Universidade Federal de Pernambuco

Bacharelado em Sistemas de Informação
Centro de Informática
2016.1

DESENVOLVIMENTO DE UMA FERRAMENTA DE ESTRATÉGIA DO SERVIÇO DE PÓS-VENDA PARA AUXILIAR LOJISTAS DO E-COMMERCE

Aluno: Jorge Rodrigues Gomes Vaz Filho (jrgvf@cin.ufpe.br)
Orientador: Profº Dr. Vinicius Cardoso Garcia (vcg@cin.ufpe.br)

Recife, 11 de julho de 2016

Universidade Federal de Pernambuco

Bacharelado em Sistemas de Informação
Centro de Informática
2016.1

Jorge Rodrigues Gomes Vaz Filho

DESENVOLVIMENTO DE UMA FERRAMENTA DE ESTRATÉGIA DO SERVIÇO DE PÓS-VENDA PARA AUXILIAR LOJISTAS DO E-COMMERCE

Trabalho apresentado à disciplina de Trabalho de Graduação em Sistemas de Informação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: *Profº Dr. Vinicius Cardoso Garcia*

Recife, 11 de julho de 2016

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter me dado saúde, pela oportunidade de estudar na Universidade Federal de Pernambuco e por me dar forças para persistir no curso até aqui.

A minha família, amigos e namorada, por sempre me darem total apoio e incentivo para que eu pudesse chegar até aqui, me motivando e apoiando com os obstáculos que apareceram e com minhas decisões.

Aos professores e monitores, por todo o conhecimento passado e tempo dedicado ao longo do curso. Em especial ao Profº Dr. Vinicius Cardoso Garcia, que me orientou neste trabalho com paciência.

Ao Centro de Informática (CIn-UFPE), por ter sido um local de muito aprendizado e crescimento no âmbito profissional e acadêmico. Assim como aos meus colegas do curso de Sistemas de Informação, por toda a ajuda durante todos os períodos que passamos juntos, principalmente àqueles com quem formei grupos.

A todos que direta ou indiretamente contribuíram com a minha formação, dedico este trabalho e me sinto muito feliz por ter chegado até aqui.

RESUMO

O e-commerce surgiu na década de 70, cresceu rapidamente e em pouco tempo ficou disponível para o público em geral. Hoje, qualquer pessoa pode abrir o seu próprio e-commerce, e passar a vender na internet com muito mais simplicidade do que em uma loja física. Essa simplicidade levou a um constante crescimento no faturamento do setor de comércio eletrônico, atraindo a todo instante novos empreendimentos e, consequentemente, aumentando a concorrência. Com esse aumento, o sucesso ou falha poderá ser determinado pelos diferenciais percebidos pelos consumidores.

A experiência de compra no e-commerce compreende todo um processo que tem início desde antes da compra propriamente dita, passa pelo pagamento, envio, entrega da mercadoria e o pós-venda. Um dos diferenciais que vem crescendo e ganhando atenção é o pós-venda, que ocorre após a transação ter sido efetuada e o cliente receber o produto.

Alguns fatores podem fazer com que um cliente jamais volte a comprar do mesmo lojista e ainda passe uma imagem ruim da loja para outros consumidores. Alguns desses fatores são: qualidade do produto, estado de conservação e adequação ao cliente. Problemas assim poderiam ser evitados se o lojista estabelecesse uma forma de acompanhamento do pós-venda de seus clientes, contudo alguns não dão atenção necessária a essa etapa do processo e não exploram o potencial de fidelização dos consumidores. Outros, até acreditam que o processo é encerrado após o envio da mercadoria. Porém, para aqueles que procuram realizar a etapa do pós-venda, algumas dificuldades são encontradas, tais como: informação descentralizada, contato espalhado em múltiplos canais e processo manual.

O objetivo deste projeto é desenvolver uma ferramenta web para auxiliar os lojistas do e-commerce a realizarem a etapa do pós-venda. A ideia é realizar o envio de notificações para os consumidores, com perguntas sobre a venda, tornando possível a análise das respostas recebidas, possibilitando a identificação dos problemas encontrados pelos consumidores. Após essa etapa, o comerciante poderá atuar mais rapidamente nesses pontos fracos. A ferramenta permitirá aos lojistas que, com poucas configurações, automatizem o processo desde a importação de dados até o envio das notificações.

Palavras-Chave: E-commerce, Pós-venda, Acompanhamento do pós-venda, Identificação dos problemas, Diferencial, Fidelização dos consumidores.

ABSTRACT

The e-commerce emerged in the 70s, grew rapidly and soon became available to the general public. Today, anyone can open your own e-commerce, and to start selling on the Internet with more simplicity than in a physical store. This simplicity led to a steady growth in sales of e-commerce industry, attracting all the time new projects, and thereby increasing competition. With this increase, the success or failure can be determined by differential perceived by consumers.

The shopping experience in e-commerce comprises a whole process that begins long before purchase itself, through the payment, shipping, delivery of goods and after-sales. One of the differences that is growing and gaining attention is the after-sales, which occurs after the transaction has been made and the customer receives the product.

Some factors may cause a customer never again buy the same shopkeeper and still pass a bad image store for other consumers. Some of these factors are: product quality, condition and suitability to the customer. Problems like this could be avoided if the shopkeeper establish a form of monitoring of aftermarket customers, but some do not give necessary attention to this step of the process and do not exploit the potential of loyalty of consumers. Others even believe that the process is terminated after sending the goods. But for those looking to take the step of after-sales, some difficulties are encountered, such as decentralized information, contact spread across multiple channels and manual process.

The goal of the project is to develop a web tool to help retailers e-commerce to conduct post-sales stage. The idea is to perform sending notifications to consumers with questions about the sale, making possible the analysis of the responses received, enabling the identification of the problems encountered by consumers. After this step, the trader may act quickly on these weaknesses. The tool will allow shopkeepers with few settings , automate the process from importing data to the sending of notifications.

Keywords: E-commerce, After-sales, Monitoring after-sales, Identification of problems, Differential, Loyalty of consumers.

LISTA DE FIGURAS

SUMÁRIO

1. Introdução

O rápido crescimento do faturamento do e-commerce desperta o interesse dos comerciantes e a cada dia o número de lojistas no comércio eletrônico brasileiro só faz crescer. Devido a esse crescimento a concorrência aumenta e os diferenciais vão determinar o sucesso ou fracasso do empreendimento. Esse trabalho surgiu com o interesse em ajudar os lojistas a realizarem a etapa de pós-venda e se destacarem com esse diferencial, sendo direcionado ao desenvolvimento de uma ferramenta web com possibilidade de automação dos processos. A oportunidade de poder investir e explorar essa área coincide com a área de atuação e interesse do autor, despertando o interesse em desenvolver uma ferramenta de estratégia do serviço de pós-venda para auxiliar lojistas do e-commerce.

Nesta seção, serão vistos a motivação, o problema de pesquisa, os objetivos e a estrutura deste documento.

1.1 Motivação

Em uma área em que a concorrência aumenta a cada dia é crucial apresentar diferenciais aos consumidores em busca da sobrevivência. A fidelidade dos consumidores em relação as empresas está cada vez menor, pois há uma maior oferta no mercado, aumentando a concorrência e tornando os clientes mais exigentes. O mercado e as necessidades do consumidor mudam em um ritmo acelerado, diante disso uma oportunidade que as empresas encontram de fidelizar um consumidor tão volátil é realizando o pós-venda de forma efetiva [1].

Manter uma base de clientes e conquistar novos clientes não é simples, muitos empresários passaram os últimos anos investindo em novas maneiras de vender seus produtos e negligenciaram o contato com seus consumidores.

Para uma parte dos lojistas o foco se traduz no aumento dos lucros, enquanto para outros o foco se traduz na diferenciação dos concorrentes e na valorização da empresa. O caminho para fracasso pode ser determinado por pontos fracos na operação que não são encontrados e pouco a pouco acabam desvalorizando a empresa e colocando abaixo dos seus concorrentes.

O caminho para o sucesso pode ser facilitado quando é estabelecida alguma forma de acompanhamento do processo de venda. Se a empresa lidar com as dificuldades e

problemas encontrados pelos consumidores de forma pró-ativa ela estará diferenciando-se dos concorrentes e valorizando sua empresa.

Pós-venda é a etapa do processo que ocorre quando a transação já foi efetuada e o consumidor recebeu seu produto. O pós-venda está sendo cada vez mais praticado para descobrir os pontos fracos do processo, refletir e agir sobre a melhoria do processo de venda. Muitas vezes o pós-venda é o que garante a fidelização do cliente e o fator determinante entre o sucesso e o fracasso.

O E-Commerce Brasil, projeto que apoia o e-commerce no Brasil através de portal, revista, congressos, fórum, treinamentos, convenções, conferências e outras atividades, disponibiliza constantemente artigos para ajudar os lojistas em sua operação. Dentre esses artigos, aparecem muitos artigos relacionados a retenção de cliente e o pós-venda. Em abril de 2015 foram apontados 9 sinais ‘de que chegou a hora de investir em retenção de clientes’, dentre os sinais alguns refletem a motivação desse projeto, como a falta de pós-venda e pensamentos como: automação é despesa e a culpa é dos outros [2].

O pós-venda por muitas vezes é o que garante a fidelização do cliente e valorização da empresa. A automação libera o empresario para executar outras tarefas, analisar o perfil dos clientes, acompanhar o andamento do seu processo e criar novas estratégias para melhoria contínua. Diante das dificuldades do mercado é necessário ficar atento a possíveis erros cometidos e buscar corrigi-los o mais rápido possível.

Dessa forma esse projeto tem como principal motivação investigar as dificuldades encontradas pelos novos e antigos lojistas em sobreviver ao mercado do e-commerce brasileiro.

1.2 Problema de Pesquisa

Com o interesse em identificar e analisar as características fundamentais necessárias para uma ferramenta web de estratégia do serviço de pós-venda para auxiliar lojistas do e-commerce, este trabalho pretende investigar: Quais características e funcionalidades uma ferramenta web precisa ter para apoiar os lojistas do e-commerce a aplicar o serviço de pós-venda?

1.3 Objetivos

Este trabalho tem como objetivo geral implementar uma ferramenta web que possibilite a criação de estratégias do serviço de pós-venda automatizada, enviando

notificações ao consumidores, coleta e análise das respostas. A ferramenta proverá os meios de envio das notificações, coleta de feedback dos consumidores e disponibilizará os resultados na aplicação.

Para alcançar o objetivo geral, alguns objetivos específicos foram definidos:

1. Integração direta com a base de dados do lojista para importação dos dados dos consumidores e das vendas.
2. Automatização de fluxos de execução dentro da ferramenta de forma que execute todas as atividades necessárias sem precisar de interação com o usuário.
3. Diferentes meios de notificação para aumentar as chances de coleta de feedback dos consumidores.

1.4 Estrutura do Documento

Os capítulos seguintes estão estruturados da seguinte forma:

Capítulo 2: Apresenta a contextualização deste trabalho, fundamento para qualquer leitor sobre os conceitos utilizados neste trabalho. Esses conceitos são relacionados ao e-commerce e ao pós-venda.

Capítulo 3: Apresenta e descreve a ferramenta criada nesse trabalho.

Capítulo 4: Apresenta as considerações finais referentes aos objetivos atingidos, bem como uma análise de limitações do projeto.

2. Contextualização

Neste capítulo serão abordados os conceitos essenciais para o entendimento deste trabalho. Serão descritos os principais conceitos relacionados à e-commerce e pós-venda, para apoiar o entendimento do que o projeto implementado se propõe a fazer e justificar suas motivações.

Este capítulo está estruturado da seguinte maneira: a Seção 2.1 apresenta os conceitos e definições sobre e-commerce, a Seção 2.2 apresenta os conceitos e definições sobre pós-venda e a Seção 2.3 apresenta algumas considerações.

2.1 E-commerce

O e-commerce está em constante mudança e as classificações e definições acabam explicando uma perspectiva que não englobará tudo o que o e-commerce possui, um exemplo de definição é:

"O comércio eletrônico é a realização de toda a cadeia de valor dos processos de negócio em um ambiente eletrônico, por meio da aplicação intensa das tecnologias de informação e de comunicação, atendendo aos objetivos de negócio." - Albertin (1999, p 15)

Tecnologias como *Electronic Data Interchange* (EDI) e *Eletronic Funds Transfer* (EFT) surgiram na década de 60, e criaram condições para que usuários trocassem informações financeiras e de negócios de qualquer lugar do mundo [3]. Na década de 70, Michael Eldrich, inventor e empreendedor inglês, inventou o teleshopping conectando uma TV a um computador de processamento em tempo real utilizando uma linha telefônica [4]. Mas o início efetivo do e-commerce se deu em 1991, quando a internet foi aberta para uso comercial e impactou o mundo inteiro [3].

A evolução do e-commerce é diretamente ligada a evolução da Internet, consequentemente o período da bolha da internet alavancou o crescimento do e-commerce, assim como o *Payment Card Industry Data Securiry Standard* (PCI DSS) em 2004 [5].

