UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

HELYEDERSON NAVES

APLICAÇÃO DE METODOLOGIAS ÁGEIS E DE STARTUPS NA CRIAÇÃO DE UM FRAMEWORK DE DOMÍNIO

PONTA GROSSA

2022

HELYEDERSON NAVES

APLICAÇÃO DE METODOLOGIAS ÁGEIS E DE STARTUPS NA CRIAÇÃO DE UM FRAMEWORK DE DOMÍNIO

Application Of Agile and Startup Methodologies In The Creation Of A Domain Framework

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Departamento de Informática da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientadora: Profa. Dra. Simone Nasser Matos
Coorientador: Prof. Me. Giovane Galvão

PONTA GROSSA

2022

HELYEDERSON NAVES

APLICAÇÃO DE METODOLOGIAS ÁGEIS E DE STARTUPS NA CRIAÇÃO DE UM FRAMEWORK DE DOMÍNIO

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Departamento de Informática da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 24/maio/2022

_____

Profa. Simone Nasser Matos
Doutora
Universidade Tecnológica Federal do Paraná

_____

Profa. Giovane Galvão
Mestre
Centro Universitário Campo Real

_____

Profa. Simone Bello Kaminski Aires
Doutora
Universidade Tecnológica Federal do Paraná

_____

Prof. Tarcizio Alexandre Bini
Doutor
Universidade Tecnológica Federal do Paraná

PONTA GROSSA

2022

RESUMO

O desenvolvimento de *frameworks* de domínio pode ser realizado por metodologias específicas e é um dos produtos que podem ser criados em *startup*. As metodologias voltadas para *frameworks* consideram aspectos relacionados a compreensão do domínio sem a participação do cliente e o processo de desenvolvimento exige a execução de diversas fases. Devido a isso, metodologias ágeis e de *startup* podem auxiliar a modelagem e desenvolvimento do *framework* como produto. Este trabalho aplicou as metodologias *Design Thinking, Lean Startup* e *Scrum* na modelagem e implementação de um *framework* de domínio na área de atendimento automatizado. A metodologia *Design Thinking* ajudou a modelar o *framework* com aspecto de produto, compreendendo o problema e definindo objetivos para a solução. *Lean Startup*, por sua vez, permitiu realizar a análise das ideias abrangendo a participação dos usuários. O *Scrum*, que é uma metodologia ágil de desenvolvimento proposto pela *Lean Startup*, possibilitou que metas de desenvolvimento sejam estabelecidas e que implementações incrementais no produto pudessem ser criadas. A aplicação das metodologias ágeis e de *startup* no desenvolvimento de um *framework* de domínio permitiu obter artefatos e realizar análises que não são usualmente abordadas por metodologias de desenvolvimento de *frameworks.* O *framework* de domínio foi instanciado para aplicações-exemplos relacionadas a tipos de serviços de atendimento automatizados a fim de realizar sua validação.

Palavras-chave: *framework; startup*; *Lean Startup*; *Scrum*.

ABSTRACT

The development of domain frameworks can be accomplished by specific methodologies and is one of the products that can be created in startup. These methodologies consider aspects related to domain comprehension without customer involvement and the development process requires the execution of several phases. Due to this, agile and startup methodologies can assist the modeling and development of the framework as a product. This paper applied the Design Thinking, Lean Startup, and Scrum methodologies to the modeling and implementation of a domain framework in the area of automated customer service. The Design Thinking methodology helped model the framework with a product aspect, understanding the problem and defining goals for the solution. Lean Startup, in turn, allowed the analysis of the ideas, including user participation. Scrum, which is an agile development methodology proposed by Lean Startup, allowed development goals to be established and incremental implementations of the product to be created. The application of the agile and startup methodologies in the development of a domain framework made it possible to obtain artifacts and perform analysis that are not usually addressed by framework development methodologies. The domain framework was instantiated for some types of automated services in order to perform its validation.

Keywords: *framework; startup*; *Lean Startup*; *Scrum*.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O termo “*startup*” se popularizou internacionalmente ao final do século XX e foi utilizado para classificar as empresas de tecnologia da informação que apareceram durante o período conhecido como “bolha da internet”, com as chamadas empresas ponto-com. Uma *startup* pode ser definida como um novo projeto empresarial focado no desenvolvimento de novos produtos e serviços sobre condições de incerteza (RIES, 2012), ou seja, uma ideia, uma esperança do que pode vir a ser e com um objetivo que poucos conseguem ver (BLANK, 2013).

Empreendedores de sucesso e suas *startups* contam suas histórias e afirmam que determinação, *timming* e um ótimo produto são o caminho para fama e fortuna, mas isto é mais complexo que obter sucesso. Um começo promissor geralmente resulta em fracasso e um ótimo produto pode nunca encontrar sua utilidade (RIES, 2012). Nesse cenário, cerca de 70% das *startups* de tecnologia fracassam (CB INSIGHTS, 2022).

Quando uma ideia surgir para o desenvolvimento de um produto inovador, geralmente acompanhado pela criação de uma *startup*, ele deve ocupar o mais breve possível o nicho de mercado para o qual é destinado. O desenvolvimento de um *framework* de domínio como produto de *software* de uma *startup* pode ser realizado por metodologias específicas de *frameworks.* Porém, essas metodologias consideram o entendimento do domínio sem a participação do cliente e o processo de desenvolvimento é formado por diversas fases que exigem mais tempo de desenvolvimento do que a *startup* pode dispor. Com o objetivo de acelerar o desenvolvimento de um *framework* como produto de software de uma *startup* é possível utilizar metodologias de desenvolvimento de produto abordadas pelas metodologias de *startups*, que muitas vezes utilizam métodos ágeis de desenvolvimento.

Métodos ágeis são focados na solução, sendo aplicados a situações em que o problema é bem conhecido e entendido. Mas a solução não, entretanto, no âmbito de *startups*, em que o próprio termo é definido como sendo uma situação em que a incerteza prevalece, muitas vezes nem o problema, nem a solução são bem compreendidos (BJÖRK; LJUNGBLAD; BOSCH, 2013). Mesmo que desta situação resulte em um *software* funcional que atenda às necessidades para o qual foi projetado, uma documentação deficiente pode resultar em problemas de refatoração

que, mesmo na comunidade ágil como na Engenharia de *Software*, é relacionado a uma redução na complexidade do código e impactos na capacidade de compreensão e manutenibilidade de um sistema de *software* (MOSER et al., 2008). Metodologias ágeis como o *Scrum* (SUTHERLAND, 2016) permite o rápido desenvolvimento e incremento de um produto ao mesmo tempo que produz artefatos que auxiliam a criação de uma documentação característica da Engenharia de *Software*.

Existem metodologias voltadas para *startups*, algumas se destacando mais, outras menos, mas cada uma defende sua visão de como levar o produto ou serviço ao sucesso. As metodologias utilizadas por *startups* como *Customer Development* (BLANK, 2013), *Design Thinking* (MUELLER-ROTERBERG, 2018) e *Lean Startup* (RIES, 2012) fornecem importância aos negócios, descoberta de mercado e participação do usuário se afastando de práticas comuns a Engenharia de *Software* (LEONESSA, 2016). A metodologia *Lean Startup* possui uma interseção entre as áreas de planejamento e negócios e conceitos de Engenharia de *Software*. Mas, mesmo com focos diferentes, tanto as filosofias modernas empresariais quanto de engenharia têm como objetivo produzir experiências de alta qualidade para o público alvo (RIES, 2012) e, da maneira mais rápida e eficiente possível, preencher o mercado com seu produto.

Este trabalho aplicou metodologias para *startups* como *Lean Startup* e *Customer Development* e ágeis por meio do *Scrum* e utilizando conceitos de *Design Thinking* com o intuito de combinar a agilidade dos métodos ágeis com os métodos que tem foco no consumidor e no mercado como as metodologias para *startups*. A metodologia foi aplicada no desenvolvimento de um *framework* de domínio de serviço de Atendimento Automatizado (AA) ao cliente (GONÇALEZ, 2020), também chamado de *chatbot*, com a finalidade de prover funcionalidades a novos produtos de *startups*, podendo dar origem a soluções que auxiliem as *startups* em seus objetivos. Gonçalez (2020) afirma que a grande inovação dos *chatbots* é a capacidade da tecnologia em alcançar os clientes em qualquer dispositivo e nos canais de mensagem mais populares, ampliando o alcance e a comunicação da central de atendimento com seus clientes.

1.1 Justificativa

O desenvolvimento de um produto de *software* em uma *startup* exige rapidez e muitas vezes um produto deve ser criado para rapidamente ocupar um nicho ou suprir uma necessidade do mercado. Um produto para *startup* não pode demorar em seu desenvolvimento, pois isto pode deixar uma oportunidade se perder. Porém, se tratando de produto de *software*, é necessário manter a manutenibilidade e capacidade de refatoração a fim de proporcionar uma maior vida útil ao produto.

O uso de *frameworks* de domínio e a correta utilização de metodologias para *startup* pode contribuir para desenvolver um melhor produto de *software* de maneira ágil e ainda evitar futuros problemas na manutenção do mesmo. Através de um *framework,* a solução pode ser replicada no desenvolvimento de futuros produtos de maneira que se adapte as necessidades da *startup*. O domínio escolhido para a criação do *framework* foi de atendimento automatizado, porque em uma *startup* o produto é oriundo por meio da descoberta do cliente e seus interesses.

1.2 Objetivos

O objetivo geral do trabalho é aplicar metodologias ágeis e de *startup* no desenvolvimento de um *framework* de domínio e analisar as contribuições de se utilizar essas metodologias na criação do *software*.

Os objetivos específicos são:

- Modelar o domínio do *framework* sobre serviço de atendimento;
- Implementar o *framework*;
- Validar e Estender o *framework* proposto.

1.3 Organização do trabalho

Este trabalho está organizado em cinco capítulos. O Capítulo 1 apresenta a motivação para seu desenvolvimento. O Capítulo 2 aborda o referencial teórico necessário para o desenvolvimento do trabalho, tais como *design*, *startups*, metodologias ágeis e *framework*. O Capítulo 3 retrata a metodologia e a utilização das metodologias ágeis e de *startup* no desenvolvimento de um *framework* de domínio. O Capítulo 4 relata os resultados obtidos a partir da aplicação da metodologia. Por fim, o capítulo 5 apresenta as conclusões e trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico utilizado para o desenvolvimento deste trabalho. A Seção 2.1 descreve a metodologia de desenvolvimento de negócio denominada de *Design Thinking* utilizada para definir os objetivos do produto a ser desenvolvido. A Seção 2.2 define o que é uma *startup*, apresenta sua importância e explica as metodologias para *startup Customer Development* e *Lean Startup*. A Seção 2.3 discorre sobre a origem das metodologias ágeis e detalha os passos, papeis e artefatos da metodologia ágil *Scrum*. A Seção 2.4 define o que é um *framework* de domínio através do processo de Johnson (1993) para o desenvolvimento de *frameworks* de domínio. A Seção 2.5 apresenta considerações finais sobre o capítulo.

2.1 *Design Thinking*

Quando se tem a finalidade de criar um novo produto, é necessário idealizar o que se deseja, definir objetivos, metas e o escopo geral. Para isso é possível utilizar conceitos e técnicas de *design* de produto a fim de definir ideias e validá-las antes de se iniciar o desenvolvimento de um produto de *software*.

Banach e Ryan (2009) definem *design* como um processo interativo de tomada de decisão que produz planos pelos quais recursos são convertidos em produtos ou sistemas que atendem às necessidades humanas. Löbach (2001) afirma que *design* é uma ideia ou plano para a solução de um problema e o ato de *design*, então, é dar corpo à ideia e transmiti-la ao outros. A definição implica que *design* é focado em resolver problemas, exigindo uma intervenção ativa e não apenas entendê-los (BANACH; RYAN, 2009).

Ambrose e Harris (2011) compilam o processo de *design* de um produto em sete etapas ou estágios:

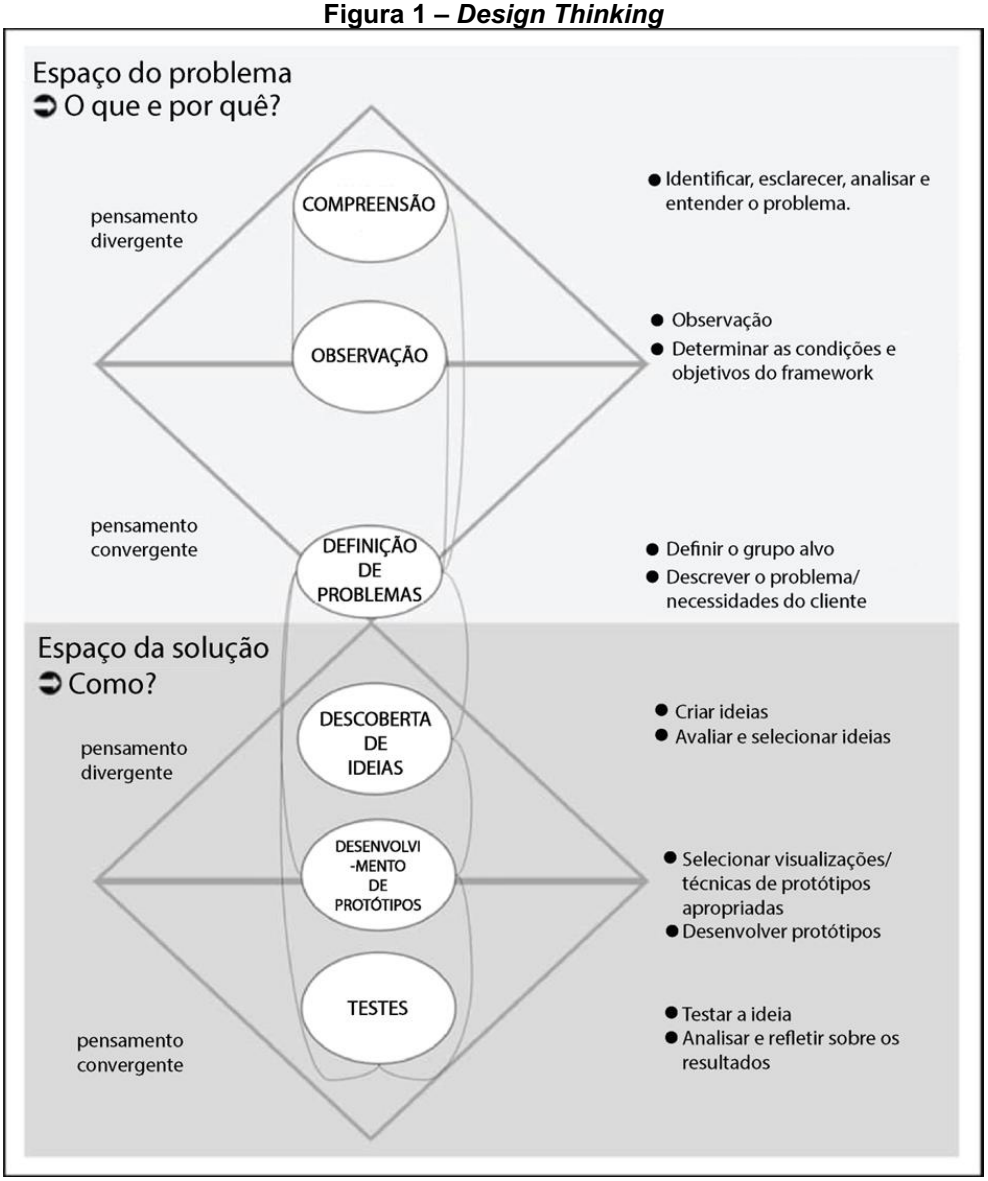
- Definir – Estabelecer qual é o problema.
    - *Briefing* – O *briefing* de um *design* apresenta os requisitos, objetivos e expectativas de um cliente para o trabalho. Os objetivos devem ficar claros ao *designer*.
- Pesquisar – Coletar informações básicas. Ao analisar e aprovar o *briefing* o *designer* deve buscar informações que possam ser inseridas no processo de idealização.

- Idealizar – Criar potenciais soluções para o problema. Durante o estágio de idealização, a equipe de projeto baseia-se nas informações da pesquisa e nas restrições estabelecidas no *briefing*.
- Prototipagem – Resolver soluções. O estágio de idealização gera uma variedade de soluções potenciais para o *briefing* antes da seleção da solução ser escolhida. A prototipagem de algumas possíveis soluções permite testar aspectos específicos e fornece uma base melhor de comparação na fase de seleção.
- Selecionar – Fazer escolhas. O estágio de seleção é onde uma das ideias potenciais idealizadas serão escolhidas para o desenvolvimento. A ideia deve ser selecionada seguindo o critério de adequação ao que foi definido no *briefing.*
- Implementar – Entregar a solução para o *briefing*. Nesta etapa os responsáveis pelo desenvolvimento do produto devem implementar a ideia escolhida e garantir que ela atenda aos requisitos.
- Aprender – Obter *feedback*. O estágio final consiste em aprender com o ocorrido durante o design do produto.

Design Thinking é uma metodologia de negócio que tenta projetar as abordagens e métodos do *design* nos processos de negócios. Mueller-Roterberg (2018) define *Design Thinking* como uma abordagem integrativa, visando o interesse do cliente e com foco em empatia caracterizada por iterações no processo de desenvolvimento do projeto.

De acordo com Plattner, Meinel e Leifer (2011), *Design Thinking* consiste em seis fases iterativas: *Compreensão*, *Observação*, *Definição de problemas*, *Descoberta de ideias*, *Desenvolvimento de protótipos* e *Testes*, como pode ser visto na Figura 1.

Figura 1 – *Design Thinking*



Fonte: Müller-Roterberg (2018).

A Compreensão é a primeira etapa e consiste em ter compreensão do problema, dos objetivos, requisitos e expectativas. Todo o escopo do problema deve ser compreendido.

