UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

AFONSO HENRIQUE TOMBINI DE SOUZA

FERRAMENTA WEB PARA GESTÃO DE PROJETOS IMPLEMENTANDO CONCEITOS DO SCRUM

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2022

AFONSO HENRIQUE TOMBINI DE SOUZA

FERRAMENTA WEB PARA GESTÃO DE PROJETOS IMPLEMENTANDO CONCEITOS DO SCRUM

Web tool for project management using scrum concepts

Trabalho de Conclusão de Curso de Graduação, apresentado como requisito para obtenção do título de Técnologo em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Campus Pato Branco

Orientador: Prof. Vinicius Pegorini

PATO BRANCO
2022

AFONSO HENRIQUE TOMBINI DE SOUZA

FERRAMENTA WEB PARA GESTÃO DE PROJETOS IMPLEMENTANDO CONCEITOS DO SCRUM

Trabalho de Conclusão de Curso de Graduação, apresentado como requisito para obtenção do título de Tecnologo em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco

Data de aprovação: 22/novembro/2022

Vinicius Pegorini
Mestrado
Universidade Tecnológica Federal do Paraná

Lucília Yoshie Araki
Mestrado
Universidade Tecnológica Federal do Paraná

Robison Cris Brito
Doutorado
Universidade Tecnológica Federal do Paraná

PATO BRANCO
2022

RESUMO

No desenvolvimento de projetos de software é indispensável atender prazo (cronograma do projeto), custo definido (orçamento monetário e previsão de recursos como pessoas, equipamentos, instalações e outros) e requisitos de qualidade estabelecidos para o produto e que visam atender interesses dos usuários do software. A gestão dos projetos de software tem como objetivo que eles sejam desenvolvidos dentro do tempo e com os recursos estabelecidos e que atendam os interesses dos usuários da melhor forma possível. É fato estabelecido que a qualidade do produto, que é o software, está intrinsecamente relacionada à qualidade do processo empregado no seu desenvolvimento. São diversos os modelos e normas de qualidade definidos para software que visam efetividade dos seus projetos. Alguns deles são bastante custosos de implementar em termos de recursos financeiros e qualificação da equipe e, assim, não se aplicam para projetos ou empresas de desenvolvimento de software de pequeno porte. Agilidade no desenvolvimento com um mínimo de organização das atividades e de controle dos seus resultados é o que as empresas e equipes em geral esperam. Os mecanismos de controle não podem ser demasiado dispendiosos e tempo e de custos porque desencorajam as empresas a segui-los, mas algum controle deve existir para que um planejamento possa ser seguido e resultados alcançados. A proposta de metodologias ágeis como o Scrum é prover uma gestão eficiente do processo de software, sem excesso de controles e rotinas. Considerando esse contexto, neste trabalho é apresentado um sistema web para a gestão de projetos de software implementando a metodologia Scrum. O sistema foi desenvolvido em linguagem Java e tecnologias associadas para a implementação de sistemas web. Como resultado, tem-se um sistema com funcionalidades simples, mas centrado na gestão das solicitações de uma equipe de projeto de software.

Palavras-chave: scrum; metodologias ágeis; gestão de projetos.

ABSTRACT

In the development of software projects, it is essential to meet the deadline (project schedule), defined cost (monetary budget and forecast of resources such as people, equipment, facilities and others) and the quality requirements established for the product that aim to meet the interests of the users of the software. The management of software projects has the objective that they are developed within time, with the established resources and that they meet the interests of the users in the best possible way. It is an established fact that the quality of the product, which is the software, is intrinsically related to the quality of the process used in its development. There are several models and quality standards defined for software aimed at the effectiveness of their projects. Some of them are quite expensive to implement in terms of financial resources and staff qualifications and, therefore, do not apply to projects or small software development companies. Agility in development with a minimum of organization of activities and control of their results is what companies and teams in general expect. Control mechanisms cannot be too costly and time-consuming because they discourage people and companies from following them, but some control must exist so that a plan can be followed, and results achieved. The proposal of agile methodologies such as Scrum is to provide an efficient management of the software process, without excess controls and routines. Considering this context, this work presents a web system for the management of software projects implementing the Scrum methodology. The system was developed with the Java language and associated technologies for the implementation of web systems. As a result, we have a simple system, but focused on managing the requests of a software project team.

Keywords: scrum; agile methodologies; project management.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGEM DE CÓDIGOS

LISTA DE SIGLAS

CSS	*Cascading Style Sheet*
HTML	*HyperText Markup Language*
IDE	*Integrated Devlopment Environment*
MVC	*Model View Controller*
PMBOK	*Project Management Body Of Knowledge*
PMI	*Project Management Institute*
RF	Requisitos Funcionais
RNF	Requisitos não funcionais
TI	Tecnologia da Informação
UML	*Unified Modling Language*
XP	*eXtreme Programming*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais que definem o contexto no qual se insere o sistema desenvolvido como resultado da realização deste trabalho. Neste capítulo também são apresentados os objetivos e a justificativa do trabalho. E por fim está a organização do texto por meio da apresentação dos capítulos subsequentes.

1.1 Considerações Iniciais

Projeto é conceituado pelo Guia *Project Management Body Of Knowledge* (PMBOK) como um esforço temporário empreendido para criar um produto, um serviço ou um resultado único (GUIA PMBOK PARTE 1, p. 3). O desenvolvimento de software é essencialmente realizado por meio de projetos que atendem o conceito proposto pelo Guia PMBOK porque eles são empregados para o desenvolvimento de um produto (software) único. Para que os requisitos do produto e a qualidade planejada sejam atendidos e que o projeto seja executado de acordo com o cronograma e o orçamento definidos, o desenvolvimento desses projetos precisa ser gerenciado tendo como base processos definidos e que forneça informações confiáveis para os diversos envolvidos (*stakeholders*).

O gerenciamento de projetos ocorre pela aplicação do conhecimento, das habilidades dos envolvidos na gestão das atividades realizadas, com o auxílio de ferramentas e técnicas visando atender aos seus requisitos, o cronograma e o orçamento. O gerenciamento permite que as organizações executem projetos de forma eficaz e eficiente (GUIA PMBOK PARTE 1, p. 10).

O gerenciamento eficaz e eficiente pode ser traduzido como o projeto sendo concluído com sucesso. Esse resultado é alcançado quando o projeto é finalizado dentro do tempo e do orçamento prévios atendendo aos requisitos estabelecidos na etapa de planejamento do projeto. Ou seja, quando concluído, o resultado deve ser satisfatório em relação ao prazo e aos custos estabelecidos (ARTIA, 2019), além de possuir as funcionalidades planejadas e de forma atender os interesses dos seus usuários. O gerenciamento de projetos auxilia a manter o controle dos processos e das atividades, a monitorar prazos e recursos, minimizar riscos, auxiliar na tomada de

decisão, definir as atribuições de cada envolvido em relação às tarefas que devem ser realizadas e a gestão dos resultados planejados para essas tarefas.

Existem diversas metodologias para gerenciar projetos em geral, de Tecnologia da Informação (TI) e mais especificamente de desenvolvimento de software. Quando voltadas para TI, no que se refere ao desenvolvimento de software, essas metodologias são também conhecidas como *frameworks* de gestão de projetos de TI. Há diversas ferramentas disponíveis que implementam estes conceitos. Como exemplo podemos citar o Scrum.