O crescimento das compras na internet mudou o comportamento dos consumidores, as pessoas estão mais confiantes ao fazer compras online e passaram a utilizar a internet também para decidir onde farão algumas compras em lojas físicas. Com o tempo os consumidores vão cobrando mais qualidade e agilidade das lojas virtuais.

2.1.1 Tipos de E-commerce

Hoje o e-commerce engloba todo tipo de produto que se possa imaginar, basta procurar e encontrará alguém vendendo o que você deseja. Devido a enorme possibilidade de negócios que o e-commerce proporciona, surgiram diversos tipos de e-commerce, entre eles alguns se destacam: B2C, B2B, B2G, B2E, B2B2C, C2B e C2C [6, 7].

B2C (*Business to Consumer*): É o tipo mais comum, a venda a varejo. É quando a venda acontece da loja diretamente ao cliente final.

B2B (*Business to Business*): Nesse tipo o negócio acontece entre empresas, a diferença se dá porque existem detalhes adicionais nesse tipo de negociação, como níveis de desconto pelo porte das compras e impostos diferenciados.

B2G (*Business to Gogivernment*): Acontece quando uma empresa vende para o governo, se diferencia do B2B pois existem diversas regulamentações e regras que desem ser respeitadas por imposições de lei.

B2E (*Business to Employee*): Esse tipo é semelhante ao B2C, a diferença que ocorre quando a empresa realiza a venda a seus próprios funcionários. Normalmente os preços são diferenciados e existem limites de compra.

B2B2C (*Business to Business to Consumer*): Acontece quando uma empresa faz negócios com outra empresa visando uma venda para o cliente final.

C2B (*Consumer to Business*): Esse é um modelo onde os consumidores que ofertam seus produtos e serviços para as empresas, pouco utilizado no Brasil, mas já possui um grande mercado em outros países. Um exemplo é a oferta de serviços freelancer.

C2C (*Consumer to Consumer*): Acontece quando a relação comercial se dá entre duas pessoas, através de uma plataforma de e-commerce que promove a intermediação da negociação.

2.1.2 E-commerce no Brasil

Com o crescimento de usuários com acesso a internet no Brasil e o crescimento do números de dispositivos móveis o e-commerce no Brasil vem tendo um bom crescimento. A seguir segue um resumo dos anos de 2012 até 2015 dos números do e-commerce B2C brasileiro de acordo com os estudos da E-bit/Buscapé.

2012: O e-commerce brasileiro fechou o ano com R$ 22,5 bilhões de faturamento, com um total de 66,7 milhões de pedidos e um registro de consumidores virtuais de 42,2 milhões de pessoas que tinham feito, ao menos, uma compra online. As cinco categorias com maior volume de pedidos foram: 'Eletrodomésticos' (12,4%), 'Moda e Acessórios' (12,2%), 'Saúde, beleza e medicamentos' (12%), 'Informática' (9,1%) e 'Casa e Decoração' (7,9%) [8].

2013: O e-commerce brasileiro fechou o ano com R$ 28,8 bilhões de faturamento, apresentando 28% de crescimento em relação a 2012. Com um total de 88,3 milhões de pedidos e um registro de consumidores virtuais de 51,3 milhões. As cinco categorias com maior volume de pedidos foram: 'Moda e Acessórios' (19%), 'Saúde, beleza e medicamentos' (18%), 'Eletrodomésticos' (10%), 'Livros e Revistas' (9%) e 'Informática' (7%). [9]

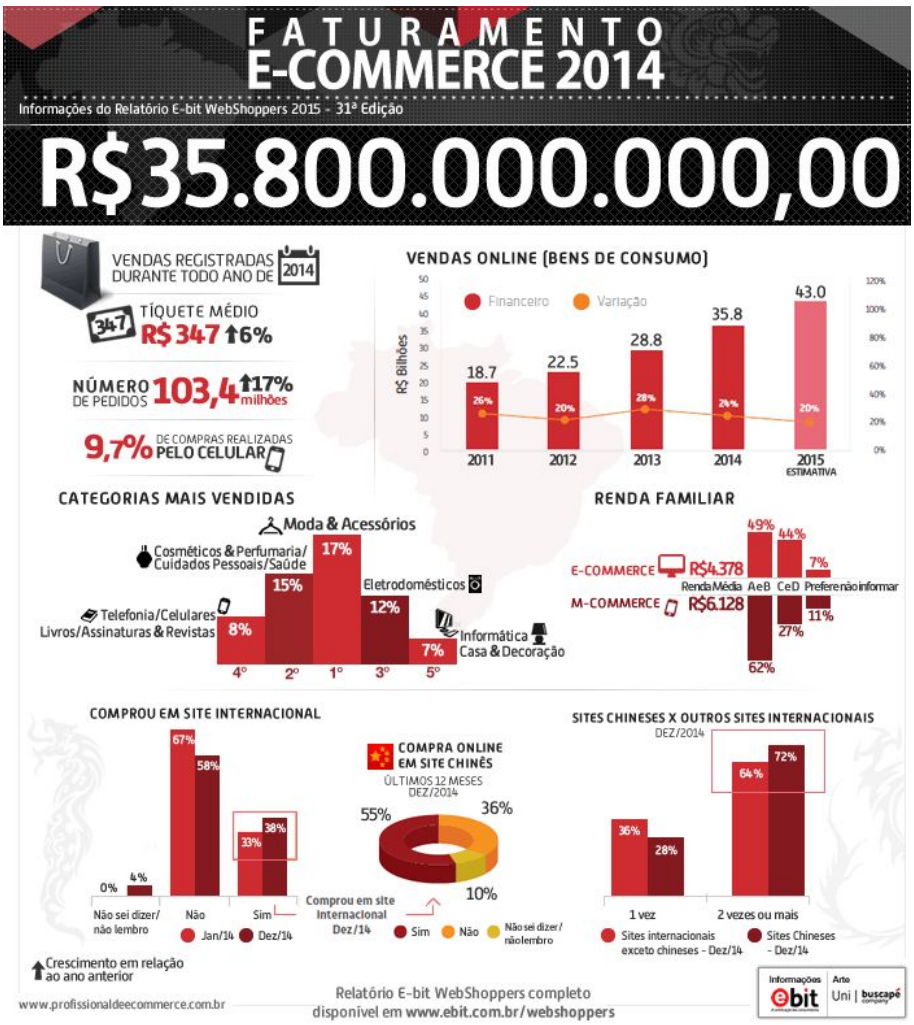

Figura 1. Faturamento e-commerce 2014 [10]

2014: O e-commerce brasileiro fechou o ano com R$ 35,9 bilhões de faturamento, apresentando 17% de crescimento em relação a 2013. Com um total de 103,4 milhões de pedidos e um registro de consumidores virtuais de 61,6 milhões. As cinco categorias com maior volume de pedidos foram: 'Moda e Acessórios' (17%), 'Saúde, beleza e medicamentos' (15%), 'Eletrodomésticos' (12%), 'Telefonia e Celulares' e 'Livros e Revistas' (8%) e 'Informática' e 'Casa e Decoração' (7%) [10].

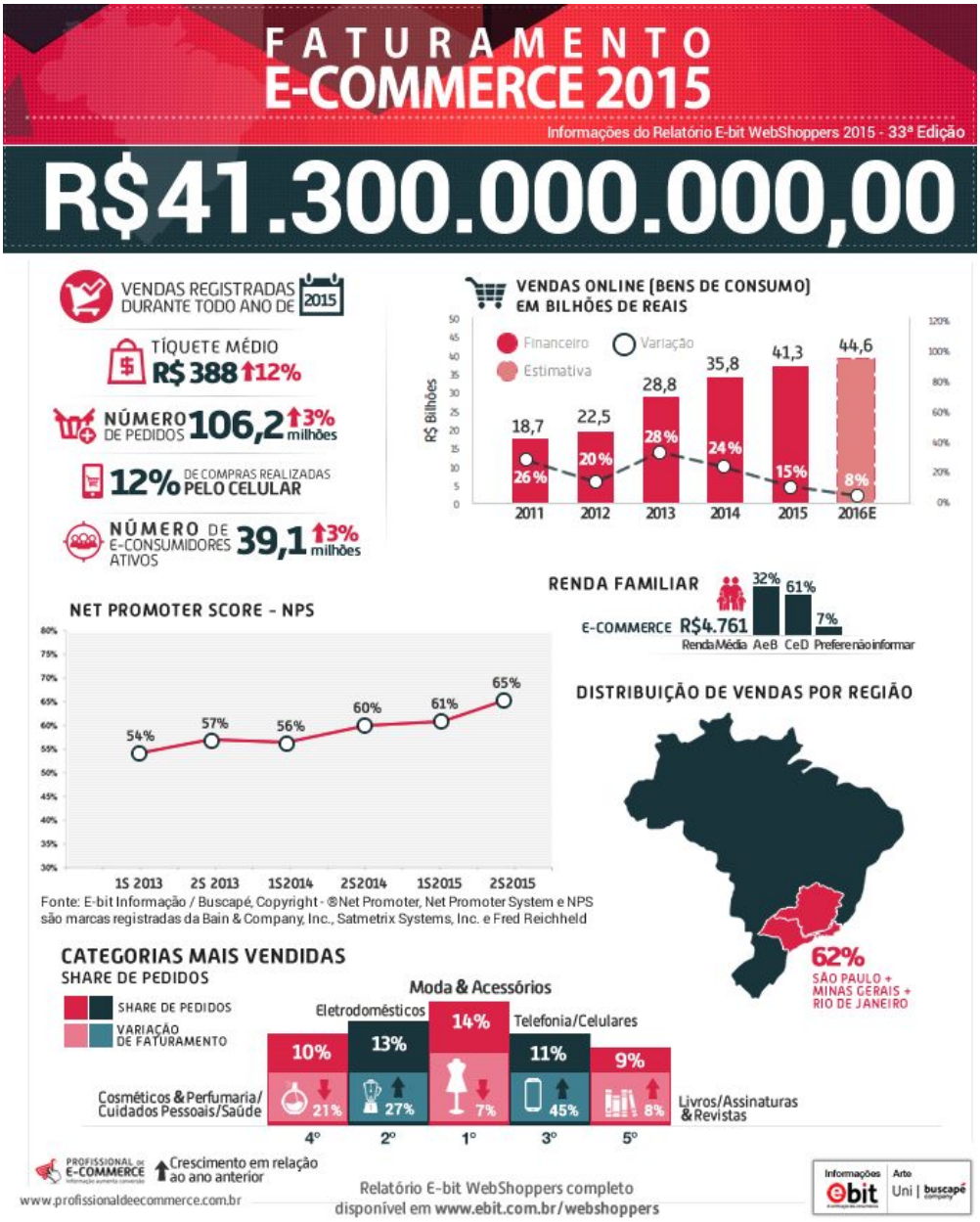

Figura 2. Faturamento e-commerce 2015 [11]

2015: O e-commerce brasileiro fechou o ano com R$ 41,3 bilhões de faturamento, apresentando 15% de crescimento em relação a 2014. Com um crescimento de 3% no

número de pedidos e um registro de consumidores virtuais ativos de 39,1 milhões. As cinco categorias com maior volume de pedidos foram: 'Moda e Acessórios' (14%), 'Eletrodomésticos' (13%), 'Telefonia e Celulares' (11%), 'Saúde, beleza e medicamentos' (10%) e 'Livros e Revistas' (9%) [11].

De 2012 até o final de 2015 o e-commerce B2C brasileiro registrou um aumento de 18,8 bilhões no faturamento, e apesar de ter registrado uma taxa de crescimento menor de 2014 pra 2015 do que a de 2013 pra 2014 o crescimento ainda chama bastante atenção em meio a crise econômica que o país passa. Com o crescimento do mercado a concorrência aumenta e os consumidores passam a ficar mais exigentes sobre a experiência de compra, sendo necessário se destacar dos concorrentes para sobreviver.

2.1.3 Processo de Compra

O processo de compra no e-commerce começa quando o consumidor realiza a busca pelos produtos que deseja comprar, geralmente iniciando por uma pesquisa pela melhor marca para determinado tipo de produto que deseja adquirir, em seguida passa pela busca da melhor oferta [12].

Após escolher qual o produto específico e onde comprar o processo segue pela adição do produto ao carrinho, preenchimento dos dados para o pedido, confirmação da compra e forma de pagamento.

Após a etapa de pagamento ser concluída o processo fica aguardando a confirmação de pagamento para continuar o fluxo. Uma vez confirmado o recebimento dos devidos valores o processo continua com o faturamento do pedido, envio das mercadorias, entrega das mercadores e por fim o pós-venda.

2.2 Pós-venda

O pós-venda é a etapa do processo que ocorre quando a transação já foi efetuada e o consumidor recebeu seu produto [13]. O pós-venda está sendo cada vez mais praticado para descobrir os pontos fracos do processo, refletir e agir sobre a melhoria do processo de venda. Apesar da importância dessa etapa do processo de venda, muitos empreendedores ainda não dão atenção suficiente e não exploram o potencial de fidelização de consumidores contido no pós-venda [13].

