UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS VII – GOVERNADOR ANTÔNIO MARIZ
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MARIA LUÍZA MENDES BARROS

PROPOSTA DE DESENVOLVIMENTO DE UMA FERRAMENTA WEB PARA ENGENHARIA DE REQUISITOS UTILIZANDO *DESIGN THINKING*

PATOS – PB

2021

MARIA LUÍZA MENDES BARROS

PROPOSTA DE DESENVOLVIMENTO DE UMA FERRAMENTA WEB PARA ENGENHARIA DE REQUISITOS UTILIZANDO *DESIGN THINKING*

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Área de concentração: Design de Experiência com o usuário

Orientador: Prof. Dr. Rodrigo Alves Costa

PATOS - PB
2021

B277p Barros, Maria Luiza Mendes.

Proposta de desenvolvimento de uma ferramenta web para engenharia de requisitos utilizando design thinking [manuscrito] / Maria Luiza Mendes Barros. - 2021.

92 p. : il. colorido.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências Exatas e Sociais Aplicadas , 2021.

"Orientação : Prof. Dr. Rodrigo Alves Costa , Coordenação do Curso de Computação - CCEA."

1. Engenharia de requisitos. 2. Desenvolvimento ágil. 3. Design thinking. I. Título

21. ed. CDD 005.3

Elaborada por José E. da S. Eugênio - CRB - 15/591 BSC7/UEPB

MARIA LUÍZA MENDES BARROS

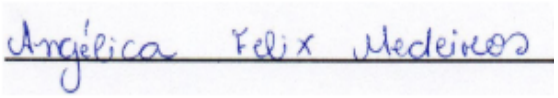
PROPOSTA DE DESENVOLVIMENTO DE UMA FERRAMENTA WEB PARA ENGENHARIA DE REQUISITOS UTILIZANDO DESIGN THINKING

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Estadual da Paraíba, em cumprimento à exigência para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 28/05/2021

BANCA EXAMINADORA

Prof. Dr. Rodrigo Alves Costa
(Orientador)



Prof. Me. Angélica Félix Medeiros
(Examinadora)

Prof. Me. Rômulo Rodrigues de Morais Bezerra
(Examinador)

A Deus, minha família e amigos, pela persistência e pela graça de chegar até aqui, DEDICO.

AGRADECIMENTOS

A Deus em primeiro lugar, por ter permitido tudo isso acontecer.

Aos meus pais Robério e Valdenira, que sempre acreditaram na minha capacidade e me ajudaram a lutar pelos meus objetivos e principalmente a nunca desistir, a vitória é nossa!

A meus avós Antônio e Luzia e toda minha família, a vitória também é de vocês.

A minhas amigas Lorranny, Arminda e Rebeca, por serem sustento e consolo em meio a tudo.

A todos os amigos que a UEPB me presenteou em especial a galera que enfrentou comigo a presidência do Centro Acadêmico, coordenações de evento entre tantas outras loucuras, Mirelly, Sulyn, Matheus, Savio e Josedi, vocês tornaram a academia um lugar melhor e mais fácil, e me proporcionaram momentos inesquecíveis dentro e fora da graduação, obrigada por todo apoio e confiança em mim depositada.

Ao meu maravilhoso orientador Prof. Dr. Rodrigo Alves Costa, que além do papel de professor assumiu também o papel de amigo, acreditando sempre no meu potencial e me fazendo sempre crescer.

A todo o corpo docente do Campus VII pela maravilhosa contribuição na minha vida acadêmica e profissional, e aos técnicos administrativos e funcionários dessa magnífica instituição, a quem fiz grandes e valiosas amizades.

Por fim, a todos que contribuíram de alguma forma seja direta ou indiretamente para a concretização deste sonho.

“Sonhar grande e sonhar pequeno dá o mesmo trabalho”

(Jorge Paulo Lemann)

RESUMO

Empresas de pequeno e grande porte, independente do segmento, necessitam da tecnologia para gerir, produzir e executar suas atividades diárias. Desse modo, a busca por softwares e sistemas que informatizam e operacionalizam atividades humanas nas organizações é ininterrupta, gerando demanda para empresas de desenvolvimento de aplicação e a crescente busca por qualidade por parte dessas, para satisfação dos seus clientes e usuários finais. Neste cenário de concepção de produtos de software, um ponto crucial é a definição dos requisitos de negócio e requisitos técnicos e funcionais, por parte de equipes de projeto, a partir das necessidades dos clientes e usuários. Diante disso, o presente trabalho tem como objetivo apresentar a proposta de desenvolvimento de uma ferramenta web intitulada "SISREQ" para melhorar o processo conhecido como Engenharia de Requisitos que, se otimizado, deve melhorar significativamente no processo de desenvolvimento de software como um todo, gerando otimização de tempo e de trabalho dos desenvolvedores. Para comprovar essa hipótese, foi realizada uma entrevista inicial para coleta de dados, com o objetivo de compreender quais funcionalidades são indispensáveis para a um sistema como este. A proposta utilizou o padrão MVC (*Model-View-Controller*) para a estrutura da arquitetura do software, a ferramenta Jira para detalhar os requisitos do produto, e a ferramenta Figma para a prototipação da ferramenta. Uma segunda entrevista, de análise de resultados, foi então realizada, utilizando o protótipo como resultado preliminar, utilizando o mesmo grupo inicial como grupo de experimentação. Por meio do método NPS (*Net Promoter Score*), pode-se comprovar que a proposta da ferramenta apresentada por este trabalho seria considerada, do ponto de vista dos profissionais entrevistados (desenvolvedores de software) como útil para auxiliar nos seus processos internos, e é considerada indicada quanto a elaboração de um produto para a Engenharia de Requisitos/ Engenharia de Software.

Palavras-Chave: Engenharia de Requisitos. Desenvolvimento Ágil. *Design Thinking*

ABSTRACT

Small and large companies, regardless of their segment, need the technology to manage, produce and perform their daily activities. Therefore, the search for software and systems that computerize and operationalize human activities in organizations is uninterrupted, generating demand for application development companies and the growing search for quality on the part of these, for the satisfaction of their customers and end users. In this scenario of software product design, a crucial point is the definition of business and technical and functional requirements, by project teams, based on the needs of customers and users. Thus, the present work aims to present the proposal for the development of a web tool called “SISREQ” to improve the process known as Requirements Engineering which, if optimized, should significantly improve the software development process as a whole, generating optimization of developers' time and work. To prove this hypothesis, an initial interview was conducted to collect data, in order to understand which features are indispensable for a system like this. The proposal used the MVC (Model-View-Controller) standard for the structure of the software architecture, the Jira tool to detail the product requirements, and the Figma tool for the prototyping of the tool. A second interview, of results analysis, was then carried out, using the prototype as a preliminary result, using the same initial group as the experimentation group. Through the NPS method (Net Promoter Score), it can be proved that the proposal of the tool presented by this work would be considered, from the point of view of the interviewed professionals (software developers) as useful to assist in their internal processes, and is considered recommended when designing a product for Requirements Engineering / Software Engineering.

Keywords: Requirements Engineering. Agile Development. Design Thinking

LISTA DE ILUSTRAÇÕES

LISTA DE GRÁFICOS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O uso da tecnologia tornou-se inerente ao modo de viver das pessoas, com as mais variadas aplicações, tais como *smartphones*, aplicativos de textos que substituíram os blocos de papéis tradicionais, ligações e mensagens instantâneas para pessoas do outro lado do mundo, eletrodomésticos inteligentes e até mesmo aparelhos eletrônicos com acesso à internet.

Silva et al. (2016) afirmam que durante anos o uso da tecnologia era exclusivo para armazenamento e transmissão de dados. Posteriormente, a tecnologia foi introduzida no ambiente social, tornando a execução de tarefas mais prática, rápida e eficiente, e fazendo o uso da tecnologia cada vez mais popular e indispensável na vida humana ou de uma empresa.

Organizações têm investido em computadores e *softwares* a fim de melhorias nos serviços e praticidade. Segundo O'brien e Marakas (2013, apud LONGO; MEIRELLES, 2016, p. 136), "os sistemas e as tecnologias de informação, incluindo sistemas de informação com base na internet, têm papel vital e crescente na administração, portanto, um ingrediente indispensável para o sucesso dos negócios".

Para Sommerville (2011), o mundo atual não poderia existir sem o *software*. O crescimento da demanda por *softwares* que atendam às necessidades individuais de cada empresa é significativo, aumentando dessa forma, também, a busca por empresas de desenvolvimento de software. A área da Ciência da Computação que lida com o desenvolvimento de *software* é chamado de Engenharia de Software (ES), definida por Santiago (2011) como uma área da computação focada na "especificação, desenvolvimento e manutenção de *softwares* aplicando técnicas e métodos com objetivos de melhorias na qualidade e satisfação do usuário".

A ES usa técnicas e métodos para auxiliar o processo de desenvolvimento de *software* com o objetivo de proporcionar melhorias no desenvolvimento e na consequente satisfação do cliente. Tal processo é constituído por quatro fases comuns em qualquer projeto de *software*: Especificação de *Software*, Projeto de Implementação de *Software*, Validação de *Software* e Evolução de *Software*. Uma das etapas iniciais do processo de desenvolvimento de *software* é a Engenharia de Requisitos (ER), à qual dar-se ênfase neste trabalho.

Santiago (2011) define que ER consiste na elicitação, especificação, análise e validação de requisitos do *software*. Requisitos de *software* expressam a ideia proposta pelo cliente que posteriormente é transformada em codificação. Requisitos podem ser definidos como as descrições que o sistema deve fazer, com as permissões e restrições do seu funcionamento (SOMMERVILLE, 2011). Santiago (2011) expressa que “as más conduções dessas atividades tornam projetos de engenharia de *software* criticamente vulneráveis. ” Diante desse contexto, a problemática desta pesquisa envolve a escassez de *softwares* para gerenciamento de requisitos focado no processo de design de experiência com o usuário e para melhor explanação será explorada na próxima seção.

1.1 PROBLEMÁTICA E PROPOSTA DE INVESTIGAÇÃO

A ER é responsável por compreender o funcionamento do *software*, as regras de negócio e a satisfação do usuário final (ROSA et al., 2017). Por se tratar da etapa inicial, é na ER que a ideia do cliente é analisada e transformada em diagramas que servem de base para a equipe de desenvolvimento. Contudo, mesmo com a importância mencionada acima, diversos projetos tendem a fracassar. Ricardo Alves de Melo (2011) cita dados do *Chaos Report* de 1994 que consiste em um estudo do projeto de pesquisa do *Standish Group* sobre taxas de sucessos de projetos de TI e práticas de gerenciamento de projetos. As taxas consistem em: o principal fator do sucesso de um projeto de *software* está na clareza dos requisitos (13%); em relação ao fracasso dos projetos de *software* temos: especificação e requisitos incompletos (12,3%) e mudança nos requisitos e na especificação (11,8%); quanto ao cancelamento do *software* o principal fator é a incompletude dos requisitos (13,1%). Um estudo mais recente - o *Chaos Report* 2006 - mostra que não houve mudança significativas em relação às porcentagens mesmo com o passar dos anos.

A utilização de metodologias centradas no usuário garante maior sucesso em projetos de desenvolvimento de *software*. É o caso da metodologia conhecida como *Design Thinking* (DT), que usa técnicas e serviços de forma inovadora com objetivo de resolver problemas existentes. Liedtka (2011 apud Souza et. al. 2017, p.2) afirmam que “As práticas do *Design Thinking* ajudam as organizações a resolverem problemas complexos, incentivando a inovação e apoiando o processo criativo”. Com

isso, uma das aplicações de projetos de desenvolvimento de software é o desenvolvimento web.

Aplicações para a *web* estão cada vez mais presentes nas organizações, devido à sua praticidade. De acordo com Camargo (2009), “com o avanço da utilização da *Internet*, aumentou de maneira significativa o desenvolvimento de aplicações *web*, devido à portabilidade, acessibilidade e facilidade na utilização e implantação”.

Desse modo, esta pesquisa tem como finalidade a proposta do desenvolvimento de uma ferramenta *web* que implemente um processo do método *Design Thinking* para Engenharia de Requisitos. A ferramenta deve compreender todas as etapas da ER com suporte para aplicações *web*. O objetivo é possibilitar a melhoria na qualidade do desenvolvimento de *software* nas empresas, com mais rapidez e eficiência na etapa inicial do desenvolvimento.

A ideia surgiu em razão da deficiência supracitada em projetos de desenvolvimento, e da hipótese de que tais problemas ocorrem devido à falta de prioridade dada à etapa de especificação de requisitos, gerando, além de atrasos na entrega, a insatisfação do cliente e de outras partes interessadas. Foi realizado uma busca por *strings* de busca em bases científicas por ferramentas semelhantes, e por busca cega em aplicativos similares em ferramentas de pesquisa na internet, conforme explicado no capítulo 3 deste trabalho.

Assim, o desenvolvimento desta ferramenta teve como público-alvo, a princípio, empresas que realizam desenvolvimento de *software*, seja como produto final, seja como parte dos seus processos internos (ou seja, para suportar outras operações). O escopo da aplicação tomou como referência ferramentas já existentes, que se propõem a atender parte dos propósitos estabelecidos pela ER.

O público-alvo da pesquisa, portanto, também foi utilizado para validar a efetividade e usabilidade da ferramenta, em um estudo de caso, cujos resultados são apresentados na fase de análise. Os parâmetros da análise foram levantados por meio de entrevistas e reavaliados da mesma forma após o desenvolvimento de protótipo com fluxo completo, que teve com objetivo as necessidades das empresas de desenvolvimento de *software* em relação à ER, com foco na praticidade do desenvolvimento *web*.

1.2 JUSTIFICATIVAS

De acordo com Costa (2018), a ER é a atividade mais importante em um projeto de desenvolvimento de *software*, tornando-se um pré-requisito básico para o sucesso no desenvolvimento. No entanto, com o objetivo de realizar entregas o mais rapidamente possível, equipes de desenvolvimento tipicamente não dão a devida importância suficiente para essas atividades, ocasionando problemas de rastreabilidade de requisitos (PRESSMAN, 2001 apud PRIKLADNICKI, 2004). Alguns desses problemas são: requisitos sem utilidade real para o negócio, requisitos faltantes/escondidos, requisitos obscuros e requisitos incorretos, gerando retrabalho ou outros custos adicionais para o desenvolvimento como um todo, além da insatisfação do cliente.

Silva (2019) cita, em uma pesquisa de literatura oriunda de duas revisões sistemáticas, três mapeamentos sistemáticos e um estudo, que requisitos incompletos ou escondidos são um dos principais problemas e desafios encontrados em relação ao desenvolvimento de *software*, além de mudanças, documentação insuficiente, arquitetura indesejada, falta de preocupação com requisitos não funcionais, falta de comunicação, problemas de estimativa e falta de suporte do cliente. Em estudos realizados em casos reais, constatou-se que 50% das necessidades de retrabalho em projetos de desenvolvimento de *software* ocorrem devido a falhas no processo da ER (GASTALDO; MIDORIKAWA, 2003).

Esta pesquisa justifica-se, portanto, na medida em que oferece contribuições para a área da Ciência da Computação, mais particularmente para a grande área de Engenharia de *Software*, já que discute soluções para o problema supracitado e propõe um estudo de caso por meio de um aplicativo que suporta o processo de desenvolvimento de *softwares*. O objetivo é que o uso da ferramenta aqui proposta ofereça uma melhora significativa na ER em projetos de desenvolvimento de *software*, com a facilidade de ser uma aplicação voltada para a *web*.

1.3 OBJETIVOS

A seção a seguir aborda os objetivos gerais e específicos que compõem este trabalho.

1.3.1 Objetivo Geral

O presente trabalho tem como objetivo geral propor o desenvolvimento de uma ferramenta *web* para Engenharia de Requisitos tornando-a flexível e orientada a requisito, alinhando o seu processo de elicitação de requisitos com o processo de *Design Thinking.*

1.3.2 Objetivos Específicos

Os objetivos específicos foram:

- Analisar e identificar as ferramentas para auxílio na Engenharia de Requisitos já existentes;
- Identificar os requisitos necessários para a construção da ferramenta proposta de modo que atenda aos processos da Engenharia de Requisitos por meio de entrevistas com grupos de foco e com base em ferramentas semelhantes;
- Desenvolver um processo orientado a requisito de modo que a elicitação de requisitos utilize um processo de *Design Thinking*;
- Desenvolver um protótipo da ferramenta *web* baseado no processo anterior;
- Validar o protótipo desenvolvido juntamente a empresas de desenvolvimento de *software* da cidade de Patos - PB, por meio de protótipos de alta fidelidade e fluxo contínuo.

1.4 TRABALHOS RELACIONADOS

Nesta seção, será apresentada algumas ferramentas pesquisadas e trabalhos relacionados que serviram como base inicial para a proposta realizada por este trabalho. Para cada uma das ferramentas a seguir, destacam-se as suas características técnicas principais, propósitos, pontos positivos e negativos. Com efeito, ao levantar trabalhos relacionados no início do trabalho, foi possível compreender como uma nova proposta pode complementar as já existentes e ser solução otimizada.

1.4.1 RE-Tools

Em Supakkul e Chung (2012) é apresentada a ferramenta RE-Tools, que consiste em um kit de ferramentas de código aberto usando uma modelagem UML (StarUML). O kit suporta raciocínio qualitativo (original do NFR Framework) e também oferece suporte às principais ferramentas de modelagem de requisitos em diversas notações (I*Framework, KAOS, *Problem Frames*).

Como ponto positivo da ferramenta é possível mencionar as várias notações diferentes, oferecendo ao usuário do kit de ferramentas auxílio durante todas as etapas do processo de desenvolvimento de *software*. Por exemplo, o kit propõe a notação StarUML, uma extensão da modelagem UML para código aberto. Cada notação oferece aspectos exclusivos com funcionalidades voltadas para esse tipo específico de projeto.

Como ponto negativo pode-se citar que para instalação da ferramenta é necessário instalar também a ferramenta StarUML, o que está disponibilizado exclusivamente para o Sistema Operacional (SO) *Windows*. A ferramenta também é voltada para equipes de projeto na fase de arquitetura, sem oferecer suporte na facilitação da experiência do usuário que deseje delinear suas necessidades.

1.4.2 SIGERAR

Grande e Martins (2006) apresentam a SIGERAR, uma ferramenta *web* desenvolvida para automatizar o processo de gerenciamento de requisitos, com objetivos de coletar, armazenar e manter os requisitos durante todo o ciclo de vida do *software*. A ferramenta deve permitir o gerenciamento de mudanças e o rastreamento dos relacionamentos entre os requisitos e os documentos produzidos.

Como ponto positivo, a ferramenta consiste no tratamento de rastreabilidade de requisitos no processo de alteração, permitindo a atribuição de valores de riscos e prioridade aos requisitos envolvidos, de modo que não foi mencionado qualquer problema existente após a análise de viabilidade.

Como ponto negativo é possível mencionar que a ferramenta não realiza o acompanhamento do gerenciamento de versões de *releases* e *builds* em função dos requisitos, o que é inclusive uma das finalidades pretendidas nos objetivos da mesma. Os autores citam que uma das limitações da ferramenta é que, após a

análise de viabilidade do projeto a funcionalidade de gerenciamento de versões não é satisfatória.

1.4.3 OpenReq

Em Grings e Sayão (2009) é apresentada o OpenReq, uma ferramenta *web* de código aberto com funcionalidades para gerenciamento de requisitos e auxílio no controle das demais etapas de um processo de construção de *software*.

A ferramenta proporciona as seguintes funcionalidades: Administração de Projetos, permitindo o acompanhamento de projetos já desenvolvidos com a ferramenta, Controle de Mudanças, que permite o gerenciamento das alterações ocorridas, Controle de Versões, que permite o gerenciamento e controle das diversas versões geradas a partir das modificações e o Gerenciamento de *Plugins*, que permite a adição e configuração de módulos que podem ser adicionados a ferramenta.

Como ponto positivo é possível mencionar o acesso *web* da ferramenta garantindo mais praticidade aos usuários; o servidor de armazenamento de dados com uso de um *framework* de persistência assegurando o uso de diferentes Sistema de Gerenciamento de Banco de Dados (SGBD) de forma transparente e a possibilidade de mudança de idiomas da ferramenta.

A ferramenta se propõe a atender vários propósitos, e aparentemente suas funcionalidades parecem estar mais focadas no gerenciamento de projetos, gestão da configuração e versionamento de código e documentos, fazendo com que ela falhe no gerenciamento de requisitos. Graças a sua característica *OpenSource*, no entanto, as equipes de projeto podem desenvolver novas funcionalidades que mapeiem essas características as suas necessidades por engenharia e gestão de requisitos.

1.4.4 Rational RequisitePro

Em Farache (2007) é apresentada a RequisitePro, uma ferramenta desenvolvida pela IBM com objetivo de auxílio na gerência de requisitos de *software*. A ferramenta organiza os requisitos em uma base de dados multiusuários em relação à prioridade, status e características definidas pelo usuário, dispondo de integração

com o *Microsoft Word*, e com semelhanças com as ferramentas CaliberRM e DOORS.

Como ponto positivo é importante destacar a integração com o *Microsoft Word*, a criação de visões de matriz de rastreabilidade, permitindo a visualização de quais funcionalidades foram implementadas ou não, e a percepção de quebra de relacionamento onde a ferramenta avisa automaticamente quando o requisito for modificado.

Como ponto negativo, destaca-se que a ferramenta só permite a importação de artefatos da Rational e Borland, não permitindo estender para outras ferramentas. Ela também é uma ferramenta de facilitação e armazenamento de requisitos, sem muita preocupação com o gerenciamento de requisitos de *software* em função do ciclo de vida de desenvolvimento de *software*.

1.4.5 Ferramenta Proposta

Após análise detalhada das ferramentas citadas, feita através de pesquisas bibliográficas, observou-se que as ferramentas buscam resolver os mesmos tipos de problema com respeito ao gerenciamento de projetos de *software* e engenharia de requisitos. Assim, em termos de funcionalidade, o foco do estudo se deu nas ferramentas mais mencionadas nos artigos acadêmicos selecionados na pesquisa detalhada, conforme o capítulo 3 deste trabalho.

A ferramenta proposta neste projeto não visa negligenciar as características atualmente existentes destas ferramentas e pretende oferecer suporte à todas as etapas da ER (elicitação, especificação, análise e validação). No entanto, este estudo entende ser necessário fornecer aos usuários do processo de engenharia de requisitos uma experiência customizável durante todo o ciclo de vida com respeito a requisitos, já que a ER é uma atividade dinâmica conforme explicado na seção 2.2.1. Assim, ferramentas específicas de *UX Design* e do *Design Thinking*, conforme explicado na seção 2.3 e 2.3.1, podem ser valiosas para facilitar a ER como um todo.