Na segunda etapa, Observação, é realizada a observação das necessidades dos clientes. Este pode ser feita por meio de pesquisas e entrevistas ao público-alvo.

A Definição de problemas é a terceira etapa e nesta são obtidos os dados os quais devem ser compilados e resumidos a um único tópico bem definido.

Na etapa de Descoberta de ideias o processo de idealização se inicia. As descobertas das fases anteriores devem ser levadas em consideração para a formalização das ideias e a mais promissora deve ser selecionada.

Na quinta etapa, Desenvolvimento de protótipos, é realizado uma prototipagem rápida da ideia escolhida. O desenvolvimento deve ser realizado o mais rápido possível e deve representar o produto e os serviços.

Por fim, na última fase de Testes, as ideias devem ser mais desenvolvidas e testadas por meio de mais experimentos e *feedback* de clientes. Outras questões de desenvolvimento, produção e mercado devem ser validadas nesta fase.

O *Design Thinking* não trata apenas de superar os desafios do projeto, mas também de acrescentar valor ao produto desenvolvido, pois permite identificar melhor as necessidades dos usuários.

2.2 *Startup*

Ries (2012) define *startup* como uma instituição humana projetada para criar novos produtos e serviços sob condições de extrema incerteza. Blank (2010) define *startup* como uma organização formada para buscar um modelo de negócio repetível e escalável. Graham (2012) caracteriza uma *startup* como uma empresa projetada para crescer rapidamente e releva características geralmente atribuídas à *startups* como inovação tecnológica, investimento de capital de risco ou ideias inovadoras de produto e define que o essencial para uma *startup* é crescimento, que atua como o compasso que guia as decisões do modelo de negócios.

O modelo de negócios e as aspirações de uma *startup* iniciam-se com o produto, o *Minimum Viable Product* (MVP) que consiste na versão de um novo produto que, com mínimo de esforço, tempo de desenvolvimento e investimento, permite que a equipe obtenha o máximo de aprendizado validado sobre os clientes (RIES, 2012). A partir do MVP, é possível compreender se o produto e, consequentemente, a *startup*, podem ter alguma chance de sucesso.

Startups, sejam pequenas empresas ou grandes companhias, desempenham um papel significativo no crescimento econômico. As *startups* criam empregos, contribuem para o dinamismo econômico, estimulam a inovação e promovem a concorrência. Muitas vezes, *startups* causam um impacto econômico que molda sua região, como a *Infosys* mudou Bangalore, o *Alibaba* afetou Hangzhou, a *Microsoft* mudou *Redmond* e o Google transformou *Mountain View*, Califórnia (DIDAR, 2016).

Algumas *startups* conseguem causar tanto impacto no mercado que a grandiosidade do feito é comparável à de encontrar uma criatura mitológica, nascendo assim a alcunha “*startup* unicórnio”, que são *startups* que atingem o valor de mercado

de 1 bilhão de dólares antes de tornar-se uma empresa de capital aberto ou IPO (*Initial Public Offering*). Em maio de 2022 existiam mais de 1000 companhias que se enquadram na categoria de *startup* unicórnio e com avaliação combinada por volta de 3,671 trilhões de dólares (CB INSIGHTS, 2022). Além das *startups* unicórnio já existem as variantes decacórnio, empresas avaliadas em mais de 10 bilhões de dólares, e hectocórnios, avaliadas em mais de 100 bilhões de dólares (CB INSIGHTS, 2022). A lista de maiores *startups,* em valor avaliado, pode ser vista na Tabela 1.

Tabela 1 - Dez *startups* unicórnio melhor avaliadas em maio/2022

Startup	**Avaliação (bilhão de dólares)**	**País**	**Setor**
Bytedance	140	China	Inteligência Artificial
SpaceX	100,3	Estados Unidos	Aeroespacial
SHEIN	100	China	*E-commerce*
Stripe	95	Estados Unidos	*Fintech*
Klarna	45,6	Suécia	*Fintech*
Canva	40	Autrália	*Software*
Checkout.com	40	Reino Unido	*Fintech*
Instacart	39	Estados Unidos	Logística e entrega
Databricks	38	Estados Unidos	*Bigdata*
Revolut	33	Reino Unido	*Fintech*

Fonte: Adaptado de CB Insights (2022).

Devido a crescente importância de *startups*, metodologias próprias para *startups*, descritas na próxima seção, surgiram a fim de parametrizar processos que levem uma *startup* a ser bem sucedida.

2.2.1 *Metodologias para startups*

No universo da administração, sabe-se que para iniciar qualquer negócio é fundamental realizar o planejamento e um plano de negócios (DORNELAS, 2001). Para Pavani, Deutscher e López (2000), a formulação de um plano de negócios realiza-se pensando e consolidando em um único documento pontos como a realidade do empreendimento, perspectivas, estratégias. A partir do estudo do documento, consegue-se um gerenciamento melhor das operações, estabelecer novas estratégias, aumentar a eficiência da gestão, aumentar o grau de confiabilidade e melhorar a imagem da empresa. O plano de negócios, no entanto, popularizou-se como uma ferramenta de captação de recursos junto a capitais de risco ou investidores anjo, em que o plano de negócios é analisado a fim de definir o potencial

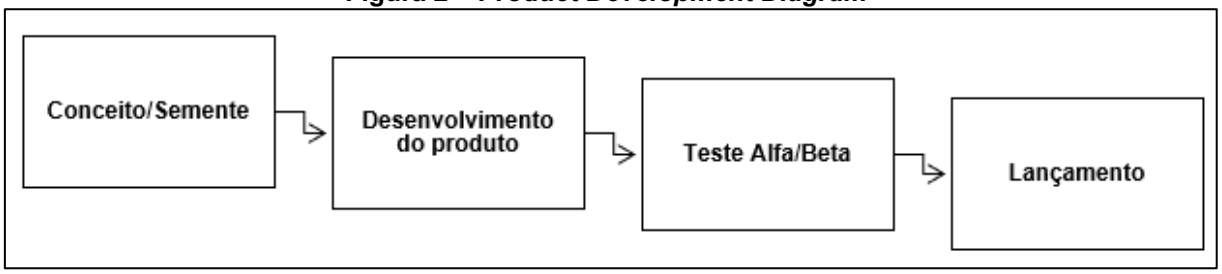
do empreendimento (DORNELAS, 2001). Um investidor anjo é uma pessoa física que realiza um investimento com capital próprio em uma empresa que acredita ter um grande potencial de crescimento.

Definir um plano de negócio, reunir uma equipe, obter investidores, desenvolver um produto e vendê-lo, foi o que predominou durante décadas na criação de novas empresas. Mas não é uma garantia de sucesso, podendo resultar em um revés fatal no plano de negócios perfeito do empreendimento, principalmente tratando-se de uma *startup* (BLANK, 2013). *Startups* não são versões menores de grandes companhias, então o modelo de negócios praticado pelas grandes companhias não é apropriado para *startups* (BLANK, 2010).

Blank (2013) explica que modelo centrado em produto, conhecido como *Product Development Model,* surgiu no início do século XX e desenvolveu-se na era da manufatura. Sendo adotado pela indústria de bens de consumo, o modelo espalhou-se para a indústria de tecnologia ao final do século, tornando-se parte da cultura de *startup*. Muitas vezes o modelo é utilizado não só para gerenciar o desenvolvimento do produto, como também um meio para encontrar clientes.

A iteração inicial do *Product Development Model* pode ser vista na Figura 2.

Figura 2 – *Product Development Diagram*



Fonte: Adaptado de Blank (2013).

As etapas são definidas por Blank (2009) como:

- Conceito/Semente – A etapa inicial consiste em definir ideias chave e desenvolver um plano de negócios. A ideia do que será o produto deve ser definida;
- Desenvolvimento do produto – Cada setor da companhia trabalha na ideia definida a fim de desenvolver um produto comercializável e funcional;
- Teste Alfa/Beta – O desenvolvimento testa as soluções com um grupo de usuários externos a fim de detectar artefatos passiveis de melhora;
- Lançamento – Primeira entrega do produto ao usuário final e final da iteração, em que o plano de negócios é colocado à prova.

Blank (2013) afirma que o modelo pode funcionar para mercados bem definidos onde a base da competição é conhecida e os clientes já estão identificados, cenário em que poucas *startups* se enquadram dada a condição de extrema incerteza.

O ultrapassado *Product Development* quando seguido à risca por uma promissora *startup* com um plano de negócios impecável, investimento farto, uma equipe fantástica pode resultar no desenvolvimento de um ótimo produto pelo qual ninguém se interessa, condenando a *startup* por se focar demais no produto e não no consumidor alvo (RIES, 2012). Como o nome sugere, *Product Development* é um modelo de desenvolvimento de produto não um modelo de *marketing*, vendas, aquisição de clientes ou gestão de finanças e muitas vezes pode não ser benéfico nem para o próprio produto, podendo causar um desvio de foco do que realmente interessa ao cliente. Ainda assim o modelo é usado por *startups* para definir todos os aspectos, seja relacionado ao desenvolvimento ou não (BLANK, 2009).

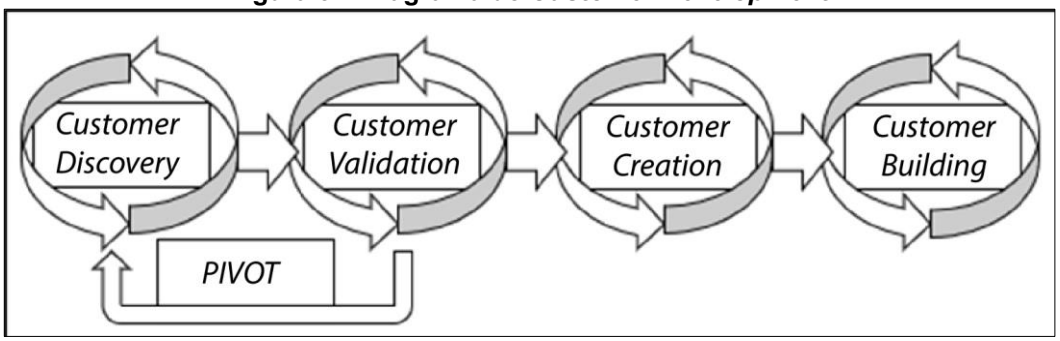
Blank (2013) explica que os principais problemas do método *Product Development* são: falta de interação com o consumidor alvo, falta de soluções para desenvolvimento contínuo e más opções de *marketing* e vendas. Para adequar-se melhor ao crescente mundo das *startups,* metodologias modernas de planejamento e gerenciamento de negócio, alternativas ao *Product Development, e* focadas no consumidor começaram a surgir.

2.2.2 *Customer Development*

A metodologia *Customer Development*, foi conceituada por Steve Blank em 2005. Blank (2013) diz que *Customer Development* parte de uma premissa simples: aprender e descobrir quem são os clientes iniciais da *startup* e em quais mercados eles se localizam. Este processo deve ser separado do desenvolvimento de produto. A soma das atividades de descoberta de mercado e desenvolvimento do produto constituí o *Customer Development*.

Blank (2013) detalha quatro etapas que constituem o *Customer Development*, chamadas por ele de “quatro passos para epifania” e são designadas para sanar os problemas do *Product Development*. Os quatro passos podem ser vistos na Figura 3.

Figura 3 – Diagrama de *Customer Development*



Fonte: Adaptado de Blank (2013).

O objetivo do passo *Customer Discovery* é descobrir quem são os clientes para o seu produto e que problema seu produto pode solucionar. Consiste em validar as hipóteses de clientes, produto e do problema definidas no plano de negócio. Para essa tarefa é necessária uma pesquisa de campo ampla, identificando em diversos grupos os potenciais clientes e se seu produto pode ser uma alternativa a problemas identificados. Categorizado e definindo um padrão de potenciais clientes, não é a função da equipe de *Customer Development* obter requisitos para o produto com os clientes. Na *startup* os fundadores e o time de *Product Development* são responsáveis por definir o produto e a função do time de *Customer Development* é identificar se existe clientes e um mercado para este produto.

A etapa de *Customer Validation* tem como objetivo definir um roteiro de vendas repetível para que as equipes de vendas e *marketing* o sigam posteriormente. Neste caso, o roteiro de vendas se torna válido quando testado em campo com sucesso. Apenas pode-se prosseguir para o passo seguinte quando se obtém um processo de vendas lucrativo e uma base de clientes recorrentes. Esta etapa, além de ser fundamental para obter *feedback* dos primeiros clientes, também prova que foi possível encontrar um grupo de clientes e um mercado com boa receptividade ao produto. *Customer Validation* juntamente com *Customer Discovery* validam o plano de negócios, comprovando dados essenciais para o empreendimento progredir.

O *Pivot* caracteriza-se como um ponto de retorno ao passo anterior quando a *Customer Validation* não foi capaz de obter um processo de vendas lucrativo e uma base de clientes recorrentes. Assim, o *Customer Discovery* e posteriormente um no *Customer Validation* deve ser realizado até que se obtenha resultados aceitáveis.

Customer Creation fundamenta-se no sucesso das vendas iniciais da *startup*, consequência da etapa *Customer Validation* bem-sucedida. O objetivo é criar uma demanda de usuário e direcionar esta demanda ao plano de vendas já consolidado.

Esta fase é caracterizada por ações do setor de *marketing* a fim de obter novos clientes.

O último passo, chamado *Company Building*, é onde ocorre a transição do informal time de *Customer Development* voltado para descobertas de mercado, público e validação de negócios para setores formais como vendas, *marketing* e *Business Development*. A tarefa é utilizar e explorar as informações obtidas nas etapas anteriores a fim de definir objetivos que guiarão a companhia.

As soluções apontadas por Blank (2013) para as falhas do *Product Development* se tornaram a *Customer Development.* Essa metodologia se tornou revolucionária para *startups*, fomentando um cenário em que os usuários tem participação ativa no desenvolvimento e validação do produto e do modelo de vendas. Iniciou-se assim o movimento *Lean Startup* (*startup enxuta*, em tradução livre), sendo os quatro passos da *Customer Development* um dos três pilares da metodologia *Lean Startup* (PRIHODKO, 2018).

2.2.3 *Lean Startup*

O movimento *Lean Startup* teve seu início com Steve Blank e a *Customer Development*, que procurava tornar o processo de criação da empresa menos arriscado (SHEPHERD; GRUBER, 2021). Afirmando que as funções de vendas e *marketing* de uma *startup* deveriam ter tanta importância quanto as de engenharia e desenvolvimento, e, portanto, mereciam uma metodologia igualmente rigorosa para orientá-las, nascendo assim a *Customer Development* (RIES, 2012). Porém, o termo *Lean Startup* surgiu em 2008 quando Eric Ries escreveu o termo e o definiu pela primeira vez em seu *blog* e posteriormente publicou o livro "*Lean Startup*", se aprofundando mais na metodologia. Ries (2008) inicialmente define *Lean Startup* como enxuta, no quesito de conseguir operar com gastos extremamente reduzidos e utilizando ideias e métodos simplificados.

O movimento da *startup* enxuta se expandiu para além das origens, em que empreendedores aplicavam seus conceitos em todos os setores possíveis, Ries (2012) dispôs cinco princípios de uma *startup* enxuta:

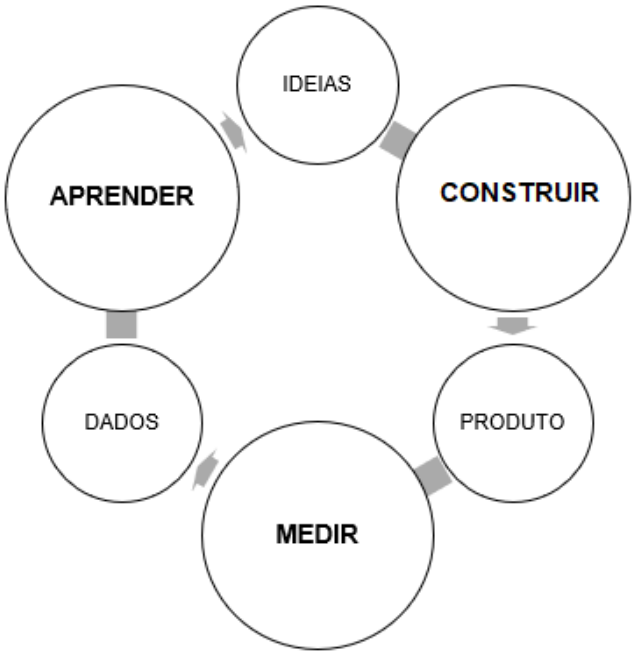
- Empreendedores estão por toda a parte – Um empreendimento pode surgir e se desenvolver em qualquer lugar e inclui qualquer *startup* que se encaixe

na definição de instituição humana projetada para criar novos produtos e serviços sobre condições e extrema incerteza.

- Empreender é administrar – *Startup* é uma instituição, não um produto, e deve ser gerida como tal.
- Aprendizado validado – *Startups* não existem apenas para gerar lucro, também existem para aprender com base na experimentação de seus métodos.
- Construir-medir-aprender – A atividade fundamental de uma *startup* é transformar ideias em produtos, medir o *feedback* dos clientes, e, então, aprender se o método é funcional e deve ser preservado ou um novo deve ser desenvolvido e validado;
- Contabilidade para inovação – Para melhorar os resultados, a *startup* deve medir o progresso, definir marcos e priorizar o trabalho.

Após definir os princípios mínimos esperado para um *startup* enxuta, Ries (2012) estabelece as fases, ou pilares, que compõe a metodologia *Lean Startup* apresentada na Figura 4.

Figura 4 – Ciclo de *feedback* Construir-Medir-Aprender



Fonte: Adaptado de Ries (2012)

A seguir se descreve brevemente cada uma das fases:

- Construir – Nesta fase a empresa deve construir, com base nas ideias, um produto mínimo viável e apresentá-lo aos clientes, comumente através de metodologias ágeis.