A fidelidade dos consumidores em relação as empresas está cada vez menor, pois há uma maior oferta no mercado, aumento a concorrência e tornando os clientes mais exigentes. Em 2013 a *LoudHouse* realizou uma pesquisa, em que 1000 brasileiros foram entrevistados, e 92% deles apontaram que a velocidade de resposta e da resolução dos problemas são características importantes na prestação de serviços. Além que 44% dos consumidores brasileiros afirmaram que gastariam mais dinheiro no futuro como forma de recompensar serviços excepcionais [14].

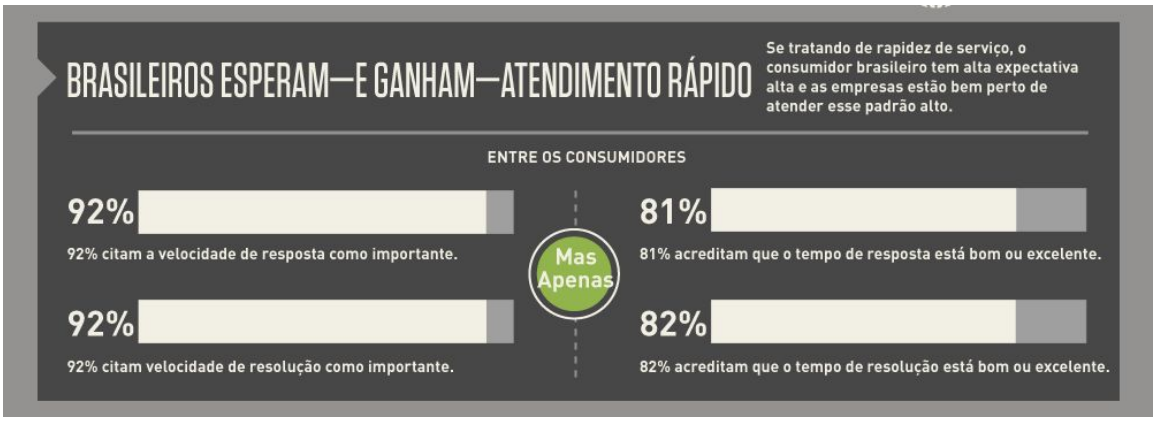

Figura 3. Pesquisa LoundHouse - Rapidez de serviço [14]

Figura 4. Pesquisa LoundHouse - Recomendação e negócios repetidos [14]

Tal pesquisa mostra que é necessário estabelecer alguma forma de acompanhamento do processo de venda e que se a empresa lidar com as dificuldades e problemas encontrados pelos consumidores de forma pró-ativa ela estará diferenciando-se dos concorrentes e valorizando sua empresa, é nesse ponto que o pós-venda mostra sua importância.

Alguns motivos que encorajam a fidelização de consumidores [1]:

1. Reter clientes pode custar até 5 vezes menos do que prospectar novos clientes.
2. Perde-se, em média 10% de consumidores ao ano. Reduzir esse número pode levar a um aumento de 25% ao ano;.
3. A taxa de lucro aumenta por tempo de permanência do cliente fidelizado.
4. O pós-venda eficaz é um bom marketing para a empresa.
5. Um consumidor satisfeito compartilha a boa experiência que obteve com a empresa com as outras pessoas, mais uma forma de marketing para a empresa.

2.2.1 Tipos de Serviços

Existem alguns tipos de serviços que podem ser utilizados para realizar a etapa de pós-venda, entre eles: ‘cartões’, ‘embalagens de qualidade e únicas’, ‘cupons’, ‘ligação’, ‘tutoriais’, ‘pesquisas periódicas’ e ‘anúncios’ [15].

Cartões: Uma correspondência física sempre foi uma forte ferramenta de marketing, servindo como lembrança da sua empresa para o consumidor e para quem for visitá-lo.

Embalagens de qualidade e únicas: Os consumidores criaram o hábito de reutilizar as embalagens que recebem ou utilizarem no seu dia-a-dia, uma sacola diferenciada de qualidade e distinta é uma ótima forma de beneficiar sua marca.

Cupons: Dependendo do produto ou serviço oferecido, oferecer um cupom para consumidores recentes passa a imagem de valorização e aumenta a chances de uma segunda compra ser realizada em sequência por esse consumidor. É uma das formas mais comuns de fidelização dos consumidores.

Ligação: Ligar para os consumidores alguns dias após a venda ser realizada, com perguntas simples sobre o funcionamento do produto ou serviço e se precisa de mais alguma coisa. Esse lado humano do processo passa uma mensagem de que sua empresa se preocupa com o consumidor e de que está disponível para qualquer questão.

Tutoriais: Alguns produtos podem ser diferenciados e difíceis de usar para alguns consumidores. Um tutorial de como utilizar o determinado produto ajuda consumidores com

esses problemas e passa a imagem única de engajamento da empresa com os consumidores.

Pesquisas periódicas: Alguns consumidores apreciam terem o seu feedback solicitado sobre sua compra. Técnica bastante utilizada em conjunto com o envio de cupons, aumentando assim os feedbacks recebidos dos consumidores.

Anúncios: Um anúncio é uma simples e ótima forma de mostrar aos consumidores que a empresa os valoriza. Geralmente utilizado com promoções.

2.3 Considerações

Neste capítulo foi apresentado o conceito de e-commerce, com uma explicação detalhada dos termos que o compõe. Além disso, mostrou-se os diversos tipos desta forma de transação comercial com suas respectivas definições. Foi apresentado um pouco sobre o comércio eletrônico no Brasil, e o seu faturamento anual. Explicando como ocorre o processo de compra, detalhando as etapas realizadas até o consumidor receber o produto ou serviço. Após essas etapas inicia-se o pós-venda, onde é mostrado como essa etapa é importante para a empresa manter os seus clientes e ter um diferencial no mercado. Outro ponto abordado neste capítulo são os tipos de serviços que podem ser utilizados na etapa do pós-venda, e como cada um ajuda.

Frente ao exposto, é possível perceber que o pós-venda é uma etapa que ajuda o lojista a descobrir falhas no seu processo de venda, e no aumento da lucratividade da sua empresa. Gerando o aumento da retenção dos clientes e da divulgação da loja pelo cliente para outros consumidores. O próximo capítulo será sobre a ferramenta construída neste trabalho.

3. A Ferramenta

Este capítulo visa apresentar a ferramenta, denominada *BackHere,* implementada neste trabalho, descrevendo: a ferramenta, suas características, suas funcionalidades, a implementação e o funcionamento.

3.1 Descrição

Foi implementada uma ferramenta com a intenção de ajudar responder ao problema de pesquisa, denominada *BackHere*. A *BackHere* foi desenvolvida para ser um software comercial, e é pretendido realizar uma cobrança pelo acesso a ferramenta e pelo número de feedbacks recebidos. Atualmente a ferramenta encontra-se em um repositório privado no Bitbucket e o controle de versão é realizado utilizando Git.

A *BackHere* é um software web que tem como funções principais o envio de notificações e coleta de feedback dos consumidores sobre as vendas de forma automática desde a importação dos dados da base do lojista. A *BackHere* foi criada com o objetivo de ajudar os lojistas a realizar a etapa do pós-venda com um baixo esforço, liberando os lojistas para consumirem seu tempo resolvendo os pontos fracos do processo e dificuldades encontradas pelos consumidores.

Será possível para o lojista utilizar a ferramenta como um meio de comunicação com os consumidores sobre a experiência de compra de forma automatizada. Com o uso da ferramenta, o lojista poderá importar as informações dos clientes e dos pedidos da sua plataforma, criar pesquisas com perguntas sobre a experiência de compra, com base no status do pedido enviar notificações para os consumidores e verificar as respostas recebidas.

3.2 Características

A seguir, são descritas as principais características da ferramenta *BackHere*:

Possibilita a criação de uma plataforma contendo os dados necessários para a comunicação via API com a base do lojista;

Realiza a importação dos dados sobre os consumidores e vendas diretamente da base do lojista, assim como possibilita a visualização dos dados importados;

Realiza a verificação dos dados de e-mail e telefone dos consumidores e possibilita a visualização se esses dados são válidos ou não;

Possibilita ao lojista a criação e visualização prévia de pesquisas a serem enviadas aos consumidores através de notificações;

Possibilita ao lojista determinar um tipo de status para cada status recebido dos pedidos, permitindo assim que seja feito um mapeamento com as pesquisas.

As notificações ocorrem através de um mapeamento entre o status em que o pedido se encontra, a pesquisa a ser enviada e quais serviços devem ser utilizados. Como serviços de notificação estão disponíveis e-mail e SMS;

Possibilita aos consumidores, que receberem as notificações, um meio de responder a pesquisa enviada. Assim como possibilita ao lojista verificar as respostas recebidas.

Todas as tarefas podem ser agendadas para que rodem de forma automática de acordo com os dias, horários e intervalo de tempo selecionado;

Todas as tarefas criadas ou agendadas serão realizadas em uma estrutura de processamento em segundo plano, para que o processamento não cause nenhum impacto na navegação pela ferramenta;

A aplicação deve ser dividida em, pelo menos, dois servidores para que o processamento das tarefas não cause impacto na aplicação e para que possíveis problemas na aplicação não causem impacto no processamento das tarefas. Assim deve ser dividida em servidores de *Front* onde a aplicação estará rodando e disponível para acesso dos usuários e em servidores de *Worker* onde a estrutura de processamento em segundo plano estará rodando e executando todas as tarefas criadas ou agendadas;

A ferramenta deve ficar o menor tempo possível indisponível, mesmo quando realizado o processo de deploy;

3.3 Implementação

Este capítulo demonstrará com foi o processo de desenvolvimento da ferramenta, mostrando: tecnologias utilizadas, padrões de projetos, arquiteturas de software, visão geral da ferramenta e visão de implantação.

3.3.1 Tecnologias Utilizadas

A *BackHere* foi implementada com a utilização de algumas tecnologias de livre acesso disponibilizadas na web, serviços com nível de gratuidade ou versão para testes. A definição de quais tecnologias seriam utilizadas na implementação da ferramenta foi baseada na experiência do autor desta monografia. Alguns profissionais da área de

desenvolvimento de software foram consultados e colaboraram no processo de escolha das tecnologias.

A seguir, algumas das tecnologias utilizadas na implementação e a justificativa da utilização:

Ruby

Ruby é uma linguagem dinâmica, open source com foco na simplicidade e na produtividade. Tem uma sintaxe elegante de leitura natural e fácil escrita. Desde que foi tornado público em 1995, o Ruby arrastou consigo programadores devotos de todo o mundo. Em 2006, o Ruby atingiu aceitação massiva, com a formação de grupos de usuários em todas as principais cidades do mundo e com as conferências sobre Ruby com lotação esgotada [16].

A *BackHere* foi desenvolvida na linguagem Ruby pela simplicidade da linguagem, ótima documentação, comunidade ativa e por experiência do autor desta monografia.

Ruby on Rails

Além de Ruby como linguagem principal, a *BackHere* foi implementada com o auxílio do framework Ruby on Rails. Frequentemente referenciado como Rails ou RoR, o Ruby on Rails é um projeto de código aberto escrito na linguagem de programação Ruby, ele é um framework usando uma estrutura MVC (Model-View-Controller), bastante conveniente para a construção de aplicativos Web [17].

O Ruby on Rails é um “meta-framework”, um framework de frameworks, composto por diversos frameworks e bibliotecas, chamadas de Gems, onde é possível adicionar diversos outros frameworks e bibliotecas de maneiras simples. O Ruby on Rails foi escolhido por ser o framework mais difundido mundialmente com base em Ruby, pela simplicidade e por experiência do autor desta monografia.

Nesse contexto de adição de frameworks e bibliotecas ao Ruby on Rails, foram adicionadas diversas gems e algumas delas, que têm um maior destaque, são: RailsAdmin, Devise, CanCan, MongoID, Sidekiq e AdminLTE2.

RailsAdmin

O RailsAdmin trata-se de um Rails engine que provê uma interface de fácil utilização para gerenciar os dados da aplicação. O RailsAdmin será utilizado para a criação de contas, verificação dos dados e exportação de dados. Existem diversas opções que realizam a função do RailsAdmin, que foi escolhido devido a fácil integração ao Rails, suporte ao MongoDB, a serviços de autenticação e autorização.

Devise

A Devise é uma solução flexível de autenticação para Ruby on Rails com base no padrão de arquitetura MVC. Utilizada para a realizar a criação de autenticação de usuários na ferramenta. Foi escolhida por ser amplamente utilizada e testada pela comunidade, fácil utilização e ampla documentação.

CanCan

A CanCan é uma biblioteca de autorização para Ruby on Rails que restringe quais recursos o usuário tem permissão de acesso. Foi escolhida por ser amplamente utilizada e testada pela comunidade e de fácil utilização.

MongoID

O MongoID é um ODM (Object-Document-Mapper) framework para MongoDB em Ruby. Foi utilizado parar gerir todo a relação do Ruby on Rails com o MongoDB, escolhida por ser amplamente utilizada e testada pela comunidade, fácil utilização e ampla documentação.

