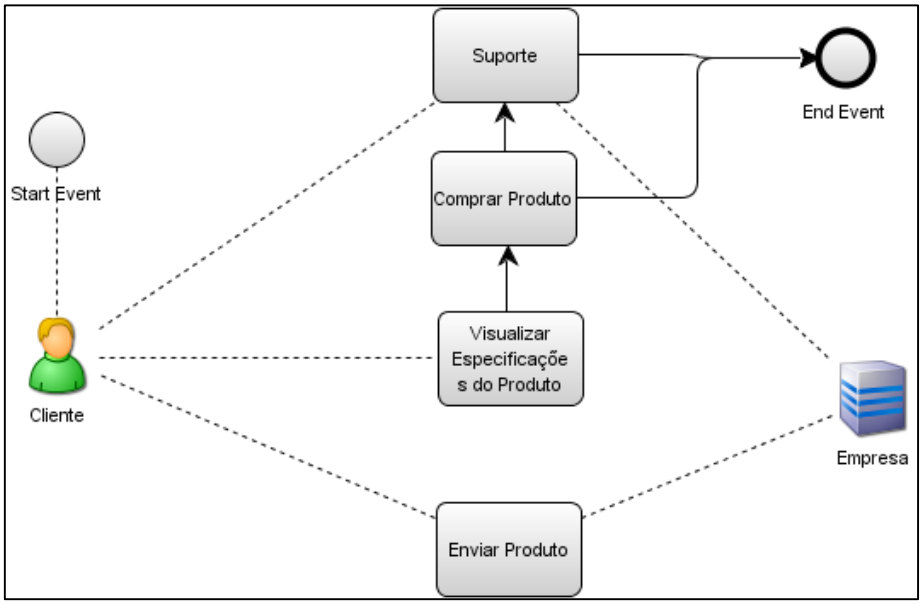

Figura 15 - Relacionamento Cliente e Empresa – Categoria B2C
Fonte: Autoria própria

O fluxo de processos da Figura 15 mostra que o cliente não possui uma forte ligação com a empresa, ele apenas verifica e compra determinado produto e o seu vínculo com a empresa é apenas no momento em que é enviado o produto para o mesmo e caso ele precise de suporte, irá contatar a empresa novamente.

c) Listagem do catálogo de produtos

O processo para exibir o catálogo de produtos aos clientes está ilustrado na Figura 16.

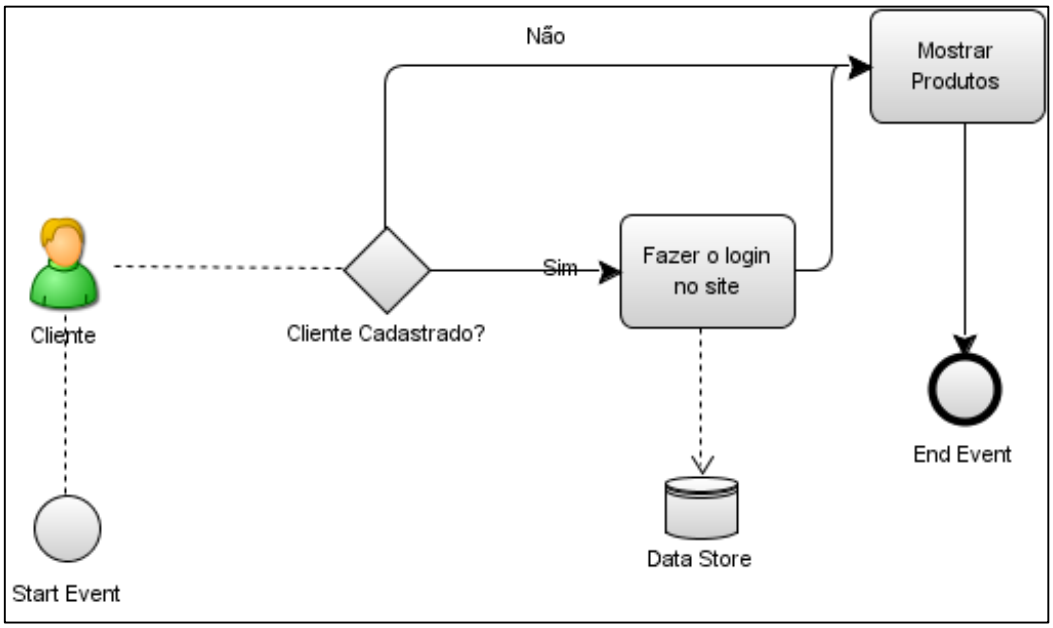

Figura 16 - Catálogo de Produtos Apresentados aos Clientes
Fonte: Autoria própria

Em um *website* B2C não se têm distinção de clientes para que os produtos sejam mostrados, ou seja, todos os clientes possuem acesso aos mesmos produtos e todos eles possuem o mesmo preço.

d) Transações

A Figura 17 mostra um fluxo de processos sobre as transações realizadas considerando a categoria B2B.

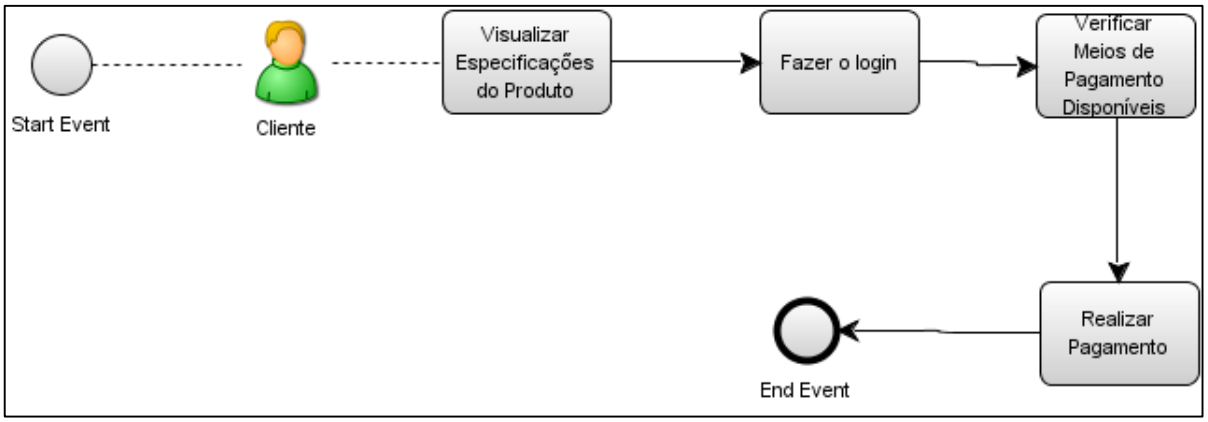

Figura 17 - Transação entre Cliente/Empresa – Categoria B2C
Fonte: Autoria própria

O fluxo de transação da Figura 17 se refere aos processos de transações entre o cliente e a empresa. O cliente verifica determinado produto que possui interesse, faz o *login* no *website*, analisa e escolhe as opções de pagamento disponíveis e realiza o pagamento. Tudo é feito pelo cliente e não há contato com a empresa, ou seja, o cliente decide quando comprar e como pagar.

e) Entrega

O fluxo de processo para a entrega de mercadorias em B2C está apresentado na Figura 18.

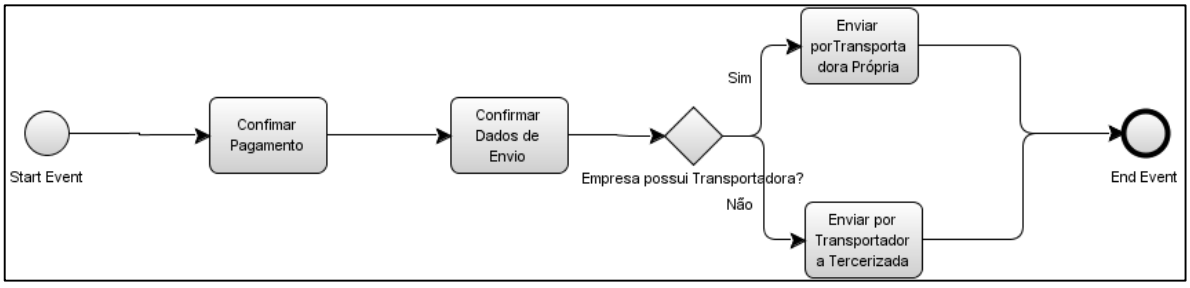

Figura 18 – Processo de Entrega – Categoria B2C
Fonte: Autoria própria

Após a confirmação de endereço de envio e a empresa receber a confirmação de pagamento realizado, o produto é enviado pela própria transportadora ou por uma outra especializada em realizar entrega. O processo de entrega B2C ocorre da mesma maneira do B2B, explicada na seção 2.4 deste trabalho.

2.6 ANÁLISE DO CENÁRIO DE B2B E B2C A NÍVEL BRASILEIRO E MUNDIAL

A facilidade em realizar compras em plataformas de *e-commerce* vem atraindo usuários a cada ano. Segundo uma pesquisa realizada pela *E-bit*, empresa especializada em informações do comércio eletrônico, em 2002 apenas 2 milhões de brasileiros compravam pela internet enquanto que em 2014 o número ultrapassou os 60 milhões de adeptos (EBIT, 2014).

A Figura 19 mostra o crescimento do faturamento anual do *e-commerce* brasileiro em bilhões de reais.

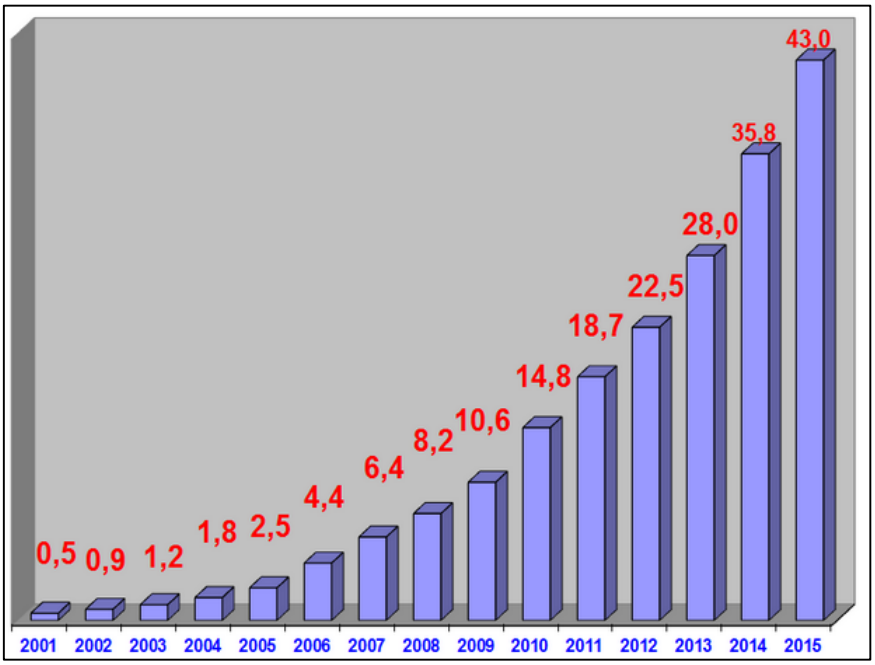

Figura 19 - Faturamento Anual *e-commerce* Brasileiro em bilhões de Reais
Fonte: E-COMMERCE.ORG (2015)

Em uma pesquisa pela *Forrester*, estima-se que as vendas *e-commerce* na categoria B2B nos Estados Unidos, chegue em 2020 a U$ 1.1 trilhão. Considerando o principal *e-commerce* B2B, *Alibaba*, o *website* da empresa Alibaba na China, possui um número de vendas tão grande que seu valor se aproxima a soma do *Ebay* e da *Amazon* juntos, os quais são os maiores *e-commerce* B2B dos Estados Unidos (BONIFACIO, 2015).

A Figura 20 mostra o faturamento anual em bilhões de dólares, dos líderes do *e-commerce* mundial e a posição em que o Brasil se enquadra.

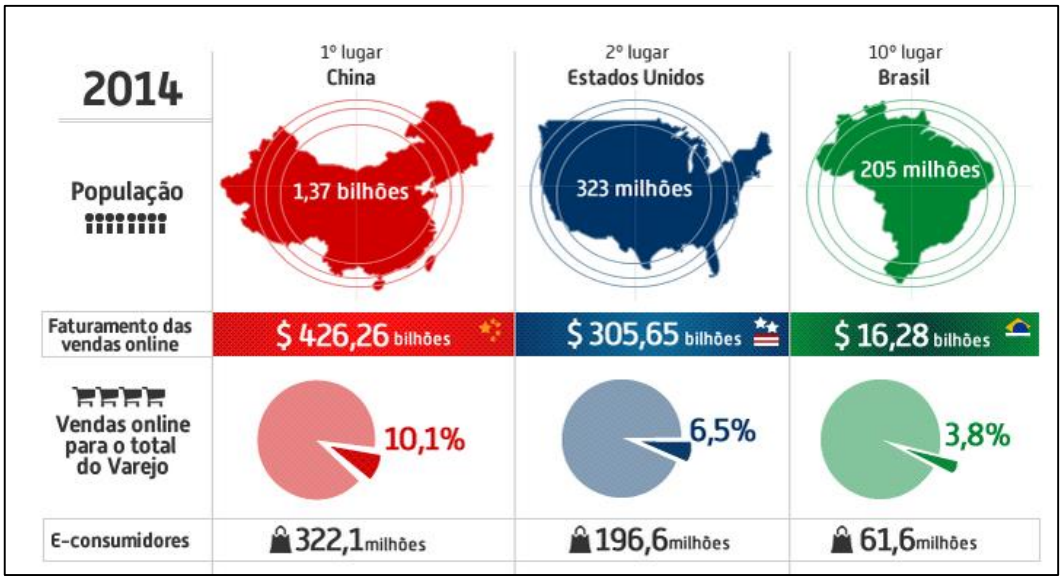

Figura 20 - Faturamento Anual *e-commerce* Mundial em bilhões de Dólares
Fonte: MONTEIRO (2015)

A China é a principal líder de vendas em *e-commerce* comparando a quantidade populacional pelo total de vendas, porém a grande habilidade em vendas é a questão de não se limitar em realizar transações em apenas seu território. Devido a China apresentar produtos com preços atrativos, sua venda externa torna-se cada vez mais ampla (MONTEIRO, 2015).

2.7 FERRAMENTAS DE DESENVOLVIMENTO *E-COMMERCE*

Existem diversas opções para se criar uma loja virtual, dentre elas: a contratação de empresas que oferecem uma plataforma personalizada ou utilizando uma plataforma *e-commerce* de código aberto onde o próprio cliente personaliza a própria loja. A plataforma *e-commerce* é um sistema responsável por criar, gerenciar e realizar transações de uma loja virtual (VALLE, 2015).

Através da pesquisa realizada pela Associação Brasileira de Comércio Eletrônico (ABCOMM, 2015), as três principais plataformas de desenvolvimento *e-commerce* de código aberto são: Magento, WordPress e Prestashop, conforme apresenta o Quadro 2.

Plataforma	Porcentagem de uso	É de Código Aberto?
Magento	**20,2%**	**Sim**
Fastcommerce	5,9%	Não
Vtex	5,6%	Não
WordPress	**4,7%**	**Sim**
Ciashop	4,4%	Não
Jet	3,8%	Não
Tray / Locaweb	3,5%	Não
Mercado Shop	3,2%	Não
Prestashop	**3,2%**	**Sim**
Rakuten	3,2%	Não
Dotstore	2,9%	Não
Uol Host	2,9%	Não
Loja Integrada	2,3%	Não
Nuvem Shop	1,8%	Não
Ez Commerce	1,5%	Não
Loja Mestre	1,5%	Não
Moovin	1,5%	Não
Vertis	1,2%	Não
00K Shop	0,9%	Não
Shop Delivery	0,9%	Não
ATG	0,6%	Não
Hybris	0,6%	Não
Bizcommerce	0,6%	Não
Iluria	0,6%	Não
Lojas Virtuais BR	0,6%	Não
Shopify	0,6%	Não
Webvenda	0,6%	Não
Accurate	0,3%	Não
Bis2Bis	0,3%	Não
Boxloja	0,3%	Não
Chleba	0,3%	Não
Clickweb	0,3%	Não
Elo7	0,3%	Não
F1 Soluções	0,3%	Não
IBM	0,3%	Não
Interspire	0,3%	Não
Loja Eficaz	0,3%	Não
Loja2	0,3%	Não; Possui um plano para uso gratuito, porém limitado.
Martin	0,3%	Não
Maxistore	0,3%	Não
NetSuite	0,3%	Não
Plusnet	0,3%	Não
Primordia	0,3%	Não
Quality Press	0,3%	Não
S3Commerce	0,3%	Não
Saiteria	0,3%	Não
SM Commerce	0,3%	Não
Spree Commerce	0,3%	Sim
TMWXD	0,3%	Não
Usina Store	0,3%	Não
Vannon	0,3%	Não
VExpress	0,3%	Não
ViaEcommerce	0,3%	Não
Xtech Commerce	0,3%	Não
Outros	12,0%	-

Quadro 2 - Ranking de Plataformas *e-commerce* mais usadas
Fonte: Adaptado de ABComm (2015)

O Magento é a principal e mais usada plataforma de *e-commerce*. Ele foi desenvolvido em 2007 por uma empresa privada chamada *Varien*, em 2010 a empresa *Ebay* fez um investimento tornando-a proprietária de 49% do mesmo e em 2011, ela comprou o restante, sendo 100% proprietária (JILL, 2015). Este possui facilidade para customização e é indicado para grandes empresas. Contém ferramentas para busca de produtos e suporta tradução para uma vasta quantidade de idiomas. Por ser uma plataforma mais robusta, para seu uso exige-se um maior investimento e conhecimento para sua manutenção. Sendo assim, apenas profissionais habilitados conseguem desfrutar da sua variedade de personalizações (GOCACHE, 2015).

O *WordPress* é uma plataforma responsável por possibilitar aos usuários criarem páginas para seus *websites* de maneira facilitada. Ele é escrito em *PHP* e utiliza o banco *MySQL* (COSTA, 2015). Para torná-lo uma plataforma de *e-commerce* é necessário utilizar um *plugin* chamado *Woocommerce*. Seu foco é prover uma plataforma para empresas de pequeno porte. O *WordPress* possui uma interface simples e eficiente, porém apresenta desvantagem para usuários que não estão familiarizados (GOCACHE, 2015).

O *Prestashop* é uma plataforma *e-commerce* com forte presença na Europa, este possui as características fundamentais de um sistema *e-commerce* e suas principais vantagens são seu painel administrativo com um visual agradável e de fácil configuração e a possibilidade de migrar seu *website* de outra plataforma (por exemplo, Magento) para a plataforma *Prestashop*. O problema apresentado é que não tem uma comunidade de usuários tão ativa no Brasil (B4W, 2015).

3 DOMAIN-DRIVE DESIGN (DDD)

Este capítulo descreve o modelo de desenvolvimento adotado neste trabalho para a criação da ferramenta proposta. A seção 3.1 apresenta os conceitos sobre o modelo baseado em *Domain-Drive Design* (DDD). As seções 3.2, 3.3, 3.4 e 3.5 relatam as fases que o compõem.

