UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS

Matheus Alberto Ambrosi

Desenvolvimento de uma ferramenta de gestão de relacionamento com o cliente para o pós-venda

Florianópolis
2018

Matheus Alberto Ambrosi

Desenvolvimento de uma ferramenta de gestão de relacionamento com o cliente para o pós-venda

Relatório submetido à Universidade Federal de Santa Catarina como requisito para a aprovação na disciplina **DAS 5511: Projeto de Fim de Curso** do curso de Graduação em Engenharia de Controle e Automação.
Orientador(a): Prof. Ricardo José Rabelo

Florianópolis

2018

Matheus Alberto Ambrosi

Desenvolvimento de uma ferramenta de gestão de relacionamento com o cliente para o pós-venda

Esta monografia foi julgada no contexto da disciplina DAS5511: Projeto de Fim de Curso e aprovada na sua forma final pelo Curso de Engenharia de Controle e Automação.

Florianópolis, 30 de julho de 2018

Banca Examinadora:

Ricardo José Rabelo
Orientador na Empresa
Universidade Federal de Santa Catarina

Prof. Ricardo José Rabelo
Orientador no Curso
Universidade Federal de Santa Catarina

Prof. Rodrigo Castelan Carlson
Avaliador
Universidade Federal de Santa Catarina

Gabriel Monteiro de Souza
Debatedor
Universidade Federal de Santa Catarina

João Henrique Renon Heinzen
Debatedor
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Antes de tudo gostaria de agradecer a minha família, que sempre me apoiou e me encorajou a seguir os meus sonhos, mesmo que eles fossem longe de casa. A vocês devo esse projeto, devo esses anos de faculdade e principalmente todo o fruto que virá com a consolidação do diploma de uma engenharia cujo curso é considerado o melhor do pais. A vocês, Nicanor Ambrosi, Marilei Rigo Ambrosi e Eduarda Ambrosi, obrigado.

Não poderia deixar de agradecer também, todos aqueles que me apoiaram nesse caminho e fizeram com que essa jornada pudesse ser de diferentes aprendizados e experiências, como por exemplo, meus colegas e amigos que me acompanham desde o primeiro dia de aula, Roger Perin, Murilo Rodegheri, Victor Gabriel Petrassi e Guilherme Cornelli e a turma 13/1.

A todos que participaram dos primeiros ensinamentos sobre gestão e me acompanharam no movimento empresa júnior, Pedro Casali, Gustavo Kremer, Victor Gabriel Petrassi, Mateus Gabriel Bosa e a equipe e gestão 2014 da Autojun.

A todos que me ensinaram sobre trabalho em equipe sobre pressão em uma competição de nível internacional na equipe Céu Azul Aeronaves.

A todos que me ensinaram a aprender a qualquer custo durante minha experiência no Einstein Floripa, curso pré-vestibular gratuito sem fins lucrativos voltado para pessoas de baixa renda da região da Grande Florianópolis.

A todos que ensinaram os primeiros passos da análise estratégica de negócios e me mostraram como um time bom é o principal ativo de uma empresa e como trabalhar se torna leve e prazeroso nesse contexto, Felipe Rosa, Lui Pillmann, José Lacerda, Rafael Savi e Tatiana Casarotto, muito obrigado.

Obrigado a todos que me receberam e me possibilitaram contribuir com startups early stage de maior impacto no Oriente Médio e norte da África, a equipe do AUC Venture Lab, muito obrigado.

Como não agradecer uma das experiências mais desafiadoras e inspiradoras que me motivaram a empreender e a buscar sempre evoluir, ao time da 99labs, Pedro Shioga, Lui Pillmann e Pedro Casali.

Ao time que me acompanha em um dos projetos mais desafiadores da minha vida e com um grande potencial, Vinícius Dall'Agnol, Vinicius Neves, Leandro Mangini. Aos meus companheiros que compartilham dessa jornada, Gustavo Kremer, Eduardo Schimdt, Matheus Winter.

Finalmente, não poderia deixar de agradecer a pessoa que me acompanhou durante esse projeto e nos momentos de pressão subsequentes, Victória Milanez, muito obrigado.

RESUMO

Este trabalho tem como objetivo desenvolver uma ferramenta para gestão do relacionamento com o cliente no pós-venda como prova de conceito. O problema a ser resolvido é a dificuldade de gerir e monitorar a jornada do cliente com a empresa de forma consistente e previsível. Por meio das metodologias de desenvolvimento ágil de software e de negócios como Design Science e Lean Startup, o projeto teve interações com possíveis clientes em suas etapas, como para a definição do problema, definição das funcionalidades e requisitos, implementação e como avaliação desses resultados. Dessa forma, esse sistema permite a empresa gerenciar as etapas do relacionamento com o cliente, suas interações e dessa forma avaliar o desempenho do processo com o objetivo de aumentar sua eficiência e escalabilidade. No projeto utiliza-se Unified Modeling Language (UML) para descrever o funcionamento e arquitetura da solução. Na implementação foi utilizado uma arquitetura Model-View-Controler (MVC) por meio do framework Laravel PHP no backend e Vue.js no frontend. O resultado foi o Minimum Viable Product (MVP) com uma avaliação de importância e potencial de resultados com um cliente.

Palavras-chave: Desenvolvimento ágil de software. Sucesso do cliente. Relacionamento com o cliente.

ABSTRACT

This work aims to develop a tool for customer relationship management for post-sale as proof of concept. The problem to be solved is the difficulty of managing and monitoring the client's journey with the company in a consistent and predictable way. Through agile software and business development methodologies such as Design Science and Lean Startup, the project had interactions with potential customers in its stages, such as problem definition, functional and requirements definition, implementation, and evaluation of these results. In this way, this system allows the company to manage the stages of the customer relationship, its interactions and thus evaluate the performance of the process in order to increase its efficiency and scalability. In the project, Unified Modeling Language (UML) is used to describe the operation and architecture of the solution. In the implementation, a Model-View-Controller (MVC) architecture was used through the Laravel PHP framework in backend and Vue.js in the frontend. The result was a Minimum Viable Product (MVP) with an assessment of the importance and potential of results with an actual client.

Key-words: Agile Software Development. Customer Success. Customer Relationship Management.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

B2B – Business-to-Business

EUA – Estados Unidos das Américas

CRM – Customer Relationship Management ou Manager

CS – Customer Success

CSM – Customer Success Management ou Manager

API – Application Programming Interface

NPS – Network Promoter Score

AWC – Academic Working Capital

ACATE – Associação Catarinense de Tecnologia

ABES – Associação Brasileira de Software

TI – Tecnologia da Informação

MVP – Minimum Valuable Product

UML – Unified Modeling Language

SaaS – Software as a Service

SUMÁRIO

1 INTRODUÇÃO

Vivemos em um momento único na história em que um cliente tem mais acesso a informação como nunca, tanto para explorar o problema encontrado quanto em avaliar a melhor solução possível para o dado momento. No contexto de um negócio isso se traduz em uma grande dificuldade em adquirir novos clientes e em uma maior importância em aproveitar ao máximo os clientes atuais.

A importância do pós-venda transparece nos negócios assim como nos EUA que, só em 2016, foram mais de 62 bilhões de dólares perdidos (NEW VOICE MEDIA, 2016) pelas empresas americanas por experiências ruins dos clientes ao passo que hoje é de 5 a 25 vezes mais caro para uma empresa adquirir um novo cliente do que investir em sua retenção (HARVARD BUSINESS REVIEW, 2014).

O problema se agrava ainda mais em empresas de serviço que tem empresas como cliente (*business-to-business* ou B2B) em que o relacionamento com o cliente é visto como diferencial competitivo dado o investimento e o risco desses negócios (FORBES, 2014).

Ainda, além da importância da retenção dos clientes, seu sucesso é considerado um novo modelo de vendas, já que é 60% a 70% mais fácil vender novamente para alguém que já é cliente (FORBES, 2013) e que suas indicações geram interessados cerca de 20% mais lucrativos (HARVARD BUSINESS REVIEW, 2011).

Isso se traduz nos negócios em investimentos para melhorar a experiência do cliente e garantir que ele obtenha resultado e se sinta satisfeito, sendo que cerca de 84% das empresas nos EUA já se consideram preocupadas com a experiência do cliente (GARTNER, 2014).

Porém hoje no Brasil, não existe uma ferramenta específica, focada no pós-venda para empresas que investem no sucesso do cliente como estratégia e que atendam não só empresas de tecnologia, mas também empresas tradicionais e suas necessidades.

Esse trabalho é fruto da criação e desenvolvimento da Centrics, uma empresa e também uma ferramenta de relacionamento com o cliente focada no pós-venda e como tal tem como missão criar conteúdo e desenvolver tecnologia para empresas que investem no sucesso do cliente como ferramenta de crescimento.

Mais detalhes sobre a empresa no capítulo 2.

O projeto e criação do negócio teve como base entrevistas e interações com mais de 80 empresas de tecnologia de todo o Brasil e teve como inspiração a experiência profissional dos sócios trabalhando em empresas de tecnologia sediadas em Florianópolis, onde hoje é considerado um ecossistema (G1, 2018). Esse ecossistema de empresas de tecnologia tem como característica o investimento nas melhores práticas de desenvolvimento de produto e em garantir com que seus clientes obtenham resultados dos mesmos.

Uma das estratégias que se destacaram nesse meio foi a denominada ‘Gestão do Sucesso do Cliente’ (RESULTADOS DIGITAIS, 2016). Essa estratégia coloca os clientes no centro da empresa e busca, por meio do relacionamento proativo com a base de clientes, fazer com que sistematicamente atinjam seu sucesso desejado fazendo com que: fiquem mais satisfeitos, indiquem o serviço para outros clientes, e que renovem e aumentem seus contratos.

1.1 PROBLEMA

Para que essa estratégia seja bem executada, as empresas passam por 4 passos (LINCOLN MURPHY, 2017):

- Segmentar a base de clientes
- Definir o sucesso do cliente para cada segmento
- Implementar a gestão do sucesso do cliente
- Medir e aprimorar o processo

Segundo Lincoln, o Sucesso do Cliente é quando o cliente encontra seu sucesso desejado por meio de interações com a empresa. As interações são qualquer tipo de contato entre o cliente e a empresa, seja ele por motivos de marketing, vendas, suporte, atendimento, relacionamento ou qualquer outro. Já o sucesso desejado é dividido em sucesso requerido e uma experiência adequada, podemos verificar a diferença dessa estratégia com as demais na Figura 1.

Customer Success	Pós-venda / SAC / Atendimento
• É uma estratégia da empresa que reflete sua cultura • Mantra que vem do empreendedor e é transmitido ao time • É proativo: a área de CS engaja o cliente • É estruturado: existem processos claros de engajamento	• É um departamento da empresa • É responsabilidade das áreas de atendimento • É reativo: o cliente liga para o SAC • É transacional

Figura 1 - O que é e o que não é sucesso do cliente (ENDEAVOR, 2018).

O sucesso requerido se refere ao motivo de compra de uma solução, sendo geralmente uma métrica de negócio (vender mais, ser mais eficiente). A experiência adequada é o diferencial competitivo do fornecedor, o por que ele comprou essa solução com esse fornecedor.

Definindo o sucesso do cliente, o próximo passo é fazer a gestão desse sucesso, ou seja, garantir proativamente que seus clientes atinjam seu sucesso desejado. Para que isso aconteça, tanto o cliente, quanto o serviço contratado precisam realizar ações ou atividades (que podem ser agrupadas em etapas, dada as entregas parciais) e que são chamadas de jornada do sucesso do cliente.

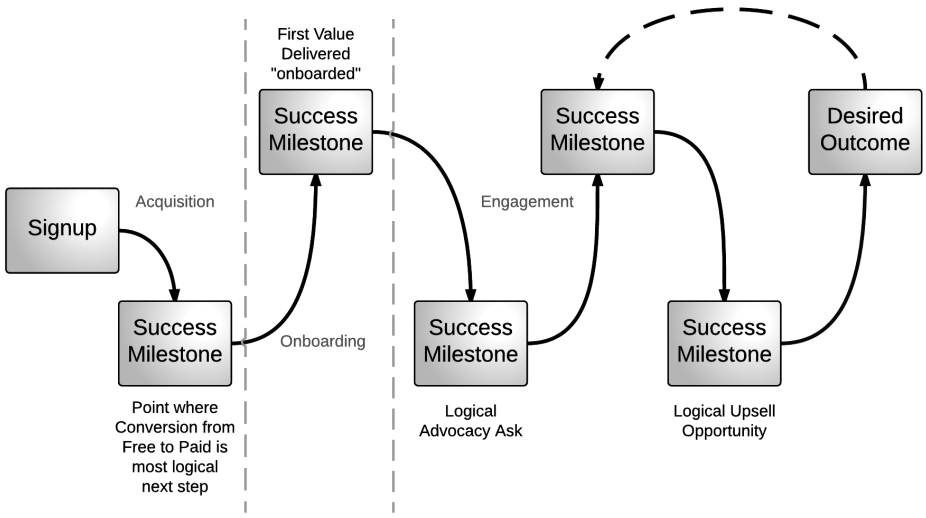


Figura 2 - Gestão do sucesso do cliente (GAINSIGHT, 2016).

O meio pelo qual essa jornada é orquestrada é o relacionamento com o cliente e suas ações, sejam elas internas (criação de relatórios, cobranças, conteúdos, gestão melhoria em outras áreas, entre outros), sejam elas com o cliente (treinamentos, revisão executiva de negócios, consultoria).

Com o passar do tempo a utilização do serviço evolui, seja por meio de uma maior contratação ou contratação de serviços complementares. Com essa evolução, o sucesso desse segmento de cliente muda, e sua experiência adequada também, fazendo com que o processo precise ser melhorado continuamente.

Com base nas entrevistas feitas com mais de 80 empresas que possuem áreas, times e cargos de sucesso do cliente, o aluno identificou 3 grandes dificuldades, tanto por testemunho das empresas quanto pela a análise de suas operações:

- Visualizar a situação do relacionamento de cada cliente
- Monitorar o desempenho desses clientes no processo de relacionamento
- Avaliar o desempenho do processo e da equipe
- Escalar o processo com o aumento do número de clientes

1.2 METODOLOGIA EMPREGADA

Para o desenvolvimento do projeto foi utilizada uma metodologia mais geral, a *Design Science* (JÄRVINEN, 2007), organizando o desenvolvimento do trabalho através dos passos de: entendimento do problema, sugestão, desenvolvimento, avaliação, até chegar na conclusão dessa pesquisa.

No escopo do *Design Science* e seus procedimentos, o aluno adicionou o ciclo construir, medir e aprender do *Lean Startup*, envolvendo técnicas de desenvolvimento de negócios, gestão de produtos e desenvolvimento ágil de software, passando por esse ciclo a cada procedimento do Design Science.

Nessas metodologias se tem como entregas artefatos, que em determinados procedimentos são uma lista de necessidades dos clientes, e em outros a prova de conceito do sistema em si.

Mais informações sobre a metodologia empregada no Capítulo 3.

1.3 ALTERNATIVAS EXISTENTES

De acordo com as entrevistas realizadas durante esse projeto, hoje, as empresas que possuem times de relacionamento com o cliente ou até mesmo times de sucesso do cliente, utilizam 4 alternativas para fazer a gestão do sucesso do cliente ou a gestão das atividades desse time, são elas: ferramentas de gestão do relacionamento com o cliente (Customer Relationship Management - CRM), planilhas, ferramentas de automação de e-mails ou soluções feitas especificamente para o sucesso do cliente e que são chamadas: ferramentas de gestão do sucesso do cliente (Customer Success Management - CSM).

Porém, também pelas entrevistas realizadas, identificou-se algumas limitações dessas soluções na visão do usuário, como a linguagem, funcionalidades específicas e as métricas utilizadas.

Mais informações sobre as alternativas existentes no Capítulo 4.

1.4 SOLUÇÃO ABORDADA

O objetivo desse trabalho é por meio de metodologias de desenvolvimento de software e de negócio, conceitualizar, desenvolver e protótipo uma prova de conceito como solução de relacionamento com o cliente para pós-venda, com foco em empresas de serviço e que dará base a criação dessa empresa.

A prova de conceito nesse caso, é o desenvolvimento de um Minimum Viable Product (MVP) como descrito por Eric Ries no livro *Lean Startup.* Nesse sentido, trata-se da maneira mais rápida de percorrer o ciclo construir-medir-aprender de *feedback* com o menor esforço possível e que, ao contrário do desenvolvimento de produto tradicional que aspira a perfeição do produto, o objetivo do MVP é começar o processo de aprendizagem, não termina-lo (RIES, 2012). Também, diferente de um protótipo, um MVP é projetado não só para responder perguntas técnicas ou de design de produto, seu objetivo é testar hipóteses fundamentais de negócio (RIES, 2012).

No caso desse projeto, o MVP será capaz de:

- Gerenciar etapas e atividades: Permitirá ao usuário adicionar, editar e remover etapas e as atividades dentro delas. A essas atividades serão atribuídas datas, responsáveis e a que cliente pertencem;

- Visualizar a base de clientes da empresa: Permitirá ao usuário visualizar todos os clientes que estão em sua base, com informações básicas como nome, segmento de mercado e contato do responsável pela empresa. A inclusão do cliente poderá ser feita manualmente ou pela integração da aplicação com um .csv ou API aberta de uma aplicação terceira de CRM de vendas;
- Gerenciar interações: Permitirá ao usuário adicionar, editar e remover interações proativas com o cliente, que são anotações feitas pelo usuário, atribuídas a um cliente e que possuem um indicativo de "humor do cliente" para verificar a qualidade da interação. Essa funcionalidade também permitirá a inclusão de resultados de uma pesquisa de Net Promoter Score (NPS) para armazenar essa informação.
- Visualizar desempenho do processo: Permitirá ao usuário visualizar diferentes métricas da aplicação, relacionadas as atividades e as interações, organizadas de acordo com o nível de granularidade como: atividades executadas por cliente, por usuário e pela empresa.