Sidekiq

O Sidekiq é uma estrutura de processamento em segundo plano completo para Ruby. Destinado a ser simples de integrar com qualquer aplicação Rails moderna e apresenta um desempenho muito superior do que outras soluções existentes. O Sidekiq trabalha utilizando armazenando as tarefas em filas, utilizando o Redis para manipulação das filas e outros dados, e *workers* que executam as tarefas enfileiradas.

O Sidekiq foi escolhido para execução das tarefas geradas pela aplicação em background, por ser amplamente utilizado e testado pela comunidade, fácil utilização, ampla documentação e experiência do autor desta monografia.

AdminLTE2

O AdminLTE2 é um template open source para aplicações web, trata-se de um template HTML responsivo baseado em no framework Bootstrap. Possui um design modular que permite fácil customização.

O AdminLTE2 foi escolhido para ser o template da BackHere, por possuir uma boa documentação e fácil utilização.

MongoDB

O principal sistema de gerenciamento de bancos de dados utilizado neste projeto é o MongoDB, utilizado para armazenar todos os dados das principais classes deste projeto. O MongoDB é um banco NoSQL (termo utilizado para descrever banco de dados não relacionais de alto desempenho) orientado a documentos de alta performance, open source e schema-free, escrito em C++.

É uma mistura entre os repositórios escaláveis baseados em chave/valor e a tradicional riqueza de funcionalidades dos bancos relacionais.

Foi utilizado os serviços do MLab para armazenamento e gestão do MongoDB. O Mlab é um DataBase as a Service (DBaaS) e foi escolhido por oferecer um nível gratuito com 500MB de armazenameto.

O MongoDB foi escolhido por ter uma ótima interação com o Ruby on Rails através da gem MongoID e por experiência do autor desta monografia.

Redis

Outro sistema de gerenciamento de bancos de dados utilizado neste projeto é o Redis. O Redis é um banco de dados NoSQL, escrito em C, baseado em chave-valor cujo o armazenamento é feito na memória RAM, fazendo com que a escrita e leitura sejam rápidos.

O redis será utilizado para guardar cache da aplicação, sessões e pelo serviço de execução de tarefas em background. Foi escolhido por apresentar melhor performance entre as demais opções e pela dependência com o serviço de execução de tarefas.

Cron Format

O Cron Format foi utilizado para determinar quando as tarefas agendadas devem ser executadas. O *Cron Format* é simples, poderoso e flexível para definir o tempo e frequência de várias ações, utilizando 6 campos para determinar: minutos, horas, dias do mês, meses do ano, dias da semana e anos [18].

```
* * * * * *
| | | | | |
| | | | | +-- Year              (range: 1900-3000)
| | | | +---- Day of the Week   (range: 1-7, 1 standing for Monday)
| | | +------ Month of the Year (range: 1-12)
| | +-------- Day of the Month  (range: 1-31)
| +---------- Hour              (range: 0-23)
+------------ Minute            (range: 0-59)
```

Figura 5. Cron Format [18]

O Cron Format foi escolhido por ser um formato amplamente utilizado por outros serviços e bibliotecas que trabalhem com agendamento de tarefas.

Magento

Para a importação de dados foi escolhida inicialmente a plataforma de E-commerce *Magento*. O Magento foi escolhido por se tratar da plataforma de E-commerce mais utilizada mundialmente desde 2012, possuir comunicação via API (REST e SOAP), possuir uma ampla documentação e comunidade ativa. Foi escolhido a comunicação via api SOAP por possuir melhores recursos e por experiência do autor desta monografia.

Elastic Email

Para o envio de e-mail foi selecionado o serviço da *Elastic Email,* uma plataforma poderosa e eficiente para o envio de e-mails.

A Elastic Email foi escolhido por possuir um nível gratuito de 25000 e-mails e por disponibilizar a comunicação via SMTP (Simple Mail Transfer Protocol), protocolo de e-mail utilizado pelo Ruby on Rails de forma bastante simples.

Zenvia

Para o envio de SMS foi selecionado o serviço da *Zenvia,* uma plataforma de mensagens móveis líder no mercado brasileiro.

A Zenvia foi escolhida por possuir uma boa documentação dos seus recursos, oferecer uma quantidade de 50 SMS para testes, possuir grandes clientes (como a Fiat) e possuir comunicação via API REST.

AWS

Para hospedagem dos servidores foi selecionada a Amazon Web Services (AWS). A AWS é uma plataforma de serviços em nuvem segura, que oferece poder computacional, armazenamento de banco de dados, distribuição de conteúdo e outras funcionalidades para ajudar as empresas em seu dimensinamento e crescimento.

A AWS foi escolhida por disponibilizar um nível de gratuidade com duração de 12 meses, ser possível realizar cadastro na AWS Educate Application e resgatar $35 e por interesses do autor desta monografia.

Outros

Também foi feito o uso de API para a verificação de e-mail, verificação de telefone e encurtamento de URL, contudo não são tão expressivas quanto as demais características citadas anteriormente.

3.3.2 Padrões e Arquiteturas

Para o desenvolvimento da ferramenta alguns padrões de projeto foram utilizados, esses padrões contribuíram para facilitar o desenvolvimento, legibilidade e manutenibilidade do código.

MVC

O MVC é um padrão de projeto que separa a aplicação em 3 camadas, onde o Model é responsável pela leitura e escrita de dados, regras de negócio, lógica e funções, a View é responsável pela interação com o usuário realizando a exibição dos dados e o Controller é responsável por receber todas as requisições do usuário, convertendo-a em comandos e controlando qual Model usar e qual View será mostrada ao usuário. Esse padrão foi utilizado devido por ser definido no framework Ruby on Rails.

Multi-Tenancy

Multi-Tenancy é uma arquitetura de software especial para um ambiente em nuvem que permite que uma única aplicação servir múltiplos Tenant (inquilinos), compartilhando todo os recursos, mas fazendo com que permaneçam logicamente isolados. Detalhes dessa arquitetura na ferramenta são vistos no tópico de visão geral.

Essa arquitetura foi utilizada pois é pretendido que a ferramenta seja oferecida e utilizada por múltiplos lojistas ao mesmo tempo, tornando fundamental que o ambiente de cada conta seja logicamente isolado das demais contas. Assim com uma aplicação única será possível atender diversos lojistas.

Adapter

O problema resolvido pelo padrão Adapter é quando classes com interfaces diferentes, precisam trabalhar em conjunto. A solução é isolar as conversões de uma interface para a outra, do resto da lógica de negócio [19].

Na ferramenta o padrão foi utilizado para implementar um Adapter para o Magento, pois foi desenvolvida a integração com a versão 1.X e existe uma versão 2.X que necessitará de integração diferente. Assim basta identificar a versão e utilizar o Adapter respectivo.

```
class MagentoAdapter

  attr_accessor :magento, :api_url, :api_user, :api_key, :session_key, :sync_date, :page, :offset, :date_time_now

  def initialize(magento)
    @magento     = magento
    @api_url     = magento.api_url
    @api_user    = magento.api_user
    @api_key     = magento.api_key
    @session_key = nil
  end
```

Figura 6. BackHere - Exemplo de Adapter

```ruby
def magento_adapter
  @magento_adapter ||= (self.version?(1) ? MagentoAdapter.new(self) : MagentoAdapter2.new(self))
end
```

Figura 7. BackHere - Exemplo Uso de Adapter

Esse padrão foi utilizado para tratar as requisições via API SOAP que são feitas pela ferramenta ao Magento, assim como para tratar todas as requisições via API REST que são feitas pela ferramenta.

Factory

O problema que os padrões Factory resolvem é criar instâncias de objetos separadas do resto da lógica, por isso são classificados como padrões de criação [19].

Existe uma classe base para as tarefas e todas as demais são especificações dessa base, assim utilizando o Simple Factory a criação das tarefas foi simplificada e isolada em um só lugar.

```ruby
class TaskFactory

  TASKS = [
    CustomerTask, OrderTask, CustomerEmailVerificationTask, CustomerPhoneVerificationTask,
    CreateSurveyNotificationsTask, SendNotificationsTask
  ]

  def self.build(generic_type, params = {})
    TASKS.each do |task|
      if task.same_type?(generic_type)
        params.merge!({type: task.type})
        return task.new(params)
      end
    end
    false
  end

  def self.find_or_initialize_by(generic_type, params = {})
    TASKS.each do |task|
      if task.same_type?(generic_type)
        params.merge!({type: task.type})
        return task.find_or_initialize_by(params)
      end
    end
    false
  end


  def self.find_by_generic_type(generic_type)
    TASKS.each { |task| return task if task.same_type?(generic_type) }
  end

end
```

Figura 8. BackHere - Exemplo de Simple Factory

Com essa implementação da TaskFactory foi possível identificar, através dos parâmetros recebidos, qual tarefa deveria ser criada. E ao surgirem novas tarefas a inclusão delas na TaskFactory é simples.

```
task = TaskFactory.build(generic_type, task_params(platform))

if task && !task.save
  flash.keep[:error] = "(#{platform.name}) Não foi possível criar a tarefa."
  return redirect_to new_task_path
end
```

Figura 9. BackHere - Exemplo Uso de Simple Factory

Esse padrão foi utilizado para simplificar a criação das tarefas da ferramenta e manter o código legível e de fácil manutenibilidade.

3.3.3 Visão Geral

A ferramenta *BackHere* tem como base o padrão de projeto MVC e estrutura de pacotes do projeto foi gerado automaticamente pelo Ruby on Rails. As três camadas do padrão MVC estão dentro do pacote app, cada uma como um pacote. Nesse projeto foi decidido utilizar algumas classes como base para outras classes específicas, para essas classes foram criadas pacotes dentro do pacote model e todas as classes específicas ficam dentro do pacote de sua classe base.

É pretendido a criação de uma API para possibilitar a importação de dados para a ferramenta a partir de qualquer sistema, para isso foi criada uma estrutura de módulos dentro do pacote lib. Esses módulos guardam a lógica de criação das classes que poderão ser criadas pelo API e pela comunicação da ferramenta com os sistemas.

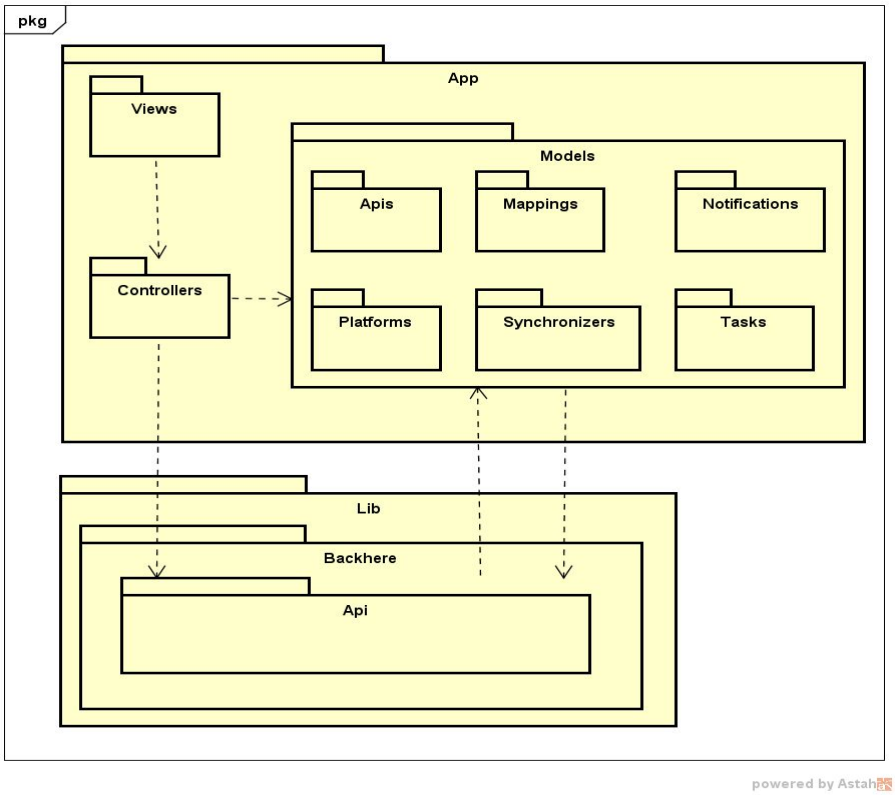

Figura 10. BackHere - Visão Geral - MVC

Models

Cada model fornece as características iniciais de cada entidade do software. Alguns models possem um controller que faz com que seus atributos sejam visualizados e manipulados. Outros models são utilizados para realizar ações da ferramenta e não possuem dados para serem persistidos.

Foi decidido utilizar o MongoDB como principal sistema de gerenciamento de bancos de dados pela sua simples integração com o Rails feita através da gem MongoID, onde o model que será persistido é responsável por declarar seus atributos que serão persistidos no MongoDB evitando a necessidade de migrations dos sistemas de gerenciamento de banco de dados SQSL. Também por ser necessário a criação de atributos dinamicamente, para cobrir todos os dados importados de um sistema para a ferramenta, e o MongoDB dá suporte a essa criação dinâmica.