Scrum é um *framework* para colaboração entre os integrantes (equipe, time, *team*) de um projeto para o desenvolvimento de produtos (SCRUM ORG, 2019). Scrum Org (2019) conceitua Scrum como uma maneira ágil de gerenciar um projeto, geralmente de desenvolvimento de software. E define que desenvolvimento ágil de software com Scrum é frequentemente percebido como uma metodologia, mas Scrum é uma metodologia pensada como um *framework* para gerenciamento de processos.

Considerando a importância do gerenciamento de projetos e do uso de metodologias para o mesmo e tendo o Scrum como uma metodologia ágil para realizar essa gestão (SCRUM, 2019), por meio deste trabalho foi desenvolvido um sistema *web* que implementa essa metodologia.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um sistema *web* para a gestão de projetos de software que implemente a metodologia Scrum.

1.2.2 Objetivos Específicos

- Permitir o gerenciamento dos projetos pela atribuição de atividades (etapas de projeto) às pessoas vinculadas ao projeto por meio do cargo (papel) desempenhado.
- Auxiliar no acompanhamento das atividades que estão sendo realizadas em um projeto por meio da visualização do *status* de cada uma das etapas definidas para o projeto e do responsável pelas referidas etapas.

- Prover uma ferramenta para o gerenciamento das atividades relacionadas ao desenvolvimento de projetos de software, que possibilite o acompanhamento das etapas definidas para o projeto e dos responsáveis pela sua realização.

1.3 Justificativa

Com o amplo e crescente uso de software nas diversas áreas de atividade humana, seja profissional ou pessoal, houve a necessidade de metodologias de gestão dos projetos de desenvolvimento desses aplicativos e sistemas. Isso porque com demandas crescentes e diversificadas, os usuários passaram a ser mais exigentes, paralelamente à expansão de ofertas de tecnologias (ferramentas, linguagens e outros) para o desenvolvimento de software. A gestão dos projetos de software com o apoio de metodologias e ferramentas específicas passou a ser uma forma de auxiliar na obtenção de produtos que atendam às especificações dos usuários e sejam desenvolvidos com os recursos planejados (sejam pessoas, financeiros, equipamentos, tecnologias ou outros) e de acordo com o cronograma previsto.

O Scrum é uma metodologia bastante difundida, conhecida pelo conceito ágil e pode agregar um valor considerável à gestão de projetos dentro de uma empresa desenvolvedora de software.

Considerando esse contexto surgiu a ideia de implementar uma ferramenta baseada no Scrum para ser utilizada por equipes de desenvolvimento de software. Atualmente é comum que as empresas que praticam a metodologia Scrum o façam tendo quadros nos quais constam as demandas do projeto e cada integrante da equipe tem uma parte desse quadro no qual apresenta, por meio de adesivos, o que está fazendo. Esses quadros podem ser físicos ou implementados por ferramentas de software. Exemplos de ferramentas existentes que implementam o Scrum e são gratuitas ou que possuem versões gratuitas (SITEWARE, 2019): Trello, IceScrum, Scrumhalf, PangoScrum, Taiga, Mingle, MeisterTask, Asana, Wrike e YouTrack.

A ferramenta desenvolvida neste trabalho visa facilitar a gestão de projetos de software, fazendo com que seja possível colocar em prática conceitos do Scrum.

1.4 Estrutura do Trabalho

Este trabalho está organizado em capítulos. Este é o primeiro e apresenta as considerações iniciais com o contexto do sistema a ser desenvolvido, os seus objetivos e a justificativa. O Capítulo 2 apresenta o referencial teórico sobre métodos ágeis, enfatizando Scrum. No Capítulo 3 estão as ferramentas e as tecnologias utilizadas na modelagem do sistema e que serão utilizadas na sua implementação subsequente. No Capítulo 4 é apresentado o resultado da realização do trabalho que são: modelagem, apresentação e implementação do sistema. No Capítulo 5 é apresentado a conclusão deste trabalho. Por fim estão as referências utilizadas no texto.

2 MÉTODOS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE: SCRUM

Os métodos ágeis surgiram como uma alternativa para tornar os processos mais dinâmicos (mais adaptáveis durante a sua execução), redução de documentação e maior participação dos usuários (cliente), entre outros aspectos. O Scrum é um dos métodos caracterizados como ágeis.

2.1 Métodos Ágeis

O gerenciamento de rotinas de trabalho é necessário para alcançar os objetivos e atender prazos no ambiente corporativo, fazer com as equipes trabalhem com clareza e foco nos resultados esperados (CARNEIRO; SILVA; ALENCAR, 2018). Além disso, obter projetos alinhados com as necessidades dos clientes, dos usuários e com ciclo de vida menor têm sido exigências do mercado (GLAIEL; MOULTON; MADNICK, 2014).

Assim, o uso de ferramentas para o gerenciamento adequado das atividades da empresa é fundamental para o uso dos recursos disponíveis e o alcance dos objetivos setoriais e institucionais (PMI, 2017, OLIVEIRA; ALENCAR; COSTA, 2017). Nesse sentido, os desafios são reais para gerentes, independentemente da área de atuação, em uma correspondência clara entre o planejamento e o que é realizado durante o gerenciamento de projetos (CARNEIRO; SILVA; ALENCAR, 2018).

Considerando esse contexto, novas abordagens para gerenciamento de projetos chamadas de métodos ágeis foram criadas, objetivando, principalmente, ser flexível em decorrência da natureza dinâmica do ambiente, que é frequente em desenvolvimento de software (ROMANO; SILVA, 2015).

De acordo com o *Project Management Institute* (PMI) gerenciamento de projetos consiste na aplicação de conhecimento, habilidades e técnicas para que projetos sejam executados efetiva e eficientemente (PMI, 2017). Métodos de gerenciamento de projetos são geralmente divididos em tradicionais e ágeis (ŠPUNDAK, 2014). Métodos ágeis de gerenciamento de projetos, como o Scrum, o Lean Kanban e o *eXtreme Programming* (XP), são mais responsivos às mudanças e mais flexíveis. Scrum provê maior interação da equipe, tornando o rastreamento dos projetos mais dinâmico, fazendo, assim, que ele seja uma alternativa versátil para o

gerenciamento organizacional das atividades fora do contexto dos projetos (SCHWABER; SUTHERLAND, 2017). As metodologias ou métodos ágeis (tratados como sinônimos neste texto, embora conceitualmente não o sejam) surgiram em resposta à necessidade de projetos alinhados com interesses dos clientes e de maior rapidez para o seu desenvolvimento (CHING; MUTUC, 2018).

Métodos ágeis de gerenciamento de projeto, em contraste com abordagens tradicionais, enfatizam o planejamento contínuo e escopos flexíveis, fazendo com que mudanças ao longo do desenvolvimento do projeto sejam mais facilmente gerenciáveis, tornado, assim, os métodos ágeis mais atrativos em ambientes incertos e passíveis de mudanças, pela capacidade de resposta rápida. Agilidade é o ponto focal das iniciativas de mudança no desenvolvimento de software (CHING; MUTUC, 2018).

