Software para Gerenciamento de Projetos Scrum

Christopher Marim[1], **Guilherme Pinheiro**[1], **Alexandre Perin de Souza**[1]

[1]Instituto Federal de Santa Catarina (IFSC) - Campus Lages
Rua Heitor Villa Lobos, 225 – 88.506-400 – Lages – SC – Brasil

cssmarim1@gmail.com, guilheme.sp12@aluno.ifsc.edu.br

alexandre.perin@ifsc.edu.br

Abstract. This paper proposes a Scrum project management tool to assist in the use of the Scrum method, through tables for organizing tasks and data visualization through dashboards. The tool consists of a web application that allows the user to register and monitor the process and results of their Scrum projects. It presents information about project data registered in the application, with an interface that seeks to improve the user interaction experience.

Resumo. *Este artigo propõe uma ferramenta de gerenciamento de projetos Scrum para auxiliar no uso do método Scrum, por meio de quadros para organização de tarefas e da visualização de dados através de* dashboards. *A ferramenta consiste em uma aplicação web que permite que o usuário cadastre e acompanhe o processo e o resultado de seus projetos Scrum. Ela apresenta informações sobre dados de projetos cadastrados na aplicação, com uma interface que procura melhorar a experiência e a interação do usuário.*

1. Introdução

A Engenharia de Software (ES) trata da aplicação de técnicas e ferramentas sistemáticas, disciplinadas e quantificáveis para desenvolver softwares com qualidade (Valente, 2022). Dentro das técnicas disponíveis na ES encontram-se os modelos de desenvolvimento de software. Eles objetivam organizar um conjunto de atividades para obtenção de software. Há duas famílias de modelos de desenvolvimento de software: modelos tradicionais, burocráticos, pesados e os modelos ágeis ou leves.

Os métodos tradicionais possuem processos preestabelecidos que são definidos no início do projeto e trazem pouca flexibilização durante seu desenvolvimento. Isto gera, em muitos casos, a falta de cumprimento de prazo, compromete a qualidade, entre outros fatores. Um exemplo disso é o método em cascata. Já os métodos ágeis são mais adaptáveis à novos fatores decorrentes do desenvolvimento do projeto, ao invés de procurar analisar previamente tudo o que pode acontecer no decorrer do desenvolvimento (Soares, 2004).

O Scrum é um método ágil para gestão e planejamento de projetos de software. Ele objetiva organizar as atividades de desenvolvimento, entregar e sustentar produtos em um ambiente complexo, com ênfase inicial no desenvolvimento de software, embora tenha sido usado em outras áreas, incluindo pesquisa, tais como vendas e marketing. Nele, procura-se otimizar a previsibilidade e controlar o risco, engajando grupos de pessoas que coletivamente têm todas as habilidades e as experiências necessárias para executar o trabalho e para compartilhar ou adquirir essas habilidades conforme o necessário.

O Scrum foi apresentado oficialmente em 1995 por Ken Schwaber e Jeff Sutherland na conferência *Object-oriented Programming, Systems, Languages, and Applications* (OOPSLA), tendo seu Guia do Scrum (Schwaber e Sutherland, 2020). O Scrum é iterativo, incremental e se concentra em gerenciar o projeto de um produto que acrescente valor para o negócio. Este valor decorre da funcionalidade propriamente dita, do prazo em que é produzido, do custo e da qualidade (Martins, 2007).

No mercado, conforme apresenta Klann (2019), há diversas ferramentas que permitem o gerenciamento de projetos Scrum, ou seja, permitem documentar suas várias fases, processos e eventos. No entanto, apesar de existir várias ferramentas para documentações de projetos de *software*, muitas delas falham na apresentação e visualização dos eventos e das atividades relacionadas ao Scrum de forma fácil e rápida. Esta carência dificulta o acompanhamento das mudanças, os avanços e as melhorias que correm ao longo do desenvolvimento de um software.

Neste contexto, este trabalho propõe a criação de uma ferramenta computacional para auxiliar no gerenciamento de projetos Scrum, que possibilite a fácil organização, navegação e registro das atividades e eventos relacionados a um projeto de *software*. Para alcançar esse resultado foram definidos os seguintes objetivos específicos:

- Identificar e conhecer as fases, atividades e eventos que fazem parte do Scrum;
- Analisar ferramentas de gerenciamento de projetos Scrum e utilizá-las como referência;
- Projetar a camada de armazenamento de dados e implementar uma aplicação *front-end* e *back-end*, para realizar a organização sistemática de projetos Scrum;
- Avaliar os módulos desenvolvidos.

Este trabalho está dividido em cinco etapas macro. A primeira etapa será identificar e conhecer as fases, atividades e eventos do modelo Scrum. Para isso foi realizado a leitura e estudos de artigos pesquisados na *Internet*. A segunda etapa corresponde em fazer uma análise e comparação entre ferramentas de gerenciamento de projetos Scrum para usar de referência. Para isso foi feita uma pesquisa sistemática das ferramentas e avaliação das mesmas, que é apresentada na seção 2. A terceira etapa será focada em projetar a camada de armazenamento de dados e criação de *Application Programming Interface (API)*[1] utilizando *NodeJS*, para que o *front-end* consiga se comunicar com o banco de dados, será utilizado o sistema *PostGresSQL*, e como Ambiente de desenvolvimento integrado (*IDE*), o *Visual Studio Code*. Na quarta etapa será feito o *front-end* seguindo padrões *UX*[2] e *UI*[3] utilizando *ReactJS*. A última etapa será a avaliação dos módulos desenvolvidos, onde será realizada uma entrevista com pessoas disponíveis a testar a ferramenta desenvolvida durante este trabalho.

Sob o aspecto metodológico, este trabalho se classifica de acordo com a natureza, sendo de pesquisa aplicada. No ponto de vista da abordagem do problema, é qualitativo. Da perspectiva dos objetivos, segue o caminho exploratório, visto que se aprofunda no problema e na solução existente, para agregar novas funcionalidades e possibilidades de

[1] *API*: É um conjunto de definições e protocolos para criar e integrar softwares.

[2] *User Experience*: Trata-se de como o cliente interage com o seu produto ou serviço a partir dos elementos disponibilizados.

[3] *User Interface*: Trata-se de uma área voltada para criação de interfaces mais fáceis e amigáveis.

melhorias. Referente ao ponto de vista dos procedimentos técnicos será uma pesquisa bibliográfica.

Além da primeira seção, o restante do trabalho está organizado da seguinte forma: a seção dois apresenta o referencial teórico, ou seja, descreve os assuntos necessários para o entendimento do trabalho como um todo, e a análise de trabalhos similares; a seção três registra como será realizado o desenvolvimento do projeto; a seção quatro apresenta a avaliação do projeto e a seção cinco finaliza o trabalho com as considerações finais.

2. Referencial Teórico

Esta seção está dividida em duas subseções. A primeira subseção explica o funcionamento detalhado do método Scrum. A segunda subseção discorre sobre projetos que têm como foco resolver o problema tratado neste trabalho, mas com aspectos diferentes.

2.1. Scrum

De acordo com Schwaber e Sutherland (2020), o Scrum é um método que facilita, através de soluções adaptativas, resolver problemas complexos. O Scrum aplica uma abordagem iterativa (cíclica) e incremental (adição de novas funções) para otimizar a previsibilidade e controlar o risco.

No modelo Scrum, implementa-se os três pilares da transparência, inspeção e adaptação (Schwaber e Sutherland, 2020):

- Transparência: determina quais pontos principais de um processo devem ser conhecidos pela equipe, apresentados a partir de um entendimento compartilhado por todos.
- Inspeção: é de suma importância haver inspeções de maneira frequente e diligente no decorrer do projeto, para se detectar potenciais variações ou problemas indesejados.
- Adaptação: visa-se a importância de ter uma equipe flexível a possíveis mudança durante o projeto. Em casos de quaisquer ajustes, os mesmos devem ser feitos o quanto antes, para minimizar desvios.

A estrutura que forma o Scrum é simplificada. São definidas algumas diretrizes gerais como: regras, papéis, artefatos e eventos. Entretanto, cada um desses componentes é de extrema importância em todo o desenvolvimento do projeto e é essencial para o sucesso manter todos os componentes da estrutura (Dinamize, 2022).

A figura 1 (desenvolvimentoagil, 2022) apresenta o funcionamento do modelo Scrum. Nela, tem-se dois artefatos que são usados para listar e priorizar os requisitos: *product backlog* e *sprint backlog*. Os ciclos iterativos são representados pelos *sprints* e os eventos representados são: *daily scrum* e *sprint review*.

