UNIVERSIDADE FEDERAL DO PAMPA

Juliano Puiati Pires

Desenvolvimento de um Sistema Colaborativo para apoio à Engenharia de Requisitos

Alegrete
2020

Juliano Puiati Pires

Desenvolvimento de um Sistema Colaborativo para apoio à Engenharia de Requisitos

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Profa. Dra. Aline Vieira de Mello

Alegrete
2020

SERVIÇO PÚBLICO FEDERAL

MINISTÉRIO DA EDUCAÇÃO

Universidade Federal do Pampa

Juliano Puiati Pires

Desenvolvimento de um Sistema Colaborativo para apoio à Engenharia de Requisitos

Trabalho de Conclusão de Curso apresentado ao Curso de (Engenharia de Software da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho de Conclusão de Curso defendido e aprovado em: 10 de dezembro de 2020.

Banca examinadora:

Profa. Dra. Aline Vieira de Mello

Orientadora

UNIPAMPA

Profa. Dra. Amanda Meincke Melo

UNIPAMPA

Profa. Dra. Andréa Sabedra Bordin

Assinado eletronicamente por **ALINE VIEIRA DE MELLO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 16/12/2020, às 17:26, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **ANDREA SABEDRA BORDIN**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 16/12/2020, às 17:36, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **AMANDA MEINCKE MELO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 16/12/2020, às 17:52, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0429159** e o código CRC **6F9EC9C1**.

Este trabalho é dedicado a todos os professores que,
contribuíram para a minha formação acadêmica
e desenvolvimento desta monografia.

AGRADECIMENTOS

Agradeço a Deus pela minha força e disciplina para continuar trabalhando, independente do que aconteça. Agradeço à minha mãe Teresinha e ao meu pai João Carlos, vocês são a minha base. Agradeço à minha orientadora Aline, pela confiança, compreensão e apoio para a conclusão deste trabalho. Agradeço também a todos os meus amigos, que de alguma forma, contribuíram para a conclusão deste trabalho, com uma palavra de apoio, uma ideia, sugestão ou participando dos testes da ferramenta.

“Não seja uma máquina de trabalhar,
consumir e pagar contas.
Tenha tempo para cultivar os afetos.
Quando você acordar, sua vida foi embora.”
(José Mujica)

RESUMO

Colaboração é a ação ou o efeito de colaborar. Remete à ideia de trabalho em conjunto para alcançar objetivos compartilhados. Sistemas Colaborativos, por sua vez, são sistemas projetados para dar apoio ao trabalho em grupo e, dessa forma, auxiliar grupos na comunicação, na cooperação e na coordenação de suas atividades. A Engenharia de Requisitos (ER) é uma área do conhecimento que engloba quatro processos principais: elicitação, análise, especificação e validação de software. Essa área também é responsável pela gestão dos requisitos durante todo o ciclo de vida do produto de software. O processo de ER tem o foco no entendimento do problema. O programa de extensão Programa C - Comunidade, Computação, Cultura, Comunicação, Ciência, Cidadania, Criatividade, Colaboração da UNIPAMPA adota uma metodologia de resolução de problemas em sua atividade Resolve!, em que estudantes dos cursos Ciência da Computação e Engenharia de Software são organizados em equipes para resolver problemas da comunidade local ou regional. Ao envolver a solução de um problema real, a equipe, além de ter problemas para gerenciar um grande número de ferramentas e artefatos envolvidos no processo de desenvolvimento, necessita realizar a Engenharia de Requisitos em colaboração com as partes interessadas na solução do problema, o que envolve, por exemplo, entrevistas, priorização dos requisitos e validação. Este trabalho tem como objetivo propor um sistema colaborativo para dar apoio à realização da Engenharia de Requisitos com a participação das partes interessadas e, assim, apoiar as interações decorrentes da execução de tarefas no contexto da resolução de problemas. Para atingir tal objetivo, foi realizada uma revisão sistemática da literatura, a qual permitiu mapear o estado da arte e conduzir uma análise detalhada sobre os trabalhos que contribuem com a presente proposta. Para a execução do trabalho, foi definida uma metodologia baseada no *framework* Scrum. Seguindo essa metodologia, foi possível definir os requisitos da ferramenta e trabalhar de forma sistemática na sua construção. O resultado obtido foi a ferramenta Clover, que permite realizar a análise e especificação de requisitos de software. A partir de uma avaliação com usuários foi possível comprovar que a ferramenta é adequada ao seu propósito, ou seja, atende ao objetivo principal do trabalho que é oferecer suporte à Engenharia de Requisitos de forma colaborativa.

Palavras-chave: Engenharia de Requisitos. Resolução de Problemas. Sistema Colaborativo.

ABSTRACT

Collaboration is the action or effect of collaborating. It refers to the idea of working together to achieve shared goals. Collaborative Systems are systems designed to support group work and thus assist groups in communicating, collaborating and coordinating their activities. Requirements Engineering (RE) is an area of knowledge that encompasses four main processes: software elicitation, analysis, specification, and validation. This area is also responsible for managing requirements throughout the software product life cycle. The RE process focuses on understanding the problem. The Programa C extension program - Community, Computing, Culture, Communication, Science, Citizenship, Creativity, Collaboration adopts a problem solving methodology in its Resolve! activity, in which students from Computer Science and Software Engineering undergraduate programs are organized in teams. to solve problems of the local community. When involving the solution of a real problem, the team, in addition to having problems managing a large number of tools and artifacts involved in the development process, needs to undertake Requirements Engineering in collaboration with stakeholders to solve the problem, which involves eg interviews, prioritization of requirements and validation. This paper aims to propose a collaborative system to support the realization of Requirements Engineering with the participation of the interested parties, and thus support the interactions arising from the execution of problems in the context of problem solving. To achieve this goal, a systematic review of the literature is carried out, which allowed us to map the state of the art and make a detailed analysis of the works that contribute to this proposal. For the execution of the work, a methodology based on the Scrum framework was defined. Following this methodology, it was possible to define the requirements of the tool and work systematically in its construction. The result obtained was the Clover tool, which allows for the analysis and specification of software requirements. From an evaluation with users, it was possible to prove that the tool is suitable for its purpose, that is, it meets the main objective of the work, which is to support Requirements Engineering in a collaborative way.

Key-words: Requirements Engineering. Problem solving. Collaborative System.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API *Application Programming Interface*

CRUD *Create, Read, Update and Delete*

ER Engenharia de Requisitos

RSL Revisão Sistemática da Literatura

SC Sistemas Colaborativos

UNIPAMPA Universidade Federal do Pampa

SUMÁRIO

1 INTRODUÇÃO

Colaboração é a ação ou o efeito de colaborar [15]. Remete à ideia de trabalho em conjunto para alcançar objetivos compartilhados, em que os participantes têm a intenção em somar algo à solução, não simplesmente trocar informações. Sistemas Colaborativos (SC), por sua vez, são sistemas projetados para dar apoio ao trabalho em grupo e, dessa forma, auxiliar grupos na comunicação, na cooperação e na coordenação de suas atividades [9]. Funcionam como uma interface para um ambiente compartilhado, oferecendo suporte às pessoas engajadas na realização de uma determinada tarefa [8].

Segundo Bourque [6], a ER é uma área do conhecimento que engloba quatro processos principais: elicitação, análise, especificação e validação de software. Essa área também é responsável pela gestão dos requisitos durante todo o ciclo de vida do produto de software [6]. O processo de ER tem o foco no entendimento do problema. Busca-se durante esse processo coincidir o que é requisitado com o que está sendo entendido por parte do engenheiro de software, através do uso de artefatos que apoiam esse processo [20].

As atividades envolvidas no processo de ER podem ser realizadas de forma colaborativa [6]. Nesse contexto, várias pessoas podem contribuir com o projeto e desenvolvimento de uma solução para um determinado problema. Apesar de a ER ser considerada uma área importante para o desenvolvimento de software, requisitos mal levantados ainda são reconhecidos como a principal causa de problemas de software, como por exemplo, falhas para atender às expectativas do cliente [23].

O curso de Engenharia de Software da Universidade Federal do Pampa (UNIPAMPA) possui componentes curriculares denominados Resolução de Problemas, em que os estudantes são organizados em equipes para aprender, praticar e resolver problemas juntos. Para realizar essas atividades, os estudantes utilizam várias ferramentas de software, por exemplo, planilhas ou editores de documentos (Excel, Word, etc.) para documentação de requisitos, ferramentas para *design* (Astah, Modelio, etc.), ferramentas para o gerenciamento de tarefas (Trello, Kambamflow, etc.) e outras para a programação (Visual Studio, Eclipse, etc).

Adicionalmente, o programa de extensão Programa C - Comunidade, Computação, Cultura, Comunicação, Ciência, Cidadania, Criatividade, Colaboração da UNIPAMPA adota uma metodologia similar em sua atividade Resolve!, em que estudantes dos cursos Ciência da Computação e Engenharia de Software são organizados em equipes para resolver problemas da comunidade local ou regional. Ao envolver a solução de um problema real, a equipe, segue uma rotina de atividades para poder realizar a engenharia de requisitos, porém, essas atividades não são triviais e podem envolver uma série de ferramentas, atividades, artefatos e pessoas, e, quanto maior for o número desses itens, maior será a complexidade para mente-los, organiza-los, compartilhar as informações obtidas com os demais membros do time e tomar decisões a partir do corpo de artefatos gerados.

Por mais que existam ferramentas que possam dar apoio a certas atividades realizadas no processo de ER, não existe uma ferramenta "coringa" que cubra todas as atividades que podem ser realizadas durante esse processo e permita a integração de todas as variáveis que envolvem um projeto de software, e ainda, se adapte a qualquer contexto ou cenário de problema, equipe, etc. Na área de engenharia de requisitos sempre vão existir oportunidades de melhorias por se tratar de uma área muito ampla e com muitas atividades para serem cobertas por ferramentas, abordagens e processos.

Assim, surge a oportunidade de serem criadas ferramentas para apoiar as interações decorrentes da execução de tarefas no contexto da resolução de problemas, que envolvam as partes interessadas na solução, de modo que atenda às áreas principais de comunicação, colaboração e coordenação.

1.1 Objetivos

É objetivo geral deste Trabalho de Conclusão de Curso (TCC) propor um SC para dar apoio à realização da ER com a colaboração das partes interessadas. Para atingir tal objetivo, são definidos os seguintes objetivos específicos:

- Verificar o estado da arte para identificar trabalhos que possam contribuir com a presente proposta;
- Realizar uma análise das ferramentas semelhantes, identificando as principais funcionalidades e possíveis contribuições para a presente proposta;
- Desenvolver a primeira versão da ferramenta seguindo a metodologia definida;
- Fornecer um meio de acesso aos problemas mapeados e em desenvolvimento;
- Contribuir para a especificação e acompanhamento de soluções.

1.2 Organização do Documento

O restante deste documento está organizado de acordo com a descrição apresentada a seguir:

- O Capítulo 2 apresenta a base teórica e tecnológica para o entendimento deste trabalho;
- O Capítulo 3 apresenta o resultado de uma revisão sistemática de literatura;
- O Capítulo 4 apresenta a metodologia proposta para alcançar os objetivos deste trabalho;
- O Capítulo 5 apresenta como foi o processo de construção da ferramenta;

- O Capítulo 6 apresenta os resultados obtidos com o trabalho realizado;
- O Capítulo 7 apresenta as considerações finais e o que se pode esperar de futuros trabalhos, envolvendo a ferramenta desenvolvida.

2 FUNDAMENTAÇÃO TEÓRICA E TECNOLÓGICA

Neste Capítulo é apresentada a fundamentação teórica e tecnológica deste trabalho. A Seção 2.1 apresenta os conceitos relacionados ao programa de extensão Programa C. A Seção 2.2 apresenta conceitos relacionados à Engenharia de Requisitos (ER). Uma visão geral sobre Sistemas Colaborativos (SC) é apresentada na Seção 2.3. Por fim, na Seção 2.4 são apresentados os conceitos relacionados ao *framework* Scrum.

2.1 Programa C - Comunidade, Computação,Cultura, Comunicação, Ciência, Cidadania, Criatividade, Colaboração

O programa de extensão "Programa C – Comunidade, Computação, Cultura, Comunicação, Ciência, Cidadania, Criatividade, Colaboração", registrado na UNIPAMPA, tem como objetivo principal atuar na resolução de problemas locais, utilizando tecnologias computacionais. Para isso conta com a colaboração de professores, alunos e comunidade externa. Segundo a coordenadora do programa Aline Vieira de Mello [3], entre os objetivos específicos do Programa C estão "ampliar o espaço da sala de aula, organizando ambientes de aprendizagem significativos, interdisciplinares e interprofissionais" e "ampliar o domínio das ações de extensão na área da Computação, promovendo maior participação docente em práticas extensionistas e atendendo a demandas discentes por experiências em extensão".

O Programa C começou suas atividades no ano de 2016 e desde então promove diversas ações que estão vinculadas a seis atividades: Gera!, Resolve!, Gurias na Computação, 5C, ComputAÇÃO, e Programa C + Educação Básica. O presente trabalho está relacionado às atividades Gera! e Resolve!

A atividade Gera! identifica problemas locais que possam ser resolvidos com uso de tecnologia. Representantes de diferentes setores da comunidade são convidados a apresentar problemas para serem investigados pelos estudantes. Os problemas identificados podem ser abordados e resolvidos na atividade Resolve!. A atividade Resolve! adota uma metodologia de resolução de problemas, em que estudantes dos cursos de Engenharia de Software e Ciência da Computação trabalham em equipes para desenvolver soluções tecnológicas. Dessa forma os alunos podem praticar os conceitos aprendidos em aula e fornecer soluções para os problemas da comunidade.

Foi através da participação e colaboração para a resolução de problemas, bem como a discussão com os demais membros da equipe da atividade Resolve!, que surgiu a ideia inicial e os primeiros *insights* em torno das funcionalidades que a ferramenta proposta poderia dar apoio, servindo assim, como base para a produção deste trabalho.

2.2 Engenharia de Requisitos

Requisitos de software são descrições que expressam as funcionalidades do sistema em relação as necessidades do usuário ou partes interessadas, ou seja, o que ele deve fazer e como deve se comportar, os serviços que deve oferecer e as restrições que pode ter. Descrevem também as propriedades que devem estar presentes no software para que ele possa resolver problemas do mundo real [23] [6].

Segundo Pressman [20], ER é o nome dado ao conjunto de diversas técnicas que levam a um entendimento dos requisitos. Sommerville [23], por sua vez, define ER como o nome dado ao processo de analisar, documentar e verificar serviços e restrições do sistema. Já para Bourque [6], a ER é uma área do conhecimento que engloba quatro processos principais: elicitação, análise, especificação e validação de software. O presente trabalho adota o conceito de ER apresentado por Bourque [6], o qual é descrito nas próximas seções.

2.2.1 Elicitação de Requisitos

O processo de Elicitação de Requisitos está relacionado à compreensão básica do problema que o software precisa resolver. É o processo no qual as partes interessadas são identificadas e a relação com o cliente é estabelecida. Através desse contato, o engenheiro de software extrai informações para formular os requisitos [6]. Para ter sucesso na obtenção dos requisitos, é importante utilizar métodos de comunicação eficazes entre usuários/partes interessadas e engenheiro de software. A partir das informações extraídas, o engenheiro de software deve formular os requisitos. Entre as principais fontes de requisitos, destacam-se:

- *stakeholders* - segundo Sommerville [23], um *stakeholder* do sistema é quem tem alguma influência direta ou indireta sobre os requisitos do sistema. Os *stakeholders* incluem os usuários finais que irão interagir com o sistema e qualquer outra pessoa em uma organização que será afetada por ele;

- documentos - alguns documentos podem conter informações importantes ou relevantes e contribuir na descoberta de requisitos.

- outros sistemas - esses incluem sistemas em operação, anteriores ou concorrentes, os quais podem servir como base para a elicitação.

Para a obtenção dos requisitos existem diversas técnicas [6], como:

- entrevista - técnica na qual o *stakeholder* responde a um conjunto de perguntas e o engenheiro de software fica responsável por explorar questões relacionadas ao sistema, com a finalidade de obter uma melhor compreensão sobre o mesmo;

- *brainstorming* - esta técnica tem como finalidade gerar um conjunto de ideias em um curto intervalo de tempo, em que um grupo discute um problema de maneira informal para alcançar esse objetivo;
- cenário - é uma descrição que exemplifica interações com o sistema. Existem diferentes tipos de descrições de cenários, que oferecem tipos diferentes de informações, em variados níveis de detalhamento sobre o sistema;
- observação - com esta técnica o engenheiro de software observa os processos e procedimentos executados no ambiente real de trabalho, documentando e fazendo observações sobre esses.

2.2.2 Análise de Requisitos

O processo de Análise de Requisitos consiste em analisar um conjunto inicial de requisitos, com a finalidade de identificar e descrever os conceitos relacionados ao domínio do problema. Durante esse processo obtém-se uma visão geral do sistema, definindo como os objetos do domínio da aplicação se relacionam e como atuam para atender aos requisitos [6]. Entre as atividades realizadas durante o processo de análise de requisitos, destacam-se:

- detecção e resolução de conflitos entre requisitos;
- elaboração dos requisitos de sistema para derivar requisitos de software;
- classificação dos requisitos;
- modelagem conceitual;
- descrição de cenários de interação;
- negociação dos requisitos;
- análise formal dos requisitos.

2.2.3 Especificação de Requisitos

O processo de especificação consiste em documentar os requisitos a fim de serem revistos, avaliados e aprovados. Segundo Sommerville [23], os requisitos devem ser descritos de forma clara, inequívoca, completa, consistente e ser de fácil compreensão, porém isso é difícil de ser alcançado devido a diferentes interpretações do problema por parte dos envolvidos.

Existem várias maneiras de serem especificados os requisitos. De uma maneira geral, leva-se em conta na descrição dos requisitos a separação dos níveis de requisitos entre requisitos de usuário e de sistema. Os requisitos de usuário são declarações em

linguagem natural e diagramas contendo as funcionalidades e as restrições sob as quais o sistema deve operar [23]. Os requisitos de sistema têm uma definição mais técnica e detalham as funcionalidades e restrições do sistema. Também acrescentam mais detalhes e explicam como os requisitos de usuário devem ser atendidos pelo sistema [23].