- Medir – Por meio do produto, MVP, disponibilizado aos clientes, cabe a empresa medir, quantificando o *feedback* dos usuários com o intuito gerar dados válidos a respeito do produto. Nessa parte o *Customer Development* é aplicado.
- Aprender – Os dados validados obtidos por meio do *feedback* dos usuários são utilizados para compreender e aprender, gerando novas ideias para o próximo ciclo do produto. Toda experiência obtida por meio do ciclo anterior será utilizada no novo ciclo.

Nessa Seção foram abordados os temas *startup* e as metodologias *Product Development, Customer Development* e *Lean Startup*. A definição *startup é* uma empresa que se desenvolve em um ambiente de incerteza, e a importância que esse tipo de companhia possui no mercado global justificam a existência de metodologias que facilitem o desenvolvimento deste tipo de negócio. Considerando as metodologias apresentadas, a *Product Development* foi abordada para compreender como uma *startup* não pode se basear apenas no produto e seu desenvolvimento. Isso levou ao desenvolvimento de metodologias mais abrangentes como a *Customer Development,* que se tornaria um dos três pilares da metodologia *Lean Startup*, a mais conceituada metodologia formal com foco em *startups.*

2.3 Metodologias ágeis para o desenvolvimento de software

Pode-se afirmar que um dos maiores desafios para organizações do século XXI é a capacidade de adaptar-se as mudanças rápidas e imprevisíveis melhor que seus concorrentes (BOHEM, 2008). Práticas de desenvolvimento de *software* utilizadas por organizações são constantemente desafiadas pela evolução do ambiente de negócios, especialmente onde a incerteza prevalece (KTATA; LÉVASQUE, 2009). As organizações precisam lidar com a volatilidade do ambiente reestruturando seus métodos (ABRANTES; TRAVASSOS, 2011).

O conceito de agilidade no campo de desenvolvimento de *software* foi formalizado em 2001 com a criação da *Agile Alliance*, formada por veteranos da área de *software*, e a publicação do Manifesto para Desenvolvimento Ágil de *Software*, mais conhecido como Manifesto Ágil, que tem como objetivo “restaurar a credibilidade da palavra método” (AGILE MANIFESTO, 2001). Conboy e Fitzgerald (2004) explicam que o Manifesto Ágil procurou transmitir a visão da indústria, a fim de mudar o

paradigma do desenvolvimento de *software* por meio de seus doze princípios que garantem uma metodologia de desenvolvimento como sendo ágil.

O Manifesto Ágil e seus princípios foram bem recebidos por profissionais e acadêmicos por apresentar um trabalho pioneiro em compilar e entender a crítica às metodologias de desenvolvimento de *software* utilizadas nas décadas antecedentes (CONBOY; FITZGERALD, 2004). Metodologias ágeis se mostraram alternativas aos tradicionais modelos de cascata e sequencial no gerenciamento de projetos, normalmente utilizados no desenvolvimento de *software*. Auxiliam as equipes de desenvolvimento a lidar com a imprevisibilidade por meio de ciclos de trabalho iterativos conhecidos como *sprints* (MISKI, 2014).

Ktata e Lévasque (2009) explicam que o desenvolvimento ágil, ao invés de tentar entender todas as necessidades do usuário logo ao início do desenvolvimento, aposta em combinar planejamento de recursos e ciclos iterativos progressivos. Ao invés de otimizar o produto ao máximo antes de entregá-lo ao consumidor, utiliza-se do princípio de entregar apenas um *software* em funcionamento ao invés de envolver o cliente por meio de documentação. Interação e colaboração pessoal com o consumidor é promovida. Ao invés de analisar os riscos e incertezas empiricamente, utilizar-se de prototipagem para identificar estes pontos.

Métodos ágeis reinventaram o desenvolvimento de *software* incluindo o esforço de desenvolvimento no princípio de auto-organização da equipe de desenvolvimento, ou seja, engenharia de requisitos, gerenciamento de projetos, codificação e teste. Essa auto-organização é executada por uma equipe auto-organizada sem restrições hierárquicas. Além disso, o princípio de desenvolver apenas o essencial poupa esforços desnecessários e a refatoração é promovida entre as iterações dos ciclos de desenvolvimento (KTATA; LÉVASQUE, 2009).

A agilidade no desenvolvimento proporcionado por métodos ágeis é especialmente recomendada por metodologias de *startup* que pregam maior interação com o cliente. É apreciada por *startups* que atuam em cenários e condições que podem mudar a qualquer momento, pois as equipes que o empregam são capazes de mudar de direção com rapidez, permanecer ágeis, e ser altamente responsivas às mudanças nos requisitos de negócios do dono do produto (RIES, 2012).

Em *tech-startups*, metodologias ágeis definitivamente possuem benefícios em relação aos métodos não-ágeis tradicionais (YAU; MURPHY, 2013). Métodos ágeis possuem princípios sólidos, atenuando certos problemas enfrentados por empresas

(YAU; MURPHY, 2013), além de serem usados mundialmente como a melhor maneira de desenvolver, manter e dar suporte a software (SUTHERLAND, 2021). Uma metodologia ágil como o *Scrum* pode ser utilizada por uma *startup* na fase de desenvolvimento do produto e foi selecionada por ser uma metodologia formal, com atores, artefatos e papéis bem definidos e processos que podem ser utilizados no desenvolvimento de um produto. Esta metodologia é descrita na próxima seção.

2.3.1 *Scrum*

Scrum foi descrito pela primeira vez no paper "*The New New Product Development Game*" no *Harvard Business Review* em 1986 por Hirotaka Takeuchi e Ikujiro Nonaka, influenciados pelas melhores práticas de desenvolvimento da indústria japonesa à época (SUTHERLAND, 2021). Proposto como um conceito de desenvolvimento de produto onde o mesmo surge a partir de interação constante de uma seleta equipe multifuncional cujos membros trabalham juntos do início ao fim do desenvolvimento, sendo assim capazes de produzir melhores resultados (TAKEUCHI; NONAKA, 1986).

O *Scrum* não se trata de um método de desenvolvimento ou processo formal, mas uma ferramenta que se baseia em mais de 50 anos das melhores práticas de desenvolvimento de *software* (SUTHERLAND, 2021). A adoção do *Scrum* não garante a solução de todos os problemas, porém, se bem utilizada, pode proporcionar diversos benefícios em comparação a outras formas de se conduzir projetos (SABBAGH, 2013). O uso de metodologias que encorajam e auxiliam na flexibilidade e possuem grande tolerância a mudanças pode melhorar as chances de sucesso no desenvolvimento de sistemas em que o ambiente se mostra volátil (SCHWABER, 1996).

Scrum é ágil porque, assim como outros métodos, metodologias e *frameworks*, sua utilização deve seguir os princípios e valores do Manifesto Ágil (SABBAGH, 2013). Sutherland (2016) explica que além de seguir os princípios do Manifesto Ágil, o *Scrum* possui seus próprios fundamentos, ou pilares, que são: transparência, o trabalho em equipe e a colaboração. Sabbagh (2013) explica que o *Scrum* é simples e leve, onde não se gera ou aplica nada que não será utilizado. De acordo com Sabbagh (2013), o *Scrum* possuí elementos bem definidos, sendo três papéis, três eventos e três artefatos:

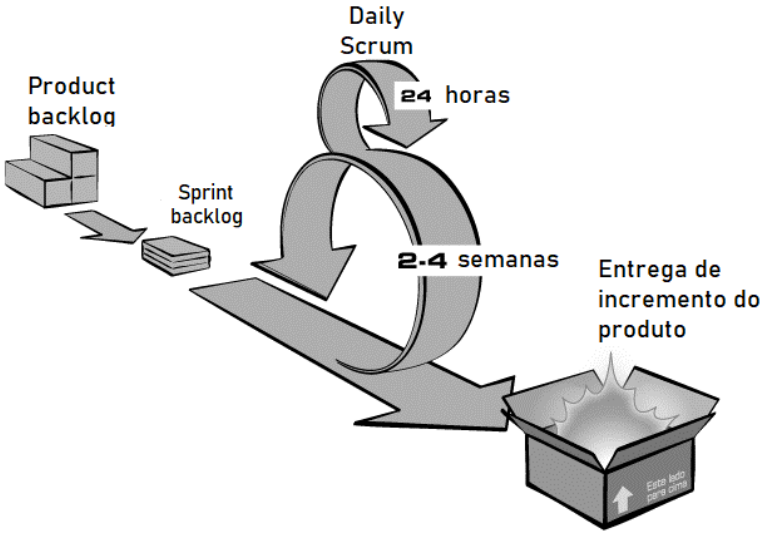
- Papéis - *Product Owner*, Time de Desenvolvimento (*Team*) e *ScrumMaster*;
- Eventos - *Sprint Planning*, *Daily Scrum*, *Sprint Review*;
- Artefatos - *Product Backlog*, *Sprint Backlog* e o Incremento do Produto.

Para começar com o *Scrum*, todos os envolvidos precisam entender seus papéis e responsabilidades assim como o fluxo operacional do *Scrum* (SUTHERLAND, 2021). Para tal é necessário explicar as funções que cada papel desempenha, explica Sabbagh (2013):

- *Product Owner* – É a pessoa responsável por garantir e maximizar, a partir do trabalho do Time de Desenvolvimento, o retorno sobre o investimento no produto para os clientes do projeto, definindo o produto e tomando as decisões de negócios relativas a seu desenvolvimento a partir das necessidades dos clientes (SABBAGH, 2013).
- Time de Desenvolvimento – Consiste em um grupo de desenvolvedores auto-organizado e multifuncional (KTATA; LÉVASQUE, 2009). O Time de Desenvolvimento, a cada *sprint*, tem propriedade, autoridade e responsabilidade para determinar tecnicamente como o produto será desenvolvido, planejar este trabalho e acompanhar seu progresso, de acordo com as prioridades definidas pelo *Product Owner* (SABBAGH, 2013).
- *ScrumMaster* – É o responsável por facilitar a adesão da equipe no *Scrum* (KTATA; LÉVASQUE, 2009). *ScrumMaster* também é responsável por potencializar o trabalho do Time de *Scrum*, composto pelos três papeis, utilizando-se de seu conhecimento de *Scrum*, habilidade de lidar com pessoas, técnicas de facilitação e outras técnicas (SABBAGH, 2013).

Os artefatos são enumerados e definidos por Sutherland (2021): *Product Backlog*, *Sprint Backlog* e Incremento do Produto. A Figura 5 ilustra, de maneira simplificada, uma iteração *Scrum*.

Figura 5 – Fluxograma Simplificado de Desenvolvimento com *Scrum*



Fonte: DesenvolvimentoAgil.com.br (2014)

O *Product Backlog* consiste em lista contendo o que se acredita que será desenvolvido pelo Time de Desenvolvimento no decorrer do projeto, como necessidades ou objetivos de negócios dos clientes do projeto, melhorias a serem realizadas no produto, correções de problemas, questões técnicas, pesquisas necessárias, entre outras (SABBAGH, 2013). A lista é atualizada o tempo todo e ordenada de acordo com a importância para os clientes do projeto e possui apenas o nível de detalhes que é possível de se ter (SABBAGH, 2013).

O *Sprint Backlog* é uma lista de itens selecionados do *Product Backlog* para serem aplicados no *sprint* seguinte (SABBAGH, 2013). A lista consiste em itens que *Product Owner* reconhece serem necessárias e que Time de Desenvolvimento julga capaz de realizar (SABBAGH, 2013).

O Incremento do Produto é o resultado do trabalho realizado pelo Time de Desenvolvimento nos itens selecionados do *Sprint Backlog* durante a *sprint* (SABBAGH, 2013). Consiste no conjunto de funcionalidades desenvolvidas que o *Product Owner* julgou necessárias adicionar ao produto (SABBAGH, 2013).

Os eventos do *Scrum* consistem no próprio ciclo de desenvolvimento, o sprint, e são (SABBAGH, 2013): *Sprint Planning*, *Daily Scrum* e *Sprint Review*. O *Sprint Planning* é o evento que ocorre no primeiro momento do primeiro dia do *sprint*, com a participação do *Product Owner* e de todo o Time de Desenvolvimento, em que é negociado e formulado o *Sprint Backlog* do atual *sprint*).

A *Daily Scrum* é uma reunião curta, realizada diariamente, com o objetivo de promover a comunicação e integração entre o Time de Desenvolvimento contextualizando todos os membros sobre o trabalho realizado para cada um durante o dia de trabalho anterior e os planos para o dia atual, além de informar ao *ScrumMaster* os impedimentos no fluxo de trabalho.

Por fim, *Sprint Review* é uma reunião realizada ao final do *sprint* onde o Time de Desenvolvimento e *Product Owner* trabalham em conjunto, com a facilitação do *ScrumMaster*, para demonstrar o trabalho realizado durante o *sprint* a clientes e demais partes interessadas afim de obter um *feedback* dos incrementos realizados, assim incrementando o *Product Backlog.*

2.4 *Framework de Domínio*

Os *frameworks* são uma técnica de reutilização orientada a objetos que compartilham características com diversas outros métodos de reutilização (JOHNSON, 1997). De acordo com Johnson (1997), as definições de *framework* variam, mas a mais comum é um *design* reutilizável de parte ou todo de um sistema que são representadas por um conjunto de classes abstratas e na forma em como elas interagem, um *design* em grande escala de como um programa pode ser decomposto. Para Fayad e Schmidt (1997) *frameworks* são aplicações orientadas a objetos reutilizáveis e semicompletas que podem ser especializadas para produzir aplicações.

Sobre o a definição de *framework* no domínio de produto de *software*:

> Um *framework* descreve a arquitetura de um sistema orientado a objetos; os tipos de objetos nele e como eles interagem. Ele descreve como um tipo específico de programa, como uma interface de usuário ou software de comunicação em rede, é decomposto em objetos. Ele é representado por um conjunto de classes (geralmente abstratas), uma para cada tipo de objeto, mas os padrões de interação entre os objetos fazem parte do *framework* tanto quanto as classes. (JOHNSON, 1997, p. 5).

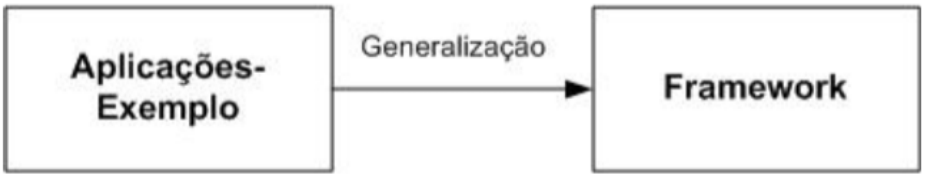
Galvão (2020) relata que os *frameworks* podem ser classificados quanto a sua reusabilidade e seu domínio. Yassin e Fayad (2000 apud Galvão, 2020) afirmam que *frameworks* podem ser classificados, quanto a reusabilidade, em três categorias, caixa branca, preta ou cinza. De acordo com Matos (2008 apud Galvão, 2020) *frameworks* caixa-branca ou *white-box* fundamentam-se em ligações dinâmicas e mecanismos de herança pertencentes a orientação a objetos. Maldonado et al. (2001 apud Galvão,

2020) classifica como caixa-preta ou *black-box frameworks* que permitem a extensão por meio de interfaces para componentes e define caixa-cinza (*gray-box*) como mais extensível e flexível que as caixas branca e preta, ocultando informações que não são pertinentes ao desenvolvedor.

O desenvolvimento de um *framework* de domínio decorre do processo de aprendizado de um domínio e é obtido a partir do desenvolvimento de aplicações-exemplo ou do estudo de aplicações já desenvolvidas (JOHNSON, 1993, apud MATOS; FERNANDES, 2007, p. 8).

Para Matos e Fernandes (2007) a abstração do domínio, que é o próprio *framework*, é obtida através da generalização de casos concretos, que possuem aspectos semelhantes às aplicações-exemplo, como pode ser visto na Figura 6.

Figura 6 – Processo de generalização

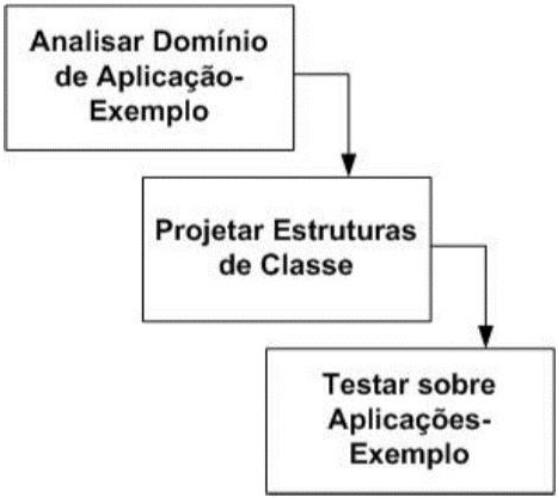


Fonte: Matos e Fernandes (2007).

O processo de desenvolvimento de Johnson (1993, apud MATOS; FERNANDES, 2007, p. 9), como pode ser visto na Figura 7, é composto pelas seguintes etapas:

- Analisar o Domínio de Aplicação-Exemplo – assimilar as abstrações conhecidas e exemplificar programas aos quais o *framework* poderia originar.
- Projetar Estruturas de Classe – Projeta-se a estrutura de classes de modo que possa ser especializada para abranger as aplicações-exemplo, construindo o *framework*.
- Testar sobre Aplicações-exemplo – usa-se o *framework* projetado no desenvolvimento de aplicações-exemplo, validando o *framework*.

Figura 7 – Processo baseado em análise segundo Johnson



Fonte: Matos e Fernandes (2007).

Johnson (1997) afirma que no mínimo três aplicações-exemplo devem ser selecionadas para a construção de um *framework* de domínio. A primeira versão de um *framework* usualmente deve ser capaz de implementar as aplicações-exemplo e então ser utilizado para construir novas aplicações. As futuras aplicações apresentarão os pontos fracos do *framework* que então devem ser aprimorados.

