UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Paulo Vitor de Queiroz Zanele

Uma abordagem sobre documentação de testes de software no contexto ágil

São José do Rio Preto
2022

Paulo Vitor de Queiroz Zanele

Uma abordagem sobre documentação de testes de software no contexto ágil

Trabalho de Conclusão de Curso (TCC) apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Conselho de Curso de Bacharelado em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Orientadora:

Profa. Dra. Rogéria Cristiane Gratão de Souza

São José do Rio Preto
2022

Z28a

Zanele, Paulo Vitor de Queiroz
Uma abordagem sobre documentação de testes de software no contexto ágil / Paulo Vitor de Queiroz Zanele. -- São José do Rio Preto, 2022
72 p.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto
Orientadora: Rogéria Cristiane Gratão de Souza

1. Engenharia de software. 2. Documentação. 3. Desenvolvimento ágil de software. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Paulo Vitor de Queiroz Zanele

Uma abordagem sobre documentação de testes de software no contexto ágil

Trabalho de Conclusão de Curso (TCC) apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Conselho de Curso de Bacharelado em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Comissão Examinadora:

Profa. Dra. Rogéria Cristiane Gratão de Souza
Orientadora

Prof. Dr. Carlos Roberto Valêncio

Prof. Dr. Leandro Alves Neves

São José do Rio Preto
2022

Para Deus, meus pais e todas pessoas queridas da minha família

Agradecimentos

Em primeiro lugar, agradeço a Deus, por proporcionar todas minhas alegrias e também por me ajudar a superar as dificuldades encontradas.

Agradeço aos meus pais, por todo carinho e dedicação que recebi durante minha vida, principalmente minha mãe que sempre me apoiou e confiou a minha decisão de estudar fora, em uma universidade pública de qualidade. Seu apoio foi fundamental para meu ingresso no ensino superior e agora sua conclusão. Sem ela não teria chegado aonde cheguei.

Agradeço a minha orientadora Profa. Dra. Rogéria Cristiane Gratão de Souza, pela paciência, conselhos e orientações ao longo da realização desta disciplina, e também pelo tempo e empenho depositados por ela durante realização deste trabalho, me auxiliando a atingir meus objetivos ao final da disciplina.

Agradeço aos membros da banca avaliadora, composta pelo Prof. Dr. Carlos Roberto Valêncio e pelo Prof. Dr. Leandro Alves Neves, pelas críticas e elogios que certamente irão enriquecer este trabalho.

Por fim, agradeço aos amigos que fiz durante a graduação, especialmente João Otávio Calis, Guilherme Gervaes, Marco Guebarra e Samuel Lopes, os quais me auxiliaram em momentos importantes da graduação e viveram tudo isso comigo. Sem vocês, não conseguiria.

Resumo

Diante das mudanças de requisitos que ocorrem frequentemente no cenário atual e com a necessidade de uma resposta rápida e ágil a essas mudanças, surgiram as metodologias ágeis. A mais usada atualmente é o Scrum que tem como um dos seus princípios a aceitação de mudanças durante o desenvolvimento. Junto a essas metodologias também surgiram novas abordagens de desenvolvimento baseadas em testes para assegurar maior qualidade ao produto e aumentar a produtividade. Entretanto, nota-se que existem poucas ferramentas que auxiliam no processo de comunicação entre as equipes envolvidas no projeto, com o intuito de gerar uma documentação que contribua com a qualidade dos testes, fato que pode gerar falhas e consequentemente o aumento de custos. Diante disso, o presente trabalho apresenta uma abordagem para auxiliar e melhorar a comunicação entre as equipes do projeto referente a documentação dos testes em um ambiente ágil. Para tanto, foi desenvolvido um protótipo de software denominado *HelpTest*, o qual foi avaliado por 19 pessoas, sendo cinco alunos da graduação e 14 profissionais da área de Tecnologia da Informação. Com os resultados obtidos, evidencia-se que a abordagem apresentada auxilia e facilita a criação de uma documentação de testes de qualidade em projetos de software, além de que as funcionalidades presentes no protótipo facilitam a modificação de tal documentação no decorrer do projeto, em consonância ao contexto ágil. Assim, este trabalho contribui para o avanço científico por meio de uma solução que visa a qualidade do processo de desenvolvimento de software.

Palavras-chave: Palavras-chave: Documentação de testes. Abordagem voltada a testes. Metodologia ágil.

Abstract

Given the frequent requirements changes in the current scenario and the need for a fast and agile response, agile methodologies have emerged. Scrum is the most used agile methodology, and one of its principles is the acceptance of changes during development. Along with these methodologies come to light new test-based development approaches to ensure higher product quality and increase productivity. However, few tools help in the communication among the project teams to generate documentation that contributes to the quality of the tests, a fact that can result in failures and consequently increased costs. Therefore, this paper proposes an approach to help and enrich the communication among project teams regarding test documentation in an agile environment. So, it was developed a software prototype called HelpTest. It was evaluated by 19 people, including five undergraduate students and 14 Information Technology professionals. With the results obtained, it is evident that the approach presented helps and facilitates the creation of documentation of quality tests in software projects, and that the functionalities present in the prototype facilitate the changes in such documentation during the project, in line with the agile context. Thus, this work contributes to scientific advance through a solution that aims at the quality of the software development process.

Keywords: Agile methodology. Test documentation. Test driven approach.

Sumário

Lista de Figuras **xi**

Lista de Tabelas **xiii**

Lista de Siglas e Abreviações **xiv**

1 Introdução **1**

1.1 Justificativa e Motivação 4

1.2 Objetivos 6

1.3 Metodologia 6

1.4 Organização da Monografia 7

2 Revisão Bibliográfica **9**

2.1 Metodologia Ágeis 9

2.2 Scrum 11

2.3 Abordagem de Teste 14

2.4 Testes de Software e Documentação 17

2.5 Trabalhos Relacionados 19

3 Características do Protótipo **23**

3.1 Arquitetura 23

3.2 Criação do *HelpTest* 25

3.2.1 Identidade visual 25

3.2.2 Acesso ao sistema 26
3.2.3 Perfil Administrador 28
3.2.4 Perfil Desenvolvedor 32
3.2.5 Perfil Testador 34
3.2.6 Perfil Documentador 36
3.2.7 Opção Visualizar projeto 37

4 Resultados e Discussões **39**
4.1 Metodologia de avaliação e perfil dos participantes 39
4.2 Análise dos resultados 42

5 Conclusão **48**
5.1 Contribuições 48
5.2 Trabalhos Futuros 49

Referências Bibliográficas **51**

Apêndice A Documento instrucional para avaliação do *HelpTest* **55**
A.1 Introdução 55
A.2 Conhecendo o *HelpTest* 56
A.2.1 Criação do projeto pelo Administrador 56
A.2.2 Criação de narrativas pelo Desenvolvedor 57
A.2.3 Adição dos dados referente ao teste pelo Testador 57
A.2.4 Criação da documentação referente ao teste pelo Documentador 58
A.2.5 Aba “Visualizar projeto” 58
A.3 Responder o questionário 59

Lista de Figuras

2.1 Componentes do Scrum 12

2.2 *Framework* Scrum 14

2.3 Processo do BDD 16

3.1 Arquitetura do *HelpTest* 24

3.2 Nome e logomarca do protótipo 26

3.3 Ilustração da tela de login 26

3.4 Ilustração da tela de cadastro do usuário 27

3.5 Ilustração da tela de perfis 28

3.6 Ilustração da tela perfil administrador 28

3.7 Ilustração da tela novo projeto 30

3.8 Ilustração da tela perfil administrador com o projeto 31

3.9 Ilustração da tela visualizar projeto 32

3.10 Ilustração da tela criação narrativa e seus cenários 33

3.11 Ilustração da tela perfil desenvolvedor 34

3.12 Ilustração da tela criação documentos referentes ao teste 35

3.13 Ilustração da tela de criação da documentação referente ao teste . . . 37

3.14 Ilustração da tela visualizar projeto 38

4.1 Função exercida pelos avaliadores 40

4.2 Função exercida pelos profissionais de TI 41

4.3 Respostas dos avaliadores em relação a A1 42
4.4 Respostas dos avaliadores em relação a A2 43
4.5 Respostas dos avaliadores em relação a A3 44
4.6 Respostas dos avaliadores em relação a A4 44
4.7 Respostas dos avaliadores em relação a A5 45
4.8 Respostas dos avaliadores em relação a A6 46
4.9 Respostas dos avaliadores em relação a A7 46

Lista de Tabelas

2.1 Requisitos baseados no BDD 15

Lista de Siglas e Abreviações

XP: *eXtreme Programming*

TDD: *Test Driven Development*

BDD: *Behavior Driven Development*

IEEE: *Institute of Electrical and Electronics Engineers*

DSDM: *Dynamic System Development*

FDD: *Feature Driven Development*

XML: *eXtensible Markup Language*

Capítulo 1

Introdução

O cenário global passa por constantes mudanças, como demandas de mercados e ambiente de trabalho, além de novas regulamentações governamentais, o que influencia os requisitos almejados para os projetos de software. Mesmo com as mudanças frequentes, a necessidade de um processo de desenvolvimento e entrega rápidos tornou-se cada vez maior. Com isso, as empresas trabalham em resposta às oportunidades novas, condições que mudam continuamente (NURMULIANI; ZOWGHI; POWELL, 2004; AKBAR et al., 2018).

Dada a necessidade de se adequar a mudanças constantes, os processos de desenvolvimento de software baseados nas metodologias tradicionais de Engenharia de Software, como o modelo Cascata, já não conseguiam suprir este cenário (SOMMERVILLE, 2003). Assim, como alternativa ao uso de metodologias tradicionais, surgiram as metodologias ágeis que visam atender à crescente pressão do mercado por processos mais ágeis e leves, com ciclos de desenvolvimento mais curtos (ABRAHAMSSON, 2003).

Nas metodologias ágeis a premissa é de que sempre haverá mudanças no projeto que não podem ser previstas em seu início, e essas mudanças são bem-vindas, com todos os envolvidos no projeto preparados para elas (NUNES, 2016). No geral, são propostos diferentes processos ágeis baseados no desenvolvimento de software, onde

todos seguem um grupo de princípios, filosofias e valores definidos formalmente em 2001, por meio da aliança ágil e o estabelecimento do Manifesto Ágil (AGILE, 2021). Dentre os métodos ágeis existentes, destacam-se o *eXtreme Programming* (XP) e o Scrum.

O XP é uma metodologia de desenvolvimento de software que combina rapidez, produtividade, qualidade de forma simples e que atende as necessidades do cliente, voltada para o desenvolvimento onde os requisitos se modificam constantemente. O XP busca gerar o máximo de valor possível para o cliente num curto espaço de tempo, sendo que durante esse espaço de tempo o cliente pode analisar o produto recebido (PONTES; ARTHAUD, 2018). Conforme Beck (2004), seu criador, *eXtreme Programming* é uma metodologia ágil para equipes médias e pequenas, onde os requisitos para o desenvolvimento de software são vagos e em constantes mudanças. Um dos principais pilares deste método é o *Test Driven Development* (TDD), no qual testes unitários automatizados são escritos de forma incremental antes mesmo do código de produção ser desenvolvido (BECK, 2010). Busca-se, com isso, assegurar maior qualidade ao produto e aumentar a produtividade, por meio da criação de uma rede de segurança para mudanças futuras e do incentivo à refatoração do código, o que resulta em um círculo virtuoso de qualidade. Assim, os testes são criados com base em um cenário antes mesmo do software ser desenvolvido (AKBAR et al., 2019).

O Scrum, por sua vez, pode ser definido como um *framework* dedicado ao desenvolvimento e gerenciamento de produtos complexos de forma iterativa e incremental, com o intuito de maximizar os resultados e aperfeiçoar a previsibilidade ao longo do desenvolvimento (SCHWABER; SUTHERLAND, 2017). Por ser baseado nos princípios ágeis, é prescrito testes de unidade para garantir que o código em fase de desenvolvimento esteja conforme os requisitos estabelecidos (COLLINS; LOBÃO; DE LUCENA, 2011). Na busca por uma maior integração das atividades de testes e o gerenciamento de produtos são utilizadas diversas técnicas, com destaque para o TDD

e sua evolução: o *Behavior Driven Development* (BDD) (DE CARVALHO; MARQUES, 2019).

Enquanto o TDD se concentra no aspecto de desenvolvimento do sistema com enfoque ao desenvolvedor, o BDD foca no aspecto comportamental, com um enfoque maior ao cliente. Para tanto, o processo do BDD busca se tornar o mais próximo de uma linguagem simples para facilitar o entendimento dos colaboradores não técnicos (MOE, 2019).

Cabe ressaltar que os procedimentos de testes são indispensáveis no processo de desenvolvimento de software. Logo, é necessário que sejam adotados métodos, técnicas e ferramentas que permitam a realização da atividade de teste de maneira sistematizada e com fundamentação cientifica sólida, de modo a aumentar a produtividade e a qualidade e a diminuir custos. Tal questão está relacionada à qualidade de software. Em particular, a indústria tem despertado cada vez mais para a importância da atividade de teste que, por um lado, pode contribuir para a melhoria da qualidade de um determinado produto e, por outro, pode representar um custo significativo dentro dos orçamentos empresariais (DELAMARO; JINO; MALDONADO, 2013).

Juntamente com a realização dos testes, é necessário o uso de ferramentas para o gerenciamento das atividades realizadas durante o processo de desenvolvimento do software e também uma boa documentação, a qual deve facilitar o entendimento tanto do contexto como das regras a serem implementadas na solução. Além disso, tal documentação deve auxiliar no projeto da funcionalidade para tornar a solução almejada mais palpável para seus idealizadores, o que possibilita uma visão mais clara sobre sua operação e usabilidade. Salienta-se que a importância da documentação também se aplica para os artefatos de testes, com o intuito de deixar de forma mais clara os procedimentos realizados e facilitar o entendimento de tal processo (CRESPO et al., 2004).

Com base no contexto ágil, conforme formalizado pelo Manifesto Ágil, o software

em funcionamento deve ter mais prioridade do que uma documentação abrangente. Porém, ressalta-se que isso, de forma alguma, tira a importância da documentação de um projeto, uma vez que tal documentação é fundamental para a comunicação do cliente com a equipe de desenvolvimento e vice-versa, bem como para a comunicação interna da equipe (RUSSO; DA SILVA; LARIEIRA, 2021).

1.1 Justificativa e Motivação

O surgimento e a difusão dos métodos ágeis foram impulsionados pelo acúmulo de documentação e retrabalhos em projetos de empresas de pequeno e médio porte, gerados pela falta de dinamismo dos processos existentes (PRIKLADINICKI; WILLI; MILANI, 2014). Assim, tornou-se necessário uma reavaliação do rigor adotado e a apresentação de novas abordagens, que culminou no Manifesto Ágil.

Em um cenário global, a qualidade de software tem sido um fator de diferenciação no mercado (ANDRADE, 2015). As empresas de software, para se manterem no mercado de desenvolvimento, devem possuir uma metodologia para realizar testes em seu produto para garantir que o mesmo, além de atender os requisitos e o cronograma estipulados, tenha qualidade (BARTIÉ, 2002). Com o aumento da produção de software, os clientes se tornaram mais exigentes, o que obriga os produtores de software a refinarem cada vez mais o produto final para continuarem a competir no mercado (ANDRADE, 2015).

Conforme o estudo sistemático apresentado no trabalho de Wagennar et al., (2018), o qual analisa as escolhas dos artefatos nas metodologias ágeis em diversas empresas de software, observa-se que as equipes de desenvolvimento consideram a documentação voltada para qualidade de software, como por exemplo testes, mesmo que de forma simplória, essencial durante o processo de produção do software, uma vez que influencia na comunicação interna da própria equipe.