O objetivo principal dos métodos ágeis é desenvolver projetos melhores em período de tempo mais curto e isso pode ocorrer porque esses métodos adotam iteratividade junto com o processo incremental (HARB; NOTEBOOM; SARNIKAR, 2015). Já Darwish e Rizk (2015) destacam que a meta principal desses métodos é envolver mais os usuários com a equipe de desenvolvimento de software. Um usuário entende e analisa os requisitos de software e se o desenvolvedor enfrentar dificuldades em relação às funcionalidades, o usuário e o desenvolvedor podem discutir sobre visando encontrar a melhor solução (KHALIL; KOTAIAH**,** 2017). Assim, a proximidade do usuário com a equipe de desenvolvimento proverá alta comunicação e mínima documentação necessária.

No Scrum o desenvolvimento é segmentado em ciclos. E nos ciclo os projetos são segmentados em partes funcionais que são desenvolvidas iterativamente em *sprints* (JAAKKOLA; THALHEIM, 2014). Uma *sprint* é um ciclo iterativo de projeto. Os projetos são segmentados em *sprints*, com resultados bem definidos e incrementais em relação ao projeto. Assim, métodos ágeis permitem que as empresas de desenvolvimento de software lidem com as necessidades dinâmicas das suas demandas que são representadas por interesses de clientes e usuários de software (HIGHSMITH; COCKBURN, 2001).

2.2 Scrum

Scrum é caracterizado por ser um *framework* para gerenciamento de projetos (CARNEIRO; SILVA; ALENCAR, 2018). Os três pilares principais do Scrum são: transparência, inspeção e adaptação (SCHWABER; SUTHERLAND, 2017).

a) transparência – procura manter aspectos do processo visíveis para todos os que compartilham responsabilidades por resultados;

b) inspeção – deve ser realizada com frequência suficiente para detectar variações e não prejudicar a execução do trabalho sendo desenvolvido, inspecionar os artefatos e o progresso do trabalho tendo como parâmetro os objetivos da *sprint*.

c) Adaptação – ocorre quando inspecionados um ou mais aspectos da entrega em desenvolvimento, desvios são verificados em relação aos limites aceitáveis, podendo gerar um produto que poderá ser rejeitado. Os ciclos Scrum, definidos pelas suas *sprints*, são iterativos e focam na melhoria contínua (SCRUM STUDY, 2013).

Cervone (2011) destaca que Scrum é formado por três elementos principais: papéis, processos e artefatos. Entre os papeis estão: Scrum Master, a equipe de projeto e o Product Owner. Entre os processos estão: *kickoff*, reunião de planejamento da *sprint*, *spirnt*, *scrum* diária e encontro de revisão da *sprint*. E, ainda, a reunião de retrospectiva (SCHWABER; SUTHERLAND, 201). Os artefatos incluem: *backlog* do produto, *backlog* da *sprint* e *burndown charts* (gráficos).

O Scrum Master é o facilitador, o responsável por disseminar os valores e as práticas do Scrum e remover obstáculos durante a execução das atividades. A equipe de desenvolvimento de projeto é multifuncional, auto-organizada e, geralmente, trabalha no projeto de maneira holística (CARNEIRO; SILVA; ALENCAR, 2018). O *Producter Owner* orienta os requisitos do projeto e a sequência de entregas a serem feitas, fornecendo à equipe uma visão clara do que deve ser desenvolvido. Schwaber e Sutherlan (2017) ressaltam que não há relação de autoridade entre qualquer um dos três papéis.

Na reunião de *kickoff*, um *backlog* de produto geral (não detalhado) e os objetivos principais do projeto são definidos. Nas reuniões e planejamento das *sprints*, que incluem a equipe de projeto, o Scrum Master e o *Product Owner*, as seguintes atividades são realizadas (CERVONE, 2011): detalhamento do *product backlog* e definição do objetivo da *sprint*, que é o resultado a ser alcançado após realizada a

iteração; e a definição da *sprint backlog*, isto é, a lista de requisitos que serão desenvolvidos na *sprint* sendo planejada.

As *sprints* são limitadas à duração máxima de um mês (SCHWABER; SUTHERLAND, 2017). *Sprints* incluem reuniões de planejamento da *sprint*, reuniões diárias, realização do trabalho, reunião e revisão da *sprint* e encontro de retrospectiva. Reuniões de *check-in*, chamados de *Dailly* Scrum, ocorrem diariamente, durando não mais de 15 minutos e ocorrem entre o Scrum *Master* e a equipe Scrum. Durante a reunião, cada membro da equipe informa o que fez desde a última reunião, o que fará até a próxima reunião diária (*daily scrum*) e se há algo impedindo o avanço do trabalho de forma que o Scrum *Master* possa agir para auxiliar a resolver (KENETT, 2013).

A reunião de revisão da *sprint* é realizada ao final de cada *sprint*, sendo uma oportunidade programada de inspeção e adaptação do que está sendo desenvolvido e do trabalho em andamento. Durante a reunião, funcionalidades são discutidas entre a equipe, o Scrum *Master, o Product Owner*, com os *stakeholders*. Após isso, a reunião de retrospectiva é realizada e o Scrum Master, a equipe e o Product Owner discutem as principais ocorrências na *sprint* e quais ações serão realizadas para corrigir o que é necessário ajustar para a próxima *sprint* (KENETT, 2012 *apud* CARNEIRO; SILVA; ALENCAR, 2018).

Para os artefatos Scrum, o produto *backlog* constitui os requisitos dos projetos listados em ordem de prioridade. Essa lista é gerenciada e mantida pelo *Product Owner*. Durante a reunião de planejamento da *sprint*, a equipe estima o esforço requerido para completar cada item do *product backlog* e gera um *sprint backlog* (MAY; YORK; LANE, 2016).

Scrum enfatiza intencionalmente o trabalho realizado por meio do uso de gráficos bem como o monitoramento das *sprints* que é realizado por meio de *Scrumboard*, que é um *framework* de tarefas nas quais as estórias são formadas, escritas em papéis adesivos, afixadas em um quadro e o *status* (a ser feito (*to be done*), em progresso (*in progress*) ou finalizada (*completed*)) é atualizado nos encontros diários. Assim é possível monitorar o progresso do trabalho planejado (SCRUM STUDY, 2013).

3 MATERIAIS E MÉTODO

A seguir estão os materiais e o método utilizados para a modelagem e a implementação do sistema obtido resultado deste trabalho.

3.1 Materiais

O Quadro 1 apresenta a lista de ferramentas e tecnologias utilizadas para o desenvolvimento deste trabalho.