3.1 VISÃO GERAL SOBRE O DDD

Diversas metodologias de desenvolvimento podem ser utilizadas quando se deseja construir um sistema de forma padronizada e planejada. O modelo de desenvolvimento proposto por Erick Evans, *Domain-Drive Design* (DDD), possui como objetivo construir um software de fácil manutenção e é indicado para projetos que contemplam domínios complexos.

Evans (2003) afirma que utilizar DDD é um desafio, pois requer um grande esforço da equipe de desenvolvimento e um conhecimento aprofundado dos objetivos deste software, porém ressalta vantagens de possuir um sistema de fácil manutenção e a capacidade de não se tornar obsoleto quando novas tecnologias surgirem devido a sua capacidade de adaptação.

Este modelo de desenvolvimento não se trata de uma metodologia específica, e sim da união de diversas metodologias orientadas pelas famílias dos processos ágeis mais utilizados e efetivos, aliados a uma relação próxima entre modelo e a implementação.

O modelo de desenvolvimento DDD é composto pelas melhores práticas de desenvolvimento e os padrões de projeto mais utilizados, visa compor uma linguagem ubíqua entre os membros da equipe de desenvolvimento com a equipe responsável pela modelagem do domínio. Sua abordagem de desenvolvimento divide-se em quatro etapas (EVANS, 2003):

- Trabalhar o modelo do domínio;
- Construir o modelo baseado em domínio;
- Refatorar o modelo;
- *Design* estratégico.

Ao adotar a estratégia de desenvolvimento baseada em domínio, a implementação irá se espelhar no modelo, portanto, a maioria dos esforços será em seu aperfeiçoamento.

3.2 TRABALHAR O MODELO DO DOMÍNIO

O domínio de um negócio é o foco principal, é preciso atender de forma integral suas funcionalidades. O objetivo é conceitualizar o domínio em forma de um modelo que represente o conhecimento do funcionamento de um negócio, assim como Evans (2003, p. 24) define: “A model is a selectively simplified and consciously structured form of knowledge. An appropriate model makes sense of information and focuses it on a problem”.

O modelo deve ser analisado e melhorado até que se consiga reproduzir de forma simples e concreta uma funcionalidade do domínio, havendo sempre a colaboração de desenvolvedores e especialistas na área.

Kerievsky (2003) aponta que equipes altamente produtivas aumentam o seu conhecimento conscientemente, praticando um aprendizado contínuo. A falta de conhecimento pode levar a uma modelagem fraca.

A etapa de Trabalhar o Modelo de Domínio é composta por duas subetapas: *Design Consistente* e *Modelo baseado em comunicação*, descritas a seguir.

3.2.1 Design Consistente

Evans (2003) frisa que o conhecimento representado por um modelo deve ser consistente, regras de negócio são tão importantes quanto as entidades envolvidas. Um exemplo é o cálculo do frete em sistemas *e-commerce*: dado um sistema de uma loja qualquer, quando um usuário realiza uma compra são oferecidas as opções de frete disponíveis para a sua região, o preço e a duração estimada do prazo de entrega deste produto.

Para o cálculo do valor final do frete um valor previamente tabelado por cada transportadora entre origem e destino é acrescido de algumas taxas referentes a pedágio, gerenciamento de risco (GRIS), despacho, taxa de administração das secretarias de fazenda (TAS), taxa de restrição ao trânsito (TRT), etc (JAMEF,

2015). As taxas são cobradas de forma diferente por cada transportadora, na Figura 21 encontra-se o modelo inicial para a situação-problema do cálculo do frete.

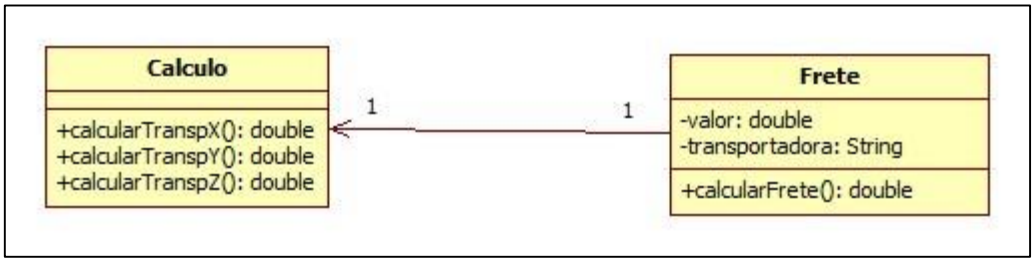

Figura 21 - Modelo inicial cálculo de frete
Fonte: Autoria própria

O modelo representa as operações que realizam apenas a soma das taxas ao valor já tabelado referente ao transporte entre localidades. Na Figura 22 observa-se um trecho do código destas operações implementadas em PHP.

```php
public function calcularTranspX(){
    $GRIS = 2.5;
    $TAS = 3.75;
    $TRT = 0.85;
    $despacho = 5;

    // acesso ao banco para obter valor tabelado do frete entre origem e destino //

    $valorFinal = $valorTabelado + $GRIS + $TAS + $TRT + $despacho;

    return $valorFinal;
}

public function calcularTranspY(){
    $GRIS = 5;
    $TAS = 0.75;
    $TRT = 2.5;
    $despacho = 3;

    // acesso ao banco para obter valor tabelado do frete entre origem e destino //

    $valorFinal = $valorTabelado + $GRIS + $TAS + $TRT + $despacho;

    return $valorFinal;
}
```

Figura 22 - Código das operações de frete
Fonte: Autoria própria

Através do trecho de código é possível observar que a classe *Calculo* fica sobrecarregada de métodos com o cálculo do frete de cada transportadora, para um especialista do negócio fica fácil identificar cada método e o significado de cada taxa, mas alguém sem conhecimento das regras teria dificuldades em compreendê-

las. Uma forma de melhorar esta classe é utilizar padrões de projeto, neste caso em específico, algum padrão que encapsule as regras de negócio das transportadoras de forma mais organizada. Hedin descreve a importância da utilização de padrões de projetos no desenvolvimento de software:

> Padrões de projeto são maneiras sistemáticas de documentar experiências e melhores práticas em soluções de software. Um padrão possui tipicamente um nome, uma motivação exemplo, uma sugestão exemplo, e uma discussão acerca das vantagens e desvantagens desta solução [...] usando padrões de projeto podemos obter uma terminologia comum entre os especialistas, fazendo com que se comuniquem de forma mais eficiente sobre diferentes soluções, e também ensinam rapidamente as soluções aos novatos (HEDIN, 2000, tradução nossa).

Padrões de projeto facilitam e padronizam o desenvolvimento de software. Evans (2003) sugere a utilização do padrão *Strategy* em casos semelhantes ao apresentado. Sua motivação é de substituir regras desnecessárias e métodos sobrecarregados de regras em classes do domínio, porém, não são todos os casos que necessitam um *design* elaborado como este, aplica-se somente nos pontos mais importantes referentes ao domínio do sistema.

O padrão de projeto *Strategy* possui dois princípios básicos, encapsular o conceito que varia e criar uma interface que irá usar tais conceitos, desta maneira o código encapsulado poderá sofrer mudanças quando necessário e a interface não será afetada, obtendo um código de fácil manutenção e reusabilidade (GEARY, 2002).

Ao aplicar o padrão *Strategy* no cálculo do frete, pode encapsular o cálculo das taxas de cada transportadora em uma classe concreta que implementa a interface *calculoStrategy*, como observa-se na Figura 23.

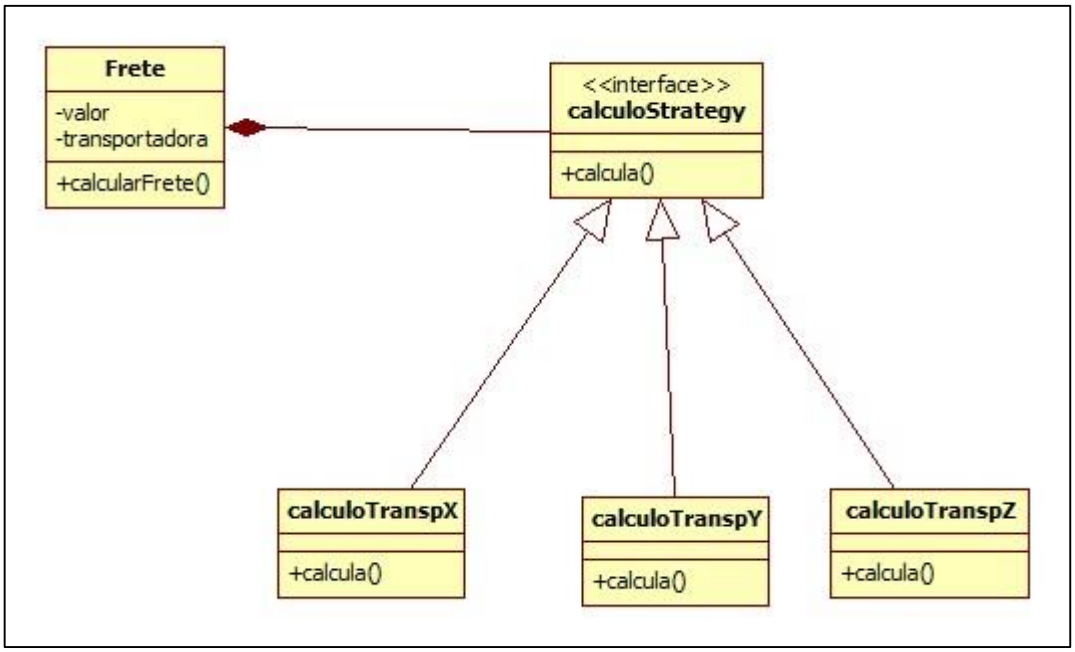

Figura 23 - Modelo com aplicação do padrão *Strategy* para o cálculo do frete
Fonte: Autoria própria

O modelo representa a aplicação do padrão *Strategy* considerando apenas um contexto básico do módulo de entregas de um sistema *e-commerce*, um trecho de código gerado pelo modelo da Figura 23 pode ser visualizado na Figura 24.

```
public function calcularFrete(){ //método implementado na classe Frete

    $frete = calculoStrategy::calcula(); //chamada pro cálculo através da interface calculoStrategy
}


class calcularTranspX implements calculoStrategy{ //classe concreta que implementa
    public function calcula(){                      // o método calcular para a transportadora X
    }
}
```

Figura 24 – Código após a aplicação do padrão *Strategy* para o cálculo do frete
Fonte: Adaptado de Evans (2003)

O método *calcula()*, localizado na classe *calculaTranspX()*, concentra a regra de negócio desta transportadora, enquanto que a interface *calculoStrategy* possui um código mais limpo e organizado e gera uma classe concreta para cada transportadora utilizada pelo sistema. Neste exemplo é utilizado apenas o padrão de projeto: *Strategy*, porém há outros padrões que também podem ser utilizados para proporcionar um *design* consistente ao modelo e podem ser consultados através do livro do Gamma *et al*. (2007).

3.2.2 Modelo Baseado em Comunicação

Um modelo não se limita apenas a diagramas em *Unified Modeling Language* (UML), para que um modelo baseado em comunicação seja efetivo todos os meios possíveis deverão ser utilizados: diagramas informais com representações básicas de funcionalidades do sistema e comentários com explicações importantes, documentos que expliquem funcionalidades de difícil representação em diagramas ou que expliquem algum diagrama, conversas casuais, entre outros (EVANS, 2003).

Modelos possuem inúmeros termos técnicos referentes ao código, em modelos sem uma linguagem comum, desenvolvedores constantemente precisam realizar tradução para especialistas do domínio. Esta tradução em excesso confunde conceitos do domínio e acaba gerando refatoração, que por sua vez, inibe a criação de modelos mais profundos (EVANS, 2003).

Para resolver este problema, desenvolvedores e especialistas do domínio deverão utilizar uma linguagem ubíqua. O vocabulário desta linguagem inclui nomes de classes, operações proeminentes, termos que se referem às regras específicas do domínio, nomes de padrões, entre outros que se destacam no projeto. O modelo será o esqueleto da linguagem e qualquer mudança nesta será uma mudança no modelo (EVANS, 2003).

A linguagem ubíqua deve ser a única usada para expressar um modelo e todos os membros da equipe devem concordar com todos os termos específicos sem que existam ambiguidades e a necessidade de traduções (BERTOLINI, 2015).

Um exemplo prático de uma possível ambiguidade em um sistema de *e-commerce* pode ser encontrado em módulos de desconto. Atualmente, para atrair consumidores, as lojas virtuais oferecem inúmeros descontos: para primeira compra, quando o gasto é acima de determinado valor, fretes grátis, em feriados, entre outros. Devido ao grande número de descontos possíveis, se não houver o cuidado o código poderá conter ambiguidade e ocasionar confusão. Problemas de ambiguidade são facilmente solucionados com o uso de prefixos ou sufixos relacionando-os ao contexto. (BERTOLINI, 2015).

Em determinados momentos, o comportamento de objetos não é facilmente ilustrado por meio de diagramas, tornando-se necessária a utilização de documentos por escrito, mesmo em UML, constantes e asserções acabam utilizando textos explicativos. Diagramas são comunicativos e explicativos, facilitam a representação

de ideias, porém não devem conter demasiadas informações para que possam ser compreendidos facilmente (EVANS, 2003). Documentos devem apenas complementar o código e a comunicação, tendo como principal finalidade explicar funcionalidades:

> [...] documentos podem clarear funcionalidades quando a linguagem de programação não suporta a implementação clara de algum conceito [...], se os termos explicados em um documento não começarem a aparecer no código, significa que o documento não está cumprindo o seu devido propósito[...], ao manter os documentos simples e apenas complementares ao código e à comunicação é possível mantê-los conectados ao projeto (EVANS, 2003, tradução nossa).

Uma das características do DDD é que o código se torna uma representação do modelo, portanto, qualquer alteração no código deverá ser realizada no modelo.

3.3 CONSTRUIR O MODELO BASEADO EM DOMÍNIO

Desenvolver um bom modelo é uma tarefa que exige criatividade, enquanto que o *design* e a implementação são tarefas relativamente mais sistemáticas. Para construir um bom modelo baseado em domínio, é necessário isolá-lo de outras regras de negócio do sistema. A utilização de padrões de projeto é um passo fundamental para realizar o isolamento da melhor forma possível.

Há maneiras de se dividir um sistema, convencionalmente a indústria divide um software em camadas e no DDD os projetos são criados em quatro camadas (EVANS, 2003) conforme ilustra a Figura 25.

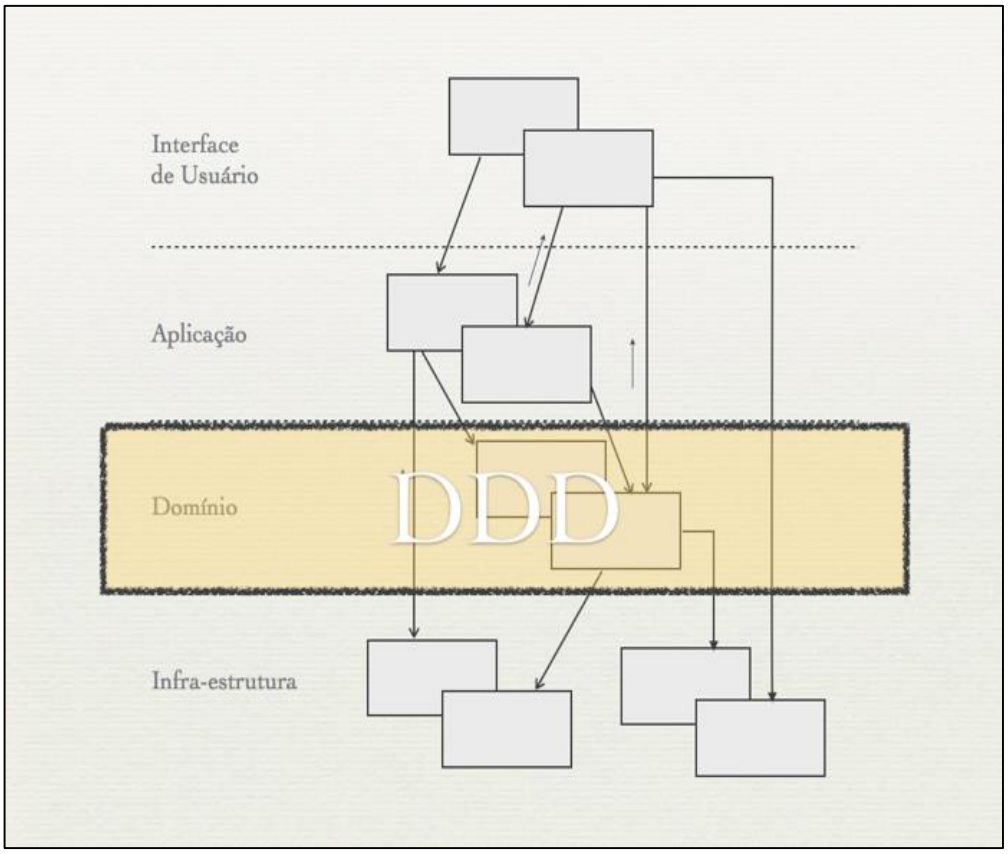

Figura 25 - Camada de Domínio
Fonte: Cukier (2010)

A *Camada de Interface* é responsável por mostrar informações para o usuário e interpretar os seus comandos. A *Camada de Aplicação* define as funcionalidades do sistema, é uma camada de comunicação entre a camada de Interface e a camada do Domínio, não possui nenhuma regra de negócio, apenas delega tarefas aos objetos do domínio.

A *Camada de Domínio*, detalhada na seção 3.3.1, implementa as principais regras de negócio do sistema, as classes encontradas nesta camada fazem parte do domínio, facilmente identificadas na descrição do sistema. Controla as principais funcionalidades apresentadas pelo software. Por fim, a *Camada de Infra-estrutura* provém serviços para as demais camadas, persistência para o domínio, envia mensagens para a aplicação, desenha *widgets* para a interface, entre outros. Padrões de interação entre as camadas também podem ser implementados nesta camada através de um framework.

Arquiteturas em camadas são usadas na maioria dos sistemas atuais, com diversos tipos de divisões, ao desenvolver utilizando o DDD, se requer apenas uma

única camada, a camada de domínio. Em um desenvolvimento baseado em domínio esta espelha os conceitos do modelo (EVANS, 2003).

3.3.1 Camada de Domínio

Esta camada é base de todo o projeto, a conexão entre o modelo e a implementação em DDD é minuciosa, a correta concepção de entidades, objetos de valor e serviços, assim como a utilização correta de módulos irá favorecer esta conexão (EVANS, 2003).