Referente a documentação de testes de software, tem-se a norma *Institute of Elec-*

trical and Electronics Engineers - 829 (IEEE-829, 1998), um padrão para documentação de teste de software que relaciona e descreve uma série de documentos e formatos para a realização de oito tarefas para a equipe de teste. Tais tarefas são relacionadas a especificações, relatórios de testes e planejamento, definidas por:

- Plano de testes - escopo dos testes a serem realizados, bem como a definição de ferramentas e cronogramas;
- Especificação do projeto de teste - detalhes da abordagem de teste são definidos, bem como as características a serem testadas;
- Especificação dos casos de testes - define os itens a serem testados, além das premissas e restrições onde ocorrerá os testes;
- Especificação de procedimento de teste - o passo a passo para a execução dos casos de testes são definidos;
- Relatório status de teste - identifica cada caso de teste, assim como os resultados obtidos e o responsável pela execução;
- Log de teste - registra informações e detalhes sobre a execução dos testes;
- Relatório de incidente de testes - os eventos que ocorreram durante a execução dos testes são registrados;
- Relatório sumário de testes - condensa todos resultados obtidos durante o teste e determinadas avaliações sobre este processo.

Observa-se que a norma apenas destaca tais tarefas e suas importâncias, sem recomendar diretamente quais documentos devem sem produzidos. Logo, tal escolha fica a critério da própria equipe de teste (BLANCO, 2012).

Diante do exposto, dado o fato de que a documentação dos métodos de qualidade, como os testes, é essencial (WAGENNAR et al., 2018; BRUNELI, 2006), o estudo

sobre documentação de artefatos de teste no contexto ágil para identificar carências e saná-las, pode contribuir para uma melhor gerência e facilidade de comunicação interna entre os envolvidos no processo de construção de um software, o que caracteriza a motivação do presente trabalho.

1.2 Objetivos

O objetivo principal deste trabalho é contribuir para a qualidade de software, por meio de uma documentação efetiva das atividades de testes, capaz de possibilitar o seu gerenciamento, sem comprometer a agilidade almejada nas metodologias ágeis. Detalhadamente, os objetivos são:

- Identificar os documentos de teste capazes de apoiar a abordagem BDD e Scrum, com base na norma IEEE-829;
- Definir templates para os documentos de teste identificados, de forma a considerar as informações que efetivamente contribuam para a documentação das atividades almejadas sem comprometer a agilidade de sua execução;
- Criar um protótipo de software de apoio às atividades de teste no contexto da metodologia ágil Scrum, de forma a contribuir para o gerenciamento das atividades de teste e, consequentemente, para a qualidade dos resultados obtidos, com base nos templates definidos.

1.3 Metodologia

O processo metodológico adotado no presente trabalho foi dividido em cinco etapas principais.

Na etapa 1, tem-se um levantamento e estudo de materiais relacionados ao trabalho, tais como: contexto histórico sobre o surgimento das metodologias ágeis e seu

uso; a importância da qualidade dos produtos de software para as empresas atuais; a abordagem BDD e Scrum; a documentação dos processos de testes, com base na norma IEEE-829; e os resultados apresentados em trabalhos disponíveis na literatura da área, com o intuito de identificar carências a serem supridas em relação ao estado da arte.

Na etapa 2, a partir dos conceitos estudados, tem-se a seleção de documentos de teste, baseado na norma IEEE-829, e a identificação de informações imprescindíveis para tais documentos, conforme a abordagem BDD, para o estabelecimento de templates para documentação dos testes e, portanto, a maximização da eficiência desta documentação.

Na etapa 3, tem-se a modelagem da arquitetura do protótipo de software capaz de automatizar a documentação de testes e, assim, tornar ágil sua definição e gerenciamento. Para tanto, considera-se o estabelecimento de um modelo que possa ser escalável e, portanto, ser utilizado em trabalhos futuros.

Na etapa 4, tem-se a implementação do protótipo de software, com base na arquitetura estabelecida, por meio do uso de recursos computacionais gratuitos.

Por fim, tem-se a etapa 5 com a avaliação do protótipo por profissionais convidados quanto a usabilidade e a adequação das funções implementadas, com o intuito de identificar contribuições, bem como futuras melhorias.

1.4 Organização da Monografia

Além deste capítulo introdutório, a presente monografia é composta por mais quatro capítulos.

No Capítulo 2 são apresentados os conceitos relevantes para a monografia, por meio um levantamento bibliográfico, capaz de fornecer um embasamento teórico efetivo e sólido que sirva como base para a compreensão do trabalho. Para tanto, a metodologia Scrum e a abordagem BDD são descritas, bem como trabalhos relacionados a

documentação de testes de software, com o intuito de evidenciar a sua importância e fornecer comparações entre os modelos tradicionais e modelos ágeis.

No Capítulo 3, por sua vez, mostra-se a arquitetura estabelecida para o projeto, tecnologias utilizadas para a implementação. Além de apresentar a descrição das funcionalidades implementadas.

No Capítulo 4 apresenta-se o perfil dos avaliadores do protótipo, assim como o processo utilizado para avaliação, as tarefas analisadas, além dos resultados obtidos seguido pela análise dos mesmos.

Por fim, no Capítulo 5, discute-se os resultados obtidos e suas contribuições juntamente com propostas de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Neste capítulo são apresentados os conceitos teóricos que serviram como base para o desenvolvimento do trabalho, bem como uma discussão sobre trabalhos relacionados presentes na literatura. Para tanto, na seção 2.1, apresenta-se o conceito de metodologias ágeis e sua importância e na seção 2.2, aborda-se especificamente o Scrum. Na seção 2.3, detalha-se a abordagem BDD e na seção 2.4 são abordados conceitos sobre testes e documentação. Por fim, na seção 2.5 são apresentadas ferramentas relacionadas ao trabalho em andamento.

2.1 Metodologia Ágeis

O desenvolvimento de software surgiu como uma incógnita. Não existia um plano a ser seguido, decisões eram tomadas com base em cronogramas feitos a partir de suposições. Com o crescimento dos sistemas e o aumento da complexidade, tal prática deixou de funcionar, pois acarretava diversos problemas durante o processo de desenvolvimento (PONTES; ARTHAUD, 2018).

Em busca da solução para este problema foram propostos modelos prescritivos com intuito de trazer ordem ao cenário de caos. Um modelo prescritivo é determinado por um conjunto de elementos que inclui ações de engenharia de software, com a indicação

de produtos de trabalho e de mecanismos que garantam a qualidade do software e estabeleçam um controle sobre as modificações necessárias para o desenvolvimento do sistema (PRESSMAN, 2010).

Tais modelos ficaram conhecidos como tradicionais, como por exemplo o Modelo Cascata e o Modelo Espiral. Entretanto, o trabalho referente ao processo de desenvolvimento, junto com levantamento de requisitos e sua documentação, tornava-se muito extenso e burocrático. Tal fato fez com que alguns dos praticantes ficassem frustrados, pois uma possível mudança nos requisitos, por exemplo, acarretava uma grande sequência de passos longos que consumiam muito tempo do projeto (SOARES, 2004).

Em meio a esta insatisfação, surgiram as metodologias ágeis focadas em melhorar a abordagem do desenvolvimento de software por meio da colaboração com o cliente, de respostas rápidas a mudanças, do funcionamento do software e do foco nos indivíduos e interações entre eles. A entrega da solução ao cliente é fracionada e se dá por meio de ciclos de desenvolvimento menores, o que torna possível a mudança de requisitos e também uma maior agilidade no processo de replanejamento, caso necessário (CARVALHO; ABRANTES; CAMEIRA, 2011).

Em 2001 foram definidos princípios básicos que serviram como base para o surgimento de diferentes metodologias ágeis, as quais adequavam-se de diferentes modos a tais princípios, definidos conforme o Manifesto Ágil. Os princípios são (KENT et al., 2001):

- Indivíduos e interações mais que processos e ferramentas
- Software em funcionamento mais que documentação abrangente
- Colaboração com o cliente mais que negociação de contratos
- Responder a mudanças mais que seguir um plano.

A maior diferença entre as metodologias ágeis e tradicionais é que as metodologias ágeis são adaptativas ao invés de prescritivas e são orientadas a pessoas e não

a processos (FERREIRA; LIMA, 2006). Metodologias tradicionais tentam prever os detalhes do processo de desenvolvimento na parte inicial do projeto, o que acarreta alguns problemas ao surgir a necessidade de realizar mudanças no projeto, como atrasos na entrega. Enquanto nas metodologias ágeis a premissa é de que sempre haverá mudanças no projeto que não podem ser previstas na sua fase inicial de desenvolvimento, e essas mudanças são sempre bem-vindas, onde os envolvidos no projeto estão preparados para elas (FOWLER, 2005).

Existem diversos tipos de metodologias ágeis presentes no mercado como *eXtreme Programming* (XP), *Dynamic System Development Model* (DSDM), *Feature Driven Development* (FDD), Scrum. Dentre estas, a mais presente e utilizada no mercado atualmente é o Scrum (RUSSO; DA SILVA; LARIEIRA, 2021; FELIPE, 2021; ALQUDAH; RAZALI, 2016).

2.2 Scrum

O Scrum é uma estrutura de desenvolvimento de software ágil incremental e iterativa para gerenciar produtos. Foi definido pela primeira vez em 1986 como “uma estratégia de desenvolvimento de produto flexível e holística, onde uma equipe de desenvolvimento funciona como uma unidade para alcançar um objetivo comum” (TAKEUCHI; NONAKA, 1986). Suas características incluem entregáveis, cronograma flexível, times pequenos, revisões frequentes e colaboração (SCHWABER,1997). No Scrum existem três componentes: Funções, Artefatos e Cerimônias, conforme listados na Figura 2.1.

Figura 2.1: Componentes do Scrum

Fonte: Adaptado de Schdeva (2016)

Da análise da Figura 2.1, observa-se que, com relação às funções no Scrum, existem três: *Product Owner*, responsável por criar os requisitos e pelo financiamento do projeto (HRON; OBWEGESER, 2018); *Scrum Master*, cuja responsabilidade é gerir e verificar se os valores, práticas e regras do Scrum estão em conformidade com o esperado; e Time, responsável pelo desenvolvimento dos requisitos estabelecidos pelo *Product Owner* (SCHDEVA, 2016).

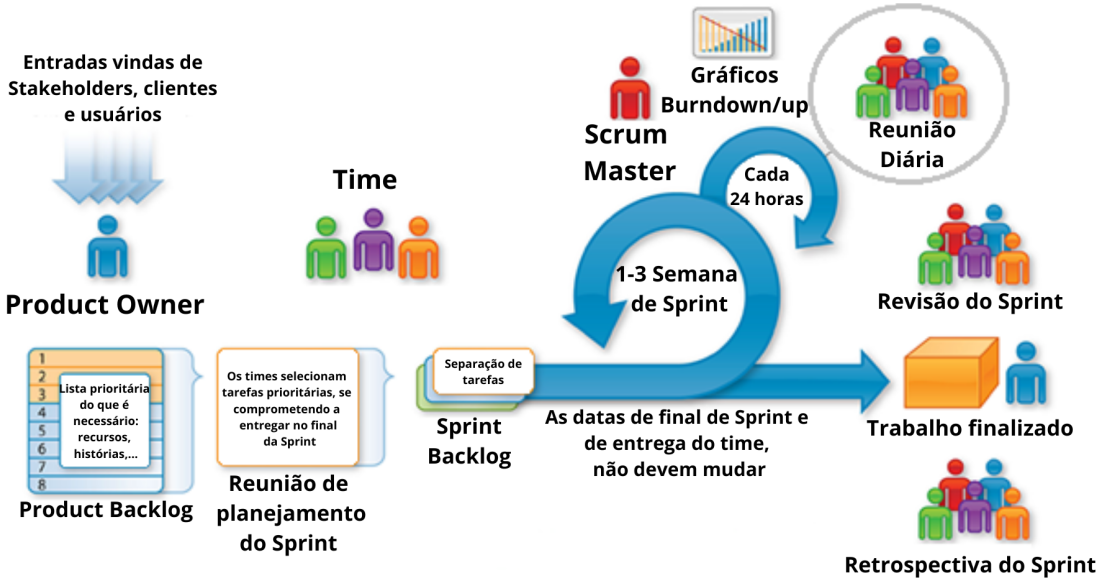
Referente aos artefatos, tem-se, primeiramente, o *product backlog*: uma lista dos requisitos a serem desenvolvidos que são documentados no formato de história de usuário, a qual contempla todas a funções desejadas para o produto, onde cada item tem uma descrição, prioridade e uma estimativa de esforço para sua conclusão, gerenciado pelo *Product Owner* e adaptável a mudanças de acordo com as necessidades e valor de negócio. O plano de lançamento descreve os principais riscos e características gerais do produto, bem como um objetivo para lançamento junto com uma provável data de entrega e custo associados, contempla-se os itens de maior prioridade no *product backlog* e, assim, reflete as funções que a versão conterá (SCHDEVA, 2016). A *sprint*, no *framework* Scrum, é um ciclo de desenvolvimento, limitado por um período de tempo, onde uma versão incremental e usável do produto é desenvolvida e entregue ao *Product Owner*. A *sprint backlog* consiste em uma lista de tarefas a serem feitas durante um *sprint* e que são derivadas de histórias presentes no *product backlog*.

Tais histórias são divididas em tarefas que são atribuídas pelo time que se compromete a terminá-las no tempo estimado. Já os gráficos *burn-down* são artefatos visuais, como quadro de tarefas e gráficos de finalização de tarefas, visíveis tanto para a equipe quanto aos demais frequentadores do ambiente de desenvolvimento, que atuam como um elemento motivador. A ideia é criar um espírito de realização e dever cumprido para a equipe (PAULK; DAVIS; MACCHERONE, 2011).

Por fim, as cerimônias: planejamento do sprint, revisão do *sprint*, retrospectiva do *sprint* e a reunião diária. O planejamento do *sprint* é o planejamento da iteração para determinar o que será feito no *sprint* e como a equipe vai construir uma parte do produto durante o *sprint*. A equipe se compromete que tem a capacidade para a realização das tarefas e assim concluir a meta, tal processo é acompanhado e facilitado pelo *Scrum Master*. Revisão do *sprint* é uma reunião de revisão realizada ao final de um *sprint*, na qual o time apresenta ao *Product Owner* e demais interessados a funcionalidade incluída no produto em busca de um *feedback*. A equipe também discute junto ao *Scrum Master* o trabalho realizado e as entradas para o próximo *sprint*. Retrospectiva do *Sprint* é uma reunião onde a equipe e o *scrum master* discutem o que deu certo, o que precisa ser melhorado e o que tem de errado, com o intuito de gerar melhorias para o próximo sprint. A reunião diária é realizada junto ao *Scrum Master* em que os membros da equipe, cada um por vez, explicam o que foi feito desde a última reunião, o que será feito antes da próxima e os obstáculos que encontraram no caminho. Nesta reunião ocorre também a atualização do progresso de cada tarefa com o uso de ferramentas como os gráficos *burn-down* (AULK; DAVIS; MACCHERONE, 2011).

Todos os relacionamentos entre os componentes do Scrum estão presentes na Figura 2.2.

Figura 2.2: *Framework* Scrum



Fonte: Adaptado de Mandryk (2016)

A partir da Figura 2.2, verifica-se que o *Product Owner* elabora o *Product Backlog*, a partir das necessidades dos *stakeholders*, clientes e usuários. Após tal processo, realiza-se uma reunião com o Time sobre o planejamento da *sprint* e a definição do *sprint backlog*. Com isso, o *sprint* tem início com um ciclo em média de 1-3 semanas e com reuniões diárias, apoiadas sempre pelo *Scrum Master* e artefatos como gráficos *burn-down/up*. Ao final do *sprint* é entregue o trabalho finalizado, seguido pelas cerimônias de revisão do *sprint* e sua retrospectiva.