2.1.1. Artefatos

De acordo com Schwaber e Sutherland (2020), os artefatos do Scrum representam o trabalho do produto que será desenvolvido. É de suma importância que tais artefatos estejam de acordo com os três princípios do Scrum, para que não haja falhas no entendimento do que será executado. Tais artefatos são: *Product Backlog* e *Sprint Backlog*.

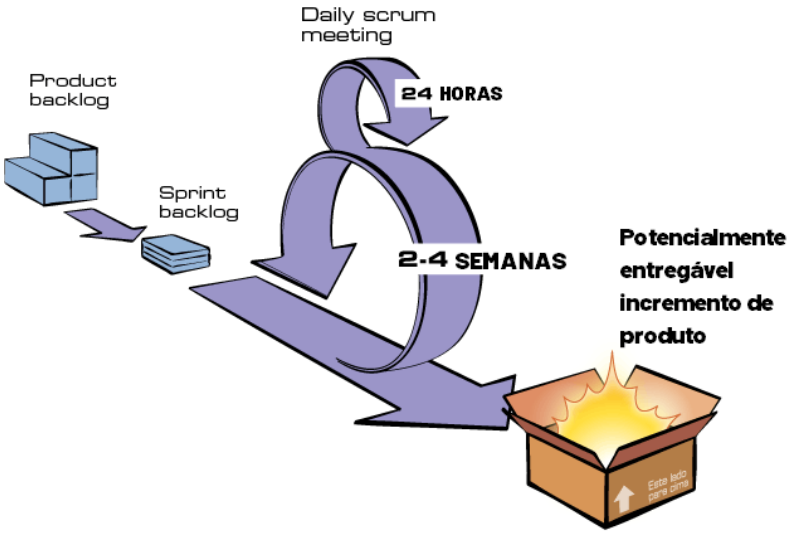

Figura 1. Funcionamento de um projeto Scrum

O *Product Backlog* ou Backlog do Produto é uma lista ordenada por prioridade do que é necessário para criar o produto. Tal lista é dinâmica, podendo ser alterada durante a criação do projeto.

O *Sprint Backlog* ou Backlog da *Sprint* é uma lista de atividades criadas com base no *Product Backlog* que precisam ser feitas durante uma *Sprint*. Essa lista retém as atividades que serão feitas com base na capacidade da equipe de entregar as atividade.

2.1.2. Eventos em um projeto Scrum

No Scrum, os projetos são divididos em janelas de tempo (tipicamente mensais) chamados de *Sprints*. O *Sprint* representa um *Time Box* (janela máxima de tempo) dentro do qual um conjunto de atividades deve ser executado. Uma *Sprint* é um evento com duração fixa e estabelecida antes do início do projeto. Uma nova *Sprint* começa imediatamente após a conclusão da *Sprint* anterior.

No início de cada *Sprint*, faz-se um *Sprint Planning Meeting*, ou seja, uma reunião de planejamento na qual o *Product Owner* prioriza os itens do *Product Backlog* e a equipe seleciona as atividades que ela será capaz de implementar durante o *Sprint* que se inicia. As tarefas alocadas em um *Sprint* são transferidas do *Product Backlog* para o *Sprint Backlog*. A cada dia de uma *Sprint* a equipe faz uma breve reunião com todos os membros em pé, chamada *Daily Scrum*. O objetivo desta reunião é disseminar conhecimento sobre o que foi feito no dia anterior, identificar impedimentos e priorizar o trabalho do dia que se inicia.

Ao final de uma *Sprint*, a equipe apresenta as funcionalidades implementadas em uma *Sprint Review Meeting*. Por fim, faz-se uma *Sprint Retrospective* e a equipe parte para o planejamento do próximo *Sprint*. Assim, reinicia-se o ciclo (desenvolvimentoagil, 2022).

2.1.3. Time Scrum

Para que um projeto seja iniciado seguindo o método Scrum, certos integrantes são fundamentais na equipe, tais como: o *Product Owner*, uma equipe de desenvolvedores e um *Scrum Master* (Schwaber e Sutherland, 2020).

O *Product Owner* é o principal responsável por desenvolver e comunicar explicitamente o objetivo do produto, criar e ordenar claramente o *backlog* do produto garantindo que o mesmo seja transparente, visível e compreendido pela equipe. O mesmo também é responsável por dar valor ao produto resultante do trabalho, além de poder delegar responsabilidade para os membros da equipe.

O *Scrum Master* é responsável por garantir a execução do Scrum. Ele ajuda todos a entenderem a teoria e a prática do Scrum, tanto dentro do Time Scrum quanto em toda a organização. Ele age como *coach* e líder sobre auto gerenciamento e multifuncionalidade para a equipe, ajuda sempre o time a focar na criação dos incrementos de alto valor e que atinjam o tempo estipulado. Além de ajudar a equipe de desenvolvimento, o *Scrum Master* auxilia o *Product Owner* a explicar à equipe a necessidade de certos itens do *product backlog*.

Os desenvolvedores são os profissionais que tem por objetivo entregar partes funcionais do projeto a cada atualização, podem assim adaptar seu plano diariamente para atingir o objetivo da *Sprint*. Suas habilidades de desenvolvimento são amplas e variam de acordo com o domínio do trabalho. Não há subdivisões claras para cada membro sobre a execução da *Sprint*, todos devem se empenhar em entregar dentro do prazo previsto pelo *backlog*.

2.1.4. Visualização de dados

A visualização de dados é a apresentação visual de dados ou informações com o objetivo de comunicá-los de forma clara e eficaz aos leitores, o que se dá através da união entre arte e ciência. Normalmente, isso acontece na forma de um gráfico, infográfico, diagrama ou mapa (Venngage, 2022).

Um infográfico é uma coleção de imagens, gráficos e texto mínimo que fornece uma visão geral e fácil de entender sobre um assunto. Um gráfico é uma representação gráfica de informações, que utilizam símbolos visuais como linhas, barras, pontos, fatias e ícones para representar pontos de dados. O diagrama é uma representação visual da informação podendo ser bidimensionais e tridimensionais. Tais diagramas são usados para mapear processos, ajudar na tomada de decisões, identificar as raízes de um problema, conectar ideias e planejar projetos. Um mapa é uma representação visual da área de um terreno. Os mapas mostram características físicas, como regiões, paisagens, cidades, estradas e corpos de água (Venngage, 2022).

A visualização de dados pode ajudar a contar a história, transmitindo claramente problemas complexos. Pode desempenhar um papel fundamental para identificar as informações significativas do ruído, incluindo discrepâncias e anomalias. Pode ajudar com o crescente volume de dados. A interação visual com grandes conjuntos de dados pode simplificar a análise, revelando novas informações de negócios (Oracle, 2022).

2.1.5. Padrão de design de interação

Para o desenvolvimento do *design* de interface da aplicação, foi utilizado 'As dez heurísticas de Nielsen', que consistem em:

1. Visibilidade do estado do sistema;
2. Correspondência entre o sistema e o mundo real;
3. Liberdade e controle do usuário;
4. Consistência e padrões;
5. Prevenção de erros (design defensivo);
6. Reconhecimento em vez de memorização;
7. Flexibilidade e eficiência de uso;
8. Estética e design minimalista;
9. Ajudar os usuários a reconhecerem, diagnosticarem e recuperarem-se de erros;
10 . Ajuda e documentação.

Desenvolvida em 1990 por Jakob Nielsen, esse é um padrão utilizado frequentemente para melhorar a qualidade das soluções de *design*, e procura facilitar a comunicação entre o usuário e o *software*. São princípios de avaliação da usabilidade de interfaces, e podem ser aplicadas em qualquer momento do desenvolvimento (Nielsen, 1990).

2.2. Análise de ferramentas similares

A análise foi feita com o propósito de verificar aspectos importantes de cada uma das ferramentas para o gerenciamento de projetos Scrum, sendo a análise realizada com ferramentas gratuitas e também proprietárias, através de versões de demonstração (versão *trial*). O Quadro 1 apresenta os dados obtidos e em seguida um texto breve sobre cada um dos aplicativos.

Foram avaliados os seguintes aspectos: inclusão de colaboradores para participar do projeto (Ad. Membros), possibilidade de visualização de dados no projeto (Visualização de dados), gerenciamento de eventos como *Sprints* (G. *Sprints*), *Daily Scrum* (R. Daily) e retrospectivas (R. Retros.), gerenciamento de tarefas relacionadas aos requisitos (G. Tarefas), para que ao menos simulasse o funcionamento de uma *Sprint*. Designação de funções para os colaboradores (G. Equipe), avaliação de progresso do projeto (Progresso), se a ferramenta possui uma versão gratuita(Gratuito), pois todas possuem planos pagos. As funcionalidades não gratuitas foram avaliadas pela versão de testes disponibilizadas pelas aplicações.