Pree (1995, apud Galvão, 2020) afirma que no processo de implementação de um *framework* é importante identificar pontos de estabilidade (*fronzen spots*) e flexibilidade (*hot spots*) e definindo-os como:

- Pontos de estabilidade – Aspectos comuns a todas as aplicações do domínio;
- Pontos de flexibilidade – Características e comportamentos específicos a cada aplicação.

Compreender o processo de criação de um *framework* de domínio é compreender o próprio domínio. O processo de Johnson (1993) fornece ferramentas para realizar essa análise e foi utilizado neste trabalho com a finalidade de definir e compreender um domínio no qual metodologias ágeis e de *startup* foram aplicadas no desenvolvimento do *framework*.

2.5 Considerações finais do capítulo

Este capítulo apresentou o conteúdo utilizado como referencial teórico no desenvolvimento deste trabalho. Primeiramente discorreu sobre os processos de *design* e as motivações do *Design Thinking*. As ideias abordadas no capítulo podem

fornecer parâmetros para a análise de requisitos de um produto a ser desenvolvido, testado e validado com constante interação e *feedback* do cliente alvo.

Posteriormente conceituou *startup*, apresentou sua importância e elucidou que existem metodologias próprias para *startups*, além de explicar os principais pontos da *Lean Startup* e *Customer Development*. Metodologias para *startups* ajudam o empreendedor a quantificar pontos relevantes do seu plano de negócio, deixando de lado situações hipotéticas e partindo para um planejamento empírico a fim de comprovar suas hipóteses. Metodologias para *startups* se mostram generalistas o suficiente para serem aplicadas nos mais diversos ramos em que uma *startup* pode atuar, porém, não tem como foco o desenvolvimento de *software*.

A respeito de metodologias ágeis, este capítulo explicou a importância das alternativas ágeis aos antigos métodos tradicionais, conceituou Método Ágil e seus princípios definidos pelo Manifesto Ágil. Escolheu como foco para este trabalho o *Scrum*. O *Scrum* e o que o compõe, quando analisados, mostraram imediatamente fortes características da metodologia, como papéis e fases bem definidas. *Scrum* se mostra uma alternativa ao tradicional modelo sequencial ou de cascata e apresenta uma solução que pode ser aplicada no desenvolvimento de qualquer produto.

Por fim, foi descrito o que são *frameworks*, suas características e classificações tal como o processo de criação de um *framework* de domínio. Metodologias de *Frameworks*, como o processo de desenvolvimento de Johnson (1993, apud MATOS; FERNANDES, 2007, p. 9), abordam apenas pontos como compreensão do domínio, definição de escopo e desenvolvimento. Essa abordagem em casos como desenvolvimento de um produto de *software* pode levar aos mesmos problemas que a metodologia *Product Development*. Dessa forma, metodologias para *startup,* focadas em desenvolvimento de produto e com foco em mercado, em conjunto com metodologias ágeis podem contribuir com um desenvolvimento objetivo e focados em criar um produto funcional por meio de etapas formais de desenvolvimento e fornece recursos para aprimorar iterativamente e interativamente o *software* desenvolvimento.

3 METODOLOGIAS ÁGEIS E DE *STARTUP* NA CRIAÇÃO DE UM *FRAMEWORK* DE DOMÍNIO

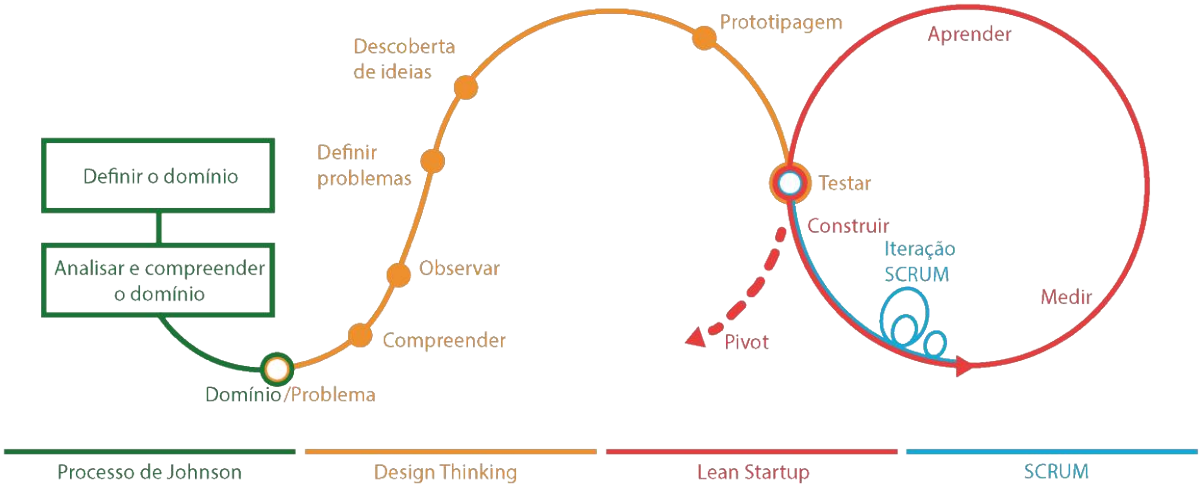
Este capítulo é designado para detalhar os passos que foram seguidos no uso de Metodologias Ágeis e de *startup* na criação de um *framework* de domínio. A Seção 3.1 relata a metodologia usada para definir as etapas que constituem o desenvolvimento do *framework* utilizando metodologias ágeis e de *startup*. A Seção 3.2 define e apresenta o domínio usado para a criação do *framework*. A Seção 3.3 relata como a análise do domínio das aplicações-exemplo foi realizada. A Seção 3.4 aborda como foi realizado a integração de metodologia de *startup* (*Design Thinking* e *Lean Startup*). A Seção 3.5 relata o planejamento do desenvolvimento do *framework* utilizando a metodologia ágil *Scrum*. A Seção 3.6 apresenta as considerações finais a respeito do que foi abordado no capítulo.

3.1 Metodologia Proposta para o Desenvolvimento do *Framework* de Domínio

O desenvolvimento deste trabalho está dividido em quatro etapas, sendo que as duas primeiras estão relacionadas as características específicas para o desenvolvimento de *frameworks* de domínio que foram relatados por Johnson (1993, apud MATOS; FERNANDES, 2007, p. 9). A primeira etapa é usada para Definir o Domínio de aplicação-exemplo a qual se deseja estudar e compreender aplicações-exemplos. A segunda etapa é Analisar e Compreender o domínio usando as aplicações-exemplos e criando o diagrama de caso de uso para identificar suas funcionalidades e relação com o mundo exterior.

A terceira etapa refere-se à idealização do *framework* utilizando uma junção de metodologias ágeis e de *startup* por meio de uma combinação das metodologias *Design Thinking* e *Lean Startup*. Por fim, a quarta etapa refere-se ao planejamento e construção do *framework* de domínio utilizando a metodologia ágil *Scrum* e seus processos. A Figura 8 exibe as etapas da metodologia usada.

Figura 8 – Integração das Metodologias *Design Thinking*, *Lean Startup* e *Scrum*



Fonte: Adaptado de Grossman-Kahn e Rosensweig (2012).

Pode-se observar que a metodologia proposta consiste em uma integração dos conceitos gerais definidos nas metodologias de *Design Thinking* e suas etapas descritas por Plattner, Meinel e Leifer (2011), *Lean Startup*, descrito por Ries (2012) e *Scrum*, descrito por Sabbagh (2013), bem como possui como entradas as características de *frameworks*. O fluxo originado da junção das metodologias contempla um número mínimo de 9 subetapas obtidas por meio da combinação das metodologias *Design Thinking* e *Lean Startup* com o processo de construção do produto sendo guiado por iterações de *Scrum*, as quais são explicadas nas próximas seções.

3.2 Definir o domínio

Para a definição do domínio foram estudadas necessidades iniciais de uma *startup* baseada em desenvolvimento de produto. Artefatos como suporte técnico e atendimento ao cliente estão intrinsecamente ligados ao próprio produto e a maneira como o mesmo chega ao cliente. Com o objetivo de facilitar a entrega do produto da *startup* ao consumidor, foi escolhido como domínio: Atendimento Automatizado (AA) ao cliente. A justificativa da escolha do domínio dá-se por meio dos conceitos de Blank (2013) de que o produto de uma *startup* acontece por meio da descoberta do cliente, desenvolve-se com o auxílio do usuário e é constantemente validada pelo mesmo e os *chatbots* alcançam os clientes por meio dos canais de comunicação.

Um *framework* que possibilite a rápida criação de canais de comunicação automatizados para os mais diferentes tipos de produtos torna mais ágil a

implementação de um MVP (*Minimum Viable Product*) ao mercado, fornecendo suporte técnico e coletando *feedback*.

3.3 Analisar e compreender o domínio de aplicações-exemplo

Definido o domínio, de acordo com o processo de desenvolvimento de Johnson (1993, apud GALVÃO, 2020, p. 34), foi necessário selecionar no mínimo três aplicações-exemplo no domínio para análise. As aplicações-exemplo selecionadas consistem em soluções de AA fornecidos como serviço por diferentes empresas.

O critério para a seleção das aplicações-exemplo foram os resultados de uma busca na internet de soluções de atendimento automatizado. Para não ter problema em relação a divulgação do nome das empresas, este trabalho as nomeou como AA1, AA2 e AA3. Estas foram analisadas com o intuito de identificar e compreender suas funcionalidades. As funcionalidades levantadas são:

- Multicanal – As soluções podem atender o cliente por meio de diferentes canais de comunicação como *WhatsApp*, *webchat*, *Messenger* e *Instagram*;
- Configurável – As aplicações-exemplos podem ser configuradas para efetuar diferentes tipos de atendimentos;
- Coleta de dados – Podem solicitar, validar e armazenar dados do cliente a fim de efetuar algum processo;
- Replicável – Múltiplas instâncias das aplicações podem ser criadas e atender diferentes demandas;
- Reativo – A ferramenta responde adequadamente de maneira automática;
- Disponibilidade – A ferramenta interage automaticamente, independentemente do horário, a qualquer mensagem enviada por meio do canal;
- Árvore decisória – As possibilidades de interação e opções são pré-configuradas na ferramenta, mas a direção tomada nas interações depende das escolhas e entradas do cliente em contato com o atendimento, ramificando de acordo com opções selecionadas.

3.4 Integração das metodologias de *startup*

O processo de abordagem do domínio por meio da integração das metodologias *Design Thinking* e *Lean Startup* iniciou-se adotando o domínio de AA ao cliente como o problema a ser solucionado.

Utilizado o desenvolvimento do *framework*, a etapa *Customer Discovery* da metodologia de *startup Customer Development*, conceituada por Blank (2013), ajudou a entender qual a relação do domínio escolhido e a descoberta de potenciais clientes que utilizarão a solução, o *framework*, para a criação de soluções de AA.

Como abordado na justificativa da escolha do domínio na Seção 3.2, a criação do produto inicia-se com a descoberta do cliente e desenvolve-se com o auxílio e validação do cliente. Dessa forma, pode-se entender as motivações que uma *startup* a utilizar uma solução que possibilite um canal de comunicação seja estabelecido tão rápido quanto a criação do MVP e possa ser específico a ponto de atender o atual e futuros MVPs de futuras incursões de *startup*.

Com as considerações sobre o problema e a necessidade de uma solução constatada pode-se iniciar a elaboração do *framework* de acordo com as subetapas da metodologia *Design Thinking* e *Lean Startup* representada na Figura 8.

3.4.1 *Compreender*

A própria compreensão do domínio caracteriza-se como a compreensão do problema. Este processo se mostrou bem amparado pela seleção e caracterização de aplicações-exemplo concretas como definido por Johnson (1993, apud GALVÃO, 2020, p. 34). As características isoladas ajudaram a visualizar como o problema pode ser abordado, ajudando a identificar comportamentos e definir objetivos, requisitos e expectativas. Com diagrama de caso de uso resultante da etapa de Analisar e Compreender o domínio, como pode-se observar na Figura 10, percebeu-se comportamentos do ambiente que respaldam características isoladas, por exemplo, o processo de contato entre o cliente e o AA, sempre se inicia com a ação por parte do cliente entrando em contato com o canal coberto pelo AA, ou solicitando contato do mesmo.

Também se percebeu a diversidade de produtos e serviços em que o AA pode ser aplicado. Através de uma conversa segmentada em etapas entre o cliente e o atendimento, é possível realizar tarefas como coleta de dados, pesquisa, vendas e

suporte técnico. O entendimento do problema por meio da perspectiva de *Design Thinking*, como definido por Mueller-Roterberg (2018) reforçou a importância de se focar em um AA que possa atender qualquer situação.

Atendo-se ao processo de compreensão do problema como definido por Plattner, Meinel e Leifer (2011), foram firmados os seguintes parâmetros a respeito do escopo do problema:

- Problema: atendimento automatizado (AA) ao cliente;
- Objetivo: fornecer um *framework* que possibilite a rápida implantação de canais de AA para consumidores de produtos e serviços diversificados de *startups*;
- Requisitos: uma solução criada por meio do *framework* deve atender clientes automaticamente através de diversos canais e pode ser customizado para realizar atendimento de qualquer serviço que a *startup* dispor;
- Expectativas: com pouco ou nenhum desenvolvimento o *framework* fornece origem a um AA para qualquer MVP desenvolvido.

Compreendidos os parâmetros, pôde-se iniciar o processo de observação do problema levando em consideração os artefatos analisados.

3.4.2 *Observar e Definir o Problema*

O processo de observação também se mostrou bem amparado pela seleção e caracterização de aplicações-exemplo concretas ajudando a compreender o que se dispõe de soluções pagas disponíveis no mercado. Sob a definição de Ries (2012) do que é uma *startup* entende-se que nem sempre existe a possibilidade de investir financeiramente em uma solução de atendimento. Dessa forma, um *framework* que possibilite que a solução seja desenvolvida domesticamente pela *startup* como componente do MVP pode auxiliar nos processos posteriores ao desenvolvimento.

Como citado anteriormente, a própria compreensão do domínio caracteriza-se como a compreensão do problema, o estudo do *framework* e definição do domínio, abordando como o problema permite que o tópico tratado esteja bem definido, como o contemplado neste trabalho, o atendimento automatizado ao cliente.

3.4.3 *Descoberta de ideias*

A partir das análises de Seções anteriores e informações obtidas por meio de aplicações-exemplo, algumas ideias surgiram e foram ponderadas e pré-modeladas. A mais promissora delas foi em relação aos dados obtidos a partir dos pontos em comuns das aplicações-exemplo e que pode ser visualizado na Figura 10.

O conceito foi validado por meio de um estudo a respeito do fluxo de informação elaborado por meio de um diagrama de atividades de um produto do *framework* visando atender as aplicações-exemplo. A partir do estudo percebeu-se que a informação principal que passa pela aplicação e é fornecida pelo usuário são as mensagens. A partir da descoberta de ideias é possível iniciar o planejamento da construção do *framework* de domínio.

3.5 Integração da metodologia *Scrum*

O desenvolvimento iniciou-se com a criação e teste de um protótipo funcional com a finalidade de provar os pontos identificados nas Seções anteriores como parte do *Design Thinking*.

O desenvolvimento do *framework* como um produto por meio de iterações de *Scrum* iniciou-se firmando os elementos de acordo com a definição de Sabbagh (2013). O *Scrum* foi selecionado como metodologia ágil pois possui grande tolerância a mudanças e pode melhorar as chances de sucesso no desenvolvimento de sistemas em que o ambiente se mostra volátil (SCHWABER, 1996). O *Scrum* é aplicado em iterações durante o processo de construção de produto da metodologia *Lean Startup*.

Como o *framework* foi desenvolvido pelo autor deste trabalho, alguns papéis puderam ser abstraídos ou reduzidos do *Scrum*. Detalhes de como a abstração de papéis foi realizada pode ser visualizado no Quadro 1.

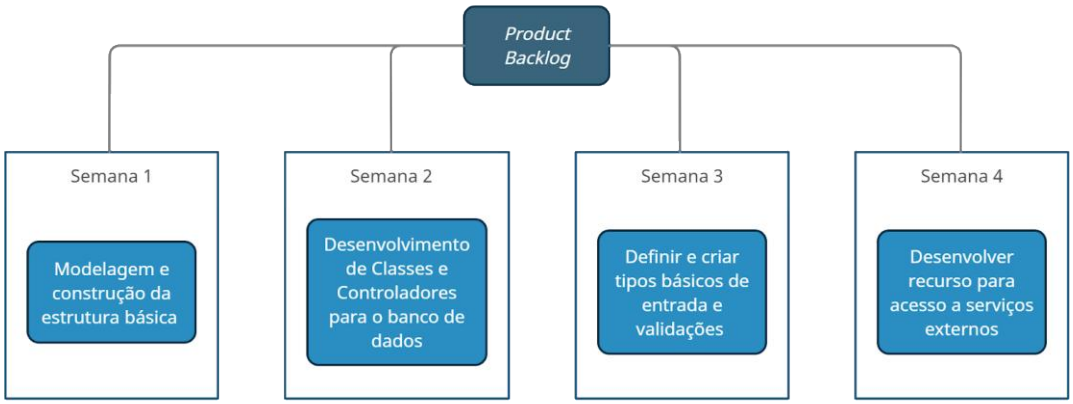
Quadro 1 - Papéis do *Scrum*

Papel	Atribuições
1. *Product Owner*	O *ProductOwner* foi abstraído ao entendimento das necessidades do cliente e compromisso que os requisitos definidos na Seção 3.4.1 sejam seguidos.
2. Time de Desenvolvimento	O time de desenvolvimento foi composto por um único desenvolvedor, neste caso responsável por realizar as tarefas definidas no *sprint*. Como todo o trabalho foi desenvolvido de maneira individual, um time de apenas um indivíduo foi utilizado.
3. *ScrumMaster*	O *ScrumMaster* foi abstraído à autoanálise do desenvolvedor e o planejamento das *sprints* de maneira conservadora, de modo que tudo possa ser cumprido e ocorra como planejado.