2.3 Abordagem de Teste

Com o objetivo de maximizar a eficiência dos processos de validação e verificação de software, foram propostas diversas estratégias e metodologias. Entre essas estratégias, tem-se o *Test-Driven Development* (TDD) e o *Behavior Driven Development* (BDD).

O BDD surgiu baseado na abordagem TDD, trata-se de uma evolução de um pro-

cesso voltado para o desenvolvimento para um processo que inclui, além dos programadores e testadores, todas as partes interessadas importantes, técnicas ou não (SOLIS, WANG, 2011).

Assim, o objetivo do BDD é fornecer um processo sistemático que suporte um vocabulário comum e entendimento compartilhado sobre os requisitos entre clientes, engenheiros de requisitos, desenvolvedores e testadores (NORTH, 2006).

No BDD, os requisitos são descritos em duas partes: a central, chamada de narrativa ou comportamento; e o cenário, sendo que pode haver vários cenários em um comportamento (NORTH, 2006), conforme exemplificado no Tabela 2.1. A narrativa é um comportamento ou ação que o sistema oferece ao seu usuário, possui uma entrada ("realizar o login") e uma saída ("visualizar os produtos"). Já nos cenários, tem-se os critérios de aceitação que servem para alcançar um resultado especifico da narrativa. Por exemplo: para que a visualização dos produtos seja realizada, é necessário que o usuário seja autentificado com sucesso por meio do nome e senha. Esses comportamentos/narrativas, atuam como um diferencial nos documentos de testes e requisitos, por serem de fácil interpretação, o que facilita a comunicação e promove um melhor entendimento dos requisitos.

Tabela 2.1: Requisitos baseados no BDD

Narrativa/Comportamento
Como Usuário
Quero realizar o *login*
Para que eu possa visualizar os produtos
Cenário 1: Um usuário consegue fazer login no sistema
Dado as informações do usuário, como nome e senha
Quando a autentificação for bem sucedida
Então o usuário faz o *login* no sistema
Cenário 2: Um usuário pode visualizar os produtos
Dado o usuário autentificado com as credenciais corretas
Quando o usuário faz *login* no sistema
Então o usuário pode ver os produtos

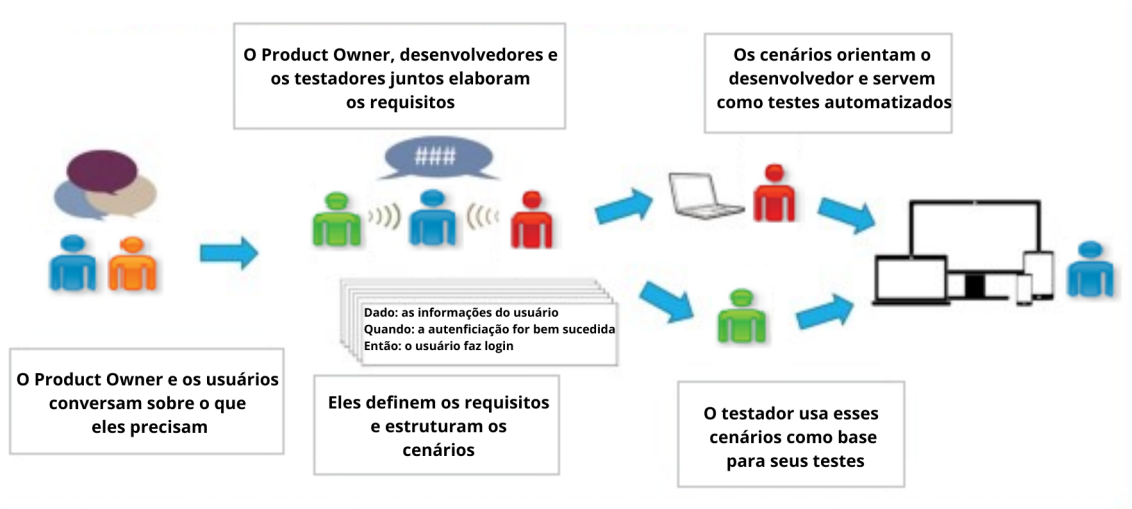
Fonte: Adaptada de North (2006)

O BDD possui seis características (SOLIS; WANG, 2011), a saber:

- O uso de uma linguagem comum, fácil de ser interpretada e compreendida pelos clientes finais;
- Um processo que decomponha de forma iterativa as especificações de alto nível do projeto;
- Modelos para escrever histórias de usuários, comportamentos e cenários;
- Testes automatizados de aceitação;
- Código limpo e legível;
- Elaboração de comportamentos e cenários com base nas necessidades da fase de desenvolvimento.

Assim, a abordagem BDD é capaz de descrever comportamentos complexos de sistemas de software, com o intuito de permitir que *stakeholders* sem conhecimento técnico consigam participar e até escrever comportamentos, o que minimiza, consideravelmente, o esforço gasto entre essas atividades. Seu ciclo é representado na Figura 2.3.

Figura 2.3: Processo do BDD



Fonte: Adaptada de Antonov (2018)

Da análise da Figura 2.4, observa-se que o processo de BDD tem enfoque na colaboração entre os membros do projeto. Nota-se que inicialmente *Product Owner* junto aos usuários, definem suas necessidades. Na sequência, o *Product Owner* em companhia dos testadores e desenvolvedores criam os requisitos e estruturam cenários. Assim, a criação desses cenários serve como base para o desenvolvimento e a realização de testes do projeto, para que o produto final entregue, cumpra o esperado.

2.4 Testes de Software e Documentação

Para a obtenção de qualidade é fundamental a realização de testes de software. O teste tem como objetivo encontrar defeitos e assim mostrar que o funcionamento de um requisito do software, em uma determinada situação, não está de acordo com o esperado (KOSCIANSKI; SOARES, 2006).

Existem várias abordagens para testar um sistema, pode-se utilizar o código fonte, os cenários definidos nos casos de uso, diagramas de atividades e de componentes, entre outros (ANDRADE, 2015). O plano de testes é o documento mais abrangente do projeto de testes, pois mostra o projeto em toda sua dimensão. Em uma possível manutenção futura de um sistema de software, pode-se ter a necessidade de testá-lo outra vez, sendo que o documento que servirá de orientação para testadores e também para a criação de novos testes será o plano de teste (BARTIÉ, 2002).

A norma IEEE-829 define uma documentação para os projetos de testes de software, o único padrão de documentação conhecido e emitido por uma instituição de credibilidade internacional. É estabelecido o formato de documentações e informações a serem registradas. Isso permite que a norma seja utilizada como um ponto de partida por uma empresa que visa organizar as atividades de validação e verificação (KOSCIANSKI; SOARES, 2006).

A norma tem como finalidade criar uma descrição do escopo, recursos, abordagem, cronograma das atividades e a identificação dos itens, funções a serem testadas, tarefas

a serem executadas, além de vincular as pessoas responsáveis por tais tarefas e os riscos associados ao plano de teste.

Conforme a norma IEEE-829 (1998), um plano de testes completo deve conter:

- Identificador de plano de testes, com um atributo de identificação único de especificação;
- Introdução, com um resumo dos itens de software e recursos a serem testados, sendo indicado colocar a necessidade do item, assim como seu histórico de desenvolvimento;
- Itens de teste, identificando a inclusão dos itens de testes e seus níveis de versão e revisão;
- Recursos que serão testados, identifica as características do software e combinações das características a serem testadas definidas pelo escopo;
- Recursos que não serão testados, onde todas características e combinações que não serão testadas são identificadas;
- Abordagem, descrevendo todas as abordagens a serem testadas;
- Critérios de liberação e critérios de falha;
- Critérios de suspensão e de retomada, especifica o critério utilizado para suspender tudo ou alguma parte das atividades de testes, nos itens associados;
- Entrega de testes, identifica os documentos a serem entregues , sendo eles, plano de teste, especificações de projeto de teste, especificações de casos de teste, especificações de procedimento de teste , relatórios de status de teste, log de teste, relatórios de incidentes de teste, relatórios de sumário de teste;

- Tarefas testadas, identifica um conjuto de tarefas necessárias para preparar e inicializar os testes, analisando todas as dependências entre tais tarefas e a habilidade necessária para realizá-las;
- Necessidades de ambientes, especifica as necessidades e as propriedades desejadas de um ambiente de testes;
- Responsabilidades, separação dos grupos de gerenciamento, *designing* preparação, execução, testemunhas, checagem e resolução dos testes. Esses grupos são compostos por desenvolvedores, representantes de usuários, suporte técnico, equipe de qualidade, entre outros;
- Necessidades de pessoas e treinamentos, identifica os níveis de habilidades e provê treinamentos fornecendo as habilidades necessárias, além de especificar os materiais de testes necessários;
- Cronograma, especifica um cronograma para cada tarefa e os marcos de testes , estimando um tempo necessário para cada um destes itens;
- Riscos e contingências, os riscos do plano de teste são analisados e são criados planos de contingência para cada risco;
- Aprovação, especifica títulos e nomes das pessoas responsáveis que devem aprovar o plano de teste, além de conter espaço para as datas e assinaturas.

2.5 Trabalhos Relacionados

Foram encontradas poucas ferramentas, sejam elas comerciais ou não, que se propõem a documentar testes de software. Por outro lado, existe uma ampla variedade de ferramentas que propõem a auxiliar na construção e organização dos casos de testes. Dentre os trabalhos correlatos ao tema IEEE-829 e de suporte ao gerenciamento de testes, são apresentados a seguir os que tiveram maior relevância.

A documentação de um software é valiosa tanto para seu desenvolvimento quanto para sua evolução, pois é vista como a principal forma de comunicação de informações sobre o software e isto não é diferente ao se tratar de testes (PINTO, 2021).

Os testes de software são umas das atividades mais custosas do processo de desenvolvimento, já que podem envolver uma quantidade significativa dos recursos de um projeto. Assim, manter uma documentação referente a esses testes é de suma importância (PINTO, 2021). Tal documentação influencia positivamente em diversos aspectos no desenvolvimento do projeto. Além de ser uma ótima fonte de informação sobre o processo de testes, o que facilita o processo de evolução e manutenções futuras no software.

A documentação também permite o reuso de histórias e casos de testes, com isso minimiza a possibilidade de retrabalhos e provém aos desenvolvedores acesso a informações que facilitam o entendimento do software (SILVA, 2021).

O apoio ferramental para qualquer atividade do processo de teste é importante como um mecanismo com a finalidade da redução de esforço associado à tarefa que será realizada, seja ela seu planejamento, projeto, execução e documentação dos testes. Logo, encontrar ferramentas que se encaixam na estratégia definida pode reduzir significantemente o esforço de realização de tal tarefa (BORGHI; POVOAS; LEMOS, 2019).

Em 2004, a ferramenta TestCen forneceu suporte ao planejamento de teste funcional por meio de diagramas de caso de uso. As funcionalidades apresentadas pelo projeto auxiliam as equipes de testes no processo de verificação e validação dos requisitos de software. Foi baseada em boa parte nos padrões existentes na norma IEEE829, para facilitar a organização e processo de planejamento, além da execução dos testes (BIANCHINI, 2004). Porém, observa-se que a interface de tal ferramenta possui limitações, é considerada sobrecarregada visualmente, o que pode induzir o usuário ao erro.

TestLink é uma ferramenta que oferece um ambiente que facilita a criação e manutenção dos casos de testes e também a organização em planos de testes. Tais planos permitem aos usuários executar casos de testes, gravar seus resultados, gerar relatórios, traçar e priorizar requisitos de software, além de atribuir tarefas (TESTLINK COMMUNITY, 2010). Logo, trata-se de uma ferramenta mais técnica, a qual é utilizada mais pela equipe especializada de testes, com enfoque em planejar e gerar relatórios de testes automatizados os quais devem ser repassados para a equipe de documentação.

O TaRGeT de 2010 permite que os casos de teste sejam gerados automaticamente a partir de cenários de casos de uso escritos em linguagem natural. Os casos de uso são escritos conforme um esquema *eXtensible Markup Language* (XML), que foi projetado para conter as informações necessárias para gerar o procedimento de teste, descrição e requisitos relacionados (BORBA; SILVA, 2010). Com isso, observa-se que a ferramenta é bem completa na questão de realização de casos de testes, entretanto nela não estão inclusas funcionalidades referentes à documentação de testes. Isso pode dificultar a comunicação entre a equipe de testes e a equipe de documentação.

O Cucumber é uma ferramenta usada para executar testes de aceitação automatizados que foram criados em um formato BDD. Um de seus recursos mais destacados é a capacidade de realizar descrições funcionais de texto, como testes automatizados. O teste do Cucumber não apenas segue os requisitos como seus cenários de teste, mas também ajuda os analistas de negócios ou o gerente de produto a ajustar facilmente os dados de teste (LAWRENCE; RAYNER, 2019). Sua utilização é incentivada por gestores aos desenvolvedores devido a cultura do código limpo, pelo fato de que a construção dos cenários de testes utiliza-se da abordagem do BDD e, assim como no TDD, gera um código mais coeso e limpo. Porém, muitas vezes a utilização dessa ferramenta se dá de forma individual pelo desenvolvedor sem a integração com a equipe de testes, ao contrário do que é proposto no BDD, no qual o centro é a comunicação, colaboração e detalhamento de cenários juntamente com sua automação.

Diante deste cenário, observa-se que tais ferramentas, embora auxiliem no processo de realização de testes e documentação, são independentes. Logo, o uso isolado de cada uma impossibilita uma melhor comunicação entre as equipes participantes do projeto: documentação; testes; desenvolvimento. Isso pode resultar em falha na comunicação e, consequentemente, gerar futuros erros, atrasos e mais gastos.

Logo, a partir do estudo dessas ferramentas e dada a importância do processo de documentação, foi possível analisar a forma como os testes são realizados e refinar o processo de documentação. Como resultado, depreende-se que a integração do processo de documentação de testes, com os relatórios feitos pela equipe de testes e de desenvolvedores, permitirá uma melhoria na comunicação dos envolvidos no projeto. Com isso, será possível rastrear as ações realizadas durante os testes, bem como compartilhar informações sobre o trabalho entre a equipe de testes, desenvolvimento e documentação. Assim, por consequência, fomenta-se a cultura de integração de equipes dentro de uma empresa e promove a obtenção de uma documentação de testes mais sólida.

Capítulo 3

Características do Protótipo

Neste capitulo são apresentados a arquitetura do protótipo *HelpTest* na Seção 3.1 e os detalhes sobre a criação do protótipo na Seção 3.2, com subseções sobre as tecnologias que foram utilizadas, a criação de uma identidade visual e as funções desenvolvidas com a finalidade de suprir a carência de ferramentas que auxiliam na criação de uma documentação voltada aos testes.