São formas de especificação de requisitos:

- linguagem natural - textos expressos em linguagem natural;
- linguagem formal - a descrição dos requisitos seguem uma notação matemática rígida;
- linguagem natural estruturada - a descrição dos requisitos é estruturada através de uma determinada padronização, formulário ou modelo.

2.2.4 Validação de Requisitos

Por fim, o processo de validação de requisitos consiste na verificação de todo o artefato resultante durante o processo de ER, a fim de avaliar se o que está definido representa o real e correto entendimento do que se espera do software [23]. Também é importante verificar se um documento de requisitos está em conformidade com os padrões da empresa e se é compreensível, consistente e completo. Deve-se garantir que os requisitos definam o software correto (ou seja, o software que os usuários esperam). Algumas técnicas utilizadas nesse processo são:

- revisão de documentos;
- aplicação de *checklists*;
- prototipagem;
- validação de modelos;
- modelagem conceitual;
- testes de aceitação.

2.3 Sistemas Colaborativos

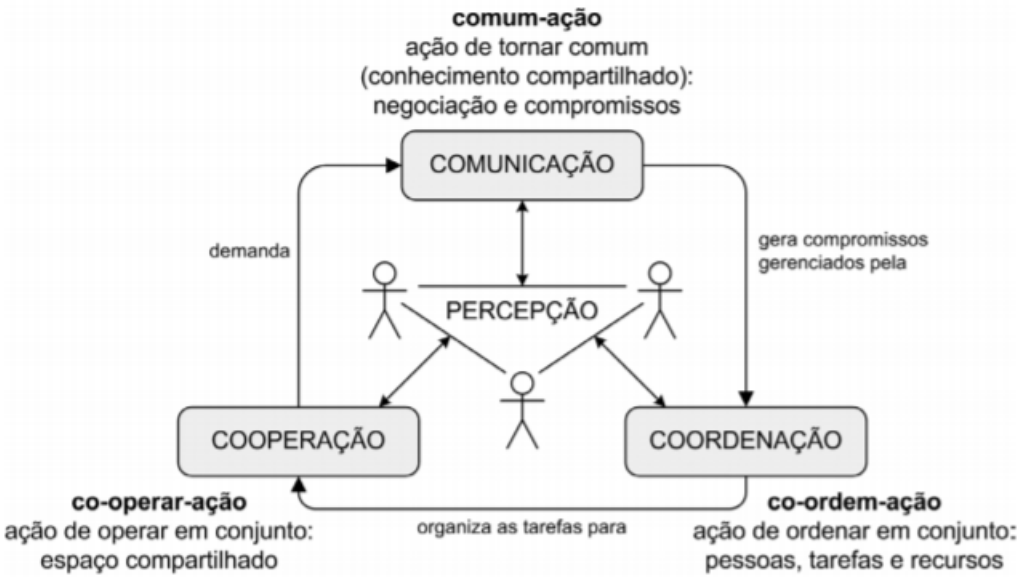
Os avanços tecnológicos influenciam diretamente na forma de trabalho. O paradigma de comando e controle, no qual as pessoas são preparadas para reagir a ordens claras, procedimentos bem definidos e atividades de preferencia individual, perde eficácia, devido à complexidade e interdisciplinaridade das tarefas e a demandas da sociedade da informação [19]. Assim, a forma de trabalho acaba sendo substituída por outra menos hierarquizada e mais participativa, onde predominam a comunicação, a coordenação, a cooperação e a formação de grupos para resolverem os problemas.

Segundo Ellis et al. [8], sistemas colaborativos (SC) ou *groupware* são sistemas baseados em computador que suportam grupos de pessoas envolvidas em uma tarefa (ou objetivo) comum e que fornecem uma interface para um ambiente compartilhado. No Brasil, segundo Fuks [9], SC é a tradução adotada para representar tanto o termo ‘“*groupware*” quanto o termo “CSCW” (*Computer Supported Cooperative Work*). Ambos os termos estão relacionados a sistemas computacionais para apoiar a colaboração, e podem ser usados como sinônimos [9].

O objetivo do SC é ajudar os grupos a se comunicarem, cooperarem e coordenarem suas atividades. Ou seja, esses sistemas servem para apoiar as interações que ocorrem ao se trabalhar em tarefas que ocorrem no contexto de um grupo, de modo que atenda às áreas principais de comunicação, colaboração e coordenação [9]. Assim, grupos diversos podem cooperar e centralizar seus trabalhos de forma dinâmica e organizada.

Segundo Fuks [9], um modelo científico é uma representação lógica ou matemática de um fenômeno. É uma descrição do fenômeno de forma abstrata, conceitual, gráfica ou visual. É usado para explicar, analisar e fazer previsões sobre um fenômeno. Dentre os modelos relacionados a sistemas de colaboração, destaca-se o Modelo 3C de Colaboração, apresentado na Figura 1.

Figura 1 – Representação da colaboração.



Fonte: Fuks [9].

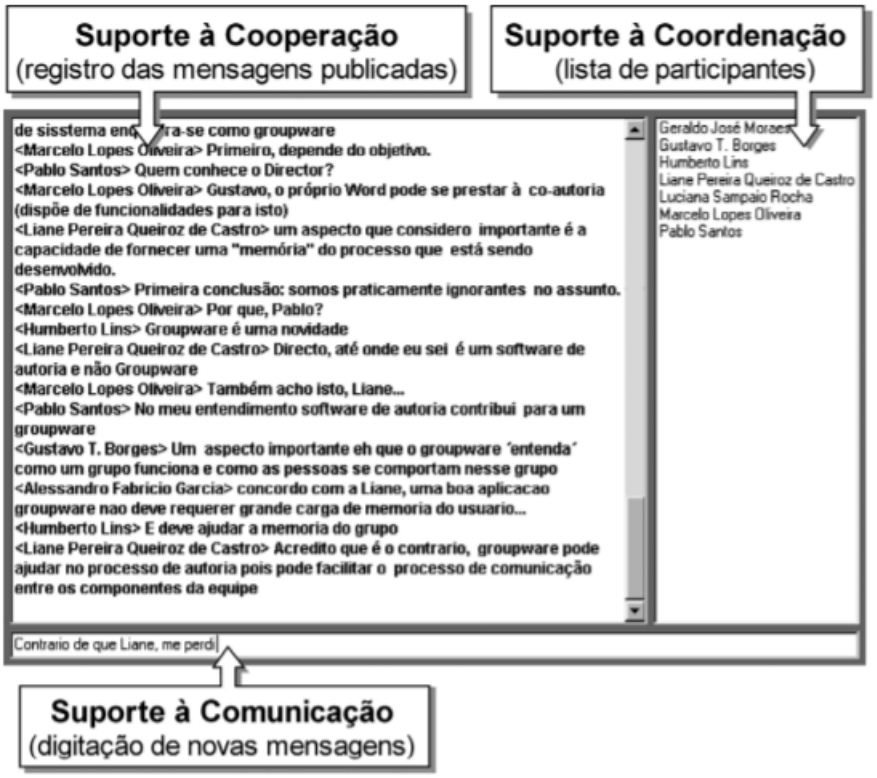
Esse modelo representa como a colaboração é abordada em SC. Segundo Pimentel [19], a comunicação se realiza através da troca de mensagens; a coordenação se realiza através do gerenciamento de pessoas, atividades e recursos; e a cooperação se realiza através de operações num espaço compartilhado para a execução das tarefas.

Segundo Fuks [9], no trabalho em grupo, a comunicação é voltada para a ação. Enquanto se comunicam, as pessoas negociam e tomam decisões. Enquanto se coorde-

nam, os membros do grupo lidam com conflitos e organizam as atividades. A necessidade de renegociar e tomar decisões sobre situações imprevistas que ocorrem durante a cooperação demanda comunicação que, por sua vez, demanda coordenação para reorganizar as tarefas. Por meio de informações de percepção, o indivíduo obtém *feedback* de suas ações e *feedthrough* das ações de seus colegas.

Esses elementos relacionados à colaboração podem ser implementados em diferentes níveis da aplicação. A Figura 2 exemplifica os elementos do modelo em um bate-papo. Por mais que o bate-papo seja projetado para permitir a comunicação, ele pode conter elementos que contribuem para a coordenação, como controlar os participantes presentes na conversa, e também, dar apoio para a cooperação através do registro da contribuição de cada usuário.

Figura 2 – Elementos para a comunicação, cooperação e coordenação em um bate-papo.



Fonte: Fuks [9].

O trabalho em grupo segue alguns padrões, que são definidos por Fuks [9] como padrões de colaboração. Os padrões de colaboração foram elaborados a partir da pressuposição de que todo trabalho em grupo se resume a certos tipos de atividades. Segundo Fuks [9], o trabalho em grupo envolve uma composição das seguintes atividades:

• Geração - é a atividade em que o grupo trabalha para aumentar a quantidade de informação sobre um assunto. Para isso, são coletadas, produzidas e detalhadas informações. O *brainstorming* é um exemplo de uma técnica usada nesse contexto;

• Redução - é a atividade em que o grupo reduz o número de informações sobre um assunto. Essa atividade consiste na seleção de um subconjunto de informações, abstração em conceitos mais genéricos, ou resumo de informações. A redução é realizada para que se dê mais atenção aos elementos resultantes. Objetiva-se diminuir a carga cognitiva e o trabalho posterior;

• Esclarecimento - é a atividade em que o grupo esclarece o significado dos termos compartilhados pelo grupo. Nessa atividade, o grupo descreve o significado de termos com o objetivo de aumentar o conhecimento e definir um vocabulário de referência compartilhado por todos. Um exemplo é a definição de um dicionário de termos da área do trabalho do grupo;

• Organização - é a atividade em que o grupo estabelece os relacionamentos entre as informações. O grupo pode classificar as informações em categorias, ou estruturar as informações de alguma forma (por exemplo, em uma estrutura hierárquica);

• Avaliação - é a atividade em que o grupo define o valor relativo das informações. O grupo pode: votar em informações, ranquear numa escala de valores, ou avaliar as informações com julgamentos qualitativos. Um exemplo é a filtragem colaborativa em que cada usuário pontua itens, como livros e filmes, o que resulta na recomendação dos itens em função do interesse ou experiência do grupo;

• Comprometimento - é a atividade em que grupo aumenta o número de membros dispostos a se comprometer com uma proposta. A meta é chegar a acordos aceitáveis pelos membros. Um exemplo é a busca de um consenso pelo grupo.

2.4 Scrum

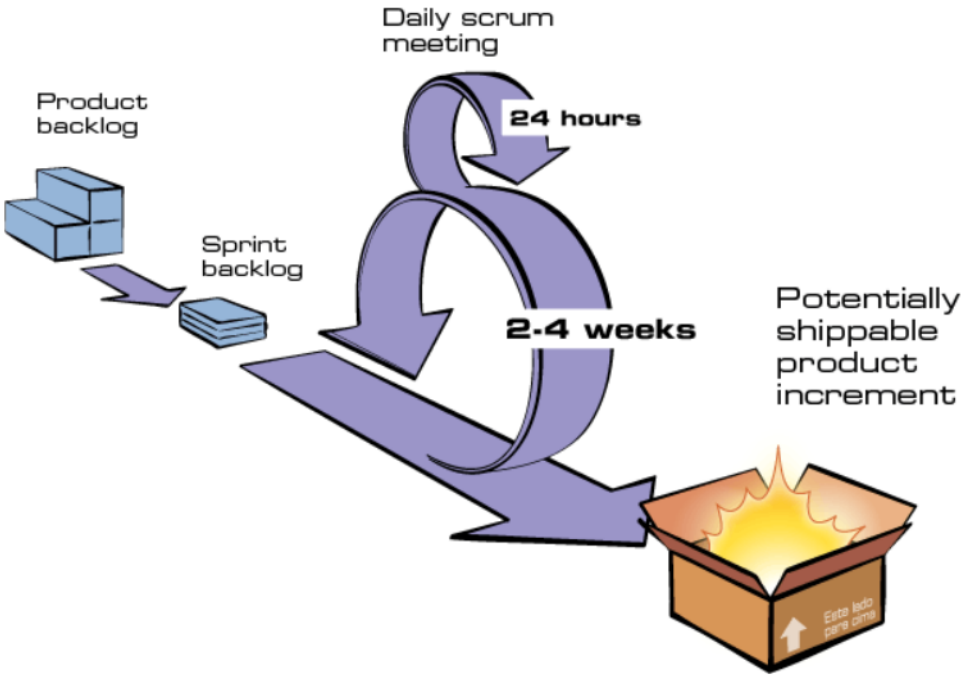
Segundo Schwaber [22], Scrum é um *framework* estrutural para desenvolver e manter produtos complexos. Ele começou a ser utilizado para desenvolvimento de software no início de 1990. Nesse contexto, o Scrum é aplicado como uma metodologia ágil para gestão e planejamento de projetos [1].

A estrutura do Scrum é formada por papéis bem definidos associados a eventos, artefatos e regras [22]. A Figura 3 representa a estrutura base do Scrum com suas principais etapas.

Para o cumprimento dessas etapas, são definidos prazos visando à entrega de um produto de forma rápida e que ao mesmo tempo atenda ao que o cliente espera [22]. Os papéis que compõem o *Scrum* são:

- ***Product Owner*** é a pessoa que define os itens que compõem o *Product Backlog* e os prioriza durante as reuniões de planejamento do *Sprint*, ou seja, descreve as funcionalidades de maior prioridade para a equipe [1]. Representa o desejo das partes interessadas em relação ao produto [22];

Figura 3 – Etapas do Srum.



Fonte: Scr [1].

- **Time de desenvolvimento** são as pessoas que realizam o trabalho de entregar uma versão usável do software, ou seja, um incremento "pronto" do software, ao final de cada *Sprint* [22];

- ***Scrum Master*** é a pessoa responsável por garantir que o Scrum seja entendido e aplicado, assegurando que a equipe respeite e siga os valores e as práticas do Scrum [22], [1]. Ele também atua como um facilitador, pois deve ser responsável por remover obstáculos que estejam dificultando o trabalho da equipe [22].

As atividades que compõem o *Scrum* são:

- ***Sprint*** é o coração do *Scrum*, um período de tempo de um mês ou menos, durante o qual um incremento potencialmente utilizável do produto é criado. Um novo *Sprint* inicia imediatamente após a conclusão do *Sprint* anterior [22];

- **Reunião de planejamento do *Sprint*** corresponde à realização da reunião para planejar o trabalho que será realizado no *Sprint*. Esse planejamento é criado com o trabalho colaborativo de todo o Time Scrum [22]. Segundo Schwaber [22], a reunião de planejamento do *Sprint* deve responder as seguintes questões: O que pode ser entregue como resultado do incremento da próxima *Sprint*? Como o trabalho necessário para entregar o incremento será realizado?

- **Reunião diária** é a reunião que serve para discutir sobre o trabalho realizado desde a última reunião diária e prever o trabalho que deverá ser feito antes da próxima Reunião Diária [22]. Segundo Schwaber [22], durante a reunião os membros do Time de Desenvolvimento esclarecem: O que eu fiz ontem que ajudou o Time de Desenvolvimento a atender a meta do *Sprint*? O que eu farei hoje para ajudar o Time de Desenvolvimento atender a meta do *Sprint*? Eu vejo algum obstáculo que impeça a mim ou o Time de Desenvolvimento no atendimento da meta do *Sprint*?

- **Revisão do *Sprint*** é executada ao final de cada *Sprint*. Serve para inspecionar o incremento e, se necessário, adaptar o *Backlog* do Produto. Esta é uma reunião informal entre o Time Scrum e as partes interessadas. A apresentação do incremento destina-se a motivar e obter comentários e promover a colaboração [22];

- **Retrospectiva do *Sprint*** consiste em um momento de reflexão pelo Time Scrum, para se auto inspecionar e identificar melhorias para serem aplicados no próximo *Sprint*. Essa etapa acontece após a Revisão do *Sprint* [22].

Os artefatos que compõem o *Scrum* são:

- ***Backlog* do Produto** é uma lista ordenada de tudo que deve ser necessário no produto [1]. Um *Backlog* do Produto nunca está completo. Os primeiros desenvolvimentos apenas estabelecem os requisitos inicialmente conhecidos e melhores entendidos. O *Backlog* do Produto existirá enquanto o produto também existir;

- ***Backlog* do Sprint** é um conjunto de itens do Backlog do Produto selecionados para a Sprint. Corresponde à previsão do Time de Desenvolvimento sobre qual funcionalidade estará no próximo incremento e sobre o trabalho necessário para entregar essa funcionalidade em um incremento "Pronto" [1];

- **Incremento** é a soma de todos os itens do *Backlog* do Produto completados durante um *Sprint* e o valor dos incrementos de todas os *Sprints* anteriores [22].

Por fim, a definição de "Pronto" corresponde ao item de *Backlog* do Produto. É utilizado para assegurar quando o trabalho está completo em relação ao incremento do produto. Cada *Sprint* deve entregar incrementos de funcionalidades potencialmente utilizáveis, que atendam à definição de "Pronto" do Time Scrum [22].

3 TRABALHOS RELACIONADOS

Neste Capítulo, é reportado o resultado de uma Revisão Sistemática da Literatura (RSL), a qual permitiu mapear estudos relacionados ao da presente proposta. Na seção 3.1 é descrita a motivação da realização dessa RSL. A seção 3.2 apresenta o protocolo da RSL. A seção 3.3 apresenta os Resultados da execução da RSL. Por fim, na seção 3.4 são apresentadas algumas lições aprendidas com a realização dessa RSL.

3.1 Motivação da RSL

Segundo Kitchenham and Charters [13], uma RSL é um método para examinar em profundidade e descrever a metodologia e os resultados dos estudos primários. Ela possui questões de pesquisa mais específicas relacionadas a um estudo, o que permite uma análise mais aprofundada do escopo que está sendo investigado.

Em uma análise exploratória preliminar, foi realizada uma busca na literatura por outras revisões que abordassem a presente temática, porém, elas não respondem às questões de pesquisa propostas, pois não abordam especificamente ferramentas colaborativas para apoio ao processo de ER. Também, através da análise exploratória preliminar, foram levantados os principais termos relacionados ao presente trabalho. Esses termos foram usados para compor a *string* de busca, que será apresentada na seção seguinte.