Para a construção desta camada são propostos alguns padrões de projetos em DDD, estes padrões são conhecidos como blocos de construção. São eles:

- Entidades – possuem uma modelagem especial, um ciclo de vida e podem radicalmente mudar de forma e conteúdo. Suas identidades devem ser definidas para que possam ser encontradas de forma efetiva (EVANS, 2003). Por exemplo, um Cliente realiza um cadastro e uma compra, futuramente pode se tornar inativo e ser excluído, etc.
- Objetos de valores – “são instanciados para representar elementos do design que somente importa o que são, e não quem são” (EVANS, 2003, tradução nossa). Dentro de um sistema *e-commerce* existem vários tipos de produtos, ao analisar um deles como uma camisa, se pode ter diversas cores, uma mesma cor pode pertencer a diversos produtos, então este objeto pode ser referenciado por diversas entidades e são imutáveis, ou seja, o objeto representa um atributo, caso contrário é uma entidade.
- Serviços – algumas operações importantes para o domínio não se encaixam em nenhum objeto ou objetos de valor, ao forçar estas operações em objetos, acaba-se por ter uma programação procedural e foge à definição do objeto, fazendo com que o objeto perca sua clareza e dificulte seu entendimento e sua refatoração. Tais operações são reunidas em objetos mascarados no modelo e geralmente acompanham o sufixo “*Manager*” (EVANS, 2003).
- Módulos – copiosamente utilizados em sistemas, poucas vezes são tratados da maneira correta, módulos não devem apenas dividir códigos, mas conceitos. Quanto menor a relação entre os módulos do

sistema, mais facilmente cada módulo poderá ser analisado e refatorado sem que haja a necessidade de grandes mudanças no sistema (EVANS, 2003). Por exemplo, em um sistema *e-commerce* se pode ter um módulo específico de cadastro de usuários.

- Agregados – servem principalmente para manter a integridade do sistema, compostos de entidades e/ou objetos de valor possuem uma classe raiz. Quando houver a necessidade de manipulação de algum dado que esteja neste agregado, o acesso só será possível através da raiz (CUKIER, 2010). Como exemplo, em um *e-commerce* se pode ter uma classe *Conjunto*, que reúne produtos que não são vendidos separadamente.
- Fábricas – classes responsáveis pela instanciação de objetos, objetos de valor e agregados. Quando tais instâncias se tornam complexas ou revelam muito da estrutura interna do sistema são utilizadas fábricas para encapsulá-las (EVANS, 2003). Como por exemplo a criação de diferentes plataformas de e-commerce seria necessária a utilização de uma classe EcommerceFactory para encapsular tais instâncias.
- Repositórios – responsáveis pelo gerenciamento do ciclo de vida de entidades, objetos de valor e agregados, centralizando as operações de criação, edição e remoção de objetos (CUKIER, 2010). Em DDD repositórios representam uma coleção de entidades do domínio e utilizam a linguagem ubíqua em suas interfaces (EVANS, 2003).

A partir destes blocos de construção, elabora-se um modelo que representa o domínio do sistema.

3.4 REFATORAR O MODELO

Após elaborar um modelo inicial utilizando-se de blocos de construção propostos pela estratégia de desenvolvimento DDD, deve-se prosseguir com o seu aperfeiçoamento para obter um modelo cada vez melhor:

> O modelo deve ser refatorado e melhorado na medida em que se obtém maior compreensão de conceitos importantes do domínio. Muitas vezes alguns conceitos do domínio estão implícitos no código. O trabalho do desenvolvedor é tentar identificar esses conceitos implícitos e torná-los explícitos (CUKIER, 2010).

Refatorar o modelo é redesenhá-lo de maneira que não altere o seu funcionamento. As refatorações mais impactantes nos sistemas são aquelas que trazem compreensões novas ao domínio, tornando-o cada vez mais claro, e consequentemente o código também (EVANS, 2003).

O processo de refatoração é árduo, quanto mais é realizado mais claro fica o modelo e maiores as chances de ocorrer um *breakthrough*, que Evans (2003) descreve como o momento em que o modelo, após passar por inúmeras refatorações, atinge um ponto em que novas compreensões e melhorias evoluem drasticamente, valorizando o processo, na Figura 26 é possível observar este ponto.

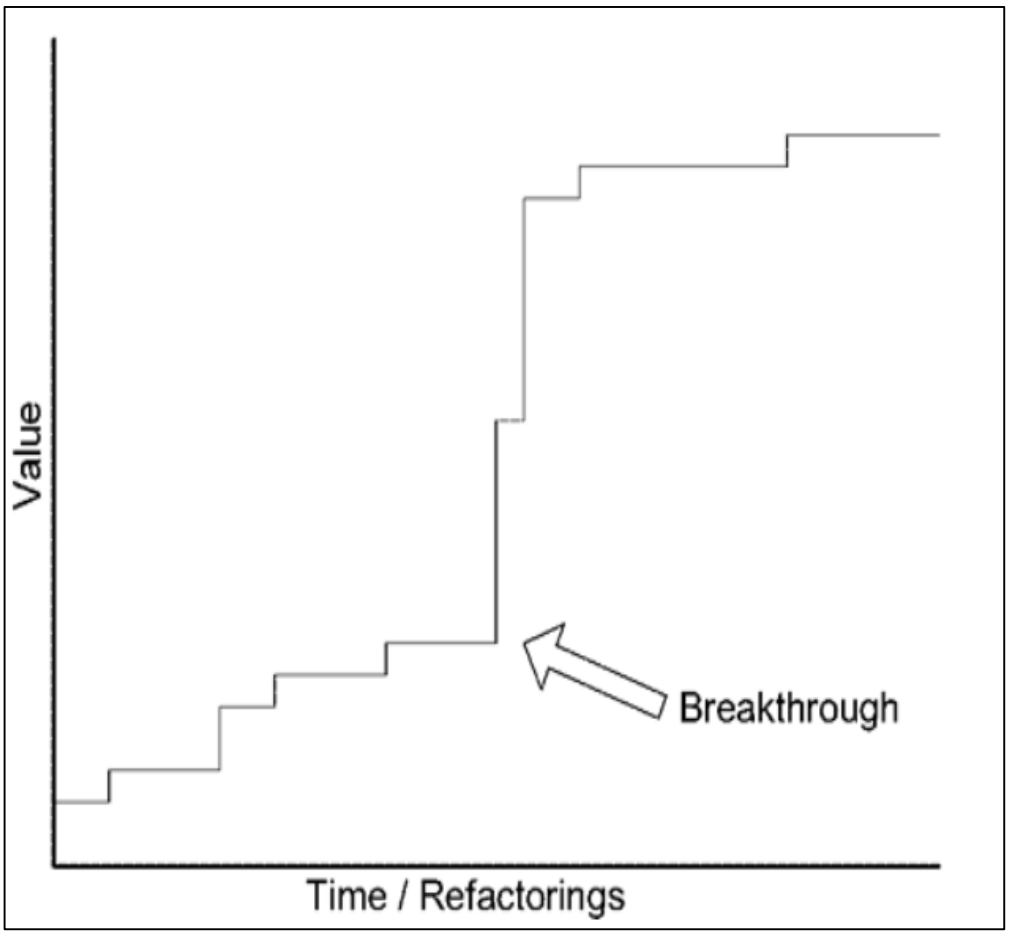

Figura 26 – *Breakthrough*
Fonte: Evans (2003)

Para os processos de refatoração, Evans (2003) sugere a utilização de alguns padrões. São eles:

- Interface de intenção revelada – nomes dos métodos de objetos deverão indicar “o que” realizam, mas não podem dizer “como”, pois quebrariam o encapsulamento destes métodos (CUKIER, 2010).
- Funções sem efeitos-colaterais – colocar a maior parte da lógica do programa dentro de funções que não alteram o estado de objetos (EVANS, 2003).
- Asserções – para métodos que alteram o estado de objetos, criar unidades de testes automáticos ou colocar asserções no código que realizem a validação das alterações (CUKIER, 2010).

Assim como sistemas crescem de maneira complexa através das técnicas de refatoração e refinamento, cresce a necessidade de técnicas para manipular e compreender estes grandes modelos e manter a sua integridade, para isto se deve traçar uma estratégia para que os esforços estejam realmente focados no que interessa para o sistema.

3.5 PROJETO ESTRATÉGICO

As fases inicias do DDD são importantes para o desenvolvimento do produto final, porém a parte mais relevante desta estratégia é quando o sistema torna-se complexo, por isso se deve dividir o sistema em *contextos delimitados*. Para isto, utiliza-se técnicas para a manipulação e compreensão de grandes modelos (EVANS, 2003).

Cada contexto delimitado deve ser dividido de forma clara para todos os integrantes da equipe de desenvolvimento, as fronteiras entre os contextos devem ser conhecidas por toda a equipe, cada trecho de código deve pertencer a exatamente um contexto (CUKIER, 2010).

Muitas vezes integrantes de equipes envolvidas em outros contextos do sistema usam termos diferentes, por isso, o projeto deverá usar um mapa de

contextos como o observado na Figura 27, facilitando a comunicação e criando um ponto de conexão entre desenvolvedores de equipes diferentes.

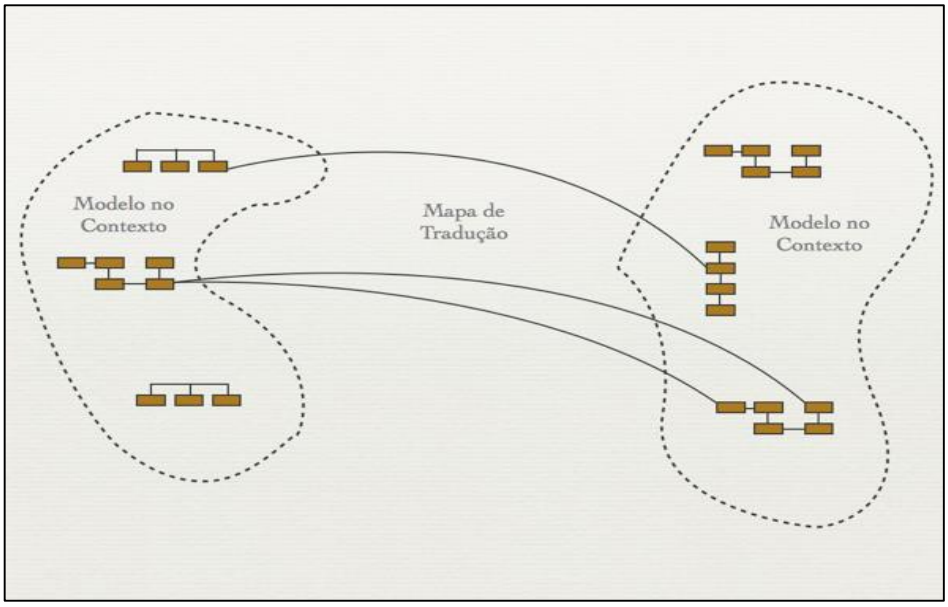

Figura 27 - Mapa de Contextos
Fonte: Cukier (2010)

Para dividir o sistema em contextos delimitados são utilizadas algumas técnicas:

- Núcleo compartilhado – isto é um contexto delimitado que subdivide um contexto do domínio do sistema, requer excelente comunicação entre os envolvidos (KHAN, 2015). Cukier (2010) ressalta a importância da presença de testes automatizados que devem ser executados por qualquer uma das equipes quando alguma alteração for necessária, conforme Figura 28. Ao utilizar núcleos compartilhados, alterações devem ser comunicadas a todas equipes envolvidas, o código compartilhado tende a ser mais difícil de mudar pois depende da aceitação de duas ou mais equipes, contudo, evitam-se duplicações de código.

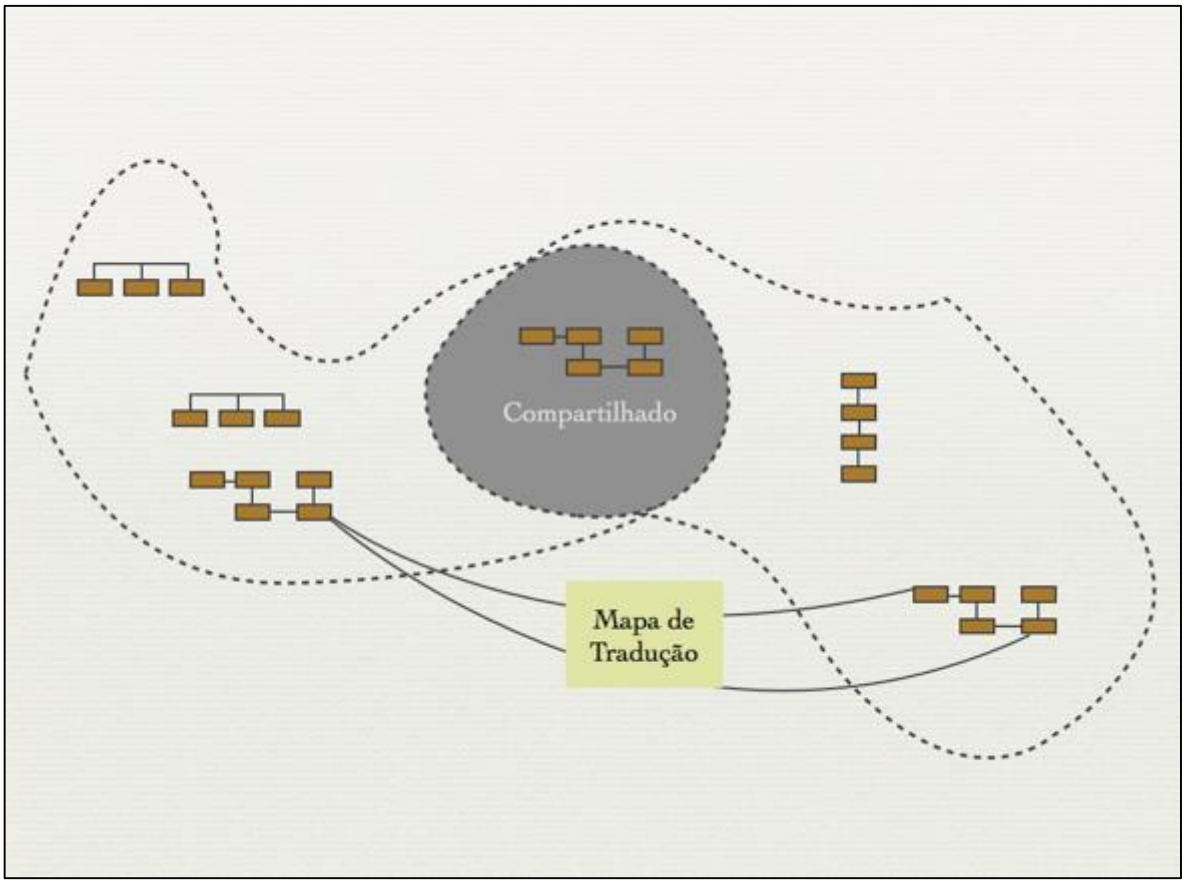

Figura 28 - Núcleo Compartilhado
Fonte: Cukier (2010)

- Produtor-Consumidor – relação bem definida entre as equipes, Produtor é a equipe que desenvolve o software para o time Consumidor, neste tipo de relação o produtor assumirá o compromisso de entregar aquilo que o seu consumidor espera (CUKIER, 2010).
- Conformista – quando uma equipe que trabalha com um contexto possui uma dependência em outro e não possui nenhuma influência ou oportunidade para colaborar com este outro contexto. Quando esta relação de conformidade é estabelecida, não se pede funcionalidades para a outra equipe, mas sim, adapta-se ao que está sendo oferecido (KHAN, 2015).
- Camada anticorrupção – quando contextos coexistem em diferentes partes do sistema, há uma tendência destes contextos colidirem conceitos entre os modelos e a intenção de cada um acaba se perdendo. Neste caso, se deve criar uma camada para isolá-los, esta camada será responsável pela comunicação

bidirecional entre os contextos (KHAN, 2015). Uma camada anti-corrupção também pode ser criada para separar um sistema antigo com código bagunçado de um novo bem organizado, a camada será responsável em traduzir e adaptar as chamadas para o sistema antigo (CUKIER, 2010). A Figura 29 apresenta um exemplo de funcionamento desta camada por meio da criação de adaptadores, serviços e tradutores responsáveis pela comunicação entre uma camada e outra.

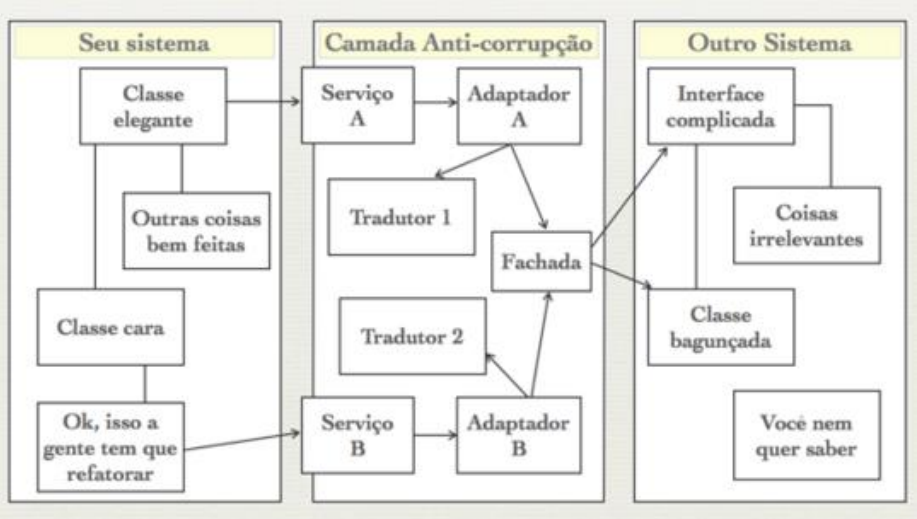

Figura 29 - Camada Anti-corrupção
Fonte: Cukier (2010)

- Caminhos separados – utilizado quando o custo de integração não compensa o benefício, times seguem caminhos diferentes de forma que dois sistemas não dependam um do outro. Um exemplo é implementar um cadastro de CEP local em um sistema *e-commerce* ao invés de integrar com o sistema dos correios (CUKIER, 2010).
- Serviço aberto de funcionalidades e linguagem integrada – quando inúmeros clientes interagem com uma mesma parte do sistema, torna-se inviável a utilização de adaptações para cada um destes clientes, a solução é criar um Serviço Aberto de Funcionalidades e uma Linguagem Aberta que pode ser utilizada pelos clientes interessados em integrarem com os sistemas. Serviços de empresas como o *Google*, *Twitter*, *Amazon*, dentre outros, já disponibilizam este tipo de ferramenta na forma de API's públicas (CUKIER, 2010).

Conforme mencionado o DDD possui várias etapas e estas foram aplicadas no desenvolvimento da modelagem de um sistema *e-commerce* descrita do capítulo seguinte.

4 MODELAGEM DE UMA FERRAMENTA PARA GERAR *WEBSITE E-COMMERCE* B2B E B2C BASEADA EM DDD

Este capítulo apresenta a aplicação do DDD para a criação do modelo da ferramenta *e-commerce* capaz de gerar *websites* baseados nas categorias B2B e B2C. Foram aplicadas as três primeiras etapas propostas pelo DDD, exceto a de Projeto Estratégico, quarta etapa, que permite dividir o domínio em contextos ficando esta para ser realizada em trabalhos futuros. A Seção 4.1 apresenta como foi aplicado a etapa *Trabalhar o Modelo de Domínio* no estudo de caso. A Seção 4.2 relata a aplicação da etapa do DDD *Construir o Modelo Baseado em Domínio* na criação do modelo da ferramenta. A Seção 4.3 descreve como foi realizada a refatoração do modelo.

4.1 TRABALHAR O MODELO DO DOMÍNIO

Para esta etapa de aplicação do DDD foram levantados seus módulos fundamentais nas categorias B2B e B2C. Estes módulos, foram descritos nas Seções 2.4 e 2.5 do Capítulo 2, são:

- Cadastro e *login* de usuários;
- Relacionamento Cliente e Empresa;
- Catálogos de produtos;
- Transação e Entrega.