Quadro 1 – Lista de ferramentas e tecnologias

Ferramenta / Tecnologia	Versão	Finalidade
HyperText Markup Language (HTML)	5	Linguagem de marcação de hipertexto
Cascading Style Sheet (CSS)	3	Linguagem de folhas de estilo em cascata
Java	8	Linguagem de programação
Spring Boot: Spring Data JPA, Spring Web MVC, Spring Security	2.1.8	*Frameworks* para o lado servidor
PgAdmin	4	Gerenciador de banco de dados
PostgreSQL	9.6	Administrador de banco de dados
Lucidchart	4	Ferramenta de modelagem *Unified Modling Language* (UML)
Webstorm IDEA		*Integrated Devlopment Environment* (IDE) para implementação do código
IntelliJ IDEA	3.5	*Integrated Devlopment Environment* (IDE) para implementação do código
Apache Tomcat	8.0	Servidor da aplicação
Maven	3.6.1	Ferramenta de Automação *back-end*
PrimeNg	11.3	Biblioteca *front-end*
Bootstrap	5.0	Biblioteca *front-end*
Bulma	9.3	Biblioteca *front-end*
TypeScript	3.5.3	Linguagem de programação
Angular	8.3	*Framework front-end*
Angular CLI	8.3	*Framework front-end*
Chart.js	2.9.4	Biblioteca para gráficos
Html2pdf	10	Biblioteca para geração de PDF

Fonte: autoria própria(2022).

3.2 Método

O método consiste nas atividades de levantamento de requisitos, modelagem desses requisitos e na implementação. A seguir descrita sucintamente a realização dessas etapas.

Levantamento de requisitos

O levantamento dos requisitos foi realizado a partir da observação da realização de atividades e dos procedimentos e de ferramentas de gestão utilizados em uma empresa de desenvolvimento de software.

Análise e projeto

A análise foi baseada no levantamento dos requisitos organizados em funcionais e não funcionais, na definição dos casos de uso e nos dados a serem persistidos. Esses foram organizados em tabelas relacionadas entre si e com a definição dos dados a serem armazenados.

Implementação

A implementação teve como base o padrão *Model View Controller* (MVC) dividida em:

1) Implementação do *model*

A etapa inicial de implementação teve como foco implementar a parte do *model* do projeto, implementando todas as classes necessárias para o acesso ao banco de dados.

2) Implementação do *controller*

Na segunda etapa, o foco foi desenvolver as classes de *service*, *repository* e *controller*, que tem como objetivo realizar a comunicação da parte de acesso aos dados à parte visual. Interpretando as requisições trazidas pela parte visual para realizar as ações mapeadas para essas requisições.

3) Implementação da parte visual dos cadastros

Implementado a parte visual de todos os cadastros, fazendo com que sejam geradas requisições para o servidor para incluir, editar ou excluir dados dos cadastros.

4) Implementação da tela Scrum

Nessa etapa foi feita a implementação da tela principal do sistema, que é a tela de Scrum. Foi desenvolvida uma tela com várias funcionalidades visuais para que todo

o andamento de um projeto seja facilmente visível e identificável aos usuários. Tendo como foco apresentar o andamento de cada solicitação do projeto que estão sendo trabalhadas no momento.

5) Desenvolvimento de consultas

As consultas trazem informações sobre os projetos, solicitações, usuários, cargos e outros. Todas as consultas poderão ser filtradas.

6) Desenvolvimento de relatório

Implementado a possibilidade de gerar um relatório que pode ser impresso ou gerado em arquivo. O relatório traz as informações contidas no quadro scrum no momento da impressão.

4 RESULTADOS

Este capítulo apresenta o resultado obtido da realização deste trabalho.

4.1 Escopo do Sistema

Ferramenta *web* de gestão de projetos na qual cada usuário tem acesso às solicitações que fazem parte de projetos que são realizados por um ou mais usuários. A ferramenta tem como base o Scrum e implementa o uso dos conceitos dessa metodologia de maneira visual, tendo como requisitos principais:

a) Cadastro de cargos - papel ou perfil dos usuários (equipe do projeto), com atribuição de permissão de acessos às funcionalidades do sistema. Os usuários cadastrados no sistema devem, obrigatoriamente, estar vinculados a um cargo que define as funções da pessoa no projeto. Há uma tela para consultar todos os usuários do sistema, tendo suas informações, seu cargo e os projetos e solicitações nas quais ele está trabalhando.

b) Cadastro de solicitações - solicitações são atividades ou tarefas que podem ou não fazer parte de um projeto. Essas solicitações serão divididas em etapas e cada etapa deve ter um responsável, que trabalhará nessa solicitação e a concluirá ou passará para um próximo status. Cada solicitação terá uma descrição geral do que será feito, nome e data máxima de entrega. As solicitações devem, obrigatoriamente, ser divididas em *status*. Cada solicitação deve, obrigatoriamente, possuir um responsável associado.

c) Cadastro de etapas - são as divisões das solicitações, são etapas essenciais para que uma solicitação seja concluída. Cada etapa tem um responsável, a descrição específica do que deve ser feito nessa etapa e um *status*. As opções de *status* serão: “Backlog”, "*To do*", "*Doing*", "*Done*", "*Test*", "*Done*".

d) Cadastro de projetos - contendo uma descrição básica, nome e data de entrega. Um projeto novo somente pode ser inserido por um usuário com um cargo que tenha permissão para fazê-lo.

e) Cadastro de *Sprints* - contendo um nome, o projeto vinculado e as datas limites. A *sprint* definirá as solicitações que estão sendo feitas naquele momento pela equipe.

e) Tela Scrum - tela principal do sistema. Essa tela é o centro de informações para a gestão dos projetos. Tem um visual semelhante com um quadro Scrum ficando bem visível o que está sendo feito por cada um dos usuários vinculados ao referido projeto, o *status* de cada solicitação e etapa. O usuário autenticado no sistema ao abrir essa tela pode escolher um dos projetos, *sprint* ou o responsável para filtrar as solicitações mostradas na tela. Todos os cadastros pertencentes às solicitações filtradas são apresentados nesta tela. Nessa tela os usuários podem alterar o *status* das suas solicitações

4.2 Modelagem do Sistema

O Quadro 2 apresenta os Requisitos Funcionais (RF) definidos para o sistema desenvolvido como resultado da realização deste trabalho.

Quadro 2 – Requisitos funcionais

Identificação	Nome	Descrição
RF01	Cadastrar usuários	Usuários são todos aqueles que terão acesso ao sistema e participação em alguma solicitação ou projeto. Existem usuários membros da equipe, que utilizaram as funcionalidades normais do sistema e o usuário admin, que pode definir o cargo dos demais usuários.
RF02	Cadastrar cargos	Para uma melhor divisão das atividades dentro de um projeto, o cargo define qual função o usuário está desempenhando dentro da equipe.
RF03	Cadastrar solicitações	Para ser concluída, uma solicitação deve obrigatoriamente passar por status. As solicitações são o que formam o projeto e estão vinculadas, obrigatoriamente, a um projeto, são todas as atividades que serão realizadas pelo usuário ou por uma equipe.
RF04	Cadastrar etapas	Etapas têm como objetivo segmentar solicitações, para que a equipe responsável pelo projeto divida o que será feito entre os membros da equipe.
RF05	Cadastrar projetos	Todo projeto tem um responsável e os seus membros. É formado por solicitações, que definem o que será feito no projeto. O projeto como um todo é o que será disposto no quadro Scrum, para que o usuário consiga ter uma visão geral do andamento do projeto.
RF06	Cadastrar Sprints	Sprints tem como objetivo fazer a separação das solicitações que devem ser trabalhadas para aquele momento específico do projeto.
RF06	Quadro Scrum	É a tela para acompanhamento do desenvolvimento do projeto conforme descrito no

		escopo do sistema, visando atender os pressupostos de gestão de projetos definidos pelo Scrum.
RF07	Emitir consultas e relatório	Consultas de todos os cadastros possíveis no sistema e um relatório do quadro Scrum.
RF08	Alterar status das solicitações	Como todas as etapas possuem *status*, é possível alterar o *status* delas entre as opções fixas: “Backlog”, “To Do”, “Doing”, “Test”, “Done”.
RF09	Vincular usuários a projetos	O usuário criador de um projeto pode inserir outros usuários no projeto, sendo assim, dando a eles permissão de criar solicitações para o referido projeto.