1.5 ESTRUTURA DO TRABALHO

O presente trabalho apresenta sete capítulos e está organizado da seguinte maneira: no capítulo 1 foi apresentada uma introdução ao estudo, incluindo a

contextualização do problema, justificativa, objetivos e os trabalhos relacionados; no capítulo 2 é apresentado o referencial teórico da pesquisa; no capítulo 3 é apresentada a metodologia utilizada na pesquisa; no capítulo 4 são apresentados os materiais e métodos utilizados no desenvolvimento da pesquisa; no capítulo 5 é apresentado o desenvolvimento da pesquisa contendo a entrevista de motivação, protótipo da ferramenta e a arquitetura proposta; no capítulo 6, é apresentada a análise da pesquisa contendo a entrevista de feedback e os resultados; finalmente, no capítulo 7, são apresentadas as considerações finais, contribuições e trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo, será apresentada a fundamentação teórica com base em trabalhos e áreas que contribuem para o norteamento desta pesquisa.

2.1 ENGENHARIA DE SOFTWARE

No início da era digital, as demandas em pesquisa e desenvolvimento eram concentradas na melhoria do poder de processamento dos processadores e *hardwares* (parte física do computador).

Com o avanço das tecnologias e com a crescente demanda por sistemas nas organizações, as soluções de *software* (parte lógica do computador) passaram a implementar processos das empresas e se tornaram uma prioridade para os departamentos de Tecnologia da Informação. Com o surgimento da necessidade de aplicativos que oferecessem suporte a alguma aplicação do mundo real, o desenvolvimento de *software* passou a ser mais comum e mais difundido.

De acordo com Sommerville (2011, p. 4), "*softwares* são programas de computadores e documentação associada". Pressman (2011, p. 32), por sua vez, conceitua *softwares* como "instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados". *Software* compreende um conjunto de instruções lógicas que realizam uma determinada função como os programas de computadores, sistemas operacionais, jogos, aplicativos de celulares.

Ao assumir o papel de aplicativos pessoais, os *softwares* adquiriram papel fundamental em todas as áreas da sociedade, como afirma WAZLAWICK, R (2019), que diz que a medida que se percebeu a necessidade de computadores mais flexíveis, surgiu o *software*, um conjunto de instruções que fazem a máquina produzir algum tipo de processamento. Novas necessidades surgiram e a demanda por *softwares* cresceram, desse modo, é raro uma empresa não possuir soluções computacionais com algum sistema que automatize os seus processos de negócio ou ainda algum indivíduo que não possua um dispositivo baseado em *software*, como um *smartphone*, ou um *tablet*, por exemplo, e seus diversos aplicativos, para uso diário.

Com o aumento da demanda, os desenvolvedores passaram a conceber esses *softwares* de maneira rápida, para Sommerville (2011, p. 38), “o desenvolvimento e entrega rápidos são, portanto, o requisito mais crítico para o desenvolvimento de sistemas de *software*.”. Com isso, o ambiente de mercado fez surgir demandas cada vez mais complexas, desse modo, diversas empresas optam entre a qualidade e o compromisso com os requisitos solicitados por um desenvolvimento mais rápido e sem garantia (SOMMERVILLE, 2011), gerando insatisfação nos clientes.

Com isso, várias técnicas foram desenvolvidas e aprimoradas com o intuito de aperfeiçoar o desenvolvimento dos *softwares*, com o objetivo de gerenciar a complexidade advinda das necessidades de clientes e partes interessadas inerentes ao ambiente de mercado de *software*. A Engenharia de *Software* é uma dessas técnicas, e se trata de uma área da computação voltada para criação de *softwares* e sistemas de computadores, buscando gerenciar os processos de desenvolvimento para entrega de *softwares* com mais qualidade e previsibilidade. Como afirma Pressman (2011, p. 39), “*software*, em todas as suas formas e em todos os seus campos de aplicação, deve passar pelos processos de engenharia”.

O termo Engenharia de *Software* (ES) foi utilizado pela primeira vez em 1968 em uma conferência da Organização do Tratado do Atlântico Norte (OTAN), denominada *Conference on Software Engineering* (PARNAS 1997 apud. TEIXEIRA; CUKIERMAN 2005), na qual os organizadores acreditavam que utilizando o termo ‘Engenharia’ alcançariam mais interesse em torno do assunto.

Engenharia de *Software* é compreendido pela IEEE (1990) como:

(i) a aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção de *software*, isto é, a aplicação da engenharia ao *software*;

(ii) o estudo das abordagens, como é (i).

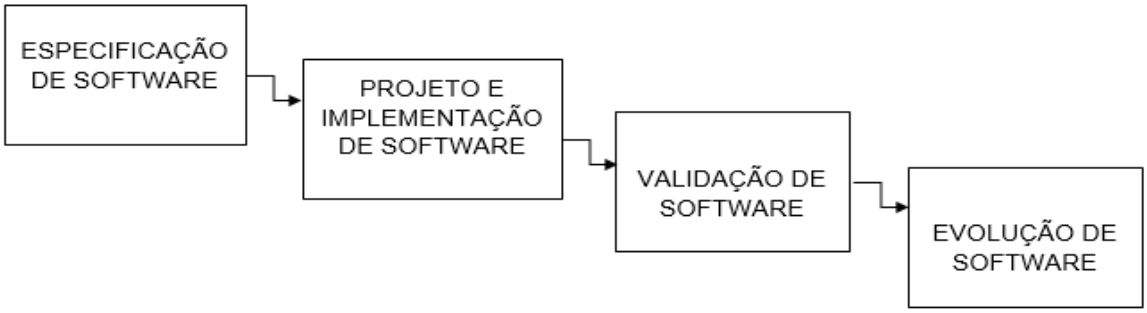
Por sua vez, Sommerville (2011, p. 5) afirma que a ES “é uma disciplina de engenharia cujo foco está em todos os aspectos da produção de *software*, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado”. ES consiste no uso de tecnologias com o objetivo de facilitar todo o processo de desenvolvimento de *software* através de um sistema de etapas que precisam ser executadas para que o produto final seja desenvolvido de maneira ágil e com qualidade.

Esse sistema de etapas da ES é conhecido como “processo de *software*”, que consiste em uma sequência de atividades que devem ser seguidas para o desenvolvimento de um produto de *software*.

2.1.1 Processo de software

Qualquer processo de *software* é constituído de 4 atividades fundamentais comuns a todo processo de *software*: especificação de *software*, projeto e implementação de *software*, validação de *software* e evolução de *software* (SOMMERVILLE, 2011), como mostra a figura 1:

Figura 1 - Processo de Software



Fonte: Autoria própria (2020)

De acordo com Sommerville (2011) as 4 etapas da ES são detalhadas abaixo:

- **Especificação de *software*:** ocorre toda a parte inicial para a criação do *software*, é aqui onde acontece o primeiro contato da equipe de desenvolvimento com o cliente ou usuário final. Nessa etapa se obtém os requisitos do sistema, que são características particulares do *software* que será desenvolvido. Essas características geralmente são obtidas através de entrevistas ou questionários, que posteriormente serão analisadas e validadas para só depois seguir para a próxima etapa. Aqui também se define todas as funcionalidades e restrições do *software*, que serão estabelecidas após a validação dos requisitos obtidos.

- **Projeto e implementação de *software*:** fase durante a qual se desenvolve do sistema, quando os desenvolvedores realizam a codificação do *software* de acordo com as especificidades definidas na etapa anterior. Desse modo, nesta etapa ocorre a prototipação do *software* (exemplo: *design* que simula o produto a ser desenvolvido), que ajuda a entender na prática a ideia do cliente, posteriormente, ocorre a codificação do *software* seguindo os requisitos já estabelecidos, com as funcionalidades e restrições também já especificadas anteriormente.
- **Validação de *software*:** acontecem os testes do *software* e a busca por defeitos no software desenvolvido. Nesta etapa a equipe de independente valida que o *software* atende a todos os requisitos já estabelecidos na etapa de especificação de *software* e aponta inconformidades.
- **Evolução de *software*:** acontecem as atualizações e manutenções do *software*, que podem ser solicitadas pelo cliente ou por outras partes interessadas. Assim sendo, a equipe deve desenvolver o *software* de maneira flexível, para que seja possível adaptá-lo sempre que solicitado.

Outras atividades inerentes às atividades principais são necessárias para o desenvolvimento de *softwares* e sistemas, não se limitando apenas às quatro citadas anteriormente, variando de organização para organização (SOMMERVILLE, 2011). Essas “subatividades” complementam as etapas do processo de *software*. Esse é o caso, justamente, da atividade de Engenharia de Requisitos, que é inerente à etapa de Especificação de *Software*. Por sua vez, a atividade de testes unitários é uma subatividade da etapa de validação de *software*.

Um fator de extrema importância no desenvolvimento de *software* que não recebe a devida importância é a documentação de *software*. De acordo com De Souza et. al. (2007) “a documentação de software tem grande importância, pois pode influenciar tanto na qualidade quanto na produtividade” com isso, equipes de desenvolvimento tendem a dar prioridade a parte de codificação do *software*, dando uma menor prioridade à documentação, esquecendo-se de quão significativo é a documentação de *software.* Após a Elicitação de Requisitos inicial, mantê-los

atualizados é normalmente entendido como documentação pelas equipes de desenvolvimento sendo, por vezes, negligenciado.

Souza et. al. (2007, p. 2) afirma que “a falta de documentação pode representar um risco para os projetos de *software*, principalmente na fase de manutenção.”. Não dá para assegurar que o conhecimento sobre o projeto estará presente na cabeça da equipe de desenvolvimento, mesmo após a conclusão do projeto quando a equipe já tenha sido dissolvida (SOUZA ET. AL., 2007).

Conforme mencionado anteriormente, um dos pontos iniciais para o projeto de *software* é entender os requisitos do sistema que consiste em compreender o que de fato o cliente deseja.

2.2 REQUISITOS

Um requisito é compreendido pela IEEE (1990) como:

i) uma condição ou capacidade necessária por um usuário para resolver um problema ou atingir um objetivo;

ii) uma condição ou capacidade que deve ser cumprida ou possuída por um sistema ou componentes de sistema para satisfazer um contrato, padrão, especificação ou outros documentos formalmente impostos;

iii) uma representação documentada de uma capacidade de condição como em i) ou ii).

Como se pode perceber, os requisitos de um projeto não são necessariamente atrelados a projetos de desenvolvimento de *software*. O conceito de requisito se aplica a qualquer esforço no qual uma parte interessada tenha necessidades. Em projetos de *software*, no entanto, definir os requisitos de um *software* representa compreender a ideia do cliente e transformá-la em codificação, com as funcionalidades solicitadas, de acordo com o que foi desejado. Uma compreensão equivocada dos requisitos pode resultar em problemas de atualizações ou insatisfação do cliente após a entrega.

Para Sommerville (2011), requisitos de *software* são costumeiramente divididos em requisitos funcionais e não funcionais.

- **Requisitos funcionais:** constituem as características do sistema, as funcionalidades, os serviços que o sistema deve fornecer, tudo o que o

sistema deve ou não fazer, as entradas e saídas. Podem ser classificados em requisitos gerais e requisitos específicos.

- **Requisitos não funcionais:** consistem nas propriedades do sistema e estão relacionados a aplicação do *software* em relação a desempenho da máquina, confiabilidade e segurança. Podem ser classificados em requisitos do produto, requisitos organizacionais e requisitos externos.

A coleta dos requisitos está normalmente presente nas etapas iniciais em um processo de desenvolvimento de *software*, visto que o que foi coletado deve ser transformado em um plano para design de arquitetura, como um conjunto de modelos ou protótipos do que se quer desenvolver, e posteriormente esse plano deve ser codificado na forma de *software*. Essa etapa inicial é denominada de elicitação de requisitos, e faz parte da disciplina denominada de Engenharia de Requisitos (ER), que será descrita na seção 2.2.1, a seguir.

2.2.1 Engenharia de Requisitos

Segundo Espindola et al. (2004), diversos projetos possuem a tendência de fracassar por não compreenderem as necessidades estabelecidas pelo cliente, sem esquecer também dos prazos e orçamentos estimado. Um dos principais motivos para o fracasso baseia-se em relação ao déficit existente na Engenharia de Requisitos.

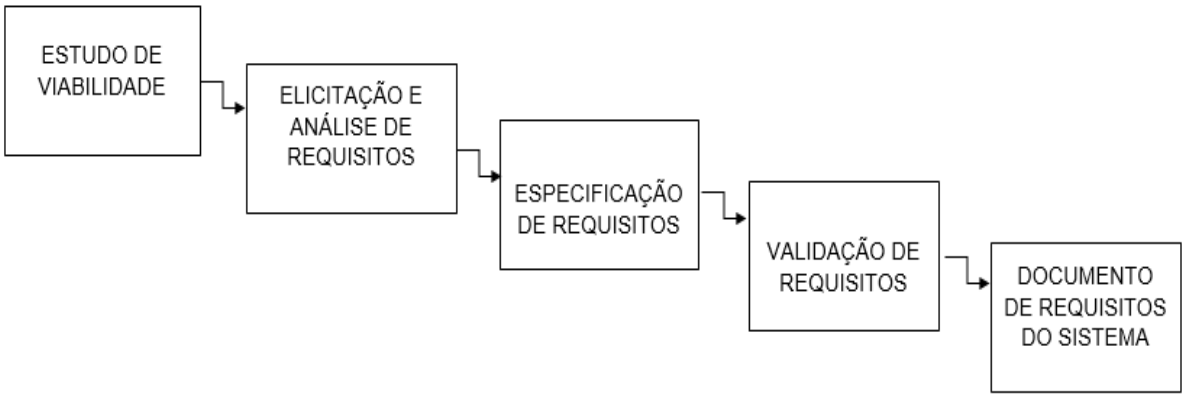
A ER é uma das etapas iniciais do processo de *software*, uma fase que é também conhecida como especificação do *software*. Coletar corretamente os requisitos é essencial para impedir problemas futuros no *software*, como atualizações constantes a pedido do cliente, alterações em funcionalidades essenciais do *software*, e insatisfação do cliente. Assim, embora seja uma fase formal da especificação, percebe-se que atividades de ER estão presentes ao longo de todo o desenvolvimento do projeto de *software*, sendo notória sua importância. Para ratificar isso, Pressman (2011, p. 127) afirma que ER constitui-se em “uma ação de engenharia de *software* que se inicia durante a atividade de comunicação e continua na de modelagem”.

De acordo com Pressman (2011, p. 127), “a engenharia de requisitos constrói uma ponte para o projeto e para a construção”. É na ER que é possível entender

aquilo que o cliente quer e precisa, compreender a sua vontade e iniciar o processo de transformação desta necessidade em código-fonte. Todavia, a realidade é diferente pois é comum a existência de inconsistências no processo da ER, o que causa diversos problemas no projeto tais como atrasos na entrega, aumento de custos e insatisfação do cliente (SOMMERVILLE, 2011).

De acordo com Sommerville (2011), o processo de ER é constituído por 5 etapas: estudo de viabilidade, elicitação e análise de requisitos, especificação de requisitos, validação de requisitos e documento de requisitos do sistema, como mostra a figura 2:

Figura 2 - Processo da Engenharia de Requisitos



Fonte: Autoria própria (2020)

Segundo Sommerville (2011) as 5 etapas da ER são detalhadas abaixo:

- **Estudo de viabilidade**: nesta fase é definida a possibilidade de prosseguir ou não com o projeto proposto para o cliente. Desse modo, a equipe de desenvolvimento se reúne, e é verificada a viabilidade do desenvolvimento daquele *software*. A importância dessa fase consiste em analisar as etapas que compõem o desenvolvimento do *software*, bem como os recursos existentes e necessários para que o trabalho seja desenvolvido para atender as necessidades do cliente.
- **Elicitação e análise de requisitos:** nesta fase são estabelecidas as necessidades do cliente em forma de requisitos. Inicialmente, o cliente detalha todo o *software* à equipe de desenvolvimento (esse detalhamento pode ser através de entrevistas, questionários e protótipos), e esse detalhamento estabelecido consiste na elicitação

dos requisitos. Posteriormente, a equipe de desenvolvimento transforma a ideia do cliente em requisitos de sistemas e de negócios - essa transformação consiste na análise de requisitos.

- **Especificação de requisitos:** nesta fase acontece a conversão dos requisitos coletados e analisados na fase anterior para a linguagem padrão compreendida pela equipe de desenvolvimento. Essa linguagem padrão pode consistir em diagramas de modelagem e protótipos que auxiliarão na codificação do *software*.
- **Validação de requisitos:** nesta fase ocorre a verificação dos requisitos estabelecidos, é analisado se o que foi detalhado pelo cliente no início corresponde ao produto final estabelecido. Isso ocorre em função da Especificação, e antes do início da codificação do *software*.
- **Documento de requisitos do sistema:** na última fase da ER é elaborado um documento contendo tudo o que foi coletado nas fases anteriores, o documento irá conter os questionários, diagramas, entrevistas, e quaisquer outros documentos necessários para garantir a implementação correta do *software*.

Vale salientar que mesmo com a ER sendo realizada, os *softwares* necessitarão de atualizações e modificações futuras, como afirma Sommerville (2011, p. 77) “uma vez que um sistema tenha sido instalado e seja usado regularmente, inevitavelmente surgirão novos requisitos”. Desse modo, a equipe de desenvolvimento deve pensar no Gerenciamento de Requisitos que consiste na compreensão e controle de mudanças dos requisitos de *softwares* (SOMMERVILLE, 2011).

A ER tem papel significativo na ES, pois é nela onde são listados aspectos de extrema importância para o *software* como: funcionalidades, permissões e restrições do sistema, cores, imagens, portabilidade e armazenamento. Fazer isso de forma superficial ocasiona uma série de atualizações desnecessárias durante o processo de desenvolvimento, além da insatisfação do cliente, que podem ser evitadas quando se tem conhecimento das necessidades do cliente.

2.3 UX DESIGN

Desenvolver *softwares* inicia-se antes da coleta de requisitos e especialmente da sua codificação. Pensar nas necessidades de um usuário que utiliza um *software* enquanto se levanta requisitos é um fator que contribui significativamente para o sucesso do produto desenvolvido. Neste sentido, é importante compreender que essas necessidades vão além de aspectos relacionados às funções disponibilizadas pelo *software* ou por seu *design*, mas compreende também aspectos relacionados à experiência do usuário ao utilizá-lo, inclusive aspectos afetivos, significativos e valiosos da Interação Humano-Computador (IHC) (BARBOSA; SILVA, 2010) e também de propriedade do produto. A área que estuda esse conjunto de aspectos é denominada de *Design UX* (do inglês User Experience, ou Experiência do Usuário).

UX (Experiência de Usuário) existe desde o primeiro contato do ser humano com algum objeto material. Esse contato inicial ocasiona uma interação de aprendizado, gerando a experiência em si (PINHEIRO, 2016). Com o uso frequente desses objetos é desenvolvida uma experiência, apesar de ser subjetiva e variar de usuário para usuário. De acordo com Falavigna (2015, p. 12) “a Experiência do Usuário envolve o estudo das sensações e emoções que os usuários vivenciam ao utilizar um produto de tecnologia”.

O termo *UX Design* foi criado por Don Norman nos anos 90, enquanto ele trabalhava na *Apple*. Segundo Norman (2004 apud Ravanello et. al. 2016, p. 2), “a experiência do usuário auxilia na definição da forma de um produto, do seu comportamento e conteúdo, assegurando a coerência". Essa coerência deve se dar também de forma consistente, em todas as dimensões de projeto.

Um conceito importante no *Design UX* é o processo pelo qual os usuários formam experiências. Ao encontrar um produto, o usuário forma uma impressão que evolui ao longo do tempo. No processo, a percepção, ação, motivação e cognição do usuário se integram para formar uma história memorável e coerente: chamada "experiência do usuário". Esse processo suscita respostas emocionais, que em grande parte determina como a experiência será considerada.

Design UX “é uma metodologia focada em melhorar a satisfação do usuário através de melhoras na usabilidade, acessibilidade e contentamento de interações de um produto “ (KUERTEN, R.; PALMA, J. G., 2019, p. 3). *Design UX* projeta o produto com base no comportamento humano a fim de que com o uso gere a satisfação do usuário, e o objetivo é tornar a relação do usuário com o produto mais amigável e comum.

Freitas (2017) define que o processo de *Design UX* é dividido em três partes:

- **Design centrado no utilizador:** fase inicial onde é analisado e projetado como os clientes irão utilizar o produto. Nesta fase é feito um estudo da concorrência em relação aos produtos disponibilizados, a criação de personagens fictícios que ajudarão na percepção das necessidades dos usuários finais, e também é criado um diagrama de fluxo de utilizador para representar como será a realização das tarefas do usuário.
- **Utilização da informação obtida na fase anterior:** nesta fase é utilizado os dados coletados na fase anterior para confecção de *wireframes* que são diagramas de baixa fidelidade de como será o *layout* final da aplicação, e também diagramas que simulam as ações do usuário.
- **Recolha de feedbacks e testes de usabilidade:** nesta fase são recolhidos os *feedbacks* de usabilidade do cliente juntamente com os testes de usabilidade que ajudam na resolução dos problemas de aplicação existentes.

Por conseguinte, o *Design UX* é centrado no desenvolvimento do produto de acordo com a experiência do usuário, com objetivo de garantir-lhes satisfação após o uso, desse modo essa satisfação existirá após a usabilidade do produto. A Internacional Standards Organization - (ISO 9241-11) caracteriza a usabilidade como “a eficácia, eficiência e satisfação com que os utilizadores conseguem atingir objetivos específicos".

Ou seja, na eficácia é observado se o utilizador conseguiu concluir a atividade proposta com resultados positivos. Na eficiência, é analisada a quantidade de esforços e recursos feito pelo utilizador para realização da atividade. Na satisfação, é observado o nível de conforto do utilizador na realização da atividade (FREITAS, 2017).

Assim, pode-se dizer que *Design UX* é sobre o ato e a saída do *design*. Isso inclui *design* de interação, *design* visual e arquitetura de informações. O *UX Design* em sua essência não define um processo específico para que isso ocorra. Existem diversas metodologias e processos propostas para a realização de *Design UX*, na próxima seção veremos um processo que pode auxiliar na resolução de problemas e que pode ser aplicada para *Design UX*, denominada *Design Thinking*.

2.3.1 Design Thinking

A Engenharia de *Software* (ES) é bem-sucedida quando existe uma parceria colaborativa entre a equipe de desenvolvimento e o cliente, o não envolvimento dos usuários no processo de desenvolvimento de *software* pode ocasionar diversos problemas. Tais problemas podem ser identificados pela falta ou envolvimento reduzido dos usuários no processo de desenvolvimento de *software*. (SOUZA et. al., 2017).

Compreender, analisar, e identificar juntamente com o usuário suas ideias e pensamentos em relação ao *software* em desenvolvimento gera, além da satisfação do cliente, a garantia de sucesso no final. Algumas metodologias existem com o objetivo de resolução dos problemas existentes em relação ao desenvolvimento de *software*, com o uso da criatividade e inovação atendendo ao que foi idealizado pelo cliente, como por exemplo o *Design Thinking*.

Design Thinking (DT) consiste em uma metodologia utilizada por equipes de projeto com o propósito de desenvolver produtos e serviços que atendam às necessidades do usuário de forma inovadora. Azenha e Fleury (2018, p. 2) afirmam que o “DT caracteriza como um conjunto de práticas capazes de elicitar as necessidades do usuário, gerar protótipos rápidos com o objetivo explorar as várias possibilidades inovadoras de solução”.