3.2 Protocolo

O objetivo dessa RSL foi mapear estudos que apresentam uma ferramenta colaborativa de suporte à ER, que possam contribuir com a presente proposta. Dessa forma, foi realizada uma análise detalhada dessas ferramentas. Para alcançar o resultado esperado na RSL, foi utilizado como base para a RSL o processo definido por Kitchenham and Charters [13].

3.2.1 Questões de Pesquisa

Para alcançar o objetivo principal da RSL, buscou-se saber como a ER é abordada por essas ferramentas, bem como as possíveis formas de colaboração que a ferramenta dá suporte, para que haja, assim, uma análise dessas informações. A partir disso, foram definidas as seguintes questões de pesquisa:

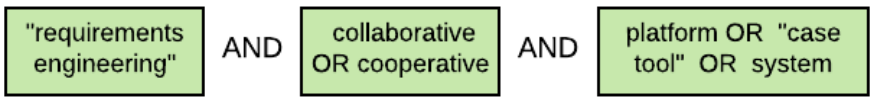
- RQ1: Como a ferramenta dá apoio à Engenharia de Requisitos?
- RQ2: Como a ferramenta permite a colaboração de pessoas no processo de Engenharia de Requisitos?

3.2.2 Processo de Busca

O processo de busca foi realizado através da consulta em base de dados. Foram escolhidas para realizar a consulta as seguintes bases: Scopus [1], IEEE[2], ACM[3] e Science Direct[4], por serem amplamente utilizadas no meio acadêmico e indexar outras bases cientificas, permitindo a busca de documentos de informática a partir de palavras chaves e possível aplicação de filtros, aumentando assim a precisão e padronização da consulta.

A *string* de busca foi estruturada identificando três conjuntos de palavras-chave. O primeiro mapeava a população, ou seja, o conjunto universo de trabalhos relacionados à ER. O segundo mapeava a intervenção, que é relacionada às características atuantes sobre a população. Por fim, o terceiro mapeava o resultado, que corresponde ao conjunto de ferramentas relacionadas à proposta do trabalho. A Figura 4 apresenta a *string* base utilizada no processo de busca.

Figura 4 – *String* base.



Fonte: Próprio autor.

3.2.3 Processo de Seleção

Os trabalhos relevantes foram selecionados aplicando critérios de inclusão (CI) e exclusão (CE) nos estudos recuperados pelo processo de busca. Dessa forma, se um artigo fosse classificado no CI ele seria incluído como estudo primário, porém se um artigo fosse associado a pelo menos um CE, ele seria excluído. Na RSL foram definidos os seguintes CI e CE:

- CI: Artigos que apresentem uma ferramenta com suporte à colaboração de pessoas, para apoio à Engenharia de Requisitos;
- CE1: Artigos que não estão em inglês ou português;
- CE2: Artigos com menos de 6 páginas;
- CE3: Artigos que não esteja completamente disponível para download;

[1] https://www.scopus.com/
[2] https://ieeexplore.ieee.org/
[3] https://dl.acm.org/
[4] https://www.sciencedirect.com/

- CE4: Artigos duplicados.

Para a seleção final, os trabalhos foram lidos por completo para garantir a integridade das informações, aplicando os critérios novamente, se necessário.

3.2.4 Coleta e Análise dos Dados

Os dados foram coletados e analisados por um único revisor. Para extrair os dados relevantes dos artigos, foi produzido um formulário [5] que ajuda a responder às questões de pesquisa. Foram coletados os seguintes dados:

- id;
- base de dados;
- título;
- autor;
- data da publicação;
- trecho do texto que descreve as principais funcionalidades da ferramenta;
- trecho do texto que descreve como a ferramenta contribui no processo de Engenharia de Requisitos;
- trecho do texto que descreve como a ferramenta permite a colaboração de pessoas através do seu uso.

Foram tabulados os dados em uma planilha para resumir as informações sobre cada estudo, ordenando por ano de publicação. Os dados foram analisados e suas informações foram cruzadas para responder as questões de pesquisa com base na seguinte estratégia:

- Para responder a RQ1, foram relacionados os processos de ER com as técnicas de ER utilizadas em cada ferramenta;
- Para responder a RQ2, foi relacionada a tecnologia utilizada no processo colaborativo com a abordagem utilizada para permitir a colaboração de pessoas.

3.3 Resultados

Foi realizado o processo de busca no dia 23 de agosto de 2019. A Tabela 1 apresenta o histórico da busca nas bases de dados. Foram recuperados 676 estudos, dos quais 43 foram aceitos aplicando apenas o CI, na etapa inicial de seleção.

[5] https://drive.google.com/drive/folders/1REITzYMneZPTo4o5Qg7U5psBOjPYk2XC?usp=sharing

Após a seleção inicial foram aplicados os critérios de exclusão ao conjunto inicial de trabalhos. Como resultado, foram rejeitados 30 dos 43 estudos previamente aceitos. Portanto, foram aceitos 13 estudos primários para esta RSL. Uma vez que a lista de estudos relevantes foi definida, foi realizada a extração de dados e analisados os resultados para responder às questões de pesquisa.

Tabela 1 – Histórico de execução do processo de busca e seleção inicial

Bases	Recuperação	Selecionados
Scopus	430	23
IEEE	144	12
ACM	77	4
Science Direct	25	4
Total	**676**	**43**

Fonte: Próprio autor.

3.3.1 RQ1: Como as ferramentas dão apoio à Engenharia de Requisitos?

Conforme análise dos trabalhos, o padrão de suporte à ER segue os interesses dos autores em solucionar problemas específicos, relacionados à ER. A Tabela 2 apresenta um resumo das principais técnicas de ER abordadas pelas ferramentas, fazendo relação aos processos que ela dá suporte.

Através da análise dos trabalhos, fica claro que todas ferramentas implementam uma técnica ou abordagem de acordo com o domínio do problema analisado. Não é explícito o interesse dos autores em cobrir todos os processos relacionados à ER, porém existem ferramentas que dão apoio a todas as etapas desses processos de alguma forma, como no trabalho de Nguyen [17] e Hu [12].

Observa-se que alguns autores implementaram técnicas e utilizaram abordagens semelhantes de acordo com o processo apresentado pela ferramenta. No processo de elicitação, os trabalhos de Kolpondinos [14] e Ribeiro et al. [21] utilizaram funcionalidades relacionadas à gamificação de atividades, para fortalecer o envolvimento e interesse das partes interessadas. Outros autores cobrem em seus trabalhos a realização de sessões de *brainstorming* eletrônico, categorização e organização de ideias e conceitos, como no caso dos trabalhos de Azevedo [5] e Gruenbacher [11]. Também são abordadas técnicas de *workshop* para descobrir os recursos e estabelecer os principais conceitos do sistema nos trabalhos de Usländer [24] e Hu [12].

A descrição de cenários e situações do domínio da aplicação, para levantamento de informações e análise do problema, ocorrem de diferentes maneiras. Pode ser tanto baseada em objetivos e especificação de metas, como no trabalho de Anton [4]; quanto na descrição de casos de uso. como nos trabalhos de Williams [25], Hu [12], Usländer [24] e

Tabela 2 – Áreas da Engenharia de Requisitos com relação às técnicas abordadas.

	Elicitação	**Análise**	**Especificação**	**Validação**	**Técnicas**
[14]	**x**	**x**	**x**		Gamificação; Priorização; Bola de neve; Questionários; Histórias de usuário.
[10]		**x**	**x**		Gerência de projetos; Rastreabilidade.
[25]	**x**	**x**	**x**		Casos de uso; Suporte ontológico; Extração de conceitos Obtenção de requisitos de segurança; Priorização.
[12]	**x**	**x**	**x**	**x**	Casos de uso; Modelos conceituais; *Workshop*; Negociação; Revisão.
[16]		**x**	**x**	**x**	Histórias de usuário; Modelos conceituais; Prototipagem; Casos de teste; Revisão.
[17]	**x**	**x**	**x**	**x**	Gerência de projetos; Gerência de recursos; Modelagem; Wiki.
[21]	**x**	**x**	**x**		Gamificação; Seis chapéus do pensamento; Votação; Grupos Focais.
[5]	**x**	**x**	**x**		*Brainstorming*; Modelos conceituais; Votação.
[24]		**x**	**x**		Especificação semi-formal; Casos de uso; Suporte ontológico; Modelagem; *Workshop*.
[2]		**x**	**x**		Suporte ontológico; Votação; Priorização; Modelagem conceitual;
[18]		**x**	**x**	**x**	Gerência de projetos; Rastreabilidade; Casos de Teste.
[11]	**x**	**x**	**x**		*Brainstorming*; Negociação; Modelagem conceitual; Casos de uso; Extração de conceitos.
[4]		**x**	**x**		Cenários de objetivos; Modelagem de objetivos.

Fonte: Próprio autor.

Gruenbacher [11]; ou histórias de usuário, de acordo com os trabalhos de Kolpondinos [14] e Moketar [16]. Alguns autores fizeram adaptações e simplificações na forma de descrever cenários, tendo em vista facilitar o uso dessas técnicas pelo usuário comum, como nos trabalhos de Hu [12] e Kolpondinos [14].

O processo de análise e especificação de requisitos é abordado de alguma forma por todos os trabalhos relacionados. Existem ferramentas que oferecem suporte semântico à análise dos requisitos, utilizando um sistema ontológico integrado em suas ferramentas, como nos trabalhos de Williams [25], Usländer [24] e Ajmeri [2]. Grande parte das ferramentas permitem a modelagem conceitual dos elementos do domínio da aplicação e suas relações, como nos trabalhos de Hu [12], Moketar [16], Azevedo [5], Ajmeri [2], Gruenbacher [11], Nguyen [17], Usländer [24] e Anton [4]. Existem algumas ferramentas que cobrem o gerenciamento e a rastreabilidade de artefatos, como nos trabalhos de Gho-

frani [10], Nguyen [17] e Ochoa [18], oferecendo também apoio ao acompanhamento de múltiplos projetos e reutilização de artefatos.

A geração e o compartilhamento de artefatos e relatórios para a validação de requisitos são abordados no trabalho de Ochoa [18], Hu [12],Moketar [16] e Nguyen [17], fornecendo, também, apoio através da ferramenta a revisão e acompanhamento das partes interessadas sobre o atendimento aos requisitos. Destaca-se a utilização de mecanismos sociais, implementados nas ferramentas para manter contato com as partes interessadas, e assim favorecer à validação de requisitos.

3.3.2 RQ2: Como as ferramentas permitem a colaboração de pessoas no processo de Engenharia de Requisitos?

A Tabela 3 apresenta como os trabalhos abordam a colaboração em suas ferramentas. Os trabalhos foram agrupados em categorias distintas, de acordo com a forma de colaboração proposta em cada ferramenta.

Tabela 3 – Formas de colaboração através da ferramenta.

Método Colaborativo	**[14]**	**[10]**	**[25]**	[12]	[16]	[17]	[21]	**[5]**	**[24]**	**[2]**	**[18]**	**[11]**	**[4]**
Serviços de mídias sociais.	x					x				x			
Sistema de recomendações.			x							x			
Edição de documentos.	x	x	x	x	x		x				x	x	x
Validação com partes interessadas.					x						x		
Ambiente de modelagem								x	x			x	
Reuso de artefatos.		x									x		
Ambiente de ideação comum.							x					x	

Fonte: Próprio autor.

O método colaborativo mais utilizado é a edição *online* de documentos, abordado por sete das ferramentas analisadas. Nesse contexto, destaca-se o trabalho de Moketar [16], que oferece um ambiente colaborativo integrado ao *Entherpad*, um editor *online* para edição colaborativa em tempo real de documentos. Além da edição, esse recurso oferece suporte a discussões entre clientes e o engenheiro de *software*, possibilitando a participação dos clientes sobre qualquer parte dos resultados.

Algumas ferramentas são baseadas em serviços de mídias sociais, fornecendo assim, um ambiente integrado de vários serviços para colaborações no trabalho em equipe. Dessa forma, todos os usuários se comunicam de forma síncrona e assíncrona usando serviços de rede social da ferramenta. Os serviços de redes sociais incluem postagens de atuali-

zações, comentários de status, curtir uma postagem, marcação, notificação e visualização de conteúdo por meio de *feeds* de notícias e painéis, como nos trabalhos de Kolpondinos [14], Ghofrani [10] e Ajmeri [2].

A elaboração de ideias e representação de suas formas através da diagramação colaborativa permite que os usuários participem simultaneamente de sessões de suporte a essas atividades, como apresentado nos trabalhos de Ribeiro et al. [21] e Gruenbacher [11]. A modelagem colaborativa, utilizada para representar elementos do sistema é abordada nos trabalhos de Azevedo [5], Usländer [24] e Gruenbacher [11].

Nos trabalhos de Williams [25] e Ajmeri [2], são utilizados sistemas de recomendações colaborativos para auxiliar engenheiro de software na obtenção e análise de requisitos. Através de repositórios de conhecimento existentes, o sistema fornece recomendações à medida que o engenheiro vai trabalhando na ferramenta.

Os trabalhos de Ghofrani [10] e Ochoa [18] apresentam ferramentas que permitem a gerência de artefatos. Funcionam como um repositório compartilhado, gerenciado por uma ferramenta. Permite que vários usuários trabalhem em conjunto no gerenciamento de projetos, funcionalidades e artefatos, possibilitando o trabalho colaborativo e reuso desses artefatos. Por fim, alguns trabalhos apresentam ferramentas que permitem a realização de validações com as partes interessadas e coleta de suas opiniões através da ferramenta, como nos trabalhos de Moketar [16] e Ochoa [18].

3.4 Considerações do Capítulo

Neste Capítulo foram relatados os resultados de uma RSL, que teve como objetivo investigar estudos que apresentem uma ferramenta colaborativa para suporte à ER e como ela permite a colaboração de pessoas através do seu uso. Os principais pontos do Capítulo são:

- Através do processo de busca definido, foi possível mapear trabalhos relacionados à presente proposta;
- O protocolo especificado permite a replicação da revisão;
- A análise dos trabalhos possibilitou realizar observações sobre os estudos, de acordo com a sua contribuição para a ER;
- A análise dos trabalhos possibilitou realizar observações sobre os estudos, de acordo com a forma de colaboração abordada em cada ferramenta;
- Através da análise e cruzamento dos dados levantados, foi possível responder às questões de pesquisa;
- Respondendo como as ferramentas dão apoio a ER, foi possível identificar as técnicas mais utilizadas e sua relação com cada processo de ER;

- Respondendo Como as ferramentas permitem a colaboração de pessoas no processo de ER foi possível analisar quais são as principais formas de colaboração abordadas por cada ferramenta;

- A análise dos trabalhos e das ferramentas permitiu não somente responder às questões de pesquisa, como também, mapear as principais funcionalidades de cada ferramenta.

A partir das funcionalidades mapeadas, busca-se, neste trabalho, a elaboração de uma ferramenta que atenda as especifidades da nossa realidade, que é uma ferramenta que permita o suporte à ER de forma colaborativa.

Por mais que algumas ferramentas apresentem funcionalidades interessantes para o presente contexto de problema, não existe uma ferramenta "coringa" que supra todas demandas do problema em questão, ou ainda, que implemente simultaneamente certas funcionalidade que são muito relevantes para o nosso contexto de problema, como o acompanhamento do andamento das tarefas do projeto junto com a elaboração de diagramas, permitindo juntamente a colaboração entre o time de desenvolvimento, entre outras.

Outro fato, é que através da elaboração de uma nova ferramenta pode-se adaptar ou evoluir as funcionalidades de acordo com as necessidades do contexto do problema espera-se, assim, ter um impacto maior na qualidade final da ferramenta e na experiência do usuário ao utilizá-la.

Sendo assim, foi identificado que uma nova ferramenta pode ser desenvolvida para resolver os problemas do contexto deste trabalho, utilizando-se do conhecimento adquirido até o presente momento sobre as outras ferramentas e suas funcionalidades. No próximo Capítulo, é apresentada a metologia utilizada para alcançar os objetivos deste trabalho.

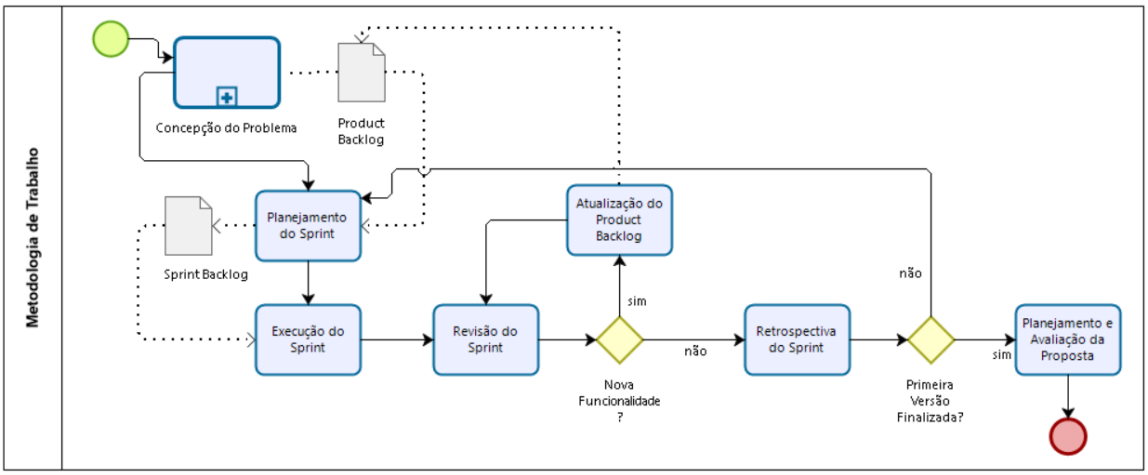
4 METODOLOGIA

Para dar suporte à realização deste trabalho e auxiliar na gestão do projeto, foi definida uma metodologia baseada no *framework Scrum* [22]. Dessa forma, foram adaptados alguns papéis do *Scrum* para serem incorporados à presente metodologia. Os papéis são os seguintes:

- *Product Owner*: esse papel é assumido pelo próprio desenvolvedor. Segundo Costa [7], não é indicado que o próprio desenvolvedor faça a priorização dos requisitos. Assim, essa atividade foi realizada em conjunto com as partes interessadas.
- *Scrum Master*: esse papel também é assumido pelo desenvolvedor. O desenvolvedor precisa ter pleno entendimento do processo que está utilizando e segui-lo corretamente. Assim, precisa fiscalizar a si mesmo e garantir que sejam seguidas as práticas definidas.
- Time *Scrum*: o desenvolvedor assume também o papel de Time *Scrum*. É responsável pelo produto construído, por utilizar o *Scrum* corretamente e por desenvolver seu produto.