A ferramenta foi criada para dar suporte a múltiplos usuários de forma logicamente isolados, para isso é utilizada a arquitetura Multi-Tenancy implementada com o uso da gem MondoID-Multitenancy. Foi utilizado o model de Account como Tenant de toda a ferramenta, assim todos os models que tem dados persistidos são dependentes da Account e em todas

as ações da ferramenta a persistência e recuperação de dados são feitas dependentes da Account. Dessa forma a arquitetura Multi-Tenancy funciona com a utilização compartilhada dos recursos da ferramenta.

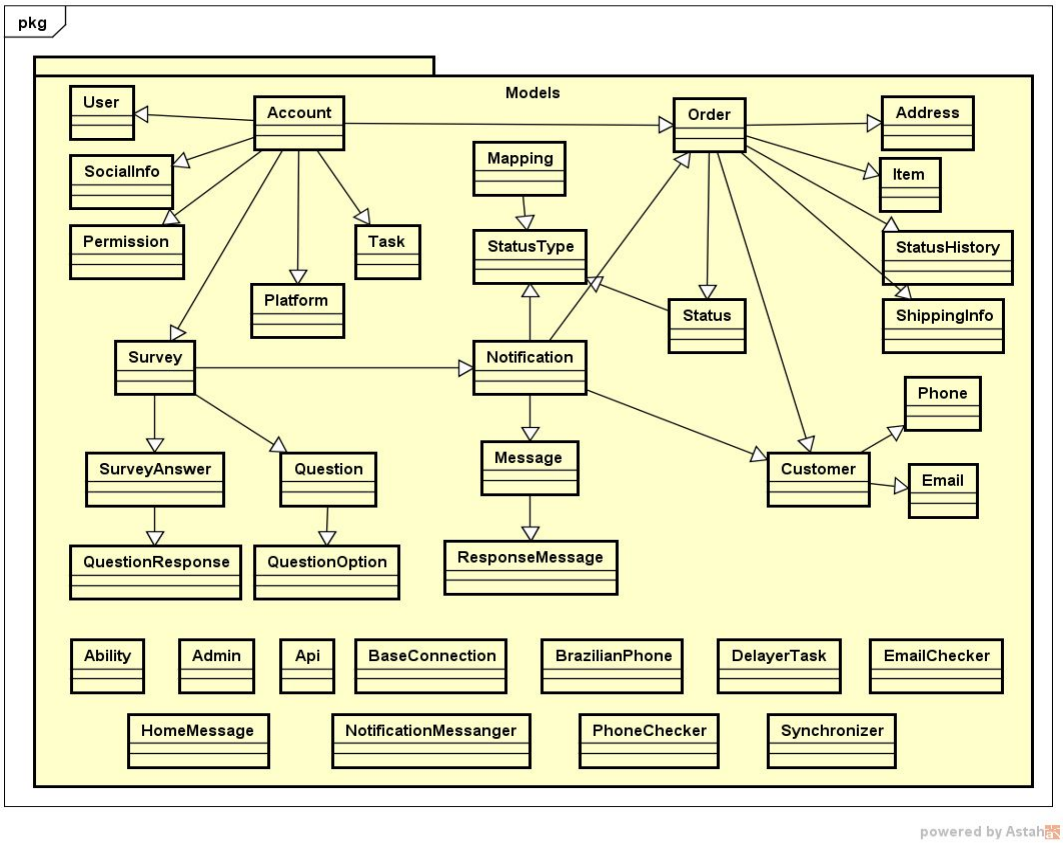

Figura 11. BackHere - Visão Geral - Models - 01

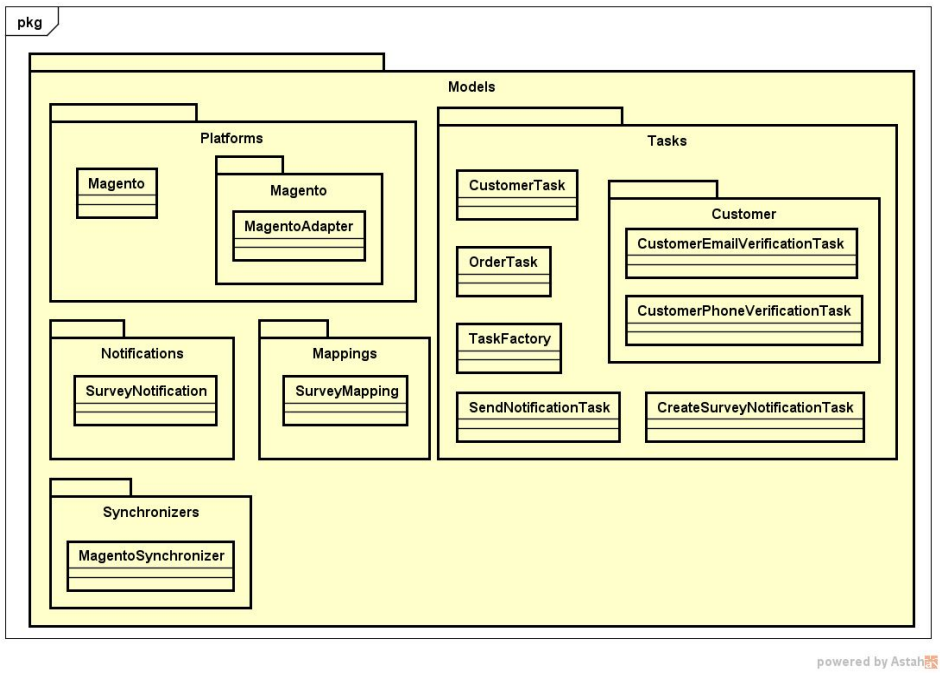

Figura 12. BackHere - Visão Geral - Models - 02

Controllers

Os controllers tem a responsabilidade de executar as ações provenientes da comunicação com o usuário, busca de informações para exibição com os usuários e responder as chamadas externas.

Foi decidido que uma *home page* da ferramenta seria criada dentro do projeto e não em uma aplicação separada, para desacoplar foi criado um controller base para todas ações dentro da ferramenta. Também foi necessário criar um controller base para a comunicação via API para receber notificações de callback da Zenvia.

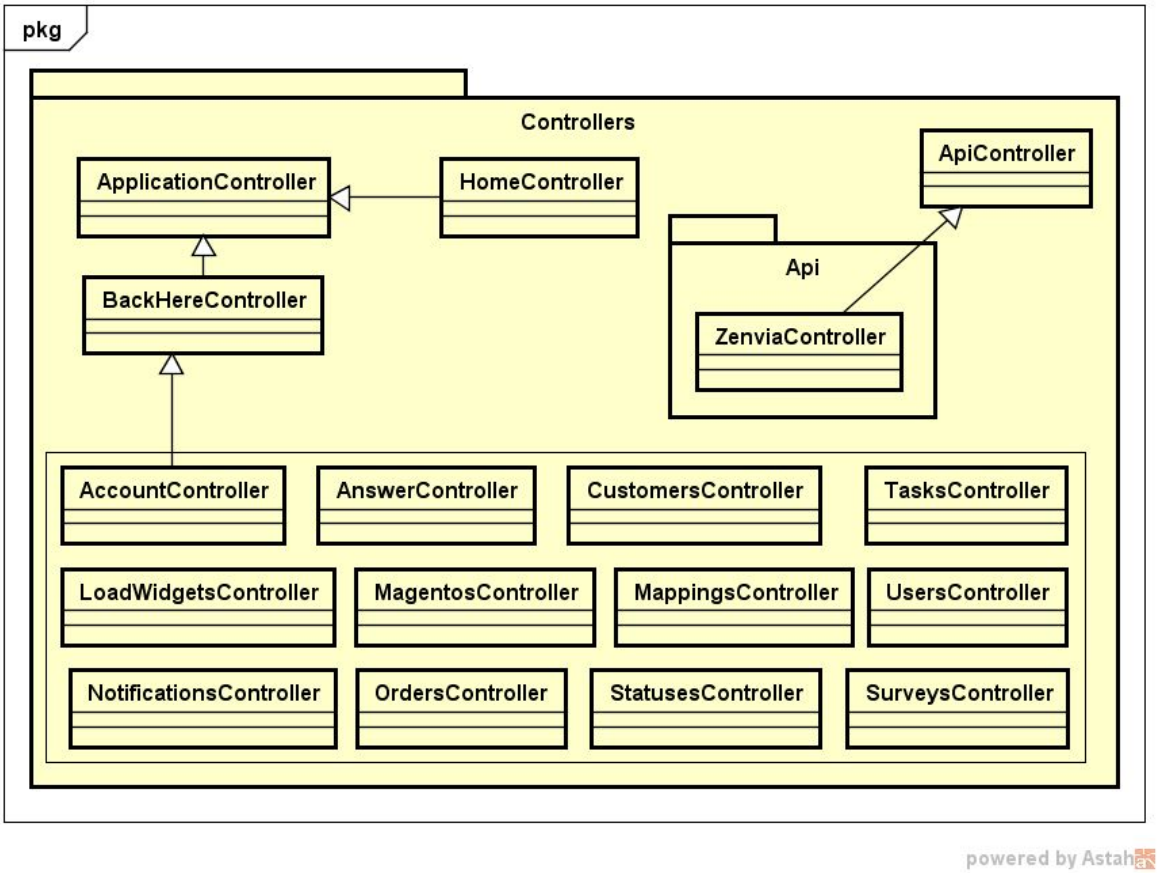

Figura 13. BackHere - Visão Geral - Controllers

Views

As views tem a responsabilidade por estabelecer a interação com o usuário. Tudo o que é mostrado ao usuário fica arquivado em pacotes dentro do pacote views, e dentro de cada pacote existem os arquivos referentes as ações do respectivo controller. Também foi utilizado do recurso de partials do Rails, onde é possível reutilizar o mesmo código em várias views e assim manter uma unicidade do que é exibido ao usuário.

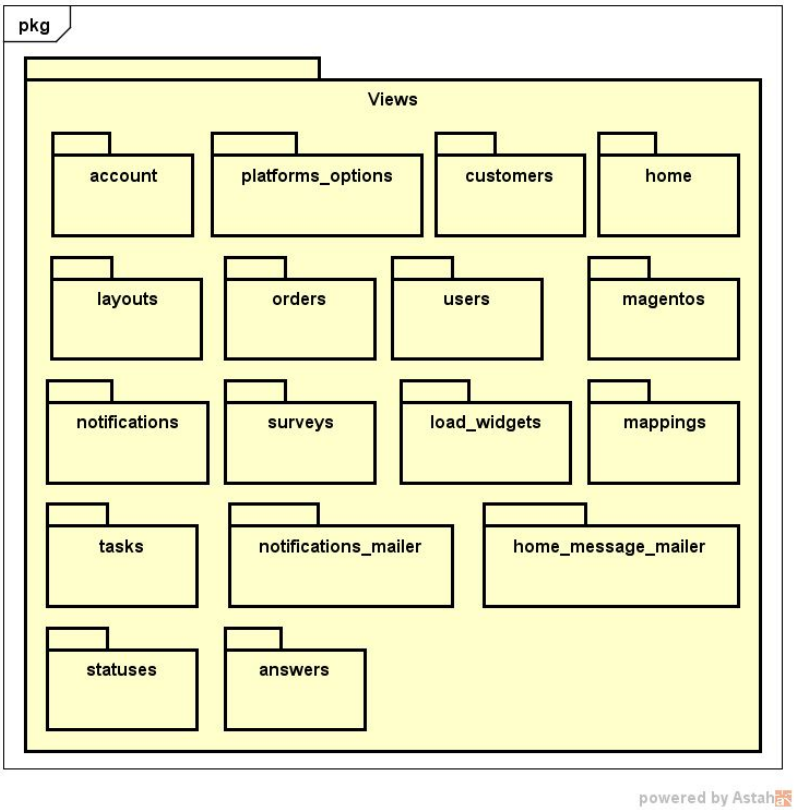

Figura 14. BackHere - Visão Geral - Views

Database

A utilização do MongoDB em conjunto com a gem MongoID acaba por eliminar o ponto principal do pacote de db, uma vez que não é necessário a criação das coleções no MongoDB através de *migrations*. Basta que a classe que será persistida tenha a configuração da gem MongoID, que ocorre através de inclusão do módulo Mongoid::Document.