Em busca de inovação, as empresas investem em novas soluções com finalidade de satisfação dos clientes, bem como o destaque entre os concorrentes. DT possui uma abordagem focada em inovação, através de métodos utilizados por designers, introduzindo técnicas que estimulam os aspectos cognitivos, emocionais e sensoriais para resolver problemas de negócio. DT utiliza formas de pensamento pouco convencionais, tais como pensamento abdutivo, que estimulam formas criativas de construção de soluções. Trata-se da criação de questionamentos através da percepção dos problemas identificados. Dessa forma, a solução não é criada para o problema, ela se encaixa no problema (VIANNA ET. AL., 2012).

Como todo processo, DT possui fases para sua realização. Brown (2008 apud Ramírez; Zaninelli, 2017) estabelece três fases para o processo do DT, conforme mostra a figura 3, abaixo.

Figura 3 - Processo do Design Thinking

Fonte: Autoria própria (2020)

- **Inspiração:** na fase inicial faz-se necessário estabelecer uma expectativa de sucesso, desenvolver um plano de implementação e analisar o problema a ser resolvido, trabalhando com os recursos e áreas disponíveis ao redor.
- **Ideação:** nessa fase é criado rascunhos da ideia proposta e a partir desse rascunho é criado um marco de referência para implementar o pensamento do DT. Posteriormente esses rascunhos são levados ao usuário para que seja analisado a UX (Experiência de usuário), após isso é necessário a criação de protótipos que devem ser testados com os usuários novamente, esses protótipos auxiliarão a equipe de desenvolvimento na codificação.
- **Implementação:** na última fase é executado o que foi desenvolvido nas fases anteriores pela equipe de desenvolvimento de *software*/produto.

Ramírez e Zaninelli (2017) ressaltam a importância da prototipação, visto que ela transforma a ideia inicial em uma ‘realidade’ tangível que a equipe multidisciplinar consegue testar, repensar e melhorar. É com a prototipação também que é possível exemplificar a ideia para o usuário e com isso eliminar erros e assegurar o sucesso da inovação.

Com efeito, *Design Thinking* é um processo. Trata-se de aplicar uma maneira específica de pensar a uma situação. *Design Thinking* não consiste em resolver um problema específico, mas em encontrar o problema certo a ser resolvido. Pode ser aplicado amplamente a áreas que tradicionalmente estão fora do nicho do "*Design*" ou do *Design UX* porque a saída do *Design Thinking* não é necessariamente um design tradicional.

Compreender as necessidades dos usuários é uma tarefa bastante complicada enfrentada pelas empresas, com isso, (MATHIAS, L., 2018) afirma que grandes empresas optaram por adaptar os seus produtos aos consumidores (e não

o contrário), e fizeram isso com a aplicação do DT. O autor cita por exemplo a Netflix, grande plataforma de *streaming*, que aplicando princípios básicos do DT compreendeu que para oferecer um melhor serviço a seus usuários precisaria entender os hábitos e padrões de consumo de cada usuário. Para isso a plataforma conta com um algoritmo que norteia os lançamentos de filmes e séries depois de uma grande análise em temas de interesse daquele determinado usuário. O autor ainda cita outras grandes empresas que fazem o uso do DT, como por exemplo: Natura, Havaiana, Totvs.

2.4 DESENVOLVIMENTO DE SISTEMAS

A tecnologia vem ganhando espaço em todos os ambientes da sociedade. Com isso, é comum o uso de computadores e sistemas automatizados que auxiliam as atividades cotidianas das empresas em todos os ramos de negócio. Centenaro (2014, p. 22) afirma que "a atualidade do mercado está contida num mundo globalizado".

O desenvolvimento de sistemas é feito por equipes de desenvolvimento de *software* composta por diversos tipos de profissionais, tais como programadores, analistas de sistemas, engenheiros de *software*, e analistas de garantia de qualidade. Desenvolvimento de *software* refere-se a uma solução criada a partir de uma ideia proposta pelo cliente através de uma sequência de códigos por meio de uma linguagem de programação. Os *softwares* podem ser desenvolvidos para ambientes *desktop* ou *web*, onde a partir da necessidade do cliente é escolhido o melhor ambiente de desenvolvimento.

Softwares para ambientes *desktop* são desenvolvidos para usar apenas no computador, necessitam de instalação e não é necessária conexão com a internet. *Softwares* para ambientes *web* podem ser usados em computadores, celulares e *tablets*, não necessitam de instalação, porém é necessário o acesso à internet. As aplicações *desktop* vêm perdendo espaço para as aplicações *web* em relação a versatilidade dos *softwares web* e sua fácil manutenção.

O que diferencia o desenvolvimento de *software web* e *desktop* é em relação ao público-alvo, às formas de acesso e sua manutenção (CENTENARO, 2014). A seguir será detalhado o desenvolvimento para sistemas *web*.

2.4.1 Desenvolvimento Web

Durante o período da guerra fria, a ARPA (empresa criada para pesquisa e desenvolvimento de novas tecnologias para defesa nacional) desenvolveu em 1962 com objetivo de compartilhamento de informações entre as bases militares, um modelo de troca e compartilhamento de informações (FARREL, 2005 apud CAMARGO, 2009), mais tarde conhecida popularmente por internet.

Posteriormente, em 1993 a internet começou a repercutir através do primeiro visualizador gráfico de páginas www (*Word Wide Web*), tornando-se popularmente conhecida nos anos seguintes com a expansão pelo mundo (CASTELLS, 2003 apud CAMARGO, 2009). Hoje, computadores e *smartphones* se comunicam por meio de *softwares* aplicativos que permitem essa funcionalidade, indústrias de carros, TVs, e eletrodomésticos também adotaram o uso da internet em seus produtos.

A conexão de um computador a outro é feita por mensagens de comunicação entre inúmeros computadores. O primeiro transmite que deseja fazer uma comunicação com outra máquina, por meio de protocolos essa mensagem é enviada passando por servidores até ser transmitida para a máquina de destino (AREND, 2013). Vale destacar a diferença entre internet e *web*: internet, como já mencionado anteriormente, refere-se à comunicação entre computadores. Arend (2013, p. 13) afirma que *web* pode ser definida como "um serviço que é aplicado sobre a Internet responsável pela difusão das informações interconectadas por *links* definidos como *hipertextos*.".

A internet influenciou também a área de desenvolvimento de *software*, de modo que empresas de desenvolvimento estão migrando seus sistemas para tecnologias *web* com objetivo de maior portabilidade e eficiência, deixando de lado os sistemas *desktop* (RIZO, 2020). Devido ao grande avanço da internet, equipes de desenvolvimento de *software* vem inovando em métodos e técnicas de desenvolvimento *web* com novas linguagens de programação, *frameworks*, abordagens e métodos.

Como exemplo de linguagens de programação para a *web*, pode-se citar Javascript, Python e PHP (CAMARGO, 2009). Por sua vez, é importante ressaltar que, variando dessas linguagens de programação, existem os chamados *frameworks*, que são ferramentas para auxílio do programador na codificação, como o Django, Laravel, React.js, Node.js, AngularJS. Tipicamente, a arquitetura de uma

aplicação na *web* é dividida em *back-end*, onde se encontra a parte de negócio da aplicação, contendo os seus controladores e métodos de acesso às bases de dados, e *front-end*, que é a camada de aplicação, responsável pela interação e interface com o usuário, que contém também validações e lógica de disposição de parâmetros de interface. Cada ferramenta utilizada dentre as citadas anteriormente é escolhida de acordo com sua funcionalidade, levando em consideração, também, o tipo de aplicação.

Centenaro (2014) afirma que softwares *web* são mais acessíveis pois são multiplataformas e acessíveis a partir de qualquer dispositivo com acesso à internet por meio de navegadores *web*. A manutenção desses *softwares* também é facilitada não necessitam de mecanismos de instalação, de modo que as atualizações são feitas nos próprios servidores *web* (local de armazenamento de dados) cuja manipulação afeta diretamente o usuário final.

2.4.2 Desenvolvimento Ágil

Uma das grandes dificuldades enfrentadas pelas equipes de desenvolvimento de *software* é conseguir concluir e entregar tudo o que o cliente quer em um curto espaço de tempo, isso é comum com o uso de metodologias tradicionais que não permitem a participação contínua do cliente, de modo que a entrega do projeto muitas vezes não é feita conforme foi idealizado, “pois as metodologias tradicionais seguem o passo a passo” como afirma (DA SILVA, R. O. et. al., 2016, p. 55), diferente da metodologia ágil que “dão mais ênfase às pessoas quando comparado aos processos” como declara (SANTOS, W. B. et. al., 2018, p. 2).

Uma busca crescente por melhorias no processo de desenvolvimento de *software* vem crescendo continuamente, desse modo, o Desenvolvimento Ágil de *Software* (DAS) vem sendo uma alternativa para alcançar esse objetivo. O DAS possui como principal característica a flexibilidade de adaptação, de maneira que o projeto não precisa ser predefinido antes e com isso impossibilite inserção de novos fatores ao projeto (SANTOS, W. B. et. al., 2018).

Da Silva et. al. (2016) diz que as metodologias ágeis são metodologias de desenvolvimento flexíveis e adaptáveis, cuja proposta é dividir o desenvolvimento em ciclos curtos, de maneira que ao final de cada ciclo o cliente receba uma versão

que agregue valor ao produto. O autor cita como exemplo de metodologias ágeis: o *Extreme Programming* (XP), o *Scrum* e o *Kanban*.

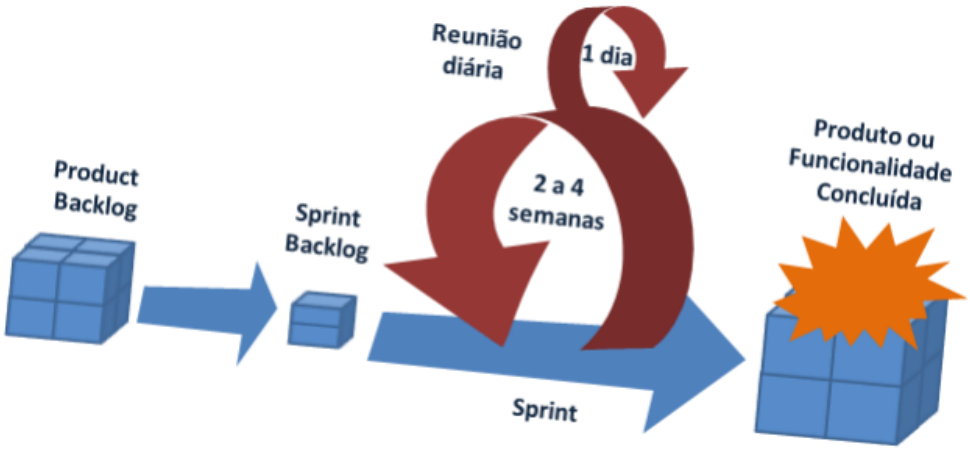
2.4.2.1 Scrum

Ainda de acordo com Da Silva et. al. (2016), o *Scrum* é um framework ágil bastante popular com foco nos aspectos gerenciais de desenvolvimento de *software*. O *Scrum* foi o método ágil escolhido para o desenvolvimento deste trabalho.

Da Silva, R. O. et. al. (2017, p. 41) afirmam que o **"***Scrum* é um *framework* para aprimorar a produção de projetos e usar suas características de desenvolvimento ágil para o desenvolvimento de *software*. " Trata-se de um método ágil popular que agrega mais valor e possui as seguintes características: agilidade para gerenciamento e controle no desenvolvimento, otimização de tempo, desempenho equivalente entre as partes e aumento de produtividade no desenvolvimento (Da Silva, R. O. et. al., 2017).

Da Silva et. al. (2016) afirma que o processo do *Scrum* é definido conforme a figura 4:

Figura 4 - Processo do Scrum



Fonte: Da Silva, R. O. et. al. (2016)

O **Product Owner** é o maior interessado no *software* e pode ser representado por um cliente. É quem teve a necessidade do produto e possui a visão de todas as funcionalidades do produto a serem desenvolvidas mantendo-as em um artefato chamado **Product Backlog**.

Product Backlog é uma lista com todas as funcionalidades do produto organizadas por prioridades.

Sprint são ciclos de máximo um mês de duração, e essa duração devem permanecer fixas ao longo do desenvolvimento do produto para aumentar a transparência, adaptabilidade e previsibilidade das entregas, com ajustes sendo realizados na reunião de planejamento da *sprint*. Esta reunião acontece no início da *sprin*t entre o Time Scrum (desenvolvedores, o *product owner* e *scrum master*), e é nela onde o objetivo da sprint e as estratégias de como atingi-lo são definidos, de acordo com a necessidade para o desenvolvimento das funcionalidades proposta (Bernardo, 2015). Durante a sprint, a cada 24 horas, é realizada uma reunião rápida com o objetivo de verificar o andamento da sprint. No final da sprint a meta é entregue e é planejada a próxima sprint – esse é o ciclo que ocorre até a conclusão do produto que está sendo desenvolvido.

O **Scrum Master** é quem comanda a equipe, responsável por manter o processo em funcionamento, verificar o andamento das atividades da equipe, liderar e organizar a equipe de trabalho, entre outras atribuições. Vale salientar que o Scrum Master não atua como gerente ou chefe da equipe pois ela é auto organizável.

O Scrum não é um processo ou uma técnica, mas sim um *framework* eficaz que emprega vários processos e técnicas, com o *framework* é possível resolver problemas complexos e adaptativos (SCHWABER; SUTHERLAND, 2020). O Scrum é um *framework* bastante conhecido e muito utilizado nas empresas devido a sua praticidade, o *framework* possui todo um processo que deve ser seguido, com um ciclo de vida que só acaba quando o produto deixa de existir, e foi escolhida para desenvolvimento deste trabalho.

2.4.2.2 Backlog do produto

Schwaber e Sutherland (2020) caracterizam o *backlog* do produto como uma lista ordenada que reúne as necessidades do produto e aponta itens essenciais para garantir o sucesso do projeto. O *backlog* do produto contém todas as características, funções, melhorias e correções que devem ser feitas no produto até sua conclusão.

De maneira ágil, o *backlog* do produto pode ser entendido como os requisitos de um *software* do método tradicional. Com efeito, a diferença é a que *Scrum* acredita que não faz sentido coletar todos os requisitos no início do projeto, pois os mesmos

estão em constante avaliação e até mesmo o entendimento do cliente sobre os mesmos pode mudar, o que é uma característica positiva de projetos ágeis.

Assim, por se tratar de uma abordagem ágil para o desenvolvimento de um produto, para este projeto não é necessário que seja feito um esforço inicial demasiadamente exagerado para colher detalhes dos requisitos do *software* a ser desenvolvido. Faz-se apenas necessário que seja inserido os itens iniciais no *backlog* do produto para a primeira *sprint*. O *backlog* do produto irá emergir à medida que é desenvolvido o projeto e é conhecido mais sobre o cliente e sobre o produto que está sendo desenvolvido (Bernardo, 2015).

O *backlog* do produto deve ser claro e detalhado, além de ser priorizado. Essa priorização deve ser realizada com base no valor dos itens constantes no *backlog* do produto (*product backlog items*). O Product Owner que juntamente com a equipe de desenvolvimento que define esses detalhes e prioridades. Essas prioridades devem ser tratadas dentro de cada *sprint* conforme foi definido na *sprint backlog*. É importante destacar que essas prioridades são definidas de acordo com as necessidades do cliente podendo ser modificadas quando necessário, características de um projeto ágil (Bernardo, 2015).

Schwaber; Sutherland (2020) afirmam que no *backlog* do produto é utilizado itens para descrever o produto que será desenvolvido, esses itens podem ser agrupados para posteriormente serem aplicados, em seguida os itens são refinados no qual são adicionados detalhes e ordem de prioridade.

2.4.2.3 Sprint backlog

Schwaber e Sutherland, (2020, p. 13) dizem que “o *sprint backlog* constitui-se em um conjunto de itens do *backlog* do produto selecionados para a *sprint*”. Assim, o *product owner*, juntamente com os desenvolvedores, definem quais são os itens do *sprint backlog* de acordo com a ordem de prioridade, a partir dos itens de mais alta prioridade no *backlog* do produto.

Os itens com maior prioridade são puxados para a *sprint backlog* e devem ser realizados dentro da iteração (*sprint*) atual, o status de concluído é adicionado e o processo se repete até que todas as atividades do *backlog* sejam concluídas.

Dentro da *sprint backlog* os itens são detalhados em tarefas e subtarefas, essas subtarefas são atividades que devem ser desenvolvidas do ponto de vista da

equipe de desenvolvimento. Conforme elas vão sendo realizadas, os *status* das mesmas são atualizados, até a conclusão da *sprint*.

O detalhamento dos itens de mais alta prioridade do *sprint backlog* (adicionando subtarefas) é realizado na reunião de planejamento da *sprint*, para que o desenvolvimento daqueles itens possa iniciar o mais rapidamente possível. No entanto, nem todos os itens precisam ser detalhados logo no início. Os itens de menor prioridade podem ser detalhados durante o andamento da *sprint*, quando os outros forem concluídos.

Da mesma forma, durante o andamento da *sprint*, recomenda-se que algum tempo do Time Scrum seja dedicado com o detalhamento do *product backlog*, antes do planejamento da próxima *sprint*, com o objetivo de facilitar o trabalho da equipe em tal cerimônia. Ou seja, o detalhamento dos itens e suas prioridades só são feitas na reunião de planejamento de cada *sprint* para os itens de maior prioridade, não fazendo sentido gastar muito tempo detalhando itens do *backlog da sprint e* do produto, sobre os quais a equipe não tem todo o conhecimento ainda.

Essa é uma das características dos métodos ágeis: deixar o processo e seus itens disponíveis para mudanças, tornando-os adaptáveis e transparentes.

Trabalhar com *scrum* é trabalhar com adaptabilidade, pois trata-se de um processo incremental no qual as entregas são feitas iterativamente - ou seja, após concluir a entrega de uma sprint todo o processo é repetido, tornando tudo mais previsível, mesmo com a possibilidade de o *backlog* mudar a qualquer momento.

Da Silva et. al. (2016) dizem que a incerteza é uma característica do método ágil, pois o cliente está aprendendo o que precisa e com isso o *feedback* pode mudar e alterar o escopo do projeto. Para responder a essa característica, se faz necessário um processo adaptável, que seja previsível, e que receba *feedback* em ciclos curtos, para diminuir ao máximo o retrabalho - por isso, em *Scrum,* o trabalho deve ser feito por *sprints* de no máximo um mês de duração.

Com efeito, métodos ágeis, em geral, caracterizam-se em ciclos de *feedback* onde a equipe de desenvolvimento deve adaptar-se a mudanças que podem acontecer.

2.4.3 MVP (Produto Mínimo Viável)

De acordo com o livro *The Lean Startup* (2011), MVP refere-se em inglês a *Minimum Viable Product* – ou Produto Mínimo Viável e foi criado no berço do empreendedorismo e exposto junto ao conceito de *Lean Startup*.

O livro prossegue e afirma que o MVP compreende na construção de uma versão simples e compreensível de um determinado produto, empregando nessa versão os mínimos recursos possíveis, entregando a principal proposta de valor comercializável da ideia.

Desse modo, o MVP torna possível a validação do produto final por meio do seu lançamento com uma versão comercializável contendo os requisitos mínimos que tornam o produto viável, validando o potencial da ideia anterior a um investimento maior. Essa perspectiva para o negócio é ideal para organizações que têm uma grande ideia e querem empreender.

3 METODOLOGIA

O presente capítulo abordará a metodologia da pesquisa utilizada neste trabalho e está subdividido nas seções a seguir: 3.1 tipo de pesquisa, 3.2 descrição da população e amostragem, 3.3 descrição das técnicas e fontes utilizadas para coleta e análise de dados.

3.1 TIPO DE PESQUISA

Esta pesquisa foi norteada pelo método da *Design Science Research* (DSR) onde, como assegura Hevner et. al. (2004), para a concretização de uma pesquisa, é necessário que os princípios dos paradigmas de Ciência do Comportamento e o paradigma de *Design Science* sejam seguidos, com questões de pesquisas. De modo que, para que uma pesquisa ofereça uma contribuição significativa, é fundamental que ela incorpore um ciclo de pesquisas complementares entre os paradigmas citados.

O paradigma de ciência do comportamento é caracterizado por teorias e conceitos que explicam fatos que tenham relação com o problema abordado. Já o paradigma de DSR, que foi abordado nesse trabalho, é conceituado pela prática, desenvolvimento e inovações em relação ao deliberado problema.

Ainda de acordo com Hevner et. al. (2004), o método DSR é composto por uma questão geral de pesquisa e segmentado em questões secundárias de pesquisas. Inicialmente, mostra-se a questão geral de pesquisa (QGP):

QGP: Como desenvolver uma ferramenta *web* para gerenciar processos de Engenharia de Requisitos em projetos de Desenvolvimento de *Software*?

Para acrescentar a questão geral de pesquisa (QGP), será abordado questões secundárias de pesquisa (QSP):

QSP1: Como alinhar o processo de Engenharia de Requisitos e as melhores práticas de *Design Thinking* no Desenvolvimento de *Software*?

QSP2: Quais as ferramentas mais utilizadas para potencializar o processo de Engenharia de Requisitos em projetos de Desenvolvimento de *Software*?

QSP3: Como validar uma ferramenta *web* que implementa o processo de Engenharia de Requisitos para projetos de Desenvolvimento de *Software*?

De acordo com Wazlawick (2010), uma pesquisa científica deve ser orientada por princípios de métodos científicos, onde através de técnicas ou métodos com cunho científico obtenha-se resultados que contribuam para um determinado problema. Entre os métodos de pesquisa científica existentes e utilizados na ciência da computação, o presente trabalho abordou os métodos de pesquisa bibliográfica, pesquisa exploratória aplicada e estudo de caso.

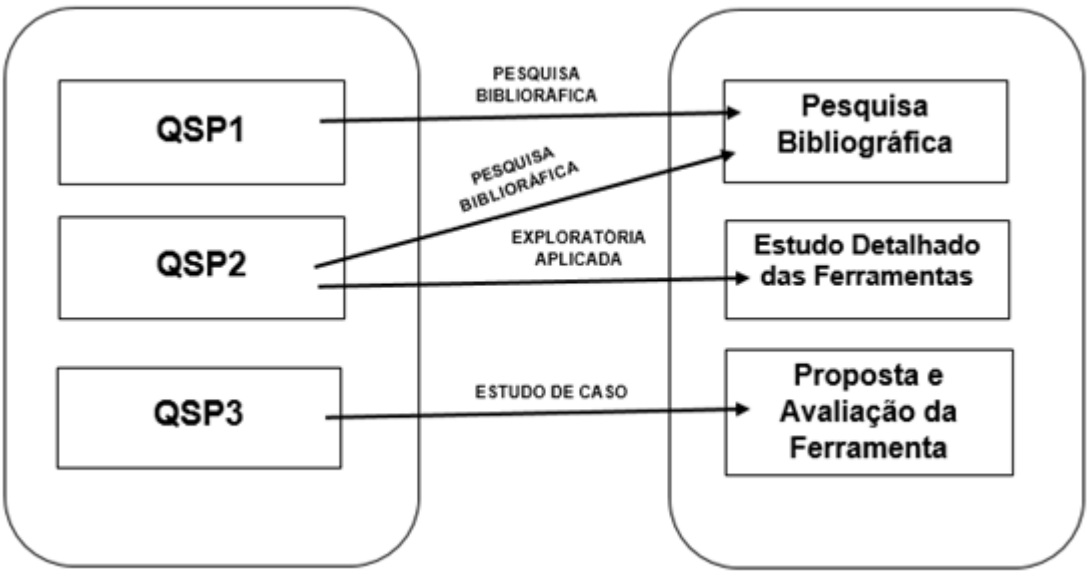
Quanto ao ponto de vista da natureza, para Fontelles et. al. (2009):

- **Pesquisa Bibliográfica**: consiste em estudos de trabalhos acadêmicos já publicados de outros autores de modo que contenha alguma contribuição para a pesquisa abordada, complementando conceitos que o autor ainda não possuía.
- **Pesquisa Exploratória Aplicada**: consiste em um estudo mais detalhado do problema que será abordado de modo que produza algum conhecimento ou inovação científica voltada para algum problema da atualidade.