Assim sendo, o objetivo final é desenvolver um MVP que possibilite times de relacionamento com o cliente a diminuir o tempo gasto para monitorar as atividades e o que é feito com cada cliente, com o objetivo final de diminuir os custos e aumentar a receita com a base de clientes.

1.5 CONTEXTO NO CURSO

A ideia do projeto é desenvolver uma prova de conceito que será o ativo central de uma empresa, usando conhecimentos de engenharia para especificar e gerenciar tanto o desenvolvimento como o negócio.

O aluno irá utilizar conhecimentos aprendidos em matérias relacionadas a área de informática como: Introdução à Informática para Automação (DAS5334), Fundamentos da Estrutura da Informação (DAS5102), Banco de dados e principalmente das matérias de Metodologia para Desenvolvimento de Sistemas (DAS5312), Avaliação de Desempenho de Sistemas de Automação Discretas (DAS5310) e Integração de Sistemas Corporativos (DAS5316).

Além disso, o aluno tomou como base conhecimentos aprendidos em matérias optativas de outros cursos como: Modelagem e automação de processos de negócio

(INE), e conhecimentos de atividades extracurriculares, mas que fazem para da engenharia, como gestão de projetos.

2 CENTRICS

Como comentado anteriormente, esse trabalho é fruto da criação e desenvolvimento da Centrics, uma empresa e também uma ferramenta de relacionamento do cliente focada no pós-venda e como tal tem como missão criar conteúdo e desenvolver tecnologia para empresas que investem no sucesso do cliente como estratégia de crescimento.

Para ajudar as empresas com essa estratégia, a Centrics oferece uma série de serviços de consultoria, como facilitação, workshops, operacionalização até a implantação da ferramenta, divididos em 3 grupos como descrito na Tabela 1:

Tabela 1 - Serviços da empresa Centrics.

Estratégia	Segmentação	Orquestração
Identificação de oportunidades de negócio	Segmentação da base de clientes	Operacionalização da gestão do sucesso do cliente
Análise da jornada do cliente	Definição dos resultados desejados e marcos de sucesso	Mentoria com o responsável da área
Redesenho de processos internos	Validação da experiência do cliente	Métricas e melhoria contínua

A Centrics já ajuda empresas de tecnologia e de recorrência a criar e operacionalizar esse processo por meio de uma metodologia própria, baseada nas melhores práticas encontradas ao redor do mundo. Com essas experiências identificamos os principais problemas encontrados em operacionalizar uma área de gestão sucesso do cliente, utilizando esse projeto de fim de curso para desenvolver a solução.

Com a construção da empresa e da proposta de valor do MVP resultante desse trabalho, obteve-se uma série de conquistas com programas de inovação, tanto em programas regionais como em programas nacionais.

Desde sua criação, em fevereiro de 2018, a Centrics conquistou diversas oportunidades para o desenvolvimento do negócio. Entre elas estão o *Academic Working Capital* (AWC), oferecido pelo Instituto TIM e o Link Lab, um programa de inovação aberta oferecido pela Associação Catarinense de Empresas de Tecnologia (ACATE)

2.1 CONQUISTAS

O AWC é um programa de empreendedorismo científico, oferecido apenas para estudantes que estão matriculados em instituições de ensino superior e em matérias que envolvam um projeto ou trabalho de conclusão de curso.

A proposta do programa é ajudar estudantes com ideias com potencial de negócio a tirarem a ideia de papel e eles fazem isso por meio de um programa que dura 1 ano. Isso envolve desde Workshops Online e Presenciais, a Orientações com mentores experientes, apoio financeiro no valor de 30 mil reais e ao final do programa, participação em uma feira de investimento onde as empresas poderão ser investidas ou aceleradas, essas iniciativas são demonstradas na Figura 3.

Figura 3 - Funcionamento do AWC (AWC).

Já o Link Lab é um programa de inovação aberta proporcionado pela ACATE e patrocinado por grandes empresas do estado de Santa Catarina, entre elas o Grupo

Nexera, Ambev, Cesusc, Engie, Flex, Marisol, Qualirede, Teltec, Brognoli e WEG. É um espaço de conexão entre grandes empresas, startups, fundos e parceiros que querem desenvolver negócios em um dos ambientes mais inovadores da América Latina.

Seguindo o conceito de inovação aberta, o Link Lab permite às grandes empresas o acesso ao ecossistema de inovação de Santa Catarina. As empresas co-fundadoras e patrocinadores convivem em um ambiente altamente inspirador onde as startups selecionadas ficarão residentes durante períodos de 4 meses, prorrogáveis a critério das grandes empresas.

Ocupa uma área de 640m² em um ambiente colaborativo com 8 salas privadas, uma para cada empresa âncora, e amplo espaço de *coworking*, dedicado às *startups* selecionadas. Neste espaço, acontecem os cursos, eventos de *network*, *webinars*, *workshops*, *pitchs*, mentorias coletivas e individuais, além do apoio especializado dos parceiros e prestadores de serviço, como: escritórios de advocacia, contabilidade, comunicação, marketing, etc.

Ao ser selecionado por uma patrocinadora, a Centrics hoje faz parte da segunda turma do Link Lab e tem a ACATE como escritório.

3 METODOLOGIA

O objetivo desse capítulo é apresentar de forma concisa como foi projetado e desenvolvido o sistema, que culminou com um a implementação do MVP da Centrics.

Para o desenvolvimento dessa prova de conceito foram utilizadas diferentes metodologias, tanto para a conceitualização, projeto quanto para o desenvolvimento. Essas metodologias foram escolhidas por se encaixarem no tipo de problema a ser resolvido.

O *Design Science* foi escolhido como metodologia geral, pois possibilita explorar diferentes aspectos do projeto, como o problema, a solução e a avaliação de forma interativa, acelerando o aprendizado e diminuindo o risco. Já o *Lean Startup* foi utilizado por possibilitar ciclos rápidos, envolvendo construção e aprendizado, também diminuindo o risco de implementação.

3.1 DESIGN SCIENCE

Segundo Gil (GIL, 2010), a pesquisa é o procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos. A pesquisa se desenvolve com a utilização de planejamento, métodos, técnicas e outros procedimentos científicos ao longo de um processo que envolve inúmeras atividades até a apresentação dos resultados finais de acordo com os objetivos traçados.

Considerando os objetivos deste projeto, o método de pesquisa utilizado para desenvolver a proposta deste projeto foi o *Design Science* (JÄRVINEN, 2004). Existem inúmeras definições para esse método, conforme o domínio de aplicação e áreas envolvidas (como as ciências sociais e as tecnológicas). Com base nas análises de Järvinen (JÄRVINEN, 2007), pode-se dizer em termos gerais que Design Science se refere ao desenvolvimento de artefatos (de conhecimento, de TI, ideias, teorias, etc.) para se resolver ou melhorar um dado problema já suficientemente compreendido. Isto é feito através de melhoramentos efetuados ciclicamente a partir de teorias e conhecimentos já existentes, e cujos resultados das melhorias possam ser concretamente avaliados, e, por fim, modificar o estado de conhecimento vigente. A Figura 4 ilustra o método *Design Science*.

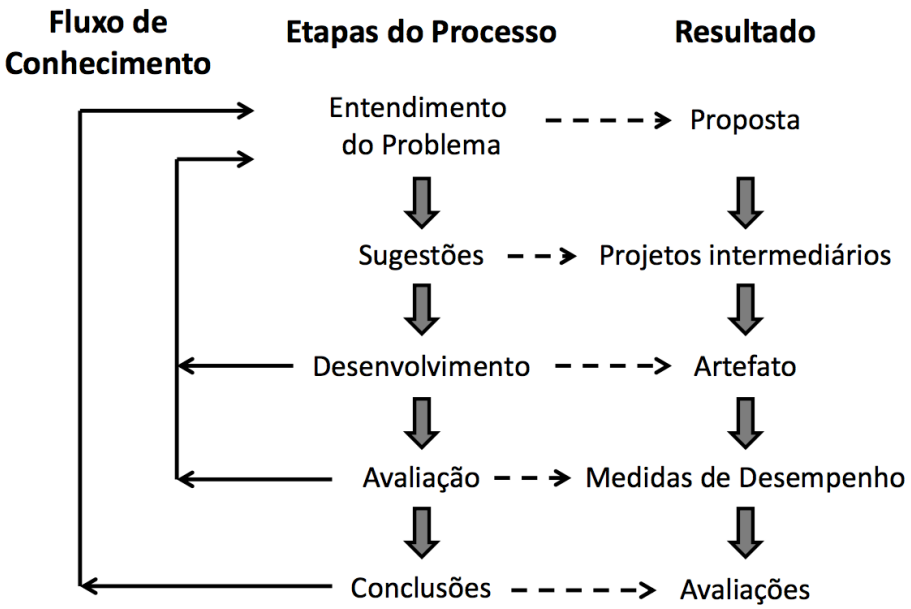


Figura 4 - Método Design Science (JÄRVINEN, 2007).

Ressalta-se que no *Design Science* pode haver um avanço e retorno de etapas, em ciclos, não necessariamente simétricos, dependendo da dinâmica determinada dos pesquisadores, do número de artefatos sendo tratados, das suas inter-relações, e da situação pesquisada (JÄRVINEN, 2004).

Neste sentido, todas as atividades desse projeto se enquadraram nas etapas do processo desse método de pesquisa e que serão sumarizadas nas seções a seguir. O detalhamento de cada uma dessas etapas dar-se-á através dos capítulos subsequentes. A adoção do *Design Science* significa essencialmente também a adoção de uma estratégia de pesquisa incremental, onde já consolidadas fundamentações teóricas existentes em relação ao relacionamento com cliente e engenharia de software serão aproveitadas, tomadas como base e adaptadas para o desenvolvimento da prova de conceito resultante do projeto.

3.2 LEAN STARTUP

Nos procedimentos de: “Entendimento de problema”, “Sugestão” e “Desenvolvimento” estipulados pelo *Design Science*, o aluno também utilizou a metodologia *Lean Startup* que é um conjunto de processos usados por

empreendedores para desenvolver produtos e serviços combinando Desenvolvimento Ágil de Software e Desenvolvimento de clientes (RIES, 2012).

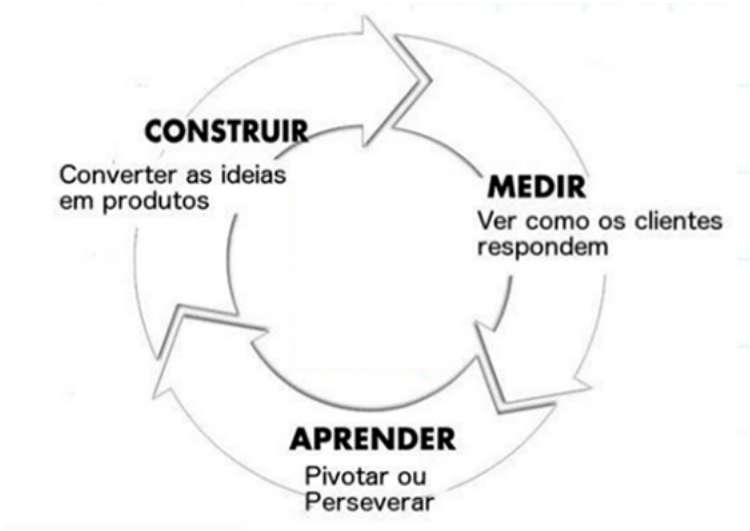


Figura 5 - Ciclo Construir-Medir-Aprender do Lean Startup

No procedimento de “Entendimento do problema”, foram utilizadas as técnicas de desenvolvimento de clientes previstas no *Lean Startup* e baseadas no *Customer Development* do autor Steven Blank (BLANK, 2007), construindo-se hipóteses de problemas de negócio enfrentados pelos diferentes segmentos de mercados para os quais a solução será feita e aprendendo se essas hipóteses são verdadeiras ou não utilizando como medida entrevistas semiestruturadas.

No procedimento de “Sugestões”, são utilizadas técnicas de *gestão de produtos enxutos* (OLSEN, 2015): a prototipação do software via *wireframes* e *mockups*, aprendendo a respeito da importância e facilidade de uso da solução, por meio de testes com usuários;

Já no procedimento de “Desenvolvimento” o aluno criou protótipos rápidos, projetados e implementados de acordo com os *wireframes* e avaliação dos usuários. O aprendizado nessa etapa é a resolução dos problemas encontrados, também será medido por meio de testes com usuário.

A utilização dessas metodologias resulta na seguinte organização desse projeto, assim como na Figura 6:

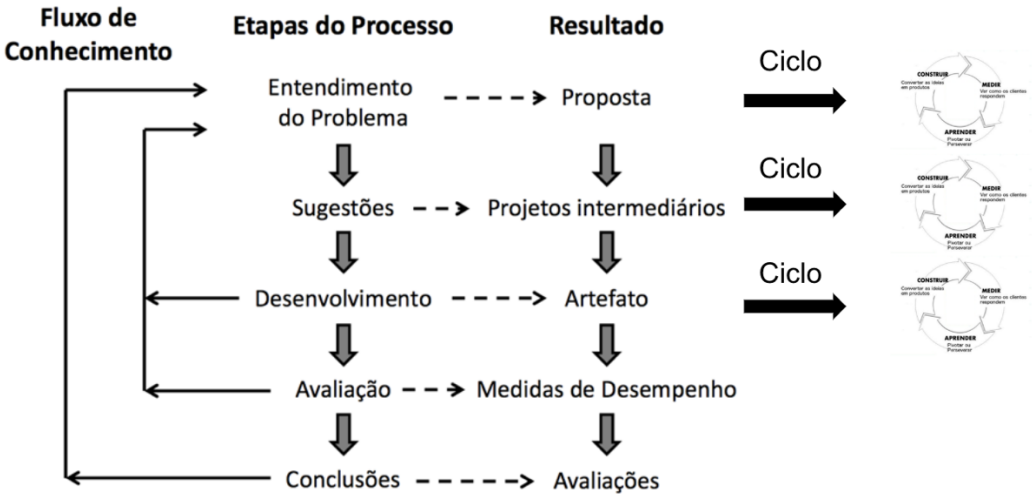


Figura 6 - Adaptação do Design Science e Lean Startup nesse trabalho.

Importante ressaltar que nos procedimentos de “Avaliação” e “Conclusão” não foram utilizados os ciclos do *Lean Startup*. O aluno considerou que esses dois procedimentos são resultantes do aprendizado no desenvolvimento e correspondentes a análise dos resultados do desenvolvimento em si, não sendo necessárias as construções de hipóteses, métricas e aprendizados.

3.3 PROCEDIMENTOS PARA O DESENVOLVIMENTO

As seções a seguir sumarizam os passos metodológicos realizados ao se seguir o *Design Science.*

3.3.1 PASSO 1 – ENTENDIMENTO DO PROBLEMA

Este passo iniciou-se pela identificação do problema em si. Isto deu-se a partir de leituras de artigos sobre relacionamento com o cliente, páginas e blogs técnicos na Internet, e principalmente por experiências empíricas relatadas por empresas entrevistadas que já possuem times de sucesso do cliente.

Complementarmente, utilizou-se da experiência profissional do autor, que vivenciou várias iniciativas de implantação de processos e times de sucesso do cliente enquanto prestava esse serviço como consultoria.

3.3.1.1 CONSTRUÇÃO

Considerando o objetivo do projeto, foi necessário compreender inicialmente as teorias de base existentes a ela relacionadas. Mais especificamente, sobre o relacionamento com o cliente nas empresas do Brasil, sobre o processo geral de gestão do sucesso do cliente, e sobre alguns modelos de referência e técnicas utilizadas para implantação e operação desse processo. Com as informações coletadas foi possível estabelecer hipóteses a serem validadas:

- As empresas já estão perdendo dinheiro com a saída de clientes?
- Gerenciar o relacionamento com o cliente após a venda é um problema?
- Elas gastam tempo e dinheiro para gerenciar esse relacionamento?

3.3.1.2 MEDIÇÃO

De acordo com as técnicas de desenvolvimento de clientes (BLANK, 2007), a validação de hipóteses foi feita por meio de entrevistas semiestruturadas, com script pré-definido, presencialmente ou por vídeo conferência, limitando somente a entrevista sem recurso visual.

3.3.1.3 APRENDIZADO

Com base em mais de 80 entrevistas com empresas do segmento de serviço B2B, sendo elas tecnológicas ou não, verificou-se dois problemas na gestão do sucesso do cliente: a dificuldade em avaliar o desempenho, tanto do processo quanto da equipe e a dificuldade em escalar esse processo com o aumento da base de clientes.

Dado o grande escopo envolvido quando se considera os diferentes tipos de serviços que existem nesse segmento e a grande diferença entre eles, o passo de entendimento do problema foi de certa forma finalizado com a validação das hipóteses, em que esses problemas são reais e que as empresas já investem para resolve-lo de diferentes formas.

3.3.1.4 ARTEFATO

Com isso foi possível delimitar um escopo de necessidades ou requisitos que o projeto abordaria. Esse passo tem como artefato, uma lista de necessidades a serem resolvidas:

- Diminuir o tempo gasto analisando a situação de cada cliente;
- Aumentar a previsibilidade da jornada do sucesso do cliente;
- Aumentar o conhecimento das interações e satisfação do cliente com o processo;

3.3.2 PASSO 2 – SUGESTÕES

Neste passo foi feito um levantamento bibliográfico com o objetivo de consolidar os conceitos relacionados à proposta geral do projeto bem como iniciar um aprofundamento na identificação dos fatores envolvidos na gestão do sucesso do cliente. Este levantamento foi realizado através da leitura de livros, artigos, analises de cases de sucesso, analise de demonstração de outros produtos que resolvem esse problema no exterior.