Em todas as ferramentas, foi simulado a criação e o rápido andamento de um projeto para avaliar o escopo desta aplicação, para que as principais funcionalidades fossem implementadas de acordo com as restrições de desenvolvedores, tempo e recursos investidos, bem com para atender às características do modelo Scrum.

O Trello (2022) é uma ferramenta para gerenciamento de projetos Scrum, mas seu foco maior é trabalho com o método *Kanban*[4]. É possível a criação ilimitada de projetos

[4]Trata-se de um sistema visual que busca gerenciar o trabalho conforme ele se move pelo processo

Ferramentas	*Ad. Membros*	*Visualização de dados*	*G. Sprints*	R. Daily	R. Retros.	*G. Tarefas*	*G. Equipe*	*Progresso*	*Gratuito*
Trello (2022)	x		x			x	x	x	x
IceScrum (2022)	x	x	x	x	x	x	x	x	
ScrumHalf (2022)			x			x		x	
Taiga (2022)	x		x	x	x	x	x	x	
Asana (2022)	x		x	x	x	x	x	x	x
Jira (2022)	x	x	x	x	x				
Wrike (2022)		x	x	x	x	x	x	x	x
YouTrack (2022)	x		x		x	x		x	x
Azure DevOps (2022)	x	x	x	x	x	x	x	x	
Marim e Pinheiro (2022)	x	x	x	x	x	x	x	x	x

Quadro 1. ferramentas simulares

(quadros no Trello), fácil criação de tarefas e a documentação de eventos, porém, como o aplicativo não foi feita especialmente para o Scrum, sendo necessário criatividade para organizar o quadro de forma que ele funcione para o gerenciamento de um projeto. Sua parte de visualização de dados é bem completa, porém só é disponível em sua versão paga.

O IceScrum (2022) oferece a grande maioria dos recursos relacionados ao Scrum, apenas não oferecendo a criação de projetos ilimitados em sua versão gratuita. A ferramenta possibilita a adição de colaboradores ao projeto, sendo possível definir suas funções ao adicioná-los, sua participação e interação no quadro de tarefas. Em seu *dashboard*, existem áreas especificas para definições de *Sprints*, retrospectivas, criação de *User Stories*, e de reuniões. Em especial, as reuniões possuem integração com vários aplicativos de comunicação.

O ScrumHalf (2022) oferece apenas recursos básicos quanto o método Scrum, como o gerenciamento de requisitos, *User Stories* e de eventos. O *software* deixa a desejar quanto a aparência de sua interface, não sendo agradável ao usuário e não utilizando-se de conceitos *UX e UI* , além de possuir *bugs* em sua interface, como *URLs* que levam a páginas de erro.

O Taiga (2022) é bastante completo para o gerenciamento de projetos Scrum, que permite a criação e gerenciamento de tarefas, requisitos e eventos por parte de vários colaboradores, onde cada um tem sua função, possuindo formas de acompanhar o progresso do projeto. Um destaque para sua interface que é agradável para o usuário e de fácil entendimento para o uso de suas funções.

O Asana (2022) é uma ferramenta com muitos recursos que abrangem o método Scrum, possuindo várias funções que permitem o gerenciamento do projeto, em seus requisitos, tarefas e eventos. A ferramenta possui uma interface que mostra várias informações importantes para todos que participem de um projeto Scrum, além de permitir a colaboração entre os membros. Em especial, sua interface e formas de acompanhar o andamento e progresso do projeto são pontos muito positivos da ferramenta.

Atlassian (2022) é uma ferramenta que atende amplamente ao método Scrum, tem uma interface agradável, tem expansão para outros modelos como o Kanban, é uma ferramenta gratuita, mas tem como deficiência de não ter uma visualização gráfica do progresso dos projetos.

O Wrike (2022) abrange grande parte dos recursos necessários para o gerenciamento de projetos Scrum. Ele possui funcionalidades para a criação e organização de tarefas, a colaboração e aprovação de tarefas de membros da equipe e o gerenciamento de

eventos.

O JetBrains (2022) é uma ferramenta que abrange vários dos recursos usados para o gerenciamento de um projeto Scrum, deixando a desejar apenas no registro de reuniões *Daily Scrum* e no gerenciamento de equipe, pois não permite designar funções para os colaboradores adicionados. Um de seus pontos fortes é a visualização de dados para o progresso do projeto, que permite a geração de vários tipos de gráficos. É uma ferramenta muito boa para uso com o método Scrum, mas que também é paga.

O Azure DevOps é um serviço da Microsoft (2022) que permite o gerenciamento de seus projetos a partir do método Scrum, sendo uma das ferramentas mais completas disponíveis no mercado, oferecendo a adição de membros para gerenciamento do projeto, uma interface agradável que utiliza conceitos de *UX/UI*, o registro e gerenciamento de todas as atividades e eventos do Scrum, mostrando o progresso de *sprints*, além de sua integração com outros serviços como o *Microsoft PowerBi*. Apesar de oferecer o gerenciamento completo de projetos Scrum, é um serviço pago.

Quanto a ferramenta a ser desenvolvida neste projeto, ela procura cobrir todas as funções utilizadas como parâmetros de avaliação para outras ferramentas, como demonstrado no quadro 1, e procurando utilizar os conceitos apresentados neste capítulo.

3. Desenvolvimento

Esta seção está dividida em quatro partes. A primeira parte descreve o processo de desenvolvimento ágil adotado para cumprir com o objetivo deste trabalho. A segunda descreve as principais tecnologias e apresenta um resumo das ferramentas utilizadas para acompanhamento e controle. A parte três detalha o desenvolvimento do back-end necessário para dar suporte ao armazenamento e consulta de dados. A parte quatro apresenta um conjunto de interfaces gráficas, bem como as funcionalidades implementadas nelas.

3.1. Planejamento

Em razão deste projeto ter como objetivo a construção de uma ferramenta para gerenciamento de projetos Scrum, o próprio modelo Scrum foi aplicado como metodologia de desenvolvimento. O professor orientador desempenhou o papel de *product owner* e o papel de *scrum master*. O grupo de alunos formaram a equipe de desenvolvimento.

3.1.1. Organização do *Product Backlog*

Conforme o modelo Scrum pressupõe, o *Product Backlog* deve ser composto por uma relação de requisitos a serem implementados. A relação de requisitos deve estar priorizada, sendo o primeiro o mais importante, ou seja, aquele que deve ser implementado em uma *sprint* inicial.

Os requisitos da ferramenta de gerenciamento foram obtidos a partir do Quadro 1 (Seção 2 - Referencial Teórico). O Quadro 2 apresenta o *product backlog* da ferramenta para gerenciamento de projetos de *software*.

Além disso, de acordo com o modelo Scrum, quem define e atualiza o *product backlog* é o *product owner*. Ele possui a liberdade de alterar a relação de requisitos quando necessitar, desde que os itens alterados não estejam ou façam parte de uma *sprint*.

	Product backlog
#1	Gerenciar projeto (CRUD)
#2	Gerenciar membros (CRUD)
#3	Gerenciar equipe (CRUD)
#4	Gerenciar atividades (CRUD)
#5	Gerenciar backlog (CRUD)
#6	Gerenciar eventos (CRUD)
#7	Criar Dashboard

Quadro 2. *Product Backlog - Relação de Requisitos*

3.1.2. Planejamento das *Sprints*

As sprints ou ciclos de desenvolvimento foram organizadas da seguinte maneira. Cada *sprint* durou cerca de duas semanas. No início de cada *sprint*, o *product owner* informou o conjunto de requisitos a serem desenvolvidos.

Para uma maior compreensão dos requisitos a desenvolver, foram utilizadas histórias de usuário (*User stories*). Histórias são descrições textuais do funcionamento de um requisito, expressando sempre um desejo (algo importante para) de um usuário. As histórias seguiram um padrão de descrição, conforme o que segue:

Como um ”de usuário”, eu quero ”algum objetivo”.

Por exemplo, segue história de usuário para o primeiro requisito: Gerenciar Projetos (CRUD).

Como um gerente de um projeto de software, eu quero inserir um novo projeto, informando a data de início, o cliente e seu contato, a lista de requisitos a serem implementadas e associar membros para participar do projeto, com suas respectivas funções.