Quanto ao ponto de vista técnico, para Martins (2008):

- **Estudo de caso:** consiste em uma metodologia onde é possível fazer a análise, descrição ou interpretação dos dados já coletados para estudo do problema proposto pela pesquisa.

Figura 5 - Métodos de pesquisa



Fonte: Autoria própria

Quanto à pesquisa bibliográfica, ela foi realizada por meio do estudo de artigos acadêmicos e livros na literatura sobre o tema abordado. A pesquisa foi realizada em bases significativas para o tema de Engenharia de *Software* e Engenharia de Requisitos, tais como o Google Acadêmico, *ACM Digital Library* e *IEEEXplore*, além de revistas e congressos especializados na área. Foi realizada uma análise dos artigos e livros, e assim um embasamento teórico para o trabalho, como exposto no capítulo 2.

A pesquisa exploratória aplicada também consiste em uma pesquisa em artigos nas fontes mencionadas no parágrafo anterior, só que o foco se encontra nas ferramentas existentes para realização da ER. Para que se limitasse a isso, foram utilizadas como palavras-chave os termos a seguir: "ferramentas para auxílio à engenharia de requisitos", "*tools for requirements engineering assistance*", e "ferramentas para engenharia de requisitos". O número de artigos retornado a essas pesquisas foi de 125 e, após realizar uma filtragem por similaridade considerando título e resumo dos artigos, foi delineado o estudo para 4 (quatro) ferramentas, mencionadas concomitantemente em 33 títulos e resumos de artigos. Ou seja, este projeto se interessou nas ferramentas mais comumente mencionadas, considerando que elas eram as mais conhecidas na literatura e, portanto, mais maduras para o propósito do gerenciamento de requisitos. O resultado da pesquisa exploratória pode ser encontrado no capítulo 5.

O desenvolvimento da ferramenta proposta consiste na realização do processo de desenvolvimento de *software* (coleta de requisitos, diagramação, protótipos, codificação e documentação). A avaliação da ferramenta se deu por meio de um estudo de caso em relação ao uso da ferramenta, e almejou-se como público-alvo deste estudo organizações que realizam desenvolvimento de *software* na cidade de Patos-PB.

3.2 DESCRIÇÃO DA POPULAÇÃO E AMOSTRAGEM

A pesquisa foi feita em empresas de desenvolvimento de *software* na cidade de Patos - PB, e com desenvolvedores *freelancers*, que realizam atividades de desenvolvimento de *software* independentes, também residentes na cidade de Patos - PB. A cidade de Patos é município do estado da Paraíba, sendo o quarto município

mais populoso do estado. A cidade é considerada uma importante cidade da região e é classificada como centro sub-regional.

Por sua vez, com respeito à amostragem, como base, para seleção dos entrevistados, em relação a funcionários de empresas de desenvolvimento de *software*, foi adotado o último relatório disponibilizado pelo Sebrae - 2015 e 2017 que disponibiliza dados de empresas de desenvolvimento de *software* da cidade de Patos. Do relatório disponibilizado, de um total de 8 (oito) empresas, apenas 1 (uma) empresa consentiu participar da pesquisa e entrevista. Desse modo, foi feito o contato com outras empresas da cidade a qual 2 (duas) consentiram participar da entrevista. Com isso, totalizando 3 empresas de desenvolvimento de software participaram da entrevista.

Em relação aos desenvolvedores independentes e às demais empresas entrevistadas, esses foram selecionados pela rede social LinkedIn (LinkedIn, 2020). A busca se deu por meio da palavra-chave “Desenvolvedor” sendo escolhido pessoas, empresas e conexões em 1º e 2º. O LinkedIn é a maior rede social profissional com aproximadamente 645 milhões de usuários, presente em 200 países com o foco em conectar profissionais e o crescimento de carreiras.

3.3 DESCRIÇÃO DAS TÉCNICAS E FONTES UTILIZADAS PARA COLETA E ANÁLISE DE DADOS

A subseção a seguir apresenta as técnicas e fontes utilizadas para coleta de dados da pesquisa e como ocorreu a análise desses dados.

3.3.1 Técnica e fontes utilizadas para coleta de dados

A coleta de dados foi feita por meio de uma entrevista com as empresas de desenvolvimento de *software* e desenvolvedores independentes com o objetivo de avaliação da ferramenta proposta.

O instrumento de coleta de dados escolhido foi a entrevista, que consiste em uma conversação face a face, de maneira metódica com o intuito de proporcionar ao entrevistado toda a informação necessária (OLIVEIRA, et. al., 2013). De acordo com

Oliveira et. al. (2013), “a entrevista tem como objetivo principal a obtenção de informações do entrevistado, sobre determinado assunto ou problema. ”

Foram produzidos dois roteiros para duas entrevistas. A primeira consistiu na coleta de dados pessoais das empresas e desenvolvedores, e na coleta de conhecimentos gerais sobre as empresas e desenvolvedores e como era feita a divisão de tarefas e armazenamento de informações. Na segunda entrevista foi questionado sobre a opinião do entrevistado em relação a criação de uma ferramenta *web* para ER.

A entrevista foi feita com 3 (três) empresas de desenvolvimento e 3 (três) desenvolvedores de software independentes e foi realizada no mês de novembro de 2020. Todos os dados coletados serviram para contribuição dessa pesquisa, sendo mantido em sigilo o nome das empresas e dos desenvolvedores.

3.3.2 Análise de dados

O critério de avaliação utilizado foi *Net Promoter Score* (NPS), criado por Reichheld (2003), que tem como objetivo mensurar a satisfação dos clientes. Sua base está nas relações de confiança das pessoas por meio da indicação de produtos e serviços. O método é calculado a partir da probabilidade de recomendação do cliente e visa descobrir indiretamente se o cliente está satisfeito ou não com os produtos e serviços oferecidos, através da pergunta, “você indicaria nosso produto para um amigo ou conhecido? ”, podendo com isso identificar a lealdade e o lado ruim dos clientes (Oliveira et. al., 2016).

Rampinelli (2019) afirma que tal pergunta única compreende uma escala entre 0 e 10 para compreensão da experiência e satisfação do usuário, as respostas são classificadas entre:

- Promotores com respostas entre 9 e 10;
- Passivos ou Neutros com respostas entre 7 e 8;
- Detratores com respostas abaixo de 6.

O resultado varia de -100 a 100, de modo que é calculado pela fórmula:

NPS = %de clientes promotores - %de clientes detratores

Após o cálculo, os dados são analisados, organizados e classificados em quatro partes:

- Zona de Excelência: NPS entre 75 a 100%
- Zona de Qualidade: NPS entre 50 a 74%
- Zona de Aperfeiçoamento: NPS entre 0 a 49%
- Zona de crítica - NPS entre -100 a -1%

Após a coleta dos dados por meio da entrevista, eles foram organizados e detalhadamente analisados como resultados da pesquisa no capítulo 6 deste trabalho.

3.4 MATERIAIS E MÉTODOS

Nesta seção serão apresentados os materiais e métodos usados para desenvolvimento deste trabalho. Na seção 3.4.1 será apresentado os materiais e na seção 3.4.2 os métodos utilizados para emprego desses materiais

3.4.1 Materiais

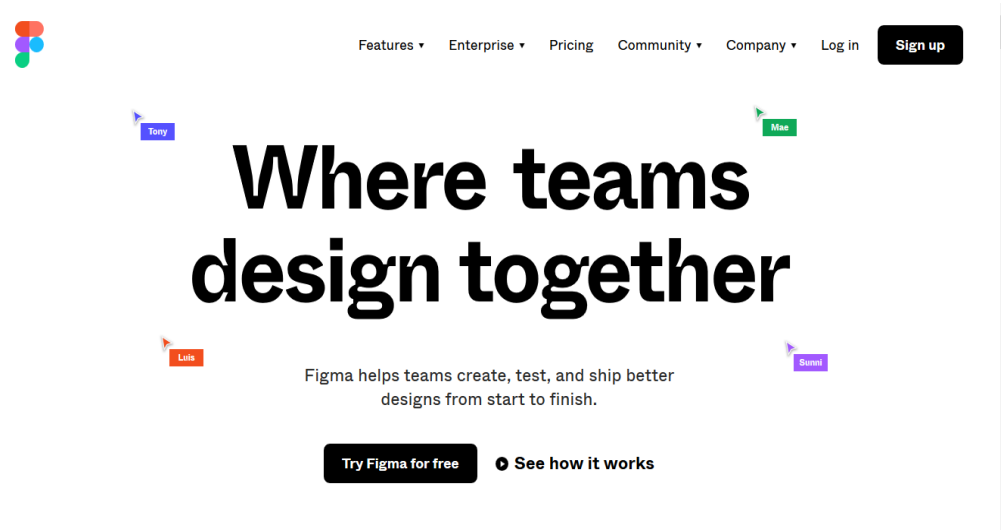
Nesta seção serão apresentadas as ferramentas utilizadas para o desenvolvimento da ferramenta proposta.

3.4.1.1 Figma

Figma é uma ferramenta *web* para *design* de interface de usuário. A ferramenta permite a colaboração simultânea de *designers* e desenvolvedores para edição de trabalhos (FARMOUDEHYAMCHEH, 2019).

Dentro do campo da prototipação de interfaces, esta ferramenta é bastante conhecida e utilizada, e permite a criação de protótipos para diversos *frames* diferentes, a criação de protótipos interativos e também contém dados em *CSS* que auxiliam aos desenvolvedores na codificação.

Figura 6 - Figma



Fonte: Disponível em Figma. Acesso em 2 de setembro de 2020

O *Figma* foi utilizado neste projeto para a criação do protótipo da ferramenta proposta.

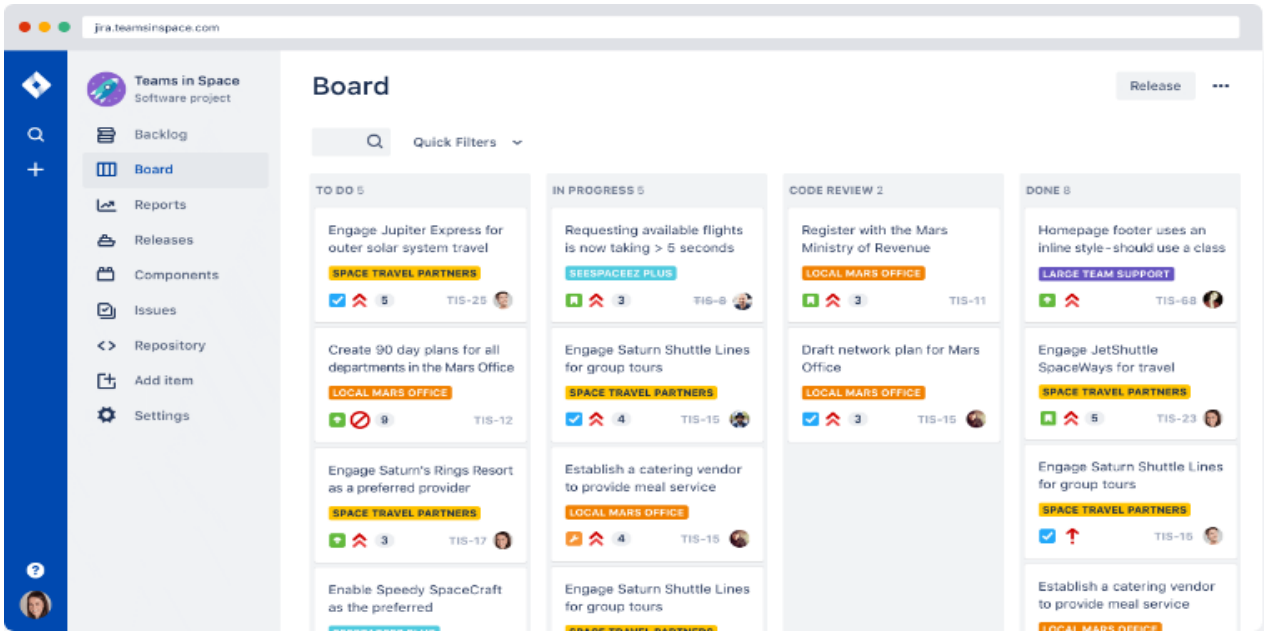
3.4.1.1.1 Material UI

É uma biblioteca de componentes *React* para desenvolvimento ágil e fácil, os componentes funcionam isoladamente e são independentes. O material UI é utilizado para melhoria no *front-end* de aplicações *web*, oferecendo *design* de *layouts* prontos com o objetivo de ajudar desenvolvedores nas suas aplicações (MATERIAL - UI, 2020).

3.4.1.2 Jira

Criado pela empresa Atlassian (2020), o *framework* Jira utiliza métodos ágeis (*Scrum e Kanban*) de forma simples e interativa, MACIEIRA, L. G. (2018). O Jira é um *framework* de gerenciamento de tarefas usado para o desenvolvimento ágil de produtos. Ele possibilita a criação de histórias de usuário, planejamentos de *sprint*, criação de quadros personalizáveis *Kanban*, e a distribuição de tarefas entre a equipe, conforme mostra figura 7 abaixo. Também é possível a escolha de um fluxo de trabalho que melhor se adeque a equipe ou a criação do próprio fluxo de trabalho.

Figura 7 - Jira



Fonte: Disponível em Jira Software. Acesso em 30 de setembro de 2020

O Jira foi utilizado neste trabalho para gerenciamento de tarefas utilizando o *framework* ágil Scrum, conforme apresentado na subseção 4.2.1.

3.4.2 Métodos

Nesta seção serão apresentados os métodos utilizados para o desenvolvimento da ferramenta.

3.4.2.1 Desenvolvimento Ágil com Scrum

Schwaber e Sutherland (p. 4, 2020) caracterizam o *Scrum* como “um *framework* leve que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos”. O Scrum foi o *framework* ágil utilizado neste trabalho e diante do que foi apresentado na seção 2.4.2 ficou definido a seguinte logística:

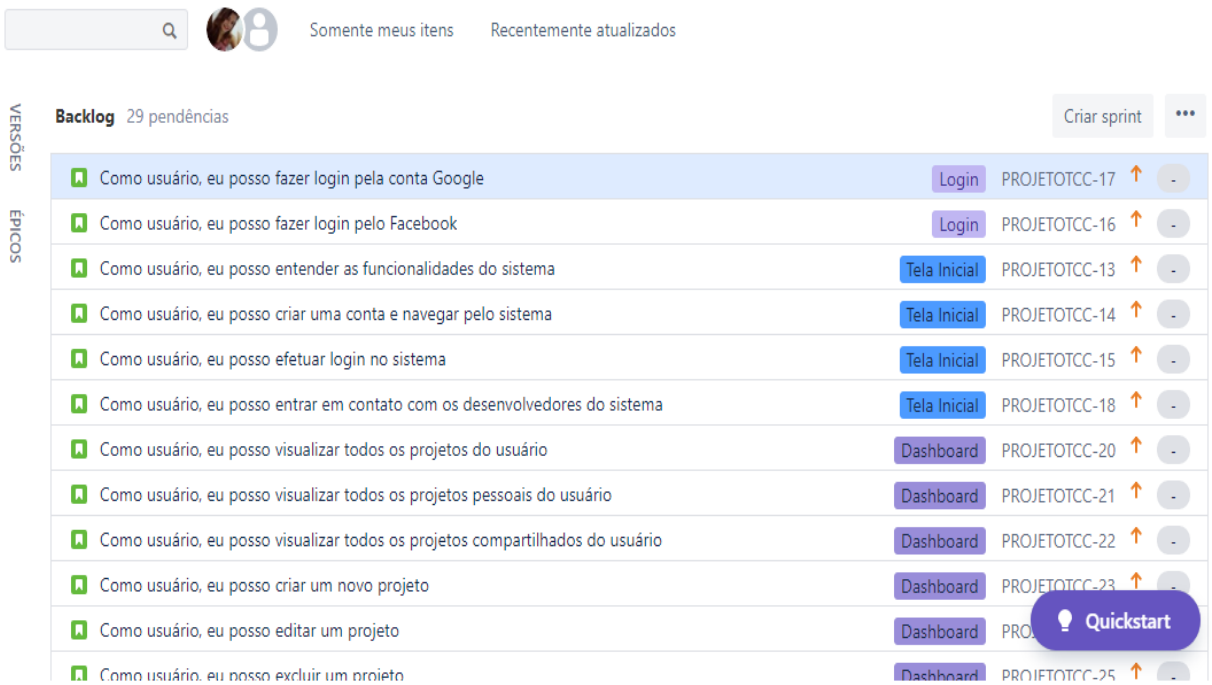
- Rodrigo Alves Costa (orientador desta pesquisa) assumiu o papel de *Scrum* Master e foi responsável pela eficácia e organização do processo de desenvolvimento proposta da ferramenta.
- Maria Luíza Mendes Barros (aluna e pesquisadora) assumiu o papel de *Product Owner* e de *Developer* de modo que: ao assumir papel de *Product Owner* foi responsável pelo gerenciamento do *backlog* do produto e iteração com os usuários; ao assumir papel de *Developer* ficou responsável pelo desenvolvimento da ferramenta executando funções de *Designer UI/UX*, e Desenvolvedora *Full Stack*.

Vale destacar que, de acordo com o Guia do *Scrum* (SCHWABER; SUTHERLAND, 2020), é possível uma mesma pessoa executar o papel de *Scrum Master* e *Developer*, ou de *Product Owner* e *Developer*, em projetos pequenos e pontuais. No entanto, não é possível que a mesma pessoa seja *Scrum Master* e *Product Owner*.

3.4.2.2 Requisitos organizados em um Backlog do Produto

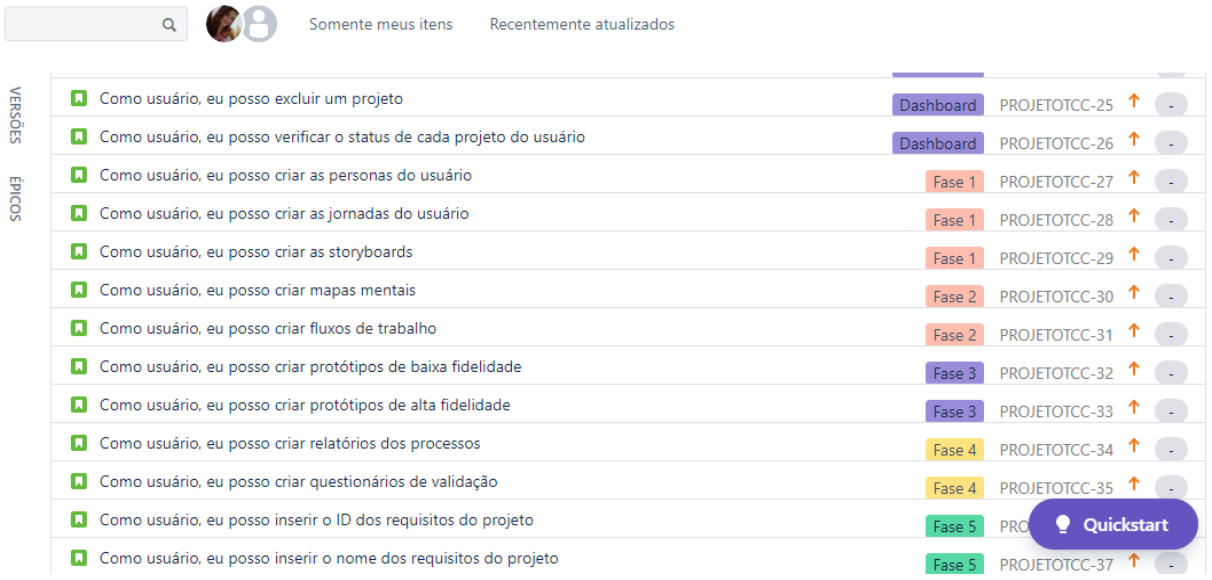
O *backlog* do produto é uma lista ordenada e emergente do que é necessário para melhorar o produto, Schwaber; Sutherland (2020). O *backlog* do produto contém todos os requisitos do produto a ser desenvolvido organizados de forma ágil, desse modo os requisitos da ferramenta proposta foram organizados na ferramenta Jira em um *backlog* do produto e podem ser observados na figura abaixo:

Figura 8 - Backlog do produto



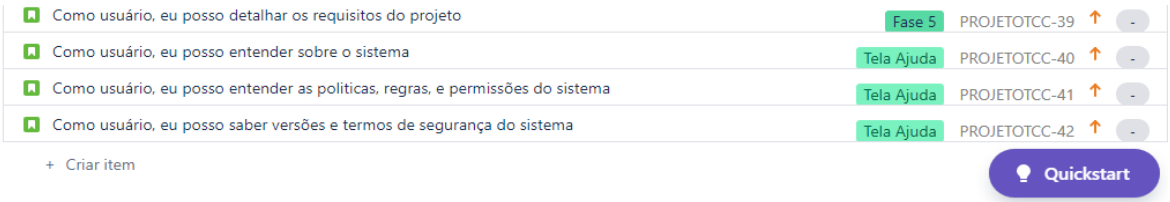
Fonte: Captura de tela do framework Jira. Acesso em 14 de novembro de 2020

Figura 9 - Backlog do produto



Fonte: Captura de tela do framework Jira. Acesso em 14 de novembro de 2020

Figura 10 - Backlog do produto

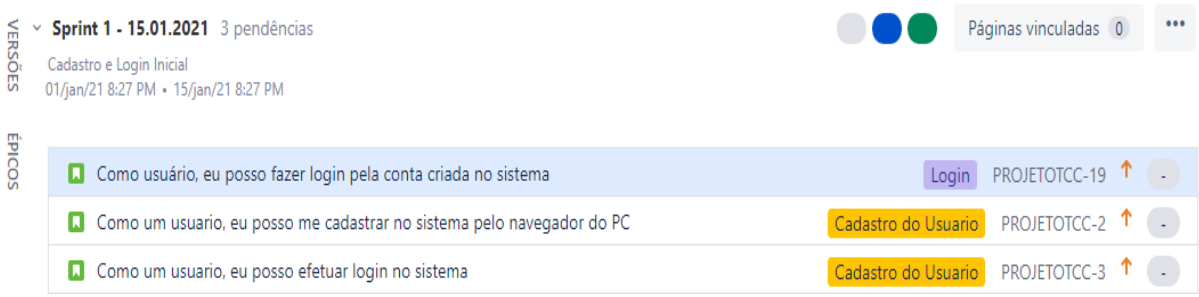


Fonte: Captura de tela do framework Jira. Acesso em 14 de novembro de 2020

Após definição dos itens do *backlog* do produto foi definido a duração de 2 semanas para a *sprint backlog* e quais os itens a serem priorizados nesta iteração, que podem ser observados na figura abaixo. O processo se repete até que todos os itens do *backlog* do produto sejam priorizados em uma *sprint.*

Para a primeira *sprint* foi definido começar pela tela de login onde no *sprint backlog* foram priorizados os itens: fazer *login* com conta criada no sistema, se cadastrar no sistema pelo navegador, efetuar *login* no sistema.