O processo de transação e entrega foram descritos separadamente nas seções 2.4 e 2.5, porém durante o processo de modelagem notou-se a necessidade de ambos compartilharem das mesmas entidades. Por exemplo, a tabela Frete é usada tanto no processo de transação − necessária para o cálculo do frete referente ao peso total dos produtos no carrinho − como também no processo de entrega − a qual possui uma ligação com uma tabela Transportadora com o objetivo de verificar dados sobre o envio e valores correspondentes. Por isso, o processo de transação e entrega foram modelados em um único módulo.

O Quadro 3 mostra cada módulo e apresenta as funcionalidades necessárias para seu funcionamento, bem como em qual categoria estão presentes.

Módulo	Funcionalidade	Categoria
Cadastro e *Login* de Usuários	• Incluir Usuário • Definir Tipo de Usuário • Realizar *Login*	• B2B e B2C • B2B e B2C • B2B e B2C
Relacionamento Cliente e Empresa	• Contrato • Atendimento ao Cliente • Suporte ao Cliente	• B2B • B2B e B2C • B2B e B2C
Catálogo de Produtos	• Incluir Produto • Listar Produtos	• B2B e B2C • B2B e B2C
Transação e Entrega	• Adicionar Produto • Excluir Produto • Alterar Quantidade do Produto • Calcular o Frete • Calcular o Valor do Produto • Verificar Formas de Pagamento • Verificar Cupom Desconto • Confirmar Pagamento • Finalizar Compra • Cancelar Compra • Verificar Carrinho • Verificar *Login* • Verificar Endereço para Entrega	• B2B e B2C • B2B e B2C • B2B e B2C • B2B e B2C • B2B e B2C • B2B e B2C • B2C • B2B e B2C • B2B e B2C • B2B e B2C • B2B e B2C • B2B e B2C • B2B e B2C

Quadro 3 - Módulos e Funções Desempenhadas para B2B e B2C
Fonte: Autoria própria

Após a definição de tais módulos, foram selecionados para a implementação os de *cadastro e login de usuários* e *catálogos de produtos*. Esses módulos foram escolhidos e implementados com o objetivo de mostrar que ambos possuem funcionalidades que é exatamente igual (*cadastro e login de usuários*) e outra que apresenta uma pequena variação (*catálogos de produtos*), pois na categoria B2B pode-se aplicar regras para mostrar os produtos aos usuários, por exemplo, se o usuário tiver efetuado o *login* no *website*, este possui alguns benefícios ao visualizar os produtos podendo visualizar por: preços, quantidade de itens disponíveis, entre

outros detalhes. Já na plataforma B2C a relação de produtos mostrados aos usuários é a mesma tanto para um que tenha efetuado o *login*, quanto para outro que apenas está navegando no *website*.

A principal função do módulo de cadastro e *login* de usuários é a inserção de um usuário, sem usuários cadastrados não é possível realizar o *login*. Quando um usuário acessa a página de cadastro, são oferecidas duas opções de usuários disponíveis: físico e jurídico. Para prosseguir o cadastro, deve-se selecionar uma dessas opções e em seguida ele pode preencher o restante dos dados.

Nesta etapa foram identificados apenas alguns dos elementos que fazem parte do domínio dos módulos escolhidos, posteriormente na seção de refatoração será refinado e trabalhado todos os elementos responsáveis pelo relacionamento dessas entidades.

A Figura 30 mostra os primeiros elementos identificados no cadastro e *login* de usuários que fazem parte de ambas as categorias, B2B e B2C. A Figura 30 apresenta alguns erros de associação entre instâncias de objetos de uma classe em relação a outra, por exemplo, *Endereco* com *Pessoa*. Este equívoco foi arrumado durante o processo de refatoração.

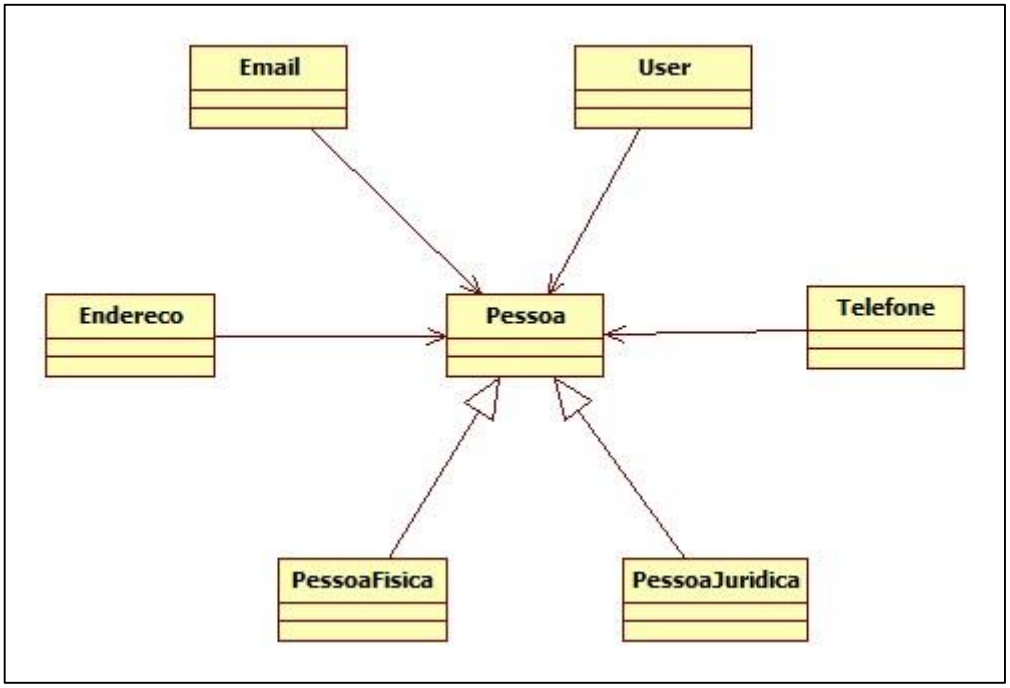

Figura 30 - Primeiros elementos identificados no cadastro e *login* de usuários
Fonte: Autoria própria

O usuário opta pelo qual tipo de pessoa que ele representa durante o cadastro, por isto foram criadas as classes *PessoaFisica* e *PessoaJuridica* as quais

herdam da classe generalizada *Pessoa*. As classes *Endereco*, *Email*, *User*, Telefone são os principais atributos que uma classe *Pessoa* deve ter.

O módulo de *catálogo de produto*, responsável pela listagem de produtos, é realizado por meio de uma busca no banco de dados de acordo com o departamento em que o usuário se encontra ou com a regra de negócio implementada para uma determinada página. Por exemplo, na página inicial pode ser apresentado determinados produtos escolhidos pelo *admin* do sistema. A Figura 31 exibe os primeiros elementos identificados inicialmente para o catálogo de produtos que fazem parte de ambas as categorias, B2B e B2C.

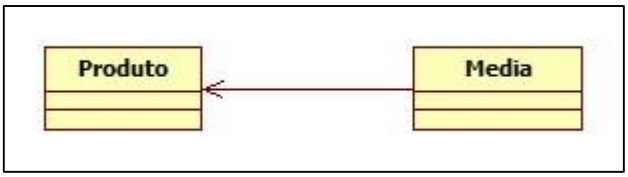

Figura 31 - Primeiros elementos identificados do catálogo de produtos
Fonte: Autoria própria

Para a listagem de produtos observou-se a necessidade de uma busca pelo objeto *Produto* em um banco de dados, o qual possuirá uma instância de *Media* para armazenamento do endereço de cada imagem. A classe *Produto* tem por finalidade armazenar todos os atributos necessários que um determinado produto possui e a classe *Media* é responsável por conter o endereço da localização de cada imagem no disco.

A modelagem das outras funcionalidades: *Relacionamento Cliente e Empresa* e *Transação e Entrega* estão nos Apêndices A e B.

4.2 CONSTRUIR O MODELO BASEADO EM DOMÍNIO

Para a construção do modelo de *e-commerce* foi necessário conhecer as principais funcionalidades das categorias B2B e B2C e isolá-las de forma a obter um modelo comum para ambas. Com isso o modelo *e-commerce* foi dividido em camadas conforme descreve o DDD: interface, aplicação, domínio e infraestrutura. A camada de infraestrutura é a primeira relatada devido a utilização de um framework para organização e controle de toda a estrutura do projeto.

4.2.1 Camada de Infraestrutura

Para o início da implementação optou-se pela escolha de um framework que pudesse atender os requisitos desta camada. Através de algumas pesquisas em *websites* relacionados à desenvolvimento web e *feedback* de usuários, foi escolhido inicialmente o *Zend Framework 2* (ZENDFRAMEWORK2, 2016). Contudo, devido à dificuldade encontrada na instalação e pela desorganização da documentação disponível aos usuários, buscou-se uma outra alternativa que foi o *Symfony* (SYMFONY, 2016).

Symfony é um framework *open source* para desenvolvimento de aplicações *web* através de um conjunto de componentes PHP. Com uma ampla documentação e comunidade ativa, a utilização deste framework tornou o trabalho mais eficaz e bem estruturado.

Na camada de infraestrutura o framework *Symfony* objetiva realizar a organização do trabalho, bem como a interação e persistência dos dados entre as camadas. Para o processo de instalação do framework *Symfony* foi necessário fazer o *download* do código-fonte (SYMFONY, 2016). Depois disso, foi realizado o *download* do *Composer* (ferramenta que gerencia dependências em PHP e permite a declaração de bibliotecas no projeto bem como a instalação e atualização das mesmas) (COMPOSER, 2016). Também foi preciso ter o PHP instalado na máquina.

Após o *download* do *Symfony* e do *Composer*, foi necessário executar o comando para criação de um projeto *Symfony* na pasta onde foi efetuado o *download*: *php symfony new meu_projeto*. A Figura 32 mostra a execução do comando responsável pela criação do projeto *Symfony*:

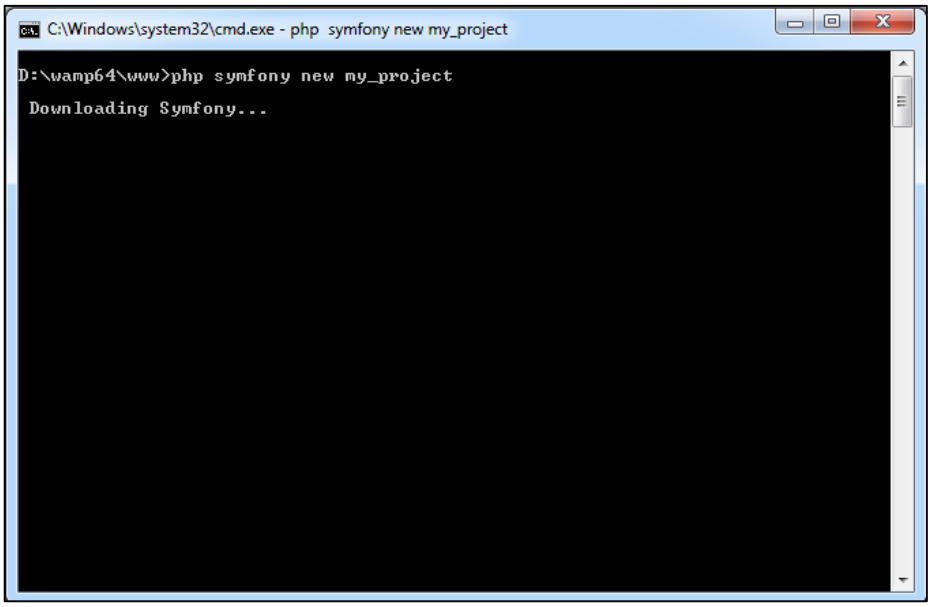

Figura 32 - Criando um projeto *Symfony*
Fonte: Autoria própria

Por meio deste comando é criado um projeto *Symfony*. Em seguida é necessário a instalação do *Composer* seguindo o comando dentro da pasta do projeto: *php composer.phar install*, conforme a Figura 33.

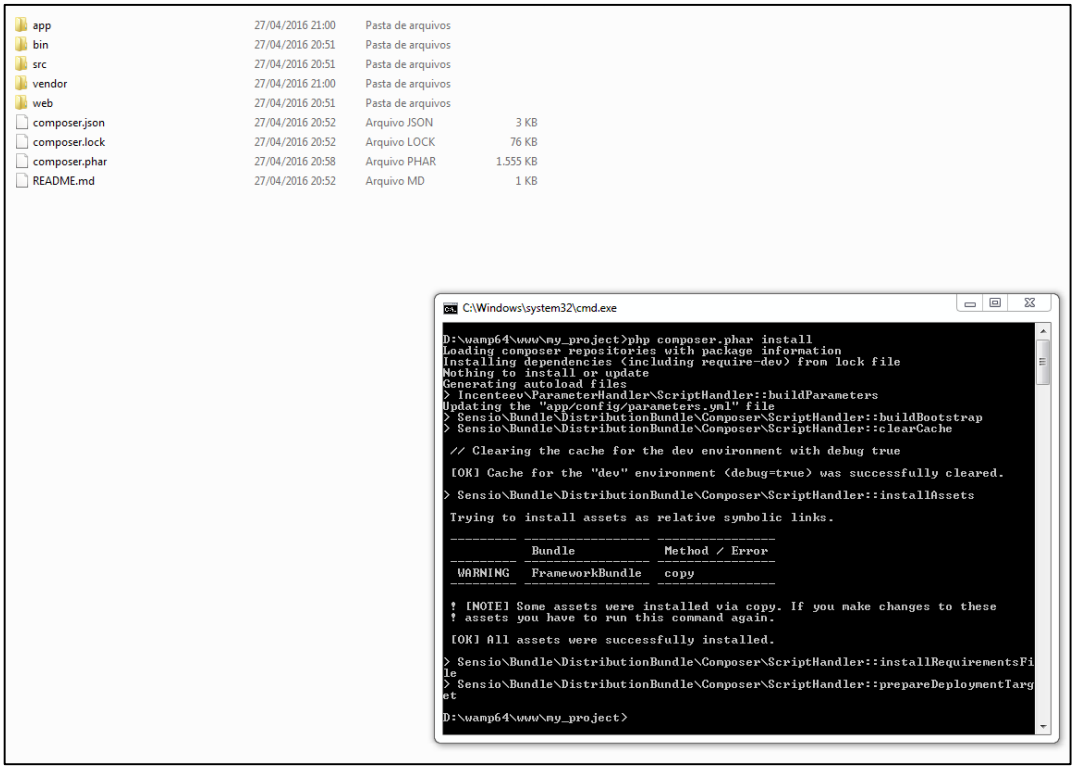

Figura 33 - Instalação do Composer
Fonte: Autoria própria

Após concluído a instalação do *Symfony* e do *Composer*, a execução do *Symfony* é feita pelo seguinte comando dentro da pasta do projeto: *php bin/console*

server:run. O acesso padrão ao framework é feito no *localhost:8000/*. A Figura 34 mostra a execução do framework *Symfony* neste endereço.

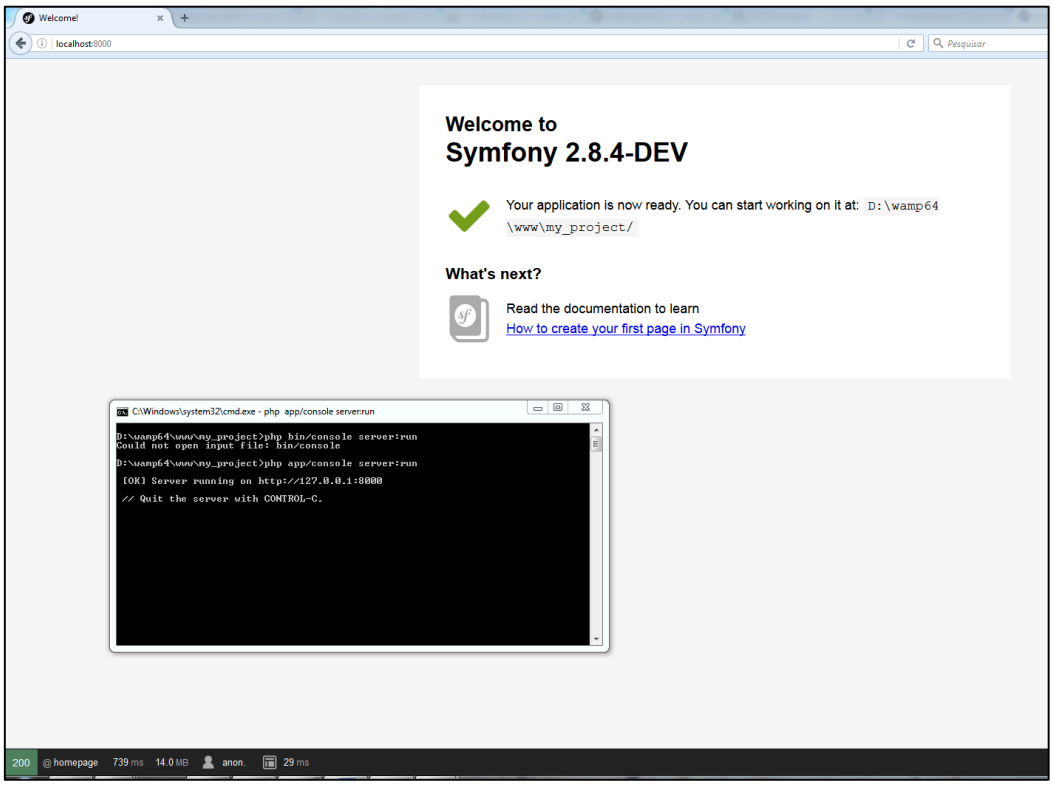

Figura 34 - Execução do *Symfony* no *localhost:8000/*
Fonte: Autoria própria

Após realizar todas essas etapas o framework *Symfony* está pronto para uso. A Figura 35 mostra como essa estrutura está organizada.

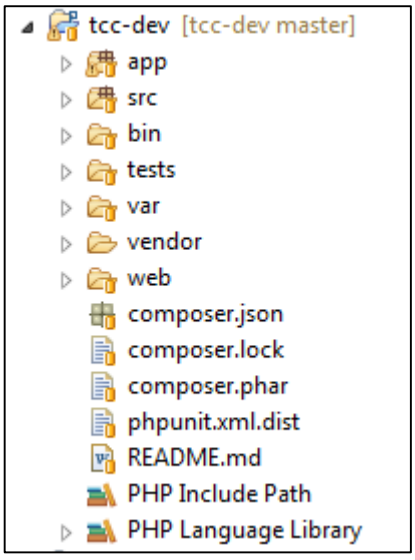

Figura 35 - Estrutura do framework *Symfony* depois de instalado
Fonte: Autoria própria

O projeto *Symfony* é dividido em sete pastas: *app*, *src*, *bin*, *tests*, *var*, *vendor, web*. A pasta *app* possui os arquivos de configuração de toda a aplicação e também os de interface. A pasta *src* é responsável pelos controladores, entidades, repositórios e formulários do sistema. A pasta *bin* controla as dependências. A pasta *tests* é destinada aos arquivos de testes da aplicação. A pasta *var* refere-se a arquivos temporários da aplicação, tais como: *cache*, *sessions* e *logs*. A pasta *vendor* contém todas as bibliotecas utilizadas pelo framework. E a pasta *web* é onde estão os arquivos públicos destinados para que os usuários possam acessá-los, por exemplo, *layouts*, imagens de produtos, entre outras.