Da mesma forma todas as classes que são persistidas pela ferramenta precisam da configuração da gem MongoID-Multitenancy que ocorre através da inclusão do módulo Mongoid::Multitenancy::Document.

```
class Order
 include Mongoid::Document
 include Mongoid::Timestamps
 include Mongoid::Multitenancy::Document

 tenant(:account)

 field :imported_from,              type: String
 field :remote_id,                  type: String
```

Figura 15. BackHere - Model - MongoID

O pacote db apenas é utilizado na ferramenta para guardar o arquivo seed que têm como objetivo persistir alguns objetos iniciais no MongoDB. O arquivo seed cria objetos da classe StatusType, que são fixos e visíveis para todos os usuários da ferramenta.

```
StatusType.find_or_create_by(code: :undefined,          label: "Não definido")
StatusType.find_or_create_by(code: :new_order,          label: "Novo Pedido")
StatusType.find_or_create_by(code: :awaiting_payment,   label: "Aguardando Pagamento")
StatusType.find_or_create_by(code: :payment_declined,   label: "Pagamento Recusado")
StatusType.find_or_create_by(code: :payment_approved,   label: "Pagamento Aprovado")
StatusType.find_or_create_by(code: :invoiced,           label: "Pedido Faturado")
StatusType.find_or_create_by(code: :shipped_order,      label: "Enviado ao cliente")
StatusType.find_or_create_by(code: :delivered_order,    label: "Entregue ao cliente")
StatusType.find_or_create_by(code: :strayed_order,      label: "Extraviado após o envio")
StatusType.find_or_create_by(code: :returned_order,     label: "Devolvido pelo cliente")
StatusType.find_or_create_by(code: :cancelled_order,    label: "Cancelado")
StatusType.find_or_create_by(code: :finalized_order,    label: "Finalizado")
```

Figura 16. BackHere - Seed

3.3.3 Visão de Implantação

A aplicação foi dividida em, dois servidores denominados de *Front* e *Worker*, onde o *Front* é o servidor que disponibiliza o acesso da ferramenta para os usuários e o *Worker* é o servidor que executa o Sidekiq, que é a estrutura de processamento em segundo plano. Essa divisão foi feita para que a execução das tarefas geradas pela ferramenta não impacte na disponibilidade da ferramenta, assim como a ferramenta não impacte na execução das tarefas. O maior impacto notado em executar em um mesmo servidor foi o consumo de memória que crescia acima do limite de memória do servidor.

A ferramenta e o Sidekiq são executados em uma instância t2.micro da AWS operando o Sistema Operacional Ubuntu 14.04 e utilizando um disco General Purpose SSD (GP2) de 8GiB. O setup das instâncias foi realizado inicialmente manualmente, mas configurado para que o processo de deploy fosse realizado utilizando o serviço do Capistrano, onde todas as tarefas necessárias, para atualização do código da ferramenta e reinicio dos serviços essenciais para operação, são executadas diretamente de um computador pessoal sem necessidade de qualquer acesso via terminal as instâncias e é realizado nos dois servidores de uma única vez.

A ferramenta necessita passar o menor tempo possível indisponível para os usuários, visando alcançar esse requisito da melhor maneira e de forma gratuita foi utilizado o Phusion Passenger em conjunto com o Nginx como servidor de aplicação web.

Ao executar o deploy com o serviço do Capistrano todas ações de atualização são executadas enquanto a ferramenta continua rodando e apenas ao final de tudo é que o servidor de aplicação web é reiniciado. É pretendido utilizar a versão Enterprise do Phusion Passenger que é capaz de reiniciar parte do serviço de aplicação enquanto outra parte continua executando e só depois reinicia essa parte, garantindo que a aplicação não fique nenhum segundo fora de operação durante o processo de deploy.

Para utilização do MongoDB foi selecionado o serviço da Mlab, onde é disponibilizado um espaço de 500mb para armazenamento gratuito. Para utilização do Redis foi instalado uma versão diretamente no servidor de *Worker*, essa ação foi tomada para evitar custos em excesso nessa etapa inicial e diminuir a latência do Sidekiq com o Redis.

Todas as informações para acesso aos serviços essenciais para o funcionamento da ferramenta são acessadas através de variáveis de ambiente para manter um nível de segurança caso o código seja exposto a terceiros. Essas variáveis de ambiente são gerenciadas através de um arquivo yml não versionado pelo Git e que é importado aos servidores no processo de deploy.

3.4 Funcionamento

Para que o usuário tenha acesso a ferramenta é necessário a criação prévia de uma conta, que só pode ser criada através do acesso ao painel de Admin da aplicação. Cada conta permite que sejam criados vários usuários e todos tem acesso ao mesmo ambiente na aplicação. Além disso é necessário a criação de permissões para ter acesso ao recurso de plataformas.

Após ter a conta criada e com a devida permissão, é necessário realizar alguns passos dentro da ferramenta para que a ferramenta possa funcionar corretamente.

3.4.1 Configurações

Para o funcionamento da ferramenta é necessário que o usuário realize algumas configurações básicas, são elas:

Criar uma plataforma informando os dados necessários para a comunicação via API com a base de dados do usuário (como URL, usuário, senha e data do início da operação);

Figura 17. BackHere - Criação de plataforma - Seleção

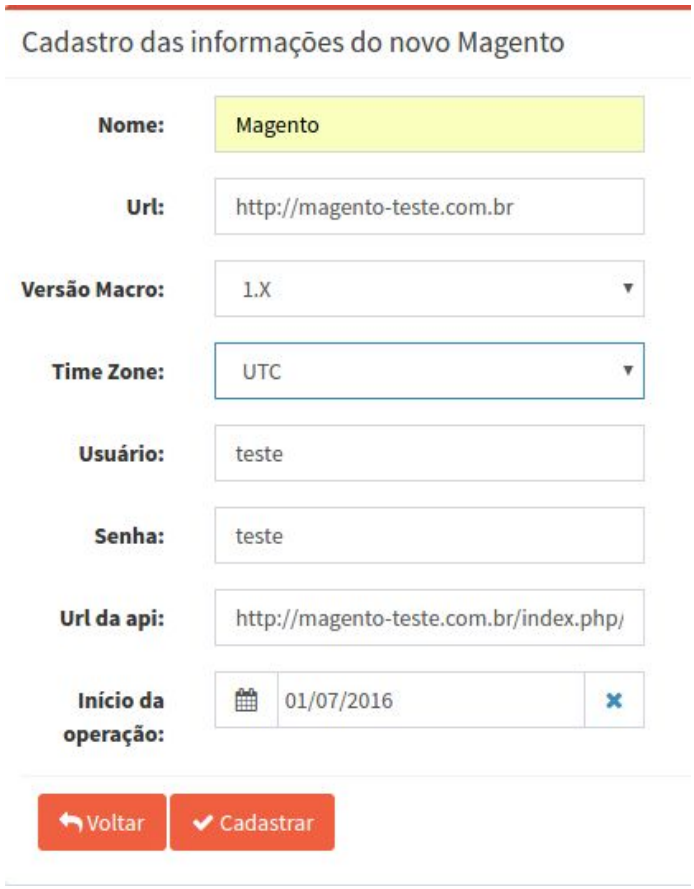

Figura 18. BackHere - Criação de plataforma - Preenchimento das informações

Selecionar o tipo de cada status criado na ferramenta e atualizar os status. Esses status são criados ao baixar os pedidos para a ferramenta e os tipos de status são fixos para todos os usuários;

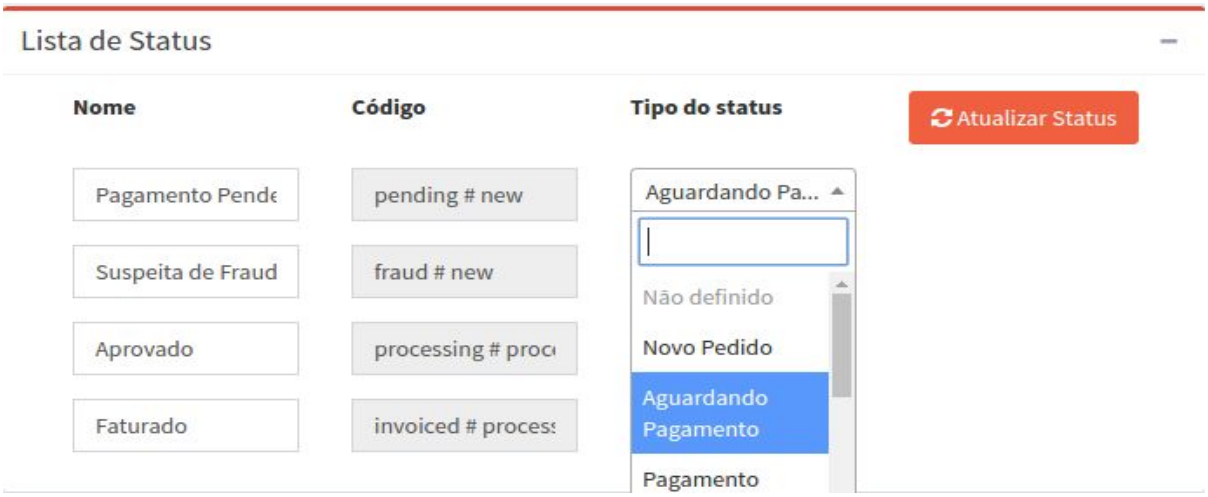

Figura 19. BackHere - Atualização de status

Criar pesquisas na ferramenta, informando um nome, descrição e as perguntas que fazem parte dessa pesquisa. As perguntas podem ser de texto ou opção, sendo que as de opção pode ser escolha única ou múltipla escolha (incluindo ou não uma opção 'outro');