Na sequência, com base nas histórias de usuários, o *Scrum master* e os desenvolvedores listaram um conjunto de tarefas necessárias para desenvolver um determinado requisito. Segundo o modelo Scrum, para cada uma das tarefas, uma estimativa de tempo de desenvolvimento foi associada. Neste trabalho, a técnica de *planning poker* foi utilizada. Em linhas gerais, a técnica requer que os membros da equipe informem estimativas de tempos e esforço até convergirem, ou seja, as estimativas foram sendo apresentadas e discutidas até que a equipe chegou a um consenso.

O quadro 3 apresenta um exemplo do *sprint backlog* para o requisito Gerenciar Projetos (CRUD).

Sprint backlog - sprint # 1	
Gerenciar Projetos (CRUD)	
Tarefa	**Tempo estimado (horas)**
Criar base de dados	2
Definir tabela de projeto	1
Criar método para inserção	4
Criar método para consulta	3
Criar exclusão	2
Criar alteração de projeto	3
Testar a API	2

Quadro 3. Gerenciar Projetos (CRUD)

Ao final de cada *sprint*, de acordo com o modelo Scrum, duas atividades foram realizadas:

- Revisão da *sprint*;
- Retrospectiva da *sprint*.

A revisão da *sprint* foi coordenada pelo *product owner*, acompanhado do *Scrum master* e dos desenvolvedores. A revisão de cada *sprint* foi composta das seguintes tarefas:

- Apresentar o que foi desenvolvido;
- Entregar uma versão funcionando do software.

Na retrospectiva de cada uma das *sprints*, participam o *scrum master* e os desenvolvedores. Neste evento, pontos positivos e negativos associados a uma *sprint*, foram identificados e discutidas possíveis soluções para serem aplicadas em *sprints* posteriores.

3.2. Ferramentas e Tecnologias

Esta seção contém as informações sobre as tecnologias utilizadas para o desenvolvimento da aplicação, assim como as suas vantagens e justificativas de utilização.

3.2.1. Visual Studio Code

O Visual Studio Code é mais do que apenas um editor, ele possui muitos recursos que permitem executar com eficiência diversas tarefas relacionadas à programação e desenvolvimento de sistemas. O grupo de software que desenvolve a programação é chamado de ambiente de desenvolvimento. O Visual Studio Code é um dos principais ambientes de desenvolvimento usados por muitas linguagens de programação (Academy, 2020).

O Visual Studio Code será de suma importância para auxiliar na programação e desenvolvimento do sistema. Será utilizado tanto para a codificação do *front-end* como para o *back-end*.

3.2.2. GitHub

O Git é um tipo de ferramenta de controle de versão de arquivos, que foi criado para gerenciar quem e quando foi editada a versão do código-fonte, qual versão é a mais recente e assim por diante. Um recurso do Git é que o controle de versão, como gravação e rastreamento de código-fonte. É mais fácil para os engenheiros reverter o código-fonte, organizar o histórico de modificações e registrá-lo, tornando-o mais conveniente para os engenheiros (Modis, 2020).

O GitHub é uma plataforma para utilização do Git, e que foi utilizado no desenvolvimento da aplicação deste estudo. Neste trabalho sua função seria criar um histórico de cada etapa que estará sendo desenvolvida, para, se necessário, podermos retroagir a versão caso necessário.

3.2.3. Figma

O Figma é uma ferramenta online para design vetorial de interfaces e protótipos. Destaca-se por ser gratuito, colaborativo com alterações em tempo real, operada através do próprio navegador, sem a necessidade de instalar um novo software em seu computador. Ele será essencial para a prototipação da interface gráfica da aplicação, sendo possível ter uma noção mais precisa de como ficará o projeto das interfaces do *front-end*.

3.2.4. Django

Django é um *framework web Python* de alto nível que incentiva o desenvolvimento rápido e um design limpo e pragmático DjangoWebsite (2018), que permite criar aplicações *web* com muitas facilidades. É gratuito e de código aberto, tem uma comunidade próspera e ativa, ótima documentação e muitas opções de suporte gratuito e pago (MozillaContributors, 2022).

3.2.5. Django Rest Framework

Django REST *Framework* é uma biblioteca que permite a construção de APIs REST utilizando a estrutura do Django, permitindo a construção de APIs em qualquer sistema operacional. É muito utilizado por toda a comunidade, pois provê uma forma simples e rápida para a construção de APIs utilizando recursos o sistema de rotas e seu mapeamento objeto-relacional para manipulação de banco de dados (de Andrade, 2020).

3.2.6. SQLite

SQLite é uma é uma biblioteca de linguagem C que implementa um mecanismo de banco de dados SQL pequeno, rápido, independente, de alta confiabilidade e completo SQLiteWebsite (2022). Funciona como um servidor próprio e independente, já que o SGBD (Sistema de Gerenciamento de Banco de Dados) pode ser executado na mesma instância,

o que elimina consultas e processos separados. Portanto, a biblioteca SQLite é gerada e armazenada diretamente no arquivo do banco de dados (Vieira, 2022).

3.2.7. PostGreSQL

O PostgreSQL é uma ferramenta que atua como sistema de gerenciamento de bancos de dados relacionais. Seu foco é permitir implementação da linguagem SQL em estruturas, garantindo um trabalho com os padrões desse tipo de ordenação dos dados (de Souza, 2020). Um de seus pontos principais é sua adequação em padrões de conformidade, ajudando a construir bancos de dados otimizados. O PostgreSQL, neste trabalho, será responsável pelo gerenciamento e armazenamento da camada de persistência, principalmente em razão da experiência dos autores com o seu uso.

3.2.8. NodeJS

Node é uma plataforma de aplicação, na qual os programas são escritos em JavaScript e são compilados, otimizados, e por fim, interpretados pela Máquina Virtual V8 de Freitas Escudelario (2019), o que permite criar aplicações que não dependem de um *browser* para sua execução. Além disso, uns de principais motivos de sua adoção é a sua alta capacidade de escala, sua arquitetura, flexibilidade e baixo custo, o que o tornam uma ótima escolha para implementação de Microsserviços (Lenon, 2018).

3.2.9. ReactJS

O ReactJS é uma biblioteca *front-end* e tem como um de seus objetivos facilitar a conexão entre diferentes partes de uma página, portanto seu funcionamento acontece através do que chamamos de componentes (Roveda, 2020). O ReactJS permite que você reuse componentes que tenham sido desenvolvido em outras aplicações, sendo a reusabilidade seja uma vantagem importante para desenvolvedores em geral. Os componentes dessa ferramenta foram desenvolvidos pelo Facebook, e foi lançada em 2013 como uma ferramenta JavaScript de código aberto (L., 2021).

3.3. Diagramas de caso de uso, modelo de banco de dados e protótipo de telas

Para uma mais clara compreensão dos requisitos da aplicação proposta, três diagramas de casos de uso da *Unified Modeling Language* (UML) foram desenvolvidos. O diagrama apresentado na Figura 2 mostra quais funcionalidades poderão ser acessadas na perspectiva do *product owner*.

O diagrama de casos de uso foi feito a partir do estudo sobre o funcionamento de projetos Scrum. O diagrama de casos de uso está apresentado nas figuras a seguir:

Figura 2. Casos de uso: Product Owner

O *Product Owner* pode fazer o gerenciamento de projetos, usuários, *product backlog*, eventos, *sprints* e visualizar *dashboards*.

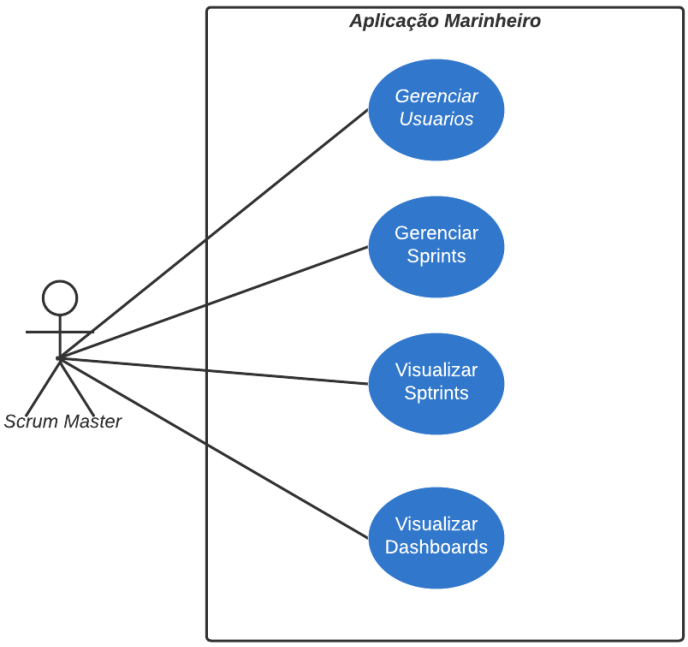

Figura 3. Casos de uso: Scrum Master

O *Scrum Master* pode gerenciar os usuários e eventos, delegar *sprints* à desenvolvedores e visualizar *dashboards*.