Figura 11 - Sprint backlog

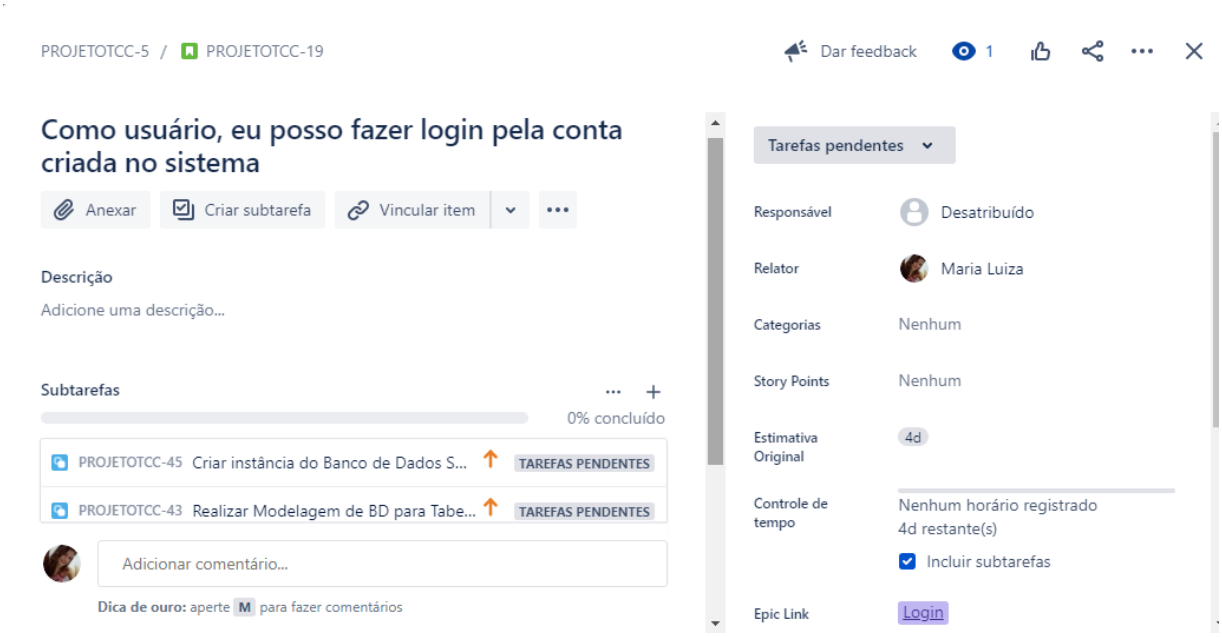


Fonte: Captura de tela do framework Jira. Acesso em 14 de novembro de 2020

Após definição de prioridades para a *sprint backlog* da iteração atual, todos os itens da *sprint* são detalhados em subtarefas, essas subtarefas são atividades que devem ser desenvolvidas do ponto de vista da equipe de desenvolvimento, conforme imagem abaixo.

Essas subtarefas são definidas por ordem de prioridades de execução, como podemos observar na figura abaixo, onde para que seja feito login em uma conta criada no sistema é necessário primeiramente criar a instância do BD, posteriormente realizar modelagem do BD e por fim fazer a integração do BD com o sistema.

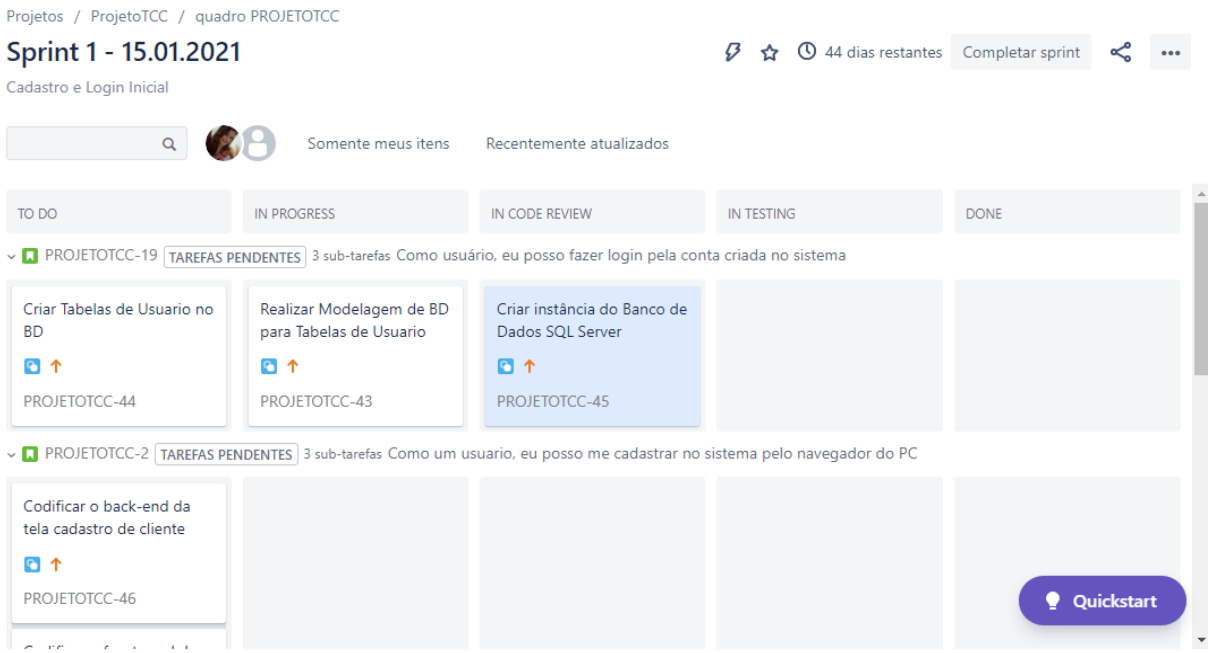
Figura 12 - Subtarefa da sprint do backlog



Fonte: Captura de tela do framework Jira. Acesso em 14 de novembro de 2020

Conforme as subtarefas são realizadas, o estado atual de cada uma delas (*status*) é atualizado em subtarefa na ferramenta até sua conclusão. O mesmo acontece com os itens de cada *sprint*, de modo que após conclusão dos itens da *sprint*, todo o processo será repetido o processo novamente até a entrega do produto ao cliente.

Figura 13 - Quadro de Tarefas da Primeira Sprint



Fonte: Captura de tela do framework Jira. Acesso em 14 de novembro de 2020

A adaptabilidade permite ao processo ágil a realização de modificações de acordo com os *feedbacks* recebidos conforme são feitas as entregas a cada processo incremental semanal. O *scrum* permite a imprevisibilidade, e é adaptável às mudanças que podem ou não acontecer, garantindo assim o sucesso do projeto e a satisfação do cliente e usuários.

3.4.2.3 Processo de Design

Pode-se compreender *design* como sendo a utilização de ideias criativas na criação de um determinado propósito que pode ser transformado em telas, protótipos, imagens, objetos, entre outros. *Design* é todo o processo percorrido para a concretização da ideação do *design* proposto.

O processo de *design* utilizado neste trabalho foi o *Design Thinking* que é composto de três etapas: Inspiração, Ideação e Implantação, tais etapas já foram explicadas no capítulo 2 deste trabalho e ele vai além do que se entende como *UX Design* normalmente, pois, no caso deste projeto, o objetivo é conceber requisitos.

Desse modo, seguiu-se as três etapas do DT para a criação dos protótipos da ferramenta proposta, os protótipos são apresentados no próximo capítulo.

5 DESENVOLVIMENTO

O capítulo a seguir apresenta o desenvolvimento deste trabalho, que compreendeu a entrevista de inicial motivação para compreensão das necessidades do grupo de foco, e o consequente desenvolvimento do protótipo da ferramenta proposta.

5.1 DEFINIÇÃO DO PROCESSO DE DESENVOLVIMENTO

O desenvolvimento da ferramenta se iniciou por meio de uma entrevista com desenvolvedores de *software*, e as respostas da entrevista foram utilizadas como base para a definição das principais funcionalidades da ferramenta. Conforme já destacado anteriormente, foram consideradas as funcionalidades básicas, também presentes na maioria das ferramentas relacionadas, e apresentamos o processo de *Design Thinking* como habilitador do gerenciamento de requisitos como um fator diferencial e exclusivo desta pesquisa. A ideia por trás da construção do protótipo segundo esses parâmetros é que eles pudessem ser criados em alta fidelidade (em uma ferramenta como o *Figma*), e posteriormente fossem novamente apresentados em uma nova seção de validação ao mesmo grupo de foco inicial, em um processo de homologação de ideias.

5.2 ENTREVISTA INICIAL COM O GRUPO DE FOCO

A entrevista inicial teve como objetivo compreender como é realizado o processo da Engenharia de Requisitos em empresas de desenvolvimento de *software*, bem como verificar se a criação de uma aplicação *web* para tal processo seria considerada como uma ferramenta útil em seus processos internos de desenvolvimento. Uma vez que essa questão fosse sinalizada como positiva, o entrevistador verificava também quais funcionalidades os entrevistados julgavam como indispensáveis para a adoção efetiva do aplicativo. Os resultados obtidos por meio da entrevista inicial podem ser encontrados no Apêndice A.

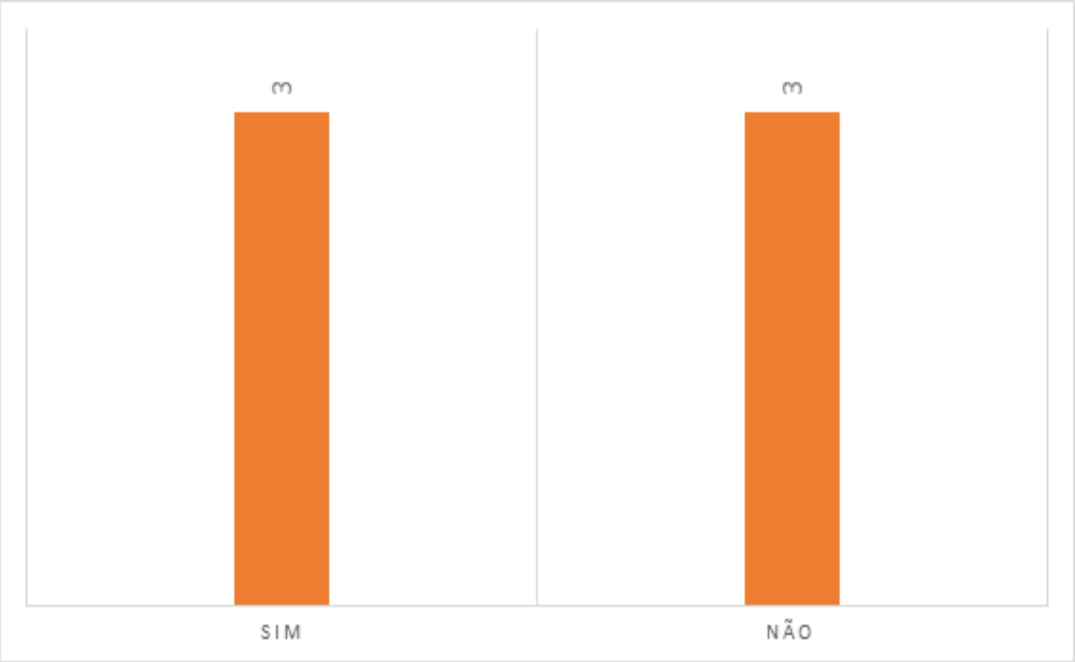
As entrevistas foram realizadas pela plataforma *Google Meet* com 6 profissionais da área de desenvolvimento, sendo 3 (três) deles funcionários de empresas de desenvolvimento de *software* e 3 (três) desenvolvedores independentes, mas que no momento da realização da pesquisa estavam atuando em projetos, e duraram entre 20 e 30 minutos cada.

Foram levantadas questões sobre o profissional para se obter dados em relação a tipos de clientes, nome da empresa e segmento em que trabalham. Posteriormente, as perguntas foram direcionadas ao modo de trabalho do entrevistado, sobre como são feitos os processos de Engenharia de *Software* e Engenharia de Requisitos e como é realizada a coleta de Requisitos. Por fim, foi questionado se a criação de uma ferramenta *web* para Engenharia de Requisitos contribuiria positivamente para o processo. Caso afirmativo, a entrevista concluía questionando quais funcionalidades julgam indispensáveis para essa ferramenta.

As respostas da entrevista serão demonstradas nos gráficos a seguir.

Inicialmente, buscou-se compreender como se dava a divisão das atividades da Engenharia de *Software* nas empresas de desenvolvimento e nos projetos nos quais os desenvolvedores independentes trabalhavam. Conforme observado no Gráfico 1, dos respondentes, 3 (três) responderam que há SIM a divisão de tarefas na equipe e 3 (três) responderam que NÃO há distribuição de tarefas.

Gráfico 1 - Distribuição de atividades e tarefas no desenvolvimento de software



Fonte: Autoria própria (2020)

Dentre os que responderam NÃO, o motivo das suas respostas foi:

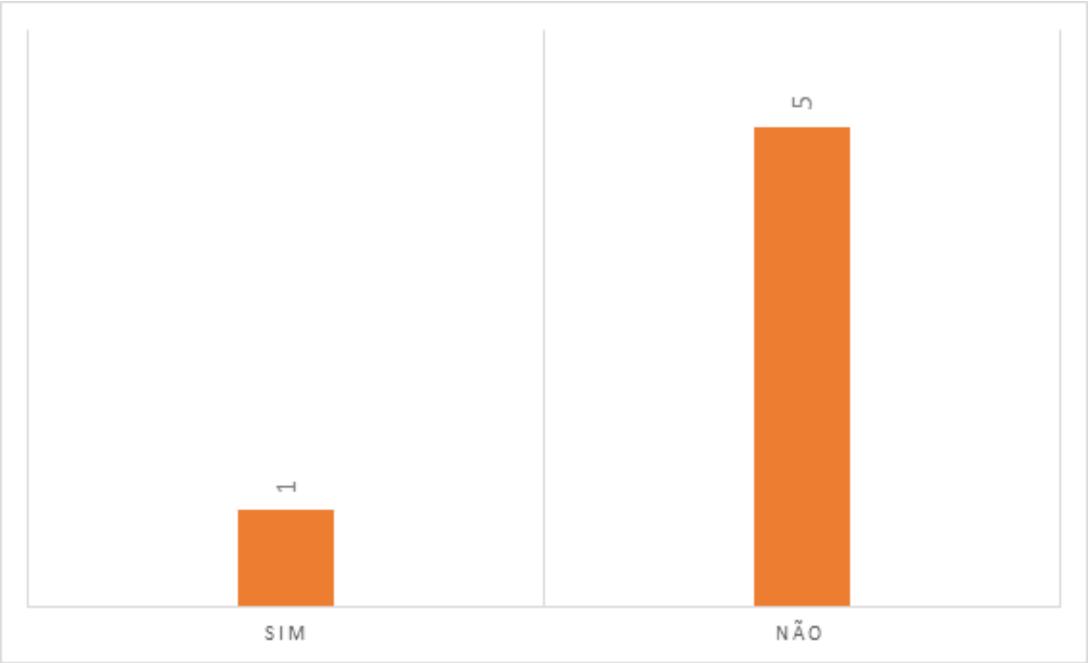
- Resposta 1: “por ser freelancer tudo é feito de forma informal e as atividades somente são repassadas com data de entrega, ele ficava apenas responsável por codificar uma parte do produto que era entregue a ele. ”
- Resposta 2: “apesar de ser empresa, 90% de todo o trabalho se concentra em uma pessoa (o dono ou o entrevistado), os demais componentes da empresa colaboram apenas em algumas etapas do desenvolvimento. ”
- Resposta 3: “o dono da empresa já chega com todos os requisitos e ideia sobre o projeto, apenas dividimos quem faz o back-end e o front-end, dúvidas sobre os requisitos também é tirado com o dono da empresa”

Dos respondentes SIM, as respostas foram semelhantes quanto a afirmarem que as atividades são sim distribuídas, de modo que essa distribuição acontece de acordo com as funções de cada componente da equipe de trabalho.

De acordo com as respostas dos entrevistados, observou-se que em alguns casos as atividades são distribuídas entre *back-end* e *front-end*, em outros casos o desenvolvedor é responsável por todo o processo da ES, de modo que a informalidade é predominante em relação ao desenvolvimento de *software*, as empresas se concentram em apenas entregar o produto solicitado ao cliente no menor tempo possível.

Posteriormente, buscou-se compreender se as etapas da Engenharia de Requisitos eram seguidas no processo de desenvolvimento de *software*. Conforme observado no Gráfico 2, dos respondentes, 5 (cinco) responderam que NÃO seguem as etapas da Engenharia de Requisitos e 1 (um) respondeu que SIM, segue as etapas da Engenharia de Requisitos.

Gráfico 2 - Etapas da Engenharia de Requisitos



Fonte: Autoria própria (2020)

Na entrevista, 5 (cinco) pessoas responderam que não seguiam etapas de Engenharia de Requisitos. Desses, 4 (quatro) declararam que não seguiam as etapas da Engenharia de Requisitos e que o contato com o cliente é feito de forma informal por meio de aplicativos de mensagens de texto (*Whatsapp*), ligações ou formulários Google (*Forms*), e 1 (um) respondente afirmou que os requisitos são descritos e pensados pelo dono da empresa e posteriormente repassado para codificação.

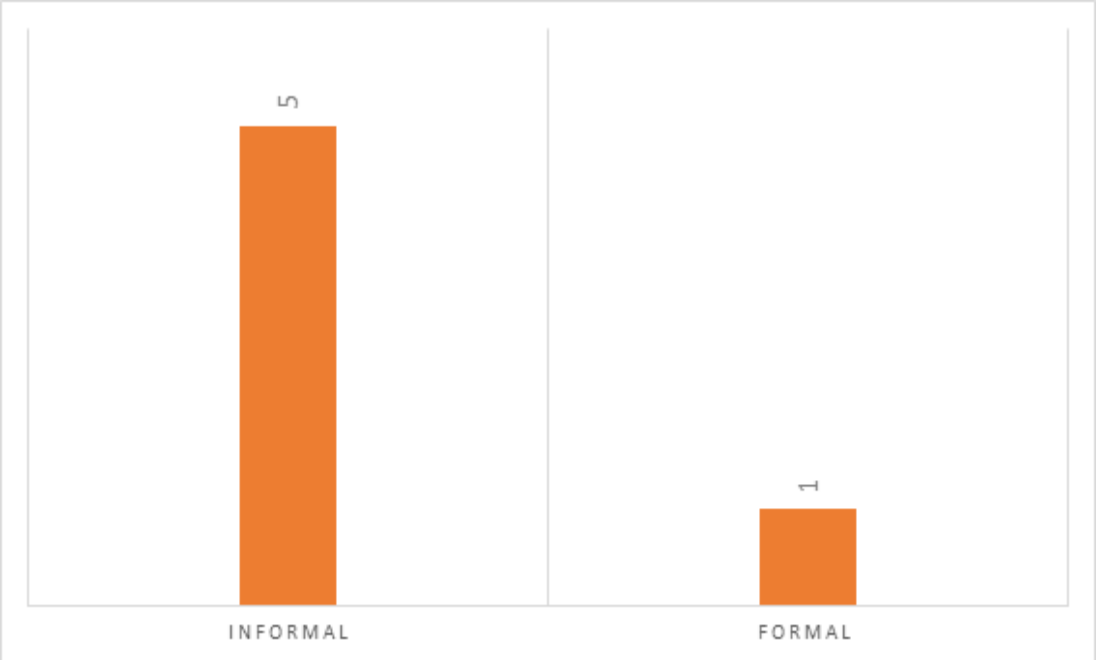
O entrevistado que respondeu positivamente relatou que o contato com o cliente é feito de forma formal por meio de entrevista para a coleta de requisitos, sendo tal entrevista gravada, com posterior prosseguimento às demais etapas da ER.

Após análise das respostas, observou-se que a informalidade é predominante no desenvolvimento de *software*, os desenvolvedores prezam pela praticidade mesmo que seja feito de forma informal, de modo que tal informalidade gera otimização de tempo e trabalho, através de aplicativos de textos e outros meios mencionados acima.

Em seguida, foram questionados os processos de coleta e o armazenamento dos requisitos dos projetos. Conforme observado no Gráfico 3, 5 (cinco) pessoas responderam que a coleta e o armazenamento são realizados informalmente

divergente de como estabelece a ER, e 1 (um) respondeu que para a coleta e armazenamento de requisitos é feito de maneira formal, seguindo todos os processos e diretrizes da ER.

Gráfico 3 - Coleta e Armazenamento de Requisitos



Fonte: Autoria própria (2020)

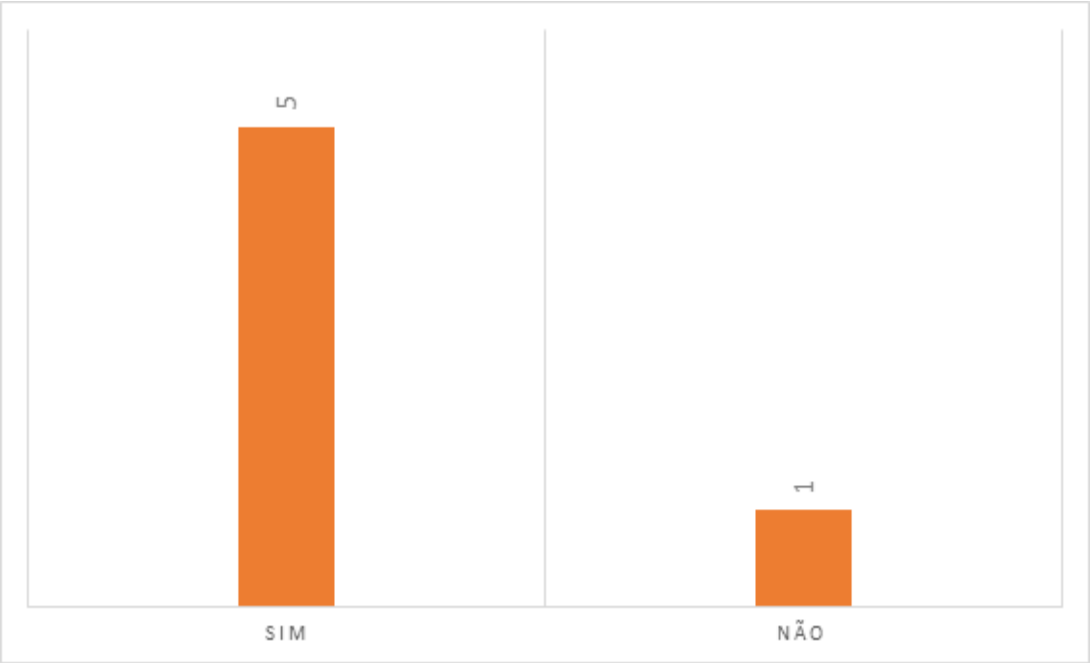
Com respeito aqueles que responderam que a coleta se dava informalmente, todos os 5 (cinco) responderam que após a coleta, os requisitos eram apenas armazenados em um documento de texto, desse modo não existe a criação de diagramas, inserção de prioridades ou divisão em requisitos funcionais e não funcionais. Os mesmos relataram que prezam pela praticidade e que tais diagramas e demais etapas, apesar de serem importantes, poderiam acabar adicionando burocracia no desenvolvimento, atrasando o processo de entrega do produto do cliente. Eles relataram ainda que qualquer tipo de documentação só é entregue caso solicitada pelo cliente.