Fonte: autoria própria(2022).

A listagem do Quadro 3 apresenta os Requisitos não Funcionais (RNF) definidos para o sistema.

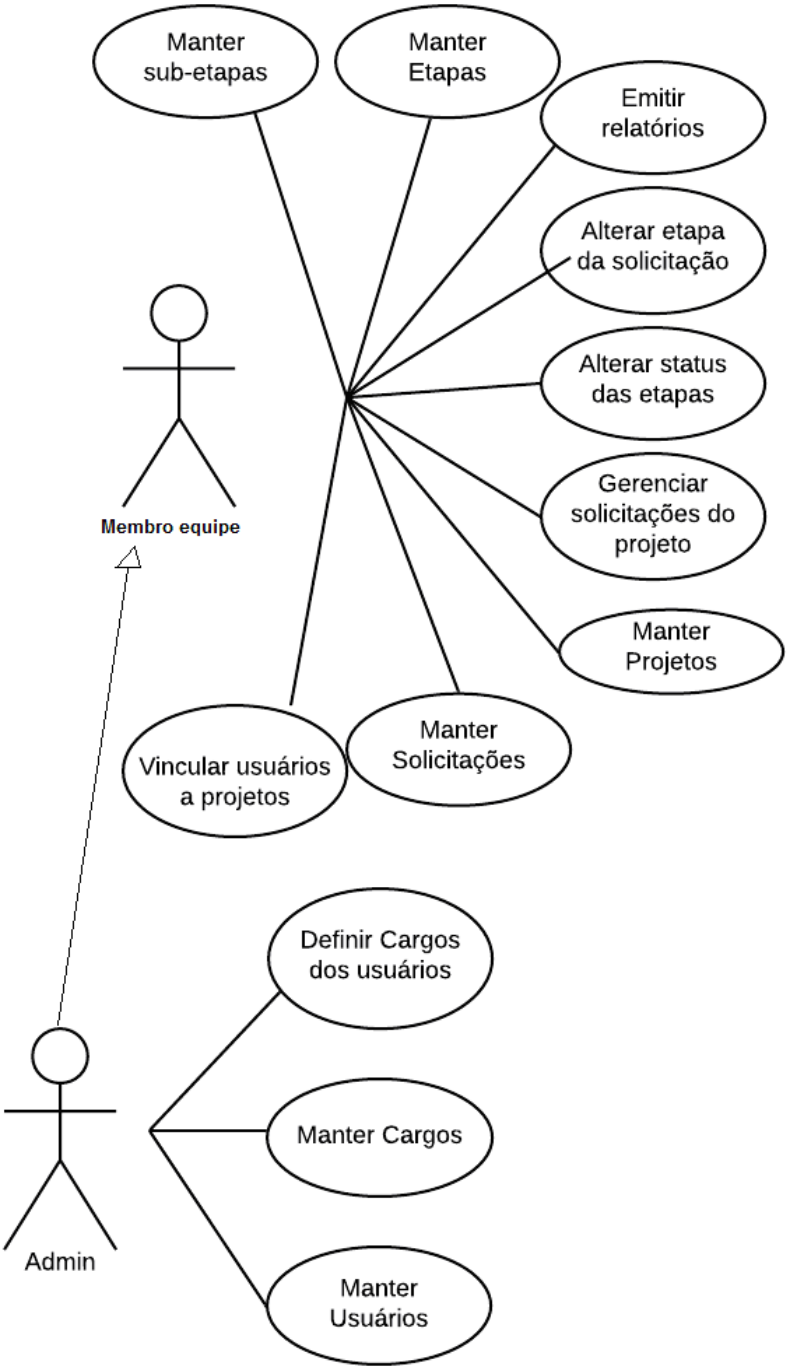
Quadro 3 – Requisitos não funcionais

Identificação	Nome	Descrição
RNF01	Acesso ao sistema	O acesso ao sistema é realizado por meio de *login* e senha.
RNF02	Acesso aos projetos	O usuário possui acesso somente aos projetos aos quais ele está vinculado.
RNF02	Compatibilidade com navegadores web	O sistema funciona nos principais navegadores como, por exemplo: Chrome, Firefox e Microsoft Edge.
RNF03	Validar dados	Campos de preenchimento obrigatório e que devem atender regras como tamanho mínimo, existência ou não de determinados caracteres são adequadamente validados.

Fonte: autoria própria(2022).

Os requisitos foram organizados em casos de uso, apresentados na Figura 1. Foram definidos dois atores. O administrador (admin) com as permissões relacionadas à manutenção de cargos e usuários do sistema e ele também herda todas as permissões do usuário membro equipe. É o usuário membro da equipe que desempenha papeis como analista, programador e testador. As permissões do usuário membro da equipe são definidas pelo cargo que ele desempenha. Os membros da equipe estarão vinculados a projetos e realizarão atividades dos projetos.

Figura 1 - Diagrama de Caso de Uso



Fonte: autoria própria(2022).

A seguir é apresentada a expansão dos casos de uso da Figura 1. O Quadro 4 apresenta a expansão da operação de inclusão dos casos manter.

Quadro 4 – Operação de inclusão de dados no sistema

Casos de uso: Manter usuários, cargos, etapas, *sprints*, projetos e solicitações: operação de inclusão de cadastros do sistema. **Descrição:** Inclusão dos dados cadastrais no sistema. **Evento Iniciador:** Administrador necessita inserir um novo usuário no sistema ou incluir um novo cargo. Membro da equipe necessita inserir nova etapa, sprint, projeto ou solicitação. **Atores:** Administrador que está incluindo um usuário ou cargo. Membro da equipe que está incluindo uma nova etapa, subetapa, projeto ou solicitação. **Pré-condição:** Não há. **Sequência de Eventos:** **1.** Ator acessa a tela para realizar o cadastro inserindo as informações necessárias. **2.** Ator seleciona o tipo de usuário. **3.** O sistema insere as informações no banco de dados e informa o *status* do procedimento. **Pós-Condição:** Usuário inserido no banco de dados.	
Nome do fluxo alternativo (extensão)	Descrição
1.Campos obrigatórios não informados	1.1. O ator deixa de informar dados obrigatórios e clica em salvar. 1.2. O sistema valida que não foram informados todos os campos obrigatórios e exibe mensagem ao usuário sem salvar o registro. 1.3. O sistema permanece na tela de inclusão mantendo os dados informados anteriormente.
2.Campos informados em formato incorreto	2.1. O ator informa dados em formato incorreto e clica em salvar. 2.2. O sistema valida que os dados não estão no formato esperado e exibe mensagem ao usuário sem salvar o registro. 2.3. O sistema permanece na tela de inclusão mantendo os dados informados anteriormente.