3.1 Arquitetura

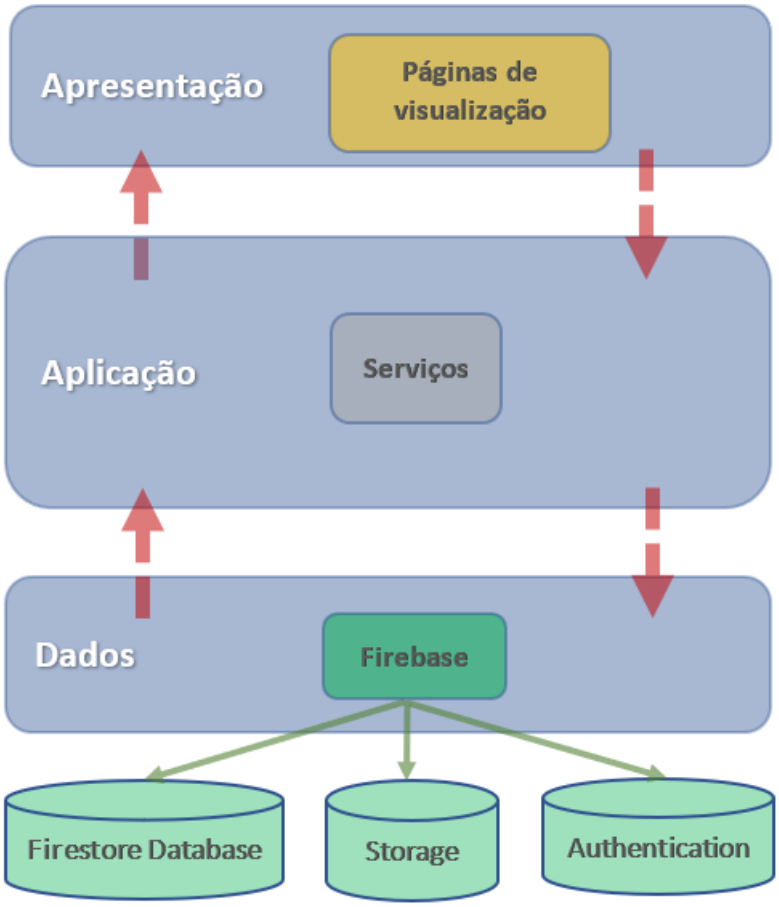
O modelo de arquitetura escolhido para a implementação do protótipo é o Modelo em Camadas, com três camadas: Apresentação, Aplicativo e Dados, conforme ilustrado na Figura 3.1.

A camada de apresentação é a interface do usuário e de sua comunicação com a ferramenta, tal comunicação é realizada por meio das páginas de visualização do sistema. Sua principal finalidade é exibir e coletar informações do usuário. A camada do aplicativo, que também pode ser chamada de camada lógica, é responsável por processar as informações coletadas na camada de apresentação. Para tanto, utiliza-se de um conjunto de diretrizes que definem ou restringem a forma como tais operações devem ser realizadas, por meio dos serviços que realizam as requisições à camada

de dados. Essas diretrizes são importantes para garantir uma visão clara do que deve ser feito, como e por qual razão. Por fim, a camada de dados ou *back-end*, onde as informações processadas pela camada do aplicativo são gerenciadas e armazenadas, conforme as separações internas do Firebase, sendo: o *Firestore Database* armazena os documentos criados, o *Storage* armazena os arquivos que foram feitos upload e o *Autentification* é responsável por gerir o cadastro de novos usuários.

Este modelo foi escolhido devido sua separação lógica e funcional, assim modificações realizadas em uma camada, não interferem diretamente nas outras, o que torna a manutenção e o desenvolvimento mais simples.

Figura 3.1: Arquitetura do *HelpTest*



Fonte: Eladorado pelo autor

3.2 Criação do *HelpTest*

As tecnologias utilizadas para a criação do *HelpTest*, foram baseadas na necessidade atual do mercado. Para a escrita do código, foi utilizada a ferramenta Visual Studio Code, React como um *framework* JavaScript juntamente com Tailwind como um *framework* CSS, para a construção do *front-end*. Isso permite que o *HelpTest* seja responsivo e funcione tanto em uma versão desktop quanto para uma versão mobile.

Para o back-end, a opção escolhida foi o Firebase que fornece serviços de hospedagem NoSQL em tempo real de banco de dados hospedado na nuvem. Provém serviços como: autenticação, suporte a diversos frameworks e linguagens, integrações a outros sistemas, como por exemplo a Jira muito utilizada em ambientes Scrum. Para realizar a comunicação entre o front-end e o back-end foi escolhido o JavaScript e para o controle de versionamento durante o desenvolvimento foi escolhido o *Git*.

Nas subseções a seguir está descrito como foi a definição da identidade visual e também as funções contempladas pelo *HelpTest*.

3.2.1 Identidade visual

Inicialmente foi definida uma paleta de cores adotada na construção do *HelpTest* com o intuito de tornar o *front-end* mais limpo. Também houve a construção da logomarca, ilustrada na Figura 3.2, para demonstrar ao usuário o mapa do Brasil, por ser uma tecnologia brasileira. Assim, cada um dos cinco traços representa os dedos de uma mão, o que remete à expressão “dar uma mão”, ou seja, oferecer uma ajuda. Devido a ferramenta ter enfoque na área de testes, o nome escolhido para associar as duas ideias foi *HelpTest*.

Figura 3.2: Nome e logomarca do protótipo

Fonte: Eladorado pelo autor

3.2.2 Acesso ao sistema

O acesso ao sistema é realizado por meio da tela de login, exemplificada na Figura 3.3, para os quatro tipos de usuários considerados: administrador do projeto, desenvolvedor, testador e documentador. Caso o usuário não tenha uma conta, o seu registro pode ser realizado ao clicar no link "Não tem conta? Cadastre-se".

Figura 3.3: Ilustração da tela de login

Fonte: Eladorado pelo autor

Na tela de cadastro, representada na Figura 3.4, é permitido ao usuário cadastrar-se com seu nome, e-mail e senha. Após realizar o login, o usuário é direcionado para a

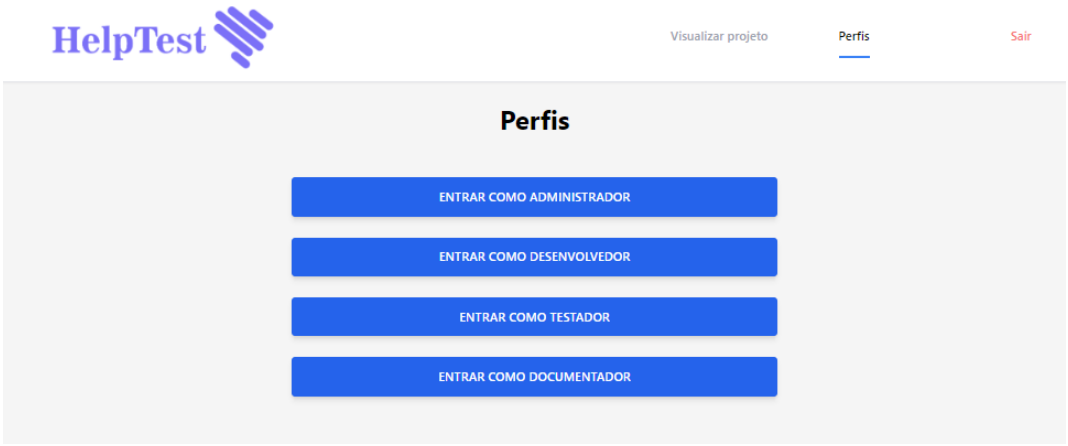
tela inicial do *HelpTest*.

Nesta tela, ilustrada na Figura 3.5, são apresentados os “Perfis” que representam cada um dos quatro tipos de usuários. Devido ao fato de que em empresas menores é comum um usuário ter múltiplas funções, a diferenciação do perfil assumido por um usuário é realizada nesta tela, assim é permitido que um mesmo usuário atue em uma ou mais funções.

Figura 3.4: Ilustração da tela de cadastro do usuário

Fonte: Eladorado pelo autor

Figura 3.5: Ilustração da tela de perfis

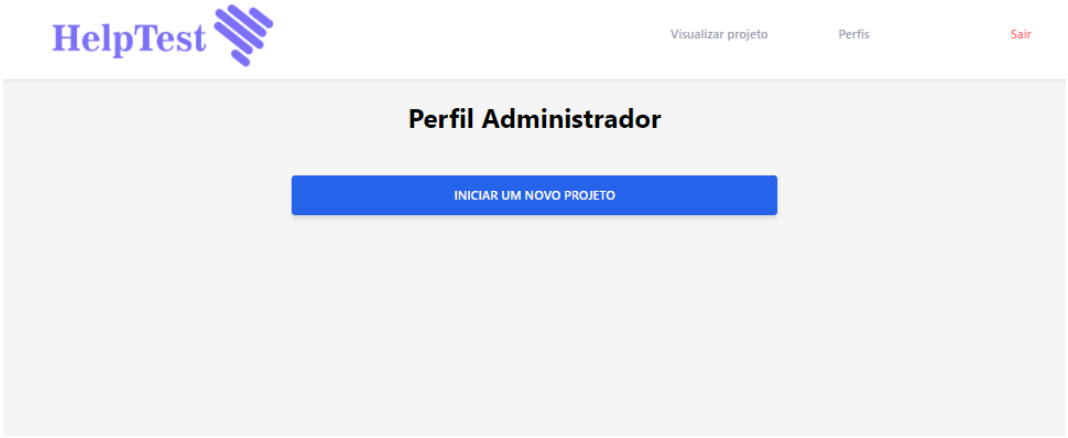


Fonte: Eladorado pelo autor

3.2.3 Perfil Administrador

Ao usuário selecionar “Entrar como Administrador” (Figura 3.5), ele é direcionado para uma tela do perfil administrador, ilustrada na Figura 3.6, onde é possível, por meio do botão “Iniciar um novo projeto”, a criação de um novo projeto.

Figura 3.6: Ilustração da tela perfil administrador



Fonte: Eladorado pelo autor

A partir disso, o usuário deve definir um nome para o projeto, uma breve descrição e selecionar os documentos que irão compor o projeto, além de poder anexar arquivos

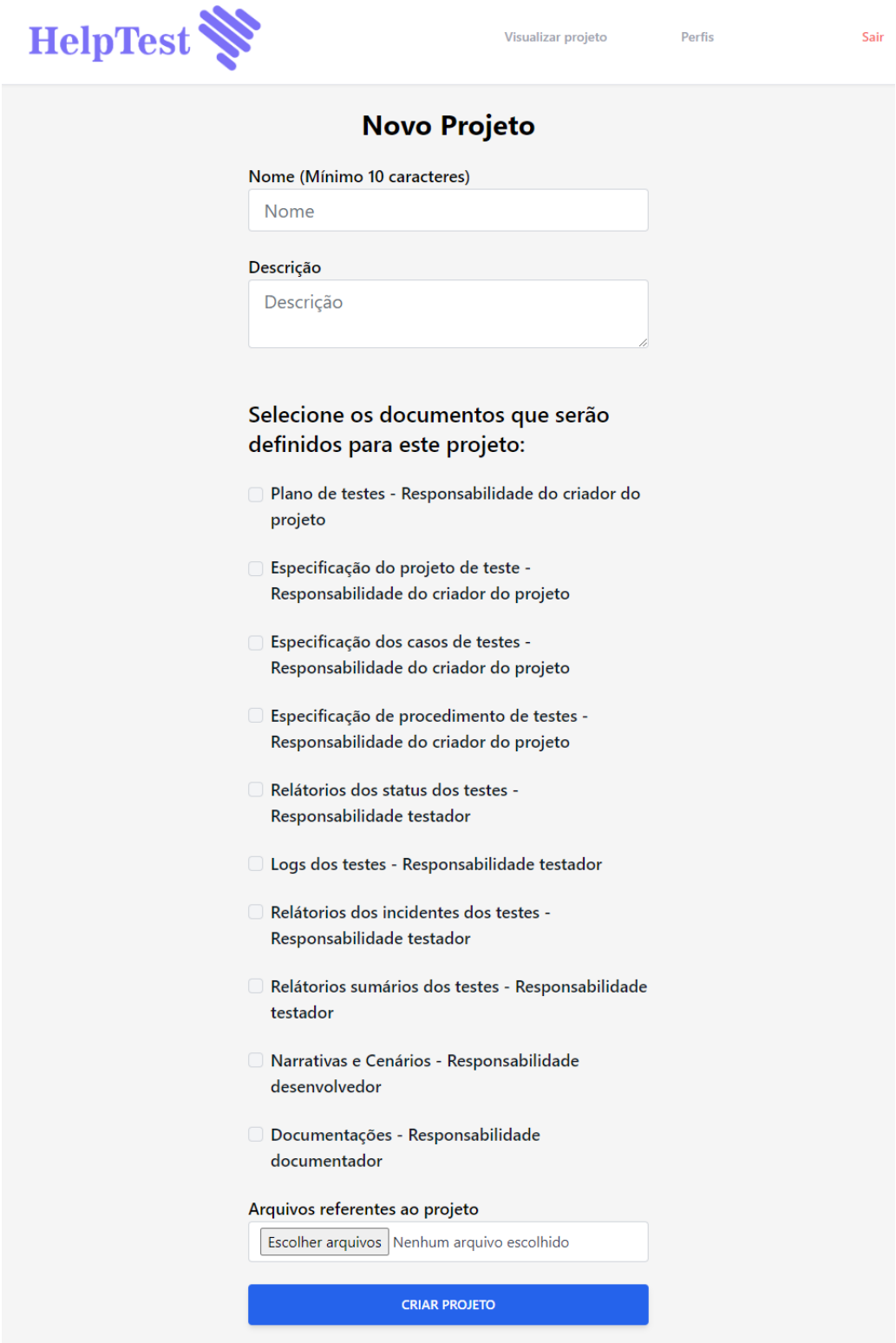
importantes referentes ao projeto, conforme ilustrado na Figura 3.7.

Esses documentos a serem selecionados pelo Administrador do projeto são referentes à norma IEEE-829 e também ao BDD. Observa-se que cada documento é representado pelo seu nome seguido pelo responsável pela sua definição.

Logo, cabe salientar que tais documentos estarão liberados a seus respectivos responsáveis para criação somente se este documento foi selecionado durante o processo de criação do projeto.

Como administrador, tem-se como responsabilidade a criação de quatro documentos: plano de testes, especificação do projeto de teste, especificação dos casos de teste e especificação do procedimento de testes. Tais documentos definem todo o processo de teste, como será testado, qual ferramenta será utilizada, casos de teste e cronogramas. Assim, pode-se afirmar que são documentos extensos e que praticamente não sofrerão mudanças. É esperado que tais documentos estejam prontos e sejam anexados durante o procedimento de criação do projeto.

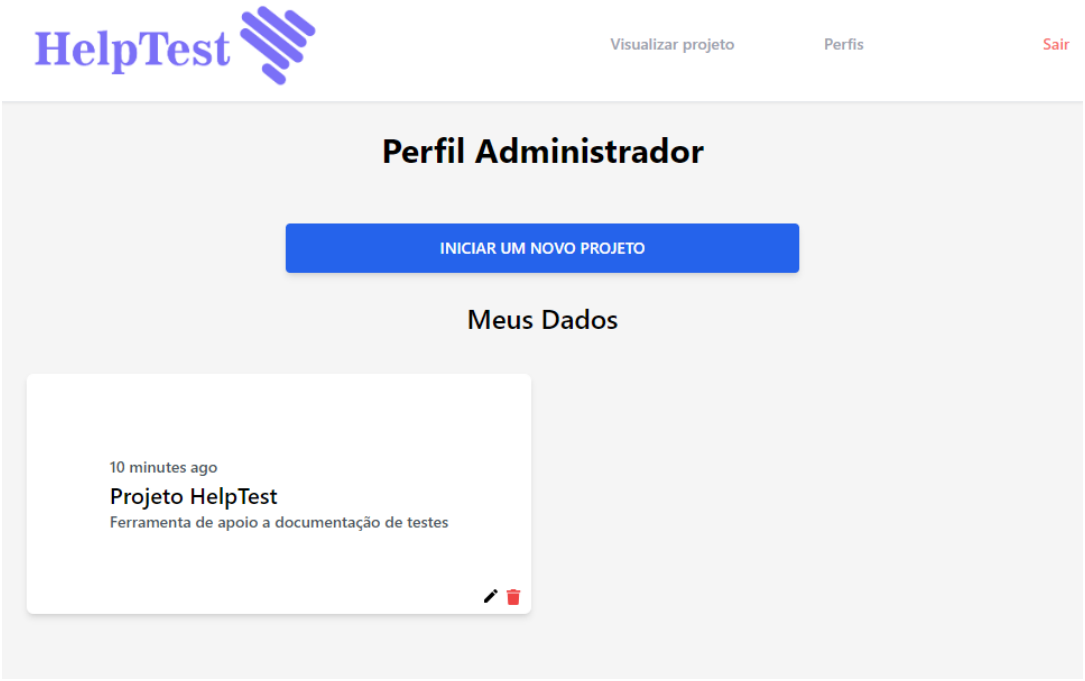
Figura 3.7: Ilustração da tela novo projeto



Fonte: Eladorado pelo autor

Após a criação do projeto, ainda no perfil administrador, o projeto ficará disponível para visualização, inicialmente por meio do seu nome e descrição, além da possibilidade de edição e exclusão, por meio de botões com ícones representativos, o que facilita a usabilidade, conforme mostrado na Figura 3.8.