3.3.2.1 CONSTRUÇÃO

Com essas informações e as necessidades declarada nas entrevistas de entendimento do problema, foi possível estabelecer hipóteses de quais eram os requisitos funcionais do sistema a ser desenvolvido, que atendiam diretamente as necessidades do usuário, seguindo os preceitos de um produto enxuto, assim como o conceito de *Minimum Value Product* (MVP) do *Lean Startup*.

Essas funcionalidades foram:

- Gerenciar a situação do cliente no processo de relacionamento;
- Gerenciar as interações com o cliente;
- Gerenciar o desempenho dos clientes no processo;

De acordo com Olsen (OLSEN, 2015), utilizou-se de *wireframes* para tangibilizar ao usuário como o sistema irá funcionar. Utilizando a ferramenta de prototipagem Pidoco, criou-se telas web que exemplificam a solução final a ser desenvolvida. Assim como o exemplo na figura abaixo:

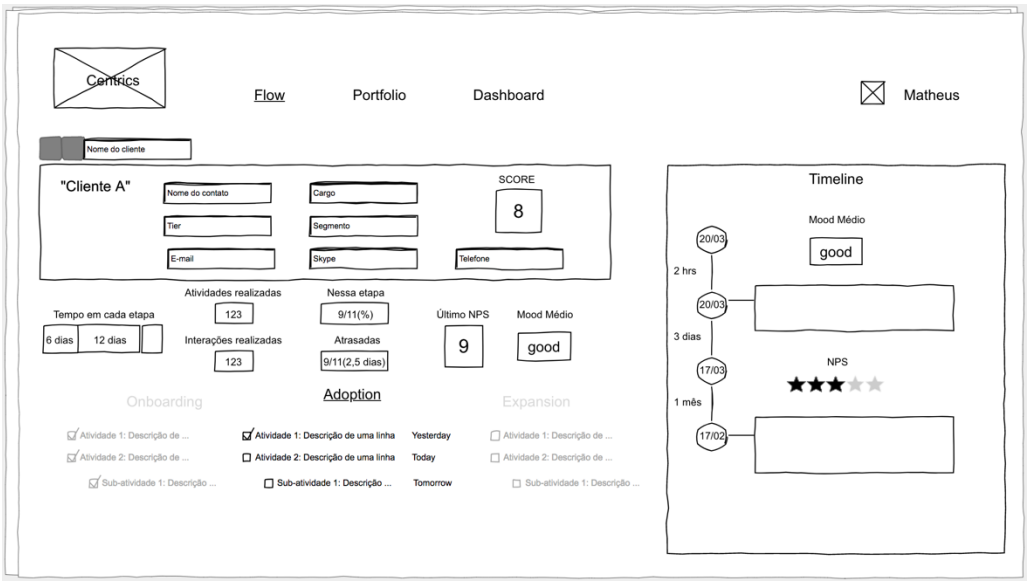


Figura 7 - Wireframe inicial do MVP.

3.3.2.2 MEDIÇÃO

Também de acordo com Olsen (OLSEN, 2015), a validação das hipóteses foi feita por meio de 10 entrevistas para avaliação, com script pré-definido, presencialmente ou por vídeo conferência, limitando somente a entrevista sem recurso visual.

O script é feito com perguntas a respeito da importância e facilidade de uso da solução conforme o usuário interage com os *wireframes*. Os resultados obtidos foram uma nota média de 7 para facilidade de uso e 9 de quão valioso é para eles.

3.3.2.3 APRENDIZADO

De acordo com a utilização dos *wireframes* e dos *feedbacks* apresentados foi possível aprimorar as funcionalidades e principalmente a maneira como os usuários iriam interagir com essas funcionalidades.

3.3.2.4 ARTEFATO

Assim, com informações a respeito da validação de usuários, foi elaborado uma lista com pontos de melhoria na experiência do usuário, nas funcionalidades e no modo de utilização dessas funcionalidades.

Com a melhoria no protótipo conceitual da solução foi possível definir os requisitos de software, bem como a modelagem de dados. O artefato de entrega desse passo é o projeto do sistema:

- Requisitos de software
- Modelagem de software em Unified Modeling Language (UML)
- Arquitetura
- Planejamento do desenvolvimento

3.3.3 PASSO 3 – DESENVOLVIMENTO

Esse procedimento é caracterizado pelo projeto e implementação da prova de conceito da solução, que consiste nas funcionalidades validadas por meio dos *wireframes* do procedimento de “Sugestão” correspondentes e que por sua vez correspondem a necessidades validadas pelo procedimento de “Entendimento do problema”.

Essa etapa foi responsável pelo desenvolvimento do MVP em si, que consiste nas funcionalidades descritas no artefato do Passo 2, e que correspondem ao objetivo desse projeto.

3.3.3.1 CONSTRUÇÃO

A implementação foi feita de acordo com os casos de uso e de acordo com o Planejamento do desenvolvimento, em que o aluno descreve quais são as funcionalidades entregues durante as *sprints.*

O desenvolvimento foi feito utilizando a arquitetura *Model-View-Controller* como estipulado pelo Passo 2, utilizando as seguintes tecnologias:

- Backend: Laravel PHP
- Frontend: Vue.js
- Persistência: MySQL

3.3.3.2 MEDIÇÃO

Ao final de cada *sprint* foram feitas rodadas de avaliação por parte do cliente, que identificou potenciais de melhoria e direcionou a priorização do desenvolvimento. Essa avaliação foi qualitativa e com o foco na facilidade do uso e na percepção do usuário de se as necessidades estavam sendo atendidas com aquelas funcionalidades.

3.3.3.3 APRENDIZADO

Ao final do desenvolvimento, entendemos o que é prioridade para o cliente e aprendemos que a solução proposta atende as expectativas do cliente/usuário.

3.3.3.4 ARTEFATO

Como entrega desse passo, temos a aplicação funcional do MVP. Define-se como MVP pois foi testada localmente, sem a utilização real em produção.

3.3.4 PASSO 4 – AVALIAÇÃO

Este passo teve por objetivo averiguar o artefato final desenvolvido (i.e. desenvolvimento do MVP) frente a um conjunto de medidas de desempenho e as necessidades encontradas. Tais medidas e necessidades são diretamente relacionadas aos objetivos geral, específicos e proposição de valor do trabalho.

Este passo foi realizado no final de cada *sprint* de desenvolvimento, conforme o Passo 3 realizando encontros com o primeiro cliente da solução e avaliar qualitativamente o quanto a ferramenta desenvolvida atende suas necessidades.

Como o objetivo da solução é resolver problemas de negócio de clientes reais, a avaliação qualitativa do primeiro cliente pagante foi considerada suficiente para avaliar o sucesso da prova de conceito.

3.3.5 PASSO 5 – CONCLUSÃO

Esta última etapa do método *Design Science* visou sintetizar e fazer uma avaliação global dos resultados obtidos, juntamente com várias observações e reflexões deste autor sobre a pesquisa em si e sobre o desenvolvimento propriamente dito.

Por fim, são descritas algumas sugestões de trabalhos futuros, de continuação e melhorias no método.

4 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo o autor descreve quais os conceitos básicos necessários para contextualizar o leitor. Nesse sentido, descreve-se a respeito da economia da recorrência e como isso influenciou no aumento do poder do cliente na decisão de compra e desligamento, aumentando a importância do pós-venda.

Também são descritos aqui os conceitos relacionados a gestão do relacionamento com o cliente, a estratégia do sucesso do cliente, uma contextualização de como é o mercado brasileiro em termos dessa estratégia e quais são os principais resultados esperados com sua aplicação.

4.1 ECONOMIA DA RECORRÊNCIA

O modelo de recorrência já é uma tendência mundial (ZUORA, 2017). Tendo como principais benefícios um menor custo para aquisição de clientes e consequentemente uma maior escalabilidade. Esse novo modelo junto ao armazenamento na nuvem, possibilitou que a indústria de software pudesse oferecer soluções a seus clientes que dispensam a aquisição e manutenção de infraestrutura, a contratação de pessoas e principalmente o alto valor inicial que envolve a compra de uma licença.

Os fornecedores de Software As A Service (SaaS) (como ficou conhecido o modelo de recorrência de software hospedado na nuvem) cobram mensalmente pela utilização do serviço ao invés de cobrar pela propriedade do software. Assim, mês a mês seus clientes decidem por uma mensalidade para poderem utilizar o produto via aplicação web, sem a necessidade de infraestrutura interna, muito menos um custo inicial de implantação e treinamento.

Nesse modelo de negócio, por cobrarem mensalidades de seus clientes, as empresas encontram uma barreira menor de entrada para contratação, tanto pelos custos iniciais, quanto pelos custos a longo prazo de manter um software (database, pessoas, manutenção, etc).

Também, por utilizarem o armazenamento na nuvem, as soluções são acessadas via o browser de internet, sem requerer instalações, diminuindo o tempo de implantação e facilitando o acesso.

Tudo isso fez com que o modelo SaaS ganhasse força e hoje representa no mundo um mercado de 55 bilhões de dólares (STATISTA, 2018), sendo utilizado em praticamente todos os setores, desde a indústria, varejo físico, e-commerce, saúde, transporte, entretenimento, entre outros. Segundo a ABES (Associação Brasileira de Software) no Brasil é mais de 746 milhões de dólares, com um crescimento de 2015 para 2016 de 47,4% (ABES - ASSOCIAÇÃO BRASILEIRA DE SOFTWARE, 2017).

4.2 GESTÃO DO RELACIONAMENTO COM O CLIENTE

A gestão do relacionamento é o foco principal da solução e por isso é importante ser contextualizada nesse documento. Assim, o autor descreve os conceitos, o contexto do sucesso do cliente dentro da literatura de gestão do relacionamento com o cliente e também contextualiza o leitor em relação a como esses conceitos são aplicados pelas empresas brasileiras.

4.2.1 CONCEITO

Segundo Buttle (BUTTLE, 2004), a gestão relacionamento com o cliente (CRM, do inglês) pode ser vista em 3 diferentes níveis, como na Figura 8 abaixo:

Level of CRM	Dominant characteristic
Strategic	A top–down perspective on CRM which views CRM as a core customer-centric business strategy that aims at winning and keeping profitable customers
Operational	A perspective on CRM which focuses on major automation projects such as service automation, sales force automation or marketing automation
Analytical	A bottom–up perspective on CRM which focuses on the intelligent mining of customer data for strategic or tactical purposes

Figura 8 - Diferentes leveis de gestão do relacionamento com o cliente (BUTTLE, 2004).

A CRM estratégica, é focada no desenvolvimento de uma cultura de negócios focada no cliente (*customer-centric*), que é dedicada a “ganhar e manter” cliente, por meio de uma entrega de valor superior do que os competidores. Como cultura, tem seu reflexo no comportamento das lideranças, no design dos sistemas da empresa e no investimento sendo colocado onde se potencializa o valor do cliente.

Já a CRM operacional, é focada na automação das interfaces que se tem com o negócio. É nesse nível que o nome CRM se popularizou, pois é onde os softwares mais geram valor, possibilitando automatizar o marketing, a venda e até o serviço ao cliente.

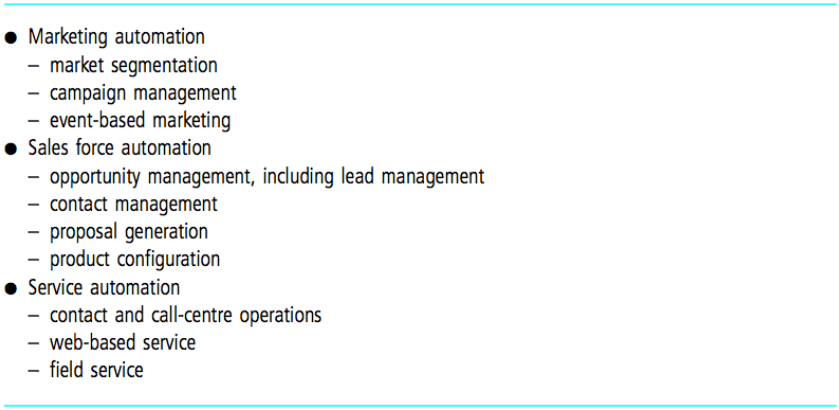

- Marketing automation
  - market segmentation
  - campaign management
  - event-based marketing
- Sales force automation
  - opportunity management, including lead management
  - contact management
  - proposal generation
  - product configuration
- Service automation
  - contact and call-centre operations
  - web-based service
  - field service

Figura 9 - Aplicações gerais de um CRM operacional (BUTTLE, 2004).

Já a gestão do relacionamento com o cliente analítica explora a utilização de dados do cliente para aprimorar tanto o valor para o cliente, quanto para a empresa. Por definição, seu foco é em armazenar e consumir informações sobre o cliente, que podem ser: histórico de compras, histórico de pagamentos, crédito disponível, resposta a campanhas, dados de fidelidade, entre outros.

Ou seja, CRM, segundo Buttle: é o centro de uma estratégia de negócios que integra processos internos, funções, redes externas para criar e entregar valor para um cliente alvo visando lucro. É baseado em uma boa qualidade de dados dos clientes e é possibilitado por Tecnologia da Informação (BUTTLE, 2004).

4.2.2 JORNADA DO CLIENTE E PÓS-VENDA

Segundo Buttle, um relacionamento é composto por uma série de episódios entre as partes ao longo de um tempo determinado. Cada episódio é composto por uma série de interações. Esses episódios tem um começo e um fim e também podem ser nomeados. O relacionamento muda com o tempo, evoluem, e podem ser divididos em 5 principais fases: consciência, consideração, avaliação, compra e lealdade (DWYER, 1987), variando de fase dependendo do foco ou da estratégia de marketing em questão.

Figura 10 - Diferentes estágios de um relacionamento com o cliente (ENDEAVOR, 2017).

Segundo o estudo da McKinsey (MCKINSEY, 2009), cerca de 60% das interações que influenciam a compra estão fora de seu controle e estão diretamente relacionadas a indicação de outros clientes. Isso demonstra a influência da satisfação do cliente e da utilização de técnicas de pós-venda como potencial de crescimento, tanto em clientes atuais, quanto para novos clientes.

4.2.3 SUCESSO DO CLIENTE

Com o aumento no número de empresas utilizando o modelo SaaS e seu modelo de aquisição de clientes agressivo, houve um grande crescimento de novos clientes, porém, com a complexidade dos produtos e seu difícil entendimento, muitos deles tornarem-se frustrados, diminuindo as taxas de adoção e uso desses produtos o que ao longo do tempo gerou um grande número de desligamentos (*churns*) (MCKINSEY, 2018).

Em resposta, negócios SaaS criaram estratégias e times para proativamente interagir com contas-alvo em risco, feitos com o foco na retenção e em ajudar os clientes a retirarem mais valor dos produtos, os times sucesso do cliente. Seguindo essas iniciativas iniciais, muitos negócios começaram a criar funções formais de sucesso do cliente. Esses esforços tornaram essa estratégia uma disciplina emergente na indústria do software, completa com ferramentas e metodologias.

Segundo estudo da McKinsey, estamos vivendo uma nova era, que não só as empresas estão focando no sucesso do cliente como ferramenta de retenção de clientes, como também estão a utilizando como estratégia de crescimento, já que os gerentes de sucesso do cliente têm um conhecimento profundo sobre o cliente e conseguem encontrar oportunidades e oferecer soluções que aumentem o valor do cliente (MCKINSEY, 2018).

Também segundo esse estudo, essa estratégia não mais se limita a empresa SaaS de tecnologia, mas também a vários segmentos de serviço *Business-to-Business* (B2B) que investem nessa metodologia como estratégia de crescimento e que demonstram custos de expansão de contas com uma fração do custo de aquisição de novos clientes.

Segundo Lincoln Murphy (LINCOLN MURPHY, 2017), o sucesso do cliente nada mais é que quando ele atinge seu sucesso desejado por meio de interações com a empresa. Sendo sucesso desejado composto por uma razão de compra (sucesso requerido) e por uma experiência apropriada (diferencial da solução contratada).

4.2.3.1 GESTÃO DO SUCESSO DO CLIENTE

Já a gestão do sucesso do cliente é a orquestração proativa para que os diferentes segmentos de clientes de sua base atinjam seu sucesso de maneira processual, previsível e quando possível, escalável (LINCOLN MURPHY, 2017).

Para que isso aconteça, é necessário:

- Segmentar a base de clientes
- Definir o sucesso do cliente para cada segmento
- Implementar a gestão do sucesso do cliente
- Medir e aprimorar o processo

Um exemplo é a jornada estipulada pela ferramenta Pipz (Figura 11), que foca em técnicas de pós-vendas automatizadas que não necessariamente envolve a atuação de uma pessoa.