De acordo com o conjunto de práticas presentes no *framework Scrum*, foram definidas as etapas que iriam compor a presente metodologia. A Figura 5 apresenta a metodologia adotada para alcançar os objetivos deste trabalho. As seções a seguir apresentam a descrição de cada uma das etapas que compõem a metodologia.

Figura 5 – Processo metodológico do trabalho.



Fonte: Próprio autor.

4.1 Concepção da Ferramenta

Esta etapa contempla as seguintes atividades:

- Análise de ferramentas similares: nesta atividade são analisadas as funcionalidades das ferramentas colaborativas e as contribuições para apoiar a ER. Para isso, são utilizadas como base as ferramentas identificadas durante a RSL. A partir dessa análise são elicitadas e especificadas as principais características de cada ferramenta.

- Classificação das funcionalidades: nesta atividade, cada funcionalidade elicitada é classificada como essencial, importante ou desejável, pelo desenvolvedor, considerando a presente proposta. Uma funcionalidade essencial é aquela que é identificada como realmente fundamental para o sistema, sem o qual o sistema não pode ser dado como "completo", ou seja, são requisitos que, se não são implementados, impedem uma implantação ou a conclusão do sistema. Uma funcionalidade classificada como importante é aquela que é identificada como fundamental para o sistema, porém sua implementação pode ser postergada. Por sua vez, uma funcionalidade desejável é um requisito que não é indispensável para o sistema estar completo ou para entrar em produção. Também não é algo que, mesmo postergado, deverá ser feito, ou seja, sem um requisito desejável o sistema deve funcionar de maneira satisfatória, atendendo completamente seu objetivo. Destaca-se que, neste trabalho, essa classificação serve apenas como base para a elaboração das histórias de usuário.

- Criação das Histórias de Usuário: nesta atividade é elaborado o conjunto inicial de histórias de usuário do sistema pelo desenvolvedor, bem como, realizada a classificação dessas histórias junto às partes interessadas. Para realizar a classificação dois *stakeholders* priorizam as histórias de usuário em uma única reunião através de um discussão informal, as histórias são classificadas em alta, média e baixa prioridade em relação a percepção deles. Neste trabalho, essas histórias representam o *Product Backlog* do sistema.

- Elaboração do modelo de domínio: nesta atividade é elaborado um modelo de domínio inicial, a partir das histórias de usuário especificadas.

4.2 Planejamento do *Sprint*

A etapa de Planejamento de *Sprint* corresponde à preparação do *Sprint*. Nessa atividade, o desenvolvedor deve responder as seguintes questões:

- O que pode ser entregue como resultado do próximo *Sprint*?

- Como o trabalho necessário para entregar o incremento será realizado?

Essa atividade se repete antes do início de cada *Sprint*.

4.3 Execução do *Sprint*

Nesta etapa são realizadas as atividades de prototipagem e validação da interface da ferramenta. Também são realizadas as atividade de projeto, desenvolvimento e teste.

4.4 Revisão do *Sprint*

A Revisão do *Sprint* ocorre sempre que o desenvolvedor finalizar um *Sprint*. Se tudo estiver correto, o requisito é dado como finalizado, caso contrário ele retorna para o *product backlog* para que seja corrigido o problema.

4.5 Atualização do *Product Backlog*

Nesta etapa, é verificada a ocorrência de novos requisitos e, caso seja confirmada a ocorrência de novos requisitos, o *Product Backlog* pode ser atualizado com novos requisitos durante a execução do projeto.

4.6 Retrospectiva do *Sprint*

Nesta etapa, o desenvolvedor pode parar para analisar o trabalho que foi feito, além de verificar os possíveis problemas encontrados durante o desenvolvimento e corrigir para o próximo *Sprint*. Caso sejam encontrados problemas, estes devem ser anotados, para que no próximo *Sprint* o desenvolvedor possa corrigí-los.

4.7 Planejamento da Avaliação da Proposta

Nesta etapa, é realizado o planejamento da avaliação da ferramenta, em que são definidos quais os atributos de qualidade serão utilizados na avaliação.

5 DESENVOLVIMENTO

Neste Capítulo é apresentado como foi executado o processo de desenvolvimento da presente ferramenta, desde como surgiu a concepção até o último *sprint* executado.

5.1 Concepção da Ferramenta

Esta fase envolve os primeiros passos do projeto, incluindo a definição das questões fundamentais para a realização deste trabalho.

5.1.1 Descrição do Domínio do Problema

O programa de extensão Programa C - Comunidade, Computação, Cultura, Comunicação, Ciência, Cidadania, Criatividade, Colaboração da Universidade Federal do Pampa adota em sua atividade Resolve! uma metodologia de resolução de problemas, em que os estudantes dos cursos Ciência da Computação e Engenharia de Software são organizados em equipes para resolver problemas da comunidade local ou regional.

Ao desenvolver a solução de um problema real, a equipe, além de ter que gerenciar um grande número de ferramentas e artefatos, necessita realizar a ER, em colaboração com as partes interessadas na solução de um problema, o que envolve, por exemplo, entrevistas, priorização dos requisitos e validação com as partes interessadas.

Dessa forma, é de interesse das pessoas envolvidas no programa o desenvolvimento de um sistema que dê suporte ao processo de ER. Além disso, esse sistema deve permitir a visualização e o acompanhamento dos problemas pelas partes interessadas. Neste estudo de caso em particular, são partes interessadas os extensionistas (acadêmicos e docentes) vinculados ao Programa C e pessoas da comunidade interessadas na solução dos problemas.

Embora o estudo de caso seja conduzido no programa de extensão Programa C, a presente proposta não se restringe a este escopo, podendo ser utilizada em diferentes domínios que se beneficiem da, especificação e acompanhamento de ideias, artefatos e problemas.

5.1.2 Análise de Ferramentas Similares

Seguindo a metodologia descrita no Capítulo anterior, como etapa inicial da concepção da ferramenta, foi feita uma análise de ferramentas similares e que pudessem contribuir com a presente proposta. Nesta atividade, foram analisadas as funcionalidades de ferramentas colaborativas e suas contribuições para suporte à ER. Para isso, foram utilizadas como base as ferramentas identificadas durante a RSL. A partir dessa análise foram elicitadas e especificadas as principais características de cada ferramenta colaborativa com relação à Engenharia de Requisitos.

5.1.3 Classificação das Funcionalidades Principais

Durante a revisão sistemática de literatura, foram mapeadas diversas ferramentas de suporte à ER. Essas foram analisadas e tiveram suas funcionalidades tabuladas, para que assim fosse possível realizar a classificação de suas funcionalidades. Conforme relatado no capítulo anterior, as funcionalidades foram classificadas como essenciais, importantes ou desejáveis, em relação aos objetivos da presente proposta.

Como um grande número de funcionalidades foram mapeadas, a Tabela 4 apresenta apenas as funcionalidades classificadas como essenciais (E) ou importantes (I). A Tabela completa está disponível no Apêndice A.

Tabela 4 – Funcionalidades extraídas das ferramentas analisadas.

	Principais Funcionalidades	**E**	**I**
01	Publicar um requisito, para que outros usuários possam ver.	x	
02	Avaliar os requisitos publicados por outros usuários.	x	
03	Priorizar os requisitos.	x	
04	Realizar a rastreabilidade entre projetos, artefatos e requisitos.	x	
05	Fazer upload de artefatos como vídeo, áudio, texto, imagem ou arquivos XML.	x	
06	Discussão sobre os requisitos em busca da validação do cliente, através da ferramenta.		x
07	Permitir que o engenheiro de software inicialize uma discussão com o cliente através da ferramenta.		x
08	Permitir que os usuários realizem comentários, em relação aos resultados em discução.		x
09	Permitir que várias partes interessadas validem o mesmo conjunto de requisitos.		x
10	Disponibilizar espaços de trabalho compartilhado.		x
11	Definir tarefas.	x	
12	Permitir que usuários possam visualizar a lista de requisitos que foram enviados pelos outros usuários.	x	
13	Controle de simultaneidade, na edição de artefatos.	x	
14	Criar projetos.	x	
15	Fornece concessões de acesso aos usuários.	x	
16	Especificação de vínculo entre pessoas e requisitos.	x	
17	Referenciar protótipos desenvolvidos a artefatos e requisitos.	x	
18	Visualização do status da evolução do projeto.	x	
19	Permitir que partes interessadas possam acompanhar o andamento do projeto.	x	
20	Geração e organização ideias ou conceitos.		x
21	Votação distribuída dos requisitos.	x	
22	Registrar histórico de informações sobre a reunião.	x	

Fonte: Próprio autor.

5.1.4 Definição das Histórias de Usuário

Com base na análise das funcionalidades de outras ferramentas (Tabela 4), foi possível definir com maior clareza as histórias de usuário que o sistema deve contemplar. Essas histórias foram validadas e priorizadas junto às coordenadoras do programa de extensão Programa C. A prioridade foi categorizada em três níveis: Alta, Média e Baixa. Essa priorização serviu para representar quais funcionalidades têm a preferência para serem desenvolvidas, conforme a opinião das partes interessadas. A Tabela 5 apresenta as histórias organizadas por nível de prioridade.

Tabela 5 – Histórias de Usuário.

	Descrição	Prioridade
US01	Como analista eu quero poder definir um requisito.	Alta
US02	Como analista eu quero poder visualizar quais requisitos, artefatos e pessoas estão relacionadas a um projeto.	Alta
US03	Como analista eu quero poder criar um novo projeto.	Alta
US04	Como analista eu quero poder visualizar qual requisito deu origem ao artefato que eu estou trabalhando.	Alta
US05	Como colaborador eu quero poder representar e manter minhas ideias em relação a um problema, em diferentes tipos de mídias.	Alta
US06	Como colaborador eu quero poder visualizar o status e evolução do projeto.	Alta
US07	Como analista eu quero poder compartilhar os requisitos com os demais membros do grupo.	Média
US08	Como analista eu quero poder exportar documentos a partir dos requisitos especificados.	Média
US09	Como analista eu quero poder associar um artefato a um requisito.	Média
US10	Como analista eu quero poder visualizar quais pessoas contribuíram para a origem de um requisito.	Média
US11	Como colaborador eu quero poder compartilhar minhas ideias com os demais membros do grupo.	Média
US12	Como colaborador eu quero poder realizar comentários sobre os artefatos gerados.	Média
US13	Como colaborador eu quero priorizar os requisitos, para o posterior desenvolvimento.	Média
US14	Como administrador, eu quero poder cadastrar usuários.	Média
US15	Como administrador, eu quero poder dar concessões a usuários.	Média
US16	Como analista eu quero definir as tarefas que serão realizadas para alcançar o sucesso dos requisitos.	Baixa
US17	Como analista eu quero iniciar uma discussão como o cliente através da ferramenta, para fins de validação.	Baixa
US18	Como analista eu quero poder me comunicar com o cliente de forma direta.	Baixa
US19	Como analista eu quero poder visualizar o histórico de contribuição dos usuários.	Baixa
US20	Como colaborador eu quero avaliar os requisitos publicados por outros colaboradores, para classificá-los.	Baixa

Fonte: Próprio autor.

A reunião com as partes interessadas para a validação e priorização das histórias de usuário permitiu uma melhor visualização do sistema. Assim, algumas funcionalidades (US11, US19 e US20), que em um primeiro momento foram classificadas como essenciais, foram identificadas com prioridade média ou baixa pelas partes interessadas.

Para representar os atores interessados no problema a ser resolvido (usuários finais), foram utilizados os seguintes papéis:

- Administrador: representa o papel dos professores e estudantes com alto envolvimento no projeto;
- Analista: representa o papel de todos os estudantes e professores envolvidos no projeto;
- Colaborador: representa o papel dos membros da comunidade interessados na solução do problema. Estudantes e professores também podem incorporar esse papel.

Destaca-se, neste trabalho, que os papéis são definidos para fins de análise do cenário do problema, logo, não devem ser confundidos com níveis de autenticação dos usuários ou permissões relacionadas ao sistema. Essas definições são pertencentes ao domínio da solução, não do problema.

Além da definição e priorização das histórias de usuário, elas foram agrupadas de acordo com o processo relacionado à ER, conforme apresentado na Tabela 6. Destaca-se que algumas histórias de usuário estão relacionadas às áreas de gerenciamento de projeto e gerenciamento requisitos, por isso não são mostradas nessa Tabela. São elas: US02, US03, US06, US09, US10, US14, US15 e US19.

Tabela 6 – Relação entre as histórias de usuário e os processos da ER.

Processo ER	História de Usuário
Elicitação	US05, US11
Análise	US04, US07, US13
Especificação	US01, US08, US16
Validação	US12, US17, US18, US20

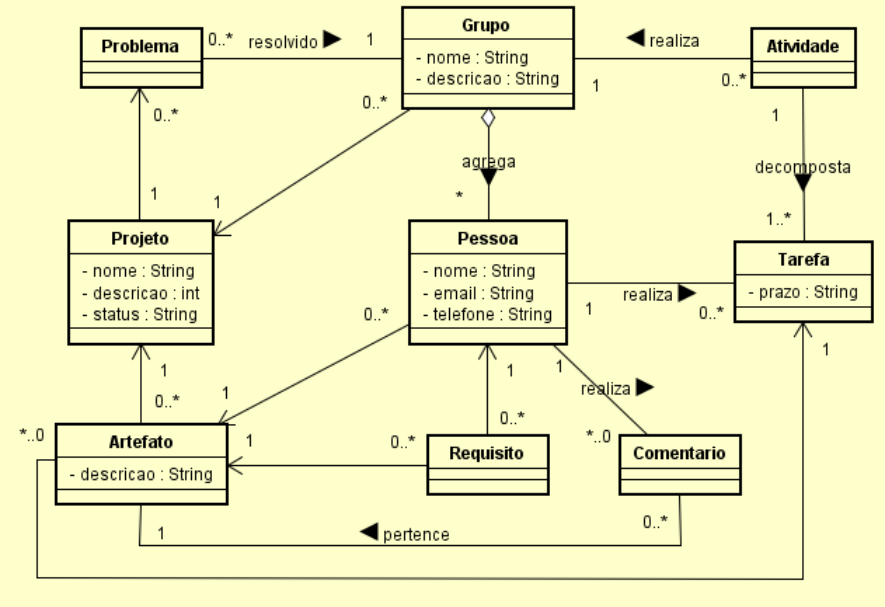
Fonte: Próprio autor.

A partir da priorização das histórias de usuário e discussão junto às coordenadoras do programa de extensão Programa C, foi definido que a primeira versão da ferramenta deveria ter o foco na análise e especificação de requisitos, artefatos e problemas. Esses processos foram escolhidos porque foram considerados importantes para uma primeira versão da ferramenta, servindo como base para uma futura evolução da ferramenta. Busca-se, dessa forma, construir a ferramenta com uma estrutura planejada, para que em futuras versões ela possa ser evoluída e cobrir mais processos e funcionalidades relacionadas a ER.

5.1.5 Modelagem de Domínio do Sistema

A análise de domínio está relacionada à descoberta de informações que são gerenciadas pelo sistema. Nessa fase de concepção do problema, foi realizado um modelo conceitual preliminar. Esse modelo foi refinado e complementado nas fases posteriores de desenvolvimento. A Figura 6 apresenta o modelo proposto, cujas classes que o compõem representam o seguinte:

Figura 6 – Modelo de domínio.



Fonte: Próprio autor.

- A classe Pessoa representa uma pessoa envolvida na solução do problema;
- A classe Grupo representa o grupo de pessoas envolvidas em um problema;
- A classe Problema representa um problema que deve ser resolvido pelo grupo;
- A classe Projeto representa o projeto no qual o grupo está colaborando;
- A classe Comentário representa um comentário que pode ser feito a respeito de um artefato;
- A classe Artefato representa um artefato relacionado ao projeto;
- A classe Atividade representa uma atividade que deve ser realizada pelo grupo;
- A classe Tarefa representa a parte de uma atividade que deve ser realizada pelas pessoas envolvidas no projeto.

5.2 Desenvolvimento da Solução

Para o desenvolvimento da solução foram realizados três *sprints*. Os *sprints* são descritos nas subseções seguintes.

5.2.1 Sprint 1

A partir das histórias de usuário definidas durante a fase de Concepção da Ferramenta e, uma previsão do tempo estimado para desenvolvê-las, foi pensado no *backlog* do primeiro *sprint*. Esse *backlog* foi definido durante a fase de Planejamento do *sprint* em questão.

Após ser feita a definição das histórias de usuário que iriam compor o *sprint*, foram definidos quais artefatos de projeto deveriam ser produzidos nesse primeiro momento, para dar auxilio ao desenvolvimento das respectivas funcionalidades. Dessa forma, foi definido que o modelo arquitetural da ferramenta deveria ser pensado e elaborado durante a execução do primeiro *sprint*. Também, foi definido que seria feita a escolha do tema do painel principal da ferramenta e protótipos para visualizar e organizar, quando necessário, a disposição dos componentes em tela.