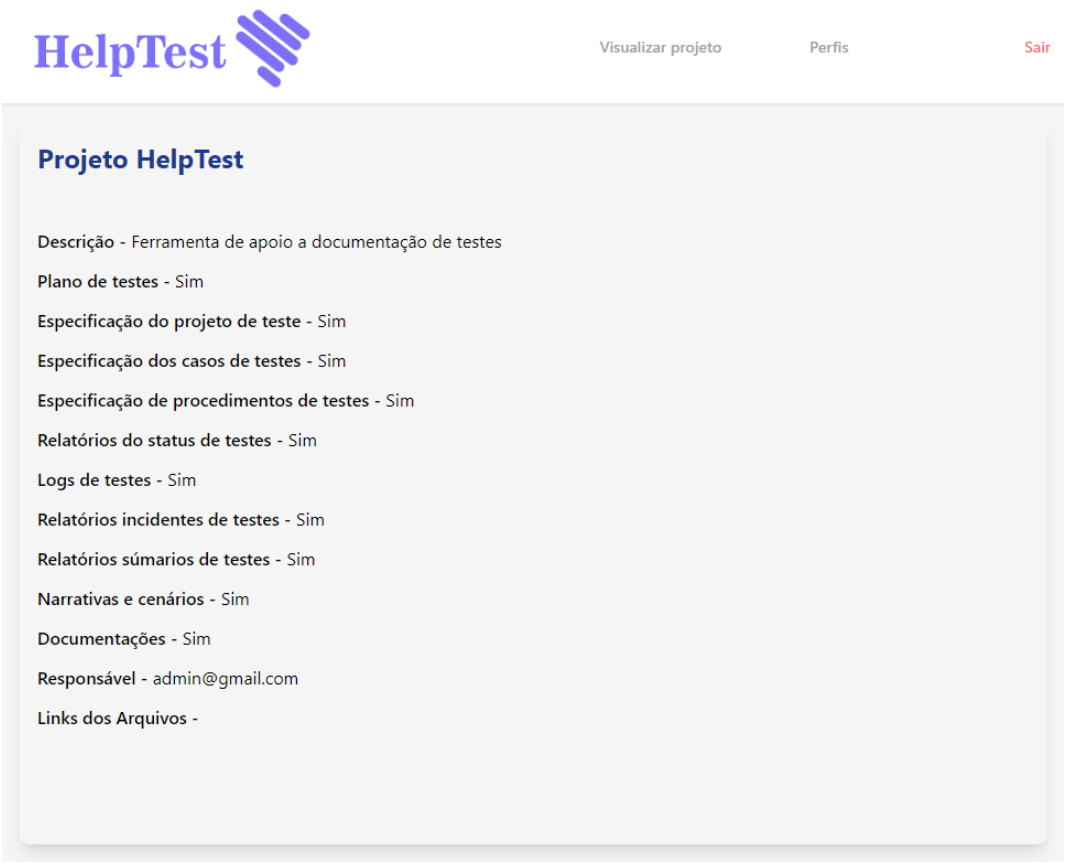
Figura 3.8: Ilustração da tela perfil administrador com o projeto



Fonte: Eladorado pelo autor

Para a visualização dos detalhes do projeto, basta clicar no quadro referente a ele. Assim, o usuário é direcionado para uma página na qual é possível visualizar os documentos selecionados, o e-mail do responsável pela criação daquele projeto, assim garante a rastreabilidade das ações realizadas no sistema e um *link* para a visualização do arquivo que foi anexado, o que possibilita também seu *download*, conforme pode ser visto na Figura 3.9.

Figura 3.9: Ilustração da tela visualizar projeto



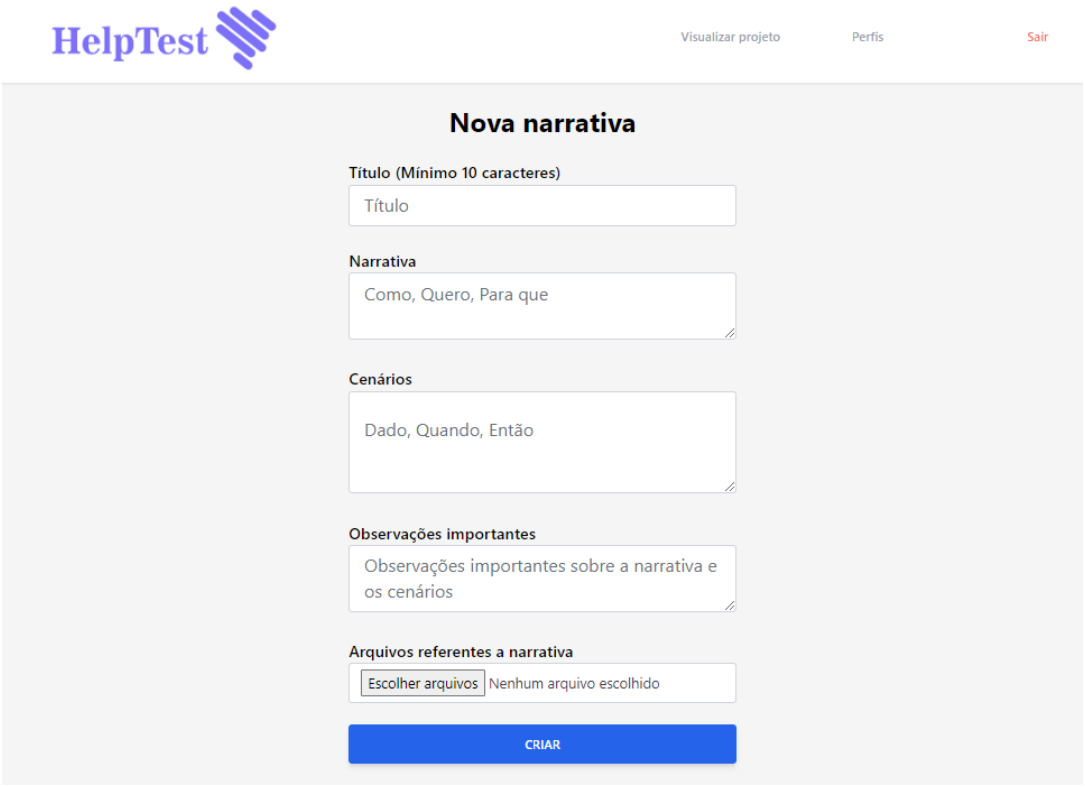
Fonte: Eladorado pelo autor

3.2.4 Perfil Desenvolvedor

A interface do desenvolvedor está disponível para o acesso por meio do botão “Entrar como desenvolvedor” (Figura 3.5). Nesta interface tem a possibilidade da criação de documentos cuja responsabilidade seja do desenvolvedor. Esses documentos são referentes às narrativas e cenários, desde que tenham sido selecionados pelo Administrador durante a criação do projeto.

A criação de um novo documento referente a narrativa e cenários é feita por meio do botão “Iniciar uma nova narrativa com seus cenários”, que irá direcionar o desenvolvedor para a página de criação representada na Figura 3.10.

Figura 3.10: Ilustração da tela criação narrativa e seus cenários

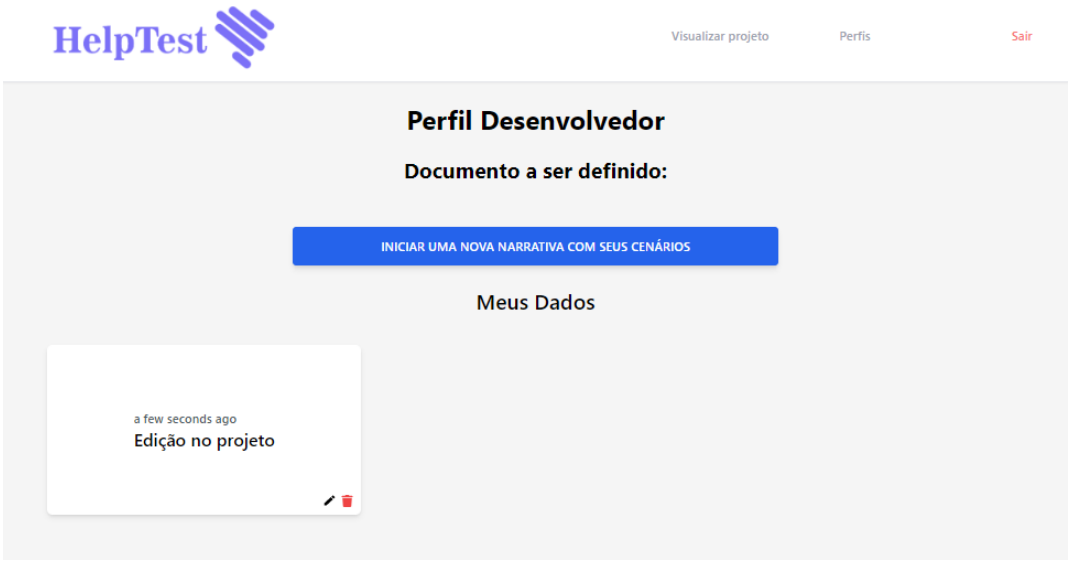


Fonte: Eladorado pelo autor

A criação da narrativa é dada por um título e pela sua descrição, com base nos conceitos estabelecidos pelo BDD. Logo, foi estabelecido um template com os seguintes elementos: Como, Quero, Para que. Para os cenários também foi estabelecido um template com os elementos: Dado, Quando, Então. Além disso, foi adicionado um campo para inclusão de observações importantes e a possibilidade de anexar arquivos importantes referentes à narrativa.

Após a criação da narrativa, a mesma ficará disponível na interface do desenvolvedor para edição, exclusão e visualização, conforme mostrado na Figura 3.11.

Figura 3.11: Ilustração da tela perfil desenvolvedor



Fonte: Eladorado pelo autor

Inicialmente a visualização se dá pelo nome da narrativa, mas é possível visualizar os detalhes, como a narrativa, os cenários, observações importantes, saber o criador e ter acesso aos arquivos fornecidos durante a criação, por meio de um clique no quadro correspondente.

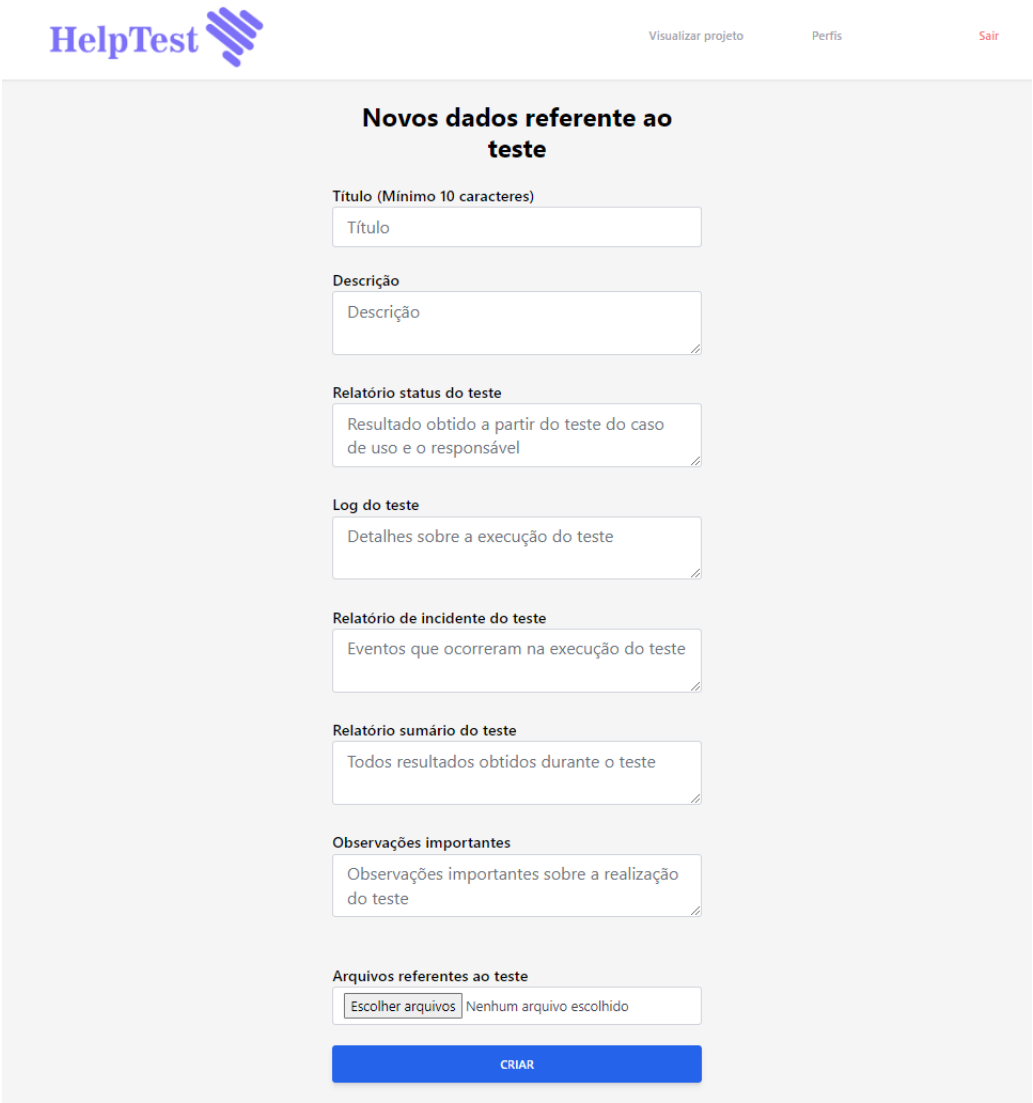
3.2.5 Perfil Testador

A interface do testador está disponível para acesso por meio do botão “Entrar como testador” (Figura 3.5). Tem-se a possibilidade de criação de documentos cuja responsabilidade seja do testador, a saber: relatório status do teste, log do teste, relatório de incidente do teste e relatório sumário do teste. Cabe citar que a criação destes documentos estará disponível somente se foram selecionados pelo Administrador durante a criação do projeto.

A criação desses novos documentos referente ao teste, se dá por meio do botão “Criar novos documentos referente ao teste” que irá direcionar o testador para a página de criação, onde os campos referentes a cada documento estão presentes, con-

forme representado na Figura 3.12. Observa-se que também é possível anexar arquivos referente ao teste.