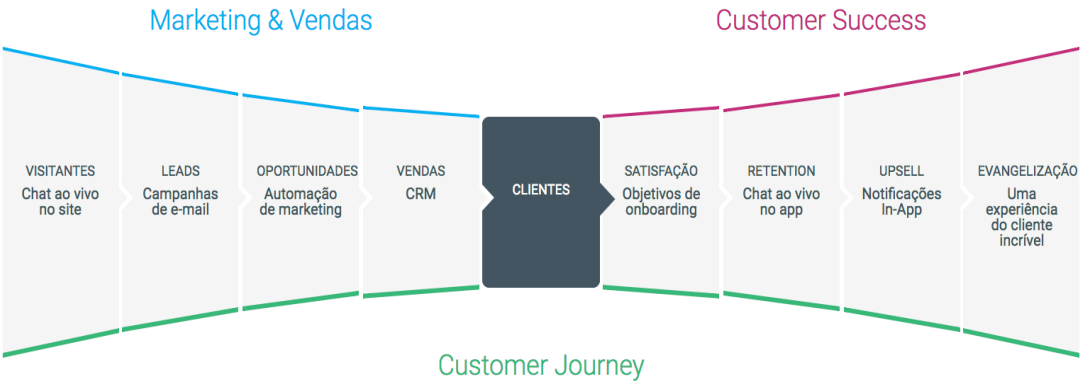


Figura 11 - Jornada do cliente segundo a empresa Pipz (Fonte: Pipz).

4.2.4 O PROCESSO HOJE

Segundo o Panorama do Customer Success (MINDMINERS, 2018), pesquisa realizada com mais de 300 profissionais da área no Brasil, hoje no Brasil 86% das empresas que não fazem sucesso do cliente (Customer Success – CS) tem o desejo de criar a área ainda em 2018. Além disso, segundo o documento, O grande desafio das empresas para 2018 é criar os processos de CS e instituir uma cultura de *Customer Centric* em todas as áreas da empresa.

De acordo com as mais de 80 entrevistas feitas com empresas do segmento de tecnologia e serviço B2B feitas durante o desenvolvimento dessa prova de conceito, as empresas que possuem times de relacionamento com o cliente ou até mesmo times de sucesso do cliente, utilizam 4 alternativas para fazer a gestão do sucesso do cliente ou a gestão das atividades desse time, são elas: ferramentas de gestão do relacionamento com o cliente (CRM), planilhas, ferramentas de automação de e-mails ou soluções feitas especificamente para o sucesso do cliente e que são chamadas: ferramentas de gestão do sucesso do cliente (CSM).

Porém, também pelas entrevistas realizadas, identificou-se algumas limitações dessas soluções na visão do usuário, como a linguagem, funcionalidades específicas e as métricas utilizadas:

Planilhas: utilizam-se para anotar diferentes aspectos da experiência do cliente, como por exemplo, em que etapa esse cliente está, como o desenvolvimento desse cliente ao longo das etapas, como ele se encontra frente ao planejado e um controle de que atividades que estão sendo feitas pelo time com cada cliente.

  - Limitação: conforme o processo ganha complexidade, aumentando o número de atividades, de pessoas na equipe e o número de clientes, se torna custoso alimentar todas as informações e aumentam as chances de erro, tudo dado a estrutura de uma planilha e suas integrações.

- CRM: Essas ferramentas são geralmente focadas em vendas e seu pipeline. As empresas o utilizam para gerenciar as etapas e principalmente armazenar as interações com o cliente.
  - Limitação: não utilizam a mesma linguagem necessária para o pós-venda, muito menos conseguem medir informações que são essenciais em relação ao cliente, como NPS e volume de interações. Ainda, a maioria dos CRMs não consegue automatizar atividades. Do contrário, caso consigam, são soluções completas e caras.
- Automação de e-mails: Existem ferramentas focadas em automação de e-mail, ou o chamado e-mail marketing que atendem algumas equipes de sucesso do cliente.
  - Limitação: suas métricas são de marketing e seus resultados não são integrados, gastando-se tempo para encontrar as métricas certas e isolar a situação de cada cliente.

- CSM: são soluções que integram diferentes áreas da empresa e criando uma única métrica que representa a saúde do cliente. Essas soluções podem integrar: suporte técnico, pagamentos, uso em produto, marketing, atendimento, vendas, entre outros.
    - Limitação: são focadas exclusivamente em empresas com modelo SaaS, são caras e requerem meses para implantação e operação.

4.2.5 RESULTADOS ESPERADOS

Todos esses esforços em entregar o resultado desejado pelo cliente tem como objetivo aumentar sua satisfação com o serviço contratado e sistematicamente utilizar isso a favor da:

- Indicação do serviço
- Do aumento dos contratos
- Diminuição de cancelamentos

E ainda, que quando esses clientes satisfeitos indiquem, os novos clientes tenham um custo de aquisição menor e uma maior facilidade na venda.

5 PROJETO DO SISTEMA

O projeto dessa prova de conceito foi realizado por meio da junção de um conjunto de artefatos resultantes dos procedimentos da metodologia adotada, descritos no capítulo 3. Em termos de engenharia de software, essa junção corresponde a alguns artefatos de projeto de software descritos em linguagem UML e de acordo com o autor Wazlawick (WAZLAWICK, 2014).

5.1 REQUISITOS FUNCIONAIS

De acordo com Wazlawick (WAZLAWICK, 2014), os requisitos funcionais correspondem à listagem de tudo que o sistema deve fazer, e podem ser ainda classificados em dois grupos. Os requisitos funcionais evidentes, que são efetuados com o conhecimento do usuário. Esses requisitos usualmente corresponderão a eventos do sistema e respostas do sistema. Já os requisitos funcionais ocultos que são efetuados sem o conhecimento explicito do usuário.

Como esse projeto é uma prova de conceito e se apresenta na forma de um *Minimum Valuable Product* (MVP) de acordo com o *Lean Startup*, os requisitos foram simplificados ao máximo e aos que são realmente necessários.

De acordo com o autor, existem diversas formas de descrever esses requisitos, a proposta por ele é a utilizada nesse trabalho:

RF01 - Permitirá ao usuário visualizar, adicionar, editar e remover etapas

Tipo: Evidente

Descrição: Permitirá ao usuário adicionar, editar e remover etapas relacionadas ao processo. A etapa terá uma data de inicio, uma data de fim, um *status* e um título.

RF02 - Permitirá ao usuário visualizar, adicionar, editar e remover atividades

Tipo: Evidente

Descrição: Permitirá ao usuário adicionar, editar e remover a atividades. A essas atividades serão atribuídas datas, responsáveis e a que cliente pertencem. Elas

também devem ter: um título, uma descrição, um status uma data para ser completada e a data completada.

RF03 - Permitirá ao usuário visualizar, adicionar, editar e remover clientes

Tipo: Evidente

Descrição: Permitirá ao usuário visualizar todos os clientes que estão em sua base, com informações básicas como nome, segmento de mercado e contato do responsável pela empresa.

RF04 - Permitirá ao usuário visualizar, adicionar, editar e remover interações com o cliente

Tipo: Evidente

Descrição: Permitirá ao usuário adicionar, editar e remover interações com o cliente, que são anotações feitas pelo usuário, atribuídas a um cliente e que possuem um indicativo de “humor do cliente” para verificar a qualidade da interação; Essa funcionalidade também permitirá a inclusão de resultados de uma pesquisa de *Net Promoter Score* (NPS) para armazenar essa informação.

RF05 - Permitirá ao usuário visualizar o número de tarefas e interações realizadas com cada cliente

Tipo: Evidente

Descrição: Permitirá ao usuário visualizar diferentes métricas da aplicação, relacionadas as atividades e as interações, organizadas por usuário.

RF06 - Permitirá ao usuário visualizar o histórico de notas do NPS feitas

Tipo: Evidente

Descrição: Permitirá ao usuário visualizar um número específico de pesquisas anteriores para poder comparar sua variação ao longo do tempo.

RF07 - Permitirá ao usuário visualizar o tempo que o cliente está em cada etapa

Tipo: Evidente

Descrição: Permitirá ao usuário visualizar quanto tempo o cliente ficou em cada etapa e assim entender quais etapas estão durando mais ou menos do que o planejado, para então tomar planos de ação.

5.2 CASOS DE USO

Também de acordo com Wazlawick, é necessário organizar os requisitos em grupos correlacionados, de forma a ajudar os ciclos iterativos de desenvolvimento. Os requisitos podem ser agrupados em: casos de uso, conceitos e consultas.

Os casos de uso é a maneira utilizada para agrupar os requisitos desse projeto e devem representar os processos de negócio dentro da prova de conceito (WAZLAWICK, 2014). Como essa ferramenta é focada para o sucesso do cliente, o ator da prova de conceito é basicamente o Gerente do Sucesso do Cliente (*Customer Success Manage*r – CSM).

Tabela 2 - Casos de uso do MVP.

Nome	Ator	Descrição	Referências Cruzadas
Gerenciar a jornada do cliente	CSM	O usuário escolhe um cliente, analisa a situação de suas etapas e atividades (jornada) no processo e toma ações de relacionamento com base nessas informações.	RF01, RF02, RF03, RF07
Gerenciar estado das interações com o cliente	CSM	O usuário escolhe um cliente, analisa todas as interações executadas com esse cliente e em que	RF03, RF04, RF06

		qualidade, tomando ações de relacionamento com base nessas informações.	
Gerenciar o desempenho da carteira de clientes	CSM	Usuário visualiza o estado atual de todos os clientes sobre sua responsabilidade.	RF01, RF02, RF03, RF04, RF05, RF06, RF07

É possível representar os casos de uso e seus atores utilizando o diagrama de caso de uso da linguagem UML (WAZLAWICK, 2014). Assim, temos o seguinte diagrama na Figura 12:

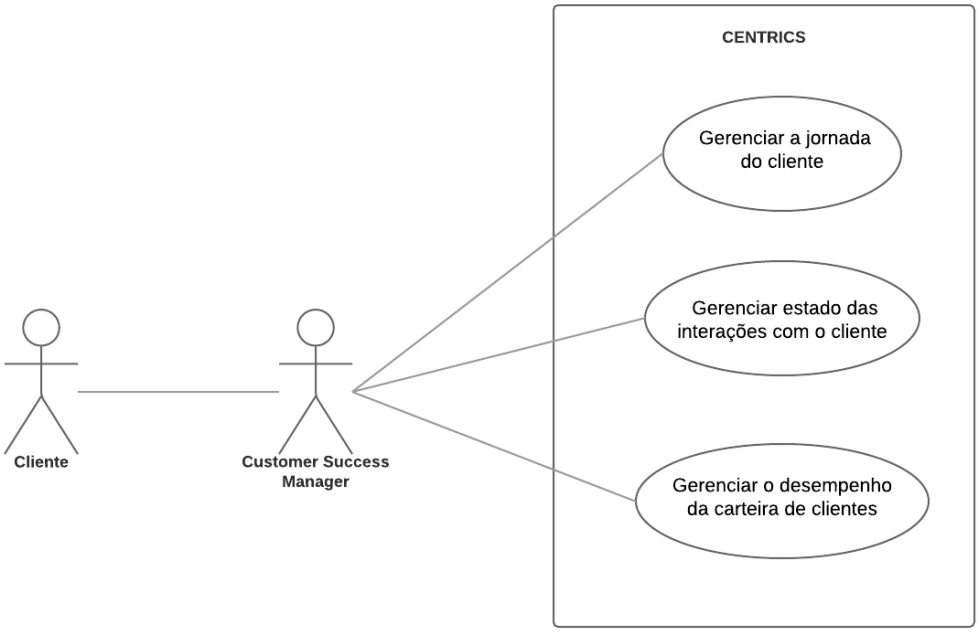


Figura 12 - Representação UML dos casos de uso.

5.3 DIAGRAMA DE SEQUÊNCIA

A linguagem UML também permite representar as operações e consultas de um sistema de acordo com uma sequência de eventos em um determinado caso de uso,

representando qual o fluxo de dados e informações entre atores, aplicação e o controlador (WAZLAWICK, 2014).

Assim, para cada caso de uso temos um diagrama de sequência, assim como na Figura 13.

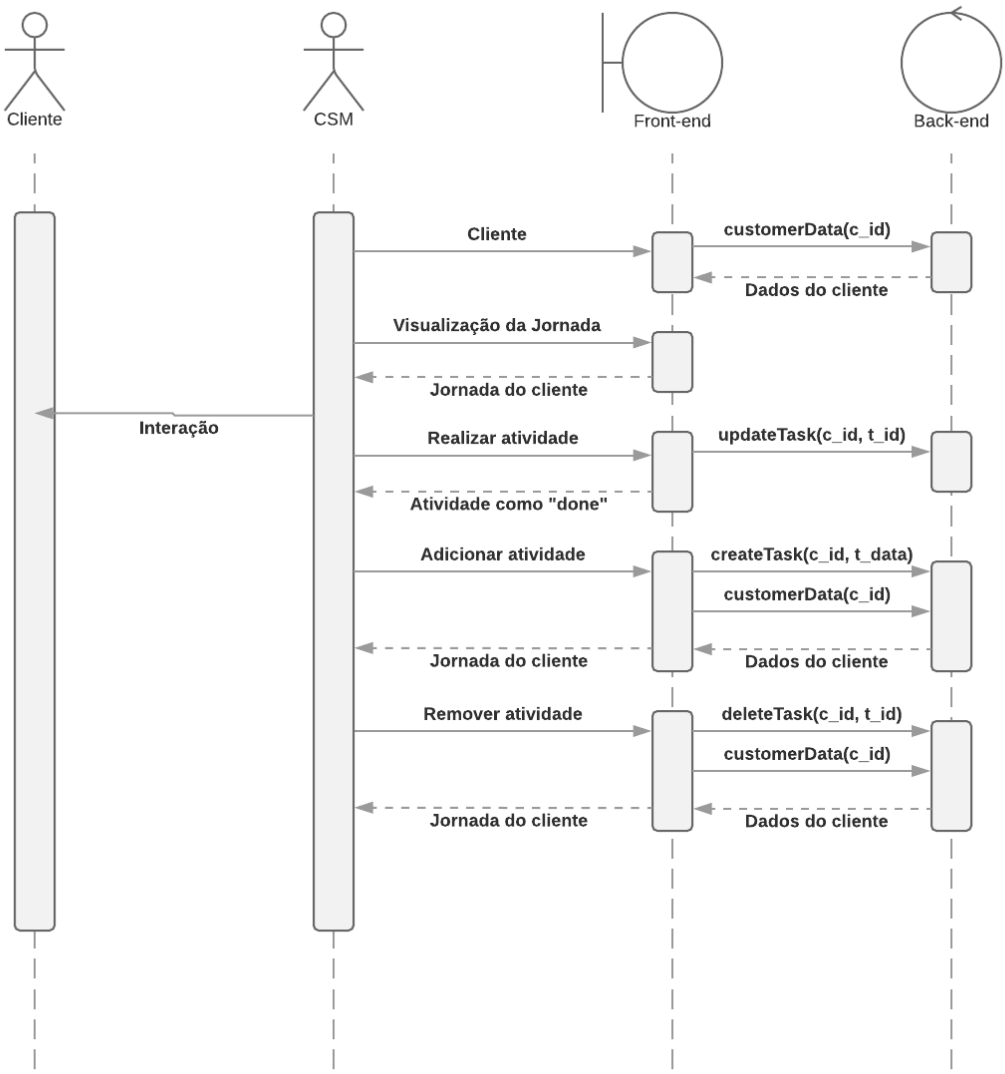


Figura 13 - Diagrama de sequência UML: “Gerenciar a jornada do cliente”.

Esse diagrama representa o usuário do sistema CSM interagindo com a aplicação (o *front-end*) e como o *front-end* deveria interagir com o *back-end* no caso de uso “Gerenciar a jornada do cliente”. Ele demonstra que o CSM deve selecionar o cliente e abrir a visualização da jornada para que o *front-end* entrega a jornada desse cliente em específico. Com a visualização da jornada do cliente, o CSM pode realizar, adicionar e remover tarefas no sistema e também interagir com o cliente.

Já a Figura 14, demonstra o diagrama de sequência do caso de uso “Gerenciar estado das interações com o cliente”, e, que dessa forma, o CSM deve selecionar o cliente, selecionar a visualização de interações para poder visualizar e analisar o histórico de interações que já foram feitas com o cliente. A partir dai ele pode fazer novas interações ou remover interações antigas.