O entrevistado que respondeu que os requisitos eram descritos, armazenados e especificados formalmente, respondeu que primeiramente ocorre a criação dos diagramas necessários e em seguida são realizadas as demais etapas da engenharia de requisitos que consiste em validação e documentação.

Compreendeu-se, após análise do Gráfico 3 (três), que a informalidade é predominante, os entrevistados relataram que em alguns casos o tempo exigido para desenvolvimento do produto é curto e que os usos de alguns aplicativos suprem a necessidade do desenvolvimento.

Seguindo o roteiro da entrevista, finalmente, foi questionado se a disponibilização de uma nova ferramenta *web* para Engenharia de Requisitos traria contribuições positivas e significativas para o mercado. Conforme observado no gráfico 4, dos entrevistados, 5 (cinco) responderam que SIM a criação de uma ferramenta seria algo favorável e positivo e 1 (um) respondeu que NÃO concordava com a criação de uma ferramenta nova para este fim.

Gráfico 4 - Criação de uma ferramenta web como contribuição positiva no mercado para Engenharia de Requisitos

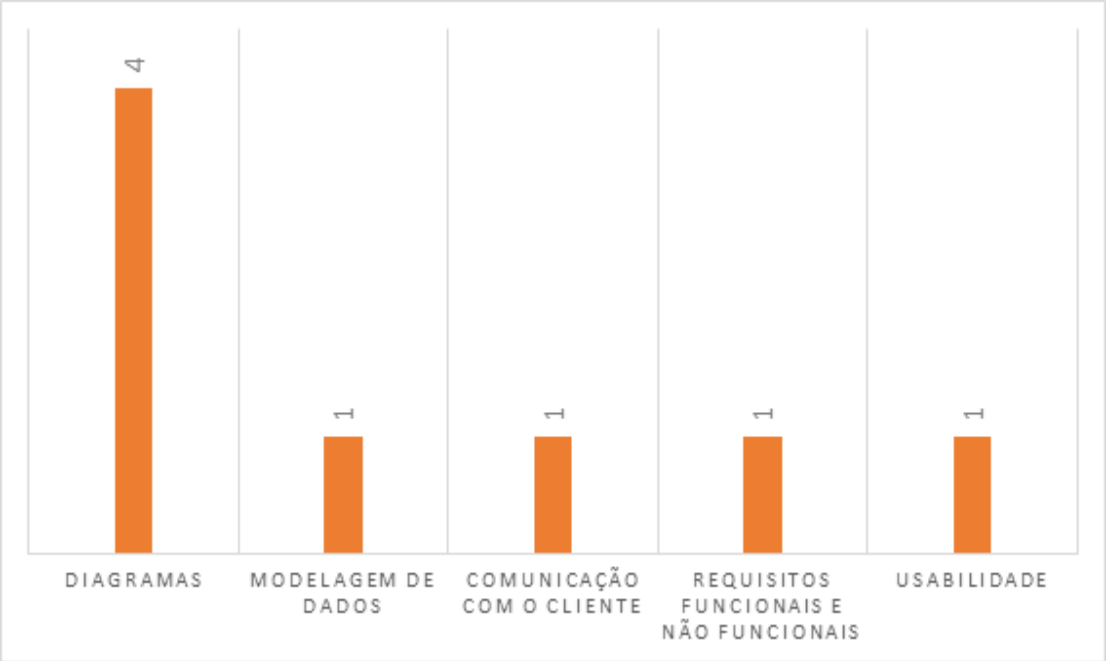


Fonte: Autoria própria (2020)

Com respeito à pessoa que respondeu NÃO, a mesma justificou que esta seria apenas mais uma ferramenta de Engenharia de *Software* como as já existentes no mercado, e afirmou que esse mercado já está saturado. O que se pode inferir desta resposta é que o processo a ser apresentado deve ser inovador e adaptado a processos de desenvolvimento mais atuais.

Dos que concordaram com o desenvolvimento de uma nova ferramenta, foi realizado um novo questionado, sobre quais funcionalidades eles julgariam indispensáveis para tal ferramenta. As respostas serão exploradas no gráfico 5.

Gráfico 5 - Funcionalidades na ferramenta web



Fonte: Autoria própria (2020)

Conforme mostra o gráfico 5, 4 (quatro) entrevistados afirmaram que a funcionalidade indispensável na ferramenta seria a “Geração dos Diagramas”. Eles também afirmaram ser indispensáveis a “Modelagem de Dados”, a “Comunicação com o Cliente”, a “Divisão dos requisitos em Funcionais e Não Funcionais” e a “Usabilidade”. Dos 5 (cinco) entrevistados, 1 (um) não soube opinar.

Tais funcionalidades coletadas na entrevista serviram como base para a criação dos protótipos da ferramenta proposta, apresentados na seção a seguir.

5.3 PROTÓTIPOS

A ferramenta proposta tem como objetivo contribuir no desenvolvimento de *software* possibilitando agilidade, economia de tempo e praticidade no desenvolvimento das fases da Engenharia de Requisitos. Intitulada SISREQ, ela possui como principais características: arquitetura *web* possibilitando o uso em

qualquer computador, integração com as fases do *Design Thinking* com objetivo de mapear as necessidades dos clientes, compreende também todas as fases da ER desde a coleta de requisitos até a fase de documentação e permite a geração dos principais diagramas da ER.

Inicialmente é apresentada a tela inicial da ferramenta SISREQ, na qual o usuário encontrará de forma descritiva e intuitiva todas as funcionalidades existentes na ferramenta que são: praticidade e eficiência, trabalho em equipe e controle de versões. É possível também inferir, logo na tela inicial, como seria a utilização da ferramenta, podendo-se, inclusive, encontrar os botões de “Criar Conta” e “Conheça Agora” para usuários que não possuem cadastro no sistema, e o botão “Login” para usuário que já possuem cadastro na ferramenta. Além disso, no final da tela, encontra-se um formulário para críticas e sugestões de melhorias sobre o uso da ferramenta, como mostra a figura abaixo.

Figura 14 - Tela Inicial da ferramenta SISREQ

Fonte: Autoria própria (2020)

Para uso da ferramenta, o usuário precisará fazer o cadastro no sistema, e posteriormente realizar o *login.* De acordo com a figura 15, é apresentado a funcionalidade, na qual o usuário insere o e-mail e a senha cadastrados para ter acesso ao sistema. Também será possível realizar *login* por *Single Sign-On* (SSO), com a conta Google ou com o Facebook.

Figura 15 - Tela Login

Fonte: Autoria própria (2020)

Em casos onde o usuário ainda não realizou o cadastro na ferramenta, é preciso clicar em “Cadastre-se” e o usuário será direcionado para a Tela de Cadastro. A figura 16 e 17 mostra a facilidade do cadastro e do *login* com SSO, isso motiva o usuário a realizar o seu cadastro, ainda que o mesmo decida inicialmente cadastrar-se por um período experimental. Ainda é possível se manter logado na ferramenta, o que pode agilizar o acesso para usuários que já decidiram incorporar a ferramenta na sua rotina diária de trabalho e não compartilham seu acesso a contas únicas em seus computadores pessoais ou estações de trabalho.

Figura 16 - Tela de Cadastro

Fonte: Autoria própria (2020)

Figura 17 - Tela de Cadastro

Fonte: Autoria própria (2020)

A Tela de Ajuda tem o objetivo de orientar o usuário sobre o uso do sistema, seu fluxo de processo, dados sobre a versão, e políticas de privacidade, possibilitando ao usuário a melhor experiência de uso com o design iterativo em todas as telas da ferramenta, conforme apresentado na figura 18. Ainda na Tela de Ajuda, o usuário poderá clicar em “Novo Projeto”, para criar um novo projeto, ou em “Meus Projetos”, para acessar projetos anteriores, sendo possível escolher de acordo com o propósito. Ao passar deste fluxo, o usuário será direcionado para a tela *Desktop.*

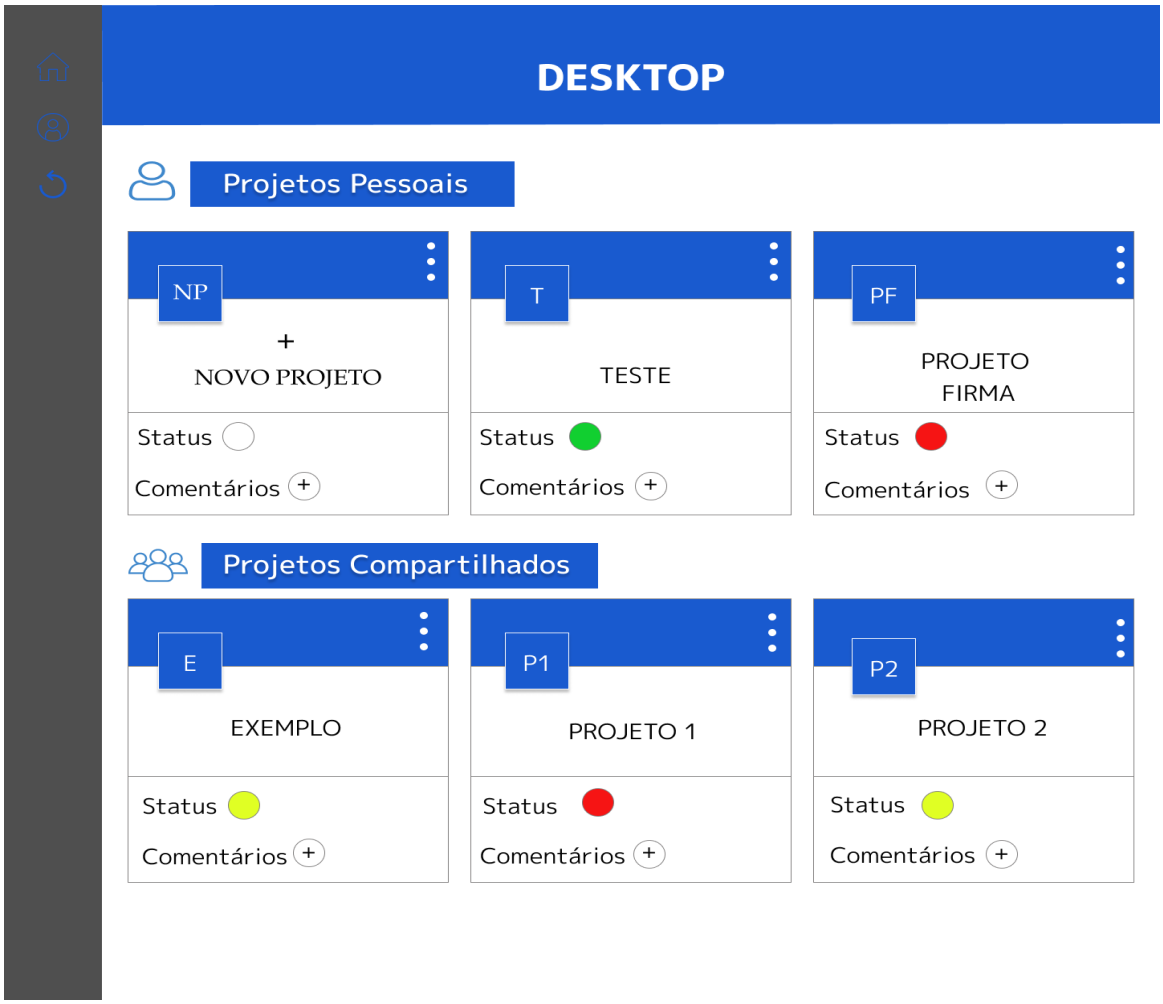
Figura 18 - Tela de Ajuda

Fonte: Autoria própria (2020)

Na tela *Desktop* o usuário pode acessar a lista de todos os seus projetos, inclusive aqueles compartilhados com outros usuários. Trata-se da área de trabalho do usuário. Conforme observado na Figura 19, a ferramenta atribui cores para os estados (representados na ferramenta por meio da nomenclatura *status*) dos projetos existentes dos seus usuários, de modo que são utilizadas as cores vermelho, verde e amarelo. Essas cores sinalizam fases de andamento do projeto: verde significa que o projeto está finalizado, amarelo significa que o projeto está em andamento o vermelho significa que o projeto está parado ou não foi iniciado. Na tela *Desktop*, o

usuário também tem a opção de criar um novo projeto ou editar um projeto já existente.

Figura 19 - Desktop



Fonte: Autoria própria (2020)

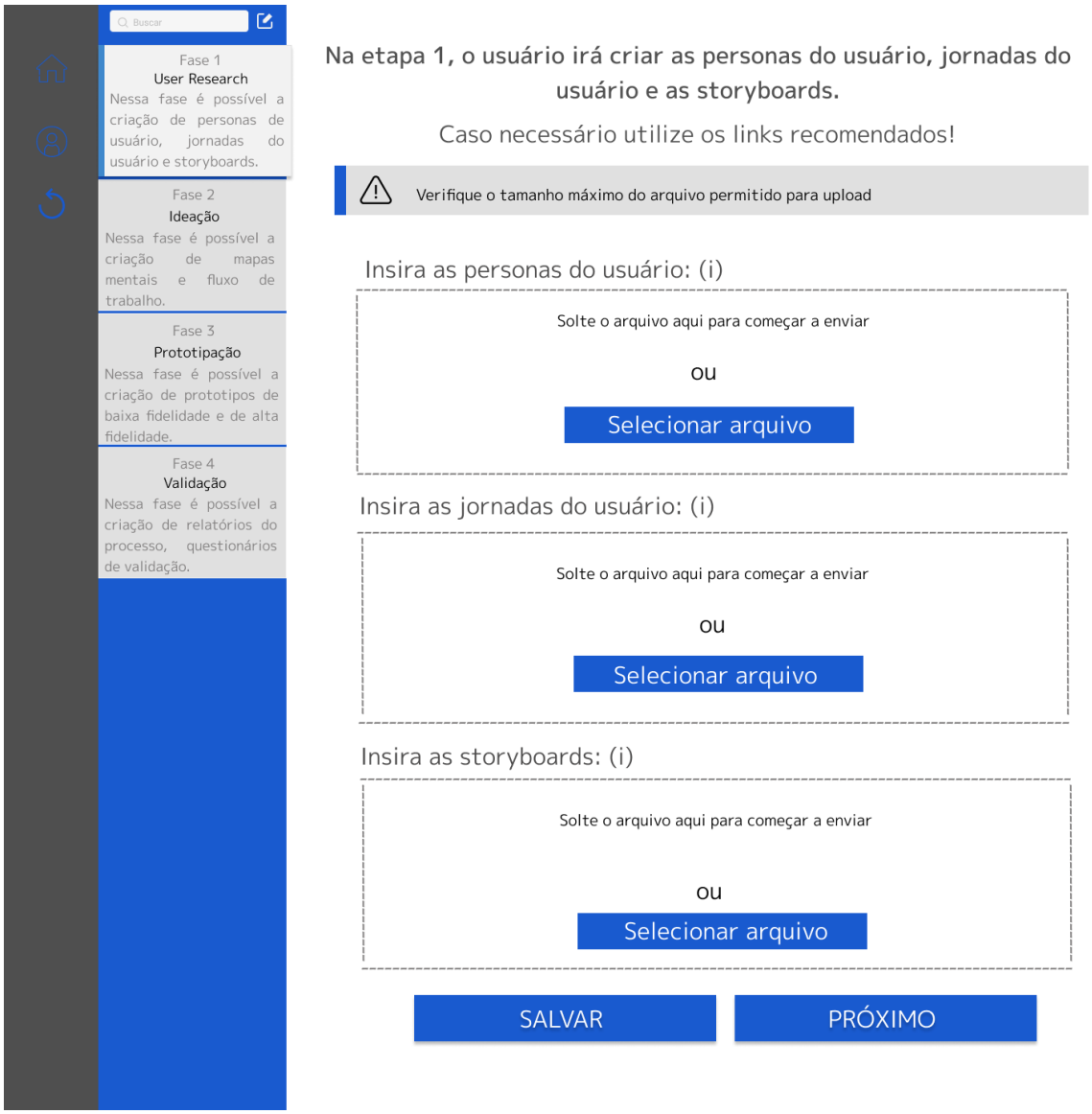
Iniciando as fases do *Design Thinking*, a figura 20 a seguir apresenta a fase 01, de *User Research* (pesquisa com o usuário), que compreende a primeira etapa do *Design Thinking*, na qual o usuário do SISREQ pode criar as personas dos usuários do sistema em desenvolvimento, as jornadas destes usuários e as *storyboards* (possíveis fluxos de utilização) daquele projeto. O sistema guia o usuário ao longo do processo de *Design Thinking*, com explicações sobre cada fase e o anexo de evidências para potencializar a compreensão da solução sendo construída. *Design Thinking* é, portanto, utilizado como uma fase de pré-projeto para elicitação de requisitos de maneira ágil, já que possibilita flexibilidade para o desenvolvimento do produto e priorização por requisito. Também é possível clicar em “Salvar” o

progresso feito na fase e concluir posteriormente ou prosseguir para as fases posteriores.

O processo de desenvolvimento da solução por meio da compreensão das necessidades do usuário através das ferramentas do *Design Thinking* prossegue para as demais fases do processo, de modo que na fase 02 (figura 21), a de Ideação, o usuário do SISREQ irá criar os mapas mentais e fluxos de trabalho. Na fase 03 (figura 22), na fase de Prototipação, o usuário do SISREQ irá criar os protótipos de alta fidelidade e os protótipos de baixa fidelidade e na fase 04 (figura 23), Validação junto ao cliente ou usuários, o usuário do SISREQ irá criar relatórios dos processos e questionários de validação, para que a ideia por trás de cada necessidade seja de fato confirmada pelo processo (incluindo protótipos). Os requisitos terão assim uma forma segura e efetiva para o seu processo de levantamento.

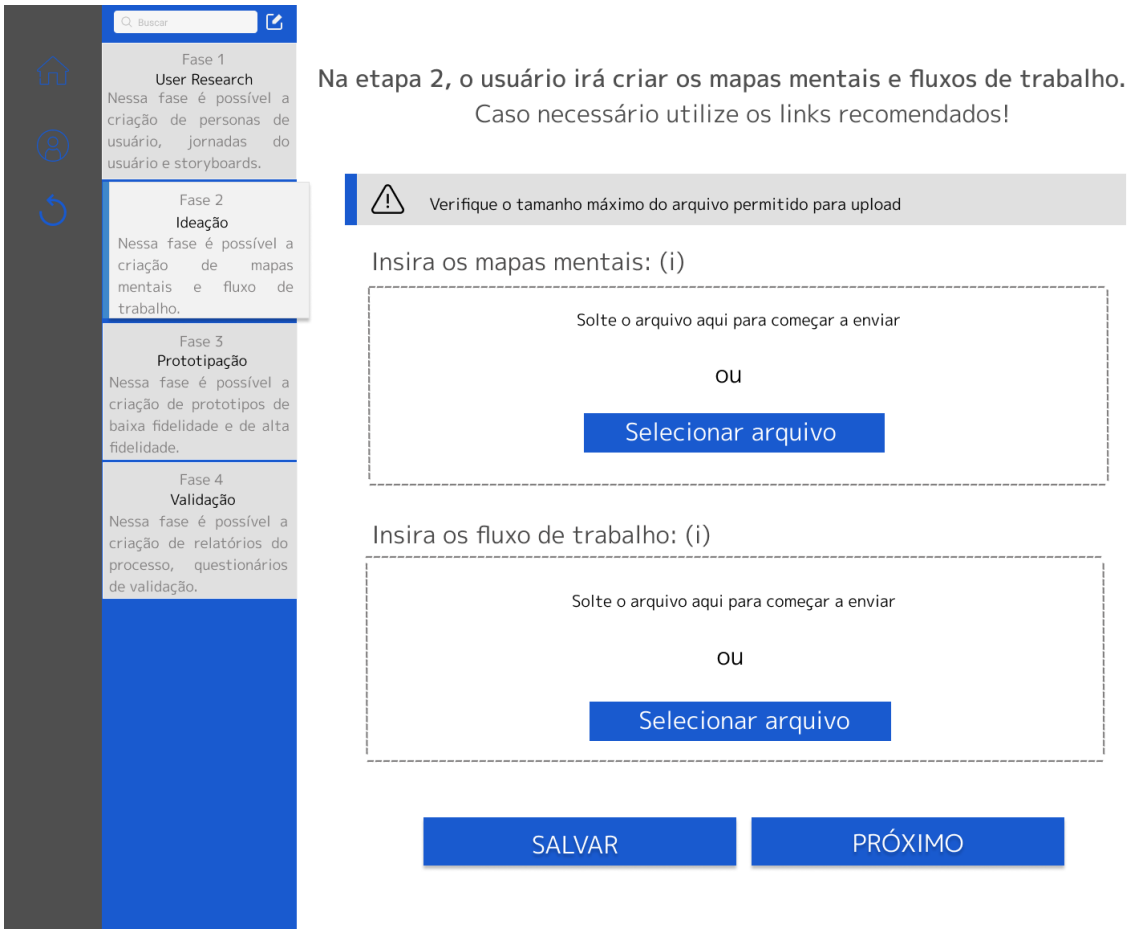
Concluindo as 4 (quatro) fases do *Design Thinking* o usuário do SISREQ é direcionado para a tela de Engenharia de Requisitos (figura 24), na qual pode ocorrer a formalização dos requisitos e a partir da qual pode-se prosseguir com o nível de formalização de documentação desejada para o produto ou projeto.