Fonte: Autoria própria (2022).

Com os papéis definidos foi possível criar o *Product Backlog* com as incrementações necessárias para que o protótipo desenvolvido atenda todos os requisitos e objetivos definidos na Seção 3.4.1. O *Product Backlog* idealizado pode ser visto na Figura 9.

Figura 9 – *Product Backlog* do processo de criação do *framework*



Fonte: Autoria própria (2022).

O desenvolvimento necessário para transformar o protótipo no *framework* foi dividido em 4 semanas e 4 *sprints*, cada uma delas com foco no desenvolvimento de uma característica específica do *framework*.

Um *Sprint Planning* foi realizado ao início de cada semana a fim de definir como os requisitos seriam desenvolvidos. O *Sprint Backlog* não se mostrou necessário no processo de transformar o protótipo em *framework*, pois os requisitos se mostraram claros desde o início. Porém, para futuros incrementos que vão além do escopo inicial, o artefato deve ser criado ao final de cada *sprint*.

3.6 Considerações finais do capítulo

Este capítulo dedicou-se a descrever detalhadamente a metodologia utilizada no desenvolvimento de um *framework* de domínio utilizando metodologias ágeis e de *startup*, tal como justificar as decisões tomadas.

Do processo de escolha, análise do domínio ao *backlog* de desenvolvimento do *framework* todos os quesitos abordados foram escolhidos visando atender as definições de *Design Thinking* e *Customer Development*, metodologias conhecidas e utilizadas por *startups*. Dessa forma, o processo de criação de um *framework* utilizando os conceitos definidos nesse capítulo e o produto resultante do *framework* podem mostrar-se úteis a *startups* que se dediquem a utilizá-los.

4 RESULTADOS

Este capítulo tem a finalidade de detalhar os resultados que foram atingidos no desenvolvimento do trabalho. A Seção 4.1 ilustra alguns artefatos produzidos em relação aos modelos de análise que ajudam a compreensão e entendimento do fluxo do *framework* de domínio. A Seção 4.2 relata o desenvolvimento da extensão do *framework* como uma base para o desenvolvimento incremental. A Seção 4.3 descreve a primeira *sprint* de implementação do *framework* utilizando *Scrum*, onde foi preparada a base do desenvolvimento do *software* e definidas as ferramentas utilizadas no desenvolvimento. A Seção 4.4 apresenta a *sprint* da semana 2 em que foram modeladas as principais classes e tabelas do banco de dados. A Seção 4.5 descreve a *sprint* da semana 3 na qual foram formalizados os métodos utilizados pela aplicação para processar e validar as entradas do usuário, tal como definir os tipos básicos de entrada e validações desenvolvidas. A Seção 4.6 relata o processo necessário para adaptar a aplicação para se tornar capaz de consumir e tratar acesso a recursos externos, como APIs. Também apresenta o final da aplicação do *framework* na qual a solução foi utilizada na criação de um serviço de atendimento que supra os objetivos e características exigidos. A Seção 4.7 apresenta uma análise do resultado e da aplicação desenvolvida e como as metodologias ágeis e de *startup* contribuíram para o desenvolvimento. A Seção 4.8 apresenta as considerações finais do capítulo.

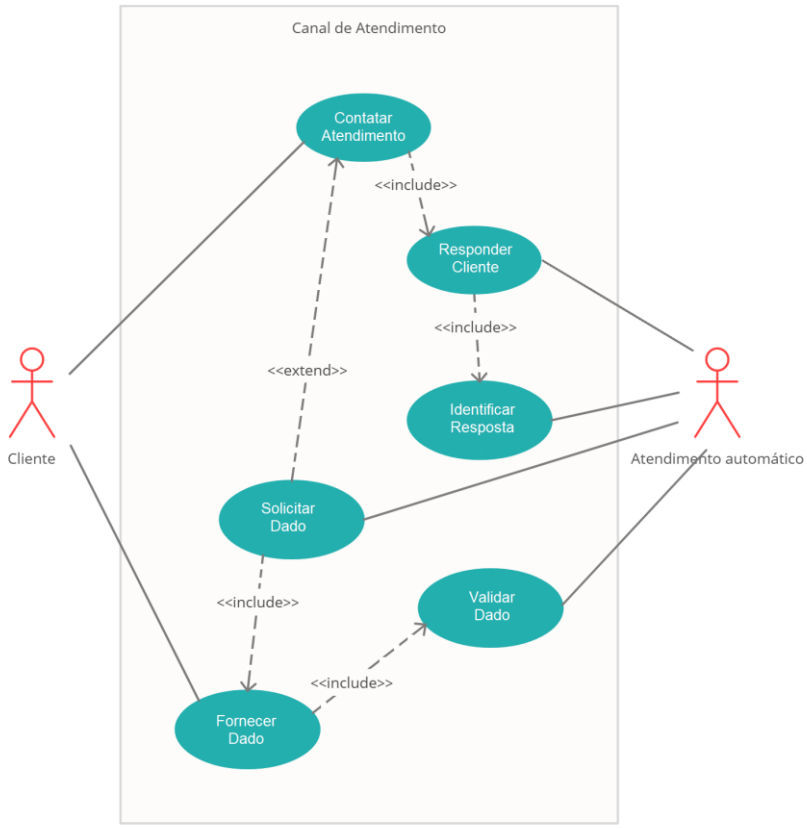
4.1 Modelagem de Análise do Domínio

O diagrama de caso de uso das funcionalidades comuns para as três aplicações-exemplo é apresentado na Figura 10. Os casos de uso identificados foram:

- Contatar Atendimento – O Cliente contata o AA através de uma mensagem no canal de atendimento;
- Responder Cliente – O AA responde de maneira adequada a mensagem enviada pelo Cliente;
- Identificar Resposta – O AA identifica a resposta adequada para a mensagem enviada pelo Cliente;
- Solicitar Dado – O AA pode responder à mensagem do Cliente solicitando algum dado ao mesmo;
- Fornece Dado – Quando um dado for solicitado, o Cliente deve fornecer uma entrada;

- Validar Dado – Quando um dado for informado o AA deve validar esse dado e responder adequadamente.

Figura 10 – Diagrama de Caso de Uso do comportamento comum das aplicações-exemplos

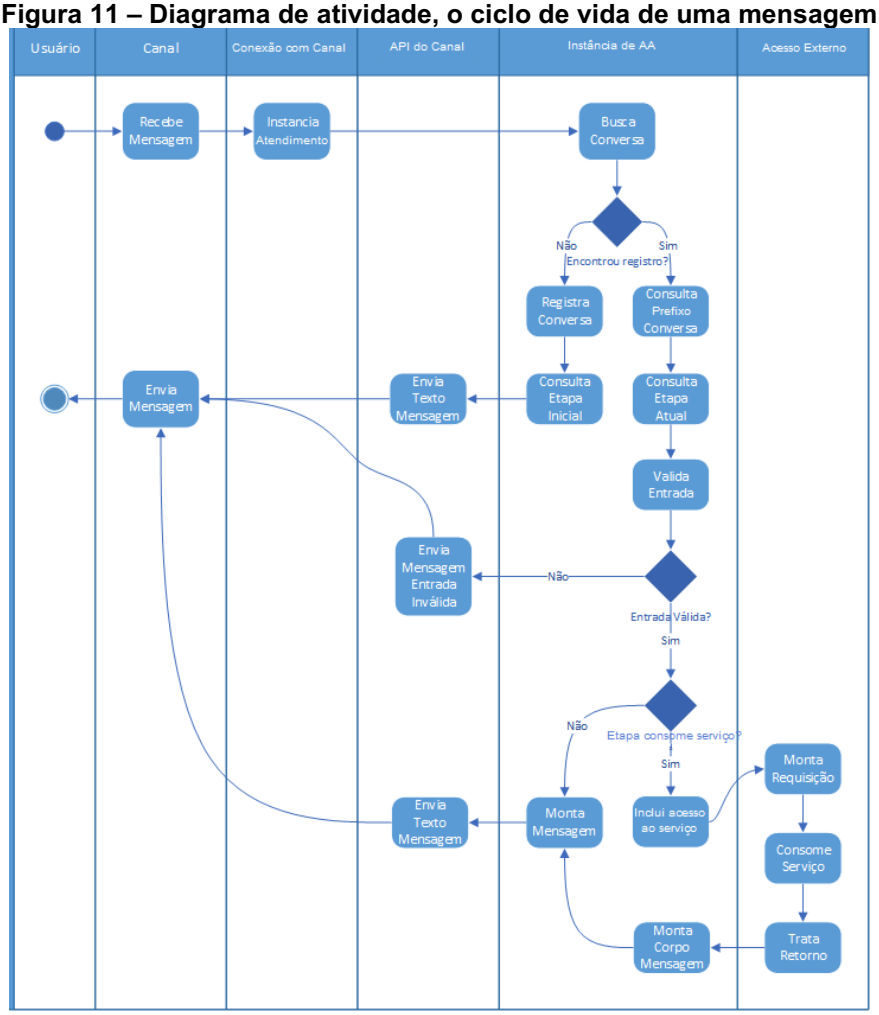


Fonte: Autoria própria (2022).

Com o entendimento das aplicações-exemplo e seu comportamento foi possível iniciar o processo de estudo do domínio utilizando a integração de metodologias ágeis e de *startup*.

A Figura 11 apresenta, com base nesse entendimento, um diagrama de atividade do ciclo de vida dessa informação. A ideia é composta por 6 elementos: Usuário, Canal, Conexão com Canal, API do Canal, Instância de Atendimento Automático e Acesso Externo. Na estratégia adotada também já ficaram visíveis algumas classes que podem compor o *framework* nos processos abstratos presentes nos elementos descritos. O Quadro 2 apresenta o funcionamento esperado de cada elemento do conceito inicial.

Figura 11 – Diagrama de atividade, o ciclo de vida de uma mensagem



Fonte: Autoria própria (2022)

Quadro 2 - Detalhamento dos elementos do atendimento automatizado

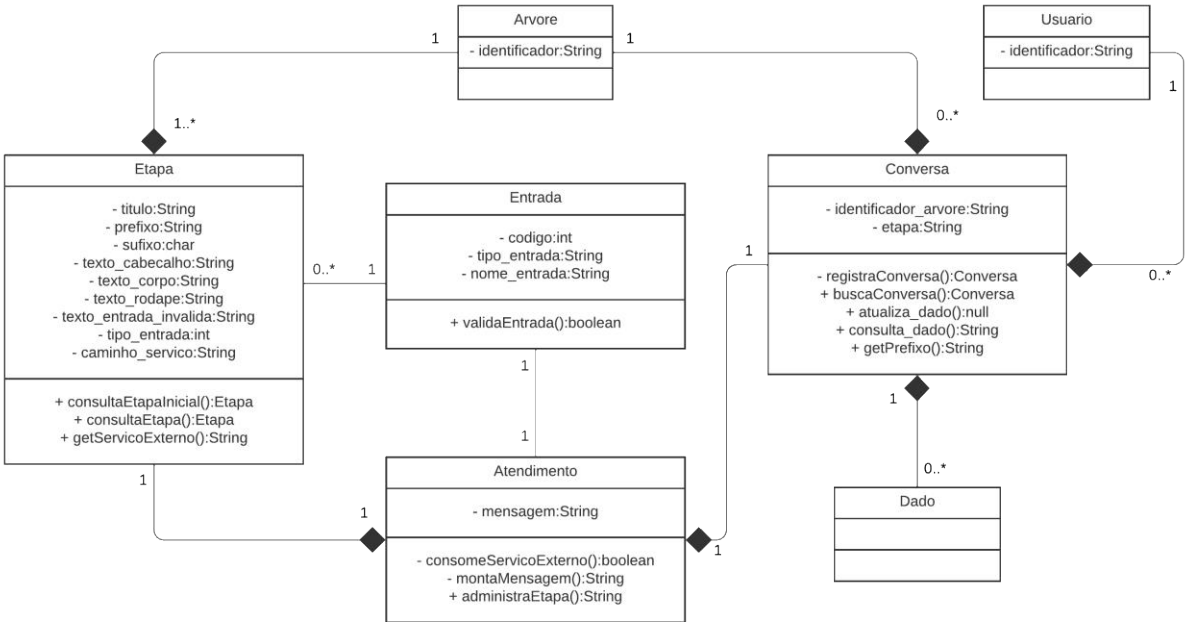
Elemento	Funcionalidade
1. Usuário	O cliente conversando com o canal de atendimento. É o ponto onde se inicia e termina o fluxo de informação. O atendimento e cada etapa que o compõem só se iniciam e progridem por ação do Usuário.
2. Canal	Canal de atendimento por onde o cliente se comunica. É o único ponto sensível ao Usuário e é externo à aplicação. Ex: WhatsApp, *Messenger*, *webchat.*
3. Conexão com Canal	Meio de comunicação por onde o Canal notifica a aplicação das interações entre o Canal e o Usuário. Essencialmente um *webhook*. Primeiro componente da aplicação.
4. API do Canal	Elemento que consome a API do Canal com a finalidade de enviar ao Canal as ações desejadas, que posteriormente serão repassadas ao Usuário, se necessário. Por onde a resposta da interação do usuário retorna da aplicação para o Canal e o Usuário.
5. Instância de AA	Instância da aplicação. Cada ação do Usuário no Canal, cada mensagem, gera uma nova instância da aplicação que processa a mensagem e responde adequadamente através da API do Canal.
6. Acesso Externo	Elemento no qual a aplicação pode consumir serviços externos à aplicação (APIs) quando necessário a fim de adquirir alguma informação para o atendimento ou registrar a interação em outra aplicação.

Fonte: Autoria própria (2022).

Por meio de testes constatou-se que os comportamentos no diagrama de atividade, reproduz com exatidão o comportamento básico de todas as aplicações-exemplo selecionadas. Também nesse ponto é possível prever pontos de estabilidade e flexibilidade do *framework* resultante.

O elemento Instância de AA e seus componentes, que realizam todas as tomadas de decisão do atendimento, podem ser reutilizados por todas as aplicações do domínio, sendo assim, um ponto de estabilidade (*frozen spots*). O restante dos elementos são pontos de flexibilidade (*hot spots*) por serem exclusivos de cada aplicação do domínio.

Figura 12 – Diagrama de classe do *framework*



Fonte: Autoria própria (2022).

O diagrama de classe visto na Figura 12 foi elaborado, a partir da compreensão dos diagramas de caso de uso e de atividade das Figuras 10 e 11, como passo do processo de desenvolvimento de Johnson (1993, apud MATOS; FERNANDES, 2007, p. 9) para o *framework* e representa os pontos de estabilidade obtidos a partir da generalização das aplicações-exemplo.

4.2 Extensão do *framework*

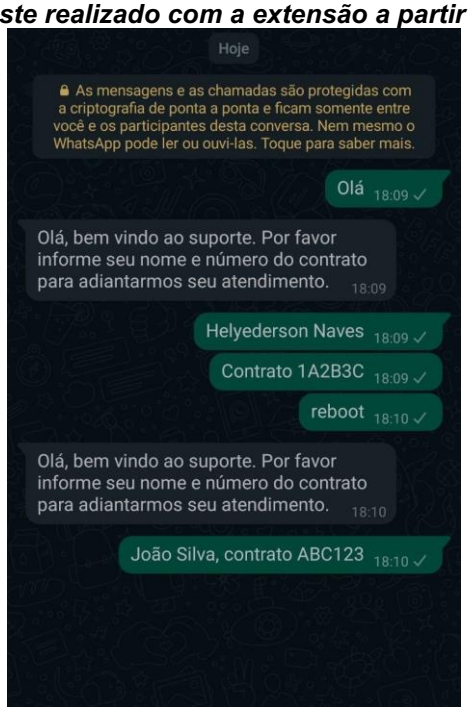
A extensão do *framework* consiste no complemento e validação dos dados analisados nas etapas de *Design Thinking* apresentados na Seção 3.4. Após a validação da ideia com base nos testes de mesa e o conceito teórico estabelecido, o desenvolvimento de um protótipo se iniciou. Para o desenvolvimento foi utilizado a

linguagem de programação PHP e o canal de comunicação escolhido foi o *WhatsApp*. Uma aplicação do tipo *webscrap,* utilizada para obter informações do *WhatsApp Web*, foi desenvolvida para realizar a conexão da Instância de AA e do Canal, enviando as mensagens recebidas e o usuário da conversa por meio do *webhook.* Também foi desenvolvida uma API simples que receba informações da Instância e permita enviar mensagens através do Canal.

A extensão inicial foi desenvolvida com a finalidade de responder a primeira mensagem enviada pelo Usuário com uma resposta padrão definida em código, com o elemento Instância de AA simplificado, sem a funcionalidade de armazenar dados no sistema ou consumir serviços externos. A mensagem reservada "*reboot*" foi criada para fins de desenvolvimento, o comando faz o AA tratar a mensagem como o primeiro contato do Usuário e consequentemente a aplicação responder novamente.

O resultado da extensão apresentou comportamento semelhante a funcionalidade de resposta automática presente apenas no *AA1*. Essa simples implementação permitiu que o usuário tenha uma resposta ao contato inicial. Outra possibilidade é que a resposta automática fixada no código solicite dados que adiantarão o processo de atendimento, como pode ser visualizado na Figura 13.

Figura 13 – *Teste realizado com a extensão a partir do framework*



Fonte: Autoria própria (2022).

Apesar da extensão apresentar algumas funcionalidades, não houve desenvolvimento de nenhum dos processos que compõem a Instância de AA como exibido na Figura 11.