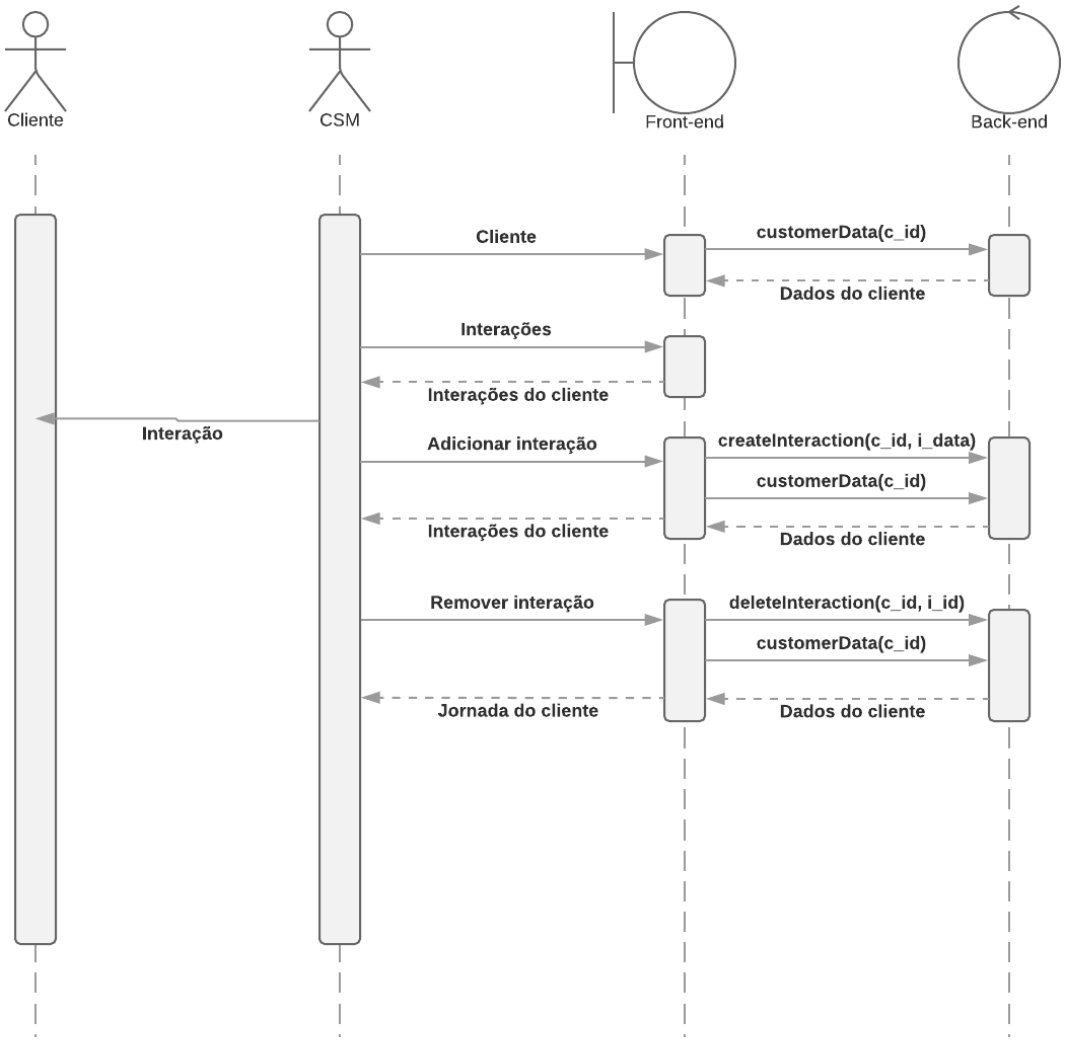


Figura 14 - Diagrama de sequência UML: “Gerenciar o desempenho da carteira de clientes”.

Já o diagrama da Figura 15, é a respeito do terceiro caso de uso, o “Gerenciar o desempenho da carteira de clientes”. Esse caso de uso é a respeito da visualização de informações acumuladas de diferentes clientes dentro da prova de conceito, nesse sentido, não há interações com o cliente e só depende do carregamento de informações para visualização do usuário.

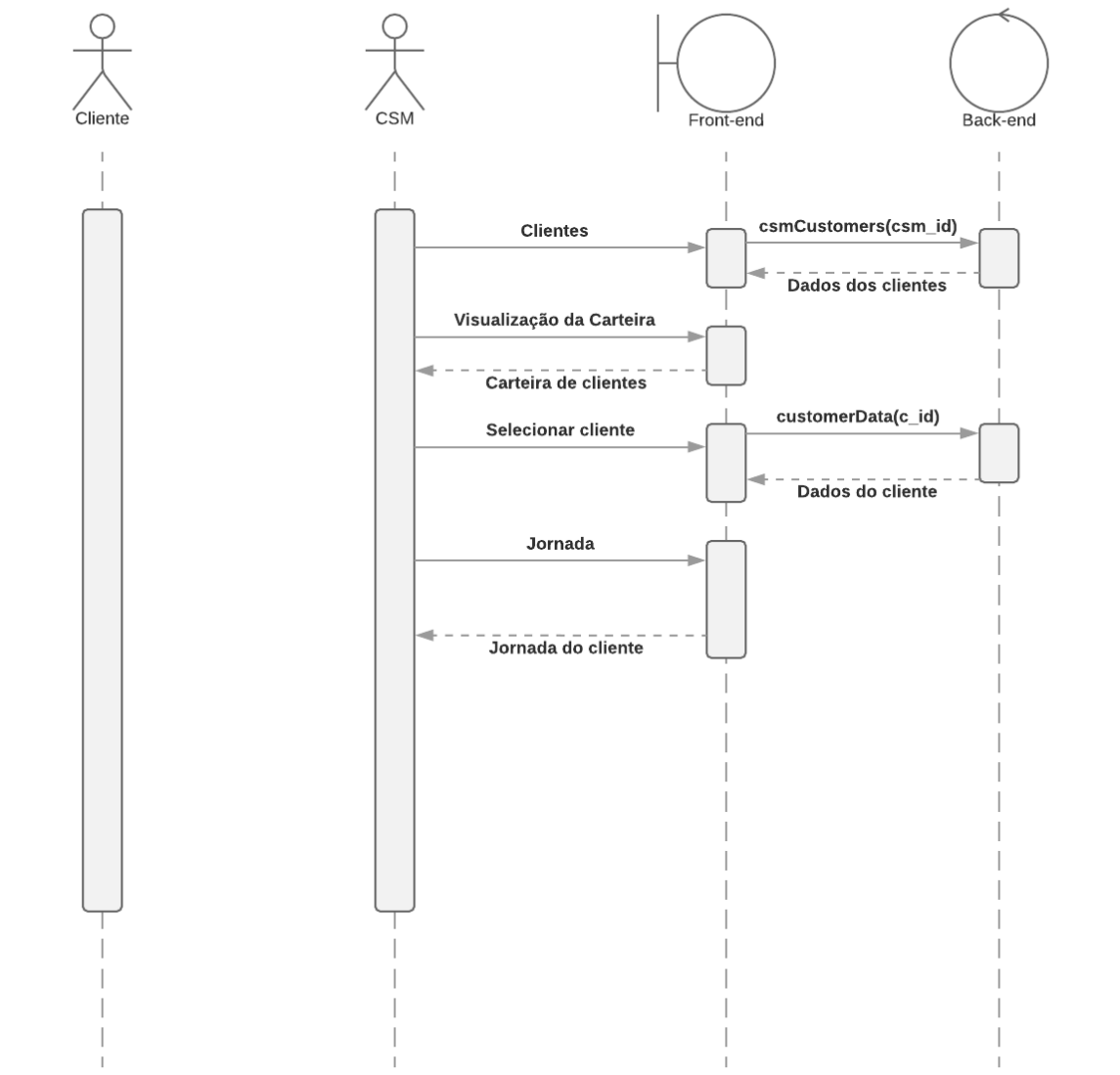


Figura 15 - Diagrama de sequência UML: "Gerenciar o desempenho da carteira de clientes".

5.4 DIAGRAMA DE CLASSES DE PROJETO

Agora, com os requisitos, casos de uso e o diagrama de sequência, é necessário a organização do projeto em termos da camada de domínio (WAZLAWICK, 2014). Aqui vamos adicionar métodos as classes e os sentidos das relações. Assim, dado os artefatos construídos até aqui, temos a seguinte representação do diagrama de classes desse projeto (Figura 16):

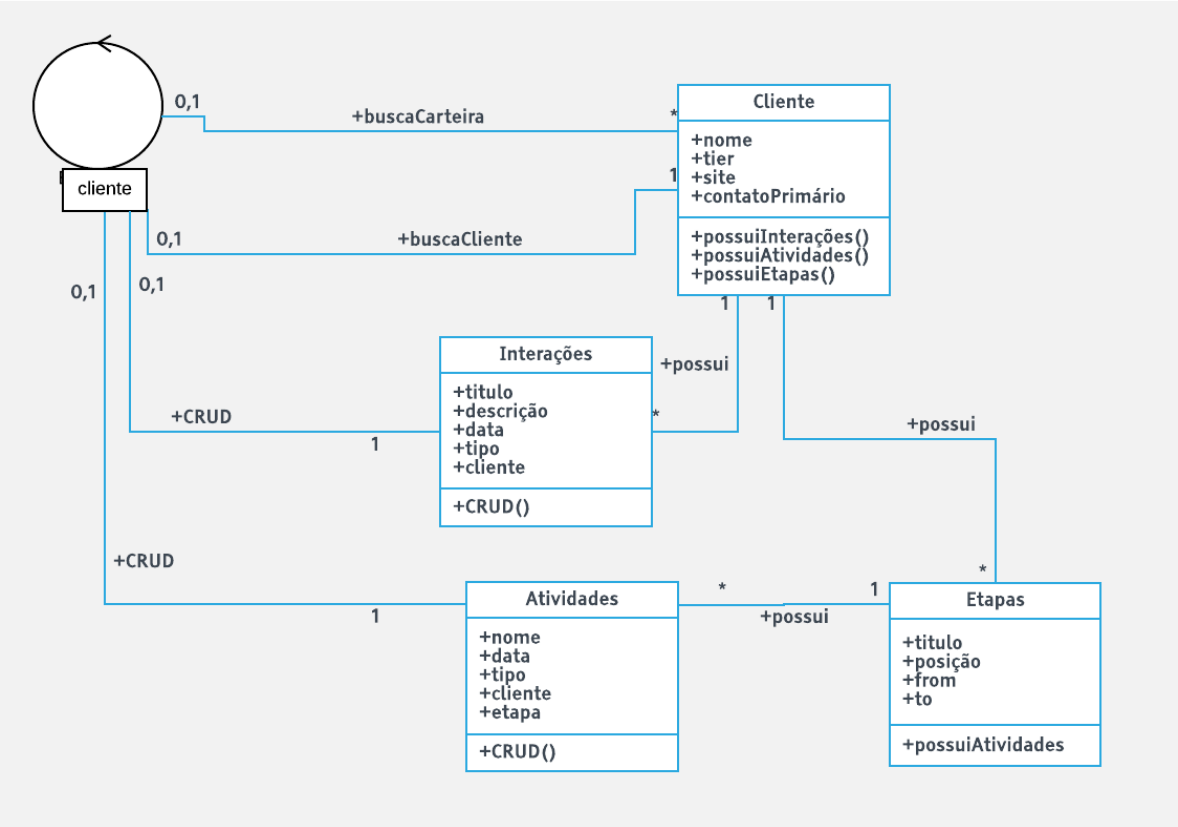


Figura 16 - Diagrama de classes de projeto UML.

Como podemos verificar no diagrama de classes acima, temos um controlador, que é o *back-end*, responsável por chamar métodos de criação, leitura, atualização e remoção (CRUD), tanto de interações quanto de atividades. Além disso, podemos verificar que a classe Cliente possui as demais classes, e que o *back-end* possui métodos tanto para buscar um cliente quanto uma carteira (vários clientes).

5.5 ARQUITETURA DE DESENVOLVIMENTO

Além de classes e métodos, um projeto de software necessita de uma arquitetura que represente como será de fato a solução do problema enunciado (WAZLAWICK, 2014). Dada as diferentes opções, o aluno optou pela arquitetura *Model-View-Controller* por ter tecnologias de desenvolvimento consolidadas a partir dessa arquitetura, fazendo com que o aluno tivesse maior produtividade (POPE, 1998).

Essa arquitetura possibilita a divisão da aplicação em três partes interconectadas. Isto é feito para separar as representações de informações internas dos modos como a informação é apresentada e aceita pelo usuário. Assim, facilita a reutilização de código e o desenvolvimento paralelo. Além de especializar a aplicação para os diferentes objetivos (POPE, 1998).

O *model* consiste nos dados e na camada de persistência da aplicação, aqui são descritas as regras de negócio, a lógica e as funções. A *view*, é a saída e representação desses dados, que podem ser tabelas, diagramas, formulários. Já o *controller* é quem faz a mediação da entrada, convertendo-a em chamados para o *model* ou o *view* (POPE, 1998).

O aumento da complexidade das aplicações faz com que a separação entre dados e apresentação se torne mais relevante, já que assim, alterações no layout não afetam a manipulação dos dados, e a manipulação/modificação dos dados também não interfere na apresentação (POPE, 1998).

Seguindo esse modelo, o aluno optou pela seguinte representação na Figura 17:

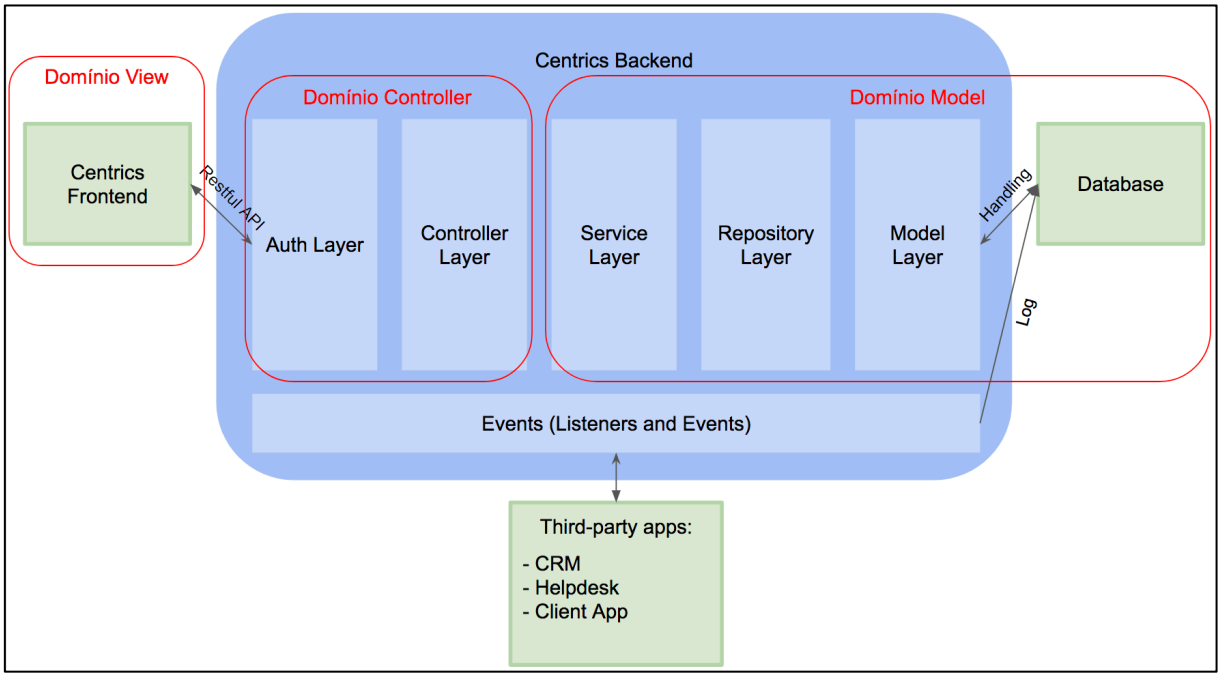


Figura 17 - Arquitetura do sistema.

Com a Figura 17, podemos verificar mais camadas do que se propõe inicialmente pelo MVC, essa escolha é tomada para separar ainda mais as funcionalidades de cada *layer*. Nessa figura, o Centrics *Frontend* corresponde ao domínio *View* e é a apresentação do sistema. A *Auth Layer* e *Controller Layer* representam o domínio do *Controller*, já a *Service Layer*, *Repository Layer*, *Model Layer* correspondem ao domínio *Model* do MVC. Além dessas camadas é adicionada uma camada de eventos, que será responsável por gerar eventos e *listeners* desses eventos, com o objetivo de criação de um *log* de utilização e interação com funcionalidades externas a aplicação em si.

Basicamente, o domínio Controller pode ser visto como as linhas que saem do cliente na Figura 16, como o motodo buscarCarteira, buscarCliente e os CRUDs, o domínio Model pode ser visto nesse mesma figura pelas relações estabelecidas entre as classes. O domínio View é a apresentação ao usuário e por isso não é representado no diagrama de classes.

5.6 DEPLOYMENT

Dentro da arquitetura especificada, optou-se para um modelo de deployment em que pudéssemos ter um custo baixo e também uma flexibilidade na infraestrutura necessária, por isso, optou-se por utilizar os fornecedores de armazenamento na nuvem, sendo que o Frontend e o Backend estarão utilizando serviços diferentes, já que é possível baratear o custo de infraestrutura, sendo que o Frontend é considerado estático não sendo feita nenhuma requisição ou persistência dos dados. A tecnologia e o custo do Backend se diferencia justamente por depender dessa persistência e de diversas requisições ao longo do tempo.

6 DESENVOLVIMENTO

De acordo com o projeto, foi escolhido uma série de tecnologias que facilitassem o desenvolvimento e principalmente aumentasse a produtividade do aluno em desenvolver esse sistema. Dessa forma, cada camada foi implementada de uma forma, usando principalmente a linguagem JavaScript para o desenvolvimento do front-end e PHP para o desenvolvimento do *back-end*.

Os códigos-fonte colocados nesse documento são uma versão resumida do caso real, já que por se tratar de uma grande quantidade de linhas poderia atrapalhar a leitura do documento.

As motivações da escolha dessas tecnologias são apresentadas caso a caso.

6.1 MODEL

A implementação dessa camada foi realizada utilizando o framework Laravel PHP, criado para simplificar o desenvolvimento web utilizando as melhores características da arquitetura MVC e assim, facilitando a organização e reaproveitamento de código. Essa tecnologia foi escolhida pelo aluno por já ter tido algumas experiências utilizando-a, tornando a curva de aprendizagem menor.

Como está descrito em maiores detalhes no capítulo do projeto do sistema, o domínio *Model* do MVC nesse projeto é composto por diferentes camadas, como: *Service Layer*, *Repository Layer* e *Model Layer*, assim, temos a implementação de cada um desses *layers* para cada classe, como por exemplo uma atividade.

O *Model Layer* é onde definimos as relações entre as diferentes entidades. Como no exemplo podemos pegar o do modelo Atividade:

```php
<?php

namespace App\Models;

use Carbon\Carbon;

class Task extends BaseModel
{
    use Accountable;

    protected $fillable = [
        'id', 'user_id', 'customer_id', 'journey_id', 'assignee',
'title', 'description', 'status', 'due_date','completed_at',
    ];

    public function customer()
    {
        return $this->belongsTo(Customer::class);
    }

    public function journey()
    {
        return $this->belongsTo(Journey::class);
    }

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function getDelayedAttribute()
    {
        return $this->status != 'done' && Carbon::now()-
>greaterThan($this->due_date);
    }
}
```

Figura 18 - Exemplo de um Model Layer: Atividade.