O *backlog* do primeiro *sprint* é composto pelas seguintes histórias de usuário:

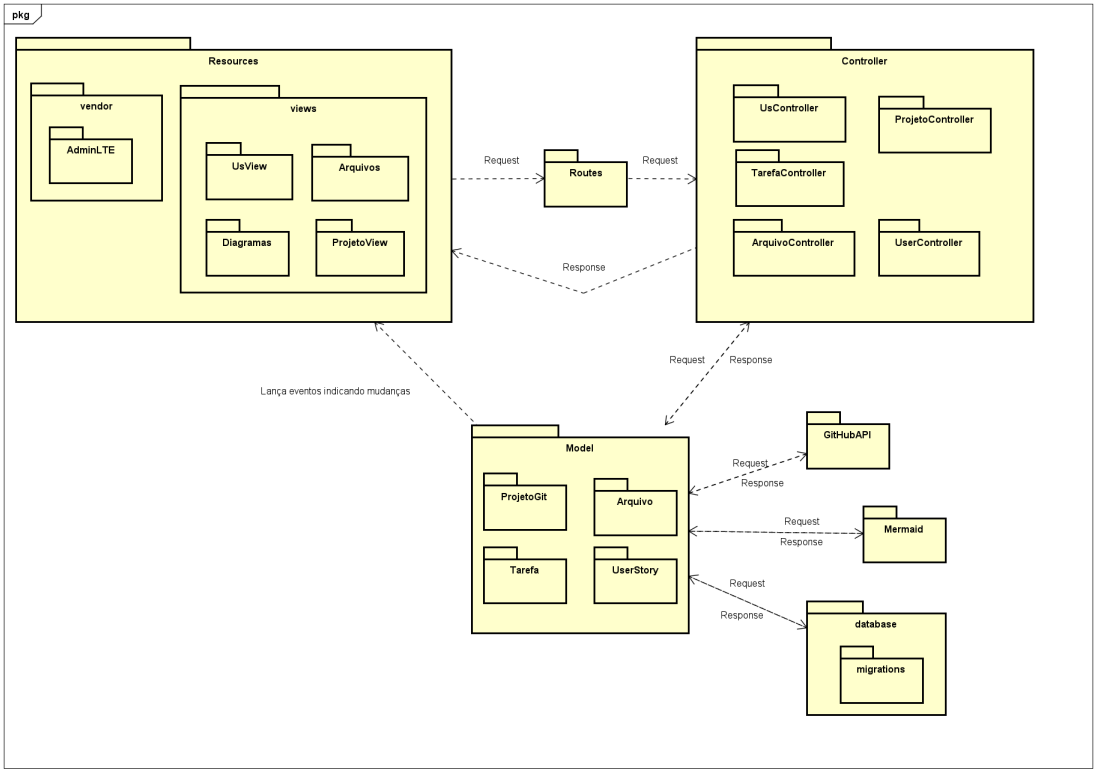
- US01 - Como analista eu quero poder definir um requisito;
- US02 - Como analista eu quero poder visualizar quais requisitos, artefatos e pessoas estão relacionadas a um projeto;
- US03 - Como analista eu quero poder criar um novo projeto;
- US07 - Como analista eu quero poder compartilhar os requisitos com os demais membros do grupo.

Levado em consideração o *framework* escolhido para apoiar o desenvolvimento, o padrão arquitetural foi definido como MVC (*Model-View-Controller*). Dessa forma, a camada *View* fica responsável pela *interface* que é apresentada para o usuário. A camada *Controller* fica responsável por receber as requisições dos usuários, utilizar a camada *Model* para obter os dados, e também, responsável por renderizar as saídas de informações utilizando as *Views*. Por fim, a camada *Model* fica responsável pelo acesso e manipulação dos dados da aplicação, nela estarão, por exemplo, as funções de consulta ao banco de dados ou de acesso a alguma *Application Programming Interface* (API) de uma aplicação externa.

Para dar apoio ao gerenciamento dos projetos que seriam trabalhados dentro da ferramenta, foi decidido, com base nas informações coletadas através dos trabalhos relacionados (seção 3), que a ferramenta seria integrada com a GitHub API. Dessa forma,

durante o desenvolvimento arquitetural, também foi planejado e estudado como deveriam ser feitas as requisições da ferramenta para a Github API. Como base para esse estudo, foi utilizada a documentação da GitHub API [1]. A Figura 7 apresenta o modelo arquitetural da ferramenta.

Figura 7 – Modelo arquitetural da Aplicação.

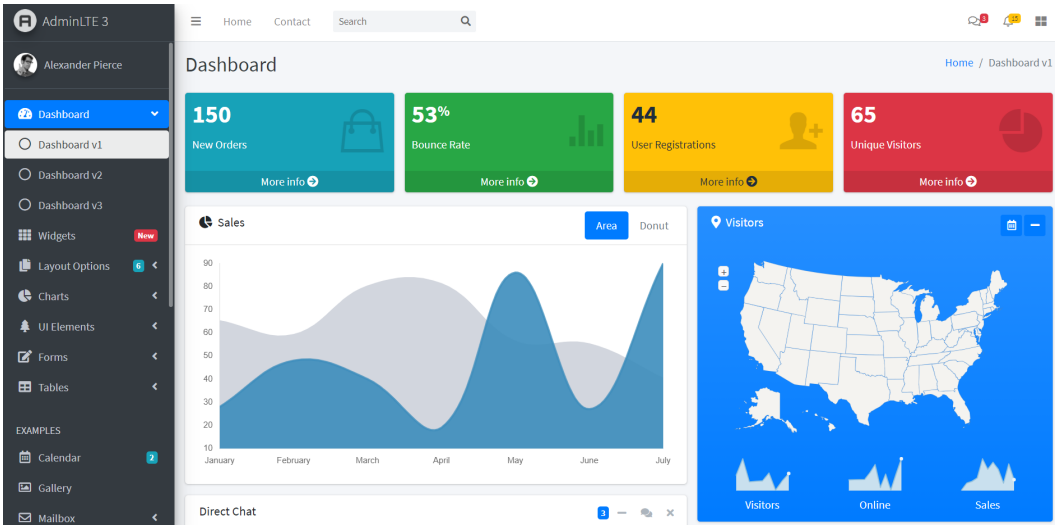


Fonte: Próprio autor.

Durante a escolha do tema para o painel principal da ferramenta, buscou-se encontrar um tema específico para painéis de controle, visto a natureza da ferramenta proposta. Dessa forma, optou-se pelo tema AdminLTE, por ser um modelo de código aberto, popular para aplicações *web* e possuir componentes específicos para painéis de controle e administração. O AdminLTE é baseado em um *design* modular e utiliza todos os componentes do Bootstrap em seu *design*. Para realizar a integração do AdminLTE com a ferramenta proposta utilizou-se como base a documentação do AdminLTE [2]. A Figura 8 apresenta um dos painéis de gerenciamento, com alguns dos componentes disponibilizados pelo AdminLTE.

[1] https://docs.github.com/en/free-pro-team@latest/rest
[2] https://adminlte.io/docs/3.0/

Figura 8 – Exemplo de *dashboard* disponibilizado pelo AdminLTE.



Fonte: Próprio autor.

Para a ferramenta poder dar suporte à definição de requisitos, foi definido que seria utilizado o padrão de escrita de histórias de usuário. Optou-se por esse padrão, por ele estar presente em alguns dos trabalhos analisados (seção 3), ser utilizado com metodologias ágeis e utilizar um formato de linguagem não técnica para dar contexto à equipe de desenvolvimento sobre a funcionalidade de software a ser desenvolvida. Na presente ferramenta, elas também são aplicadas como blocos maiores que devem ser quebrados em tarefas mais específicas, esses tarefas mais específicas são definidas na ferramenta desenvolvida, como tarefas das histórias de usuário.

As funcionalidades obtidas ao término do primeiro *sprint* foram:

- *Create, Read, Update and Delete* (CRUD) de dados dos projetos integrados com a GitHub API;
- Busca de dados dos colaboradores dos projetos através da GitHub API;
- Autenticação de usuários da ferramenta integrada com o GitHub (*login* de rede social);
- CRUD de histórias de usuário.

Após as funcionalidades passarem por uma bateria de testes funcionais e de integração com a GitHub API, foi feita uma reunião com as partes interessadas para validar as funcionalidades desenvolvidas e identificar possíveis melhorias na ferramenta, bem como, realizar o planejamento do segundo *sprint*.

5.2.2 Sprint 2

Para o segundo *sprint* foram definidas como *backlog* as seguintes histórias de usuário:

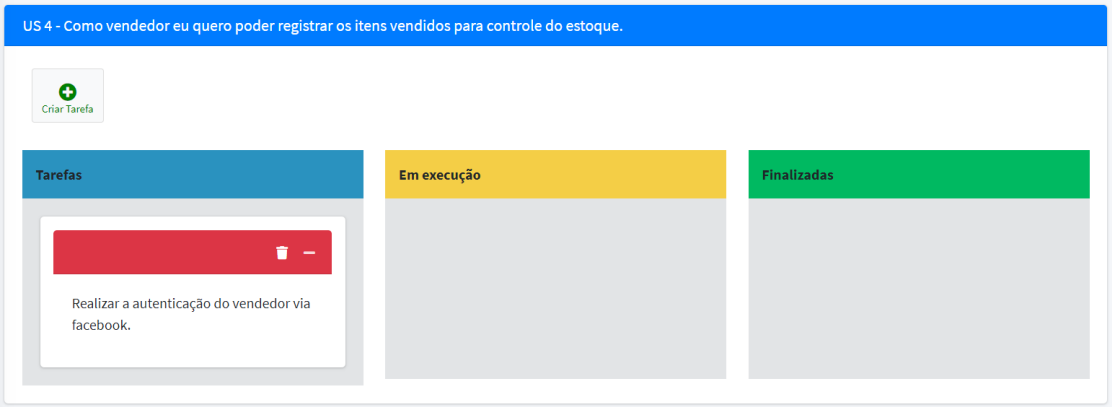
- US06 - Como colaborador eu quero poder visualizar o *status* e evolução do projeto;
- US13 - Como colaborador eu quero priorizar os requisitos, para o posterior desenvolvimento;
- US16 - Como analista eu quero definir as tarefas que serão realizadas para alcançar o sucesso dos requisitos.

Durante a reunião de planejamento do segundo *sprint*, foi definido que, para a ferramenta atender as histórias de usuário definidas no *backlog* do segundo *sprint*, seria ideal a implementação de um *kanban* para controlar os fluxos de produção, organizar as tarefas das histórias de usuário, visualizar a evolução do projeto, priorizá-las e verificar as tarefas das histórias de usuário especificadas pelos colaboradores.

Foi definido durante o segundo *sprint* que a estrutura do *kanban* seria composta por três raias: a raia "Tarefas", que estariam as tarefas da história de usuário previamente especificadas e associadas a um colaborador encarregado por realizar a tarefa; a raia "Em execução", com as tarefas em desenvolvimento e a raia "Finalizadas", com as tarefas já desenvolvidas pelos colaboradores.

Para o desenvolvimento do *kanban* foi feita uma pesquisa sobre bibliotecas que pudessem contribuir de alguma forma com o desenvolvimento dessa funcionalidade. Dessa forma, foi escolhida a biblioteca Jkanban [3] para auxiliar no desenvolvimento dessa funcionalidade, por atender ao objetivo da funcionalidade proposta e ser possível realizar sua integração ao código da presente ferramenta. A Figura 9 apresenta a estrutura base do *kanban* desenvolvido durante o segundo *sprint*.

Figura 9 – Estrutura do Kanban.



Fonte: Próprio autor.

[3] https://github.com/riktar/jkanban

As funcionalidades obtidas ao término do segundo *sprint* foram:

- CRUD de tarefas das histórias de usuário;
- Visualização do andamento do projeto pelos colaboradores do projeto;
- Priorização das histórias de usuário pelos colaboradores do projeto;
- Organização das tarefas que serão desenvolvidas para atender aos requisitos.

Após as funcionalidades passarem por uma bateria de testes funcionais, foi realizada uma reunião para validar as funcionalidades desenvolvidas e identificar possíveis melhorias na ferramenta, bem como, realizar o planejamento do terceiro *sprint*.

5.2.3 Sprint 3

Para o terceiro *sprint*, foram definidas como itens do *backlog* as seguintes histórias de usuário:

- US04 - Como analista eu quero poder visualizar qual requisito deu origem ao artefato que eu estou trabalhando;
- US08 - Como analista eu quero poder exportar documentos a partir dos requisitos especificados;
- US09 - Como analista eu quero poder associar um artefato a um requisito.

Durante a reunião de planejamento do terceiro *sprint*, foi definido que, para a ferramenta atender às histórias de usuário definidas no *backlog* do terceiro *sprint*, seria ideal implementar uma forma de produzir artefatos de projeto ou análise através da ferramenta. Esses artefatos poderiam ser tanto associados aos projetos quanto a um requisito específico.

Dessa forma, durante o terceiro *sprint*, foi realizada uma pesquisa sobre tecnologias que pudessem contribuir de alguma forma com o desenvolvimento da funcionalidade em questão. Assim, foi escolhida a ferramenta Mermaid para ser integrada à presente ferramenta. A Mermaid é uma ferramenta de diagramação e gráficos baseada em Javascript que usa definições de texto inspiradas em *Markdown* e um renderizador para criar e modificar diagramas. Para realizar a integração com a Mermaid, usou-se como base a documentação da ferramenta [4].

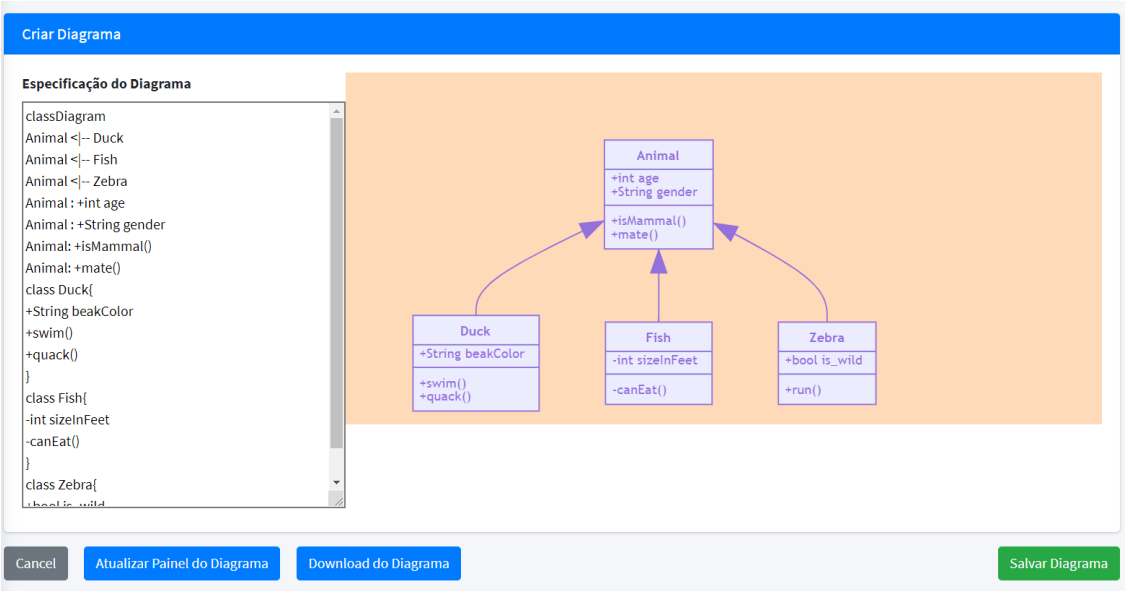
Através da integração com da Mermaid API com a ferramenta proposta, é possível chamar uma função de renderização passando por parâmetro uma *string* que contém a definição do digrama feito pelo usuário da ferramenta proposta. A função render,

[4] https://mermaid-js.github.io/

renderizará o diagrama e chamará um retorno de chamada com o código SVG (*Scalable Vector Graphics*) resultante.

A Figura 10 apresenta a estrutura de um diagrama feito dentro da ferramenta proposta, utilizando a Mermaid API.

Figura 10 – Ferramenta integrada com a Mermaid API.



Fonte: Próprio autor.

A Figura 11 apresenta o painel para visualização e seleção de artefatos relacionados ao projeto.

Figura 11 – Painel para visualização e seleção de artefatos do projeto.

Fonte: Próprio autor.

As funcionalidades obtidas ao término do terceiro *sprint* foram:

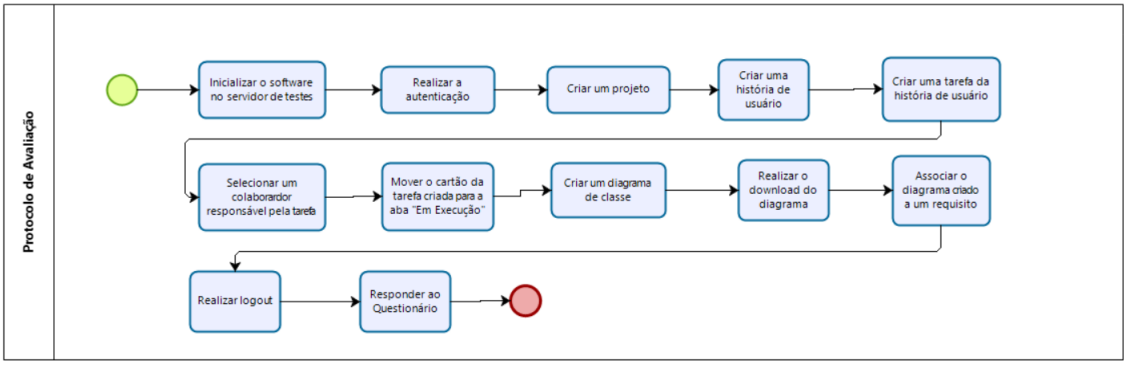
- CRUD de artefatos relacionados ao projeto;
- Associação de artefatos do projeto a requisitos;
- Exportação de diagramas no formato de imagem;
- Possibilidade de criação de todos os tipos de diagramas disponíveis na Mermaid API, através da ferramenta proposta.

Após as funcionalidades passarem por uma bateria de testes funcionais, foi feita uma reunião para validar as funcionalidades desenvolvidas e identificar possíveis melhorias na ferramenta. Ao finalizar os ajustes necessários para completar os incrementos, o terceiro *sprint* foi dado como finalizado e, posteriormente, foi dado início à fase de planejamento da avaliação da presente proposta.

5.3 Planejamento da Avaliação da Proposta

Para realizar a avaliação da presente proposta, foi elaborado um protocolo de teste, o qual foi dividido em etapas para facilitar a compreensão das tarefas executadas pelos participantes da avaliação. A Figura 12 apresenta o fluxo de atividades do protocolo de avaliação.

Figura 12 – Protocolo de avaliação da presente proposta.



Fonte: Próprio autor.