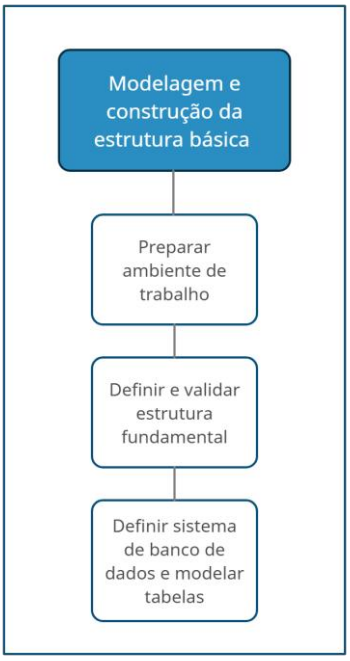
Vale ressaltar que se a partir dos testes a ideia escolhida não conseguisse ser validada ou não atendesse as expectativas, etapas anteriores da metodologia *Design Thinking* poderiam ser revisitadas a fim de reformular os conceitos e definir uma nova ideia.

A partir da extensão apresentada é possível realizar as *sprints* do *Scrum* de acordo com o *Product Backlog* mostrado na Figura 9. A Instância foi complementada e aprimorada durante as próximas etapas de construção até que se constitua um *framework* que atenda todos os requisitos identificados.

4.3 Modelagem e construção da estrutura básica do *framework*

A primeira semana de desenvolvimento teve como objetivo modelar a estrutura básica do *framework*. O resultado do *Sprint Planning* da semana 1 pode ser visto na Figura 14.

Figura 14 – *Sprint Planning semana 1*



Fonte: Autoria própria (2022)

4.3.1 *Preparação do ambiente de trabalho*

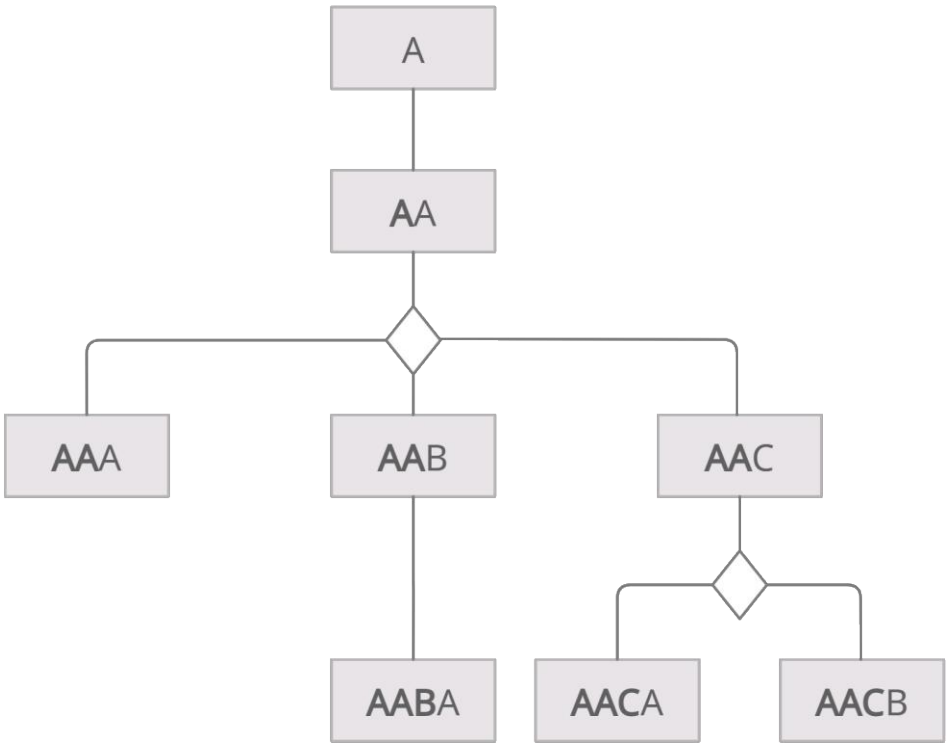
O *sprint* iniciou-se preparando o ambiente de trabalho para a construção da aplicação. Dentre as tarefas realizadas esteva a criação de um servidor Apache localmente e a configuração do PHP no servidor. A linguagem multiparadigma PHP foi escolhida pela familiaridade do desenvolvedor com a mesma.

Elementos desenvolvidos para Conexão com Canal e API do Canal foram reutilizados para possibilitar testes durante o desenvolvimento. O código desses elementos não vai compor o *framework* e sim realizar uma função fundamental para o *framework* que é conectar à Instância ao Canal. Cada aplicação do *framework* deve realizar sua própria integração com o canal desejado.

4.3.2 *Definição e validação estrutura fundamental*

O processo consistiu em idealizar como tornar possível o funcionamento da Instância de AA como mostrado na Figura 11. Para isso foi criado um modelo de Etapas, que possuem como principal parâmetro um índice alfabético, agrupadas e ordenadas em uma árvore de decisão como pode ser visto na Figura 15.

Figura 15 – *Modelo de Etapas*



Fonte: Autoria própria (2022).

O índice alfabético de uma etapa é composto por dois elementos, um prefixo e um sufixo, sendo prefixo todos os caracteres alfabéticos com exceção do último e o sufixo sendo o último caractere alfabético. O índice permite que sempre se saiba qual a Etapa antecede a outra e as opções de continuação quando houver uma ou mais possibilidades. Por exemplo, o índice da etapa anterior sempre é o da atual removendo-se o último caractere, na árvore o índice do pai é o prefixo do filho. Dessa

forma o índice da etapa seguinte sempre possuirá como prefixo o índice da etapa anterior seguido do sufixo que o caractere “A” quando for um sucessor exclusivo ou outro caractere, em ordem alfabética, em que cada caractere representa uma opção possível.

O modelo de Etapas com índice alfabético foi pensado com a finalidade de simplificar a criação e personalização de árvores de interação para o atendimento por meio de um banco de dados ou um arquivo de leitura que pode ser escrito manualmente.

Para o desenvolvimento do algoritmo a Etapa tornou-se a primeira classe idealizada. O Quadro 3 apresenta os atributos da classe Etapa, o tipo e a finalidade de cada um.

Quadro 3 - Classe Etapa

Atributo	Tipo	Finalidade
1. titulo	Cadeia de caracteres	Armazena o título da Etapa, utilizado como rótulo é exibido quando é uma opção de sequência para a Etapa atual.
2. identificador_arvore	Cadeia de caracteres	Armazena o identificador da árvore de atendimento da qual a etapa pertence. Permite que árvores de interação diferentes sejam criadas no mesmo sistema.
3. prefixo	Cadeia de caracteres	Armazena o prefixo da etapa. O índice da etapa anterior à atual.
4. sufixo	Caractere	Armazena o caractere que representa o sufixo da etapa atual.
5. texto_cabecalho	Cadeia de caracteres	Armazena o texto de cabeçalho da mensagem.
6. texto_corpo	Cadeia de caracteres	Armazena o texto de corpo da mensagem.
7. texto_rodape	Cadeia de caracteres	Armazena o texto de rodapé da mensagem.

Fonte: Autoria própria (2022).

Outra classe necessária para o desenvolvimento da estrutura é a classe conversa, que tem a finalidade de manter o controle de qual etapa e qual árvore de interações o usuário se encontra no atendimento. O Quadro 4 apresenta os atributos da classe Conversa, o tipo e a finalidade de cada um.

Quadro 4 - Classe Conversa

Atributo	Tipo	Finalidade
1. identificador_usuario	Cadeia de caracteres	Armazena o identificador do usuário obtido através do Canal de comunicação. É usado para que o sistema identifique em que etapa da conversa o usuário se encontra e é usado como destino da resposta da conversa. Pode ser um *username* único, *id* ou número de telefone, dependendo exclusivamente do canal de comunicação.
2. identificador_arvore	Cadeia de caracteres	Identifica com qual árvore o usuário realizou a interação.
3. etapa	Cadeia de caracteres	Armazena o index da Etapa na qual o atendimento se encontra.

Fonte: Autoria própria (2022).

A partir de testes realizados utilizando etapas presentes nas interações com as aplicações-exemplo foi averiguado que a estrutura proposta conseguiu reproduzir o comportamento. A estrutura consiste em uma simplificação do elemento Instância de AA proposto na Seção 4.1.

4.3.3 *Definição do sistema de banco de dados e modelar tabelas*

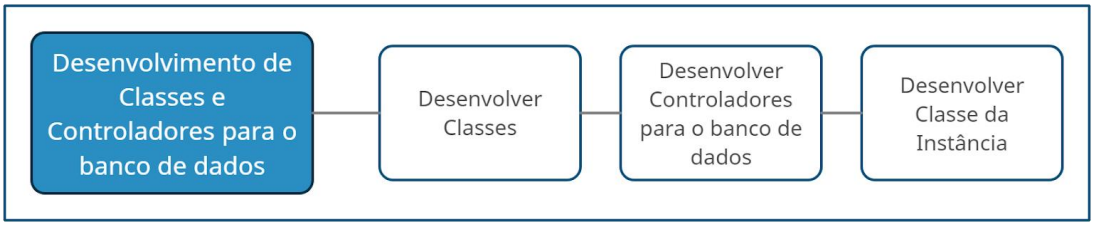
MySQL (ORACLE, 2022) foi o sistema de banco de dados selecionado para o desenvolvimento do *framework*. O sistema foi configurado no servidor e as dependências de conexão foram desenvolvidas para permitir que o PHP crie conexões com o MySQL. As tabelas Etapas e Conversas foram modeladas no MySQL com a finalidade de armazenar os atributos das respectivas classes.

No *Sprint Review* da semana 1 foi analisado que a base do *framework* foi modelada. A *sprint* não teve como foco o desenvolvimento de código e sim a modelagem do que foi desenvolvido na semana 2. Validações das ideias já haviam sido realizadas no processo de descoberta do produto, mas foram repetidas a fim de garantir que o desenvolvimento seguiu o caminho correto. Como nenhum desenvolvimento foi realizado o resultado ainda parece abstrato, por meio das futuras *sprints* o *framework* como produto deve tomar forma.

4.4 Modelagem de classes para o banco de dados

A segunda semana de desenvolvimento foi designada para o desenvolvimento das classes e controladores para as tabelas anteriormente modeladas. Neste período foi desenvolvido a Classe principal da aplicação, a Instância de AA. O resultado do *Sprint Planning* da semana 2 pode ser visto na Figura 16.

Figura 16 – *Sprint Planning semana 2*



Fonte: Autoria própria (2022).

As classes e controladores aumentaram a complexidade da aplicação, permitindo que atendimentos mais complexos sejam realizados.

4.4.1 *Desenvolvimento de classes e controladores*

As classes Conversa e Etapa modeladas na semana 1 foram desenvolvidas utilizando a linguagem PHP. As tabelas Conversas e Etapas foram criadas no banco de dados. Além disso, foi desenvolvido controladores para as classes permitindo a manipulação de suas respectivas tabelas no banco de dados, possibilitando as quatro operações básicas CRUD (*Create, Read, Update, Delete*).

4.4.2 *Desenvolvimento da classe da Instância*

Foi desenvolvida a classe principal da aplicação, chamada Atendimento. A classe Atendimento possui como atributos objetos das outras classes da estrutura e a mensagem recebida. O Quadro 5 apresenta os atributos da classe Atendimento, o tipo e a finalidade de cada um.

Quadro 5 - Classe Atendimento

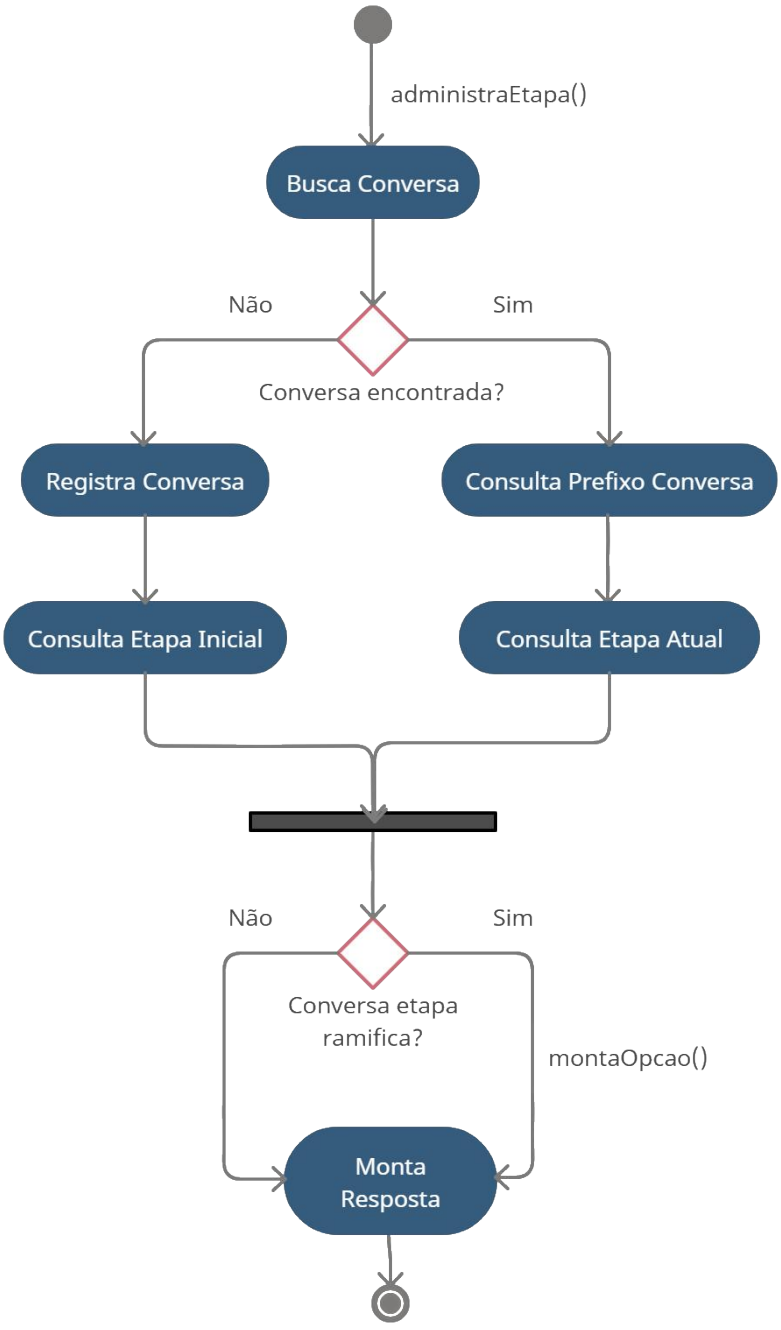
Atributo	Tipo	Finalidade
1. conversa	Conversa	Instância da classe Conversa. Utilizada para identificar a conversa. Caso identifique preenche o objeto com as informações salvas no banco de dados. Caso não identifique, uma nova conversa é criada no banco de dados.
2. etapa	Etapa	Instância da classe Etapa. Utilizada para identificar a etapa na qual a conversa se encontra. Caso uma nova conversa seja adicionada, a etapa padrão é sempre a de menor índice.
3. mensagem	Cadeia de caracteres	Armazena a mensagem recebida através do canal.

Fonte: Autoria própria (2022).

O método principal da classe chamado “administraEtapa” é responsável processar a mensagem recebida e retornar uma resposta adequada. Também foi desenvolvido um método privado chamado “montaOpcao” responsável por montar

opções e anexar ao corpo da resposta quando a etapa possuir mais de uma possibilidade de continuação. O diagrama de atividade demonstrado na Figura 17 explica a lógica por trás do método “administraEtapa”.

Figura 17 – Diagrama de atividade, processamento da mensagem na semana 2



Fonte: Autoria própria (2022).

O método “administraEtapa” é responsável pela atividade mais essencial do Atendimento, que é responder adequadamente de acordo com a entrada do Cliente.

Com o desenvolvimento das classes, o produto demonstrou uma significativa evolução do *framework* permitindo a criação de AAs mais complexos e ramificados

por meio de entradas do banco de dados e sem a necessidade de nenhuma alteração em código.

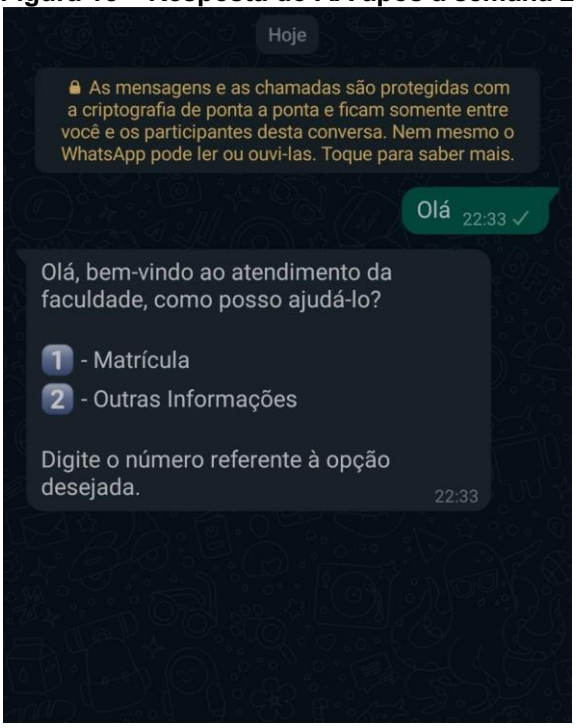
Inserindo na tabela Etapas as informações do Quadro 6 foi possível gerar a interação apresentada na Figura 18.