Já o *Repository Layer* é composto pelas interações que são feitas com o banco de dados, ou seja, com a extratação e escritas de dados em si (*queries*), dessa forma criou-se funções pré-definidas para suprir os requisitos dos casos de uso como criar, deletar, atualizar e ler. Podemos verificar no exemplo do *Repository Layer* das atividades:

```php
<?php

use Illuminate\Database\Eloquent\Model;

abstract class TaskRepository implements BaseRepositoryInterface
{
    public function all()
    {
        return $this->model::all();
    }

    public function find($id)
    {
        if (!$model = $this->model::find($id))
            return false;

        return $model;
    }

    public function create(array $data = [])
    {
        $model = new $this->model();
        foreach ($model->getFillable() as $column) {
            if ($column != $model->getKeyName())
                $model->$column = $data[$column] ?? null;
        }
        $model->save();
        return $model;
    }

    public function update(string $id, array $data = [])
    {
        $model = $this->find($id);
        foreach ($model->getFillable() as $column) {
            $model->$column = $data[$column] ?? $model->$column;
        }
        $model->save();
        return $model;
    }

    public function delete(string $id)
    {
        $model = $this->find($id);
        $this->model::destroy($id);
        return $model;
    }
}
```

Figura 19 - Exemplo de um Repository Layer: Atividade.

Além disso, por último, temos o *Service Layer*, responsável por chamar diferentes funções ligadas ao repositório, assim como os eventos específicos de cada classe

para a criação do *log*. Podemos verificar a construção de um *Service Layer* como no exemplo da classe atividade:

```php
<?php
abstract class TaskService implements BaseServiceInterface
{
    protected $repo;

    protected $events = [
        'created' => null,
        'deleted' => null,
        'updated' => null,
    ];
    public function all()
    {
        return $this->repo->all();
    }
    public function find(string $id)
    {
        return $this->repo->find($id);
    }
    public function create(array $data = [])
    {
        $new = $this->repo->create($data);
        if (!is_null($this->events['created']))
            event(new $this->events['created']($new));
        return $new;
    }
    public function update(string $id, array $data = [])
    {
        $old = $this->repo->find($id);
        $new = $this->repo->update($id, $data);

        if (!is_null($this->events['updated']))
            event(new $this->events['updated']($new, $old));

        return $new;
    }
    public function delete(string $id)
    {
        $old = $this->repo->find($id);
        $this->repo->delete($id);

        if (!is_null($this->events['deleted']))
            event(new $this->events['deleted']($old));
        return $old;
    }
}
```

Figura 20 - Exemplo de Service Layer: Atividade.

Por fim, essas camadas se interligam ao banco de dados por meio de *migrations*. As *migrations* criam tabelas no banco de dados escolhido utilizando como

base as relações do *Model Layer* e as *queries* do *Repository Layer*. Assim como os *layers* isolam os atributos e relações entre as classes de retrabalho com a troca de tecnologia (Por exemplo uma troca de banco de dados), a *migration* garante que as tabelas sejam criadas independentemente da escolha da tecnologia.

Podemos verificar tal caso na *migration* da classe Atividade:

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateTasksTable extends Migration
{
    public function up()
    {
        Schema::create('tasks', function (Blueprint $table) {
            $table->timestamps();
            $table->string('id', 30);
            $table->primary('id');
            $table->string('account_id', 30)->nullable();
            $table->foreign('account_id')->references('id')-
>on('accounts');
            $table->string('user_id', 30)->nullable();
            $table->foreign('user_id')->references('id')->on('users');
            $table->string('customer_id', 30)->nullable();
            $table->foreign('customer_id')->references('id')-
>on('customers');
            $table->string('journey_id', 30)->nullable();
            $table->foreign('journey_id')->references('id')-
>on('journeys');
            $table->enum('assignee', ['user', 'automatic',
'customer']);
            $table->string('title');
            $table->text('description')->nullable();
            $table->enum('status', ['todo', 'done']);
            $table->timestamp('due_date')->nullable();
            $table->timestamp('completed_at')->nullable();
        });
    }
    public function down()
    {
        Schema::dropIfExists('tasks');
    }
}
```

Figura 21 - Exemplo de migration: Atividade.

Para fim de teste do sistema, foi escolhido o MySql Lite como tecnologia do banco de dados, a decisão foi tomada com base na facilidade de instalação e

configuração em comparação com pacotes mais completos, além de ser um banco de dados que o aluno teve experiências passadas. Para declarar a utilização dessa tecnologia basta configurar o ambiente do Laravel, escrevendo que a tecnologia utilizada será essa, criando também um arquivo intitulado “database.sqlite” na pasta do projeto.

Para criar dados no banco de dados e testar com maior comodidade o funcionamento da solução, o Laravel possibilidade a criação de “*database seeders*” que criam dados falsos e possibilitam testar as diferentes funcionalidades do sistema. O *seeder* cria dados randômicos de acordo com o programado. O código utilizado para testar o sistema está descrito no Anexo A.

Utilizando a ferramenta DB Browser, é possível verificar a criação das tabelas e dos dados de acordo com o esperado, como na Figura 22:

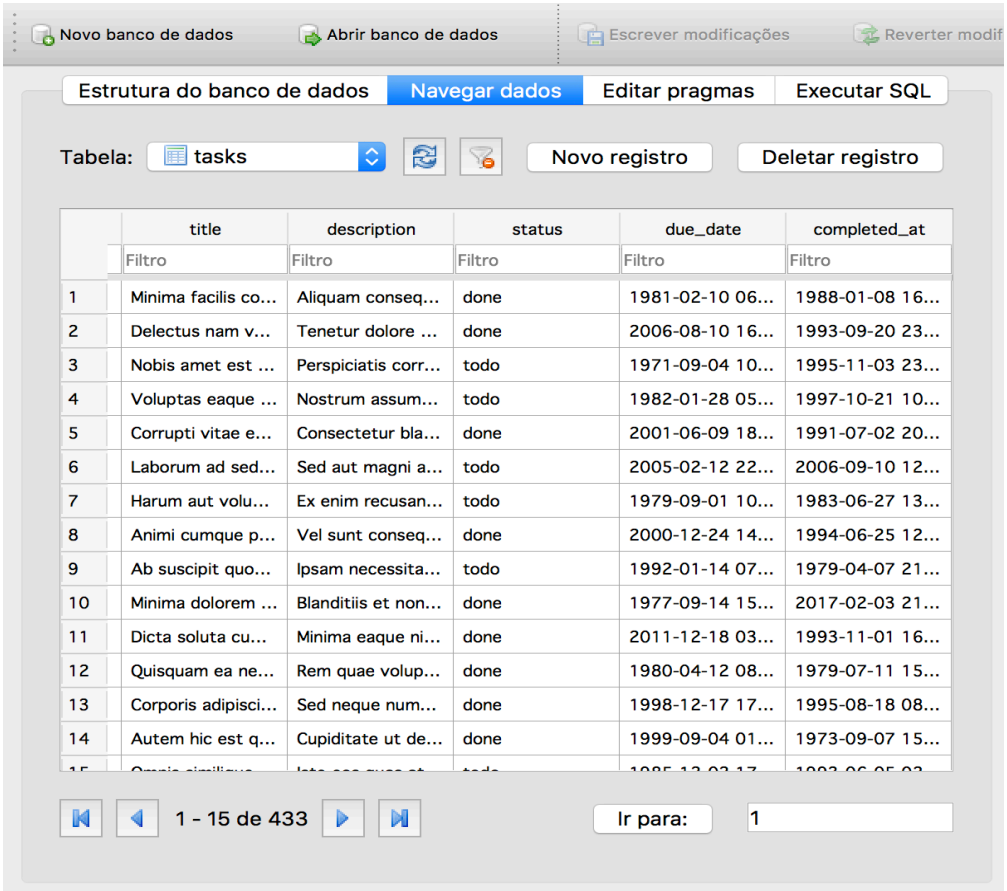


Figura 22 - Checagem do funcionamento do Banco de Dados.

6.2 CONTROLLER

Esse domínio do MVC é responsável por direcionar os *requests* conforme a regra de negócio, também de acordo com a arquitetura do projeto, ele é responsável por autenticar o usuário e garantir que as rotas do sistema estão sendo acessadas por alguém autenticado.

No caso dos controladores também foi criado um específico para cada classe, onde ele será responsável por direcionar os pedidos conforme a especificidade. Para efeito desse projeto, a função básica do controlador além da autenticação é chamar os serviços declarados no *Service Layer*.

Dessa forma, podemos utilizar como exemplo novamente o caso da classe Atividade, com o seguinte controlador:

```
<?php
abstract class TaskController extends LaravelController implements
BaseControllerInterface
{    public function service(BaseServiceInterface $service = null)
    {
        if ($service) {
            $this->service = 'TaskService';
            return $this;
        }return $this->service;}
    public function validate()
    {
        if ($this->request) Validator::make(request()->all(), $this-
>request->rules())->validate();}
    public function exists(string $id)
    {if (!$this->service->find($id))
            throw new NotFoundHttpException($this->notFoundMessage);}
    public function all()
    {return $this->service->all();}
    public function find(string $id)
    {$this->exists($id);return $this->service->find($id);}
    public function create()
    {$this->validate();
        return $this->service->create(request()->all());}
    public function update(string $id)
    {
        $this->exists($id);
        $this->validate();
        return $this->service->update($id, request()->all());}
    public function delete(string $id)
    {
        $this->exists($id);
        return $this->service->delete($id);}}
```

Figura 23 - Exemplo de um Controller Layer: Atividade.

Também é responsabilidade do domínio *Controller* do MVC a autenticação do usuário, dessa forma foi criado um controlador específico para autenticação, que utiliza de e-mail e senha para certificar se o usuário está cadastrado no banco de dados. Esse controlador utiliza o método JWT de autenticação, com 60 minutos de TTL para expirar o *token*, o controlador de autenticação pode ser verificado na Figura 24:

```php
<?php
namespace App\Http\Controllers\API;
class AuthController extends BaseController
{
    public function __construct()
    {
        parent::__construct();
        $this->middleware('auth:api', ['except' => ['login']]);
    }
    public function login()
    {
        $credentials = request(['email', 'password']);

        if (! $token = auth()->attempt($credentials)) {
            return response()->json(['error' => 'Unauthorized.'], 401);
        }
        return $this->respondWithToken($token);
    }
    public function me()
    {
        return response()->json(auth()->user());
    }
    public function logout()
    {
        auth()->logout();
        return response()->json(['message' => 'Successfully logged
out.']);
    }
    public function refresh()
    {
        return $this->respondWithToken(auth()->refresh());
    }
    protected function respondWithToken($token)
    {
        return response()->json([
            'access_token' => $token,
            'token_type' => 'bearer',
            'expires_in' => auth()->factory()->getTTL() * 60
        ]);
    }
}
```

Figura 24 - Exemplo de Controller Layer: Auth

Além de implementar os controladores de todas as classes, foi necessário estabelecer quais seriam as rotas e quais controles serão chamados de acordo com a rota. Para isso o framework Laravel disponibiliza um *router*, responsável por

```php
<?php
use Illuminate\Support\Facades\Artisan;
Route::namespace('API')->group(function() {
    // Auth Routes
    Route::post('/auth/login', 'AuthController@login');
    Route::get('/auth/logout', 'AuthController@logout');
    Route::get('/auth/refresh', 'AuthController@refresh');
    Route::get('/auth/me', 'AuthController@me');
    // App Routes
    Route::middleware(['auth:api'])->group(function() {
        // Route list
        Route::get('/help', function() {
            Artisan::call('route:list');
            return Artisan::output();
        });// Customers
        Route::get('/customers', 'CustomerController@all');
        Route::post('/customers', 'CustomerController@create');
        Route::get('/customers/{id}', 'CustomerController@find');
        Route::put('/customers/{id}', 'CustomerController@update');
        Route::delete('/customers/{id}', 'CustomerController@delete');
        Route::get('/customers/{id}/contacts', 'CustomerController@contacts');
        Route::get('/customers/{id}/tasks', 'CustomerController@tasks');
        Route::get('/customers/{id}/journeys', 'CustomerController@journeys');
        Route::get('/customers/{id}/interactions',
'CustomerController@interactions');// Journeys
        Route::post('/journeys', 'JourneyController@create');
        Route::get('/journeys/{id}', 'JourneyController@find');
        Route::put('/journeys/{id}', 'JourneyController@update');
        Route::delete('/journeys/{id}', 'JourneyController@delete');
        Route::get('/journeys/{id}/tasks', 'JourneyController@tasks'); // Tasks
        Route::post('/tasks', 'TaskController@create');
        Route::get('/tasks/{id}', 'TaskController@find');
        Route::put('/tasks/{id}', 'TaskController@update');
        Route::delete('/tasks/{id}', 'TaskController@delete');// Contacts
        Route::post('/contacts', 'ContactController@create');
        Route::get('/contacts/{id}', 'ContactController@find');
        Route::put('/contacts/{id}', 'ContactController@update');
        Route::delete('/contacts/{id}', 'ContactController@delete');//
Interactions
        Route::post('/interactions', 'InteractionController@create');
        Route::get('/interactions/{id}', 'InteractionController@find');
        Route::put('/interactions/{id}', 'InteractionController@update');
        Route::delete('/interactions/{id}', 'InteractionController@delete');
    });
});
```

Figura 25 - Exemplo de router do back-end.

organizar as rotas e direciona-las de acordo com o controlador. Nesse mesmo *router* também direciona as rotas de autenticação caso o usuário não esteja logado.

Para testar se os serviços estão funcionando é necessário rodar o servidor localmente, dessa forma, o domínio é: “http://127.0.0.1:8000/”. Utilizando a ferramenta de testes de API, o Postman, é possível simular requisições *http* a uma certa rota, colocando todas as informações necessárias para que isso aconteça. Para testar, o aluno selecionou aleatoriamente o e-mail de um usuário e o colocou em formato de um objeto JSON no *body* de uma requisição post para a rota /api/login. O resultado deve ser um *token* como resposta, como podemos verificar na imagem a seguir:

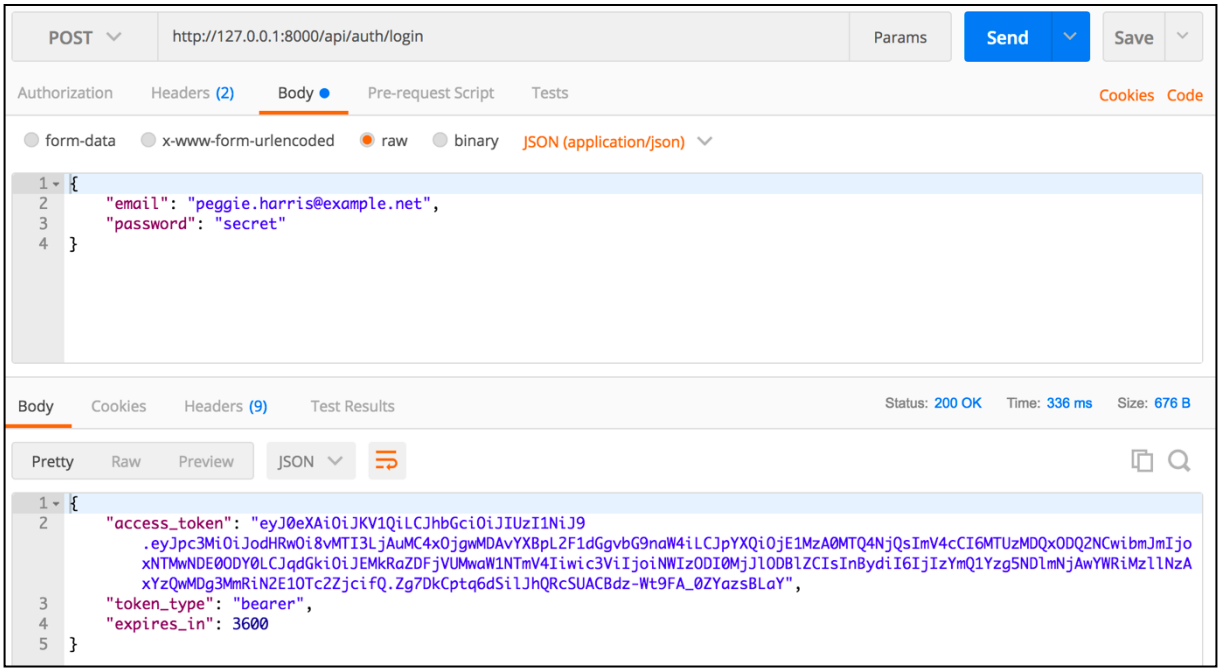


Figura 26 - Teste da rota de login.

Com o *token*, podemos simular o comportamento de um usuário em relação as rotas disponíveis. Como teste, simularemos uma requisição *http* do tipo *get* com o *token* declarado no header da requisição. Iremos fazer uma requisição para rota que busca todos os clientes de um determinado usuário, o resultado está na figura a seguir:

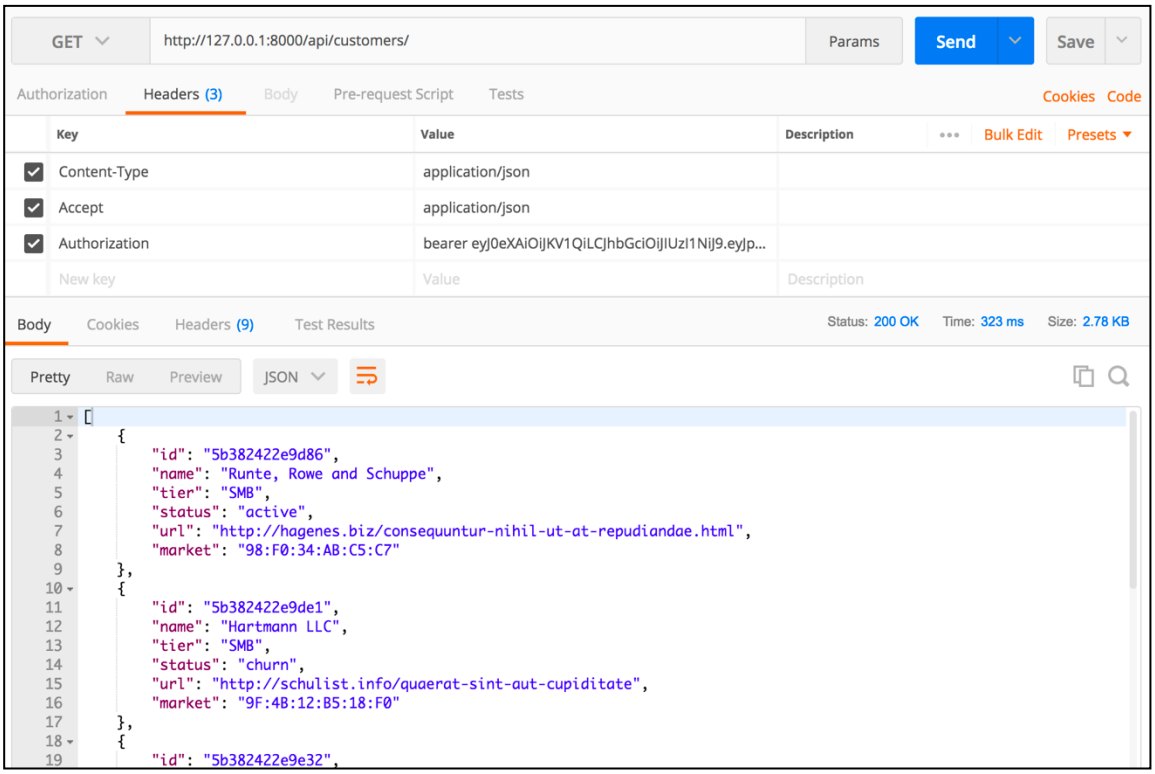


Figura 27 - Teste da rota Customers All.

Assim podemos verificar que os controladores estão realizando suas funções, tanto de autenticação quanto de distribuir os serviços de acordo com a requisição.

6.3 VIEW

O domínio View do MVC nesse projeto é responsável pela apresentação dessas informações e por requisitar os diferentes serviços de acordo com o comportamento do usuário. Para a implementação dessa parte do projeto foi utilizado a linguagem JavaScript com o framework Vue.js. A escolha dessa linguagem e desse framework foi feita por ser uma linguagem de fácil aprendizado pelo aluno, além de ter casos de implementação complementares ao Laravel que demonstraram grande sinergia dentro de sua comunidade.

Uma vantagem da utilização desse framework é a divisão da solução em componentes que podem ser acessados na forma de *tags html*, assim o aluno pode separar o código e desenvolver cada componente separadamente, facilitando a manutenção e o reuso de código.

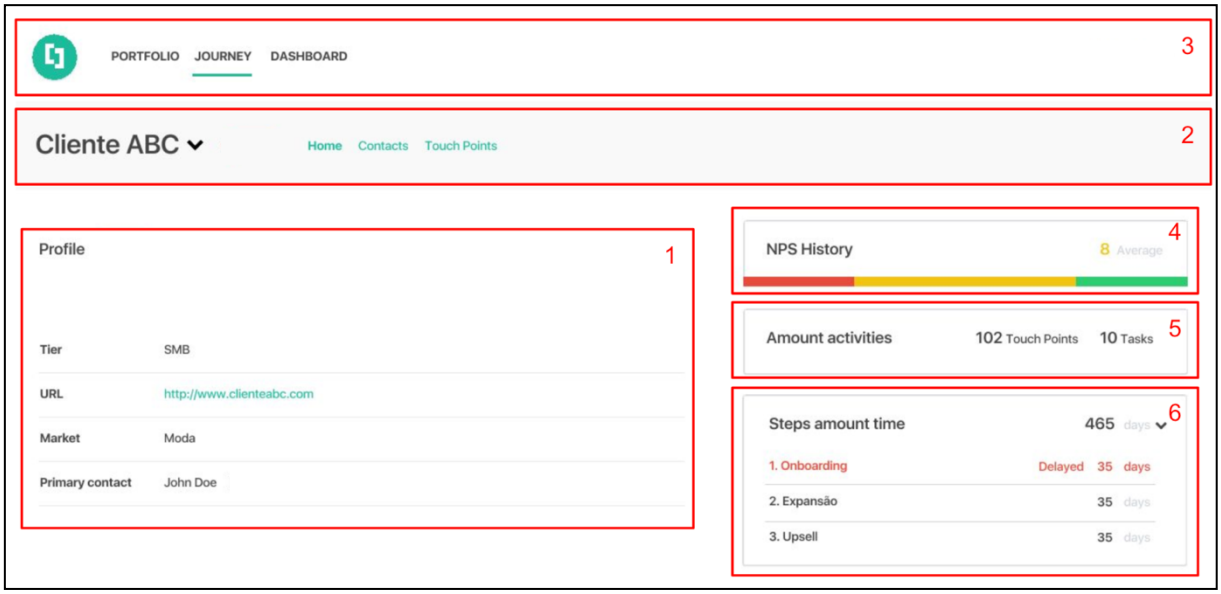
Um exemplo de componente pode ser o componente Perfil (Figura 28), que junto com as classes em CSS, ao ser colocado em produção, se parece com a Figura 29.