Figura 20 - Fase 01 - User Research



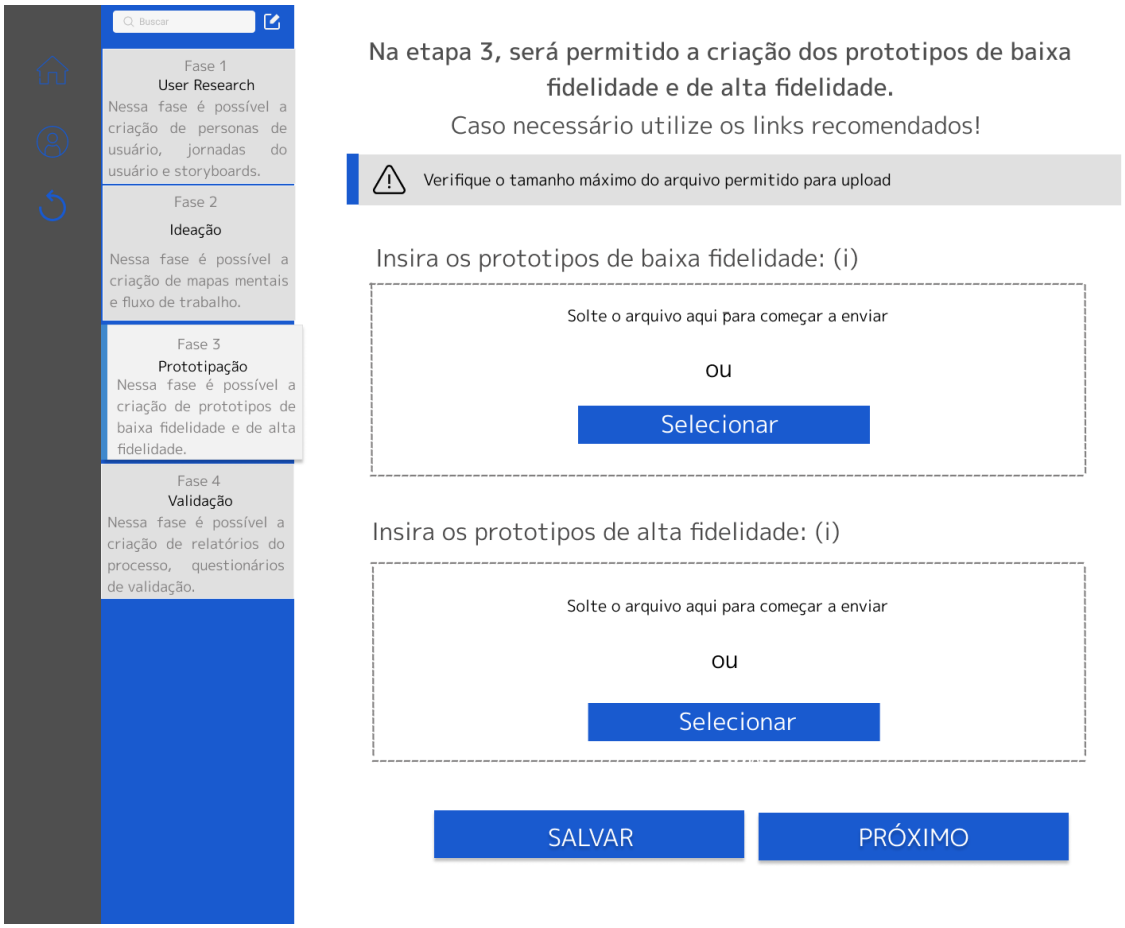
Fonte: Autoria própria (2020)

Figura 21 - Fase 02 - Ideação



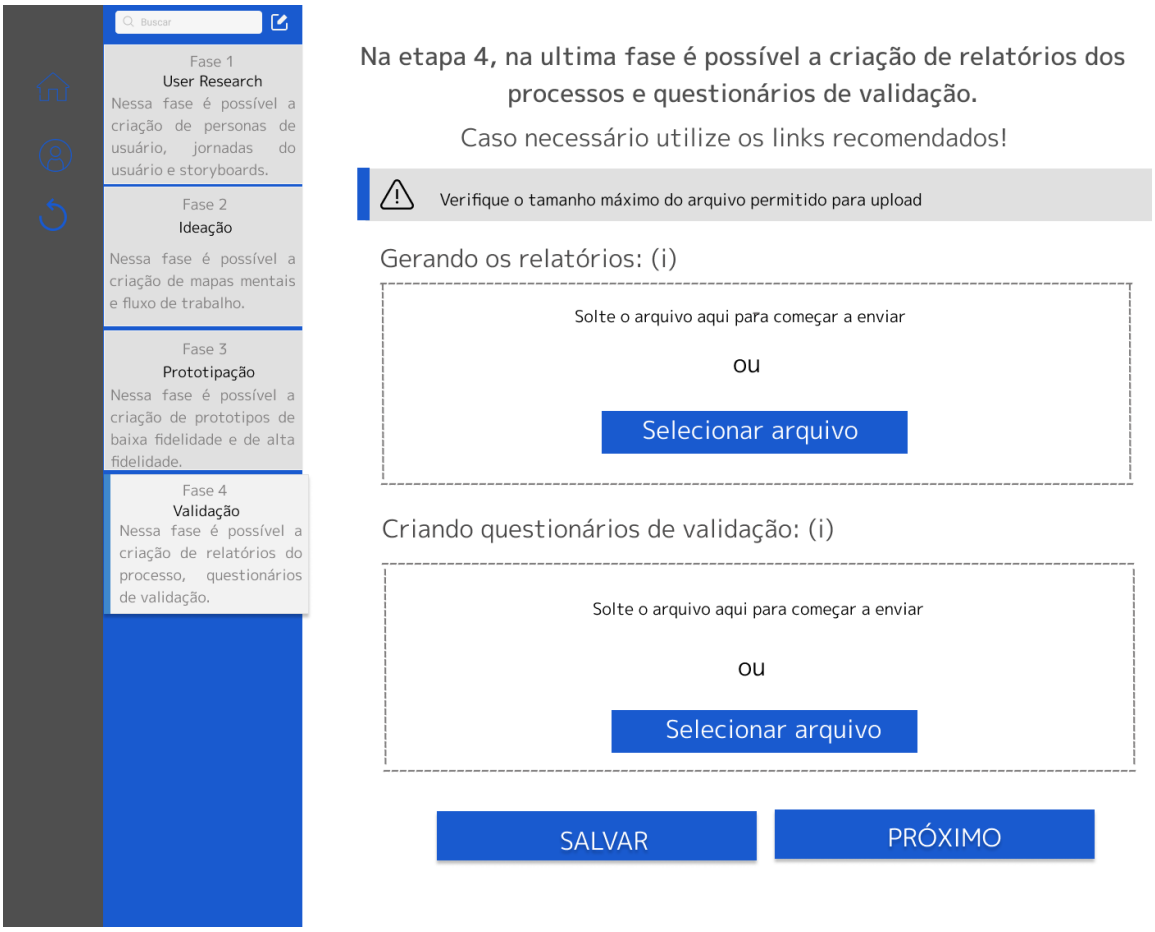
Fonte: Autoria própria (2020)

Figura 22 - Fase 03 - Prototipação



Fonte: Autoria própria (2020)

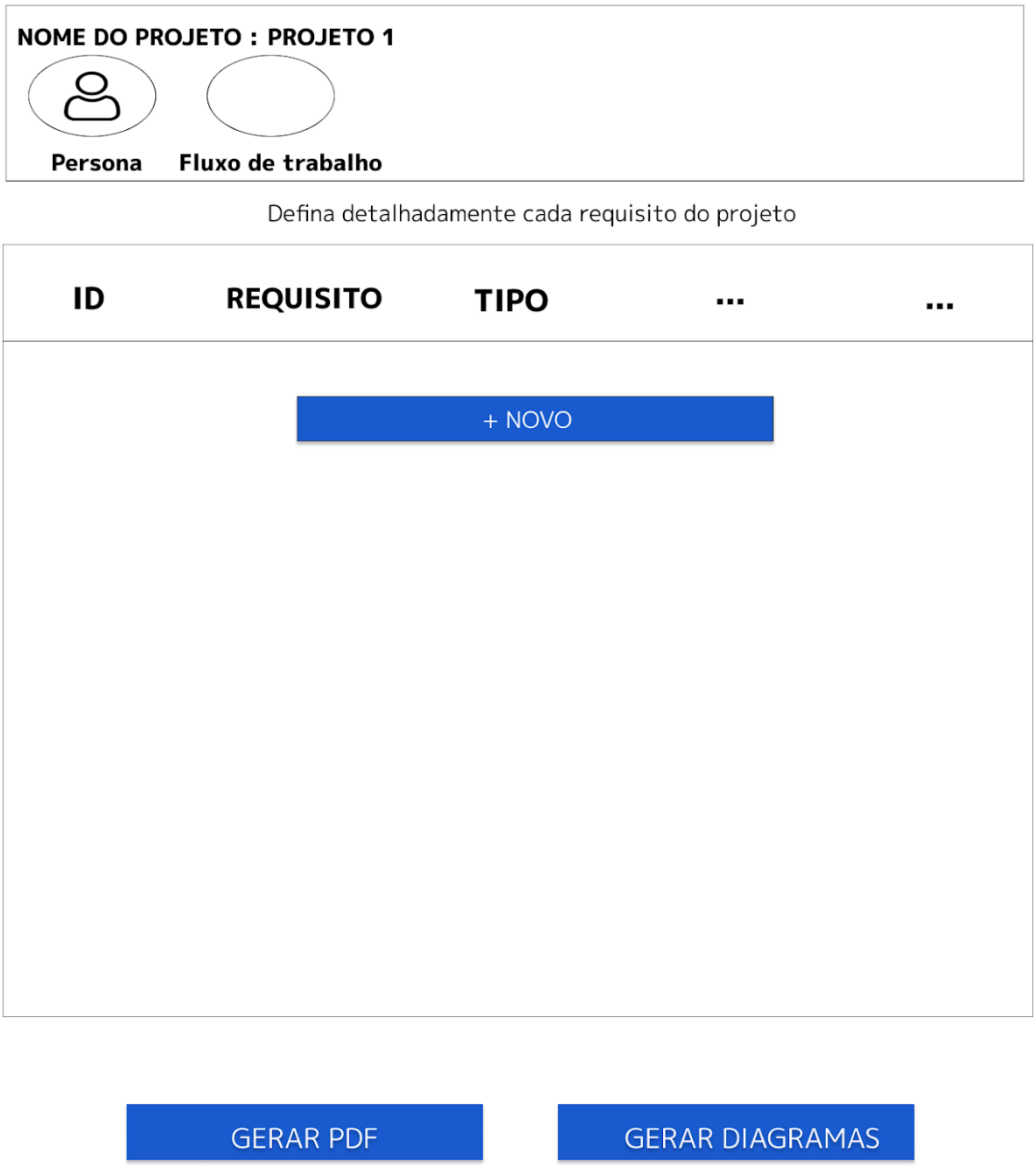
Figura 23 - Fase 04 - Validação



Fonte: Autoria própria (2020)

Na tela Engenharia de Requisitos, última fase da ferramenta, o usuário poderá inserir detalhadamente dados sobre os requisitos do produto ou projeto que está sendo desenvolvido. Após preencher a tabela completa e clicar em concluir a ferramenta irá gerar um arquivo PDF com um relatório sobre o projeto. Neste fluxo também será possível gerar alguns diagramas UML e histórias do usuário a partir dos requisitos gerados.

Figura 24 - Fase Engenharia de Requisitos



Fonte: Autoria própria (2020)

Após criação dos protótipos da ferramenta feitos com base nas funcionalidades descritas na pesquisa na seção anterior, os protótipos foram apresentados aos 5 (cinco) entrevistados que responderam SIM no questionamento sobre “A CRIAÇÃO DE UMA FERRAMENTA WEB PARA ENGENHARIA DE REQUISITOS TRARIA RESULTADOS FAVORÁVEIS” e uma nova entrevista de avaliação da contribuição e análise da ferramenta, foi realizada. Os resultados desta segunda entrevista serão apresentados no capítulo 6.

5.4 ARQUITETURA PROPOSTA

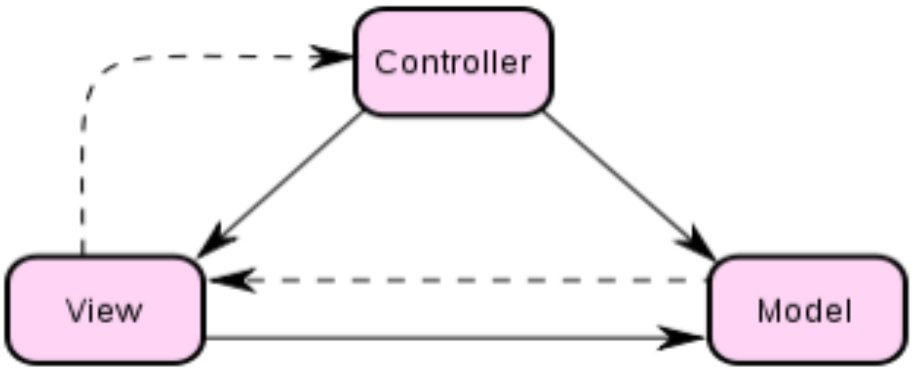
A arquitetura de um *software* pode ser entendida como o modo como um software é organizado, como seus componentes e subsistemas se relacionam (PAULA, 2018).

A arquitetura *web* é baseada na arquitetura cliente servidor, uma rede distribuída e descentralizada. Necessita de *software* para atuar como cliente e como servidor utilizando o protocolo *HTTP*, geralmente são utilizados um navegador (cliente) e um servidor *web* (servidor). O funcionamento da arquitetura *web* é simples, o cliente localiza o servidor e faz a solicitação desejada, o servidor responde a requisição com a solicitação feita, concluindo a requisição. Por ser leve e escalável, a arquitetura *web* pode ser implementada em dispositivos com pouco processamento e memória, (MARTINS, 2014).

5.4.1 Padrão MVC

De Lemos *et. al.* (2013, p. 8) afirmam que **"**o MVC é um padrão de arquitetura que auxilia desenvolvedores a construir aplicações separando os componentes, a manipulação, armazenamento, as funções, e a visualização dos dados. ” Ou seja, a dinâmica do MVC é simples, o MVC direciona todas as solicitações para a camada *Controller*, que acessa a camada *Model* para processar cada solicitação e, por fim, exibe os resultados na camada *View*, conforme mostra a figura 25 abaixo.

Figura 25 - Padrão MVC



Fonte: Luciano, J., & Alves, W. J. B. (2017)

A arquitetura *web* é tipicamente desenvolvida sob a aplicação do MVC no padrão de duas camadas: *Back-End* e *Front-End*.

5.4.2 Back-End

Na programação *web*, o *back-end* é tudo aquilo que vem por trás da aplicação. Funções, regras de negócios, validações e garantias em ambientes nos quais o usuário consegue manipular informação. O *back-end* faz a ligação com o banco de dados integrando dados e a aplicação (SOUTO, 2019).

No *back-end* também são normalmente utilizados alguns *frameworks* bastante difundidos em grandes aplicativos para a *web* pertencentes a grandes empresas, tais como o caso do Node JS, C# e PHP. Para este projeto de arquitetura, foi escolhido o *framework* Node JS para codificação do *back-end* e o banco de dados escolhido foi o NOSQL.

5.4.3 Banco de Dados

Oliveira et. al. (2018, p. 301) afirmam que **“**o banco de dados é uma ferramenta para armazenar e organizar informações. ” Esse armazenamento de informações permite ao usuário de sites e programas segurança na inserção, atualização e pesquisa de dados e também na utilização de produtos e sites.

O banco de dados pode ser relacional, cujo armazenamento é feito em tabelas com pouca escalabilidade e só pode ser codificado em *API’s* com servidores que interpretam a linguagem de manipulação de dados SQL e bancos de dados não relacionais (também conhecido por *NoSQL*) que possuem maior escalabilidade e pode ser codificado em qualquer linguagem de programação (OLIVEIRA, 2018).

5.4.4 Node JS

O Node JS é um ambiente de execução Javascript inicialmente construído no motor v8 Javascript do *Chrome*. O Node JS apresenta arquitetura, flexibilidade, baixo custo e alta capacidade de escala e tem como principal característica a sua execução

ser *single-thread*, ou seja, apenas um *thread* é responsável por executar o código Javascript da aplicação. O Node JS é bastante conhecido e é utilizado em grandes sites como a Netflix, *LinkedIn* e Uber (LENON, 2018). A figura 26 apresenta a tela inicial do site do Node JS.

Figura 26 - Node JS

Fonte: Captura do site do Node JS. Acesso em 23 de novembro de 2020

Com o advento do Node JS, tornou-se possível desenvolver tanto o *back-end* como o *front-end* em Javascript, tornando essa linguagem de programação mais escalável.

5.4.5 Front-End

Na programação *web*, o *front-end* consiste na parte visível ao usuário, ou seja, toda a parte visual de uma aplicação *web*, com a qual o usuário de tal site consegue interagir. Tipicamente, as principais tecnologias utilizadas são o HTML, CSS, Javascript, que são a base da programação web. Com o avanço da tecnologia, diversos *frameworks* que implementam tais tecnologias base surgiram, e são bastante utilizados em grandes sites e empresas, tais como React, o Angular, e o Vue Js (SOUTO, M., 2019).

O *framework* escolhido para ser usado no desenvolvimento do *front-end* deste trabalho foi o React.

5.4.5.1 React

React é uma biblioteca Javascript declarativas, eficientes, flexíveis para a criação de interfaces de usuários. Ele permite a criação de interfaces gráficas para interação com o usuário complexas, e baseada em componentes (REACT, 2020), a figura 27 apresenta tela inicial do site do React.

Figura 27 - React

Fonte: Captura do site React (2020). Acesso em 23 de novembro de 2020

6 ANÁLISE

Este capítulo apresenta o processo de análise da ferramenta proposta, compreendendo a entrevista de avaliação e análise dos dados.

6.1 ENTREVISTA

Com base nas funcionalidades coletadas, foi realizada a modelagem da ferramenta e a criação dos protótipos (conforme apresentados no capítulo anterior), e posteriormente uma segunda entrevista foi realizada, com a finalidade de avaliar a eficiência e aplicabilidade da proposta. A entrevista, que pode ser encontrada no Apêndice B, foi realizada com 5 profissionais do desenvolvimento de *software*, sendo 3 empresas de desenvolvimento de *software* e 2 desenvolvedores de *software* independentes.

6.2 ANÁLISE DE RESULTADOS DA PESQUISA

Com o objetivo de avaliar o nível de satisfação dos desenvolvedores entrevistados e compreender a eficiência da ferramenta proposta, esta pesquisa utilizou o método NPS com adaptações e adequações. Durante a segunda entrevista, de avaliação, foi aplicado um questionário, com 4 (quatro) perguntas, que tinha o objetivo de medir *eficiência e eficácia ←→ medições realizadas pelas perguntas ao adaptar o método NPS, a partir das quais foi possível realizar tais medições*. Os dados foram calculados com base no método proposto e são apresentados no Quadro 1:

Quadro 1 - Perguntas adaptadas conforme método NPS

Nº	Pergunta realizada
1	Em uma escala de 0 a 10, qual seu grau de satisfação com a ferramenta proposta?

2	Em uma escala de 0 a 10, quanto a ferramenta acrescentaria para a sua empresa ou atividade de desenvolvimento independente?
3	Em uma escala de 0 a 10, o quanto a ferramenta proposta contribuiria nos projetos da sua empresa ou atividades de desenvolvimento independentes?
4	Em uma escala de 0 a 10, quanto você indicaria a ferramenta proposta a parceiros de negócio e amigos desenvolvedores?

Fonte: Autoria própria (2020)

As perguntas 1 (um) e 4 (quatro) estão baseadas no grau de satisfação dos entrevistados e as perguntas 2 (dois) e 3 (três) estão baseadas no grau de influência da ferramenta na empresa ou na atividade independente dos respondentes. Não houve cálculo de margem de erro, considerando que o número da população e o número da amostra são iguais.

Em relação à pergunta 1, “Em uma escala de 0 a 10, qual seu grau de satisfação com a ferramenta proposta? ”, pode-se classificar os entrevistados em 4 (quatro) promotores e 1 (um) passivo ou neutro, conforme apresentado no Quadro 2.

Quadro 2 - Grau de satisfação dos entrevistados

Pergunta	Promotores	Neutros	Detratores	NPS	Total
Em uma escala de 0 a 10, qual seu grau de satisfação com a ferramenta proposta?	4 (80%)	1 (20%)	0 (0%)	(P-D) (80%)	5 (100%)

Fonte: Dados da pesquisa (2020)

Conclui-se, portanto, de acordo com as respostas apresentadas no Quadro 2, baseado no critério de avaliação do método NPS, que o grau de satisfação dos entrevistados em relação a ferramenta proposta se enquadra na “Zona de Excelência”, de modo que de acordo com ponto de vista dos entrevistados a

ferramenta apresentada foi considerada excelente e com grande grau de utilidade em projetos de *software*.

Com respeito à pergunta 2, “Em uma escala de 0 a 10, quanto a ferramenta acrescentaria para a sua empresa e atividades de desenvolvimento independentes? ”, de acordo com as respostas fornecidas, os entrevistados se classificaram em 2 (dois) promotores, 2 (dois) passivos ou neutros e 1 (um) detrator, conforme apresentado no Quadro 3.

Quadro 3 - Grau de acréscimo da ferramenta em projetos

Pergunta	Promotores	Neutros	Detratores	NPS	Total
Em uma escala de 0 a 10, quanto a ferramenta acrescentaria para a sua empresa ou atividade de desenvolvimento independente?	2 (40%)	2 (40%)	1 (20%)	P-D (20%)	5 (100%)

Fonte: Dados da pesquisa (2020)

Percebe-se, portanto, de acordo com as respostas apresentadas no Quadro 3, que o grau de contribuição da ferramenta em projetos de *software* se enquadra na “Zona de Aperfeiçoamento”, já que, devido à grande demanda de projetos, a ferramenta seria melhor aproveitada em grandes empresas de desenvolvimento de *software*. Em empresas menores, embora a ferramenta possa contribuir positivamente, a percepção de tal contribuição não seria tão notável, devido à demanda menor no desenvolvimento de requisitos e também ao fato de que requisitos de projetos de *software* em tais organizações normalmente são mais voláteis (mudam facilmente). Os entrevistados ainda relataram que como o tempo é a principal restrição em pequenas empresas, visto que não existe a divisão de atividades, inserir mais uma ferramenta poderia acarretar atrasos nos projetos.

Em relação às respostas obtidas na pergunta 3, “Em uma escala de 0 a 10, o quanto a ferramenta proposta contribuiria nos projetos da sua empresa ou atividades de desenvolvimento independentes? ”, pode-se classificar os entrevistados em 2

(dois) promotores, 2 (dois) passivos ou neutros e 1 (um) detrator, conforme apresentado no quadro 4.

Com efeito, de acordo com as respostas apresentadas no Quadro 4, o grau de contribuição da ferramenta em projetos de *software* se enquadra na “Zona de Aperfeiçoamento”, pois os entrevistados acreditam que a ferramenta teria maior contribuição em grandes projetos com grandes equipes de desenvolvimento e que, para projetos simples e com equipes menores, ela não teria o impacto esperado.