Quadro 6 - Inserções na tabela Etapas

titulo	identificador_arvore	prefixo	sufixo	texto_cabecalho	texto_rodape
Saudação	arvore1		A	Olá, bem-vindo ao atendimento da faculdade, como posso ajudá-lo?	Digite o número referente à opção desejada.
Matrícula	arvore1	A	A		
Outras Informações	arvore1	A	B		

Fonte: Autoria própria (2022)

Figura 18 – Resposta do AA após a semana 2



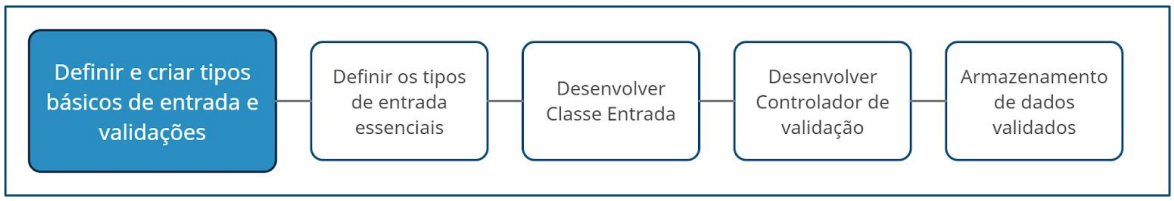
Fonte: Autoria própria (2022)

No *Sprint Review* da semana 2 constatou-se que o *framework* já era capaz de gerar um AA com etapas de múltipla escolha, porém sem a capacidade de decidir qual opção seguir. Essa situação é resolvida na próxima *sprint* através de métodos de validação de dados.

4.5 Validação do *Framework*

A terceira semana de desenvolvimento teve como objetivo definir os tipos básicos de entrada, criar a classe Entrada, desenvolver validadores para os tipos de entrada selecionados e alterar a estrutura para armazenar os dados validados quando necessário. O resultado do *Sprint Planning* da semana 3 pode ser visto na Figura 19.

Figura 19 – Sprint Planning semana 3



Fonte: Autoria própria (2022).

4.5.1 *Definição dos tipos de entrada essenciais*

O Quadro 7 apresenta os tipos de entrada que foram considerados essenciais, o código designado, tipo de dado esperado e sua descrição.

Quadro 7 - Tipos de entradas essenciais

Tipo Entrada	Código	Dado esperado	Descrição
opção	1	Inteiro [1, 2... n]	Tipo usado quando a árvore de interação ramifica em N caminhos. O usuário deverá informar qual a opção desejada.
booleano	2	[1, 2] / Sim / Não	Tipo usado quando a árvore ramifica em apenas 2 caminhos.
texto	3	Cadeia de caracteres alfanuméricos	Uma entrada de texto podendo conter letras e números.
numeral	4	Cadeia de caracteres numéricos	Uma entrada de texto contendo apenas números.
inteiro	5	Inteiro	Um número inteiro.
decimal	6	Decimal	Um número decimal.
caractere	7	Caractere	Apenas um caractere alfanumérico.
e-mail	8	Formato email	Cadeia de caracteres que possua formato de email. Exemplo: teste@teste.com
qualquer	9	-	Qualquer entrada será considerada válida, indo automaticamente para a próxima etapa.

Fonte: Autoria própria (2022)

Os tipos de entradas essenciais podem ser incrementados futuramente, de acordo com a necessidade da aplicação. Pode-se incrementar com dados como data, CPF a fim de atender um requisito do atendimento.

4.5.2 *Desenvolvimento da classe Entrada*

O Quadro 8 apresenta os atributos da classe Entrada, o tipo e a sua finalidade.

Quadro 8 - Classe Entrada

Atributo	Tipo	Finalidade
1. codigo	Inteiro	Código com a finalidade de ser utilizado como chave primária na tabela Entradas e chave estrangeira na tabela Etapas
2. tipo_entrada	Cadeia de caracteres	Nome do tipo essencial.
3. nome_entrada	Cadeia de caracteres	Nome dado a entrada recebida.

Fonte: Autoria própria (2022)

A classe Entrada foi desenvolvida e a tabela Entradas criada no banco de dados.

4.5.3 *Desenvolvimento do controlador de validação*

O controlador de validação foi desenvolvido de modo que receba a mensagem do usuário e o tipo de dado esperado e verifique se a entrada é válida ou não. O controlador possui métodos para validar todas as entradas essenciais e pode incluir arquivos que façam a validação de outros dados futuramente. O método principal do controlador recebe a mensagem e retorna se é válida ou não.

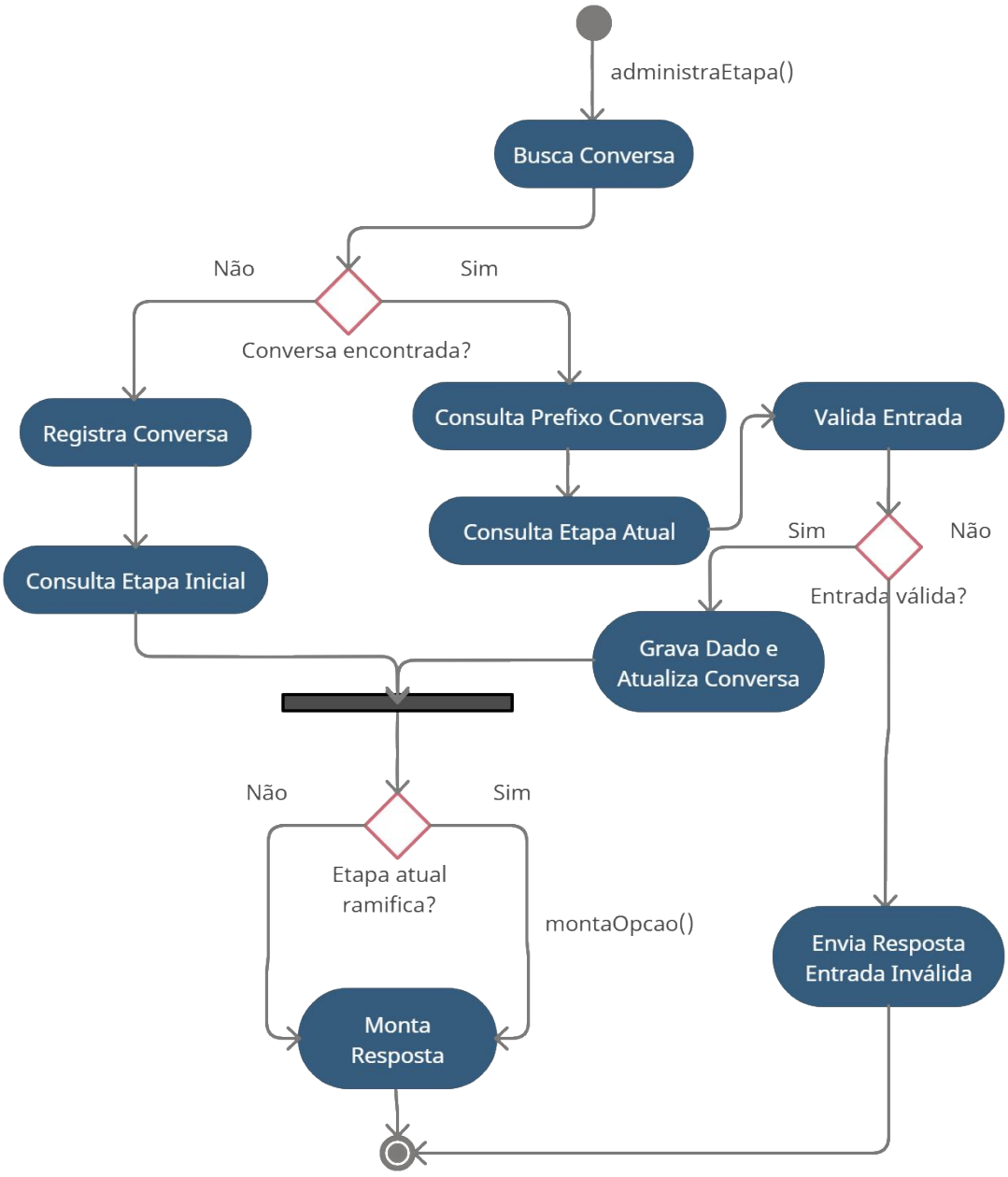
4.5.4 *Alterações para validação e armazenamento de dados*

Para possibilitar que os dados validados sejam armazenados, foi necessário alterar a classe Conversa. Foi adicionado o atributo “dados” que armazena um *json* contendo o nome da entrada, definido no atributo “nome_entrada” da classe Entrada, e a entrada informada pelo usuário e que foi validada. Outra alteração realizada foi a criação dos atributos “tipo_entrada” e “texto_entrada_inválida” na classe Etapa que definem o tipo de entrada esperado na etapa e o texto utilizado na resposta caso a entrada seja inválida, respectivamente. Todas as tabelas relativas às classes também foram alteradas para armazenar os novos atributos.

O método “administraEtapa” da classe Atendimento foi alterado para validar toda mensagem do usuário por meio do controlador de validação. Foi adicionado ao método a capacidade de substituir automaticamente *tags* presentes na mensagem com dados previamente coletados.

O diagrama de atividade apresentado na Figura 20 exemplifica a lógica do método “administraEtapa” após as alterações realizadas.

Figura 20 – Diagrama de atividade, processamento da mensagem na semana 3



Fonte: Autoria própria (2022).

Com os desenvolvimentos realizados na semana 3 foi possível criar árvores de interação no canal de atendimento que realizam tarefas ainda mais complexas, como coleta de dados e atendimento setorizado. Nesse ponto, o desenvolvimento já permite que as aplicações-exemplo sejam replicadas, validando os processos realizados até o momento da finalização da *sprint*.

Com as funcionalidades presentes ao final da semana 3, já constitui um *framework* capaz de atender ao domínio e originar diferentes soluções de atendimento automatizado. A fim de justificar essa afirmação foi realizado o teste sobre a aplicação-

exemplo AA2. Inserindo na tabela Etapas as informações do Quadro 9 foi possível reproduzir o comportamento da aplicação-exemplo, como pode ser visto na Figura 21.

Quadro 9 – Inserções na tabela Etapas da semana 3

titulo	identificador_ arvore	prefixo	sufixo	texto_ cabecalho	texto_ entrada_inválida	tipo_ entrada
Solicita nome	RoboChat		A	Vamos lá! Qual o seu nome?	Por favor informe apenas o seu nome.	3
Solicita e-mail	RoboChat	A	A	Qual o seu melhor e-mail?	Por favor informe apenas o seu e-mail.	8
Solicita telefone	RoboChat	AA	A	Obrigado! Qual seu celular/whatsapp com DDD?	Por favor informe apenas o seu número de celular.	4

Fonte: Autoria própria (2022)

Figura 21 – Reprodução da aplicação-exemplo através do *framework*

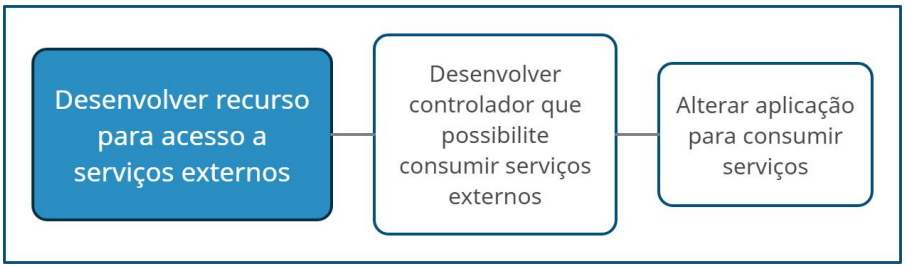
Fonte: Autoria própria (2022).

Como foi possível observar, replicar o funcionamento das aplicações-exemplo exige apenas que o AA seja configurado por meio de entradas no banco de dados. Isso permite que os textos, ordem das etapas, tipos de dados sejam facilmente modificados e com isso dê origem a novas soluções de AA.

4.6 Desenvolver o consumo de recursos externos

A quarta semana de desenvolvimento teve como objetivo incrementar a aplicação de maneira que consiga consumir serviços externos (APIs) com a finalidade de obter dados ou realizar ações em outros sistemas. O resultado do *Sprint Planning* da semana 4 pode ser visto na Figura 22.

Figura 22 – *Sprint Planning* semana 4



Fonte: Autoria própria (2022).

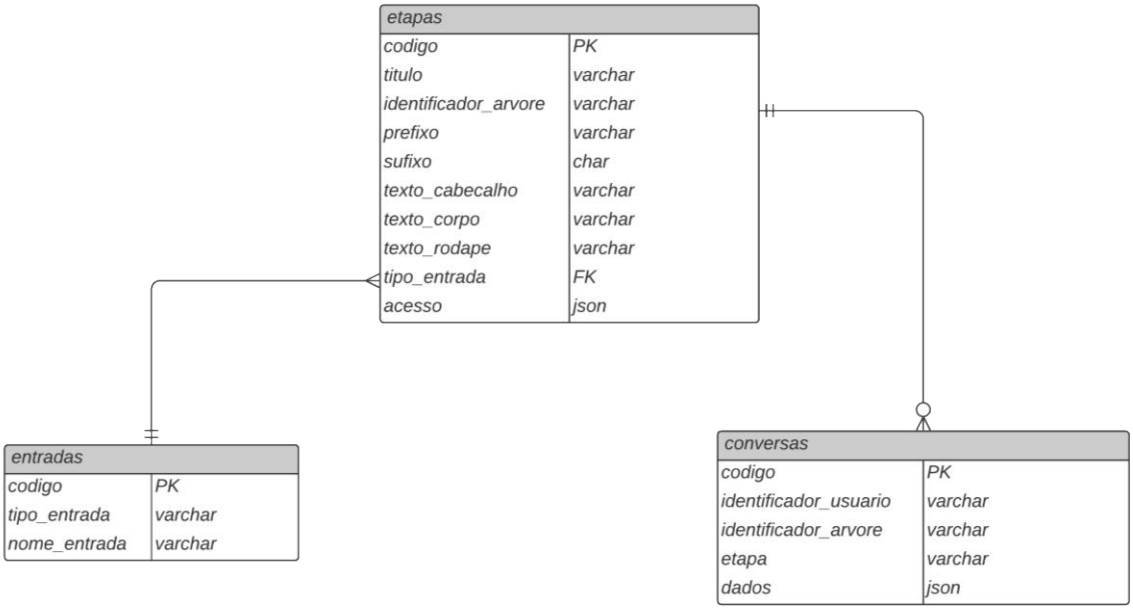
4.6.1 *Desenvolvimento de controlador para consumo de APIs*

A solução encontrada para que o *framework* consuma (acesse e utilize) serviços dinamicamente foi desenvolver um controlador que receba um *json. O json* em questão é formado pela concatenação de dois outros *json*.

O primeiro *json* é o que armazena os dados coletados em todas as etapas e fica armazenado no atributo “dados” da classe Conversa. O segundo *json* deve conter o caminho do arquivo de integração, responsável por realizar as chamadas necessárias da API e validar retornos, e outros dados necessários para o consumo do serviço, mas que são comuns a todos os usuários que passarem pela etapa que consome o serviço. Por este motivo, foi necessária a criação de um novo atributo na classe Etapa chamado “acesso” que tem como finalidade armazenar o segundo *json* citado.

O diagrama entidade relacionamento das tabelas do banco de dados após as alterações da semana 4 pode ser visto na Figura 23. O banco de dados apresenta um papel fundamental no funcionamento da aplicação implementada. Através de inserções nas tabelas descritas, diferentes sistemas de AA podem ser implementados.

Figura 23 – Diagrama Entidade Relacionamento

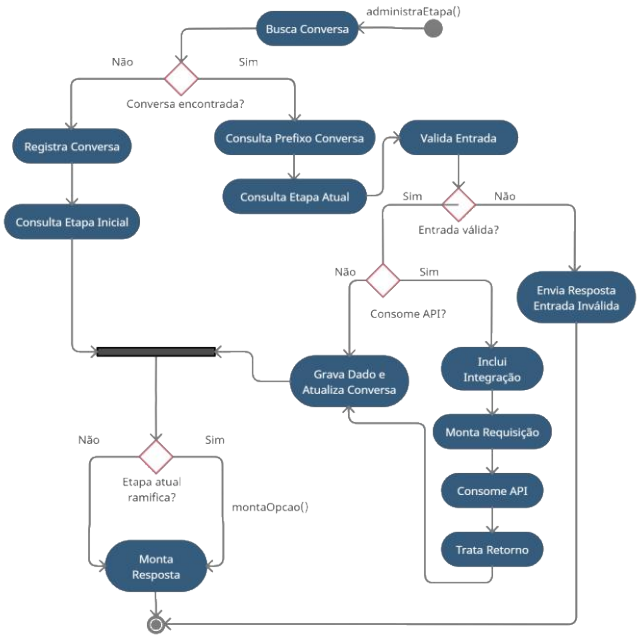


Fonte: Autoria própria (2022).

4.6.2 *Incremento do Atendimento para consumir serviços*

O controlador desenvolvido foi inserido na lógica do método “administraEtapa” tornando possível que APIs sejam consumidas em qualquer etapa com a finalidade de consulta ou ação em um serviço externo. O diagrama de atividade da Figura 24 ilustra as alterações realizadas no método “administraEtapa”.