```
<template>
  <h2 class="float-left">Perfil</h2>
  <div class="container-fluid">
    <div class="ml-5 mt-5 float-left">
      <div>
        <span class="table first-column">Tier</span>
        <span class="table second-column">>{{ctier}}</span>
      </div>
      <div>
        <span class="table first-column">URL</span>
        <span class="table second-column">>{{curl}}</span>
      </div>
      <div>
        <span class="table first-column">Market</span>
        <span class="table second-column">>{{cmarket}}</span>
      </div>
    </div>
  </div>
</template>
<script>
    import { mapGetters } from 'vuex';
    export default {
        name: "Perfil",
        data() {
            return{
                data: {},
            }},
        computed: {
            ...mapGetters({
                ctier: 'getCustomerTier',
                curl: 'getCustomerUrl',
                cmarket: 'getCustomerMarket',
            })
        },
        created () {
            this.$store.dispatch('bringCustomerData', c_id)
        }
    }</script>
```

Figura 28 - Exemplo de componente Vue.js: Perfil.

Figura 29 - Visualização do componente Perfil.

O mesmo é feito com os demais componentes e o resultado é representado pela foto a seguir:



Como exemplo, nessa imagem estão descritos cada componente, sendo a lista de componentes a seguir:

- 1 – Perfil;
- 2 – SubnavigationCliente;
- 3 – Navigation;

- 4 – NPS History;
- 5 – Cliente Activities;
- 6 – Journey Status;

Com a junção de dos componentes temos a página do cliente e a página das interações, as duas principais telas que representam os casos de uso proposto. Como podemos verificar nas Figura 30 e Figura 31.

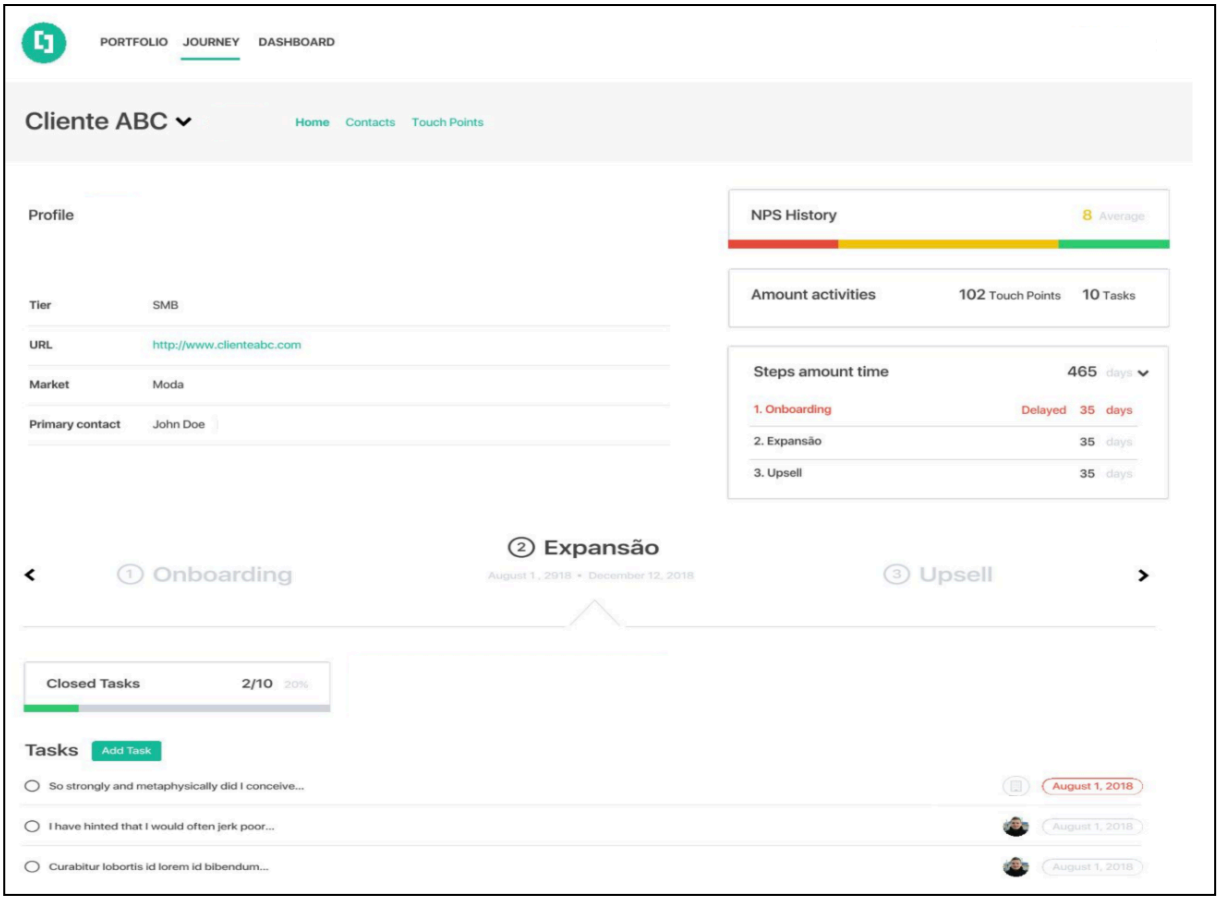


Figura 30 - Visão do cliente.

Nessa figura, podemos verificar que temos as informações necessárias para analisar a jornada do cliente e dessa forma criar iniciativas para gerencia-la, como descrito no caso de uso.

Ainda, é possível criar atividades como no exemplo, essas atividades podem ter um título, um tipo e uma data para realização, assim como descrito nos requisitos funcionais.

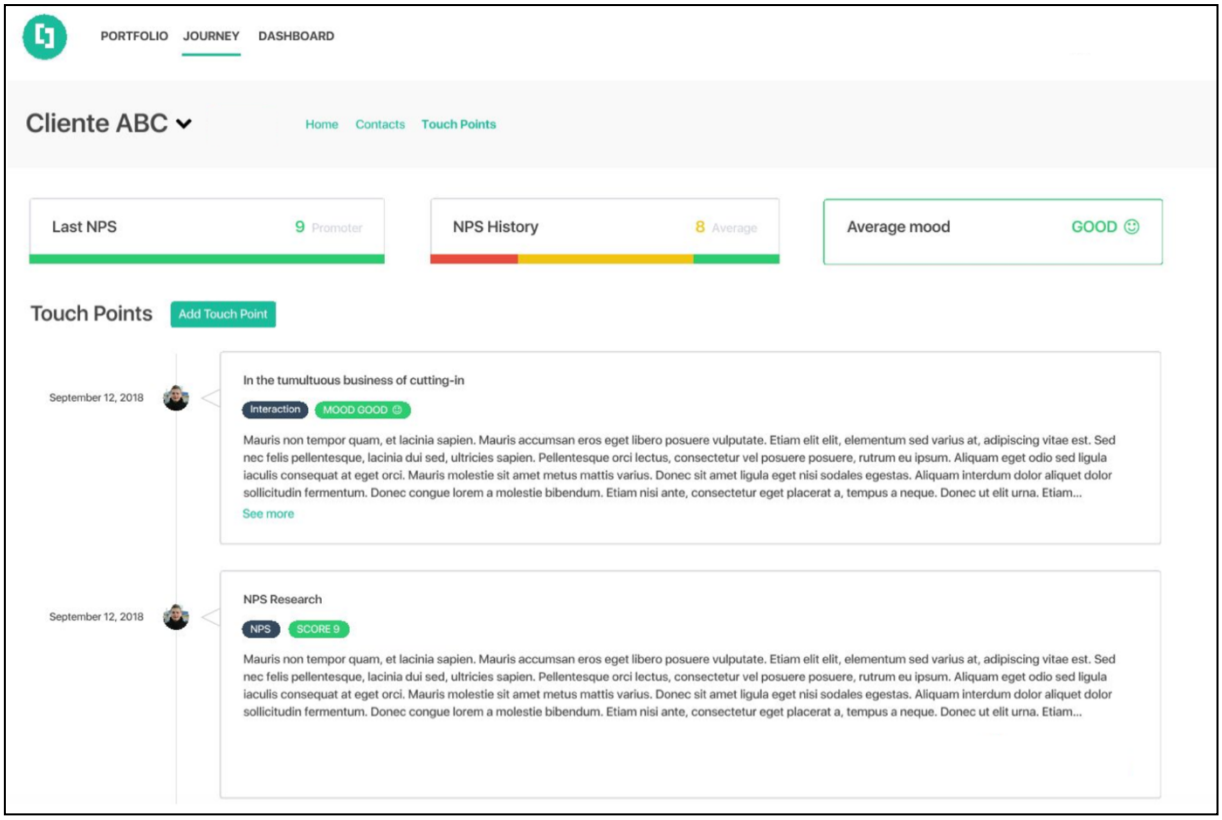


Figura 31 - Visão das interações.

Também podemos verificar que com essa tela o usuário pode realizar o segundo caso de uso: gerenciar as interações com o cliente. Aqui, é possível adicionar uma interação e declarar qual é seu tipo, podendo ser uma interação ou uma pesquisa NPS, cada uma requer um dado diferente como podemos verificar na figura. Com a criação de diferentes interações podemos verificar o histórico desses tipos de interações na parte superior da tela.

Com essas telas existe a realização do terceiro e último caso de uso, que podemos verificar na Figura 30, no componente “Cliente Activities” qual o desempenho desse cliente e dessa forma, qual o desempenho da carteira de clientes. No capítulo de perspectivas futuras o aluno cita outras possibilidades para esse caso de uso.

6.4 DEPLOYMENT

De acordo com a seção 6.4 do projeto, o deployment foi dividido de acordo com as necessidades de cada módulo da solução. Como explicitado, o Frontend e o Backend se diferem pela necessidade ou não de persistência de dados e requisições. A junção desses dois módulos em um mesmo servidor poderia gerar custos desnecessários, já que o custo de um servidor que suporte persistência de dados e requisições é diferente de um servidor que hospede uma aplicação estática.

Assim sendo, optou-se pela utilização do fornecedor de armazenamento na nuvem Amazon para o deployment do Frontend, sendo utilizado o serviço denominado S3, que é uma das opções mais baratas disponíveis pela Amazon e possui a possibilidade de somente hospedar uma aplicação estática, como é o caso do módulo Frontend desse trabalho.

Isso é feito criando uma conta na Amazon, criando um “Bucket”, que é um espaço mínimo, flexível, dentro de um endereço URL. Por meio da sincronização do repositório local o endereço URL correspondente ao banco, copia-se o repositório local para a nuvem e o configura pela própria plataforma a ser um “Bucket estático”.

Já no caso do Backend, optou-se pela solução do Heroku, que possui um plano mais competitivo do que a Amazon, possibilitando que a solução da Centrics pudesse ser testada sem custos iniciais (já que o espaço utilizado ainda é pequeno e os números de requisições também).

No caso do deployment do Backend utilizando o Heroku, sincroniza-se o repositório local da solução com a conta do aluno no fornecedor por meio do Git. Utilizando comandos do CLI é possível copiar o repositório local na nuvem e assim acessá-lo por meio da URL daquela conta. Os resultados podem ser vistos nas Figura 32 e Figura 33.

Figura 32 - Staging do Frontend da Centrics hospedado no serviço S3 da Amazon.

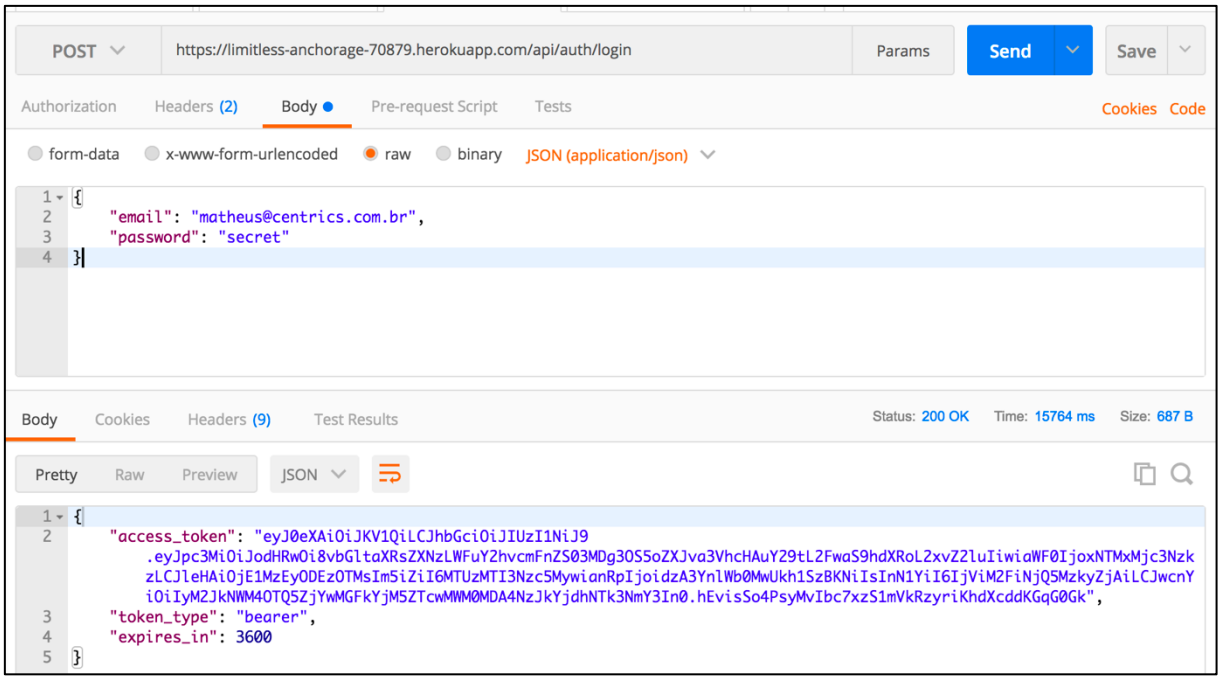


Figura 33 - Exemplo do Backend hospedado no Heroku.

7 RESULTADOS

Como resultado desse trabalho temos a prova de conceito da Centrics e que cumpre todos os casos de uso descritos no projeto. Como avaliação dessa prova de conceito, já que ela é um MVP, utiliza-se a validação da proposta de valor como métrica, assim, podemos concluir que ela cumpre os requisitos de resolver o problema de um cliente com o mínimo de funcionalidades possível, já que a Centrics finaliza esse trabalho com um primeiro cliente.

7.1 COMPARAÇÃO

Interessante ressaltar como será diferente a operação de um time de sucesso do cliente com a utilização dessa ferramenta, dessa forma se faz necessária uma comparação do estado atual das empresas que utilizam essa estratégia e de como será com a utilização da ferramenta.

Como descrito no capítulo quatro, o grande desafio das empresas no Brasil é em estabelecer um processo de sucesso do cliente. As que se aventuram por conta própria acabam por enfrentar os problemas descritos no capítulo 1:

- Visualizar a situação do relacionamento de cada cliente
- Monitorar o desempenho desses clientes no processo de relacionamento
- Avaliar o desempenho do processo e da equipe
- Escalar o processo com o aumento do número de clientes

Com os casos de uso do projeto (Capítulo 5) implementados, pode-se atender essas necessidades, excetuando-se a avaliação do desempenho da equipe, que será comentado nas perspectivas futuras (Capítulo 8.1).

Ainda, podemos comprar a prova de conceito da Centrics com as soluções que estão sendo adotadas hoje pela indústria, como descrito no capítulo 4.2.4:

Ferramenta	Limitação	Centrics
Planilhas	Conforme o processo ganha complexidade, aumentando o número de atividades, de pessoas na equipe e o número de clientes, se torna custoso alimentar todas as informações e aumentam as chances de erro	Construída com tecnologias que atendam a demanda de maneira escalável
CRM	Não utilizam a linguagem necessária para o pós-venda, muito menos conseguem medir informações que são essenciais em relação ao cliente, como NPS e volume de interações. Ainda, a maioria dos CRMs não consegue automatizar atividades. Do contrário, caso consigam, são soluções completas e caras.	Feita sobre medida para o caso de uso de um time de sucesso do cliente, tanto em linguagem como métricas
Automação de e-mails	Suas métricas são de marketing e seus resultados não são integrados, gastando-se tempo para encontrar as métricas certas e isolar a situação de cada cliente	Será centralizada no cliente, com as métricas adequadas
CSM	São focadas exclusivamente em empresas com modelo SaaS, são caras e requerem meses para implantação e operação	Mais simples, foco em empresas tradicionais, plug-and-play

7.2 FEEDBACK DO CLIENTE

Como validação da proposta de valor dessa prova de conceito podemos ressaltar o primeiro cliente da Centrics, a Brognoli, uma das maiores imobiliárias da américa latina, empresa tradicional, que não conseguia ter acesso as tecnologias que hoje são focadas para SaaS. Ainda, os CRMs mais completos que atenderiam suas necessidades eram muito caros e demorariam muito tempo para serem implantados, isso sem contar a demora no aprendizado pela equipe de quase 100 pessoas.

Como podemos ver no Anexo A, a avaliação do cliente em relação a ferramenta foi descrita como uma solução que quando implantada será adequada para suas

necessidades e poderá gerar potenciais ganhos em termos de retenção de clientes e expansão de contratos.

Assim, considera-se que o projeto atingiu seus objetivos.

8 CONSIDERAÇÕES FINAIS

Como podemos verificar ao longo desse documento, o objetivo principal foi alcançado: conceitualizar, desenvolver e prototipar uma prova de conceito como solução de relacionamento com o cliente para pós-venda utilizando metodologias de desenvolvimento de software e de negócio.

Interessante ressaltar que todo o processo de conceitualização, projeto e desenvolvimento desse trabalho foi feita conforme metodologias inovadoras de desenvolvimento de negócios tecnológicos e que propiciaram ao autor um grande aprendizado, assim como reduziram os riscos do desenvolvimento e conduziram o projeto a um cliente.

No capítulo 1 descrevemos o problema a ser resolvido com essa prova de conceito, problema que é um artefato resultante de um processo de validação de hipóteses descrito no capítulo 3, validado com clientes reais e com interações com a indústria.

No capítulo 3 elencamos todas as metodologias utilizadas para chegar ao desenvolvimento desse projeto, passando pelo procedimento de entendimento de problema, até o desenvolvimento, tanto a nível de negócio quanto a nível de engenharia de software. Foram mais de 80 interações com usuários e possíveis clientes dessa solução e entregas como: lista de necessidades, requisitos e modelagem do software e o desenvolvimento em si.

No capítulo 5 descreve-se todo o projeto resultante desse trabalho, focado em requisitos reais advindo de entrevistas com usuários e possíveis clientes, passando pela fase de protótipo conceitual com wireframes e mockups, avaliando a importância de cada um para esses usuários. Foram construindo 2 wireframes nos dois ciclos rodados com 10 diferentes usuários e possíveis clientes.

Como podemos verificar no capítulo de desenvolvimento, toda a arquitetura do desenvolvimento foi cumprida, com o desenvolvimento do domínio Model e suas camadas, o domínio Controller e suas camadas, e os componentes descritos no domínio View. Esse desenvolvimento foi feito baseado no cumprimento dos casos de uso descritos no projeto, seguindo também a arquitetura escolhida.

8.1 PERSPECTIVAS FUTURAS

Assim como falado no capítulo de resultados, existem outras necessidades apontadas pelo mercado que podem ser atacadas por essa solução, como por exemplo a análise do desempenho do time e a integração com outras ferramentas (como adiantado pela arquitetura do sistema). Porém, como se trata de uma prova de conceito esses outros casos de uso ficaram para os próximos passos do desenvolvimento.

Como a Centrics teve uma validação considerada relevante de um cliente, é possível que o aluno de continuidade ao projeto e também ao negócio.

Como próximos passos recomenda-se expandir os casos de uso para outras situações, onde a empresa utiliza um CRM de vendas ou uma ferramenta de Suporte que possam ser integradas e melhorarem ainda mais o poder de controle e gerenciamento que a ferramenta teria em relação a jornada e ao sucesso do cliente.

Um outro ponto importante que não esteve no mapa nesse trabalho é a geração de alertas para usuário caso algum evento importante ocorra. Esses alertas podem ser muito importantes conforme o número de clientes aumenta, já que não é fácil recordar tudo o que está acontecendo com esses clientes em uma linha do tempo.

REFERÊNCIA

ABES - ASSOCIAÇÃO BRASILEIRA DE SOFTWARE. **Mercado Brasileiro de Software: Panorama e Tendências**. ABES. [S.l.], p. 28. 2017.

BLANK, S. G. **The Four Steps to the Epiphany:** Successful Strategies for Products that Win. [S.l.]: [s.n.], v. 4, 2007.

BUTTLE, F. **Customer Relationship Management:** Concepts and Tools. Oxford: Elsevier, v. 1, 2004.

DWYER, F. R. Developing Buyer-Seller Relationships. **Journal of Marketing**, v. 51, p. 11-27, Abril 1987.

ENDEAVOR. www.endeavor.com. **Endeavor**, 2017. Disponivel em: <http://acaojr.com.br/importancia-do-customer-success-para-o-sucesso-do-seu-cliente-e-da-sua-empresa/>. Acesso em: 2 junho 2018.

ENDEAVOR. https://endeavor.org.br/. **Endeavor**, 2018. Disponivel em: <https://endeavor.org.br/marketing/mentoria-online-customer-success/>.

FORBES. Don't Get Lazy About Your Client Relationships. **Forbes**, 2013. Disponivel em: <https://www.forbes.com/sites/patrickhull/2013/12/06/tools-for-entrepreneurs-to-retain-clients/#41c97a1c2443>. Acesso em: 25 Março 2018.

FORBES. B2B Marketing Must Focus On 1:1. **Forbes**, 2014. Disponivel em: <https://www.forbes.com/sites/danielnewman/2014/10/21/b2b-marketing-must-focus-on-11/#510bd4124382>. Acesso em: 30 Março 2018.

G1. Empresas de tecnologia de Florianópolis faturam juntas R$ 6 bilhões ao ano. **G1**, 2018. Disponivel em: <https://g1.globo.com/sc/santa-catarina/sc-que-da-certo/noticia/empresas-de-tecnologia-de-florianopolis-faturam-juntas-r-r-11-bilhoes-ao-ano.ghtml>. Acesso em: 10 Junho 2018.

GAINSIGHT. www.gainsight.com. **Gainsight**, 2016. Disponivel em: <https://www.slideshare.net/GainsightHQ/developing-the-ultimate-customer-success-strategy>. Acesso em: 20 junho 2018.

GARTNER. Gartner Surveys Confirm Customer Experience Is the New Battlefield. **Gartner**, 2014. Disponivel em: <https://blogs.gartner.com/jake-sorofman/gartner-surveys-confirm-customer-experience-new-battlefield/>. Acesso em: 23 Abril 2018.

GIL, A. C. **Como Elaborar Projetos de Pesquisa**. São Paulo: Atlas SA, v. 5, 2010.

HARVARD BUSINESS REVIEW. Why Customer Referrals Can Drive Stunning Profits. **Harvard Business Review**, 2011. Disponivel em: <https://hbr.org/2011/06/why-customer-referrals-can-drive-stunning-profits>. Acesso em: 20 Abril 2018.

HARVARD BUSINESS REVIEW. The Value of Keeping the Right Customers. **Harvard Business Review**, 2014. Disponivel em: <https://hbr.org/2014/10/the-value-of-keeping-the-right-customers>. Acesso em: 20 Março 2018.

JÄRVINEN, P. On Research Methods, Tampere, 2004.

JÄRVINEN, P. Action Research is Similar to Design Science. **Quality & Quantity**, Tampere, v. 41, n. 1, p. 37-54, Fevereiro 2007.

LINCOLN MURPHY, D. S. E. N. M. **Customer Success:** Como as empresas inovadoras descobriram que a melhor forma de aumentar a receita é garantir o sucesso do cliente. São Paulo: Autêntica Business, v. 1, 2017.

MCKINSEY. The consumer decision journey. **McKinsey**, 2009. Disponivel em: <https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/the-consumer-decision-journey>. Acesso em: 29 Junho 2018.

MCKINSEY. Introducing customer success 2.0: The new growth engine. **McKinsey**, 2018. Disponivel em: <https://www.mckinsey.com/industries/high-tech/our-insights/introducing-customer-success-2-0-the-new-growth-engine>. Acesso em: 20 Junho 2018.

MINDMINERS. O panorama do Customer Success: Um estudo sobre como as empresas investem em Customer Success no Brasil. MindMiners. [S.l.]. 2018.

NEW VOICE MEDIA. The $62 billion customer service scared away. **New Voice Media**, Florianópolis, 2016. Disponivel em: <https://www.newvoicemedia.com/blog/the-62-billion-customer-service-scared-away-infographic>. Acesso em: 15 fevereiro 2018.

OLSEN, D. **The Lean Product Playbook:** How to innovate with minimum viable products and rapid customer feedbacks. New Jersey: Wiley, 2015.

POPE, G. E. K. A. S. T. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. **Journal of object oriented programming**, Mountain View, January 1998.

RESULTADOS DIGITAIS. Por que todo pós-vendas de uma empresa de tecnologia deveria mudar para Customer Success em 2016. **Resultados Digitais**, 2016.

Disponivel em: <https://resultadosdigitais.com.br/blog/por-que-empresas-de-tecnologia-deveriam-ter-uma-area-de-customer-success/>. Acesso em: 21 Abril 2018.

RIES, E. **A Startup Enxuta**. Rio de Janeiro: Editora Casa da Palavra, v. 1, 2012.

RIES, E. Testar. In: RIES, E. **Lean Startup**. Rio de Janeiro: Leya, v. 1, 2012. p. 84-105.

STATISTA. Finance Online. **Finance Online**, 2018. Disponivel em: <https://financesonline.com/2018-saas-industry-market-report-key-global-trends-growth-forecasts/>. Acesso em: 29 Jun 2018.

WAZLAWICK, R. S. Análise e Projeto de Sistemas de Informação Orientados a Objeto. [S.l.]: Campus, v. 4, 2014.

ZUORA. **Subscription Economy Index**. Zuora. [S.l.], p. 22. 2017.

APÊNDICE A – CÓDIGO FONTE DO DATABASE SEEDER