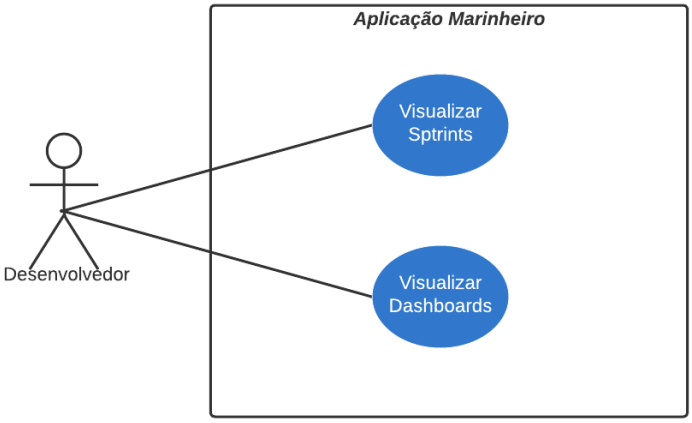

Figura 4. Casos de uso: Desenvolvedor

Os desenvolvedores terão acesso à gerenciar os eventos aos quais estão inseridos e visualizar *dashboards*.

A modelagem do banco de dados foi feita a partir de requisitos e das interfaces prototipadas. O modelo conceitual está apresentado na figura 5.

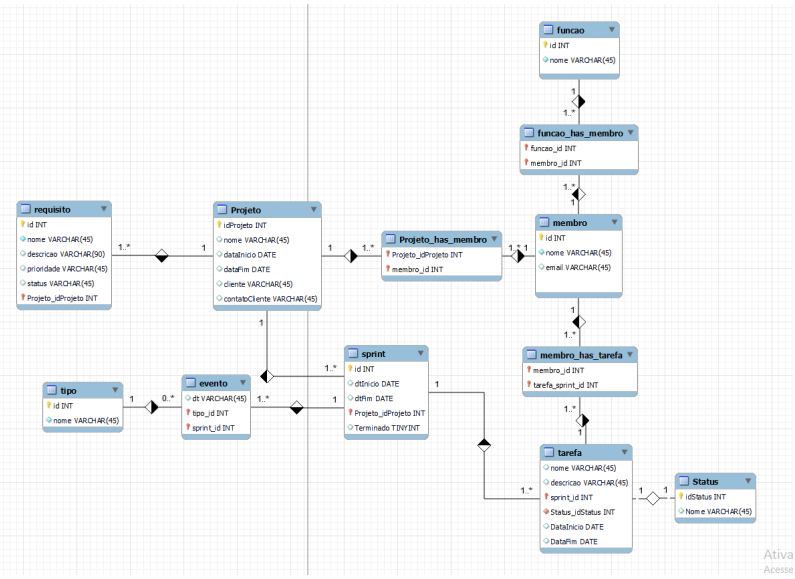

Figura 5. Modelo conceitual

3.4. Desenvolvimento do Front-end

As Figuras 6 a 9 apresentam as telas que foram desenvolvidas dentro deste projeto utilizando da biblioteca ReactJS para auxiliar na criação das interfaces de usuário.

Tais interfaces foram criadas seguindo as heurísticas de Nielsen, conforme apresentado na seção 2.1.5.

A primeira heurística foi comprida trazendo notificações para cada ação de cadastro e edição do usuário de informações além de trazer gráficos para visualização de informações de um projeto vigente;

A segunda heurística foi cumprida utilizando de conceitos e termos comumente usados em projetos usando o Scrum, como *sprint*, *produck backlog*, *Scrum Master*, etc;

A terceira heurística foi cumprida dando fácil acesso à quem utiliza o software de navegar entre as telas, de cadastrar e editar informações.

A quarta heurística foi cumprida criando componentes padrões de janelas flutuantes, caixas de texto e tabelas. Padrozinado tamanho de fontes e cores do software.

A quinta heurística foi cumprida utilizando de notificações para sinalizar o usuário ao executar ações, de cadastrar e editar informações.

Antes do desenvolvimento ser iniciado foram feitos protótipos utilizando a ferramenta Figma. Os protótipos serviram de base para o desenvolvimento da camada de apresentação da aplicação, além de ajudar a criar componentes e visualizar como ficaria a aplicação.

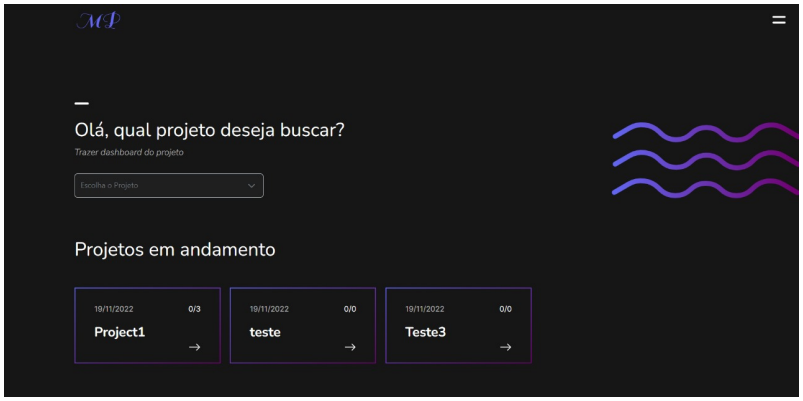

Figura 6. Tela inicial

A tela inicial referente à figura 6 tem como objetivo apresentar a lista de projetos em andamento, com informações de data de início, número de tarefas finalizadas por total de tarefas, para que, com isso, tenha uma visualização rápida sobre conclusão do projeto. A tela inicial possui também uma caixa de texto com filtro para selecionar um projeto ao qual deseja ser visualizado.

Cada item da listagem é clicável para que ao clicado, o usuário seja redirecionado para tela de *backlog* do projeto.

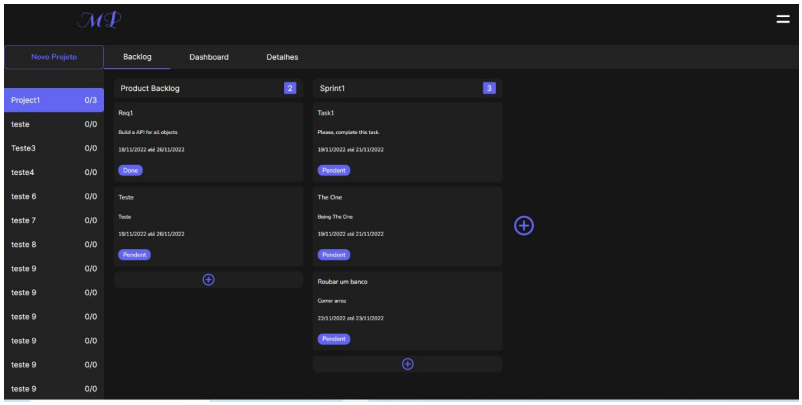

Figura 7. Tela do Projeto vigente

A tela de projeto vigente referente a figura 7, tem como objetivo listar os requisitos do *product backlog* e listar as tarefas por *sprints*, dando a possibilidade de cadastrar e editar requisitos e *sprints* com tarefas, aos quais é definido nome, data de início e fim, descrição, status de conclusão e desenvolvedores envolvidos.