O usuário do sistema deve seguir um fluxo de tarefas pré-determinado, elaborado com base nas principais funcionalidades da ferramenta. Esse fluxo de tarefas foi disponibilizado para o usuário e pode ser consultado no Apêndice B. Após a execução do fluxo de tarefas do protocolo de avaliação, o usuário encerra a atividade com o preenchimento de dois questionários, que visam coletar suas percepções sobre os atributos de qualidade externa e uso do sistema.

O primeiro questionário foi elaborado conforme os critérios de qualidade contidos na ISO/IEC 25000, Apêndice C, que trata da qualidade de produtos de software. A

Tabela 7 apresenta o critério de qualidade avaliado e as perguntas relacionadas a cada critério.

Tabela 7 – Critérios de Qualidade.

Critério de Qualidade	Questão Relacionada
C1 - Adequação Funcional	Q1 - O sistema fornece os resultados corretos para cada funcionalidade executada? Q2 - O sistema cumpre bem sua função, de acordo com a proposta?
C2 - Eficiência de Desempenho	Q3 - O tempo de resposta ao efetuar as operações é rápido?
C3 - Confiabilidade	Q4 - O sistema é instável durante o uso? Q5 - O sistema pode funcionar mesmo quando há alguma falha?
C4 - Segurança	Q6 - O sistema evita acesso não autorizado?
C5 - Usabilidade	Q7 - Você reconhece o sistema como adequado para a finalidade proposta? Q8 - Você considera fácil aprender como o sistema funciona? Q9 - É possível entender o modelo de interface proposto pela aplicação? Q10 - O sistema pode substituir outro produto de software com a mesma finalidade? Q11 - A aplicação é fácil de ser operada? Q12 - A interface do usuário é agradável?

Fonte: Próprio autor.

O segundo questionário, Apêndice D, tem o objetivo de mapear possíveis falhas no sistema. Caso alguma falha ocorresse durante a execução das tarefas, o usuário poderia expressar essa falha indicando a tarefa na qual ela ocorreu, e assim, ajudar a mapear falhas ou erros inesperados no sistema.

6 RESULTADOS E DISCUSSÃO

Neste Capítulo, são apresentados os resultados obtidos. Na Seção 6.1 é apresentada a ferramenta desenvolvida. Na Seção 6.2, são descritos os resultados obtidos com a avaliação feita através do uso da ferramenta por outras pessoas. Por fim, na Seção 6.3, são apresentadas as principais considerações do Capítulo.

6.1 Ferramenta

O principal resultado gerado através deste trabalho foi a construção da ferramenta Clover [1] (palavra na língua inglesa que significa trevo em português). Esse nome foi escolhido porque um trevo possui as folhas conectadas em um único ponto, representando a ideia por traz desse trabalho: centralizar várias tecnologias em uma única solução. A Tabela 8mostra as histórias de usuário que foram concluídas no decorrer deste trabalho.

Tabela 8 – Histórias de Usuário Concluídas.

	História de Usuário
US01	Como Analista eu quero poder definir um requisito.
US02	Como Analista eu quero poder visualizar quais requisitos, artefatos e pessoas estão relacionadas a um projeto.
US03	Como Analista eu quero poder criar um novo projeto.
US04	Como Analista eu quero poder visualizar qual requisito deu origem ao artefato que estou trabalhando.
US06	Como Colaborador eu quero poder visualizar o status e evolução do projeto.
US07	Como Analista eu quero poder compartilhar os requisitos com os demais membros do grupo.
US08	Como Analista eu quero poder exportar documentos a partir dos requisitos especificados.
US09	Como Analista eu quero poder associar um artefato a um requisito.
US13	Como Colaborador eu quero poder priorizar os requisitos para o posterior desenvolvimento.
US16	Como Analista eu quero definir as tarefas que serão realizadas para alcançar os sucessos dos requisitos.

Fonte: Próprio autor.

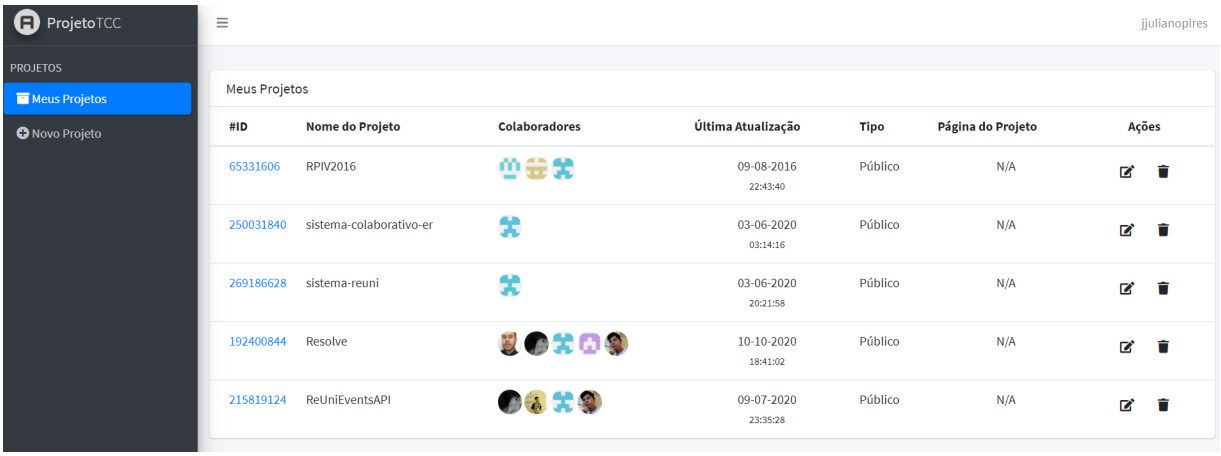
Na versão atual da ferramenta, o usuário pode criar projetos, especificar requisitos utilizando a narrativa de histórias de usuário e definir tarefas para os demais membros do projeto. A ferramenta também disponibiliza um editor para a criação de diagramas

[1] https://github.com/jjulianopires/SistemaClover

como: diagramas de estados, diagramas de fluxo e diagramas de classe. Dessa forma, o usuário pode produzir diversos tipos de diagramas mantendo a relação dos dados entre requisitos, projetos, tarefas e artefatos e também realizar o download desses artefatos através da ferramenta.

Por ser integrada com a GitHub API, a ferramenta permite a colaboração de múltiplos usuários no desenvolvimento de projetos. Através do login social integrado com o GitHub, os usuários da ferramenta são autenticados e seus projetos são listados através de consultas feitas para a GitHub API. A Figura 13 apresenta os projetos do usuário que está autenticado na ferramenta. Na imagem ainda é possível visualizar os dados mais relevantes do projeto, o avatar de cada integrante do projeto, bem como as ações de editar e remover algum projeto da lista. Além dessas funcionalidades, a ferramenta permite a criação de projetos integrados com GitHub API.

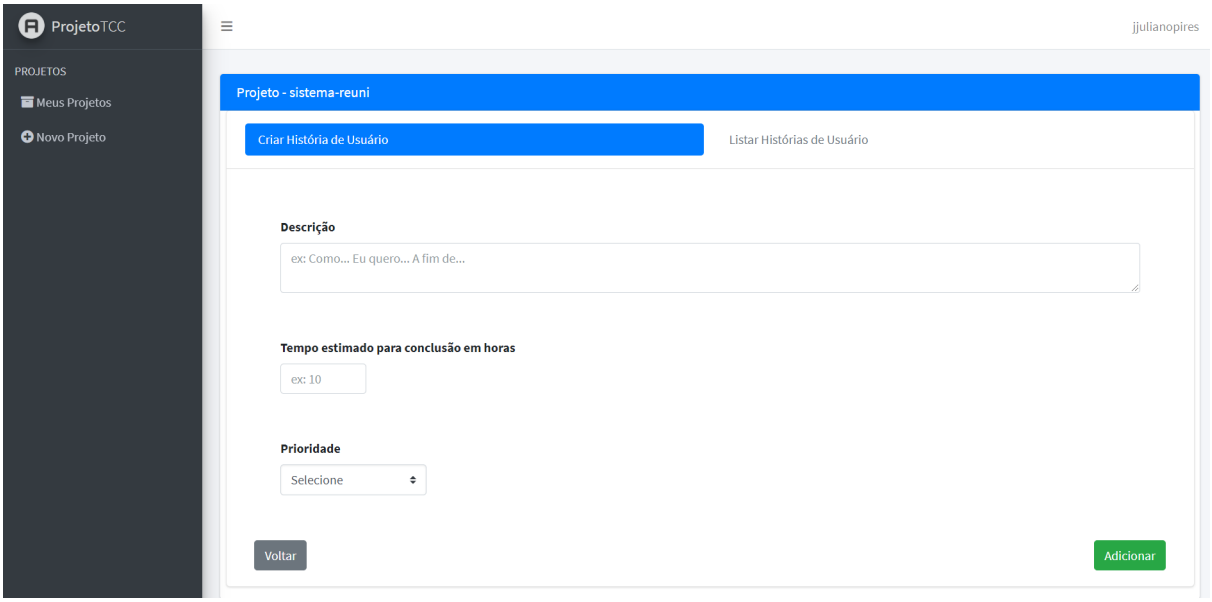
Figura 13 – Tela de Listagem de Projetos.



Fonte: Próprio autor.

Como o objetivo da ferramenta é dar suporte a ER, a ferramenta permite a criação de histórias de usuário através de uma estrutura pré-definida na qual o usuário pode marcar o tempo estimado para a conclusão da história. Dessa forma, além de manter o registro dos requisitos, eles podem ser compartilhados com os demais colaboradores do projeto. A Figura 14 apresenta a tela de criação de uma história de usuário com um exemplo de estrutura que o usuário pode seguir para escrevê-la.

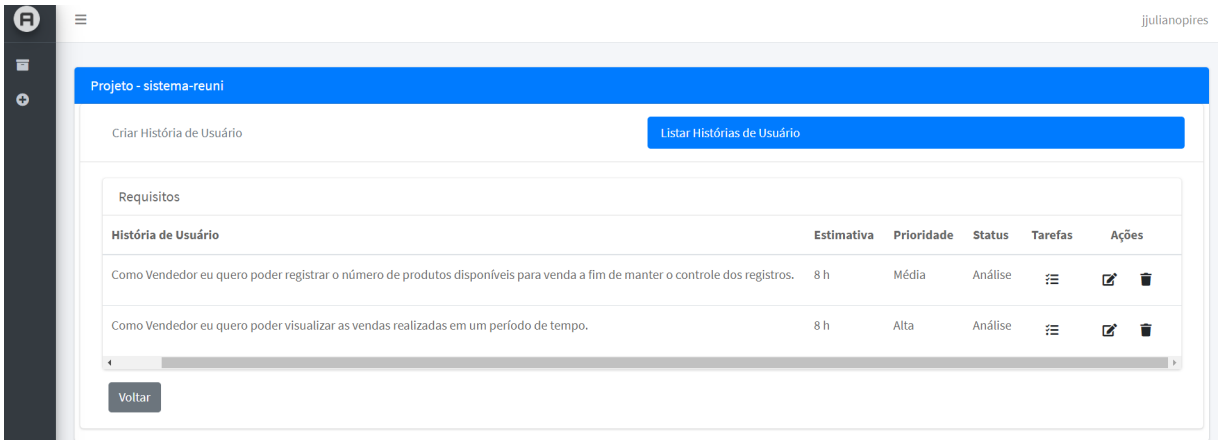
Figura 14 – Tela de Criação de Histórias de Usuário.



Fonte: Próprio autor.

A Figura 15 apresenta a tela na qual as histórias de usuário são listadas e seus dados podem ser visualizados, alterados ou removidos. Nessa imagem ainda é possível visualizar a opção de definir as tarefas que serão associadas a cada uma das histórias de usuário.

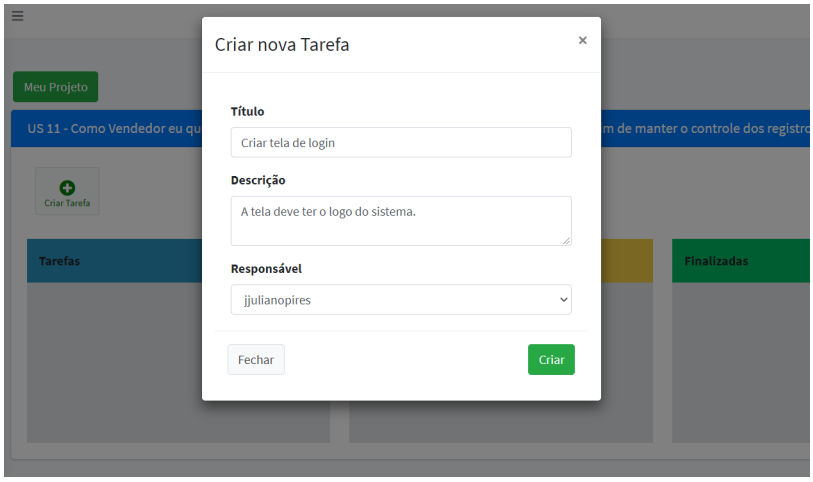
Figura 15 – Tela de Listagem de Histórias de Usuário.



Fonte: Próprio autor.

Através da definição das tarefas relacionadas as histórias de usuário, busca-se uma melhor organização e gerenciamento das atividades em realização por cada colaborador do projeto. Dessa forma, a ferramenta permite a definição dessas tarefas e associação aos colaboradores. A Figura 16 apresenta a tela de criação das tarefas e a opção de associação a um colaborador do projeto.

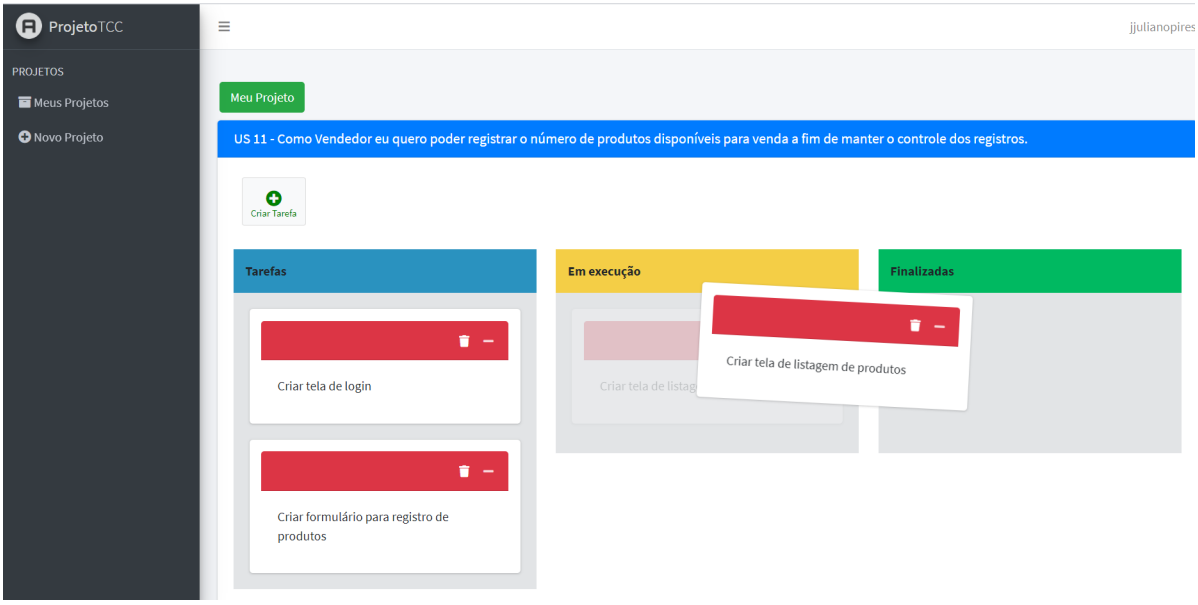
Figura 16 – Tela de definição de uma tarefa.



Fonte: Próprio autor.

O sistema visual em que as tarefas são apresentadas para o usuário segue uma estrutura de quadro kanban, no qual o usuário pode definir, alterar, remover uma tarefa, bem como, acompanhar o andamento do trabalho relacionado ao projeto. A Figura 17 apresenta o quadro de tarefas relacionadas a um requisito.

Figura 17 – Quadro de tarefas.

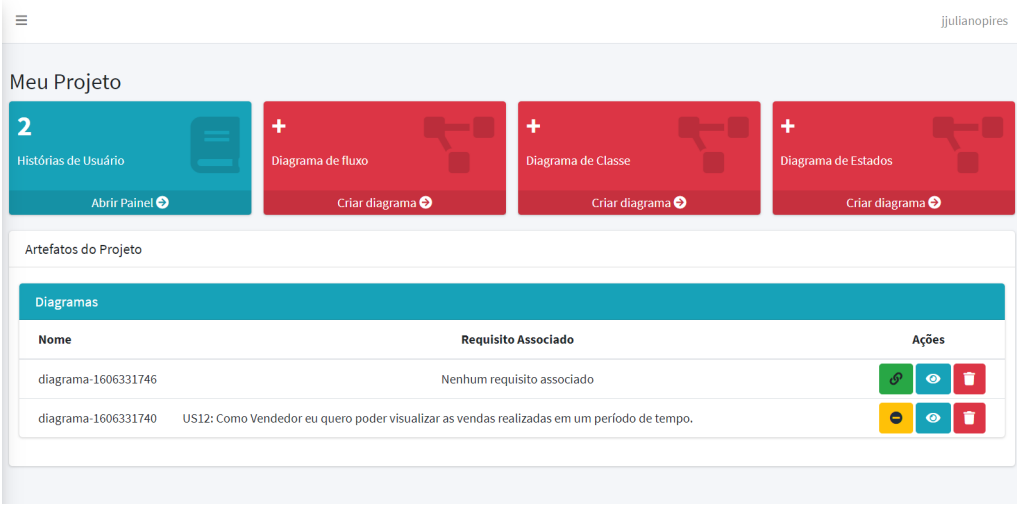


Fonte: Próprio autor.

A ferramenta desenvolvida também permite a criação de múltiplos artefatos de projetos de software. Esses artefatos, após produzidos, podem ser baixados pelos usuários ou ter seu registro mantido na própria ferramenta. A Figura 18 apresenta a tela de visualização de um único projeto. Nessa Figura é possível visualizar algumas das opções de diagramas disponíveis, que podem ser produzidos com a utilização da ferramenta.