Quadro 4 - Grau de contribuição da ferramenta

Pergunta	Promotores	Neutros	Detratores	NPS	Total
Em uma escala de 0 a 10, o quanto a ferramenta proposta contribuiria nos projetos da sua empresa ou atividades de desenvolvimento independentes?	2 (40%)	2 (40%)	1 (20%)	P-D (20%)	5 (100%)

Fonte: Dados da pesquisa (2020)

Em relação à pergunta 4, “Em uma escala de 0 a 10, quanto você indicaria a ferramenta proposta a amigos desenvolvedores? ”, os resultados de suas respostas tornaram possível classificar os entrevistados em 4 (quatro) promotores e 1 (um) passivo ou neutro, conforme apresentado no Quadro 5.

Nesse caso, os entrevistados indicam um alto grau de indicação da ferramenta, deixando-a “Zona de Excelência” do método NPS, seguindo o entendimento de que a ferramenta pode trazer grandes contribuições a projetos de *software*.

Quadro 5 - Grau de indicação da ferramenta

Pergunta	Promotores	Neutros	Detratores	NPS	Total
Em uma escala de 0 a 10, quanto você indicaria a	4 (80%)	1 (20%)	0 (0%)	P-D (80%)	5 (100%)

ferramenta proposta a amigos desenvolvedores?					

Fonte: Dados da pesquisa (2020)

De acordo com os dados analisados acima, pode-se entender que a principal contribuição da ferramenta proposta é a otimização de tempo nos processos da ER. Se utilizada em organizações e empresas que desenvolvem *software*, os entrevistados entendem que a ferramenta proposta é satisfatória e necessária para empresas de desenvolvimento. No entanto, possui funcionalidades e características que precisam ser melhor analisadas quanto ao seu grau de adequação para determinados tipos de projetos. Há também recursos faltantes, ou seja, funcionalidades que precisam ser adicionadas para aumentar a percepção da contribuição da ferramenta no dia a dia dos projetos de desenvolvimento de *software*.

Os pontos levantados acima apontam que ainda falta algum esclarecimento adicional com respeito a características e requisitos do SISREQ, mas essa necessidade já era esperada e tais esclarecimentos devem ocorrer naturalmente durante as fases posteriores do desenvolvimento da ferramenta. Assim, pode-se afirmar que a proposta realizada por esta pesquisa é indicada para desenvolvimento de uma ferramenta como produto mínimo viável (MVP), considerando o nicho de mercado da Engenharia de Requisitos/Engenharia de *Software*.

7 CONCLUSÃO

O presente capítulo apresenta a conclusão desta pesquisa, composto por considerações finais, contribuições da pesquisa e sugestões de trabalhos futuros.

7.1 CONSIDERAÇÕES FINAIS

O presente trabalho buscou elaborar a proposta de uma ferramenta *web* para Engenharia de Requisitos, intitulada “SISREQ”, que visa contribuir com melhorias significativas no processo de desenvolvimento de *software*. A ferramenta proposta utiliza o processo do *Design Thinking* que utiliza técnicas inovadoras para resolução de problemas relevantes para clientes e usuários de projetos de *software*.

As principais funcionalidades da ferramenta proposta foram coletadas em entrevista feita com profissionais de desenvolvimento de *software*, a partir da qual, posteriormente, foram gerados protótipos. Em seguida, tais protótipos foram avaliados por profissionais da área de desenvolvimento de *software*, em nova entrevista, a fim de analisar a eficiência da ferramenta proposta a partir dos protótipos apresentados.

O método de avaliação realizado foi o NPS, que classificou a ferramenta na Zona de Excelência quanto a satisfação e indicação da ferramenta e Zona de Aperfeiçoamento quanto a acréscimo e contribuição em projetos de *software*. Esse nível de classificação permite concluir que: i) alguns pontos da ferramenta precisam de um refinamento adicional, algo que deve acontecer nas fases posteriores de desenvolvimento da ferramenta, e que ii) é possível afirmar que o desenvolvimento de um produto mínimo viável (MVP) é indicado, tomando como base os resultados da pesquisa até o momento.

7.2 CONTRIBUIÇÕES DA PESQUISA

Diante do que foi apresentado, esta pesquisa traz como contribuição a proposta de um *software* profissional, especificamente uma ferramenta *web*, que serve para utilização em empresas de desenvolvimento de *software* para a coleta de

requisitos dos projetos de cliente, com o objetivo de gerar otimização de tempo e redução de trabalho.

A presente pesquisa também trouxe como contribuição um modelo de elaboração de projetos composto pelo *framework Scrum*, ferramentas auxiliares e os artefatos apresentados da Engenharia de *Software.* Tais artefatos, que servirão como base para desenvolvimento da ferramenta, são auxiliares no desenvolvimento de *software* por meio do *framework Scrum*. A ferramenta Jira foi utilizada de forma auxiliar ao método para elaborar dos requisitos na forma de *backlog* de produto, planejar as *Sprints* iniciais de desenvolvimento de código e detalhar o *backlog* das *sprints* executadas até o momento da escrita deste trabalho. A ferramenta Figma foi utilizada para a prototipação do SISREQ. Por sua vez, a arquitetura proposta da ferramenta foi o MVC juntamente com o uso de linguagens e *frameworks* tais como o Javascript, React, Node JS.

Por fim, também trouxe como contribuição o método avaliativo utilizado na avaliação da proposta, o método NPS (*Net Promoter Score*).

7.3 SUGESTÕES PARA TRABALHOS FUTUROS

Como trabalho futuro para esta pesquisa, pode-se destacar a utilização dos mecanismos aqui apresentados como ponto de partida para o desenvolvimento da ferramenta SISREQ como um MVP. Posteriormente, será possível realizar a integração do MVP com outras ferramentas de desenvolvimento de produtos. A ideia é que a Engenharia de Requisitos seja o ponto de partida e esteja integrada com as demais etapas da Engenharia de *Software* em projetos tradicionais de desenvolvimento, ou que sirva como a fase de elaboração da Visão do Produto (*Inception*) em projetos ágeis. A proposta aqui realizada se diferencia no âmbito da pesquisa no desenvolvimento ágil de projetos de *software* na medida em que serve ambos os propósitos.

REFERÊNCIAS

ATLASSAIN. **Ferramentas Ágeis para equipes ágeis**. Disponível em: https://www.atlassian.com/br. Acesso em: 30 set. 2020

AREND, Rodrigo Cesar. **Implementação de aplicação Web para controle de gastos da central telefônica da UTFPR**. 2013. 70f. Trabalho de conclusão de curso – Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná. Pato Branco, 2013.

AZENHA, Flávio Copola; FLEURY, Andre Leme. ANALISE DO DESIGN THINKING COMO UMA ABORDAGEM DE ELICIAÇÃO DE REQUISITOS PARA O DESENVOLVIMENTO DE APLICAÇÕES WEB/MOBILE. **XXXVIII ENCONTRO NACIONAL DE ENGENHARIA DE PRODUCAO.** Alagoas, 2018.

BARBOSA, Simone; SILVA, Bruno. **Interação humano-computador**. Elsevier Brasil, 2010.

BERNARDO, Kleber. **Product Backlog: O que é?** Cultura Ágil, 2015. Disponível em: https://www.culturaagil.com.br/product-backlog-o-que-e/. Acesso em: 02 nov. 2020.

CAMARGO, Danieli A. Estudo das técnicas de desenvolvimento web e validação deste estudo com um portal para recicladores e produtores de reciclados. 2009. **Anuário da Produção de Iniciação Cientifica Discente**, v. XII, n. 14, 2009.

CENTENARO, Jonas. **Desenvolvimento de um software web para gerenciamento de requisitos de software.** Trabalho de Conclusão de Curso de Especialização (Especialista em Desenvolvimento de Sistemas para Internet e Dispositivos Móveis). Francisco Beltrão, 2014. Disponível em: http://repositorio.roca.utfpr.edu.br/jspui/handle/1/7192. Acesso em: 9 nov. 2019.

DA COSTA, Elvio Carlos. A Importância da Engenharia De Requisitos no Processo de Desenvolvimento de Sistemas de Informação. **Revista Interface Tecnológica**, v. 15, n. 1, p. 203-214, 2018.

DA SILVA, Rogério Oliveira; CURADO, Luis Augusto Trindade; MACHADO, Giselle Barbosa Gomes. Entendendo o Desenvolvimento Ágil. **TECNOLOGIAS EM PROJEÇÃO**, v. 7, n. 2, 2016.

DA SILVA, Rogério Oliveira; NEVES, Douglas Martins; DE MELO, Leonardo Paiva Campos. Uma breve visão sobre a metodologia scrum dos discentes de sistema de informação da faculdade projeção de Sobradinho/DF. **TECNOLOGIAS EM PROJEÇÃO**, v. 8, n. 1, p. 40-50, 2017.

DE SOUZA OLIVEIRA, Moacir et al. BANCO DE DADOS NO-SQL X BANCO DE DADOS SQL. **South American Development Society Journal**, v. 4, n. 11, p. 298, 2018.

DE SOUZA, Sérgio Cozzetti Bertoldi et al. Documentação essencial para manutenção de software II. In: **IV Workshop de Manutenção de Software Moderna (WMSWM)**, Porto de Galinhas, PE. 2007.

DE LEMOS, Maxmilian Ferreira et al. Aplicabilidade da arquitetura MVC em uma aplicação web (WebApps). **RE3C-Revista Eletrônica Científica de Ciência da Computação**, v. 8, n. 1, 2013.

ESPINDOLA, Rodrigo Santos de; MAJDENBAUM, Azriel; AUDY, Jorge Luis Nicolas. Uma Análise Crítica dos Desafios para Engenharia de Requisitos em Manutenção de Software. In: **VII Workshop on Requirements Engeneering**, 2004. p. 226-238.

FALAVIGNA, Vinícius Deboni. Experiência do usuário: análise e aplicação de métodos de avaliação. 2015.

FARACHE, Bruno. **Trace Tracker: um sistema para gerenciamento de rastreabilidade**. 2007. 48 f. TCC (Graduação) - Centro de Informática da Universidade Federal de Pernambuco, Pernambuco, 2007. Disponível em: https://www.cin.ufpe.br/~tg/2006-2/bf.pdf. Acesso em: 26 nov. 2019.

FARMOUDEHYAMCHEH, Parisa. **A Systems Approach to Graphic Design Practice**. 2019.

FREITAS, Fátima Carolina França. **Programação Web UX/UI Design**. 2017. Relatório de Estágio de Mestrado em Novos Media e Práticas Web. FCSH: DCM - Dissertações de Mestrado. Disponível em: https://run.unl.pt/handle/10362/23790. Acesso em: 4 nov. 2019.

FONTELLES, Mauro José et al. Metodologia da pesquisa científica: diretrizes para a elaboração de um protocolo de pesquisa. **Revista Paraense de Medicina**, v. 23, n. 3, p. 1-8, 2009.

GASTALDO, Denise Lazzeri; MIDORIKAWA, Edson Toshimi. Processo de Engenharia de Requisitos Aplicado a Requisitos Não - Funcionais de Desempenho – Um Estudo de Caso. In: **Workshop em Engenharia de Requisitos**. Piracicaba. p. 302-316. 2003.

GRANDE, José Inácio de; MARTINS, Luiz Eduardo G. SIGERAR: Uma Ferramenta para Gerenciamento de Requisitos. In: **Workshop em Engenharia de Requisitos**. Rio de Janeiro: RJ. p. 75-83, 2006.

GRINGS, Claiton Luís; SAYÃO, Miriam. **OpenReq: uma Ferramenta para Auxílio à Gerência de Requisitos. In: Workshop em Engenharia de Requisitos**. Pontifícia Universidade Católica do Rio Grande Do Sul, Porto Alegre, 2009.

HEVNER, Alan R., et al. **Design Science in Information Systems Research. MIS Quarterly**, vol. 28. n. 1. p. 75-105. 2004.

IEEE Std. 610.12- 1990. **IEEE Standard Glossary of Software Engineering Terminology**. New York: The Institute of Electrical and Electronics Engineers, 1990.

KUERTEN, Rodrigo; PALMA, Jandira Guenka. **Reduzindo mudanças de requisitos no desenvolvimento de software usando Modelagem Independente de Computação e UX Design.** 2019. Sistema, v. 1, n. 2, p. 3.

Lenon. Node.js – **O que é, como funciona e quais as vantagens**. Opus-software, 2018. Disponível em: https://www.opus-software.com.br/node-js/ Acesso em: 23 nov. 2020.

LINKEDIN. **Bem-vindo a sua comunidade profissional**. Disponível em: https://br.linkedin.com/. Acesso em: 23 nov. 2020.

LONGO, Luci; MEIRELLES, Fernando de Souza. Impacto dos investimentos em tecnologia de informação do desempenho financeiro das indústrias brasileiras. **REAd. Revista Eletrônica de Administração**. Porto Alegre, 2016, vol.22, n.1, pp.134-165. Disponível em: http://www.scielo.br/pdf/read/v22n1/1413-2311read-22-01-0134.pdf. Acesso em: 16 nov. 2019.

LUCIANO, Josué; ALVES, Wallison Joel Barberá. **PADRÃO DE ARQUITETURA MVC: MODEL-VIEW-CONTROLLER**. 2017.

MACIEIRA, Leonardo Gabriel. Metodologia ágil para gestão e planejamento de projetos: implantação do Scrum e do framework Jira na qualidade em uma empresa do setor aeronáutico. **Anais do X SIMPROD**, 2018.

MATHIAS, Lucas. **5 exemplos de Design Thinking para se inspirar na busca por inovação**. Mindminers, 2018. Disponível em: https://mindminers.com/blog/exemplos-de-design-thinking/. Acesso em: 25 mar. 2020.

MATERIAL-UI. **Biblioteca de componentes React para um desenvolvimento ágil e fácil. Construa seu próprio design, ou comece com Material Design**. Material-ui, 2020. Disponível em: https://material-ui.com/pt/ Acesso em: 25 nov. 2020.

MARTINS, Gilberto Andrade. Estudo de caso: uma reflexão sobre a aplicabilidade em pesquisas no Brasil. **Revista de Contabilidade e Organizações**, v. 2, n. 2, p. 8-18, 2008.

MARTINS, Lucas Maurício Castro. **O Processamento web no cliente: O" downsizing" da arquitetura web**. 2014.

MELO, Ricardo Alves de. **Um estudo qualitativo do perfil de elicitador de requisitos orientado a obtenção do sucesso de projetos de software**. 2011. Dissertação de Mestrado. Universidade Federal de Pernambuco.

OLIVEIRA, José Clovis Pereira de et al. O questionário, o formulário e a entrevista como instrumentos de coleta de dados: vantagens e desvantagens do seu uso na

pesquisa de campo em ciências humanas. In: **III Congresso Nacional de Educação. Rio Grande do Norte**. 2013.

OLIVEIRA, Eduardo Alves; VIEIRA Filho, Fernando Castro; KOVALESKI, João Luiz. Investigação e análise da satisfação de clientes usando o método net promoter score para promover melhorias de produtos e processos. **Revista Uningá Review**, v. 28, n. 3, 2016.

PAULA, Diego Henrique Incerti. Arquitetura Rest: **Uma alternativa para melhorar o desempenho de aplicações com serviços web**. 2018. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) – Campinas, 2018.

PINHEIRO, Allan Petterson da Silva. **UX design introduzido no desenvolvimento de interfaces gráficas**. 2016. Trabalho de Conclusão de Curso (Tecnólogo em Comunicação Social – Design Gráfico) – UniCEUB. Brasília – DF, 2016. Disponível em: https://repositorio.uniceub.br/jspui/handle/235/9445. Acesso em: 28 out. 2019.

PRESSMAN, Roger S. **Engenharia de Software: uma abordagem profissional**. Porto Alegre: AMGH. 7.ed, 2011.

PRIKLADNICKI, Rafael. **Problemas, Desafios e Abordagens do Processo de Desenvolvimento de Software**. 2004. Trabalho Individual I, Mestrado em Ciência da Computação, PUCRS, 2004.

RAMÍREZ, Diana Marcela Bernal; ZANINELLI, Thais Batista. O Uso do Design Thinking como Ferramenta no Processo de Inovação em Bibliotecas. **Encontros Bibli: revista eletrônica de biblioteconomia e ciência da informação**, v. 22, n. 49, p. 59-74, 2017. Disponível em: https://doi.org/10.5007/1518-2924.2017v22n49p59. Acesso em: 8 nov. 2019.

RAMPINELLI, F. **TUDO SOBRE NPS: COMO CALCULAR, CLASSIFICAÇÕES E VANTAGENS**. DDS, 2019. Disponível em:https://www.dds.com.br/blog/index.php/entenda-importancia-nps-para-sua-empresa/. Acesso em: 30 nov. 2020.

RAVANELLO, Ivna Motta; WOLFF, Fabiane; RIBEIRO, Vinicius Gadis. Uma Revisão Sistemática da Produção Bibliográfica sobre Experiência do Usuário no Campo do Design. 2016. **Revista ErgodesignHCI**, v. 4, n. Especial, p. 1-8. Disponível em: http://periodicos.puc-rio.br/index.php/revistaergodesignhci/article/view/124. Acesso em: 4 nov. 2019.

RIES, Eric. The Lean Startup: how Constant Innovation Creates Radically Successful Businesses, **Penguin Books**. 2011.

RIZO, Victor Hugo Manduca; DO ESPÍRITO SANTO, Felipe. **MIGRAÇÃO DE PARTES DE UMA APLICAÇÃO DESKTOP PARA O FORMATO DE API REST: estudo de caso**. Revista Interface Tecnológica, v. 17, n. 1, p. 118-128, 2020.

ROSA, Luis Henrique Carvalho et al. Jogos para Ensino de Levantamento de Requisitos de Software: uma Revisão Sistemática de Literatura. **RENOTE**, v. 15, n. 2, 2017.

SANTIAGO, Marcos R. **Ensaio do SWEBOK–Software Engineering Body of Knowledge**. Universidade Gama Filho. Goiânia, 2011.

SANTOS, Wylliams Barbosa et al. Simplicidade no Desenvolvimento Ágil de Software: Um Mapeamento Sistemático da Literatura. **Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação**, v. 1, n. 8, 2018.

SILVA, Karla Michele Barbosa da. **Método para criação de requisitos completos no processo de desenvolvimento de software**. 2019. Dissertação de Mestrado. Universidade Federal de Pernambuco.

SOMMERVILLE, Ian. **Engenharia de Software**. Tradução de Ivan Bosnic e Kalinka G. de O. Gonçalves. Revisão técnica de Kechi Hirama. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SOUTO, M. **O que é front-end e back-end?** Alura, 2019.Disponível em: https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end acesso em: 21 nov. 2020.

SOUZA, Anderson Felipe Barros de; FERREIRA, Bruna Moraes; CONTE, Tayana. Aplicando Design Thinking em Engenharia de Software: Um mapeamento sistemático. 2017. In: **Proceedings on Ibero-American Conference on Software Engineering: Experimental Software Engineering e Latin America Workshop (CIbSE-ESELAW)**, pp. 719–732. Buenos Aires: Argentina, 2017.

SOUZA, Sérgio Cozzetti Bertoldi de et al. Documentação essencial para manutenção de software II. In: **IV Workshop de Manutenção de Software Moderna (WMSWM)**, Porto de Galinhas - PE, 2007.

SUPAKKUL, Sam; CHUNG, Lawrence. The RE-Tools: A multi-notational requirements modeling toolkit. In: **2012 20th IEEE International Requirements Engineering Conference (RE)**. IEEE, 2012. p. 333-334. Disponível em: https://ieeexplore.ieee.org/abstract/document/6345831. Acesso em: 23 nov. 2019.

SUTHERLAND, J. J. **SCRUM: guia prático**. Primeira Pessoa, 2020.

TEIXEIRA, Cássio Adriano Nunes; CUKIERMAN, H. Apontamentos para Enriquecer o Perfil do Engenheiro de Software. In: **XXV Congresso da Sociedade Brasileira de Computação**. São Leopoldo/RS: Unisinos. p. 2314- 2325, 2005.

TUTORIAL: Introdução ao React. React, 2020. Disponível em: https://pt-br.reactjs.org/tutorial/tutorial.html. Acesso em: 24 nov. 2020

VIANNA, Mauricio, et. al. **Design Thinking: inovação em negócios**. Rio de Janeiro: MJV Press, 2012.

WAZLAWICK, Raul. **Engenharia de software: conceitos e práticas**. Elsevier Editora Ltda., 2019.

APÊNDICE A – ROTEIRO DA ENTREVISTA INICIAL COM GRUPO DE FOCO

Olá, meu nome é Maria Luiza, sou aluna do curso de Ciência da Computação da Universidade Estadual da Paraíba. Para o desenvolvimento da minha pesquisa cientifica intitulada "**PROPOSTA DE DESENVOLVIMENTO DE UMA FERRAMENTA WEB PARA ENGENHARIA DE REQUISITOS UTILIZANDO DESIGN THINKING**" faz-se necessário uma entrevista com profissionais da área do desenvolvimento de software com objetivo de coleta de informações sobre o processo de desenvolvimento de software nas empresas. Todos os dados e respostas aqui coletados são absolutamente confidenciais e serão utilizados de forma exclusiva para contribuição cientifica aqui solicitada.

ROTEIRO DA ENTREVISTA

Parte I – Características Pessoais:

( ) Pessoa Jurídica ( ) Pessoa Física

Nome:

Segmento:

Parte II – Conhecimentos Gerais

Como é a composição da equipe de desenvolvimento, distribuição de tarefas?

Como é feito o contato com o cliente para obter detalhes/ características do projeto que será desenvolvido?

O desenvolvimento de software é composto por etapas, você segue estas etapas para o desenvolvimento de projetos dos seus clientes? E as etapas da Engenharia de Requisitos?

Em relação a Engenharia de Requisitos, como é feita a coleta dos requisitos e como e onde (forma) são armazenados/ anotados?

Na sua opinião, a criação de uma ferramenta web que auxilie o processo da engenharia de requisitos seria algo favorável?

Se sim, quais funcionalidades você julgaria indispensáveis para que essa ferramenta fosse utilizada?

Se não, porque não?

APÊNDICE B - PESQUISA DE FEEDBACK DA FERRAMENTA SISREQ

Após análise do protótipo apresentado, solicito-lhe o feedback da proposta da ferramenta web para Engenharia de Requisitos intitulada “**SISREQ**”. A ferramenta web proposta nesta pesquisa trata-se de uma ferramenta que visa contribuir para a melhoria no processo de desenvolvimento de software gerando otimização de tempo e trabalho utilizando o processo do Design Thinking, sua avaliação é muito importante para a criação e finalização do produto.
Responda o questionário abaixo enviado utilizando o método NPS (Net Promoter Score) em escala de 0 a 10 e realize sua avaliação da ferramenta proposta.

Em uma escala de 0 a 10, qual seu grau de satisfação com a ferramenta proposta?

1	2	3	4	5	6	7	8	9	10
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

Em uma escala de 0 a 10, quanto a ferramenta acrescentaria para a sua empresa ou atividade de desenvolvimento independente?

1	2	3	4	5	6	7	8	9	10
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

Em uma escala de 0 a 10, o quanto a ferramenta proposta contribuiria nos projetos da sua empresa ou atividades de desenvolvimento independentes?

1	2	3	4	5	6	7	8	9	10
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

Em uma escala de 0 a 10, quanto você indicaria a ferramenta proposta a amigos desenvolvedores?

1	2	3	4	5	6	7	8	9	10
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