Para a realizar a gravação de informações no banco de dados é utilizado a biblioteca *doctrine*, a qual tem por objetivo estabelecer a conexão com o banco, mapear os objetos através de assinaturas nas entidades, persistir as informações e realizar buscas.

Para a camada de interface utiliza-se a *engine* twig, o qual é uma *engine* de *templates* para PHP, com o objetivo de facilitar e organizar os *layouts* da aplicação.

A Figura 36 mostra a localização do *doctrine* e do *twig* dentro da pasta *vendor*, ambos vem instalados por padrão no framework *Symfony*.

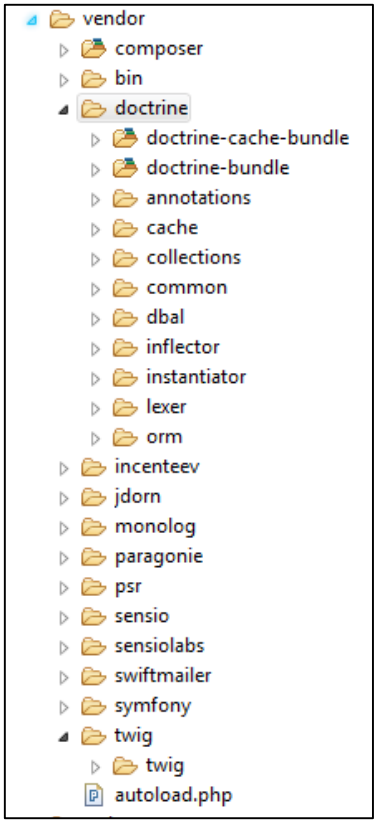

Figura 36 - Biblioteca doctrine e twig
Fonte: Autoria própria

4.2.2 Camada de Interface

A camada de interface do sistema *e-commerce* interage diretamente com o usuário através de formulários, listagens de produtos, entre outros.

O framework *Symfony* possui um diretório padrão para todas as interfaces que pode ser acessadas em */app/Resources/views/*. A Figura 37 mostra essa estrutura.

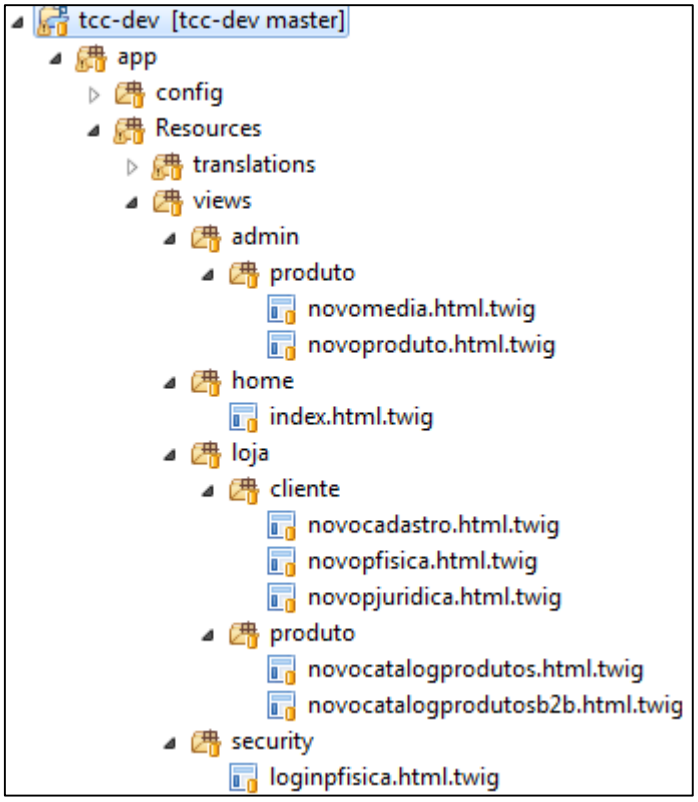

Figura 37 - Arquivos da camada interface em *html.twig*
Fonte: Autoria própria

Dentro da pasta *views* são criadas as pastas e os arquivos necessários para a implementação do *layout* dos formulários *e-commerce*. Esse diretório foi dividido em quatro subpastas: *admin*, *home*, *loja*, *security*. Na pasta *admin* são colocados os arquivos restritos ao acesso de clientes da loja, como por exemplo, o cadastro de produtos. A pasta *home* contém apenas o arquivo da página inicial; a pasta *loja* é composta pelos os arquivos mostrados aos clientes, como por exemplo, a listagem do catálogo de produtos; e a pasta *security* contém os arquivos de autenticação de usuários, como por exemplo, o *login*.

As interfaces para os módulos de cadastro e *login* de usuários e catálogo de produtos observadas na Figura 37 foram implementadas em *HTML* e *TWIG*. O *TWIG*

compila *templates* em código PHP otimizado, isto faz com que os códigos se tornem mais rápidos, seguros e flexíveis.

O *Symfony* também disponibiliza um componente para a criação de formulários diretamente em PHP em */src/AppBundle/*; com isso foi criado a pasta *Form*, exibida na Figura 38.

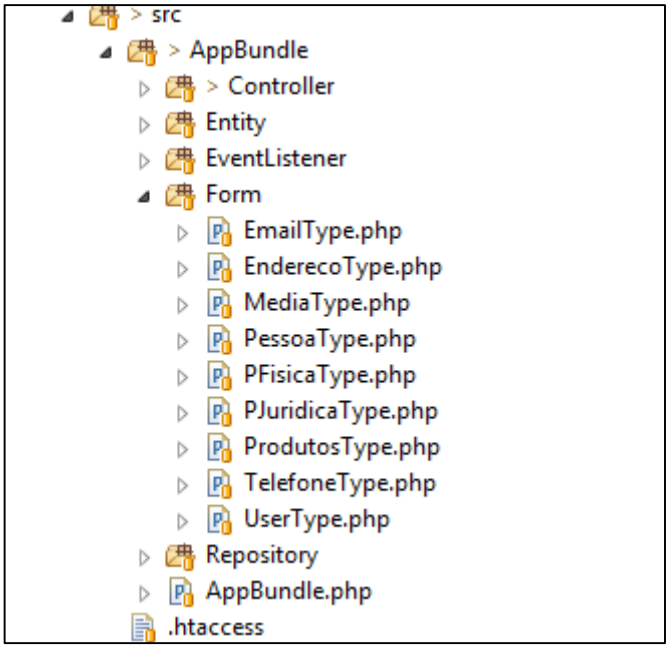

Figura 38 - Arquivos da camada interface em PHP
Fonte: Autoria própria

Através de uma componente (*abstractType*) é possível criar formulários e reutilizá-los onde necessário, tornando-se útil em caso de edição de dados. As Figuras 39 e 40 mostram trechos de código em TWIG e PHP, respectivamente, para tratamento de formulários.

```
{% extends 'base.html.twig' %}

{% block body_id 'cliente_cadastro_novo' %}

{% block main %}
    <h1>{{ 'cadastro_cliente'|trans }}</h1>

    {{ form_start(form) }}
        {{ form_row(form.username) }}
        {{ form_row(form.password) }}
        {{ form_row(form.tipopessoa) }}

        <input type="submit" value="{{ 'label.create_post'|trans }}" class="btn btn-primary" />

    {{ form_end(form) }}
{% endblock %}
```

Figura 39 - Arquivo TWIG
Fonte: Autoria própria

```
class UserType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
        ->add('username', null, array(
                'attr' => array('autofocus' => true),
                'label' => 'Nome de usuario',
            ))
        ->add('password', RepeatedType::class, array(
                'type' => PasswordType::class,
                'invalid_message' => 'The password fields must match.',
                'options' => array('attr' => array('class' => 'password-field')),
                'required' => true,
                'first_options'  => array('label' => 'Senha'),
                'second_options' => array('label' => 'Confirme a sua senha'),
        ))
        ->add('tipopessoa', ChoiceType::class, array(
                'choices'  => array(
                        'Fisica' => 'fisica',
                        'Juridica' => 'juridica',
                ),
                'mapped' => false,
        ))
        ;
    }
```

Figura 40 – UserType
Fonte: Autoria própria

Através de um trecho de código em TWIG, Figura 39, é possível criar e renderizar um formulário utilizando-se da estrutura de um formulário em PHP, na Figura 40 se observa a classe *UserType* com a estrutura de um formulário para um usuário.

4.2.3 Camada de Aplicação

Uma importante ferramenta disponível no framework *Symfony* são os *Controllers*, responsáveis por receber requisições HTTP, criar e retornar uma resposta HTTP.

A pasta *Controller (/src/AppBundle/Controller*) foi criada para reunir os controladores necessários na aplicação, a função de um controlador é renderizar o conteúdo de uma página ou até mesmo redirecionar para uma página de erro caso a página não exista. A Figura 41 mostra os arquivos da camada de aplicação.

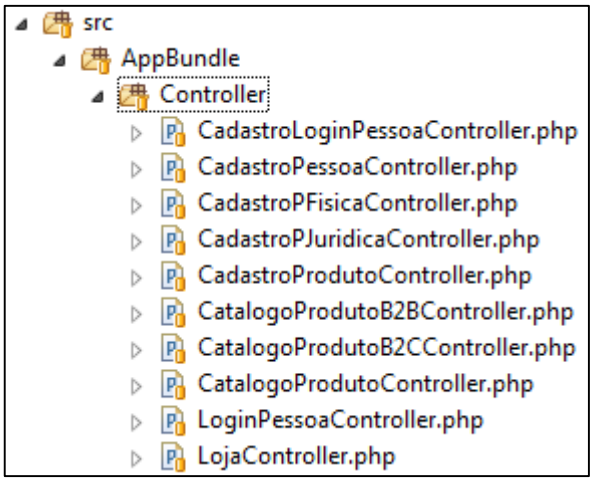

Figura 41 - Arquivos da camada de aplicação
Fonte: Autoria própria

Os controladores da Figura 41 realizam a aplicação do cadastro/*login* e *catálogo de produtos*, os quais tem por objetivo gerenciar uma requisição específica de um usuário, como por exemplo, a confirmação de um formulário de cadastro ou a visualização da página de um produto.

A Figura 42 mostra como exemplo uma parte da implementação do *Controlador de Login de Usuários*, o qual possui a função *loginAction* e tem por finalidade verificar se os campos foram preenchidos corretamente e realizar o *login* do usuário.

```
class LoginPessoaController extends Controller
{
    /**
     * @Route("/login", name="login_store")
     * @Method("GET")
     */
    public function loginAction(Request $request)
    {
        $helper = $this->get('security.authentication_utils');

        return $this->render('security/loginpfisica.html.twig', array(
                // last username entered by the user (if any)
                'last_username' => $helper->getLastUsername(),
                // last authentication error (if any)
                'error' => $helper->getLastAuthenticationError(),
        ));
    }
```

Figura 42 – LoginPessoaController
Fonte: Autoria própria

Pode-se notar também na Figura 42 as assinaturas do método *(@Route, @Method*), as quais tem por objetivo garantir que o método seja acessado somente

com o endereço do link *(@Route)* e com o método de envio correspondente *(@Method)*.

4.2.4 Camada de Domínio

Para modelagem do DDD deve ser seguido os sete blocos de construção: *entidades*, *objetos de valores*, *serviços*, *módulos*, *agregados*, *fábricas* e *repositórios*. Porém, para o desenvolvimento dos módulos de cadastro e *login* de usuários e *catálogo de produtos* foram utilizados apenas quatro blocos de construção: *entidades*, *serviços*, *módulos* e *repositórios*, não encontrando aplicação para os demais blocos nessa implementação.

O desenvolvimento da camada do domínio iniciou com a compreensão dos blocos de construção e a divisão correta dos elementos de uma aplicação *e-commerce*. Na pasta *Entity* (*/src/AppBundle/Entity*) são armazenadas as entidades do domínio que foram identificadas nos módulos implementados, cada classe contém uma assinatura *(@Entity)*, que é identificada pelo *Doctrine* como uma tabela do banco de dados. A Figura 43 mostra algumas das entidades criadas.

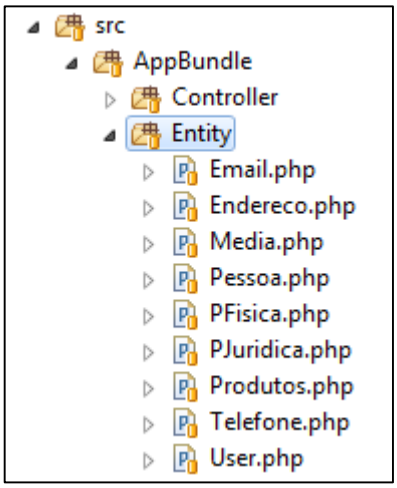

Figura 43 – Arquivos de entidades de domínio
Fonte: Autoria própria

Na Figura 44 é possível visualizar a criação de uma entidade e seus atributos. A biblioteca *Doctrine* possui ferramentas que possibilitam gerar o banco de dados por meio das assinaturas implementadas em cada classe.

```php
<?php

namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity
 */

class Pessoa{

    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\OneToOne(targetEntity="Telefone")
     * @ORM\JoinColumn(name="idTelefone", referencedColumnName="id", onDelete="CASCADE")
     */
    private $telefone;

    /**
     * @ORM\Column(type="string")
     */
    private $tipo;

    /**
     * @ORM\OneToOne(targetEntity="Endereco")
     * @ORM\JoinColumn(name="idEndereco", referencedColumnName="id", onDelete="CASCADE")
     */
    private $endereco;
```

Figura 44 - Entidade Pessoa
Fonte: Autoria própria

Para o desenvolvimento dos módulos escolhidos foi necessário implementar um cadastro de produtos e de usuários, com isso surgiu a necessidade de implementar classes responsáveis pela persistência dos dados provenientes dos formulários de cadastro. Então, foram criados serviços, na Figura 45 observa-se a classe *ProdutoManager* criada para controlar qualquer tipo de serviço relacionado com a entidade *Produto*.

```php
<?php

namespace AppBundle\Services\Admin;

use AppBundle\Entity\Produto;
use Doctrine\ORM\EntityManager;

class ProdutoManager {

    /**
     * @param EntityManager $em  The Doctrine EntityManager
     */
    private $em;


    public function __construct(EntityManager $em) {
        $this->em = $em;
    }

    public function incluir($produto, $media = null) {
        $this->em->persist ( $produto );
        if ($media)
            $this->em->persist ( $media );
        $this->em->flush ();
    }
}
```

Figura 45 - Serviço ProdutoManager
Fonte: Autoria própria

Neste trecho de código nota-se um dos serviços realizados pela classe *ProdutoManager*, o método *incluir* pode ser chamado por qualquer controlador da aplicação e realiza a persistência do objeto produto.

Não houve a necessidade de criação de fábricas para os módulos implementados, porém, as fábricas foram necessárias no modelo inicial da plataforma, o qual possibilita ao usuário escolher e criar um *website e-commerce* B2B ou B2C, a Figura 46 representa este modelo.

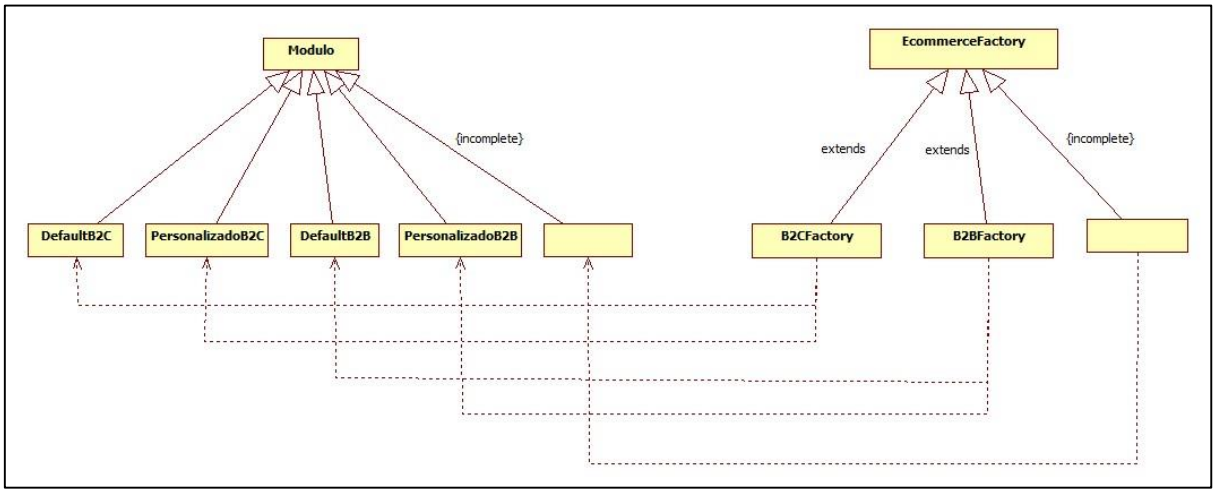

Figura 46 - Modelo responsável por gerar a aplicação *e-commerce* nas categorias B2B ou B2C
Fonte: Autoria própria

O modelo da Figura 46 representa o processo que se dará para a criação de uma plataforma *e-commerce*. Para sua modelagem foram considerados módulos personalizados que poderão ser incluídos na aplicação com o decorrer do tempo, assim como o pacote de módulos padrão.

O modelo representado na Figura 46 foi desenvolvido com o propósito de abranger a criação de outras categorias de *e-commerce* que não foram consideradas, como por exemplo, C2C, B2G e G2G. A inserção dessas novas categorias no modelo é representado pela palavra-reservada {*incomplete}*.

A classe *ProdutoRepository* é necessária para o desenvolvimento do módulo de catálogo de produtos, na Figura 47 é possível visualizar uma simples função que pesquisa os produtos existentes na aplicação e retorna os resultados ordenando-os por nome em ordem ascendente.

```php
namespace AppBundle\Repository;

use Doctrine\ORM\EntityRepository;

/**
 * This custom Doctrine repository is empty because so far we don't need any custom
 * method to query for application user information. But it's always a good practice
 * to define a custom repository that will be used when the application grows.
 * See http://symfony.com/doc/current/book/doctrine.html#custom-repository-classes
 *
 * @author Ryan Weaver <weaverryan@gmail.com>
 * @author Javier Eguiluz <javier.eguiluz@gmail.com>
 */
class ProdutoRepository extends EntityRepository
{
    public function findAllOrderedByName()
    {
        return $this->getEntityManager()
        ->createQuery('SELECT * FROM produto p  ORDER BY p.nome_prod ASC')
        ->getResult();
    }
}
```

Figura 47 - Repositório *ProdutoRepository*
Fonte: Autoria própria

Através da classe *EntitiyRepository* disponibilizada pelo *Doctrine* é possível criar repositórios para todas as entidades da aplicação e acessá-los facilmente indicando os respectivos repositórios por meio de parâmetros nas assinaturas das entidades, como mostra a Figura 48.

```php
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass="AppBundle\Repository\ProdutoRepository")
 */

class Produto{
```

Figura 48 - Assinatura de um Repositório em uma Entidade
Fonte: Autoria própria

A regra de negócio implementada para a listagem dos produtos no módulo de catálogo de produtos pode variar de acordo com cada situação nas categorias de *e-commerce* B2B, por exemplo, poderiam ser criados diversos filtros, tais como: verificar o grupo no qual o usuário pertence e quais os privilégios que esse grupo de usuários possui e com isso realizar uma listagem personalizada; verificar se um usuário efetuou o *login* no sistema, assim o mesmo pode visualizar mais informações sobre um determinado produto ao contrário de um usuário que não tenha efetuado o *login*. Para este trabalho foi criado como exemplo apenas a

verificação se o usuário efetuou ou não o *login* no sistema e com isso ele pode verificar os valores dos produtos. Para as categorias de *e-commerce* B2C não há uma diferenciação na listagem dos produtos.