Figura 24 – Diagrama de atividade, processamento da mensagem na semana 4

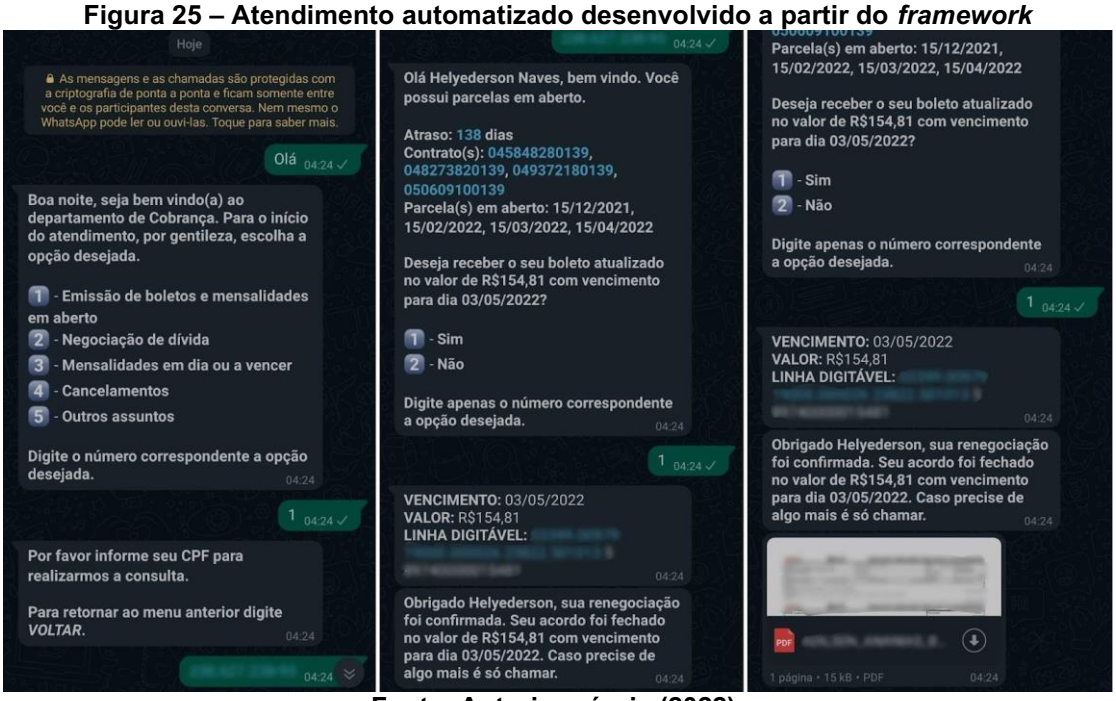


Fonte: Autoria própria (2022).

A *sprint* da semana 4 foi a última prevista no *Product Backlog*, encerrando assim uma iteração de *Scrum* e o processo de construção de produto da metodologia *Lean Startup*. Para demonstrar a aplicação foi criada uma árvore de iteração capaz de realizar atendimento de uma instituição de cobranças utilizando todos os recursos que foram desenvolvidos ao longo das 4 semanas

O produto resultante, ilustrado na Figura 25 ao final da iteração mostrou-se capaz de cumprir todos os objetivos, requisitos e expectativas definidos na Seção 3.4.1 e possuir todas as características definidas na Seção 3.3.

Figura 25 – Atendimento automatizado desenvolvido a partir do *framework*



Fonte: Autoria própria (2022).

4.7 Análise dos resultados

Ao final da iteração do *Scrum* contatou-se que a análise do domínio e a compreensão do problema por meio das metodologias para *startup* resultou na elaboração de artefatos do *Scrum* que permitiram que o desenvolvimento atingisse o resultado desejado.

Ao utilizar os conceitos de *Customer Development* na definição do domínio foi escolhido um caminho em que o desenvolvimento ocorra com participação do usuário.

A importância da *Lean Startup* se caracteriza pela aplicação dos seus três pilares em diferentes etapas da iteração inicial de desenvolvimento do *framework*. O pilar do Aprender foi realizado por meio da aplicação das etapas do *Design Thinking,* como descrito na Seção 3.4, e possibilitou compreender todos os aspectos do

desenvolvimento e produzir um *Product Backlog* onde o produto é o próprio *framework* do domínio. O pilar do Medir, onde a aplicação do *Customer Development* continua*,* pôde ser satisfeito pela própria ferramenta construída a partir do *framework,* que possibilita obter o *feedback* dos usuários e a coleta de dados necessária. O pilar do Construir, onde o produto é desenvolvido através de metodologias ágeis, consiste em iterações de *Scrum*, possibilitando melhorias mensuráveis a cada iteração do ciclo Construir-Medir-Aprender da *Lean Startup.*

Ao final do desenvolvimento abordado neste capítulo, o desenvolvedor que se propor a utilizar ou aprimorar o *framework* terá as ferramentas necessárias no ciclo Construir-Medir-Aprender da metodologia *Lean Startup*.

O produto resultante caracteriza-se como um *framework* em duas escalas, a primeira a nível de estrutura, onde utilizando-se dos processos descritos, um desenvolvedor pode implementar a estrutura de geração de AA no sistema que desejar. A segunda é na própria criação de aplicações de AA após a aplicação do *framework*, permitindo que inúmeras árvores sejam criadas com a finalidade de suprir demandas de atendimento. Após a aplicação do *framework* essas árvores podem ser criadas ou configuradas através de operações no banco de dados.

4.8 Considerações finais do capítulo

Esse capítulo detalhou a aplicação do processo de desenvolvimento do *framework* de domínio através da aplicação dos artefatos obtidos no estudo do domínio. Através de uma extensão do *framework* como base para o desenvolvimento e uma iteração completa de *Scrum,* foi realizado o desenvolvimento incremental do *framework* através de *sprints* até que o mesmo implementasse o que foi definido no *Product Backlog* e atendesse os objetivos e funcionalidades esperadas para o *framework*.

Ao final do desenvolvimento foi possível analisar o produto gerado e compreender as contribuições de se utilizar metodologia ágeis e de *startup* no desenvolvimento do *framework.*

5 CONCLUSÃO

Este trabalho aplicou metodologias voltadas para *startups* em conjunto com metodologias ágeis na criação de um *framework* de domínio de serviço para Atendimento Automatizado (AA). O processo que uniu *Design Thinking*, *Lean Startup* e *Scrum* foi utilizada para abordar pontos como *design* e modelagem de produto sobre a ótica do usuário final ao mesmo tempo que propõe um desenvolvimento iterativo e incremental. Essas características permitem que o *framework* pode ser expandido e estendido a fim se se tornar algo mais abrangente do que o inicialmente definido a partir das aplicações-exemplo, se tornando uma ferramenta flexível e reutilizável a cada iteração de desenvolvimento.

O *framework* obtido ao final da primeira iteração do *Scrum* foi utilizado para replicar o comportamento das aplicações-exemplo e dar origem a soluções de AA mais complexas, como a solução vista na Figura 24, desenvolvido como ferramenta para realizar atendimento, através do *WhatsApp*, de clientes assinantes de um plano mensal.

O AA gerado pôde fornecer diversas opções de atendimento relacionadas ao plano, e como exibido no exemplo, conseguiu solicitar e coletar dados do usuário, obter informações do mesmo através de um serviço externo e registrar uma atividade do usuário no serviço, enviando ao cliente uma renegociação dos valores em aberto.

O resultado da união das metodologias permitiu que o produto final fosse um *framework* replicável em duas escalas, em que é possível implementar a estrutura apresentada neste trabalho em outros sistemas, linguagens e canais de comunicação e também utilizar a estrutura para criar diferentes árvores de interação apenas com inserções no banco de dados.

Portanto, metodologias para *startup,* focadas em desenvolvimento de produto e com foco em mercado, em conjunto com metodologias ágeis contribui com um desenvolvimento objetivo e focado para se criar um produto funcional por meio de etapas formais de desenvolvimento e fornece recursos para aprimorar iterativamente e interativamente o *software* desenvolvimento.

5.1 Trabalhos futuros

No que diz respeito a análise e compreensão do domínio utilizando metodologias ágeis e de *startup,* é possível aplicar a metodologia utilizada neste trabalho, aqui utilizada no desenvolvimento de um *framework* de domínio de um produto de *software*, no desenvolvimento de um *framework* de domínio de um serviço.

No quesito produto de *software* para *startups*, pode-se utilizar a aplicação criada no estudo dos impactos de se oferecer um canal de AA especializado no produto da *startup* já na entrega do MVP e analisar como isso auxilia no processo iterativo da *Lean Startup* e no desenvolvimento com participação do consumidor.

A respeito do *framework* desenvolvido, uma nova iteração de *Lean Startup* pode ser realizada com a finalidade de aprimorar aspectos do *framework,* utilizando conceitos de Inteligência Artificial para transformar o processamento da mensagem do cliente em um processo cognitivo, substituindo árvores de iteração por respostas dirigidas a contexto. Também pode ser realizado um estudo sobre a aplicação do *framework* em diferentes canais de comunicação e no desenvolvimento de diferentes funcionalidades.

REFERÊNCIAS

ABRANTES, F. J.; TRAVASSOS, G. H. Common Agile Practices in Software Processes. *In:* INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT. 2011, Banff, Canada. **Anais [...]** Canada: ESEM, 22-23 set. 2011.

AGILE MANIFESTO. **Manifesto para Desenvolvimento Ágil de software**. Disponível em: https://agilemanifesto.org/iso/ptbr/manifesto.html. Acesso em: 09 mai. 2022.

AGILE MANIFESTO. **Os doze princípios por trás do Manifesto Ágil**. Disponível em: https://agilemanifesto.org/iso/ptbr/principles.html. Acesso em: 09 mai. 2022.

AMBROSE, G; HARRIS, P. **Design Thinking**. Worthing: AVA, 2011.

BANACH, S. J.; RYAN, A. The art of design: A design methodology. **Military Review**, 89, p 105-115, abr. 2009.

BJÖRK, J.; LJUNGBLAD, J.; BOSCH, J. Lean Product Development in Early Stage Startups. *In:* LIFE CYCLES OF SOFTWARE PRODUCTS, Potsdam, Germany. **Anais [...]** Germany: IW-LCSP@ICSOB, 2013.

BLANK, S. A Startup is Not a Smaller Version of a Large Company. **Site**. 14 jan. 2010. Disponível em: https://steveblank.com/2010/01/14/a-startup-is-not-a-smaller-version-of-a-large-company/. Acesso em: 09 mai. 2022.

BLANK, S. **Four Steps to the Epiphany:** Successful Strategies for Products that Win. 5. ed. K & S Ranch, 2013.

BLANK, S. The Customer Development Manifesto: Reasons for the Revolution. **Site**. 31 ago. 2009. Disponível em: https://steveblank.com/2009/08/31/the-customer-development-manifesto-reasons-for-the-revolution-part-1/ . Acesso em: 09 mai. 2022.

BLANK, S. What's A Startup? First Principles. **Site**. 25 jan. 2010. Disponível em: https://steveblank.com/2010/01/25/whats-a-startup-first-principles/. Acesso em: 09 mai. 2022.

BLANK, S. Why the Lean Start-Up Changes Everything. **Site**. mai. 2013. Disponível em: https://hbr.org/2013/05/why-the-lean-start-up-changes-everything. Acesso em: 09 mai. 2022.

BOHEM, B. Making a difference in the software century. **Computer**, v. 41, n. 3, p. 32-38, mar. 2008.

CB INSIGHTS, 406 startup failure post-mortems. **Site**. abr. 2022. Disponível em: https://www.cbinsights.com/research/startup-failure-post-mortem/. Acesso em: 13 mai. 2022.

CB INSIGHTS, The Complete List Of Unicorn Companies. **Site**. mai. 2022. Disponível em: https://www.cbinsights.com/research-unicorn-companies. Acesso em: 09 mai. 2022.

CONBOY, K.; FITZGERALD, B. Toward a conceptual framework of agile methods: A study of agility in different disciplines. *In*: WORKSHOP ON INTERDISCIPLINARY SOFTWARE ENGINEERING RESEARCH. 2004, Newport Beach. **Anais [...]** Newport Beach: WISER, nov. 2004. p. 37-44.

DIDAR, A. F. The Role of Startups and Why Should You Operate in Afghanistan?. **Site**. 26 set. 2016. Disponível em: https://tech.af/the-role-of-startups-and-why-should-you-operate-in-afghanistan/. Acesso em: 09 mai. 2022.

DORNELAS, J. C. A. **Empreendedorismo:** transformando ideias em negócios. 2. ed. Rio de Janeiro: Elsevier, 2001.

FAYAD, M. E.; SCHMIDT, D. Object-oriented application frameworks. **Communications of the ACM**, v. 40, n. 10, p. 32-38, out. 1997

GALVÃO, G. **Hortiprice:** framework de domínio para formação de preço de venda da horticultura. 2020. Dissertação (Mestrado em Computação Aplicada) - Universidade Estadual de Ponta Grossa, Ponta Grossa, 2020.

GONÇALEZ, F. F. **Chatbot para atendimento automatizado**. 2020. Dissertação (Mestrado em Engenharia Informática) - Universidade Fernando Pessoa, Porto, 2020.

GRAHAM, P. Startup = Growth. **Site**. set. 2012. Disponível em: http://www.paulgraham.com/growth.html. Acesso em: 09 mai. 2022.

GROSSMAN-KAHN, B; ROSENSWEIG, R. R. Skip the Silver Bullet: Driving Innovation through Small Bets and Diverse Practices. *In:* INTERNATIONAL DESIGN MANEGEMENT RESEARCH CONFERENCE, 2012, Boston. **Anais [...]** USA: DMI 2012, 8 ago. 2012. p. 815-829.

JOHNSON, R. E. Frameworks = (Components + Patterns). **Communications of the ACM**, v. 40, n. 10, p. 39-43, out. 1997.

JOHNSON, R. E. How to design frameworks. *In:* 8th Conference on Object-Oriented Programming: Systems, Languages and Applications, 1993, Washington. **Anais [...]** Washington: OOPSLA, 1993. p. 567-617.

KTATA, O; LÉVESQUE, G. Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects. *In:* CANADIAN CONFERENCE ON COMPUTER SCIENCE & SOFTWARE. 2009, Montreal, Canada. **Anais [...]** Canada: C3S2E-09, 19-21 mai. 2009.

LEONESSA, N. **Startup Kaizen:** Uma Metodologia Ágil para Desenvolvimento de Software em Startups. 2016. Dissertação (Mestre em Ciência da Computação) – Programa de Pós-graduação em Ciência da Computação, Universidade Federal de São Carlos, Sorocaba, 2016.

LÖBACH, B. **Design Industrial:** Bases para a Configuração dos Produtos Industriais. 1. ed. Rio de Janeiro: Blucher, 2001.

MACHADO, F. C.; GRILO, A. How Can Design Thinking and Lean Startup Improve Waste Collection Systems? *In:* PROCEEDINGS OF THE 5TH NA INTERNATIONAL CONFERENCE ON INDUSTRIAL ENGINEERING AND OPERATIONS MANAGEMENT, Detroit. **Anais [...]** USA: IEON, 2020. 10 Ago. 2020. p. 655-666.

MATOS, S. N.; FERNANDES, C. T. **Um Panorama dos Processos de Desenvolvimento de Frameworks de Domínio**. Instituto Tecnológico de Aeronáutica. Divisão de Ciência da Computação. CTA/ITA-JEC/RP-001/2007.São José dos Campos, 2007.

MISKI, A. Development of a Mobile Application Using the Lean Startup Methodology. **International Journal of Scientific & Engineering Research**, v. 5, n. 1, p. 1743-1748, jan. 2014.

MOSER, R.; ABRAHAMSSON, P.; PEDRYCZ, W.; SILLITTI, A.; SUCCI, G. A Case Study on the Impact of Refactoring on Quality and Productivity in an Agile Team. **Balancing Agility and Formalism in Software Engineering**, Berlim, v. 5082, p. 252–266, set. 2008.

MUELLER-ROTERBERG, C. **Handbook of Design Thinking**. Nov. 2018. Disponível em: https://www.researchgate.net/publication/329310644_Handbook_of_Design_Thinking. Acesso em: 03 fev. 2022.

ORACLE. MySQL. Disponível em: https://www.mysql.com/. Acesso em: 13 mai. 2022.

PAVANI, C.; DEUSTCHER, J. A.; LÓPEZ, S. M. **Plano de Negócios:** Planejando o Sucesso de seu Empreendimento. Rio de Janeiro: Minion, 2000.

PLATTNER, H.; MEINEL, C.; LEIFER, L. **Design Thinking:** Understand – Improve – Apply. Berlim: Springer, 2011.

PRIHODKO, S. The Empirical Importance of Customer Development. **Site**. 29 jun. 2018. Disponível em: https://wearebrain.com/blog/innovation-and-transformation-strategy/the-empirical-importance-of-customer-development/. Acesso em: 09 mai. 2022.

RIES, E. **A startup enxuta:** como os empreendedores atuais utilizam a inovação contínua para criar empresas extremamente bem-sucedidas. São Paulo: Lua de Papel, 2012.

RIES, E. The Lean Startup: Startup Lessons Learned. **Site**. 8 set. 2008. Disponível em: http://www.startuplessonslearned.com/2008/09/lean-startup.html. Acesso em: 09 mai. 2022.

SABBAGH, R. **Scrum:** Gestão ágil para projetos de sucesso. São Paulo: Casa do Código, 2013. p. 529.

SCHWABER, K. SCRUM Development Process. **Paper**. 1996 Disponível em: http://jeffsutherland.com/oopsla/schwapub.pdf. Acesso em: 09 mai. 2022.

SHEPHERD, D. A.; GRUBER, M. The Lean Startup Framework: Closing the Academic–Practitioner Divide. **Entrepreneurship Theory and Practice**, v. 45, n. 5, p. 967–998, set. 2021.

SUTHERLAND, J. **Scrum**. A Arte de Fazer o Dobro do Trabalho na Metade do Tempo. São Paulo: Lua de Papel, 1. ed. 2016.

SUTHERLAND, J. The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework. **Paper**. 30 mar. 2021. Disponível em: http://www.scruminc.com/scrumpapers.pdf. Acesso em: 09 mai. 2022.

TAKEUCHI, H.; NONAKA, I. The New New Product Development Game. **Harvard Business Review**, 1986. Disponível em: https://hbr.org/1986/01/the-new-new-product-development-game. Acesso em: 09 mai. 2022.

YAU, A; MURPHY, C. Is a Rigorous Agile Methodology the Best Development Strategy for Small Scale Tech Startups?. **Paper**. Jan. 2013. Disponível em: https://www.researchgate.net/publication/303968939. Acesso em: 09 mai. 2022.