Figura 3.12: Ilustração da tela criação documentos referentes ao teste



Fonte: Eladorado pelo autor

Com a criação da nova documentação referente ao teste, a mesma ficará disponível na interface do testador para edição, exclusão e visualização. Caso o testador queira visualizar os detalhes da documentação basta clicar no quadro que se refere ao teste.

3.2.6 Perfil Documentador

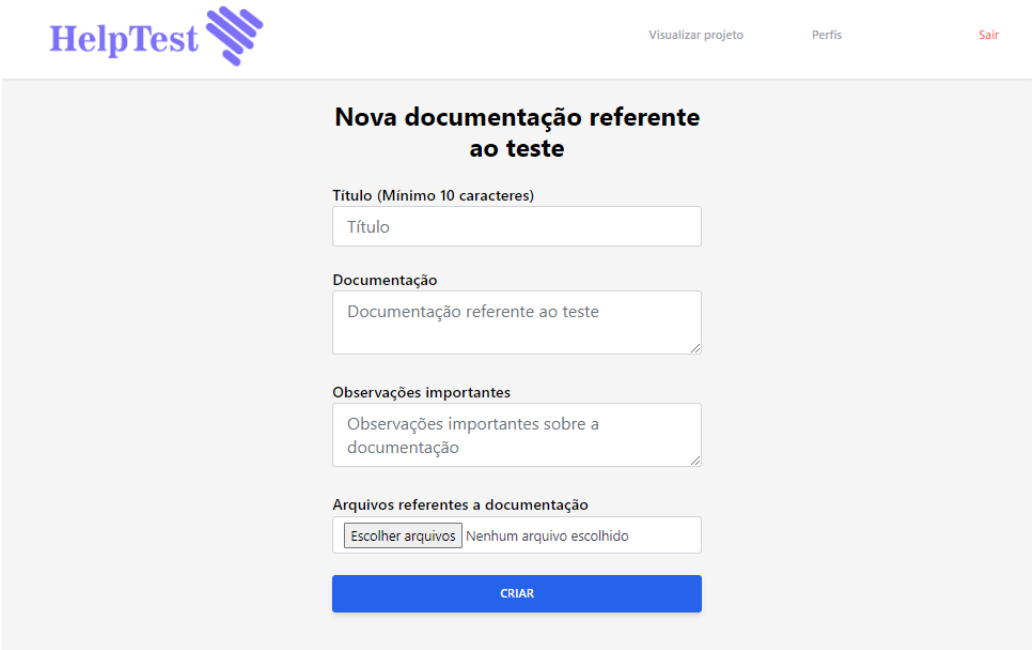
A interface do documentador está disponível para acesso por meio do botão “Entrar como testador” (Figura 3.5). Com isso, tem-se a possibilidade da criação do documento final referente a determinado teste.

Por ser o último a realizar sua função, deve-se ter disponível para visualização os dados referentes ao projeto, à narrativa com seus cenários e aos documentos referentes ao teste já finalizado daquela narrativa. Assim, espera-se que o documentador consiga transformar a linguagem técnica presente nos documentos criados pelos testadores, em uma linguagem mais simples que irá compor a documentação final referente aos testes.

A criação da documentação se dá por meio do botão “Iniciar um novo documento”, que irá direcionar o documentador para a página de criação da documentação. Nesta página estão presentes três campos: título, documentação e observações importantes. Também é permitido anexar arquivos importantes referente a documentação, conforme mostrado na Figura 3.13.

Realizada a criação da nova documentação, a mesma ficará disponível na interface do documentador para edição, exclusão e visualização. Caso o documentador queira visualizar os detalhes da documentação basta clicar no quadro que se refere ao teste.

Figura 3.13: Ilustração da tela de criação da documentação referente ao teste



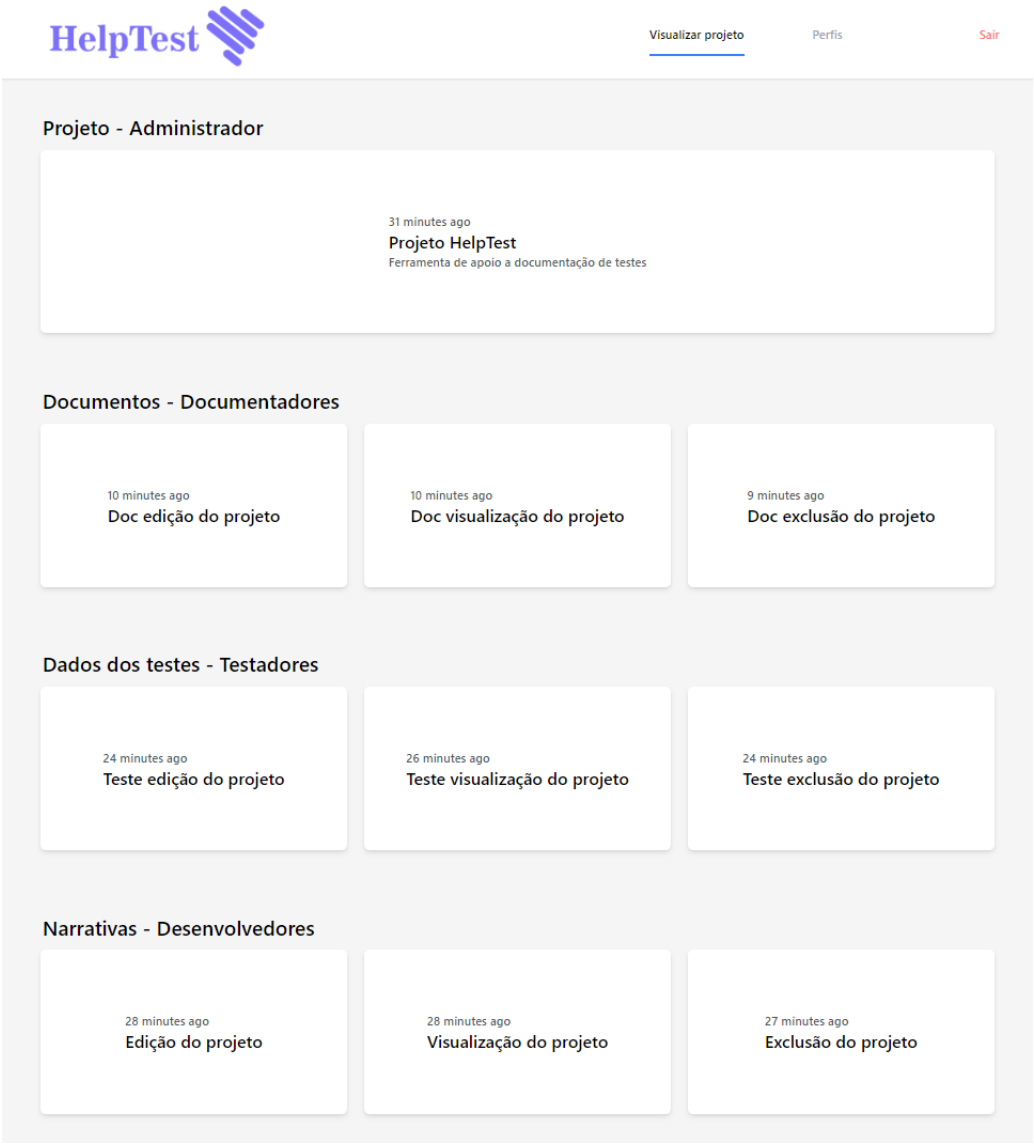
Fonte: Eladorado pelo autor

3.2.7 Opção Visualizar projeto

A opção “Visualizar projeto”, localizada no menu superior das figuras apresentadas, permite a visualização de todos dados referente ao projeto. Inicialmente a visualização se dá por meio dos títulos referente a cada documento já criado, mas é possível visualizar os detalhes por meio de um clique no documento desejado.

Conforme ilustrado na Figura 3.14, tem-se: a seção narrativa, na qual se encontram as narrativas e os cenários inseridos pelos desenvolvedores; a seção dados dos testes, que possui os resultados dos testes realizados para uma narrativa, com documentos como relatórios e logs de teste; e a seção dos documentos criados pelos documentadores, na qual se encontra a documentação final dos testes para um projeto específico.

Figura 3.14: Ilustração da tela visualizar projeto



Fonte: Eladorado pelo autor

Capítulo 4

Resultados e Discussões

Neste capítulo são apresentados a metodologia de avaliação e o perfil dos participantes na Seção 4.1 e a análise dos resultados obtidos por meio da avaliação realizada sobre o uso do protótipo *HelpTest* na Seção 4.2.

4.1 Metodologia de avaliação e perfil dos participantes

Os avaliadores foram convidados pelo autor por meio do aplicativo *Whatsapp* ou por e-mail. O convite continha uma breve descrição do trabalho e um documento instrucional (Apêndice A), com a apresentação e objetivo almejado, bem como os *links* de acesso para o protótipo *HelpTest* e para o formulário de avaliação.

De forma a uniformizar o uso do protótipo e, com isso, garantir uma experiência mínima com os recursos disponibilizados, foi descrito um roteiro de tarefas a serem seguidas para que todos os avaliadores tivessem o mesmo conjunto de ações e pudessem avaliar o protótipo em seguida. O protótipo ficou disponível para a realização de testes durante o período de 21 de novembro de 2022 a 27 de novembro de 2022, via *online*. Ao todo, 19 convidados aceitaram o convite.

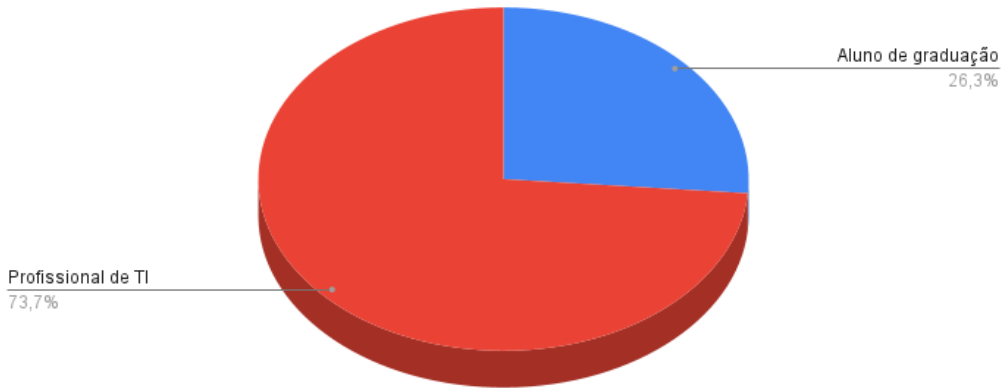
Os avaliadores foram divididos em dois grupos: cinco alunos de graduação em Ciência da Computação e 14 profissionais da área de Tecnologia da Informação – TI,

conforme a Figura 4.1.

Sobre da avaliação pelos alunos de graduação, observa-se que foram convidados colegas do autor que tinham interesse no tema abordado no trabalho.

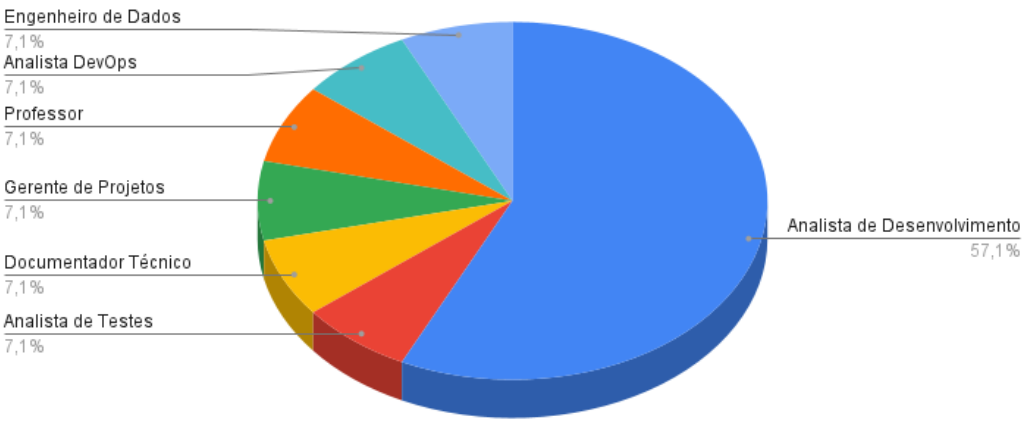
Em relação aos profissionais da área de Tecnologia da Informação, o convite foi realizado a ex-colegas de trabalho onde o autor realizou estágio e também a amigos feitos durante a faculdade que já se encontram em atuação no mercado. Dentre os 14 profissionais da área de Tecnologia da Informação, tem-se: oito analistas de desenvolvimento, um analista de teste, um documentador técnico, um gerente de projeto, um professor, um analista *DevOps* e um engenheiro de dados, exemplificados na Figura 4.2.

Figura 4.1: Função exercida pelos avaliadores



Fonte: Eladorado pelo autor

Figura 4.2: Função exercida pelos profissionais de TI



Fonte: Eladorado pelo autor

Ao longo do período de avalição, os participantes preencheram o formulário de avaliação anonimamente. Os quesitos foram avaliados por meio da escala Likert (BONE et al., 2012), composta neste trabalho pelos itens “Concordo plenamente”, “Concordo”, “Neutro”, “Discordo” e “Discordo plenamente”.

Deste modo, para avaliação dos quesitos, foram propostas sete afirmações aos avaliadores:

Afirmação 1 (A1) – Diante das mudanças recorrentes em projetos atuais, o *HelpTest* contribui para alterações na documentação

Afirmação 2 (A2) – A criação das narrativas e cenários estão bem integradas ao *HelpTest*

Afirmação 3 (A3) – A documentação (Norma IEEE-829) prevista no *HelpTest* contribui para uma documentação abrangente referente às atividades de teste

Afirmação 4 (A4) – As tarefas propostas foram fáceis de realizar

Afirmação 5 (A5) – O modelo de estrutura utilizado para a criação das narrativas e seus cenários facilita no entendimento

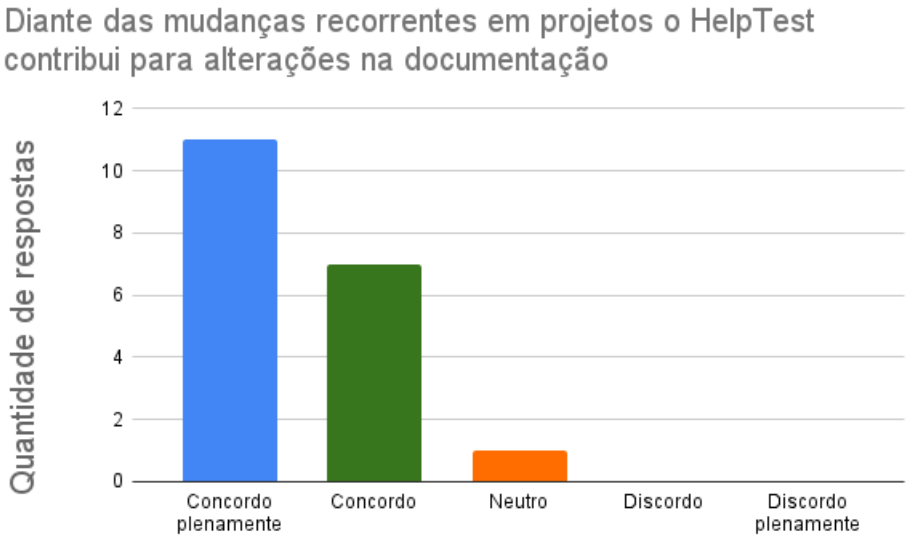
Afirmação 6 (A6) – Os campos a serem preenchidos no *HelpTest* facilitam e guiam na criação de uma documentação referente aos testes

Afirmação 7 (A7) – O *HelpTest*, por meio da aba “Visualizar projetos”, cumpre o objetivo de facilitar a integração entre os diferentes perfis envolvidos no projeto

4.2 Análise dos resultados

Os avaliadores, após seguirem as instruções e utilizar o *HelpTest*, preencheram o formulário avaliativo. A partir da afirmação A1, segundo as respostas obtidas e apresentadas na Figura 4.3, é possível afirmar que foi assegurado que o *HelpTest* no ambiente ágil é bem responsivo a mudanças, visto que ampla maioria dos avaliadores concordaram com a afirmação e apenas um dos avaliadores avaliou como “Neutro”.