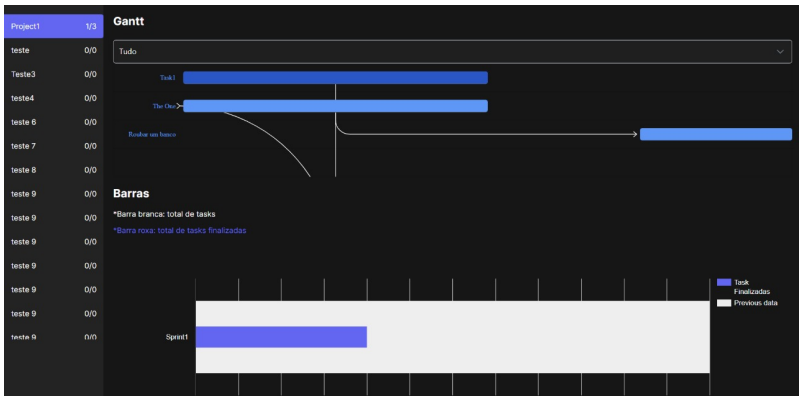

Figura 8. Tela de dashboard do projeto

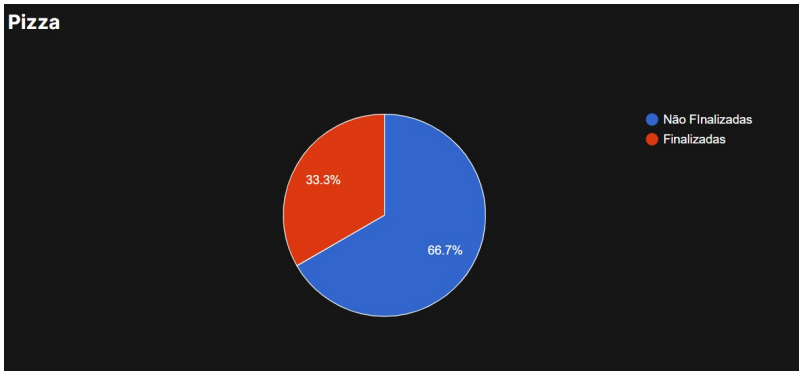

Figura 9. Tela de dashboard do projeto

A tela de *dashboard* referente às figuras 8 e 9 traz consigo três tipos de gráficos: gráfico de Gantt, gráfico de Barras horizontal e gráfico de pizza.

O gráfico de Gantt auxilia em ilustrar o avanço das diferentes etapas do projeto com base nas *sprints* e tarefas cadastradas, podendo ser filtrado por *sprints* ou trazer todas as etapas do projeto.

O gráfico de barras foi utilizado para mostrar o nível de conclusão das *sprints* com base no status de suas respectivas tarefas, dando assim essa informação mais acessível ao usuário.

O gráfico de pizza traz consigo a informação visual em porcentagem sobre o total de tarefas finalizadas e não finalizadas.

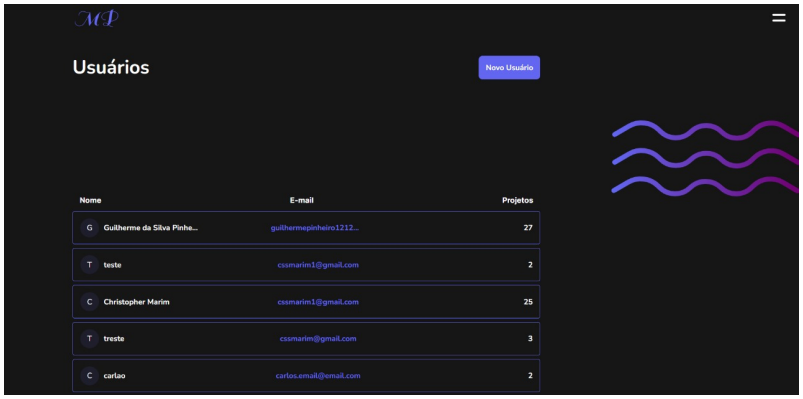

Figura 10. Tela de lista de usuários

A tela de lista de usuários referente a figura 10, informa em uma tabela, o nome dos usuários, seu e-mail e o número de projetos em que está envolvido. Cada linha referente ao usuário é clicável para que, com tal ação, apareça uma janela flutuante na tela para edição de informações do usuário.

3.5. Desenvolvimento do back-end

Após a criação dos protótipos das interfaces gráficas e a validação das mesmas, partiu-se o desenvolvimento do back-end da aplicação. O primeiro passo, foi a criação do projeto através da utilização do *Django* e do *Django Rest Framework*, que cria como padrão um projeto com uma arquitetura (*Model-View-Template*) MVT. Porém, para esse projeto, utilizou-se apenas a parte de modelos e *views* para a criação de uma API, para comunicar com o *front-end* feito em um projeto separado. De acordo com o padrão, foram criados três arquivos: *models*, *views* e *urls*. Eles são os principais arquivos que foram modificados para o desenvolvimento da API, pois neles são armazenados os dados sobre a aplicação no *back-end*. Também foi criado um arquivo para realizar a conexão com o banco de dados SQLite. O arquivo *models.py* contém as informações sobre os objetos e suas relações, o arquivo *views.py* contém as chamadas de *GET, POST, PUT e DELETE* de cada um dos objetos, e por fim, o arquivo *urls.py* é onde são definidas as rotas por onde os recursos disponibilizados pela API possam ser acessados.

A escolha do *Django Rest Framework* no projeto, se deu por três razões importantes. A primeira é pelo fato dos autores já possuírem experiência no uso deste padrão. Com isso, ganha-se em tempo de desenvolvimento. A segunda razão reside no fato da separação do *back-end* e *front-end* para que pudessem serem desenvolvidas de forma independentes. A terceira razão é pela facilidade do *framework*, primeiro na criação de

banco de dados, que cria toda a estrutura do banco a partir de como são construídos os modelos em código, e segundo na criação das chamadas básicas de *CRUD* de cada dos objetos.

A API criada recebe e envia dados através de documentos *json* para suas requisições, seja para receber dados que irão ser usados para alterações dentro do sistema, quanto para enviar informações de dados dos objetos. Isso é usado também pelo *front-end* que utiliza os arquivos de resposta *json* para mostrar as informações vindas no *back-end*. No arquivo *views.py*, foram criadas funções *CRUD* para todos os objetos, porém, foram feitas algumas modificações para adição de parâmetros na resposta *json*. Um exemplo disso é mostrado na figura 11. Cada uma das rotas em *url* são definidas em um arquivo *urls.py*, onde concentram as rotas de todas as chamadas da API, conforme é mostrado na figura 12. Isto, permite uma melhor organização do código e uma mais fácil manutenção e evolução do *back-end*.

```python
@action(detail=True, methods=['get'])
def getProject(self, request, pk):
    project = Project.objects.get(id=pk)
    sprints = Sprint.objects.filter(project=project.id)
    totalTasks = 0
    totalFinished = 0
    for sprint in sprints:
        tasks = Task.objects.filter(sprint = sprint.id)
        totalTasks += len(tasks)
        tasksDone = Task.objects.filter(sprint = sprint.id, status = 2)
        totalFinished += len(tasksDone)

    projectJson = {
        'id' : project.id,
        'name' : project.name,
        'dateBegin' : project.dateBegin,
        'dateFinished' : project.dateFinished,
        'totalTasks' : str(totalTasks),
        'totalFinished' : str(totalFinished),
        'clientName' : project.client,
        'clientContact' : project.contactClient,
        }
    return Response({
                    'success' : True,
                    'project' : projectJson})
```

Figura 11. Exemplo código de chamada *GET* de 'Projeto'

```python
router = routers.DefaultRouter()
router.register('member', views.MemberViewSet)
router.register('function', views.FunctionViewSet)
router.register('sprint', views.SprintViewSet)
router.register('event', views.EventViewSet)
router.register('eventtype', views.EventTypeViewSet)
router.register('task', views.TaskViewSet)
router.register('requirement', views.RequirementViewSet)
router.register('project', views.ProjectViewSet, basename='project')
router.register('status', views.StatusViewSet)
```

Figura 12. Exemplo código de rotas para chamadas da *API*

Após a conclusão do desenvolvimento da API, foi feito o *deploy* da aplicação

através do *Railway*. Para o *deploy*, foi criada uma conta gratuita e registrado o projeto da API. O código usado como base, encontra-se armazenado no *GitHub*, usado também para fazer o versionamento do código durante o desenvolvimento. Para a camada de armazenamento funcionar, no *Railway*, foi instalado uma extensão que permite o uso do *SQLite*. Depois, foi importado o banco de dados utilizado durante o desenvolvimento.

Após a finalização de todas as histórias de usuário sobre o desenvolvimento do *back-end*, foi feito uma reunião com os orientandos e o orientador para a entrega da API. Nessa reunião, foi constatado que seria necessário realizar alguns ajustes quanto ao banco de dados, para que ele refletisse corretamente as relações entre os objetos. Os ajustes foram feitos ao longo de uma *sprint* posterior. Após a reunião de validação, iniciou-se o desenvolvimento do *front-end*, na subseção 3.4.