4.3 REFATORAR O MODELO

A refatoração de um modelo é responsável pelo melhoramento de sua estrutura, sem alterar o modo com que o mesmo funciona. Para isso deve-se abstrair os conceitos de cada classe de forma que a mesma possa desempenhar somente as funções propostas. Isso faz com que a classe não tenha código repetido ou execute um método o qual não deveria ser atribuído a ela (FOWLER, 2013).

Durante o processo de refatoração nesse trabalho, não foram aplicadas técnicas de refatoração, como as propostas por Fowler (2013). O objetivo da refatoração foi obter um melhoramento dos modelos e aplicar padrões de projeto.

Durante esta subseção serão apresentados os modelos iniciais e finais de cada módulo. O primeiro módulo desenvolvido foi o cadastro e *login* de usuários e para sua modelagem foram identificadas as classes responsáveis pelo armazenamento das informações no banco de dados: *Pessoa*, *PessoaFisica*, *PessoaJuridica*, *Email*, *Endereco* e *Telefone*. As classes *cadastraPessoaController*, *cadastraPessoaFisica*, *cadastraPessoaJuridica*, *LoginPessoaControlller*, *LoginPessoaFisica* e *LoginPessoaJuridica* são responsáveis pela execução dos respectivos métodos de cadastro ou *login* de usuários. A Figura 49 mostra esse modelo inicial do cadastro e *login* de usuários e esta figura apresenta alguns erros de associação entre instâncias de objetos de uma classe em relação a outra e entre classes, por exemplo, *cadastraPessoaController* com *cadastraPessoaFisica* e *cadastraPessoaJuridica*. Este equívoco foi arrumado durante o processo de refatoração.

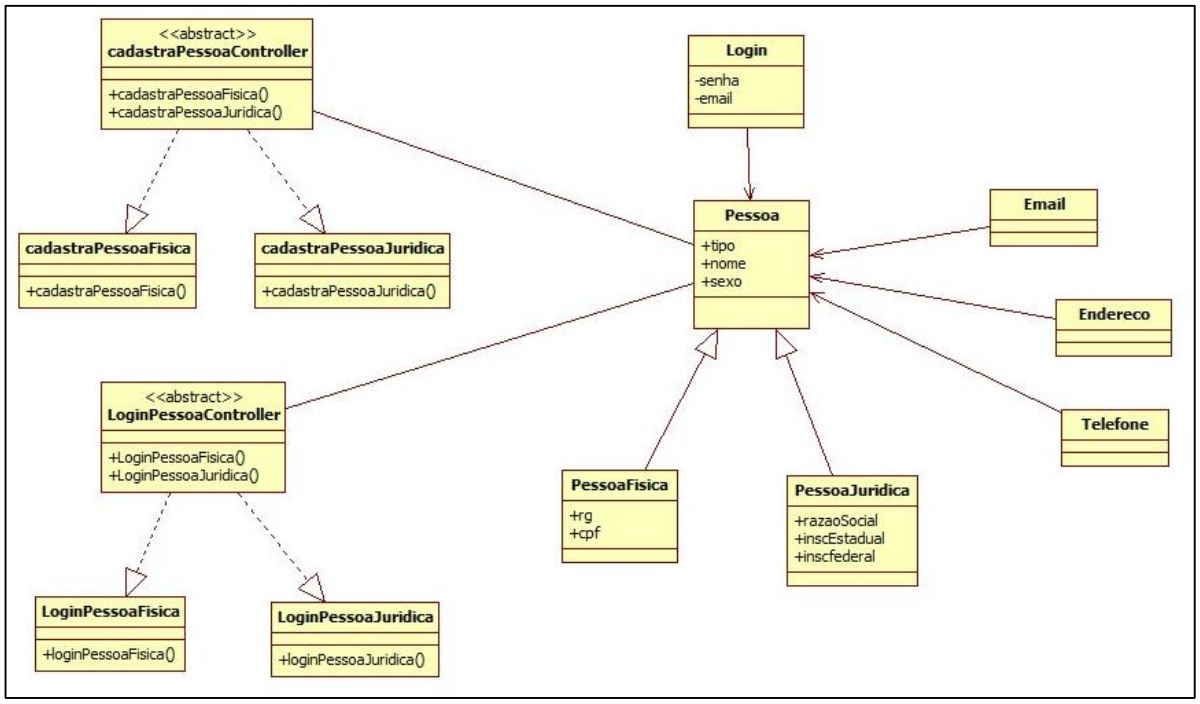

Figura 49 - Modelo Inicial Cadastro e *Login* de Usuários
Fonte: Autoria própria

Durante o processo de refatoração foi visto várias redundâncias neste modelo, tais como a classe *cadastraPessoaController* que possui dois métodos diferentes, sendo implementado cada um pelas classes *cadastraPessoaFisica* e *cadastraPessoaJuridica.* Isso também foi identificado para a classe *LoginPessoaController*. A Figura 50 mostra o modelo final com a refatoração para o cadastro e *login* de usuários.

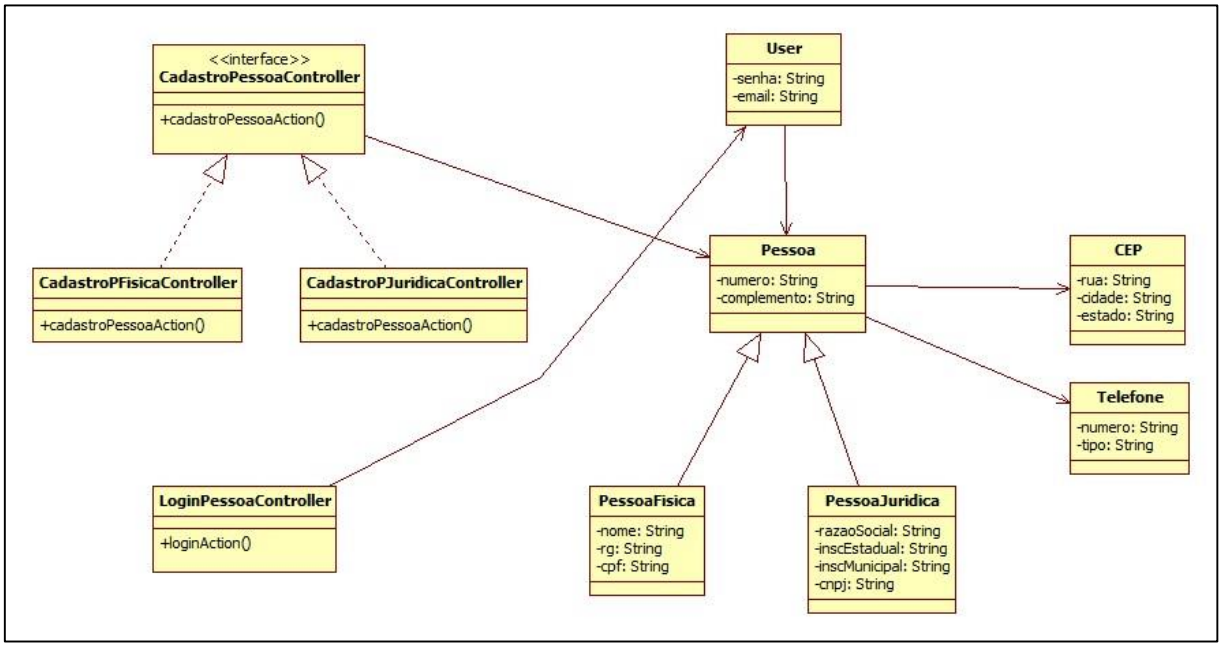

Figura 50 - Modelo Final Cadastro e *Login* de Usuários
Fonte: Autoria própria

Como pode-se notar comparando as Figuras 49 e 50, a classe *CadastroPessoaController* tornou-se uma interface, possuindo apenas um método como sua assinatura; sendo assim as classes *CadastroPFisicaController* e *CadastroPJuridicaController* implementam esses métodos de forma diferenciada. Outra alteração importante nesse modelo foi em relação a classe abstrata *LoginPessoaController* que possuía dois métodos e realizava duas classes: *LoginPessoaFisica* e *LoginPessoaJuridica*. Tendo visto que o método de autenticação de *login* não possuía diferença entre os tipos de pessoa, a classe *LoginPessoaController* passou a ser uma classe concreta contendo um método a ser implementado.

Outras refatorações foram a troca do nome da classe *Login* por *User* e a criação de uma relação direta com a classe *LoginPessoaController*; a classe *Email* e a classe *Endereço* deixou de existir, sendo o atributo *email* armazenado junto a classe *User*, e a classe *Endereço* passou a ser chamada de CEP, movendo os atributos número e complemento para a classe *Pessoa*.

No módulo de catálogo de produtos as classes responsáveis pelo armazenamento de informações no banco de dados são *Produto* e *Media*. As classes *CadastroProdutoController* é responsável pelo cadastro dos Produtos e as classes *CatalogoProdutoController*, *CatalogoProdutoB2BController* e *CatalogoProdutoB2CController* tem por objetivo fazer a listagem dos produtos. A Figura 51 mostra o modelo inicial do catálogo de produtos, nesta figura existe alguns erros de associação entre instâncias de objetos de uma classe em relação a outra e entre classes, por exemplo, *CatalogoProdutoController* com *CatalogoProdutoB2BController* e *CatalogoProdutoB2CController*. Este equívoco foi arrumado durante o processo de refatoração.

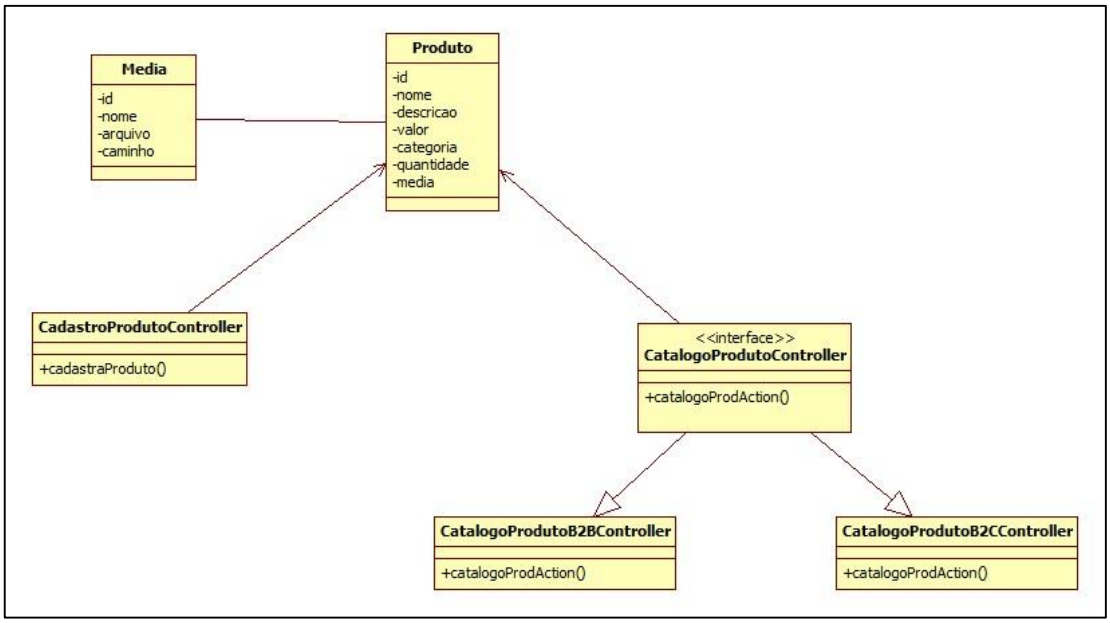

Figura 51 - Modelo Inicial Catálogo de Produtos
Fonte: Autoria própria

No processo de refatoração notou-se a falta de algumas classes, bem como métodos a serem mostrados. A Figura 52 mostra o modelo final do catálogo de produtos.

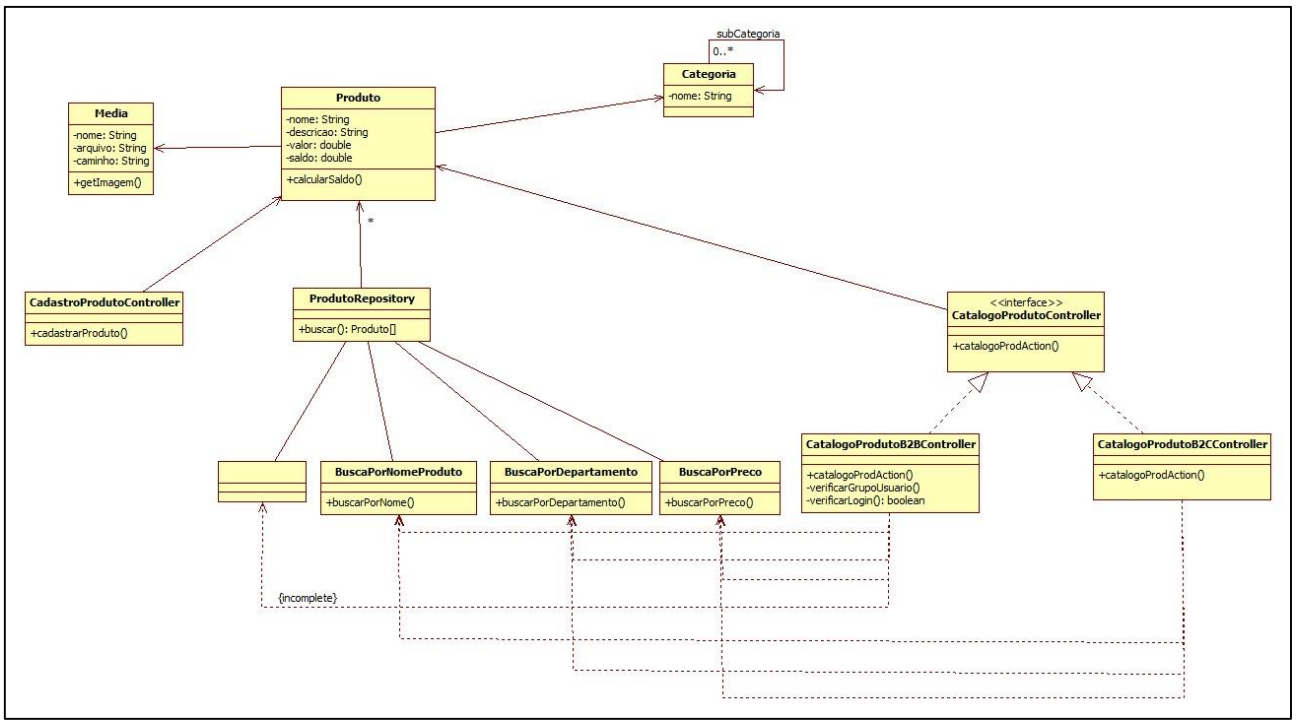

Figura 52 - Modelo Final Catálogo de Produtos
Fonte: Autoria própria

Comparando-se as Figuras 51 e 52, nota-se a inserção da classe *Categoria*, a qual possui um auto relacionamento, representando as subcategorias existentes; a

classe *ProdutoRepository* destina a realizar as consultas ao banco de dados e também as classes *BuscaPorNomeProduto*, *BuscaPorDepartamento*, *BuscaPorPreco*, as quais representam os tipos de buscas que podem ser realizadas. Nota-se que essas classes de buscas foram representadas de tal modo que seja possível incluir outros tipos em futuras implementações.

Outra diferença neste último modelo, foi a inclusão de dois métodos (*verificarGrupoUsuario* e *verficarLogin*) na classe *CatalogoProdutoB2BController*. Isto foi necessário pela categoria de *e-commerce* B2B poder apresentar diferença na listagem de produtos para o usuário, sendo assim foi possível abstrair os modelos de busca para que ambos funcionem nas categorias de *e-commerce* e apenas na classe *CatalogoProdutoB2BController* ocorre a diferenciação na implementação dos métodos.

Os módulos de relacionamento entre empresa e cliente, transação e entrega, seguiram o mesmo processo de refatoração estando presente os modelos finais nos Apêndices A e B deste trabalho.

O quadro 4 mostra os módulos trabalhados com seu respectivo número de vezes com que passaram por refatoração, e uma breve descrição sobre o que foi refatorado.

Módulo	**Número de refatorações**	**Descrição**
Cadastro e *Login* de Usuários	2	• Exclusão de classes redundantes e tornando-as atributos • Melhoramento do relacionamento entre as classes
Catálogo de Produtos	3	• Incluso a classe *Repositorio* para realizar consultas • Incluso as classes de busca • Inclusão de métodos diferenciados na categoria B2B
Relacionamento entre Cliente e Empresa	1	• Melhoramento da classe *Compra* para suprir ambas as categorias de *e-commerce*
Transação e Entrega	5	• Inclusão da classe *ItemCarrinho* • Inclusão da classe *Rota*; mudança dos atributos da classe *Transportadora* • Inclusão da classe *Compra* e *Venda* • Generalização das classes *Transportadora* e *Filial* como classe *PessoaJuridica* • Inclusão das classes: *ProdutosCompradosLoja, Movimentação e Saldo*

Quadro 4 - Módulos e Refatorações
Fonte: Autoria própria

5 RESULTADOS

Este capítulo apresenta os principais resultados obtidos com a aplicação do DDD para a modelagem de uma ferramenta capaz de gerar *e-commerce* B2B e B2C. A Seção 5.1 relata como foi a implementação do *cadastro e login de usuários* e *catálogo de produtos*, usando os modelos refatorado apresentados no Capítulo 4. A Seção 5.2 descreve um comparativo da ferramenta proposta com os trabalhos similares.

5.1 IMPLEMENTAÇÃO DOS MÓDULOS: CADASTRO E LOGIN DE USUÁRIOS E CATÁLOGO DE PRODUTOS

A página inicial desenvolvida para a ferramenta apresenta as opções de *e-commerce* disponíveis pela aplicação, ou seja, as categorias B2C e B2B, assim como mostra a Figura 53.