Figura 4.3: Respostas dos avaliadores em relação a A1

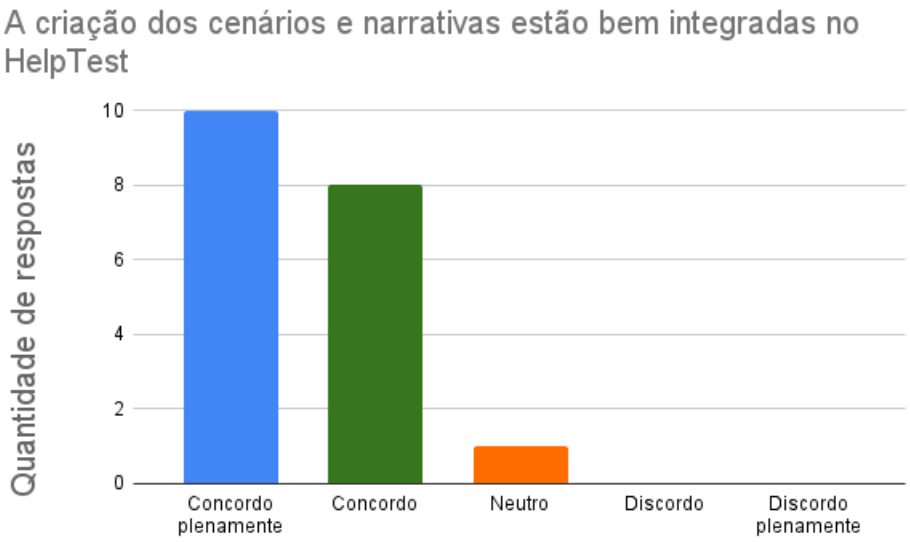


Fonte: Eladorado pelo autor

Já com relação à afirmação A2, segundo as respostas obtidas e mostradas na Figura 4.4, fica evidenciado novamente que ampla maioria dos avaliadores concordaram com a afirmação, sendo possível concluir que a abordagem do BDD, que se refere a criação

das narrativas e cenários, foi implementada de forma bem sucedida no protótipo.

Figura 4.4: Respostas dos avaliadores em relação a A2

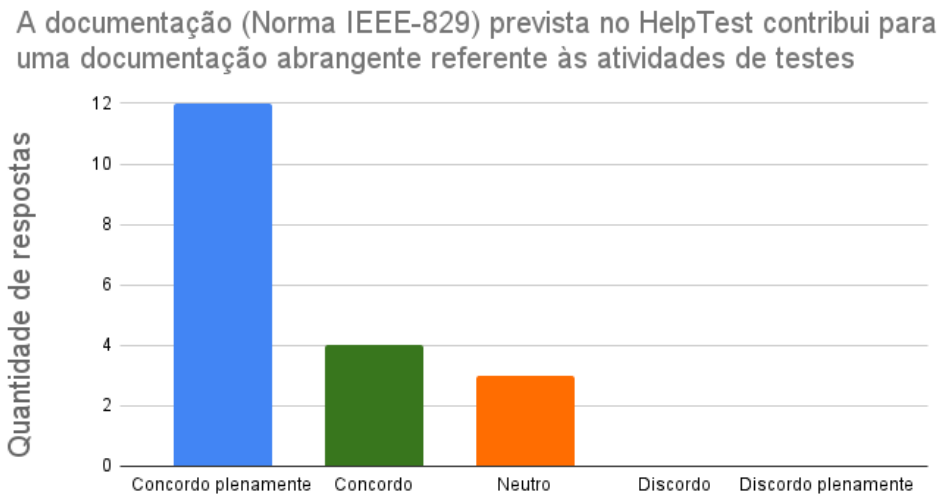


Fonte: Eladorado pelo autor

Referente à afirmação A3, segundo as respostas obtidas e mostradas na Figura 4.5, é correto afirmar que a inclusão da Norma IEEE-829 neste estudo e sua aplicação no protótipo *HelpTest* contribuiu de forma positiva para gerar uma documentação de qualidade e abrangente das atividades de teste.

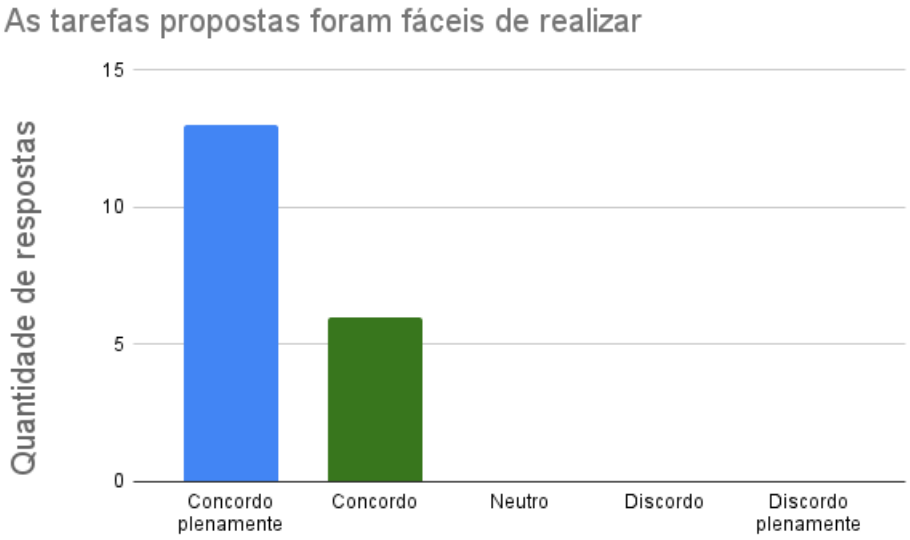
Sobre a afirmação A4, “segundo as respostas obtidas e mostradas na Figura 4.6, é correto afirmar que a usabilidade do protótipo se deu de forma simples e clara, uma vez que todos avaliadores deram um *feedback* positivo em relação a este tema.

Figura 4.5: Respostas dos avaliadores em relação a A3



Fonte: Eladorado pelo autor

Figura 4.6: Respostas dos avaliadores em relação a A4

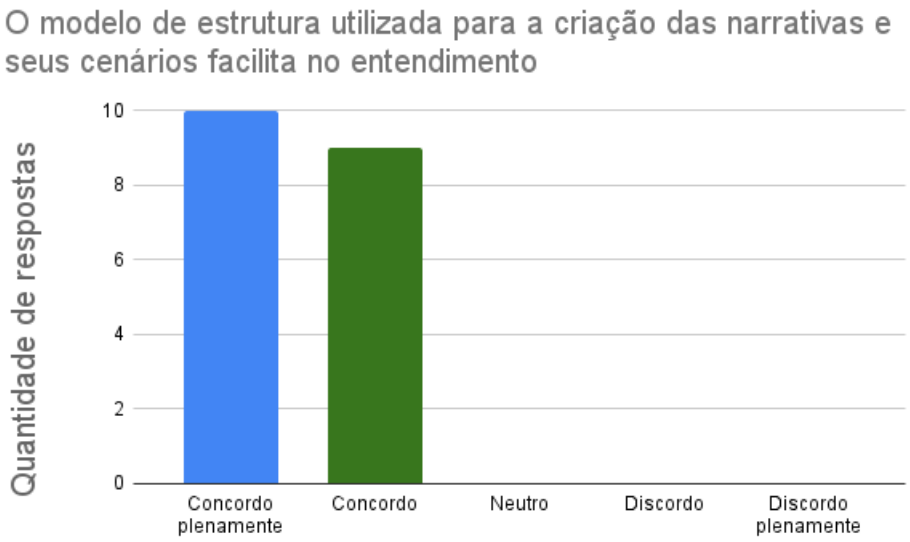


Fonte: Eladorado pelo autor

Conforme a afirmação A5, com base nas respostas obtidas e mostradas na Figura 4.7, é correto afirmar que a adoção do modelo proposto pelo BDD para a criação de narrativas e cenários com o intuito de facilitar o entendimento e melhorar a integração

entre as pessoas no projeto, foi bem sucedida.

Figura 4.7: Respostas dos avaliadores em relação a A5



Fonte: Eladorado pelo autor

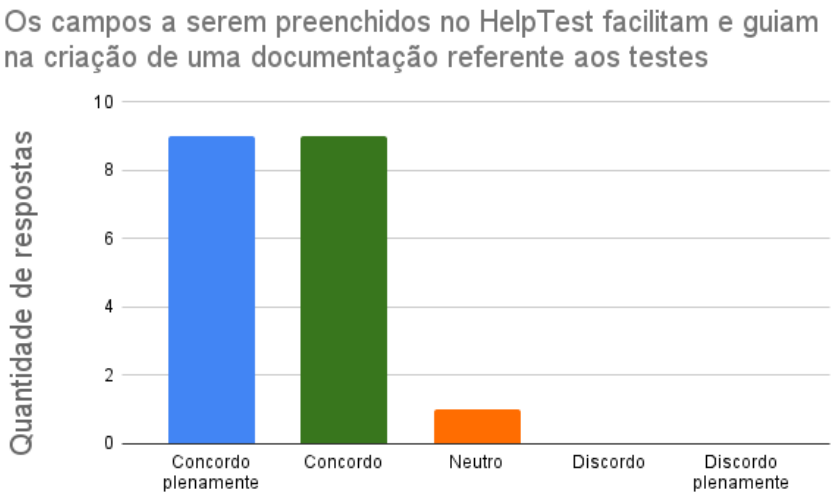
Já a afirmação A6, conforme as respostas obtidas e representadas na Figura 4.8, é possível observar que somente um avaliador optou pela opção “Neutro”. Logo é correto afirmar que os documentos presentes no protótipo e suas respectivas responsabilidades são capazes de nortear os usuários do sistema na criação da documentação, além de facilitar este processo.

Referente à afirmação A7, cujas respostas estão representadas na Figura 4.9, nota-se que a aba “Visualizar projetos” cumpriu de forma correta a integração entre os membros da equipe, assim a visualização dos documentos criados referente ao projeto é facilitada.

Por fim, disponibilizou-se um campo aberto para que os avaliadores pudessem fornecer um *feedback* sobre o protótipo e, assim, permitir a identificação de pontos positivos e carências que não foram contempladas nas afirmações presentes no questionário. Entre os *feedbacks* recebidos, múltiplos avaliadores comentaram sobre a não possibilidade de formatação do texto durante a escrita dos documentos, como edição

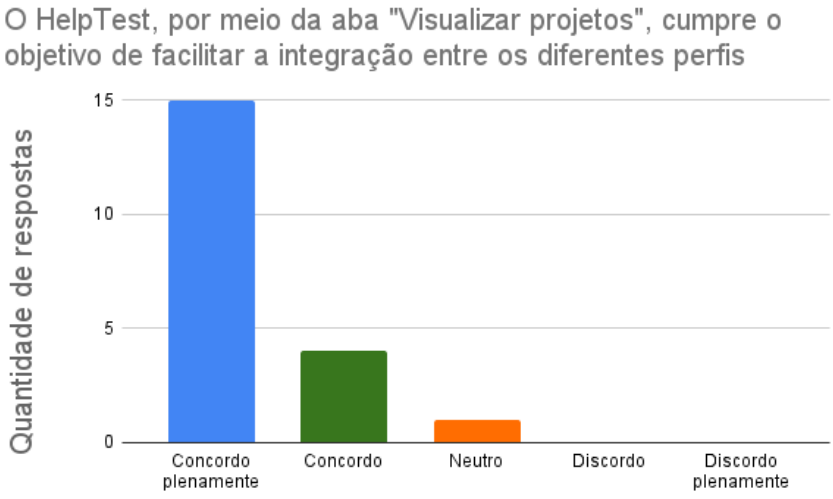
de fonte, estilos, itálico e negrito. Logo, tem-se a sugestão de adicionar um editor de texto que permita tais modificações.

Figura 4.8: Respostas dos avaliadores em relação a A6



Fonte: Eladorado pelo autor

Figura 4.9: Respostas dos avaliadores em relação a A7



Fonte: Eladorado pelo autor

Outro *feedback* recebido, tratou-se do upload de arquivos durante a criação e edição de documentos. Sempre em caso de uma edição, o arquivo que foi previamente

realizado o *upload* é apagado, assim sendo necessário anexar novamente este arquivo. Foi indicado a criação de um *hub* de gerenciamento de arquivos com uma pré-visualização do mesmo em cada etapa do documento, assim possibilita excluir e realizar o upload de arquivo somente se necessário.

Em relação a interface, foram indicadas pequenas alterações que facilitariam a experiência do usuário, como por exemplo: disponibilizar uma aba referente ao usuário logado, com e-mail, nome, assim como seu cargo; utilizar cores mais vivas para facilitar a leitura em algumas telas.

Por fim, tem-se *feedbacks* relacionados à aba "Visualizar projeto", tais como: pensar em uma forma melhorada de mostrar a hierarquia entre os documentos criados; estabelecer uma maior relação entre os documentos, por exemplo, a criação de um campo onde é possível realizar buscas sobre os documentos; e fornecer uma forma de visualização integrada de todos documentos do projeto.

Capítulo 5

Conclusão

Neste capítulo é realizado uma análise das contribuições obtidas com o desenvolvimento do protótipo na Seção 5.1 e são apresentadas propostas de trabalhos futuros na Seção 5.2.

5.1 Contribuições

Mesmo com o advento das metodologias ágeis e surgimento de novas abordagens voltadas a teste, notou-se a existência de poucas ferramentas com o intuito de facilitar a integração entre as pessoas envolvidas no projeto, por meio da criação de uma documentação abrangente referente aos testes realizados. Neste contexto, o trabalho contribuiu com um estudo sobre a criação de documentação de testes baseado na norma IEEE-829, juntamente com consideração a integração em um ambiente ágil e com o BDD. Com isso, supriu-se a carência por um sistema que gere uma documentação de testes mais abrangente e também facilite a integração das pessoas no projeto.

É importante citar que, por meio de um processo de avaliação com pessoas atuantes na área de tecnologia, foi comprovado que o protótipo desenvolvido apresenta contribuições relevantes para a documentação dos testes. Mais especificamente, segundo percepção dos avaliadores, as principais contribuições obtidas com o desenvolvimento

deste trabalho são:

- Possibilidade de criação e edição de uma documentação de testes de qualidade, baseada na Norma IEEE-829, o que contribui para a rastreabilidade durante este processo e também sua aplicação no contexto ágil;
- Aplicação da abordagem BDD que facilita a compreensão durante o processo de criação das narrativas e cenários, o que contribui para a integração, de forma mais simples, dos membros do projeto.

5.2 Trabalhos Futuros

Por meio dos *feedbacks* obtidos a partir dos avaliadores do protótipo, foi possível identificar pontos de melhorias para evolução do *HelpTest* e assim estabelecer objetivos para trabalhos futuros.