4. Avaliação da Aplicação

Esta seção trata da avaliação da Aplicação. Para a realização da avaliação, foi criado um formulário utilizando o *Google Forms*, considerando sua facilidade de uso e envio pela *internet* dos formulários. A aplicação foi avaliada por desenvolvedores e testadores de uma empresa de software, que já possuem uma familiarização com os conceitos de Scrum e os utilizam diariamente em seu ambiente de trabalho. As perguntas propostas pelo questionário são:

1. O sistema está adequado (atende) às suas necessidades?;

2. O sistema possui características (interfaces intuitivas, vocabulário conhecido pelo usuário etc) que permitem que ele seja aprendido com facilidade?;

3. O sistema é atrativo, ou seja, as interfaces estão bem projetadas, os componentes de menu são fáceis de localizar e as cores são adequadas?.

4. As funcionalidades que você acessou e utilizou funcionaram de maneira correta?;

5. O dashboard gerado pelo software apresenta informações organizadas, importantes e permite visualizar de maneira mais fácil e rápida o andamento de um projeto?;

6. Informe o que você mudaria ou aquilo que você não gostou no sistema;

7. Informe o que você gostou no sistema;

8. Qual nota geral você daria para o sistema?;

As primeiras 5 perguntas possuem uma escala com cinco alternativas e cada alternativa possui um valor associado à mesma. As alternativas são:

1. Concordo totalmente;

2. Concordo;

3. Não sei responder;

4. Discordo;

5. Discordo totalmente;

Já as perguntas 6 e 7, são perguntas abertas, onde o avaliador da aplicação tem a oportunidade de escrever sobre o que o agradou e o que o desagradou na aplicação. A

última pergunta questiona qual nota o avaliador da ao *software*, utilizando uma escala de um até dez.

4.1. Análise de resultados

Nesta subseção, são analisados os resultados sobre cada uma das questões pertencentes ao questionário de avaliação e as conclusões a serem feitas diante das respostas dos entrevistados.

O questionário foi respondido por dez pessoas. A primeira pergunta foi sobre se o sistema atende às necessidades para o gerenciamento de um projeto Scrum. A figura 13 mostra os resultados obtidos e que todos os entrevistados concordam que o sistema é capaz de ser usado para tal fim.

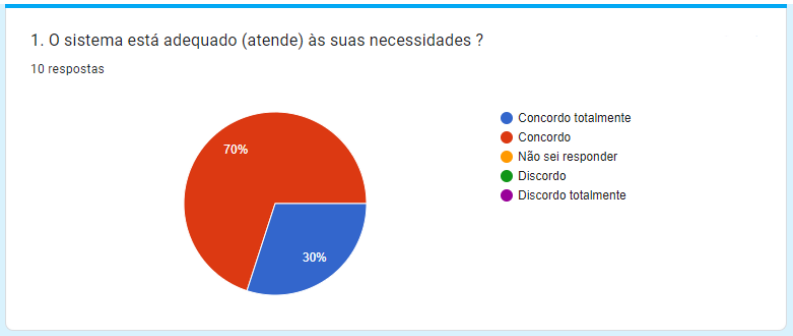

Figura 13. Respostas entrevistados à primeira pergunta do questionário.

A segunda e a terceira pergunta tem o objetivo de verificar se o sistema é intuitivo e atrativo para o usuário e se é fácil de utilizar o sistema diante de como foi feito seu *design*. A figura 14 mostra o gráfico com as respostas e que os usuários concordam que o sistema é atrativo e possui características que permitem que ele seja de fácil usabilidade.

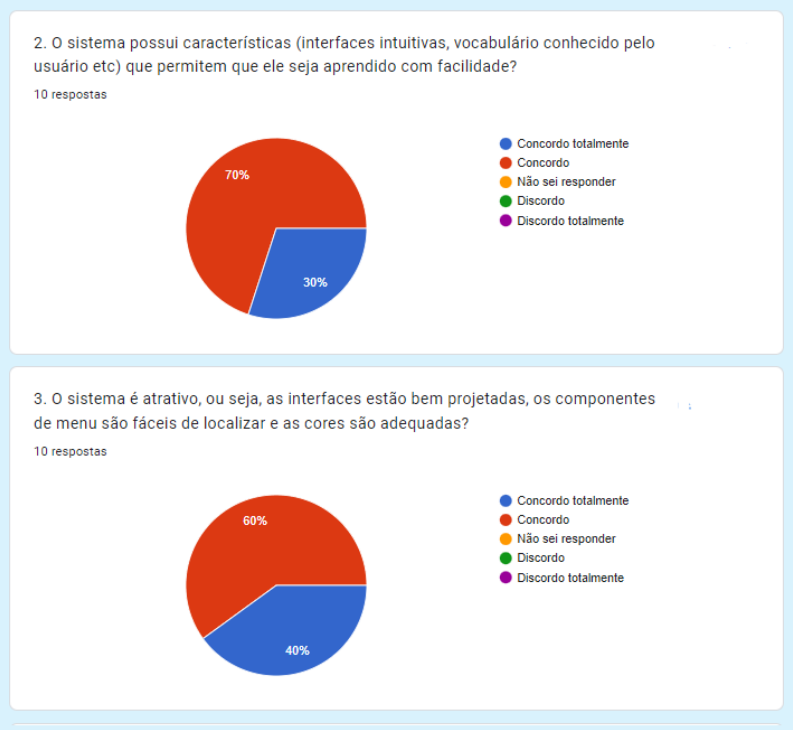

Figura 14. Respostas entrevistados à segunda e terceira perguntas do questionário.

A quarta pergunta procura saber se as funcionalidades do sistema funcionam de maneira correta. A figura 15 mostra os resultados dessa questão e demonstra que noventa porcento dos entrevistas responderam que concordam com a pergunta, sendo apenas um entrevistado que não soube responder.

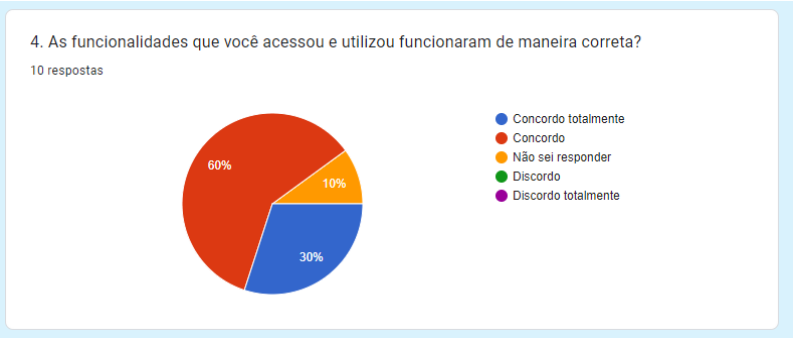

Figura 15. Respostas entrevistados da quarta pergunta do questionário.

A quinta pergunta é referente ao uso do *dashboard* gerado pelo software, se as informações apresentadas por ele são consideradas importantes pelo usuário, se permitem que dados dos projeto sejam visualizados de maneira fácil e rápida através dos gráficos. Segundo a figura 16, noventa porcento dos usuários concordam que o *dashboard* atende à necessidade de apresentação de informações do projeto.

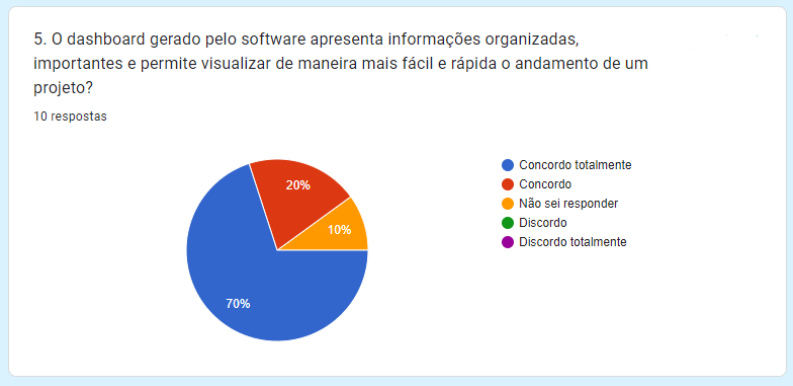

Figura 16. Respostas entrevistados quinta pergunta do questionário.

A sexta e sétima perguntas, são questões discursivas quanto ao que os entrevistados gostaram, mudariam ou não gostaram do sistema. Quanto ao que os usuários não gostaram ou mudariam, são destacados o que segue:

- A responsividade para dispositivos móveis;
- Ícones mais intuitivos para criação de novas tarefas;

Quanto ao que os usuários gostaram no sistema, suas principais considerações são sobre o *design* da aplicação e sua usabilidade e também quanto aos *dashboards* apresentados no projeto.