Além disso, nessa tela é possível visualizar a lista de artefatos relacionados ao projeto.

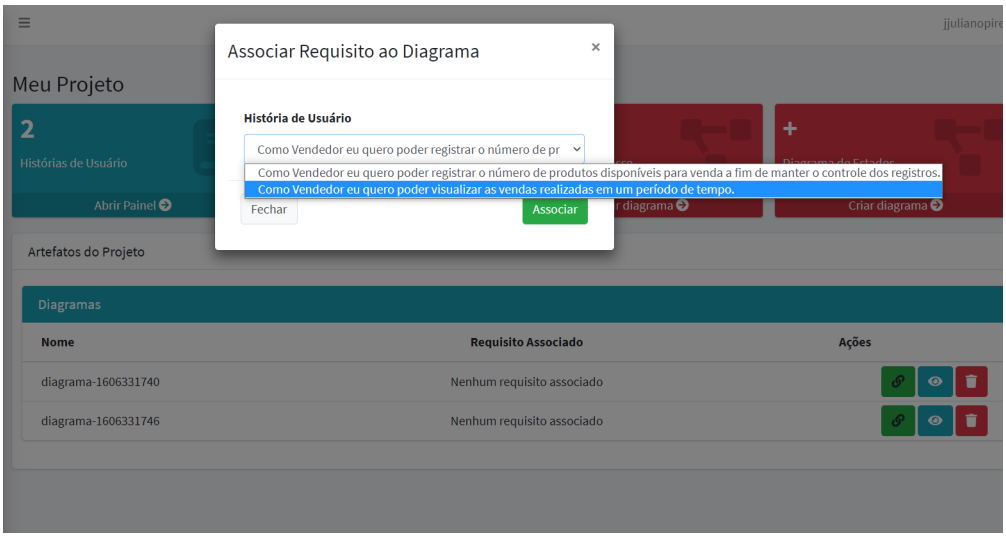
Figura 18 – Tela de visualização de um único projeto.



Fonte: Próprio autor.

Quando o usuário seleciona a opção de associar o artefato produzido a um requisito, a lista de requisitos é carregada e o usuário pode fazer a escolha de qual requisito associar. A Figura 19 apresenta a tela de associação do artefato produzido ao requisito.

Figura 19 – Tela associação do diagrama ao requisito.

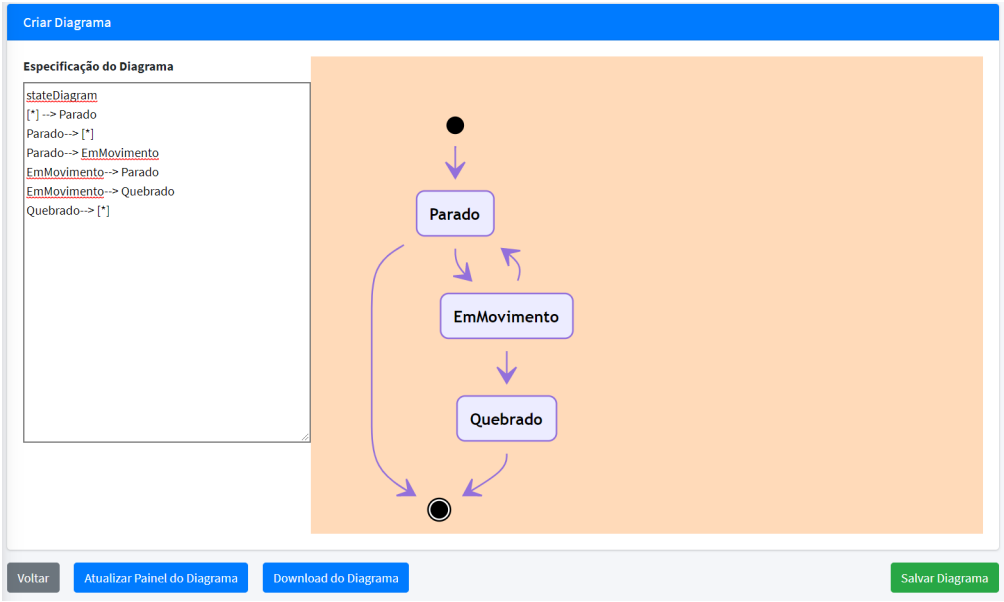


Fonte: Próprio autor.

Por fim, a Figura 20 apresenta um exemplo de diagrama de estados produzido dentro da ferramenta. Por possuir a integração com a Mermaid API, a ferramenta permite a criação de todos os tipos de diagramas disponibilizados pela Mermaid API através de seus serviços. Na tela de especificação do diagrama, ele pode ser construído seguindo a sintaxe reconhecida pela Mermaid API. Dessa forma, o diagrama é renderizado na tela

sempre que o usuário clicar no botão “Atualizar Diagrama”. Os dados do diagrama podem ser mantidos na ferramenta ou exportados para o formato de imagem.

Figura 20 – Diagrama de estados.



Fonte: Próprio autor.

Evidencia-se, através das funcionalidades relatadas neste trabalho, que a ferramenta dá suporte a ER, mais especificamente aos processos de Análise e Especificação de Requisitos, podendo ainda evoluir para cobrir mais processos da ER.

6.2 Resultados da Avaliação

O experimento foi realizado com objetivo de avaliar a aceitação dos usuários na utilização da ferramenta. A ferramenta foi avaliada durante o período de 6 de novembro a 13 de novembro de 2020. A aplicação do experimento teve cinco participantes: uma professora e quatro estudantes do curso de Engenharia de Software.

A execução do experimento foi controlada de forma remota, os testadores realizaram todas as tarefas em aproximadamente 30 minutos. As atividades eram observadas a fim de evitar qualquer tipo de interferência externa do ambiente ou interna dos próprios testadores.

Os resultados coletados através dos questionários possibilitaram uma visão geral da aceitação dos usuários nos aspectos citados no Capítulo 5.3. Na Tabela 9 é possível visualizar os resultados obtidos para cada atributo de qualidade analisado. Essa tabela foi construída com base nos resultados derivados da escala *likert*[2] de cada resposta do questionário.

[2] A escala *likert* utilizada em questionários, essa escala utiliza notas de 1 a 5 para medir a satisfação do usuário sobre alguma afirmativa.

Tabela 9 – Tabela contendo os resultados a avaliação (notas de 1 a 5).

Atributo de Qualidade	**Mínima**	**Máxima**	**Média**	**Moda**
Adequação Funcional	3	5	3.7	4
Eficiência de Desempenho	5	5	5	5
Confiabilidade	3	5	4.5	5
Segurança	4	5	4.8	5
Usabilidade	3	5	4.8	5

Fonte: autor

Para chegar a esses valores, foi considerada a nota de cada pergunta relacionada ao critério de qualidade. Conforme apresentado anteriormente na Tabela 7, cada critério de qualidade continha uma ou mais perguntas relacionadas a ele. Dessa forma, a Tabela 9 contém os dados da "Mínima", que representa a menor nota entre as questões; da "Máxima", que representa o maior valor entre as questões; da "Média", que representa a média entre as notas atribuídas e da "Moda", que representa o valor que mais se repete entre as questões.

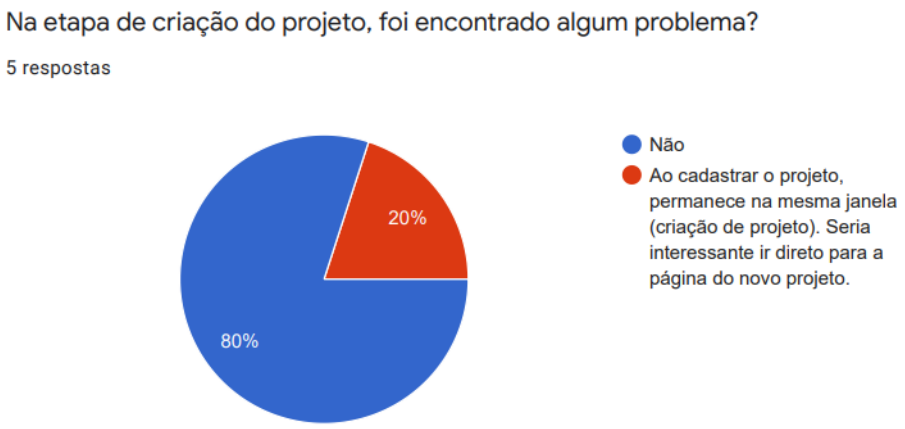
De acordo com os dados coletados, pode-se observar que a maioria dos atributos avaliados possui a nota máxima 5. Outro aspecto a ser considerado é que a menor nota foi atribuída para os atributos de Adequação Funcional, Confiabilidade e Usabilidade, visto que alguns usuários tiveram problemas durante a execução de certas funcionalidades. Mesmo assim, a moda de todos os atributos, com exceção do atributo de Adequação funcional, permaneceu com nota 5.

Outro atributo de qualidade que é importante destacar é o de Usabilidade. Esse foi o atributo que teve mais perguntas relacionadas no questionário e mesmo assim manteve-se com uma média de 4.8. Dessa forma, pode-se afirmar que a ferramenta tem uma boa usabilidade, considerando as perguntas feitas aos usuários.

Com relação a menor média que é a relacionada ao atributo de Adequação Funcional, espera-se aumentar o nível de qualidade relacionado a esse atributo, corrigindo os problemas encontrados durante a avaliação dos usuários.

O segundo formulário de avaliação, por apresentar mais perguntas abertas, possibilitou coletar uma percepção mais detalhada dos usuários. A seguir são apresentadas as questões em que os usuários expressaram sua percepção em relação à funcionalidade testada. A Figura 21 apresenta as respostas dos usuários em relação à funcionalidade de criação do projeto.

Figura 21 – Questão relacionada à etapa de criação de um projeto.

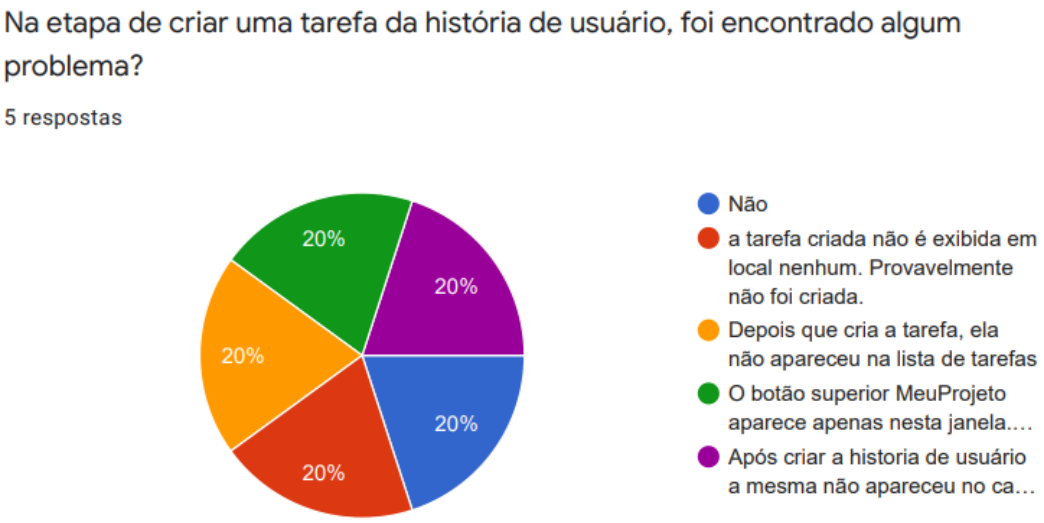


Fonte: Próprio autor.

De acordo com a resposta de um usuário, seria interessante ir para a página do projeto logo após sua criação, o que indica que o fluxo das janelas que são exibidas para o usuário final podem ser revistas para proporcionar uma melhor usabilidade da ferramenta.

A Figura 22 apresenta as respostas dos usuários em relação à funcionalidade de criação de uma tarefa da história de usuário. Essa foi a funcionalidade que, de acordo com as respostas, apresentou o maior número de problemas. Três usuários reportaram que a tarefa criada não apareceu no *kanban* de tarefas, ou seja, não foi criada, o que indica que essa funcionalidade deve ser revista e corrigida. Outro usuário indicou que o botão relacionado ao "Meu projeto" pode ser substituído por um botão "voltar", o que indica que a nomenclatura dos botões pode ser revista.

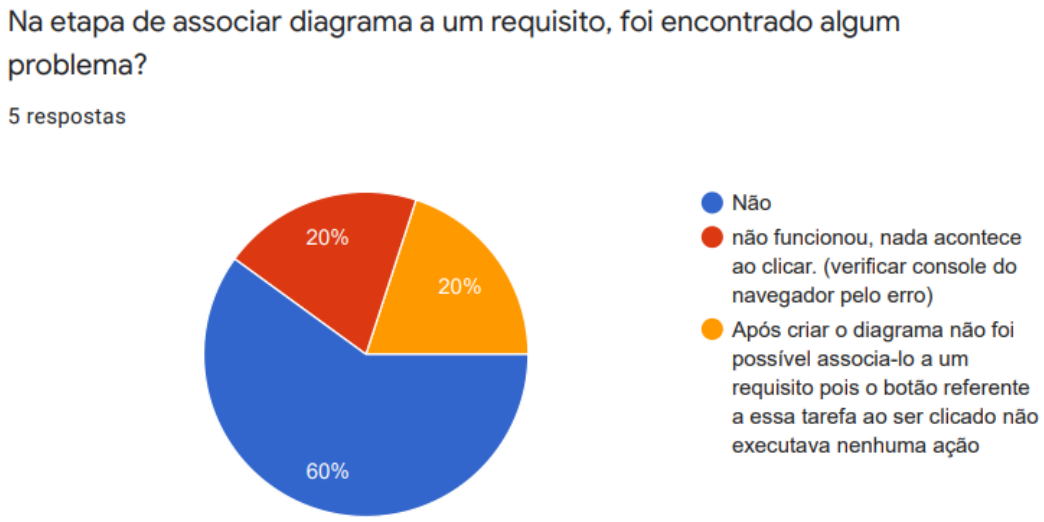
Figura 22 – Questão relacionada à etapa de criação de uma tarefa.



Fonte: Próprio autor.

Por fim, a Figura 23 apresenta as respostas dos usuários em relação à funcionalidade de associar um diagrama a um requisito. De acordo com dois usuários, o botão que é responsável por essa funcionalidade não estava sendo executado. Dessa forma, foi possível identificar mais um ponto de melhoria da ferramenta com a ajuda das pessoas envolvidas no processo de avaliação.

Figura 23 – Questão relacionada à associação de um diagrama a um requisito.



Fonte: Próprio autor.

6.3 Considerações do Capítulo

No decorrer deste Capítulo foi apresentada a ferramenta Clover. Através dela, um usuário pode gerenciar seus projetos de forma integrada ao GitHub. A ferramenta ajuda a manter o controle do andamento do projeto através da especificação e acompanhamento de tarefas relacionadas da cada colaborador do projeto. Através de uma interface simples, a ferramenta permite a especificação de requisitos e a construção de diagramas, dando assim, suporte à ER e permitindo a colaboração de pessoas envolvidas no projeto.

Para a ferramenta poder cobrir todas as funcionalidades descritas na Seção 6.1, foi demandado bastante esforço para o entendimento e construção de uma solução integrada com a GitHub API e a Mermaid API. Soma-se a isso, as outras funcionalidades implementadas, como o quadro kanban, que apesar de ser construído com a utilização da biblioteca Jkanban, não é algo trivial e também demandou bastante esforço.

Através da avaliação realizada com diferentes usuários, foi possível identificar pontos fortes da ferramenta, como a interface e a facilidade da utilização, e pontos fracos, como erros na utilização. Portanto, realizar a avaliação com os usuários foi importantíssimo, pois permitiu a detecção de erros que não aconteceriam em outros cenários de uso, além de permitir identificar aspectos que necessitam ser evoluídos na ferramenta.

7 CONSIDERAÇÕES FINAIS

O principal objetivo deste trabalho foi o desenvolvimento de uma ferramenta para apoio à engenharia de requisitos de forma colaborativa. A participação na atividade Resolve!, do programa de extensão Programa C, foi essencial para a ideia inicial da criação da ferramenta, motivada pela contribuição que ela pode trazer ao ser inserida nesse contexto ou em outro que aborde a ER em suas atividades.

Na ferramenta Clover, o usuário pode criar projetos, especificar requisitos utilizando a narrativa de histórias de usuário e definir tarefas em colaboração com os demais membros do projeto. A ferramenta também disponibiliza um editor para a criação de diagramas, assim o usuário pode produzir diversos tipos de diagramas mantendo a relação dos dados entre requisitos, projetos, tarefas e artefatos. Destaca-se que a ferramenta atualmente cobre os processos de Análise e Especificação de Requisitos, podendo ainda evoluir para cobrir mais processos relacionados à ER.

Para realizar o desenvolvimento dessa ferramenta foi verificado o estado da arte para identificar trabalhos que pudessem contribuir com a presente proposta. Através desses trabalhos foi possível identificar outras ferramentas e mapear funcionalidades. O principal diferencial de ferramenta Clover em relação aos trabalhos analisados no Capitulo 3 é que ela, através do uso de diferentes APIs, plataformas e bibliotecas, centraliza um conjunto de funcionalidades que não são encontradas de maneira integrada nas ferramentas dos trabalhos relacionados,

No decorrer do desenvolvimento da ferramenta, foram encontrados desafios ocasionados pela troca de informações entre a ferramenta desenvolvida e a GitHub API, o que demandou bastante esforço. Outro desafio foi a utilização da biblioteca Jkanban, pois utilizá-la e personalizá-la não é algo trivial e sua documentação não é ampla. Soma-se a isso as demais funcionalidades que deveriam ser pensadas e implementadas com diferentes tipos de tecnologias. Destaca-se ainda que, devido à pandemia de covid-19 ter ocorrido ao longo deste ano, a ferramenta não pode ser desenvolvida com um maior envolvimento das partes interessadas, como planejado inicialmente.

Os conhecimentos da Engenharia de Software desenvolvidos durante o curso, desde a elaboração dos requisitos do sistema até a gestão do processo de desenvolvimento do software, foram muito importantes para a execução desse trabalho. A capacidade de resolver problemas e o trabalho sistemático que também é explorado no curso de Engenharia de Software foram essenciais para a conclusão desse trabalho.