```php
<?php

class DatabaseSeeder extends Seeder
{
    public function run(Faker $faker)
    {        $this->truncate();
        factory(Account::class, 2)->create();
        $accounts = Account::all();
        foreach($accounts as $account) {
            $aid = $account->id;
            factory(User::class, rand(1, 6))->create(['account_id' =>
$aid]);     factory(Customer::class, rand(5, 30))->create(['account_id'
=> $aid]);
            $customers = Customer::where('account_id', $aid)->get();
            $users = User::where('account_id', $aid)->get();
            foreach ($customers as $customer) {
                $cid = $customer->id;
                factory(Contact::class, rand(1, 5))->create([
                    'account_id' => $aid,'customer_id' => $cid
                ]);
                $contacts = Contact::where([
                    ['account_id', $aid], ['customer_id', $cid],
                ])->get();
                factory(Interaction::class, rand(1, 10))->create([
                    'account_id' => $aid,'customer_id' => $cid,
                    'user_id' => $users->random()->id,
                    'contact_id' => $contacts->random()->id,
                ]);
                $status = $faker->randomElement(['todo', 'doing',
'done']);
                factory(Journey::class, rand(3, 5))->create([
                    'account_id' => $aid,'customer_id' => $cid,
                    'status' => $status,
                    'done_at' => ($status == 'done' ? $faker->date() :
null),
                ]);
                $journeys = Journey::where([
                    ['account_id', $aid],
                    ['customer_id', $cid],
                ])->get();

                foreach ($journeys as $journey) {
                    $jid = $journey->id;
                    $assignee = $faker->randomElement(['user',
'automatic', 'customer']);
                    factory(Task::class, rand(3, 10))->create([
                        'account_id' => $aid,
                        'customer_id' => $cid,
                        'journey_id' => $jid,
                        'user_id' => ($assignee == 'user' ? $users-
>random()->id : null),
                        'assignee' => $assignee,
                    ]);
```

ANEXO A – AVALIAÇÃO DO CLIENTE

O objetivo de qualquer empresa é fazer negócios que sejam bons para os seus clientes e que gerem uma boa percepção e crescimento da marca no mercado. E para alcançar este resultado em um mercado cada dia mais competitivo e saber se os clientes estão, de fato, tendo resultado e uma boa experiência com seu serviço, precisamos trabalhar com foco no sucesso do cliente.

Nesta linha o MVP da Centrics atinge esse objetivo e atende as demandas de negócio que são referentes a definição, monitoramento e execução da Gestão do Sucesso do Cliente. Como ainda é um MVP, não temos resultados tangíveis, mas, esta totalmente alinhado com o objetivo da empresa, que esta focada no cliente e que precisa aprender um pouco de cada cliente, seu mercado, especificidades e, principalmente, com os erros cometidos para melhorar cada dia mais para os clientes.

Esperamos que este processo de co-criação com a Centrics, nos ajude a ter um processo estruturado, tendo todos os envolvidos alinhados e focados no cliente, assim, a experiência dele será incrível e a percepção inicial deixará uma boa impressão… não estamos romantizando o cliente, estamos falando de retenção e divulgação da sua marca.

Eduardo Barbosa
CEO Brognoli