A última pergunta procura saber em uma escala de zero até dez, uma nota geral que os participantes do questionário dariam à aplicação. Segundo a figura 17, todos os participantes deram uma nota sete ou maior ao sistema, o que demonstra a satisfação quanto à aplicação e seu uso.

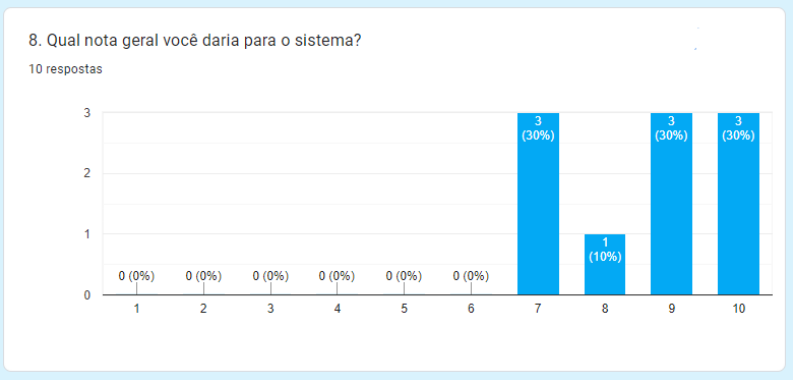

Figura 17. Respostas dos entrevistados para a oitava pergunta do questionário.

Veja o apêndice A para ter acesso à todas as respostas do formulário.

5. Conclusões

Este trabalho teve como objetivo desenvolver e implementar um sistema de gerenciamento de projetos Scrum, utilizando *dashboards* para a apresentação de informações e dados

relacionados. O foco maior do projeto reside no desenvolvimento de uma aplicação web, procurando entregar uma experiência agradável de uso ao usuário.

O Scrum é uma metodologia ágil de extrema importância para o desenvolvimento de *software* e é utilizado amplamente por desenvolvedores. A ferramenta desenvolvida é gratuita e procura entregar usabilidade para os desenvolvedores utilizarem a metodologia Scrum, podendo também aproveitar de uma funcionalidade de *dashboards*, que facilita a visualização e entendimento de dados e informações de projetos.

A utilização do Scrum no desenvolvimento do trabalho mostrou grande eficácia, pois, em razão do tempo definido para cada sprint, houve uma boa organização das tarefas e do tempo para o desenvolvimento da aplicação. Além disso, o uso do Scrum forneceu meios para que algumas dificuldades identificadas, durante o desenvolvimento, tais como: i) *deploy* da aplicação, ii) definição e escolha de gráficos para o *dashboard* e iii) alterações na API disponibilizada pelo *back-end*, não afetassem o prazo de entrega da aplicação. Ainda, ressalta-se que as reuniões realizadas permitiram uma melhor comunicação entre os envolvidos no projeto, com isso foi possível uma melhor compreensão dos requisitos.

A partir da avaliação realizada com a aplicação, conforme está na seção 4, os resultados dos questionários foram satisfatórios, uma vez que a maioria dos avaliadores se mostrou satisfeito com a utilização da ferramenta. A aplicação funciona e atende aos objetivos estabelecidos. Porém, dois aspectos importantes merecem ser destacados para serem desenvolvidos em atividades futuras, sendo eles:

- Autenticação de usuários no sistema;
- Gerenciar a parte de eventos ou cerimônias do Scrum.

A. Apêndice 1

Link para documento com todas as respostas recebidas na avaliação do *software* : https://drive.google.com/file/d/1BjR90P8HBDYYS-Pr17UCzhVLJ7Rvpo9X/view?usp=sharing

Referências

Academy, T. (2020). Visual studio codevscode. Disponível em: https://techacademy.jp/magazine/39548, Acesso em: 23 jun. 2022.

Asana (2022). Ferramenta asana. Disponível em: https://asana.com/, Acesso em: 15 abr. 2022.

Atlassian (2022). Ferramenta jira. Disponível em: https://www.atlassian.com/software/jira, Acesso em: 04 jun. 2022.

de Andrade, A. P. (2020). O que é o django rest framework? Disponível em: https://www.treinaweb.com.br/blog/o-que-e-o-django-rest-framework, Acesso em: 12 nov. 2022.

de Freitas Escudelario, B. (2019). Vale a pena aprender nodejs? Disponível em: https://imasters.com.br/back-end/vale-pena-aprender-nodejs#:~:text=Como%20muitos%20ainda%20pensam%2C%20Node,uma%20nova%20linguagem%20de%20programa%C3%A7%C3%A3o, Acesso em: 07 jul. 2022.

de Souza, I. (2020). Postgresql: saiba o que é, para que serve e como instalar. Disponível em: https://rockcontent.com/br/blog/postgresql/, Acesso em: 23 jun. 2022.

desenvolvimentoagil (2022). Scrum. Disponível em: http://www.desenvolvimentoagil.com.br/scrum/, Acesso em: 09 maio 2022.

Dinamize (2022). Scrum – o que é, suas etapas e como funciona na prática. Disponível em: https://www.dinamize.com.br/blog/scrum/, Acesso em: 09 maio 2022.

DjangoWebsite (2018). Django - overview. Disponível em: https://www.djangoproject.com/start/overview/, Acesso em: 12 nov. 2022.

IceScrum (2022). Ferramenta icescrum. Disponível em: https://www.icescrum.com/, Acesso em: 15 abr. 2022.

JetBrains (2022). Ferramenta youtrack. Disponível em: https://www.jetbrains.com/youtrack/, Acesso em: 04 jun. 2022.

L., A. (2021). O que é react e como funciona? Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-react-javascript, Acesso em: 07 jul. 2022.

Lenon (2018). Node.js – o que é, como funciona e quais as vantagens. Disponível em: https://www.opus-software.com.br/node-js/, Acesso em: 07 jul. 2022.

Martins, J. C. C. (2007). *Técnicas para gerenciamento de projetos de software*. Brasport Livros e Multimídia Ltda. São Paulo, 7 edition.

Microsoft (2022). Ferramenta azure devops. Disponível em: https://azure.microsoft.com/pt-br/services/devops/, Acesso em: 04 jun. 2022.

Modis (2020). Github. Disponível em: https://www.modis.co.jp/candidate/insight/column_30, Acesso em: 23 jun. 2022.

MozillaContribuitors (2022). Introdução ao django. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Django/Introduction, Acesso em: 12 nov. 2022.

Nielsen, J. (1990). *Hypertext and Hypermedia*. Academic Press, Boston.

Oracle (2022). O que é a visualização de dados? Disponível em: https://www.oracle.com/br/business-analytics/what-is-data-visualization/, Acesso em: 20 jun. 2022.

Roveda, U. (2020). React: O que é, como funciona e porque usar e como aprender. Disponível em: https://kenzie.com.br/blog/react/, Acesso em: 07 jul. 2022.

Schwaber, K. e Sutherland, J. (2020). O guia do scrum. Disponível em: https://andrelmgomes.com.br/wp-content/uploads/2020/11/Guia-do-Scrum-2020-PT-BR-EN-US-1.pdf, Acesso em: 15 abr. 2022. Traduzido para o Português por André Gomes.

ScrumHalf (2022). Ferramenta scrumhalf. Disponível em: https://myscrumhalf.com/, Acesso em: 15 abr. 2022.

Soares, M. d. S. (2004). Metodologias Ágeis extreme programming e scrum para o desenvolvimento de software. Disponível em: http://periodicosibepes.org.br/index.php/reinfo/article/view/146/38. Acesso em: 18 abr. 2022.

SQLiteWebsite (2022). What is sqlite? Disponível em: https://www.sqlite.org/index.html, Acesso em: 12 nov. 2022.

Taiga (2022). Ferramenta taiga. Disponível em: https://www.taiga.io/, Acesso em: 15 abr. 2022.

Trello (2022). Ferramenta trello. Disponível em: https://trello.com/, Acesso em: 15 abr. 2022.

Valente, M. T. (2022). *Engenharia de Software Moderna*. Independente. Acesso em 21 abr 2022.

Venngage (2022). O que é a visualização de dados? Disponível em: https://pt.venngage.com/blog/visualizacao-de-dados/, Acesso em: 31 maio 2022.

Vieira, D. (2022). Sqlite: o que é, como funciona e qual é a diferença entre o mysql. Disponível em: https://www.hostgator.com.br/blog/sqlite-o-que-e-como-funciona-e-qual-e-a-diferenca-entre-o-mysql/, Acesso em: 12 nov. 2022.

Wrike (2022). Ferramenta wrike. Disponível em: https://www.wrike.com/, Acesso em: 15 abr. 2022.