Fonte: autoria própria(2022).

A operação de alteração de dados cadastrais de usuário e tipo de usuário é apresentada no Quadro 5.

Quadro 5 – Operação de alteração de dados no sistema

Caso de uso:

Manter usuários, cargos, etapas, sprints, projetos e solicitações: operação de alteração de dados cadastrais do sistema. **Descrição:** Alteração dos dados cadastrais no sistema. **Evento Iniciador:** Usuário administrador ou membro da equipe acessa o cadastro desejado para alterar dados de cadastro no sistema. **Atores:** Administrador para alterar dados cadastrais de usuário ou e cargos. Membro da equipe para alterar dados cadastrais de etapa, sprint, projeto ou solicitação. **Pré-condição:** Cadastro do usuário e de tipo de usuário estar no sistema e usuário com *login* ativo. **Sequência de Eventos:** **1.** Ator acessa a tela para visualização dos dados do registro e seleciona o registro que deseja alterar. **2.** O sistema apresenta o registro selecionado para alteração. **3.** Ator altera os dados do registro. **4.** O sistema altera as informações no banco de dados e informa ao usuário o *status* do procedimento. **Pós-Condição:** Dados de cadastro alterados no banco de dados.

Nome do fluxo alternativo (extensão)	Descrição
1.Campos obrigatórios não informados	1.1. O ator deixa de informar dados obrigatórios e clica em salvar. 1.2. O sistema valida que não foram informados todos os campos obrigatórios e exibe mensagem ao usuário sem salvar o registro. 1.3. O sistema permanece na tela de inclusão mantendo os dados informados anteriormente.
2.Campos informados em formato incorreto	2.1. O ator informa dados em formato incorreto e clica em salvar. 2.2. O sistema valida que os dados não estão no formato esperado e exibe mensagem ao usuário sem salvar o registro. 2.3. O sistema permanece na tela de inclusão mantendo os dados informados anteriormente.

Fonte: autoria própria(2022).

A operação de exclusão de cadastros de usuários no sistema é descrita no Quadro 6.

Quadro 6 – Operação de exclusão de cadastros no sistema

Caso de uso: Manter usuários, cargos, etapas, sprints, projetos e solicitações: operação de exclusão de dados cadastrais do sistema. **Descrição:** Exclusão dos dados cadastrais no sistema. A operação de exclusão excluirá o registro do banco de dados. **Evento Iniciador:** Usuário administrador ou membro da equipe para excluir cadastros no sistema. **Atores:** Administrador para excluir usuários ou tipos de usuários. Membro da equipe para excluir etapa, subetapa, projeto ou solicitação. **Pré-condição:** Cadastro estar incluso no sistema.

Sequência de Eventos: **1.** Ator acessa a tela para exclusão do registro. **2.** O sistema exclui as informações no banco de dados e informa ao usuário o *status* do procedimento. **Pós-Condição:** Cadastro de usuário excluído no banco de dados.	
Nome do fluxo alternativo (extensão)	Descrição
1. Cadastro vinculado	1.1. O sistema verifica que o cadastro possui vínculos com outras tabelas do sistema. 1.2. O sistema informa que não é possível realizar a exclusão pela existência de vínculos ativos. 1.3 O sistema retorna para o formulário.

Fonte: autoria própria(2022).

No Quadro 7 está a descrição da operação consultar referente aos casos de cadastro.

Quadro 7 – Operação de consultar dados de cadastros no sistema

Caso de uso: Manter usuários, cargos, etapas, sprints, projetos e solicitações: operação consultar dados cadastrais do sistema. **Descrição:** Consulta dos dados de usuário do sistema. **Evento Iniciador:** Ator solicita consulta de um registro no sistema. **Atores:** Administrador para cadastros de usuários e cargos. Membro da equipe para cadastros de etapas, subetapas, projetos e solicitações. **Pré-condição:** Registro estar incluso no sistema. **Sequência de Eventos:** **1.** O administrador acessa a tela para visualização dos dados do registro. **2.** O administrador indica os filtros desejados para consulta. **3.** O sistema apresenta os dados da consulta ao usuário. **Pós-Condição:** Dados da consulta apresentados para o usuário.

Fonte: autoria própria(2022).

A descrição do caso de uso vincular usuário a projeto é apresentada no Quadro 8.

Quadro 8 – Vincular usuário a projeto

Caso de uso: Vincular usuário a projeto. **Descrição:** Vincular um usuário a um projeto, para que ele tenha acesso ao projeto. **Evento Iniciador:** Ator solicita a inserção de um usuário ao projeto.

Atores: Usuário membro da equipe que é dono do projeto. **Pré-condição:** Ator ser o dono do (responsável pelo) projeto em questão. **Sequência de Eventos:** **1.** Ator acessa a tela do projeto. **2.** Ator solicita inserção de um usuário ao projeto. **3.** O sistema insere o usuário no projeto. **Pós-Condição:** O usuário adicionado tem acesso ao projeto.

Fonte: autoria própria(2022).

No Quadro 9 está a expansão do caso de uso alterar etapa da solicitação.

Quadro 9 – Alterar status da solicitação

Caso de uso: Alterar status da solicitação. **Descrição:** Alterar o status de uma solicitação, para que fique claro para os demais participantes do projeto qual é a situação dessa solicitação no projeto. **Evento Iniciador:** Ator solicita a mudança de status indicando qual etapa ele deseja mudar. **Atores:** Membro da equipe que é o responsável pela solicitação. Membro da equipe que é o dono do projeto. **Pré-condição:** Solicitação estar presente em um projeto. **Sequência de Eventos:** **1.** O ator acessa a tela da solicitação. **2.** O ator indica qual status deseja alterar. **3.** O sistema determina o status escolhido como a etapa atual da solicitação. **Pós-Condição:** O status atual da solicitação é a estabelecida. **Fluxo Alternativo:**	
Nome do fluxo alternativo (extensão)	Descrição
Solicitação com apenas uma etapa	1.1. O ator solicita a mudança de status. 1.2. O sistema retorna que o status atual é a única cadastrada para essa solicitação.

Fonte: autoria própria(2022).

A expansão do caso de uso gerenciar solicitações do projeto é apresentada no Quadro 10.

Quadro 10 – Gerenciar solicitações do projeto

Caso de uso: Gerenciar solicitações do projeto. **Descrição:** Iniciar ou finalizar uma solicitação de um projeto, se ela já foi iniciada, irá finalizar.

Evento Iniciador: Ator solicita o início/fim da solicitação. **Atores:** Membro da equipe que é o responsável pelo projeto. **Pré-condição:** Solicitação estar vinculada a um projeto. **Sequência de Eventos:** **1.** O ator acessa a tela da solicitação. **2.** O ator clica no botão de início/fim da solicitação. **3.** O sistema insere a data de início/fim da solicitação com a data atual. **Pós-Condição:** A solicitação fica como iniciada/terminada.

Fonte: autoria própria(2022).

A expansão do caso de uso emitir relatório está no Quadro 11.