Em relação a trabalhos futuros, espera-se seguir evoluindo a ferramenta para dar suporte a mais processos da ER, bem como, complementar e deixar mais robustas as funcionalidades que já estão implementadas. Outro trabalho futuro é a atualização da interface da ferramenta, através da gamificação das funcionalidades que já estão implementadas, para assim torná-la ainda mais atrativa para o usuário final. Por fim, espera-se, em um trabalho futuro, poder realizar validações ou a elicitação de requisitos com as par-

tes interessadas através da ferramenta, tendo em vista que sua arquitetura já foi planejada pensando nessa evolução.

REFERÊNCIAS

[1] Scrum desenvolvimento Ágil. `https://www.desenvolvimentoagil.com.br/scrum/`. Accessed: 2019-10-2. Citado 3 vezes nas páginas 33, 34 e 35.

[2] R. e. G. S. Ajmeri, Nirav e Sejpal. A semantic and collaborative platform for agile requirements evolution. In *2010 Third International Workshop on Managing Requirements Knowledge*, pages 32–40. IEEE, 2010. Citado 3 vezes nas páginas 41, 42 e 43.

[3] A. M. M. Aline Vieira de Mello. Programa C - comunidade, computação, cultura, comunicação, ciência, cidadania, criatividade, colaboração. `https://sites.unipampa.edu.br/cienciacao/2020/10/15/programa-de-extensao-programa-c/`. Accessed: 2020-12-10. Citado na página 27.

[4] E. e. R. R. A. Anton, Annie I e Liang. A web-based requirements analysis tool. In *Proceedings of WET ICE'96. IEEE 5th Workshop on Enabling Technologies; Infrastucture for Collaborative Enterprises*, pages 238–243. IEEE, 1996. Citado 3 vezes nas páginas 40, 41 e 42.

[5] B. e. P. H. e. L. S. G. e. J. L. D. S. J. e. B. R. O. Azevedo, Diogo e Fonseca. On the development and usability of a diagram-based collaborative brainstorming component. *Journal of Universal Computer Science, 19 (7), 2013*, 2013. Citado 4 vezes nas páginas 40, 41, 42 e 43.

[6] R. E. e. o. Bourque, Pierre e Fairley. *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014. Citado 3 vezes nas páginas 23, 28 e 29.

[7] K. R. N. Costa. Personal scrum: uma alternativa ágil para desenvolvimento de indie games. 2016. Citado na página 45.

[8] C. A. Ellis, S. J. Gibbs, and G. Rein. Groupware: some issues and experiences. *Communications of the ACM*, 34(1):39–58, 1991. Citado 2 vezes nas páginas 23 e 31.

[9] M. Fuks, Hugo e Pimentel. *Sistemas colaborativos*. Elsevier Brasil, 2011. Citado 3 vezes nas páginas 23, 31 e 32.

[10] A. L. Ghofrani, Javad e Fehlhaber. Productlinre: online management tool for requirements engineering of software product lines. In *Proceedings of the 22nd International Systems and Software Product Line Conference-Volume 2*, pages 17–22. ACM, 2018. Citado 3 vezes nas páginas 41, 42 e 43.

[11] P. Gruenbacher. Integrating groupware and case capabilities for improving stakeholder involvement in requirements engineering. In *Proceedings of the 26th Euromicro Conference. EUROMICRO 2000. Informatics: Inventing the Future*, volume 2, pages 232–239. IEEE, 2000. Citado 4 vezes nas páginas 40, 41, 42 e 43.

[12] H. C. Hu, Wei-Chung e Jiau. Ucframe: A use case framework for crowd-centric requirement acquisition. *ACM SIGSOFT Software Engineering Notes*, 41(2):1–13, 2016. Citado 3 vezes nas páginas 40, 41 e 42.

[13] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering, 2007. Citado na página 37.

[14] M. Kolpondinos, Martina Z e Glinz. Garuso: a gamification approach for involving stakeholders outside organizational reach in requirements engineering. *Requirements Engineering*, pages 1–28, 2019. Citado 4 vezes nas páginas 40, 41, 42 e 43.

[15] D. Michaelis. Disponível em:< http://michaelis. uol. com. br>. *Acesso em*, 11(04), 2011. Citado na página 23.

[16] M. e. S. S. e. R. M. e. G. J. Moketar, Nor Aiza e Kamalrudin. An automated collaborative requirements engineering tool for better validation of requirements. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 864–869. IEEE, 2016. Citado 3 vezes nas páginas 41, 42 e 43.

[17] H. H. e. D. N.-K. e. T. D.-T. Nguyen, Vu e Dang. Enhancing team collaboration through integrating social interactions in a web-based development environment. *Computer Applications in Engineering Education*, 24(4):529–545, 2016. Citado 3 vezes nas páginas 40, 41 e 42.

[18] A. e. V. A. e. P. J. A. Ochoa, Sergio F e Quispe. Improving requirements engineering processes in very small software enterprises through the use of a collaborative application. In *The 2010 14th International Conference on Computer Supported Cooperative Work in Design*, pages 116–121. IEEE, 2010. Citado 3 vezes nas páginas 41, 42 e 43.

[19] M. A. e. F. D. e. R. A. e. F. H. e. L. C. J. P. d. Pimentel, Mariano e Gerosa. Modelo 3c de colaboração para o desenvolvimento de sistemas colaborativos. *Anais do III Simpósio Brasileiro de Sistemas Colaborativos*, pages 58–67, 2006. Citado 2 vezes nas páginas 30 e 31.

[20] B. Pressman, Roger e Maxim. *Engenharia de Software-7ª Edição*. McGraw Hill Brasil, 2011. Citado 2 vezes nas páginas 23 e 28.

[21] C. Ribeiro, C. Farinha, and M. M. Pereira, João e da Silva. Gamifying requirement elicitation: Practical implications and outcomes in improving stakeholders collaboration. *Entertainment Computing*, 5(4):335–345, 2014. Citado 4 vezes nas páginas 40, 41, 42 e 43.

[22] J. Schwaber, K e Sutherland. Guia do scrum–um guia definitivo para o scrum: As regras do jogo. 2013. *Acessado em 26/09/2019*. URL `https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf`. Citado 4 vezes nas páginas 33, 34, 35 e 45.

[23] Sommerville. *Engenharia de Software-9ª Edição*. Pearson Prentice Hall, 2011. Citado 4 vezes nas páginas 23, 28, 29 e 30.

[24] T. e. V. D. S. H. Usländer, Thomas e Batz. Servus–collaborative tool support for agile requirements analysis. In *Environmental Software Systems. Fostering Information Sharing: 10th IFIP WG 5.11 International Symposium, ISESS*, pages 9–11, 2013. Citado 4 vezes nas páginas 40, 41, 42 e 43.

[25] I. Williams. An ontology based collaborative recommender system for security requirements elicitation. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 448–453. IEEE, 2018. Citado 4 vezes nas páginas 40, 41, 42 e 43.

Apêndices

APÊNDICE A – PRINCIPAIS FUNCIONALIDADES IDENTIFICADAS

	Principais Funcionalidades	E	I	D
01	Publicar um requisito, para que outros usuários possam ver.	x		
02	Compartilhar um requisito publicado por outra pessoa.			x
03	Avaliar os requisitos publicados por outros usuários.	x		
04	Priorizar os requisitos.	x		
05	Realizar a rastreabilidade entre projetos, artefatos e requisitos.	x		
06	Gerar pacotes com artefatos de outros projetos, para projetos semelhantes .			x
07	Fazer *upload* de artefatos como vídeo, áudio, texto, imagem ou arquivos XML.	x		
08	Reutilização de artefatos de outro projeto.			x
09	Recomendação de requisitos de segurança.			x
10	Recomendações baseadas na representação do conhecimento de um domínio (ontologia).			x
11	Seleção de informações consideradas “boas” para a solução existente, mediante a recomendações feitas pela ferramenta.			x
12	Realizar recomendações automáticas por palavra chave ou padrões de semelhança, a partir da comparação outros casos de uso casos de uso;			x
13	Extração e visualização de conceitos de casos de uso.			x
14	Aquisição de requisitos baseado na multidão ou centrado na multidão de usuários.			x
15	Dar suporte a negociação de requisitos e a colaboração das partes interessadas.			x
16	Usuários da multidão criam e revisam casos de uso de outras pessoas simultaneamente.			x
17	Utilizar um modelo de sintaxe e gráfico visual como forma de assistência.			x
18	Fornecer dicas e atualizar as informações simultaneamente, na medida que recebe mais informações dos usuários.			x
20	Extração de conceitos a partir dos requisitos digitados.			x
21	Gerar um conjunto de testes abstratos e casos de teste de forma automática, a partir da especificação dos requisitos.			x
22	Gerar uma interface do usuário de *mock-up* para revisar e validar o comportamento esperado dos requisitos.			x
23	Permitir que o analista execute os casos de teste e indique o status do teste.			x

24	Permitir que o Engenheiro inicialize uma discussão com o cliente através da ferramenta.	x		
25	Permitir que os usuários realizem comentários, em relação aos resultados em discussão.		x	
26	Usuários podem se comunicar usando os recursos de bate-papo.			x
27	Permitir que várias partes interessadas validem o mesmo conjunto de requisitos.		x	
28	*Feeds* de notícias e painéis.			x
29	Disponibilizar espaços de trabalho compartilhado.		x	
30	Definir tarefas.	x		
31	Monitorar e acompanhar tarefas.			x
32	Gerar e modificar colaborativamente conteúdos, nesse caso, o conteúdo pode ser social ou específico do projeto.			x
33	Medição da participação no trabalho e participação na comunicação.			x
34	Dar suporte a realização de atividades de grupos focais.			x
35	Visualizações sociais.			x
36	Permitir que usuários possam visualizar a lista de requisitos que foram enviados pelos outros usuários.	x		
37	Anonimato.			x
38	Apresentar o processo de obtenção de requisitos como um jogo.			x
39	Dar suporte a participação simultânea de usuários em uma sessão de *brainstorming*;			x
40	Gerar uma coleção de possíveis requisitos de forma automática, a partir dos que foram coletados;			x
41	Gerar (blocos) conceitos com base no texto e conectar conceitos.			x
42	Controle de simultaneidade, na edição de artefatos.	x		
43	Utilizar um metamodelo de especificação de casos de uso semi-fromal.			x
44	Dar suporte a uma abordagem ágil de desenvolvimento.			x
45	Gerar documentos que possam ser importados via LATEX.			x
46	Utilizar ima ontologia e regras de inferência para auxiliar na especificação de requisitos.			x
47	Feedback social, votação e comentários sobre os requisitos de todas as partes interessadas.			x

48	Criar projetos.	x		
49	Fornecer concessões de acesso aos usuários.	x		
50	Especificação de vínculo entre pessoas e requisitos.	x		
51	Definição de artefatos de design, através da ferramenta.			x
52	Referenciar protótipos desenvolvidos a artefatos e requisitos.	x		
53	Visualização do status da evolução do projeto.	x		
54	Exportação de documentos de acordo com modelos padronizados.			x
55	Permitir que partes interessadas possam acompanhar o andamento do projeto.	x		
56	Geração e organização idéias ou conceitos.		x	
57	Votação distribuída e priorização.	x		
58	Interação simultânea entre os participantes.			x
59	Anonimato de usuários para a coleta de idéias.			x
60	Registrar histórico de informações sobre a reunião.	x		
61	Rastreabilidade de requisitos baseados em objetivos.			x
62	Especificação de agentes responsáveis por cada objetivo, obstáculos e cenários do objetivo.			x
63	Refinamento, análise e decomposição de metas.			x

APÊNDICE B – INSTRUÇÕES DA AVALIAÇÃO

Avaliação com o usuário

Sistema Clover

06 de novembro de 2020

Passos para a execução da avaliação

Etapa 1: Realizar o login e criar um projeto

- Para começarm, acesse o link https://murmuring-castle-41017.herokuapp.com/
- Clique em “Login”.
- Entre com sua conta de usuário do GitHub, caso não tenha uma conta, crie uma acessando o seguinte link https://github.com/join
- Dê autorização ao aplicativo para poder manipular os projetos.
- Após carregar a tela inicial clicar em “Novo Projeto”, no menu lateral esquerdo.
- Preencha os campos com os seguintes dados:
    - No campo “Nome do projeto”, digite: MeuProjetoTeste
    - No campo “Descrição”, digite: Descrição Do Projeto Teste
    - No campo “Tipo de repositório”, selecione a opção: Público
- Após preencher clique no botão “Finalizar Cadastro”.

Etapa 2: Criar uma história de usuário

- Clique na opção “Meus Projetos” no menu lateral.
- Após carregar a lista de projetos, selecione o projeto criado na etapa anterior (MeuProjetoTeste).
- Na opção “Histórias de Usuário”, clique em “Abrir Painel”.
- Preencha dos campos com os seguintes dados:
    - No campo “Descrição” digite: Como Administrador eu quero registrar os produtos a fim de manter registrado em estoque, todos os produtos já fornecidos.
    - No campo “Tempo estimado para conclusão em horas”, digite: 8
    - No campo “”Prioridade” selecione a opção “Média”.
- Após preencher os campos, clique na opção “Adicionar”.

Etapa 3: Criar uma tarefa da história de usuário

- ➢ Ainda na tela de criação da história de usuário, clicar na opção “Listar História de Usuário”.
- ➢ Após carregar a lista de histórias de usuário, selecione a história criada.
- ➢ Clique na opção “Criar Tarefa”.
- ➢ Preencha dos campos com os seguintes dados:
    - ○ No campo “Título” digite: Modificar título da tabela de produtos.
    - ○ No campo “Descrição”, digite: O nome título da tabela deve ser Listagem de Produtos.
    - ○ No campo “Responsável” selecione a opção relacionada ao seu nome de usuário do GitHub.
- ➢ Clique no botão “Criar”.
- ➢ Mova o cartão correspondente a tarefa criada para a aba “Em execução”.

Etapa 4: Criar um diagrama de classe

- ➢ Selecione a opção “Meu Projeto”, para retornar para a tela do projeto, ou clique em “Meus Projetos” e selecione o projeto criado na Etapa 1 “MeuProjetoTeste”.
- ➢ Clique no Cartão “Diagrama de Classe”, selecione a opção “Criar Diagrama”.
- ➢ No campo “Especificação do Diagrama”, digite o seguinte texto:

```
classDiagram

Animal <|-- Pato

Animal <|-- Peixe

Animal <|-- Zebra

Animal : +int idade

Animal : +String genero

Animal: +ehMamifero()

Animal: +companheiro()

class Pato{

+String corBico

+nadar()

+grasnar()

}

class Peixe{
```

-int tamanho

-podeComer()

}

class Zebra{

+bool ehSelvagem

+correr()}

- ➢ Após colocar o texto no campo, clique no botão “Atualizar o Painel do Diagrama”.
- ➢ Após o diagrama ser atualizado, clique no botão “Download do Diagrama” para baixa-lo.
- ➢ Após fazer o download do diagrama, clique no botão “Salvar Diagrama”.

Etapa 5: Associar diagrama a um requisito

- ➢ Dentro da tela do “Meu Projeto”, clique no botão “Associar o diagrama ao requisito”, que está na lista de diagramas na coluna “Ações”.
- ➢ Na caixa de seleção, selecione a história de usuário criada na etapa 3, “Como Administrador eu quero registrar os produtos a fim de manter registrado em estoque, todos os produtos já fornecidos.
- ➢ Clique no botão associar.
- ➢ Após associar o diagrama criado ao requisito, clique no seu nome de usuário no canto superior direito da tela, e selecione a opção “logout”.
- ➢ Na tela que abrir, clique no botão “Sign out”.

Etapa 6: Responder os questionários

Primiero formulário:

https://docs.google.com/forms/d/e/1FAIpQLSfmHpFCDotiWlov1jH24uQR3e4QYLLvdGSWRTJk8nMYXzOb8Q/viewform?usp=sf_link

Segundo formulário:

https://forms.gle/jqJgmwLJWkdHcfwf8

APÊNDICE C – PRIMEIRO QUESTIONÁRIO DE AVALIAÇÃO

Formulário de avaliação do sistema colaborativo.

Sua opinião é de extrema importância, através dessas informações será possível trabalhar em melhorias na ferramenta.

*Obrigatório

O sistema fornece os resultados corretos para cada funcionalidade executada? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

O sistema cumpre bem a sua função, de acordo com a proposta? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

O tempo de resposta ao efetuar as operações é rápido? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

O sistema é estável durante seu uso? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

O sistema pode funcionar mesmo quando há alguma falha? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

O sistema evita acesso não autorizado? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

O sistema pode substituir outro produto de software com a mesma finalidade? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Você reconhece o sistema como adequado para a finalidade proposta? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Você considera fácil aprender como o sistema funciona? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

A aplicação é fácil de ser operada? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

A interface do usuário é agradável aos olhos? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

É possível entender o modelo de interface proposto pela aplicação? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Comentários, sugestões ou críticas para ajudar a melhorar o sistema:

Sua resposta

Enviar

APÊNDICE D – SEGUNDO QUESTIONÁRIO DE AVALIAÇÃO

Formulario2

Perguntas Respostas 5

Formulário de avaliação do sistema

Este formulário tem como objetivo testar as funcionalidades do sistema colaborativo CLOVER, desenvolvido como parte do trabalho de conclusão do curso Engenharia de Software da Universidade Federal do Pampa..

Na etapa de criação do projeto, foi encontrado algum problema? *

Se sim, por favor, descreva no campo "outros".

- Não
- Outros...

Na etapa de criação de uma história de usuário, foi encontrado algum problema? *

Se sim, por favor, descreva no campo "outros".

- Não
- Outros...

Na etapa de criar uma tarefa da história de usuário, foi encontrado algum problema? *

Se sim, por favor, descreva no campo "outros".

- Não
- Outros...

Se sim, por favor, descreva no campo "outros".

- ○ Não
- ○ Outros…

Na etapa de associar diagrama a um requisito, foi encontrado algum problema? *

Se sim, por favor, descreva no campo "outros".

- ○ Não
- ○ Outros…

Espaço para outras críticas ou sugestões para melhorar o sistema.

Texto de resposta longa

ÍNDICE