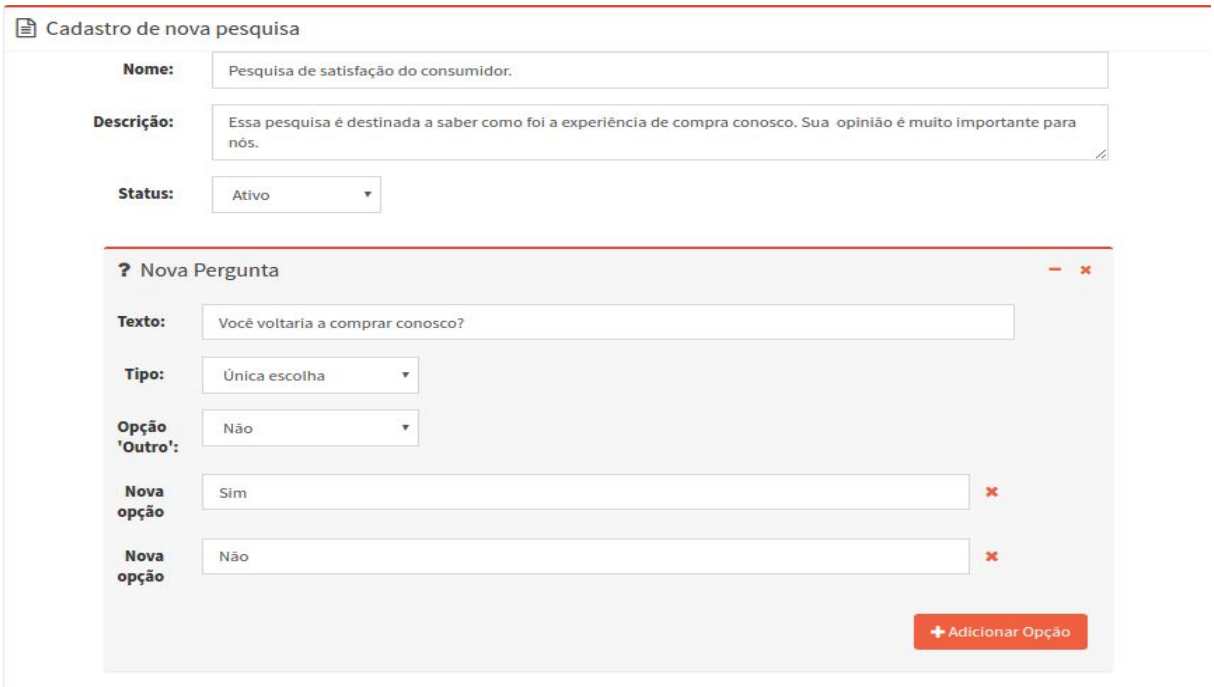

Figura 20. BackHere - Criação de pesquisa - 01

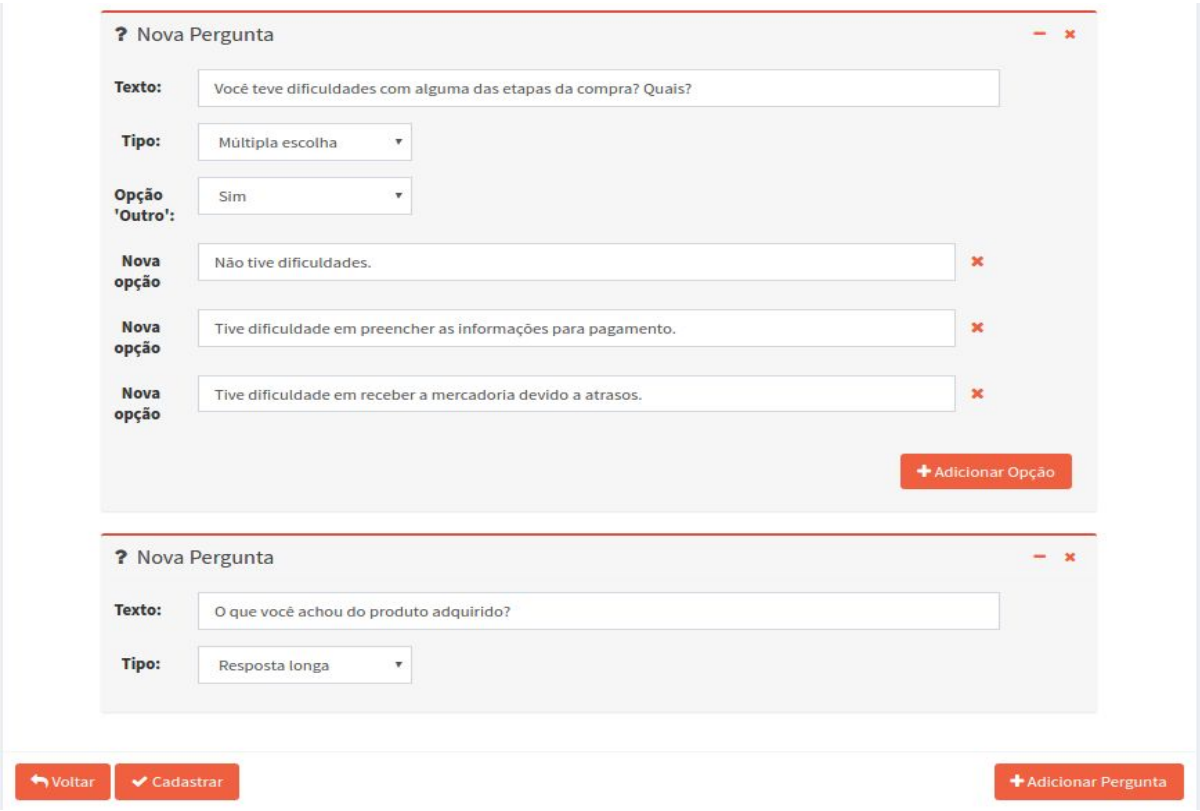

Figura 21. BackHere - Criação de pesquisa - 02

Criar a configuração do mapeamento entre um tipo de status da ferramenta, com uma pesquisa e os serviços que devem ser utilizados para notificação;

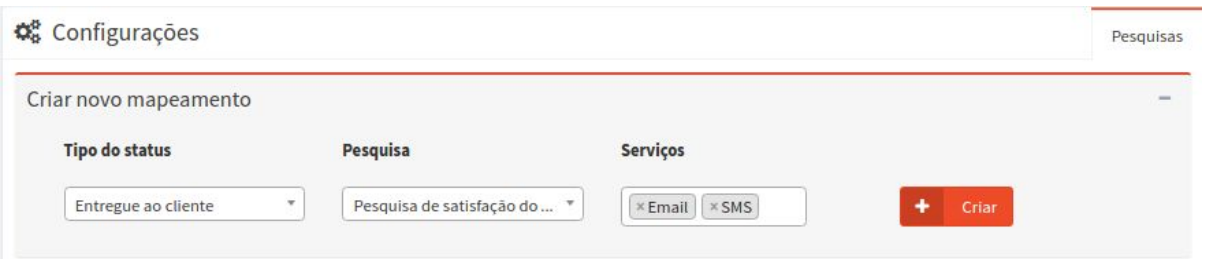

Figura 22. BackHere - Mapeamento de pesquisa

Criar o agendamento das tarefas a serem executadas, escolhendo os dias, período do dia e intervalo em que devem ser executadas, para que todo o processo seja automatizado;

Figura 23. BackHere - Agendamento de tarefas

3.4.2 Tarefas

As tarefas são a forma como a ferramenta realiza as ações necessárias para o seu funcionamento em background. Na ferramenta é possível criar tarefas para execução imediata ou criar um agendamento, permitindo uma execução automática sem necessidade de interação posterior com o usuário.

Figura 24. BackHere - Criação de tarefas

As tarefas são bem divididas para que possam executar de forma rápida possível e evite problemas de consumo de memória nos servidores, esse problema foi encontrado na comunidade e também por experiência do autor desta monografia. Algumas tarefas que podem demandar muito tempo em sua execução, são executas de forma paginada permitindo que ela seja finalizada e reiniciada até que cumpra toda a execução necessária.

Ao visualizar uma tarefa após sua execução é possível saber quantas ações foram realizadas com sucesso, quantas ações falharam em sua execução e quantas ações tiveram um erro inesperado.

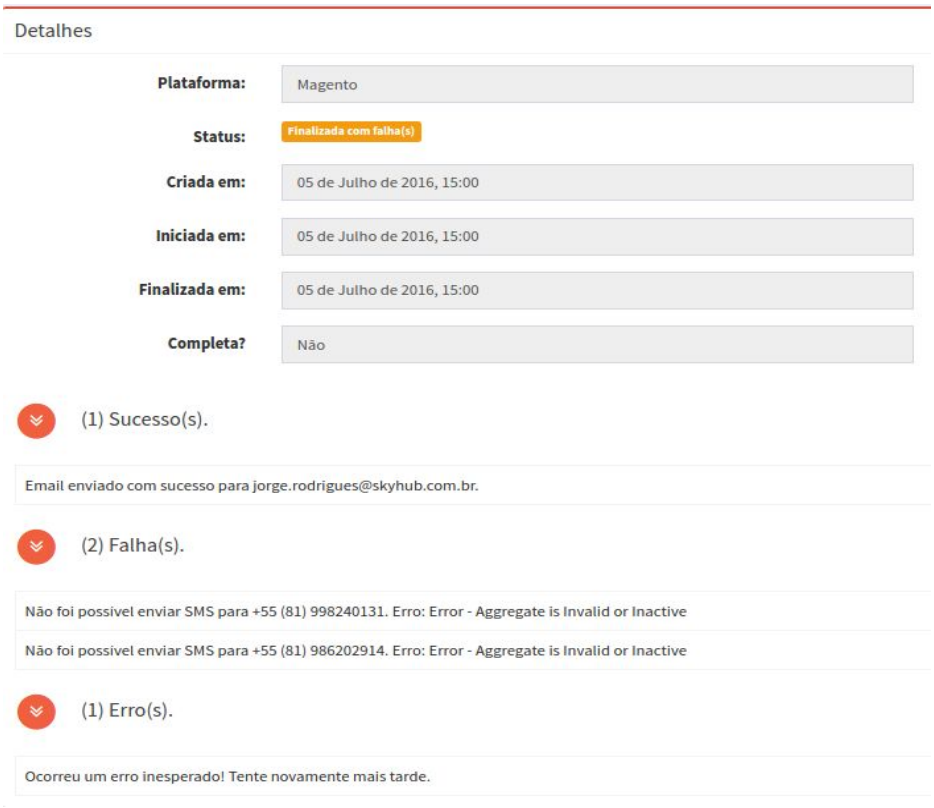

Figura 25. BackHere - Visualização de uma tarefa

A seguir, são descritas a execução das tarefas que a ferramenta realiza.

Importar Clientes

Tarefa responsável por importar as informações sobre os consumidores da base de dados do cliente. A importação primeiro realiza a busca de todas as informações sobre os consumidores via API, em seguida é feito um processamento das informações e mapeamento dos campos retornados na consulta com os campos padrões da classe Customer. Após o processamento das informações é realizado o armazenamento dessas informações na base de dados. Essa tarefa tem a execução realizada de forma paginada.

Importar Pedidos

Tarefa responsável por importar as informações sobre os pedidos da base de dados do cliente. A importação primeiro realiza a busca de todas as informações, da plataforma cadastrada, sobre os pedidos via API, sem seguida é feito um processamento das informações e mapeamento dos campos retornados na consulta com os campos padrões da classe Order e demais referências. Após o processamento das informações é realizado o armazenamento dessas informações na base de dados. Essa tarefa tem a execução realizada de forma paginada.

Verificar E-mails

Tarefa responsável por verificar todos os e-mails dos consumidores importados para a ferramenta. A verificação carrega todos os Customers que possuam algum e-mail ainda não verificado e realiza a verificação via API REST, com o serviço de verificação da QuickEmailVerification, em seguida trata o retorno e de acordo com a resposta salva a informação de que o e-mail já foi verificado e se é válido ou não.

Essa tarefa é importante para evitar envios desnecessários para e-mails inválidos e assim evitar o consumo de recursos.

Verificar Telefones

Tarefa responsável por verificar todos os números de telefones dos consumidores importados para a ferramenta. A verificação carrega todos os Customers que possuam algum telefone ainda não verificado e realiza a verificação via API REST, com o serviço ver verificação da NeutrinoAPI, em seguida trata o retorno e de acordo com a resposta salva a informação de que o telefone já foi verificado, se é válido ou não e se é mobile ou não.

Essa tarefa é importante para evitar envios desnecessários de SMS para números inválidos ou que não sejam mobile e assim evitar o consumo de recursos.

Criar Notificações

Tarefa responsável por criar as notificações que precisam ser enviadas aos consumidores. Para que as notificações sejam criadas é necessário que o usuário possua alguma pesquisa criada e realize o mapeamento de um tipo de status com uma pesquisa e

selecione os serviços utilizados para enviar as notificações (Essa ação é realizada nas configurações).

A tarefa carrega todos os mapeamentos existentes e faz uma busca dos pedidos que possuam um status do tipo selecionado no mapeamento. Em seguida é criada na base de dados uma Notification ligada ao pedido, cliente, serviços e pesquisa, que será enviada posteriormente.

Enviar Notificações

Tarefa responsável por enviar as notificações aos consumidores. A tarefa carrega todas as notificações que precisam ser enviadas e faz o envio de acordo com os serviços selecionados.

As notificações enviadas via e-mail possuem em seu corpo um botão que leva a responder a pesquisa na ferramenta, assim como um link para quem preferir copiar e colar diretamente no navegador.

Figura 26. BackHere - Notificação por e-mail

As notificações enviadas via SMS possuem sem seu corpo um link encurtado, utilizando a API do Google Shortener para caber na limitação de tamanho da mensagem, que leva a responder a pesquisa na ferramenta.

Figura 27. BackHere - Notificação por SMS

3.4.3 Respostas das Pesquisas

Ao receberem as notificações os consumidores podem acessar o link passado e serão direcionados para a *BackHere*, que identificará a pesquisa que deve ser respondida através do token único especificado no link. Na página aberta será exibida a pesquisa correspondente com todas as opções de respostas. Ao finalizar o preenchimento de todas as perguntas e realizar o envio, essa pesquisa torna-se indisponível para que seja respondida por esse mesmo consumidor.

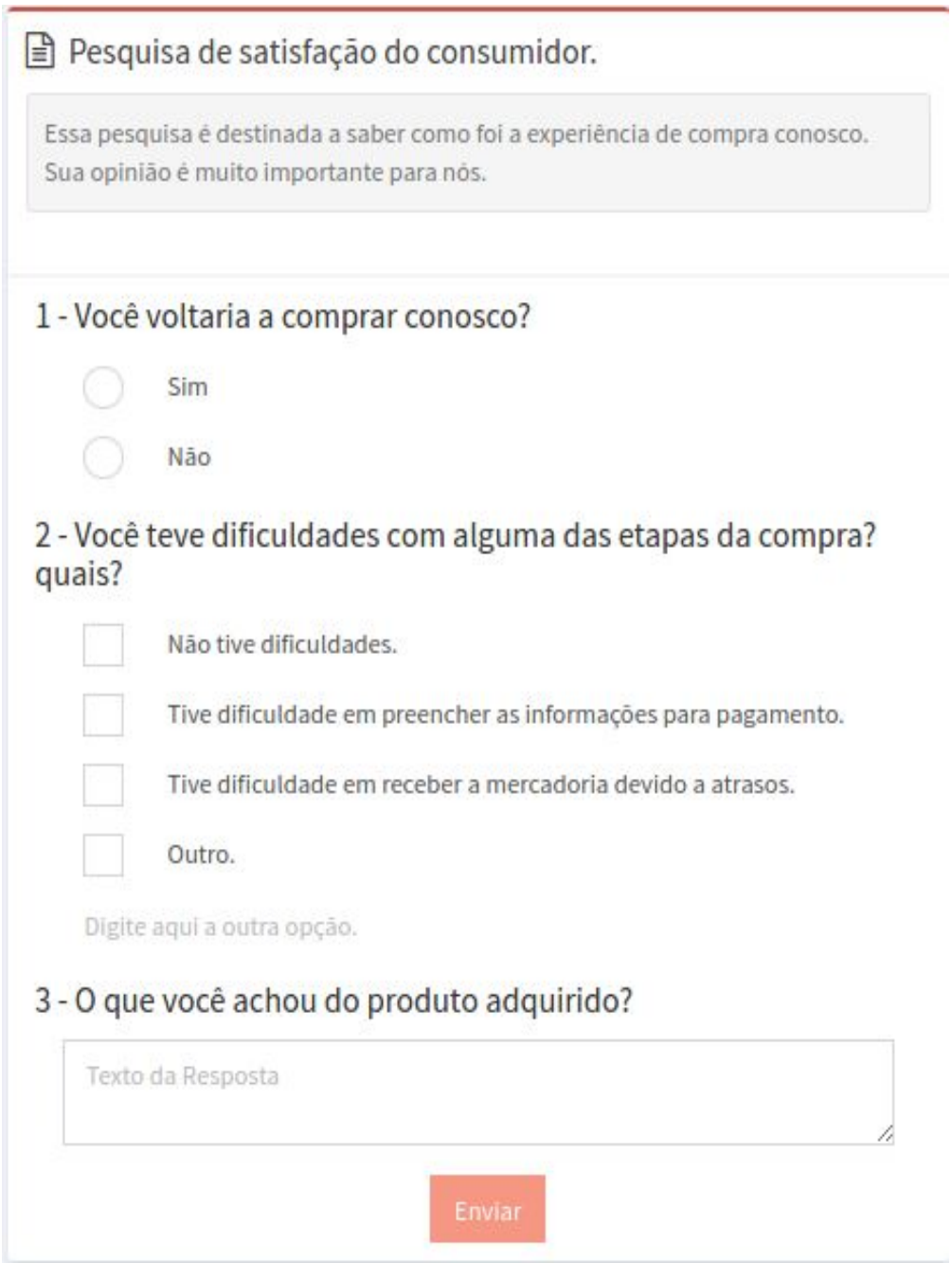

Figura 28. BackHere - Pesquisa

As Respostas recebidas ficam disponíveis para visualização na ferramenta, para que o lojista possa realizar uma análise das respostas.