Quadro 11 – Emitir relatórios

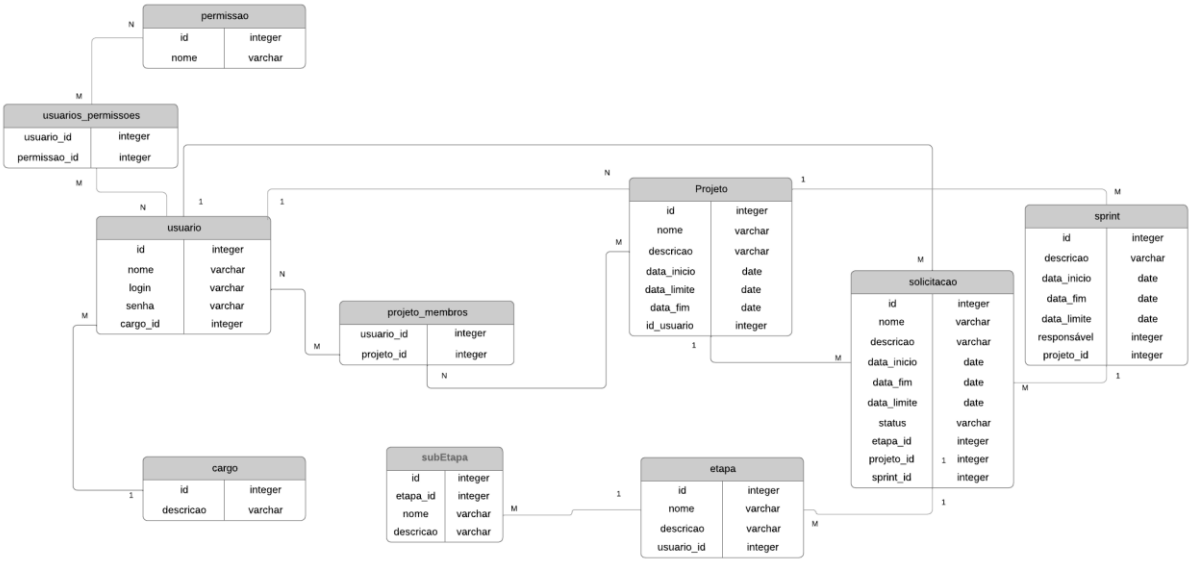
Caso de uso: Emitir relatório. **Descrição:** Emitir relatório com informações baseadas nos dados do projeto. **Evento Iniciador:** Informar os filtros desejados e gerar o relatório. **Atores:** Responsável ou membro do projeto. **Pré-condição:** Ter dados conforme o filtro informado para que o relatório seja gerado. **Sequência de Eventos:** **1.** Abrir o quadro scrum. **2.** Clicar na opção de relatórios. **3.** Informar os filtros desejados. **4.** O relatório é mostrado. **Pós-Condição:** Relatório gerado conforme filtros selecionados. **Fluxo Alternativo:**

Nome do fluxo alternativo (extensão)	Descrição
Nenhum dado encontrado com os filtros escolhidos	**1.1** Informar os filtros desejados **1.2** O sistema retorna que não há dados para exibir com esses filtros. **1.3** O sistema volta para a tela de filtros para que outros filtros sejam inseridos.

Fonte: autoria própria(2022).

A Figura 2 apresenta a modelagem das entidades do banco de dados e os seus relacionamentos.

Figura 2 - Diagrama do banco de dados



Fonte: autoria própria(2022).

O Quadro 12 apresenta os campos da tabela usuário. O usuário cadastrado no sistema estará vinculado a um tipo (administrador ou membro da equipe). As permissões de acesso do usuário ao sistema serão definidas por meio do cargo do usuário.

Quadro 12 – Campos da tabela usuário

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
id	Numérico	Não	Sim	Não
nome	Texto	Não	Não	Não
login	Texto	Não	Não	Não
senha	texto	Não	Não	Não
cargo_id	Numérico	Não	Não	Da tabela Cargo

Fonte: autoria própria(2022).

As permissões de acesso dos usuários para o sistema são definidas na tabela permissão que pode ser vista no Quadro 13.

Quadro 13 - Campos da tabela permissão

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
id	Numérico	Não	Sim	Não
nome	Texto	Não	nao	Não

Fonte: autoria própria.

Um usuário pode ter várias permissões a tabela que faz esse relacionamento pode ser vista no Quadro 14.

Quadro 14 - Campos da tabela usuario_permissoes

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
usuario_id	Numérico	Não	Sim	Da tabela usuario
permissao_id	Numérico	Não	Sim	Da tabela permissao

Fonte: autoria própria(2022).

A tabela de cargos é utilizada pela definir as permissões de acesso do usuário ao sistema. Os campos da tabela de cargo, como apresentado no Quadro 15, são apenas um identificador (chave primária) e a descrição.

Quadro 15 – Campos da tabela cargo

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
id	Numérico	Não	Sim	Não
descricao	Texto	Não	nao	Não

Fonte: autoria própria(2022).

A tabela de projeto contém os dados dos projetos realizados, em execução e de execução futura. Os campos dessa tabela são apresentados no Quadro 16. Essa tabela possui um campo que armazena a data limite de finalização do projeto definida no planejamento. Por meio dessa data é possível verificar se o projeto está atrasado em relação ao cronograma planejado ou se foi finalizado antes do previsto ou com atraso. O campo “usuario_id” define o responsável pelo projeto.

Quadro 16 – Campos da tabela projeto

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
id	Numérico	Não	Sim	Não
nome	Texto	Não	Não	Não

descricao	Texto	Sim	Não	Não
data_inicio	Data	Sim	Não	Não
data_limite	Data	Sim	Não	Não
data_fim	Data	Sim	Não	Não
usuario_id	Numérico	Não	Não	Da tabela usuario

Fonte: autoria própria(2022).

Um usuário pode fazer parte da equipe de mais de um projeto simultaneamente e um projeto pode ter uma equipe composta por diversas pessoas (usuários do sistema). Assim, é necessário haver uma tabela de relacionamento entre as tabelas de usuários e de projeto, os campos dessa tabela são apresentados no Quadro 17.

Quadro 17 – Campos da tabela projeto_membros

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
usuário_id	Numérico	Não	Não	Da tabela Usuario
projeto_id	Numérico	Não	Não	Da tabela Projeto

Fonte: autoria própria(2022).

Solicitações definem as tarefas a serem realizadas em um projeto e elas estão vinculadas às etapas do ciclo de vida do projeto. O Quadro 18 apresenta os campos da tabela de solicitações.

Quadro 18 – Campos da tabela solicitacao

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
id	Numérico	Não	Sim	Não
nome	Texto	Não	Não	Não
descricao	Texto	Sim	Não	Não
data_inicio	data	Sim	Não	Não
data_fim	Data	Sim	Não	Não
data_limite	Data	Sim	Não	Não
status	Texto	Sim	Não	Não
etapa_id	Numérico	Não	Não	Da tabela Etapa
projeto_id	Numérico	Não	Não	Da tabela Projeto
sprint_id	Numérico	Não	Não	Da tabela Sprint

Fonte: autoria própria(2022).

As etapas das solicitações são cadastradas por meio dos campos apresentados no Quadro 19.