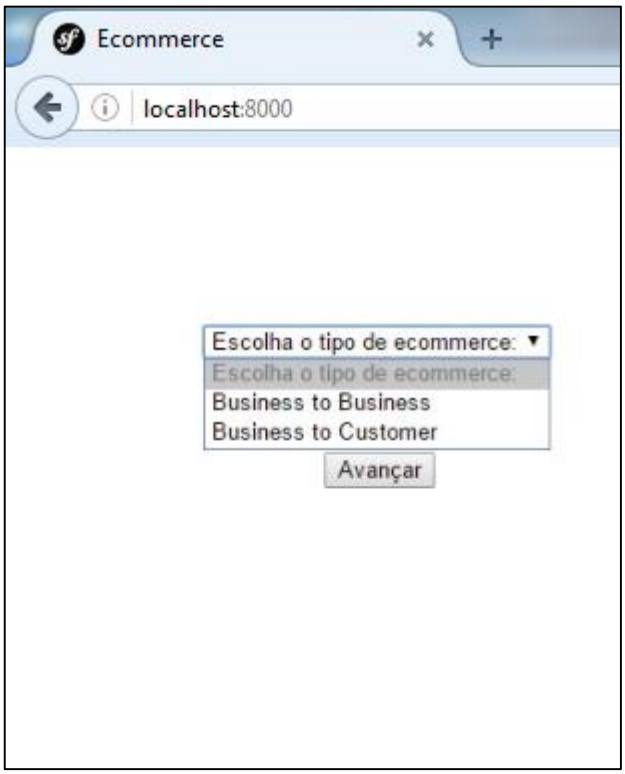

Figura 53 - Página Inicial da ferramenta
Fonte: Autoria própria

Ao selecionar uma das opções, o usuário é redirecionado para a página ilustrada na Figura 54 onde irá escolher quais os módulos que sua aplicação deverá possuir.

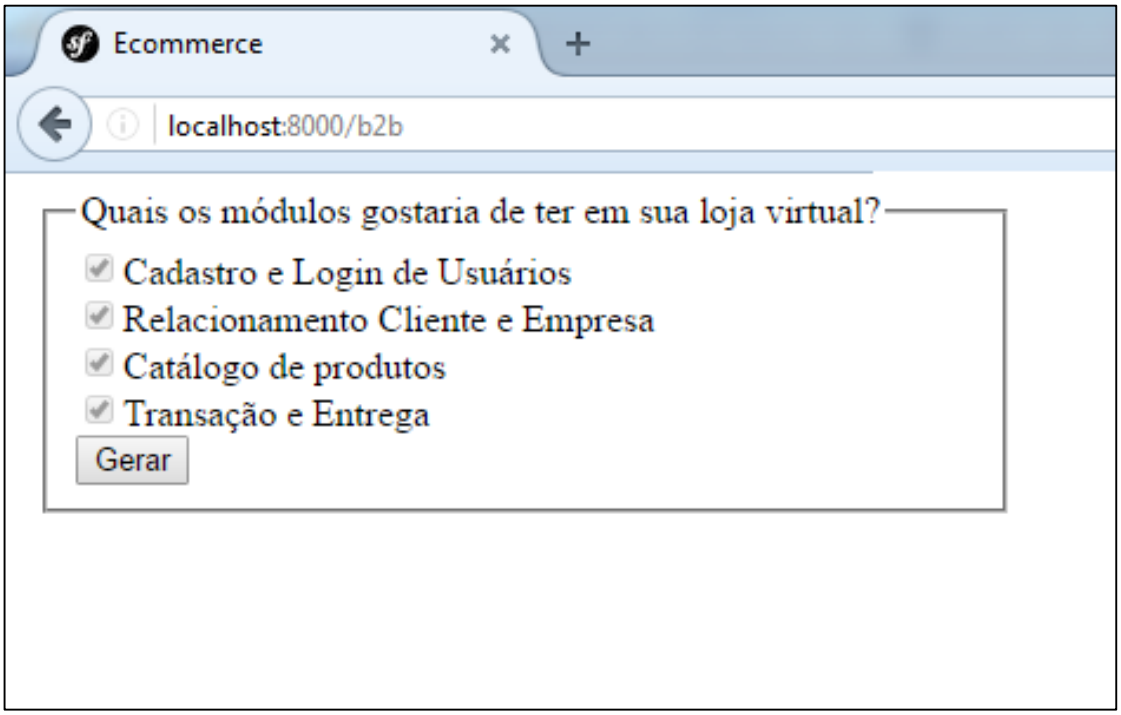

Figura 54 - Página para escolha dos módulos para o e-commerce
Fonte: Autoria própria

Ao pressionar em *Gerar*, a aplicação será gerada por meio da implementação de um *factory* correspondente a categoria de *e-commerce* selecionada, o modelo é apresentado na Seção 4.2.4 deste trabalho.

5.1.1 Cadastro de Usuários

O cadastro de usuários foi dividido em dois passos, no primeiro, como o apresentado na Figura 55, o usuário informa os campos referentes ao email, nome de usuário, senha e tipo de pessoa.

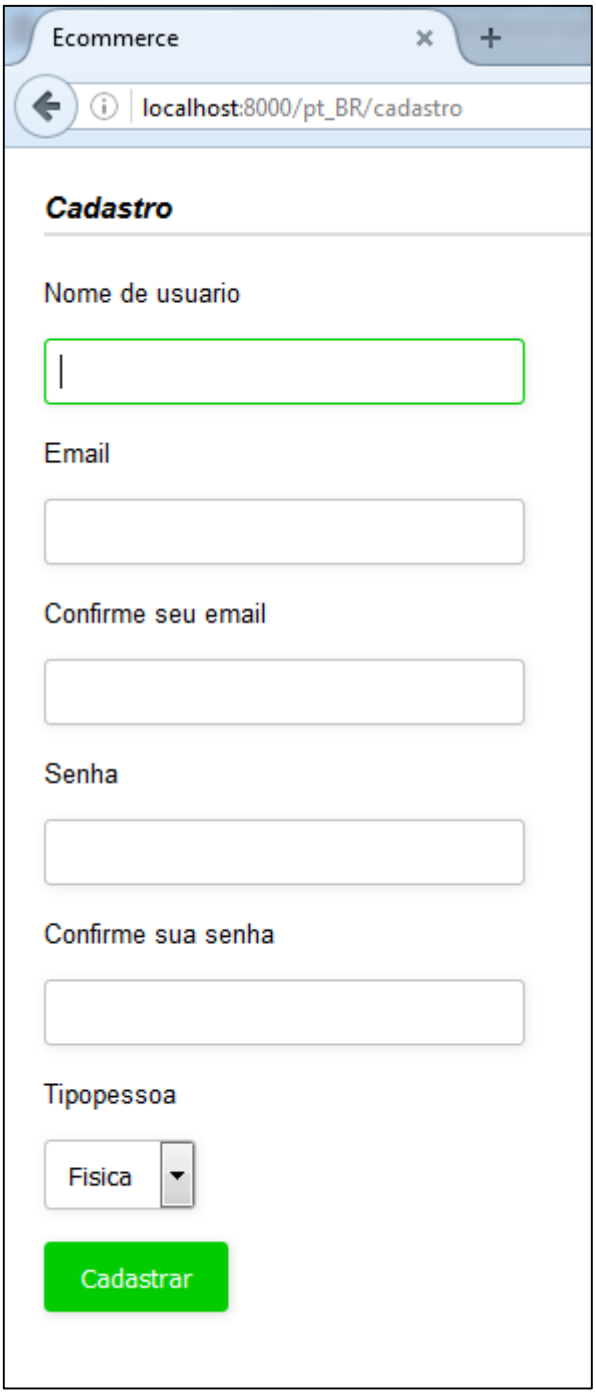

Figura 55 - Página de Cadastro de Usuário
Fonte: Autoria própria

Após a submissão deste formulário e a validação dos dados, o usuário é apresentado a um novo formulário referente ao tipo de pessoa escolhido no anterior. A Figura 56 exibe o formulário gerado quando da escolha do usuário é pessoa física.

Figura 56 - Página de Cadastro Pessoa Física
Fonte: Autoria própria

Ao submeter o segundo formulário, a ferramenta apresenta uma mensagem de sucesso e realiza a persistência dos dados, caso sejam válidos. Do contrário, o erro encontrado é informado ao usuário.

5.1.2 Login de Usuários

Para a funcionalidade de login de usuário criou-se um simples formulário com os campos de usuário e senha como mostrados na Figura 57.

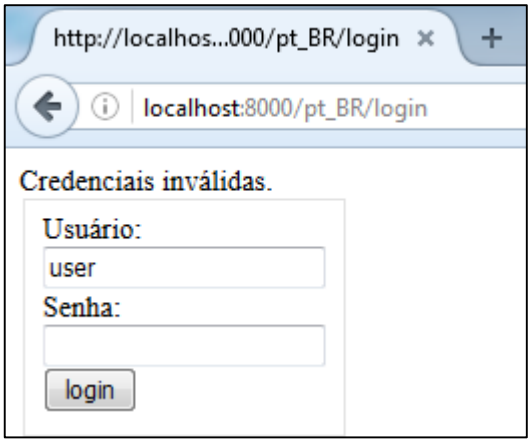

Figura 57 - Página Formulário de *Login*
Fonte: Autoria própria

O usuário informa os seus dados de login e os submete, caso informe os seus dados de maneira errônea, também é possível observar na Figura 57 a mensagem de erro mostrada pela aplicação. Em caso de credenciais válidas, o usuário é logado e redirecionado para a tela inicial da aplicação.

5.1.3 Cadastro de Produtos

Como parte importante do módulo de catálogo de produtos, foi implementado um formulário de cadastro de produtos como exibe a Figura 58, em uma aplicação real este formulário é disponibilizado somente por usuários administrativos. Este módulo não está totalemente implementado neste trabalho, apenas o cadastro do produto.

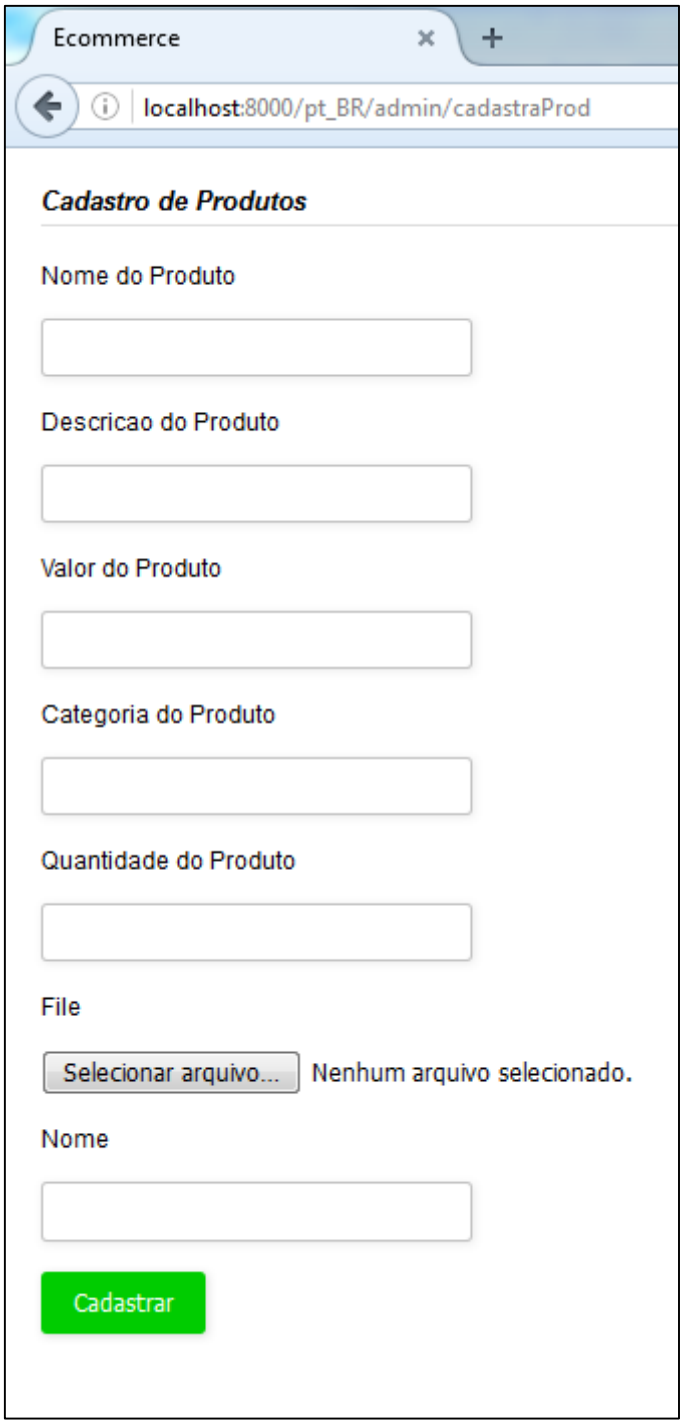

Figura 58 – Página do Cadastro de Produtos
Fonte: Autoria própria

O formulário contém campos referentes ao nome do produto, valor, categoria, quantidade, descrição, e uma opção para selecionar uma imagem. Após a submissão e validação do formulário, os dados são persistidos no banco e uma mensagem de sucesso é mostrada pela aplicação.

5.1.4 Catálogo de Produtos

Como foram encontradas diferenças para o catálogo de produtos entre as categorias B2C e B2B, foram criadas duas implementações diferentes para cada categoria *e-commerce*. Para o catálogo de produtos B2C foi implementado um catálogo em que o usuário pode visualizar os produtos com informações de preço disponíveis, independente se ele é um usuário que está logado no sistema ou não, ou seja, pode estar navegando anônimamente na aplicação e mesmo assim visualizar informações referentes ao preço dos produtos.

Na implementação do catálogo B2B, um usuário tem acesso limitado as informações dos produtos e precisa fazer parte de um grupo de usuários específico para ter acesso aos preços e outras informações importantes, geralmente este grupo de usuários ganha este acesso através de concessões administrativas. A regra de negócio irá depender da empresa que vai gerenciar o *e-commerce*.

Para o catálogo B2B desta aplicação é verificado se o usuário está logado e se ele pertence ao grupo privilegiado, caso contrário ele não terá acesso as informações referentes ao preço do produto.

A Figura 59 ilustra um exemplo de como é a exibição do catalogo B2B e a Figura 60 ilustra um exemplo de catalogo B2C.

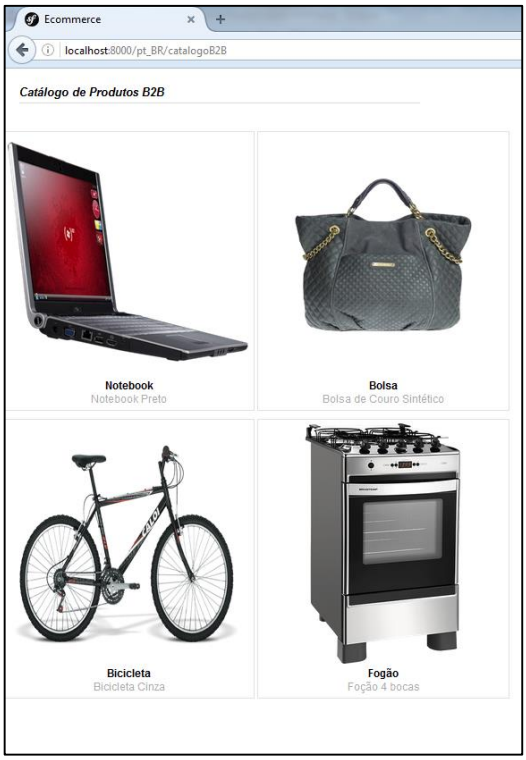

Figura 59 - Visualização do Catálogo de Produtos B2B
Fonte: Autoria própria

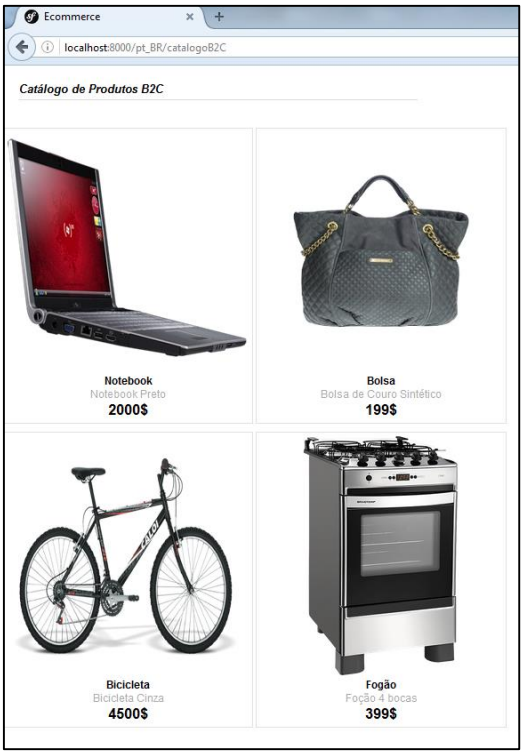

Figura 60 - Visualização do Catálogo de Produtos B2C
Fonte: Autoria própria

Também foi implementado no catálogo de produtos a funcionalidade de busca de produtos por nome para ambas as categorias como é apresentado na Figura 61 e na Figura 62.

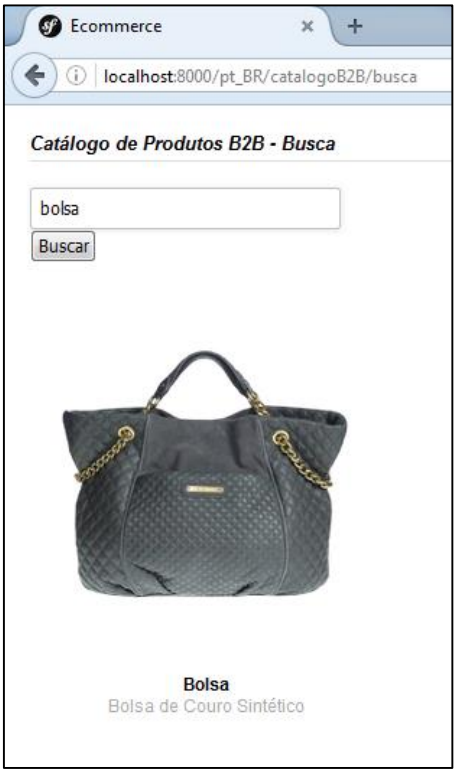

Figura 61 – Visualização da Busca B2B
Fonte: Autoria própria

Figura 62 – Visualização da Busca B2C
Fonte: Autoria própria

5.2 UMA BREVE COMPARAÇÃO DO MODELO PROPOSTO E TRABALHOS RELACIONADOS

Foram escolhidas duas entre as aplicações de código aberto mais utilizadas no mercado para realizar a comparação com a aplicação implementada neste trabalho, as aplicações escolhidas foram *Magento* e *Prestashop*.

A comparação foi realizada baseada na disponibilidade de funcionalidades apresentada pela aplicação aos usuários. O Quadro 5 apresenta as diferenças encontradas.

Possui?	Modelo Proposto	Magento	Prestashop
1. Diferentes categorias de *e-commerce*	Sim	Sim. Mas não no mesmo ambiente	Sim. Mas, não no mesmo ambiente
2. Módulos Personalizados	Sim	Não	Sim
3. Plataforma B2C gratuita	Sim	Sim	Sim
4. Plataforma B2B gratuita	Sim	Não	Sim
5. *Download* direto da aplicação	Não	Sim	Sim

Quadro 5 – Comparação de funcionalidades do modelo proposto e plataforma e-commerce: Magento e Prestashop
Fonte: Autoria própria

Ambas as aplicações oferecem diferentes categorias de *e-commerce* para os usuários, Magento oferece B2C e B2B de forma separada, o usuário tem acesso a uma aplicação B2C gratuita disponibilizada para a comunidade magento, mas também pode comprar as versões B2C e B2B pagas, sendo que esta última só possui uma versão paga.