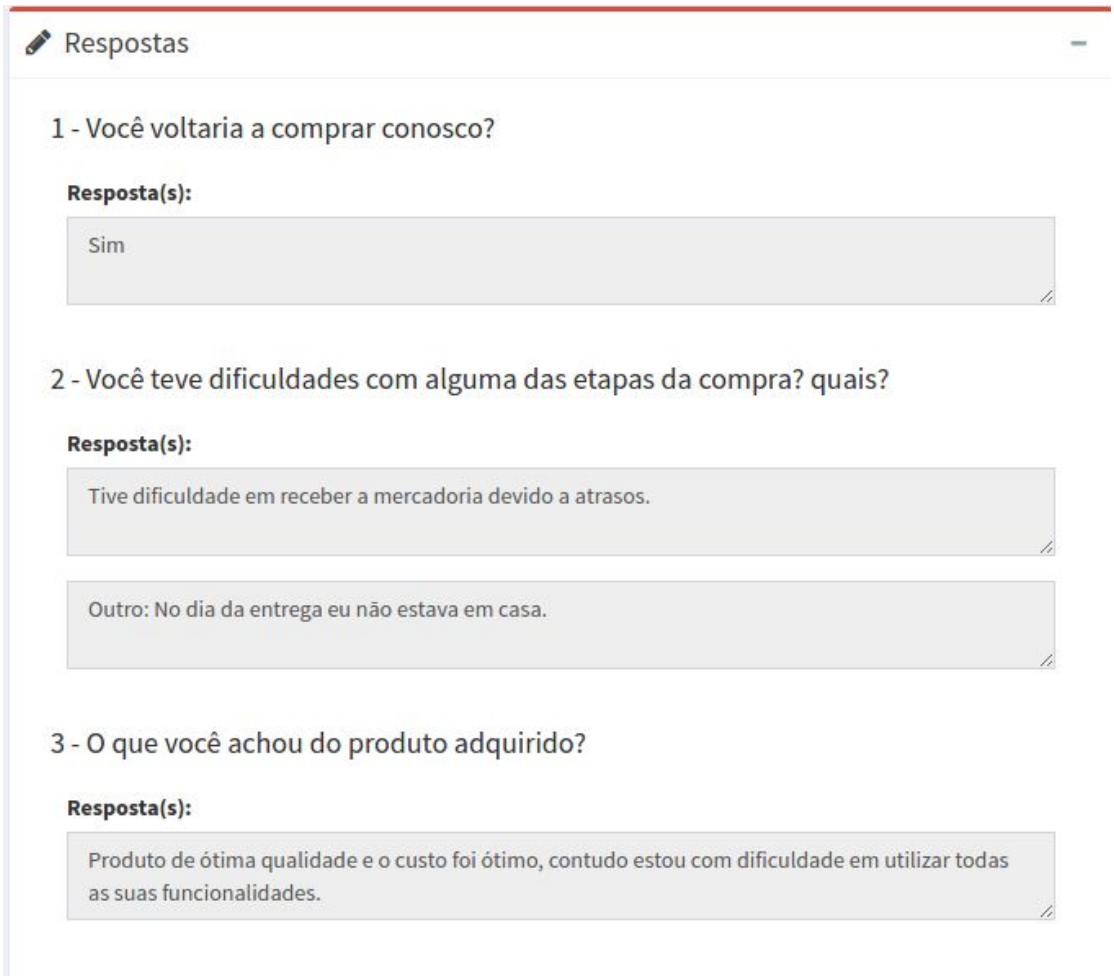

Figura 29. BackHere - Resposta de uma pesquisa

4. Considerações Finais

Este trabalho teve o objetivo e principal contribuição uma ferramenta web que possibilite a criação de estratégias do serviço de pós-venda automatizada, denominada de BackHere. Inicialmente foram introduzidos conceitos e definições sobre e-commerce e pós-venda. Em seguida, uma descrição da ferramenta, suas características, funcionalidades e os aspectos tecnológicos importantes.

4.1 Dificuldades Encontradas

As maiores dificuldade encontradas durante o desenvolvimento deste trabalho foram o desenvolvimento e e validação da ferramenta BackHere.

Para o desenvolvimento a maior dificuldade foi a importação de dados a partir do Magento, onde foi necessário um estudo sobre os recursos de suas APIs e escolha da API SOAP em detrimento da REST. O problema encontrado foi que inicialmente foi cedido por terceiros as credenciais de um Magento para realizar os testes, contudo durante o desenvolvimento deste trabalho esse Magento ficou indisponível e só foi liberado novamente na metade de Junho, o que gerou um atraso no desenvolvimento da ferramenta como um todo.

Para a validação deste trabalho foi acordado com um lojista do e-commerce brasileiro que ele realizaria as validações necessárias sobre a ferramenta e estaria envolvido no processo de melhoria após ser disponibilizada uma primeira versão mínima. Com o atraso gerado no desenvolvimento a ferramenta só teve sua primeira versão mínima finalizada em Julho, esse atraso impediu que as validações fossem realizadas devido a indisponibilidade do lojista no momento.

4.2 Trabalhos Futuros

O próximo passo importantíssimo é realizar o processo de validação da ferramenta com lojistas do e-commerce brasileiro, para detectar os pontos negativos dessa primeira versão mínima e detectar melhorias que precisam ser implementadas.

Contudo, algumas melhorias já são conhecidas e ficam como trabalhos futuros no âmbito da BackHere:

1 - Existem muitas outros sistemas, utilizados pelos lojistas do e-commerce, além do Magento e para que a ferramenta seja capaz de importar os dados para ser utilizada por

muitos clientes torna-se necessário a integração com outros sistemas, por exemplo: Main Retail, BSeller e VTex.

2 - Também com o foco em garantir que a ferramenta possa ser utilizada por muitos clientes, torna-se necessário a implementação e disponibilização de uma API para que os lojistas possam integrar seus dados diretamente com a ferramenta. Muitos lojistas possuem sistemas próprios, o torna inviável desenvolver uma integração direta com esses sistemas.

3 - Um estudo mais aprofundado para melhorar o uso dos recursos do AdminLTE e do Bootstrap.

4 - Geração de relatórios sobre os clientes, vendas e respostas recebidas. Com o atraso no desenvolvimento essa primeira versão não faz um tratamento sobre as respostas e acaba apenas disponibilizando a visualização das respostas recebidas.

5 - Inserção de novas informações no dashboard da ferramenta, assim como um filtro por data.

6 - Implementação do envio de alertas para os lojistas configuráveis de acordo com situações específicas.

7 - Criação de uma suíte de testes que cubra completamente todos os recursos da ferramenta. Essa suíte se torna necessária para evitar que bugs sejam inseridos posteriormente e para garantir o bom funcionamento da aplicação quando tiver alguma das tecnologias utilizadas atualizada.

8 - Remover o Redis do servidor de Worker e passar para um servidor na AWS próprio e posteriormente para o serviço de ElasticCache da AWS.

9 - Remover o uso do serviço do Mlab para hospedagem do MongoDB e passar para um servidor na AWS próprio e posteriormente para um Cluster.

10 - Um estudo sobre segurança e modificações nesse sentido para garantir que a ferramenta seja em um nível seguro como ferramenta web.

5. Referências

[1] KRUGER, G. **O papel do pós-venda na fidelização dos clientes da sua loja virtual.** Disponível em: <https://www.ecommercebrasil.com.br/artigos/o-papel-pos-venda-na-fidelizacao-dos-clientes-da-sua-loja-virtual/>. Acesso em: 08 julho 2016.

[2] MARTINS, P. I. **O papel do pós-venda na fidelização dos clientes da sua loja virtual.** Disponível em: <https://www.ecommercebrasil.com.br/artigos/o-papel-pos-venda-na-fidelizacao-dos-clientes-da-sua-loja-virtual/>. Acesso em: 08 julho 2016.

[3] MENDES, L. Z. R. **E-commerce: origem, desenvolvimento e perspectivas.** Disponível em: <http://hdl.handle.net/10183/78391>. Acesso em: 08 julho 2016.

[4] THE MICHAEL ALDRICH ARCHIVE. **Inventor's Story.** Disponível em: <http://www.aldricharchive.com/inventors_story.html>. Acesso em: 08 julho 2016.

[5] MIVA. **The History Of Ecommerce: How Did it All Begin?.** Disponível em: <https://www.miva.com/blog/the-history-of-ecommerce-how-did-it-all-begin/>. Acesso em: 08 julho 2016.

[6] GUIA DE E-COMMERCE. **Modelos de E-commerce.** Disponível em: <http://www.guiadeecommerce.com.br/modelos-de-ecommerce/>. Acesso em: 09 julho 2016.

[7] DIBONIFACIO, M. **Entendendo os Diferentes Tipos de e-Commerce: B2C, B2B, B2G, B2E, B2B2C, C2C.** Disponível em: <http://www.administradores.com.br/artigos/tecnologia/entendendo-os-diferentes-tipos-de-e-commerce-b2c-b2b-b2g-b2e-b2b2c-c2c/83011/>. Acesso em: 09 julho 2016.

[8] E-BIT. **E-commerce B2C fatura R$ 22,5 bilhões em 2012.** Disponível em: <http://www.profissionaldeecommerce.com.br/faturamento-ecommerce-2012/>. Acesso em: 09 julho 2016.

[9] E-BIT. **E-commerce já tem 51,3 milhões de consumidores no Brasil.** Disponível em: <http://www.profissionaldeecommerce.com.br/e-commerce-ja-tem-51-3-milhoes-de-consumidores-no-brasil/>. Acesso em: 09 julho 2016.

[10] E-BIT. **E-commerce Cresce 24% e vende 35,8 bilhões em 2014.** Disponível em: <http://www.profissionaldeecommerce.com.br/e-commerce-cresce-24-e-vende-358-bilhoes-em-2014/>. Acesso em: 09 julho 2016.

[11] MENDES, R. **E-commerce mantém crescimento em 2015.** Disponível em: <http://www.profissionaldeecommerce.com.br/e-commerce-mantem-crescimento-em-2015/>. Acesso em: 10 julho 2016.

[12] MORAIS, F. **O Processo de Compra Online.** Disponível em: <https://www.ecommercebrasil.com.br/artigos/o-processo-de-compra-online/>. Acesso em: 10 julho 2016.

[13] ILHE, G. **Pós-venda surpreendente: cliente fiel.** Disponível em: <https://www.ecommercebrasil.com.br/artigos/pos-venda-surpreendente-cliente-fiel/>. Acesso em: 10 julho 2016.

[14] ZENDESK. **Brasil e Canais de Atendimento (Infográfico).** Disponível em: <https://www.zendesk.com.br/recursos/pesquisa-canais-de-atendimento-brasil/>. Acesso em: 10 julho 2016.

[15] INFUSIONSOFT. **How After Sales Service Creates Loyalty and Leads.** Disponível em: <https://www.infusionsoft.com/after-sales-service>. Acesso em: 10 julho 2016.

[16] RUBY. **Sobre o Ruby.** Disponível em: <https://www.ruby-lang.org/pt/about/>. Acesso em: 10 julho 2016.

[17] FUENTES, V. B. **Ruby on Rails - Coloque sua Aplicação Web nos Trilhos.** 19.2.12 v. São Paulo: Casa do Código, 2016.

[18] NNSOFT. **Cron Format.** Disponível em: <http://www.nncron.ru/help/EN/working/cron-format.htm>. Acesso em: 10 julho 2016.

[19] BRIZENO, N. **Refatorando com Padrões de Projeto - Um Guia em Ruby.** 19.5.13 v. São Paulo: Casa do Código, 2016.