Em relação às funções presentes no protótipo, chegou-se à conclusão que possíveis melhorias futuras se referem a artefatos da interface e assim melhorar a experiência do usuário: melhorar a hierarquia de exibição da aba “Visualizar projetos”; adicionar um módulo de edição de texto dentro dos campos onde os documentos serão criados, o que possibilita formas de manipular as fontes do documento; permitir a adição de novos cenários em uma narrativa sem ser pela forma de editar; melhorar detalhes do design da interface voltados para a exibição de informações.

Sobre de novas funcionalidades é recomendada a criação de uma aplicação que forneça um meio de visualização e edição integrada com todos os documentos presentes no projeto e assim criar uma possível reformulação da aba “Visualizar projetos” para adicionar campos de buscas. Além disso, técnicas relacionadas a *Data Visualization* podem ser consideradas, com a finalidade de facilitar o processo de visualização e exibição dos dados presentes no *HelpTest*, por meio de estatísticas e representações gráficas. Também vislumbra-se a criação de um método que possibilite adicionar e

excluir separadamente novos arquivos ligados ao projeto.

Finalmente, após a melhoria do protótipo conforme os *feedbacks* recebidos, a aplicação deste trabalho em uma empresa com projetos reais contribuirá para avaliar a sua efetividade no mercado de trabalho, bem como para identificar outros tipos de documentação que carecem de apoio para definição e acompanhamento durante um projeto.

Referências Bibliográficas

ABRAHAMSSON, P.; SALO, O.; RONKAINEN, J.; WARSTA, J. New directions on agile methods: A comparative analysis. In: IEEE. 2003 Proceedings of the 25th International Conference on Software Engineering (ICSE 03). Portland, OR, USA, 2003.

AGILE, M. Manifesto for Agile Software Development. 2001. Disponível em: <https://agilemanifesto.org/>.

AKBAR, M. A.; JUN, S.; NASRULLAH; KHAN, A. A; MAHMOOD, S.; QADRI, S. F.; HU, H.; XIANG, H. Success factors influencing requirements change management process in global software development. Journal of Computer Languages, v. 51, 2019.

AKBAR, M. A.; SHAFIQ, M.; AHMAD, J.; MATEEN, M.; RIAZ, M. T.; NASRULLAH. Az-model of software requirements change management in global software development. In: IEEE. 2018 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube). Quetta, Pakistan, 2018.

ALQUDAH, M; RAZALI, R. A review of scaling agile methods in large software development. International Journal on Advanced Science, Engineering and Information Technology, v.6, n.6, 2016.

ANDRADE, M. Qualidade de Software. SESES. 2015.

BARTIÉ, A. Garantia da qualidade de software: adquirindo maturidade organizacional. Rio de Janeiro: Campus. 2002.

BECK, K. Programação Extrema (XP) Explicada. Bookman. 2004.

BECK, K. TDD Desenvolvimento Guiado Por Testes. Bookman. 2010.

BIANCHINI, J. Ferramenta de suporte ao planejamento de teste funcional de software a partir de diagramas de casos de uso. 2004. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Universidade Regional de Blumenau, Centro de Ciências Exatas e Naturais, Blumenau, SC.

BLANCO, M. Z. Documentação de Teste Baseado na Norma IEEE 829-Estudo de Caso:"Sistema de Apoio a Tomada de Decisão". Revista TIS, v. 1, n. 1, 2012.

BORBA, P.; SILVA, M. TaRGeT: test and requirement generation tool, 2010. Disponível em: <https://twiki.cin.ufpe.br/twiki/bin/view/CInBTCResearchProject/ToolTarget>.

BORGHI, T. D; POVOAS, P. G; LEMOS, G. S. Testes de Aceitação para a Verificação de Entregas de Parceiros. In: Anais Estendidos do X Congresso Brasileiro de Software: Teoria e Prática. São Bernardo do Campo, 2019

BOONE, H. N.; BOONE, D. A. Analyzing likert data. Journal of extension, v.50, n.2, 2012.

BRUNELI, M. V. Q. A utilização de uma metodologia de teste no processo de melhoria da qualidade de software. 2006. Dissertação (mestrado profissional) - Universidade Estadual de Campinas, Instituto de Computação, Campinas, SP.

CARVALHO, C. E. C.; ABRANTES, C. T.; CAMEIRA, R. F. Métodos ágeis de desenvolvimento de software: um caso prático de aplicação do Scrum. 31th Encontro Nacional de Engenharia de Produção, v. 5, n.1, 2011.

CRESPO, A. N.; DA SILVA, O. J.; BORGES, C. A.; SALVIANO, C. F.; DE TEIVE, M.; JUNIOR, A.; JINO, M. Uma metodologia para teste de Software no Contexto da Melhoria de Processo. In: Anais do III Simpósio Brasileiro de Qualidade de Software. SBC, 2004.

COLLINS, E. F.; LOBÃO, L. M. A.; LUCENA JR, V. F. Experiência em Aplicação de Processo de Teste de Software Iterativo e Automático. In: Brazilian Workshop on Systematic and Automated Software Testes, São Paulo. 2011. p. 1-6.

DE CARVALHO, D. M; MARQUES, D. Proposta de usa da técnica BDD para otimizar a escreita e automação de testes no framework Scrum. 2018. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Instituto Federal de São Paulo, Centro de Computação, São Paulo, SP.

DELAMARO, M.; JINO, M.; MALDONADO, J. Introdução ao teste de software. Elsevier Brasil. 2013.

FELIPE, L. P. O estado da arte em metodologias e práticas ágeis de desenvolvimento de software. 2021. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Pontifícia Universidade Católica de Goiás, Centro de Computação, Goiânia, GO.

HRON, M; OBWEGESER, N. Scrum in practice: an overview of Scrum adaptations. In: 2018 Proceedings of the Hawaii International Conferente on Sytem Sciences. Honolulu, HW, USA, 2018.

KOSCIANSKI, A; SOARES, M. S. Qualidade de software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. Novatec, 2006.

LAWRENCE, R.; RAYNER, P. Behavior-Driven Development with Cucumber: Better Collaboration for Better Software. Addison-Wesley Professional, 2019.

MOE, M. M. Comparatives study of test-driven development (TDD), behavior-driven development (BDD) and acceptance test-drive development (ATDD). International Journal of Trend in Scientific Research and Development, v.3, n.4, 2019.

NORTH, D. Introducing behavior-driven development, 2006. Disponível em: <https://dannorth.net/introducing-bdd/>.

NURMULIAN I, N.; ZOWGHI, D.; POWELL, S. Analysis of requirements volatility during software development life cycle. In: IEEE. 2004 Proceedings of the Australian Software Engineering Conference. Melbourne, VIC, Australia, 2004.

NUNES, R. D . A Implantação das metodologias ágeis de desenvolvimento de software scrum e extreme programming (XP): uma alternativa para pequenas empresas do setor de tecnologia da informação. ForScience, v. 4, n. 2, 2016.

PAULK, M.; DAVIS, N; MACCHERONE, L. On empirical research into Scrum. Institute for Software Research, Pittsburgh: Carnegie Mellon University, 2011.

PINTO, V. H. M. O papel da documentação no desenvolvimento de softwares open source: Uma análise e um estudo de caso. 2021. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade de São Paulo, Instituto de Matemática e Estatística, São Paulo, SP.

PONTES, T. B.; ARTHAUD, D. D. B. Metodologias ágeis para o desenvolvimento de softwares. Ciência E Sustentabilidade, v. 4, n. 2, 2018.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. Métodos ágeis para desenvolvimento de software. Bookman Editora. 2014.

RUSSO, R. F. S. M.; DA SILVA, L. F.; LARIEIRA, C. L. C. Do manifesto ágil à agilidade organizacional. Revista de Gestão e Projetos, v. 12, n. 1, 2021.

SCHDEVA, S. Scrum Methodology. International Journal of Engineering and Computer Science, 2016.

SCHWABER, K. Scrum development process. In: Business object design and implamentation. Business Object Design and Implementation, v.3, n.2, 1997.

SCHWABER, K; SUTHERLAND, J. Guia do Scrum: Definição do Scrum e Regras do Jogo, 2017.

SOARES, M. S. Comparação entre metodologias Ágeis e tradicionais para o desenvolvimento de software. INFOCOMP Journal of Computer Science, v.3, n.2, 2004.

SOLIS, C.; WANG, X.; A study of the characteristics of behaviour driven development. In: IEEE. 2011 Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications. Oulu, Finland, 2011.

SOMMERVILLE, I. Engenharia de Software. Addison Wesley. 2003.

TAKEUCHI, H; NONAKA, I. The new new product development game. Harvard business review, v.64, n.1, 1986.

TESTLINK COMMUNITY. What is Testlink?, 2010. Disponível em: <https://www.testlink.org/>.

WAGENAAR, G.; OVERBEEK, S.; LUCASSEN, G.; BRINKKEMPER, S; SCHNEIDER, K. Working software over comprehensive documentation–Rationales of agile teams for artefacts usage. Journal of Software Engineering Research and Development, v. 6, n. 1, 2018.

Apêndice A

Documento instrucional para avaliação do *HelpTest*

Esse documento apresenta as informações necessárias para a realização do processo de avaliação do protótipo Help. Para o controle do processo de avaliação e confiabilidade do resultado final, informa-se que todos os dados gerados durante o processo de avaliação são tratados de forma anônima. O processo de avaliação é dividido em três partes, sendo imprescindível a realização de todas elas visando à completude do processo de avaliação. Salienta-se que sua opinião é muito importante e, por isso, agradeço sua participação.

A.1 Introdução

Com a popularização das metodologias ágeis também surgiram novas abordagens de desenvolvimento baseadas em testes. Entretanto foram encontradas poucas ferramentas que auxiliam no processo de comunicação entre as equipes envolvidas no projeto com o enfoque em direcionar a documentação dos testes realizados. Assim, inspirando-se na abordagem de testes proposta pelo *Behavior Driven Development* (BDD), levando em consideração a metodologia ágil mais utilizada no mercado, o

Scrum, e um conjunto de documentos referentes aos testes propostos pela norma IEEE-829, foi proposto uma abordagem para gerenciar tal processo. O intuito é gerar um arcabouço de documentação que contribua com o acompanhamento, assim contribuindo para a qualidade dos testes.

A.2 Conhecendo o *HelpTest*

Acesso ao sistema: «link de acesso»

Credenciais: Primeiramente se cadastre na aba de "Cadastre-se", com seu nome, senha e e-mail. Na sequência, na aba entrar, faça o login com as credenciais criadas para iniciar a avaliação do protótipo. Ao realizar o login, estarão disponíveis os quatro perfis do sistema: administrador, desenvolvedor, testador e documentador. Levando em consideração que em empresas menores uma pessoa pode ser responsável por várias funções e visando facilitar a avaliação do sistema desenvolvido, é possível realizar todas as funções a partir da criação de um único usuário.

A.2.1 Criação do projeto pelo Administrador

Ao entrar no sistema, selecionando o botão "Entrar como administrador", você será direcionado a uma nova tela, onde poderá criar um novo projeto e também verificar seus projetos criados. Ao clicar no botão "Iniciar um novo projeto", são apresentados campos a serem preenchidos com dados gerais do projeto, como nome e descrição, além de um campo para seleção dos documentos que deverão ser preenchidos para o referido projeto. Tais documentos incluem aqueles referentes à norma IEEE-829 e também dois adicionais correspondentes ao BDD, sendo eles "Narrativas e cenários" e "Documentações. Logo, tais documentos estarão liberados para os demais usuários responsáveis pelo seu preenchimento somente se forem selecionados durante a criação do projeto. Como administrador, tem-se como responsabilidade quatro documentos

referentes à norma, sendo eles: Plano de testes, especificação do projeto de teste, especificação dos casos de teste e especificação do procedimento de teste. Devido ao fato de serem documentos extensos e praticamente não possuírem mudanças, a ideia é que tais documentos estejam prontos e sejam anexados durante a criação do projeto.

Ao criar o projeto, ele estará disponível para visualização, edição ou exclusão do projeto no seu perfil de administrador enquanto estiver logado. A aba "Visualizar projeto" estará disponível para todos os possíveis usuários do sistema apenas consultarem os detalhes de um projeto.

A.2.2 Criação de narrativas pelo Desenvolvedor

Na aba "Perfis", selecione "Entrar como desenvolvedor" para criar uma narrativa com seus respectivos cenários. Tal conceito foi inspirado na criação de narrativas propostas pelo *Behavior Test Driven Development* (BDD), seguindo o modelo para narrativas Como, Quero, Para Que, além de cenários com o modelo Dado, Quando, Então. Ao criar a narrativa, ela estará disponível para a visualização, edição ou exclusão enquanto estiver logado no seu perfil. A aba "Visualizar projeto", estará disponível para todos os possíveis usuários do sistema apenas para consultarem os detalhes da narrativa e seus cenários.

A.2.3 Adição dos dados referente ao teste pelo Testador

Como testador sua responsabilidade está ligada a quatro documentos da norma IEEE-829: Relatórios dos status dos testes, logs dos testes, relatórios de incidentes dos testes e relatórios sumários dos testes. Como testador, é possível visualizar a narrativa e seus cenários na aba "Visualizar projeto", sendo seu dever preencher os campos de sua responsabilidade que foram selecionados durante a criação do projeto pelo Administrador. Assim, acesse a aba "Perfis", selecione "Entrar como testador", após a realização dos testes conforme a empresa estabelecer, e pressione o botão "Criar

novos documentos referente ao teste" para preencher os campos disponibilizados.

Ao criar um novo documento, ele estará disponível para a visualização, edição ou exclusão enquanto estiver logado no seu perfil. A aba "Visualizar projeto", estará disponível para todos os possíveis usuários do sistema apenas para consultarem os detalhes dos documentos referentes ao teste.

A.2.4 Criação da documentação referente ao teste pelo Documentador

Após a realização dos passos anteriores, a narrativa com seus cenários, os dados referentes ao teste e as especificações do projeto estão disponíveis para a consulta na aba "Visualizar projeto". Logo, é possível iniciar o processo de documentação de responsabilidade do documentador, onde a linguagem dos relatórios do testador, muitas vezes mais técnica, é transcrita de maneira mais simples e de fácil entendimento. Na aba "Perfis", selecione "Entrar como documentador", crie a documentação referente ao teste, levando em consideração as informações do teste disponíveis na aba "Visualizar projeto". Observa-se que tal documentação somente poderá ser criada se o Administrador selecionou este documento para o projeto.

Novamente ao criar a documentação, ela estará disponível para a visualização, edição ou exclusão enquanto estiver logado no seu perfil. A aba "Visualizar projeto", estará disponível para todos os possíveis usuários do sistema apenas para consultarem os detalhes da documentação.

A.2.5 Aba "Visualizar projeto"

Nesta aba, o projeto criado pelo administrador, as narrativas e cenários criados pelo desenvolvedor, documentos referentes aos testes e as documentações finais dos testes estarão disponíveis somente para a consulta. Para tanto, foi subdividida em seções sendo elas: Projeto - Administrador, Documentos - Documentadores, Dados

dos testes – Testadores, e Narrativas - Desenvolvedores. Cada uma exibindo seus respectivos conteúdos, onde ao clicar em algum dos itens, os detalhes são mostrados, assim como o responsável pela sua criação.

A.3 Responder o questionário

Nessa última parte deve-se acessar o link: «link de acesso» e responder um questionário sobre sua experiência de uso do *HelpTest*. O prazo máximo para o preenchimento do formulário é dia 26/11/2022. Novamente, salienta-se que a sua opinião é muito importante e que tal opinião é tratada de forma anônima. Mais uma vez, agradeço a sua disponibilidade, colaboração e empenho na participação desse processo de avaliação. Em caso de dúvida, encaminhe e-mail para paulo.zanele@unesp.br.