Prestashop oferece um *download* direto da aplicação, o usuário pode alterar as configurações da aplicação para que ela se torne uma aplicação B2B. A aplicação gratuita possui módulos básicos, cabe aos usuários a opção de adquirir outros módulos pagos.

5.3 VANTAGENS NO USO DO DDD

Como uma plataforma de e-commerce envolve inúmeras transações importantes, a utilização do DDD se torna eficaz justamente porque o desenvolvedor pode focar no domínio do problema, onde podem ser identificadas todas as regras de negócios referentes as transações e demais funcionalidades.

Esta abordagem proporciona integridade ao sistema, pois levanta-se as entidades fundamentais para o domínio e a questão de implementação fica para ser resolvida em uma etapa posterior.

Os blocos de construção trazem princípios essenciais da orientação a objetos, desta forma o sistema é desenvolvido de forma clara, objetiva e organizada,

facilitando futuras manutenções e inclusive ao próprio processo de refatoração pertencente a abordagem do DDD.

6 CONCLUSÃO

Este trabalho criou um modelo para as categorias de *e-commerce* B2B e B2C utilizando DDD. Na primeira etapa do DDD foi estudado o domínio, por meio de consultas a *website* e experiências adquiridas durante o estágio curricular, e desenvolveu-se uma versão inicial do modelo sem muito refinamento. Nesta etapa foram usados os processos dos módulos: cadastro e *login* de usuários, relacionamento cliente e empresa, catálogos de produtos, transação e entrega, criados no Capítulo 2.

Na segunda etapa da aplicação do DDD construiu-se a o modelo usando sua arquitetura como base: interface, aplicação, domínio e infraestrutura. Para facilitar a implementação da arquitetura foi utilizado um framework denominado de *Symfony*. A terceira etapa foi aplicada realizando várias refatorações no modelo para deixá-lo mais flexível e reusável por meio da aplicação de padrões de projeto.

Para validar o modelo proposto foi implementado os módulos de cadastro e *login* e catálogo de produtos das categorias B2B e B2C os quais apresentavam características que eram iguais e similares, respectivamente.

O modelo da ferramenta proposta difere das plataformas de geração de *e-commerce* porque contempla duas categorias, diferente das outras que atendem apenas a uma das categorias. O modelo é flexível porque funcionalidades podem ser acrescentadas sem que haja grandes mudanças, isto mostrado nos diagramas por meio da palavra *{incomplete}*.

Embora a implementação do modelo não esteja completa, pode-se constar que a utilização do DDD para a criação do modelo inicial foi fundamental, pois focou-se em compreender as funcionalidades do domínio e o uso de um framework ajudou no processo de implementação do modelo, pois muitas tarefas puderam ser realizadas por ele.

6.1 TRABALHOS FUTUROS

Com base nos modelos criados e a implementação de dois deles, para um trabalho futuro pode-se continuar com a implementação dos módulos restantes,

tornando a plataforma e-commerce completa para uso nas categorias B2B e B2C. Também pode-se considerar a implementação das outras categorias C2C, B2G, C2G, pois o modelo inicial responsável por gerar a plataforma e-commerce abrange a inserção de novas categorias de e-commerce.

REFERÊNCIAS

ABComm. Associação brasileira de comércio eletrônico. **Pesquisa de plataformas e-commerce**. Disponível em: <http://www.abcomm.org/noticias/pesquisa-de-plataformas-de-e-commerce>. 2015. Acesso em: 17 set. 2015.

ADERMANN, Nils et al. **Dependency Manager for PHP.** 2016. Página para download do Composer. Disponível em: <https://getcomposer.org/download>. Acesso em: 15 mar. 2016.

ALBERTIN, Alberto Luiz; MOURA, Rosa Maria de. Comércio Eletrônico: mais evolução, menos revolução. **RAE - Revista de Administração de Empresas**. São Paulo, v.42, nº 3, p.114-117, Jul./Set. 2002.

ASCENSÃO, Carlos P. **O que é e-Commerce?**. 2014. Disponível em: <http://www.gestordeconteudos.com/tabid/3850/Default.aspx>. Acesso em: 24 nov. 2015.

BONIFACIO, Mauricio Di. **10 Diferenças entre uma plataforma de e-Commerce B2B e B2C**. 2015. Disponível em: <http://www.universob2b.com.br/artigos/ecommerce-b2b/10-diferencas-entre-uma-plataforma-de-e-commerce-b2b-e-b2c>. Acesso em: 14 nov. 2015.

BONIFACIO, Mauricio Di. **Diferenças entre B2B e B2C**. 2015. Disponível em: <http://www.universob2b.com.br/artigos/ecommerce-b2b/diferencas-entre-b2b-e-b2c>. Acesso em: 30 out. 2015.

BRANDOLINI, Alberto. **Strategic Domain Driven Design with Context Mapping**. 2009. Disponível em: <http://www.infoq.com/articles/ddd-contextmapping>. Acesso em: 29 out. 2015.

CARNEIRO, Eliane. **Criação de E-commerce com Magento ou Prestashop:** Qual é a melhor plataforma?. 2015. Disponível em: <http://b4w.com.br/news/ecommerce-comercio-eletronico/criacao-de-e-commerce-com-magento-ou-prestashop-qual-e-a-melhor-plataforma>. Acesso em: 13 nov. 2015.

COSTA, Paulo N. **O que é WordPress? Saiba suas principais características**. 2013. Disponível em: <http://www.2wp.com.br/artigos/o-que-e-wordpress>. Acesso em: 13 nov. 2015.

CUKIER, Daniel. **DDD – Introdução a Domain Driven Design**. AgileAndArt. jul/ago. 2010. Disponível em: <http://www.agileandart.com/2010/07/16/ddd-introducao-a-domain-driven-design/>. Acesso em: 18 nov. 2015.

DOTSTORE (Mogi das Cruzes). **E-Commerce 2016**: Projeção de R$56,8 bilhões em faturamento. 2016. Disponível em: <http://site.dotstore.com.br/loja-virtual/e-commerce-2016-projecao-de-r568-bilhoes-em-faturamento/>. Acesso em: 12 maio 2016.

E-BIT. **Prêmio 2013 Excelência em Qualidade Comércio Eletrônico B2C**. 2013. Lista de lojas premiadas. Disponível em <http://www.ebit.com.br/premiacao-exelencia-em-qualidade-2013>. Acesso em: 21 nov. 2015.

_____. **Relatório WebShoppers 2014**. 2014. Arquivo do relatório. Disponível em <http://img.ebit.com.br/webshoppers/pdf/WebShoppers2014_2oSeme.pdf>. Acesso em: 17 set. 2015.

E-COMMERCE.ORG. **Evolução da Internet e do e-commerce**. 2015. Disponível em: <http://www.e-commerce.org.br/stats.php>. Acesso em: 17 out. 2015.

EVANS, Eric. **Domain-Drive Design: Tackling Complexity in the Heart of Software**. Prentice Hall, 2003.

EVIGO.COM. **Forrester: ranking of the best B2B commerce suite vendors, IBM took first place**. 2013. Disponível em: <http://evigo.com/5899-forrester-ranking-best-b2b-commerce-suite-vendors-ibm-took-first-place>. Acesso em: 14 nov. 2015.

FABIEN POTENCIER (California) (Org.). **SYMFONY.** Página do framework. Disponível em: <http://symfony.com/>. Acesso em: 15 mar. 2016.

FOWLER, Martin. **Catalog of Refactorings.** 2013. Catálogo de refatorações. Disponível em: <http://refactoring.com/catalog/>. Acesso em: 05 maio 2016.

GAMMA, Erich *et al*. **Padrões de Projeto:** Soluções Reutilizaveis de Software Orientado a Objetos. Porto Alegre: Bookman, 2007. 350 p. Tradução Luiz A. Meirelles.

GEARY, David. **Strategy for sucess**: The powerful Strategy design pattern ais object-oriented design. Java World, abr./jun. 2002. Disponível em: <http://www.javaworld.com/article/2074195/swing-gui-programming/strategy-for-success.html>. Acesso em: 06 nov. 2015.

GOCACHE. **Woocommerce x Magento: Compare e escolha a melhor plataforma para seu negócio online**. 2015. Disponível em: <http://www.gocache.com.br/ecommerce/woocommerce-x-magento-compare-e-escolha-a-melhor-plataforma-para-o-seu-negocio-online>. Acesso em: 13 nov. 2015.

GPA. **Ponto Frio.** 2015. Disponível em: <http://www.pontofrio.com.br>. Acesso em: 21 nov. 2015.

HEDIN, Görel. **Software Architectures**: Patterns and Frameworks. 2000. Disponível em: <http://fileadmin.cs.lth.se/cs/Personal/Gorel_Hedin/SoftArch/ArchPatFW.html>. Acesso em: 06 nov. 2015.

IBM. **International Business Machines.** 2015. Página da IBM. Disponível em: <http://www.ibm.com/us-en>. Acesso em: 20 nov. 2015.

JAMEF. **Central de Relacionamento**: Dúvidas Frequentes. 2015. Disponível em: <http://www.jamef.com.br/jamef/ecp/comunidade.do?&app=portal&pg=20004&view=faq>. Acesso em: 05 nov. 2015.

JILL. **Infographic: Magento History and Evolution**. 2014. Disponível em: <http://sherodesigns.com/infographic-magento-history>. Acesso em: 02 nov. 2015.

KHAN, Aslam. **Getting Started with Domain-Driven Design**: Object Orientation Done Right. 2105. DZone. Disponível em <https://dzone.com/refcardz/getting-started-domain-driven>. Acesso em 23 nov. 2015.

MENDES, Renan. **O dinheiro de verdade está no mercado de E-commerce B2B**. 2015. Disponível em: <http://www.profissionaldeecommerce.com.br/dinheiro-mercado-e-commerce-b2b>. Acesso em: 10 nov. 2015.

MONTEIRO, Keine. **A líder China**. 2015. Disponível em: <http://www.profissionaldeecommerce.com.br/lider-china>. Acesso em: 02 nov. 2015.

PAGSEGURO. **PagSeguro**: Sua solução completa para pagamentos. 2015. Disponível em: <https://pagseguro.uol.com.br/atendimento/perguntas_frequentes/categoria/seguranca/pela-internet/o-que-e-pagseguro.jhtml#rmcl>. Acesso em: 22 nov. 2015.

PETER THIEL (California). **PayPal.** 2015. Disponível em: <https://www.paypal.com/br/home>. Acesso em: 22 nov. 2015.

PORTALDACOMUNICACAO. **Prêmio iBest revela os 10 melhores sites de comércio eletrônico**. 2015. Disponível em: <http://portaldacomunicacao.uol.com.br/graficas-livros/0/artigo201511-1.asp>. Acesso em: 14 nov. 2015.

QIN, Z. **Introduction to E-commerce.** Beijing: Springer Verlay, 2009.

SILVA, Edson dos Santos. **Branding para empresas B2B**. Fundação Instituto de Administração, Pós Graduação em Gestão Estratégica de Marcas. São Paulo: Provar, 2013. Disponível em: <http://www.cidademarketing.com.br/2009/sysfotos/tesesmono/tesem_40b478b412cce7278d88fb18869ca034.pdf>. Acesso em: 17 out. 2015.

TASSABEHJI, R. **Applying E-Commerce for Business.** London: SAGE Publications Ltd, 2003.

VALLE, Alberto. **O que é Plataforma de E-commerce?** 2015. Disponível em <http://www.cursodeecommerce.com.br/blog/o-que-e-plataforma-de-ecommerce>. Acesso em: 17 set. 2015.

ZEND TECHNOLOGIES (Comp.). **Zend Framework 2.** 2016. Página do framework. Disponível em: <http://framework.zend.com>. Acesso em: 08 mar. 2016.

APÊNDICE A - Modelo relacionamento cliente/empresa

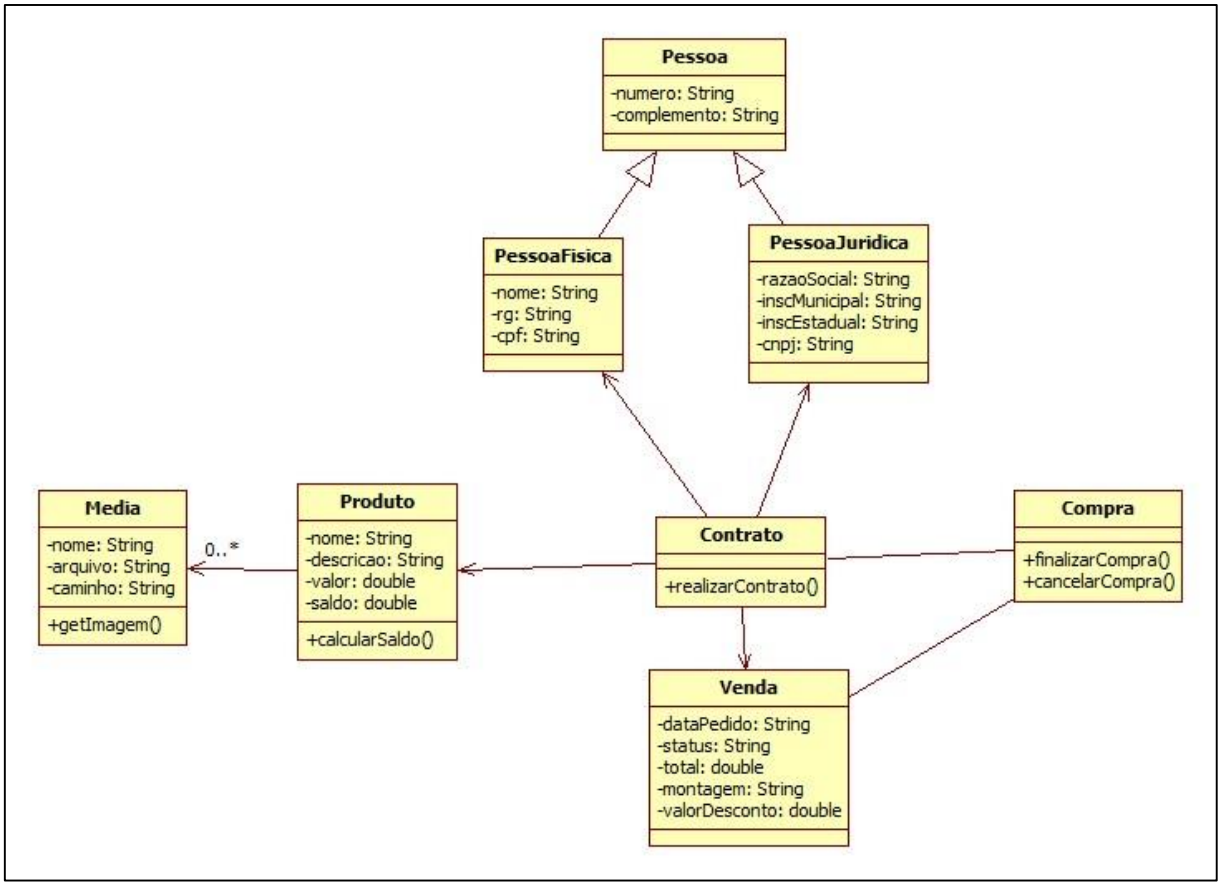

Figura 63 - Modelo Relacionamento Cliente/Empresa
Fonte: Autoria própria

APÊNDICE B - Modelo transação e entrega

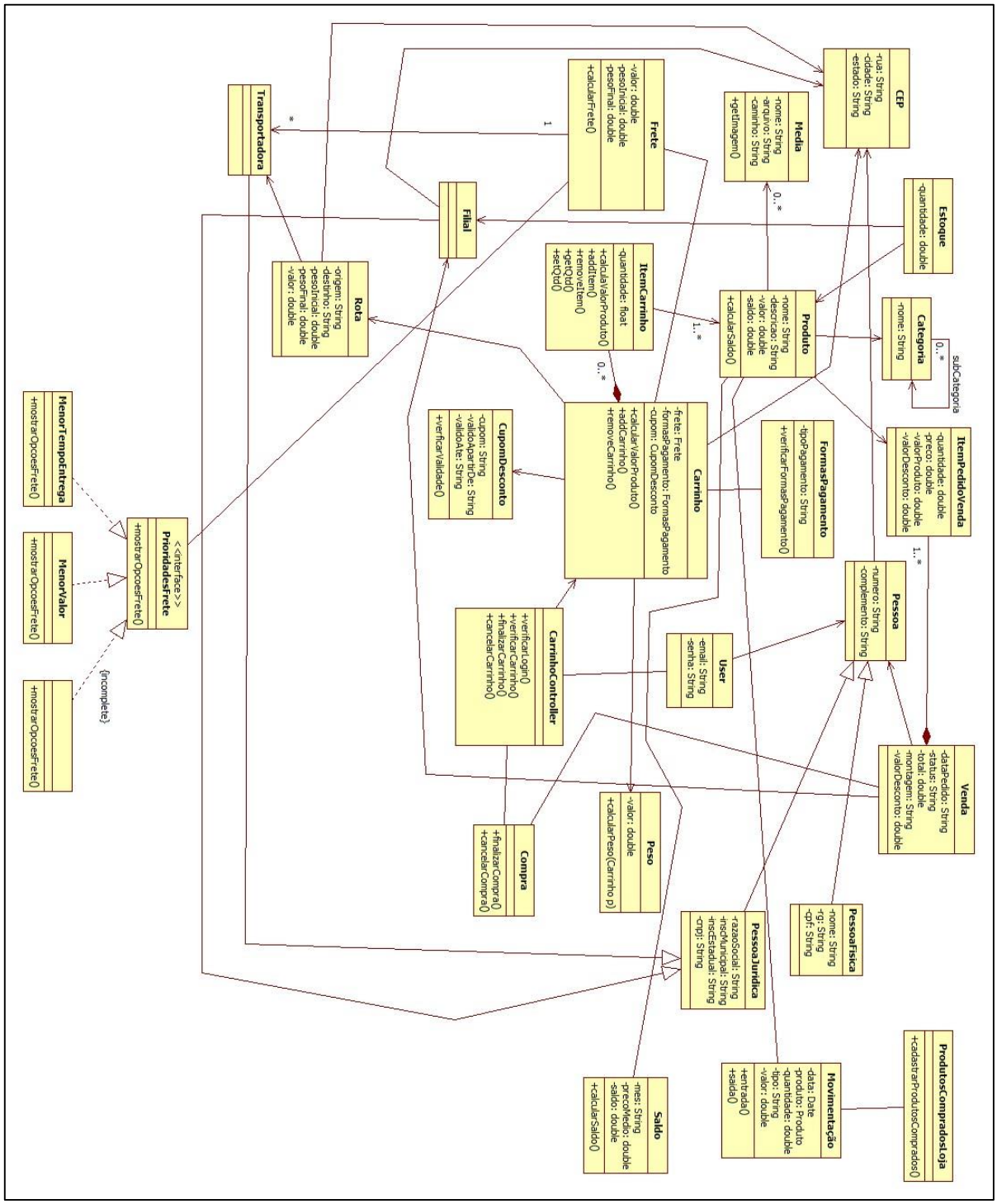

Figura 64 - Modelo Transação e Entrega
Fonte: Autoria própria