Quadro 19 – Campos da tabela etapa

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
id	Numérico	Não	Sim	Não
Nome	Texto	Não	Não	Não
Descricao	Texto	Sim	Não	Não
usuario_id	Numérico	Não	Não	Da tabela usuario

Fonte: autoria própria(2022).

As etapas podem ser decompostas em subetapas. Os campos da tabela de subetapas são apresentados no Quadro 20.

Quadro 20 – Campos da tabela subetapa

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
id	Numérico	Não	Sim	Não
etapa_id	Numérico	Não	Não	Da tabela Etapa
nome	Texto	Não	Não	Não
descricao	Texto	Sim	Não	Não

Fonte: autoria própria(2022).

Sprints podem ser geradas para a divisão dos projetos. Os campos da tabela sprint são apresentados no Quadro 21.

Quadro 21 - Campos da tabela Sprint

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
id	Numérico	Não	Sim	Não
descricao	Numérico	Não	Não	Da tabela Etapa
data_inicio	Data	Sim	Não	Não
data_fim	Data	Sim	Não	Não
data_limite	Data	Sim	Não	Não
responsavel	Numérico	Não	Não	Da tabela Usuario
projeto_id	Numérico	Não	Não	Da tabela Projeto

Fonte: autoria própria(2022).

4.3 Apresentação do Sistema

Nesta seção serão apresentadas as principais funcionalidades desenvolvidas no sistema por meio de telas e descrições de suas funções.

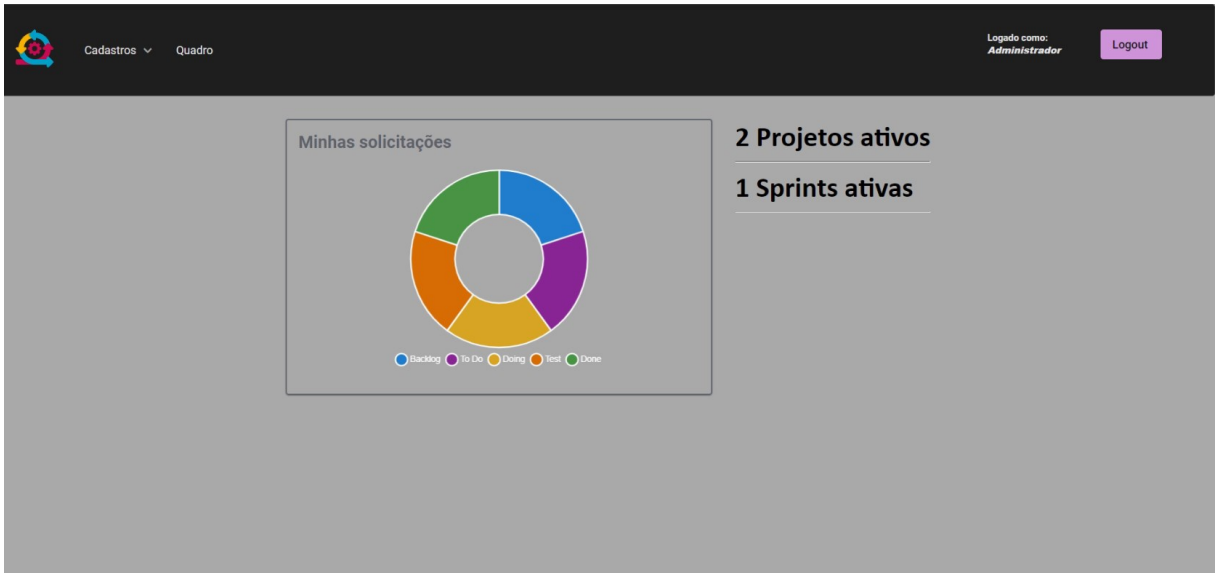
Na Figura 3 é apresentada a página de autenticação do sistema. Para autenticar-se é necessário informar o usuário e senha cadastrados. Não é possível realizar o seu próprio cadastro, apenas usuários com permissão de administrador podem cadastrar novos integrantes.

Figura 3 - Tela de Autenticação

Fonte: autoria Própria (2022).

Ao realizar a autenticação, o usuário é direcionado para a página inicial do sistema, na qual é apresentado uma tela de resumo (*dashboard*), que pode ser observado na Figura 4.

Figura 4 - Dashboard do Usuário



Fonte: Autoria Própria (2022).

O menu com acesso às rotinas do sistema fica localizado no lado esquerdo da página, nele estão os atalhos para todas as funcionalidades do sistema. Os usuários que possuírem permissão de administrador, terão acesso a todas as opções do menu. Já os com permissão de usuários comuns, não poderão acessar o cadastro de usuários, o cadastro de cargos e cadastrar novos projetos.

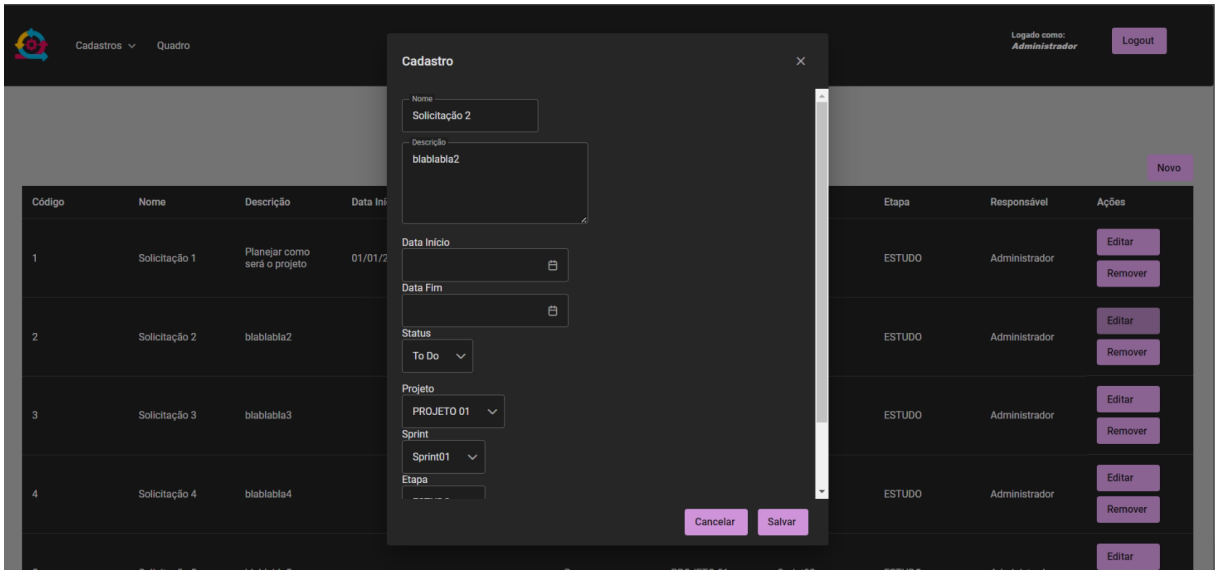
A Figura 5 apresenta a lista de solicitações cadastrados dentro do sistema. Nela são apresentadas todas as informações sobre a solicitação. Uma solicitação representa uma atividade, trabalho ou exercício a ser realizada por um membro da equipe, as solicitações são bem genéricas para que possam se adaptar para qualquer tipo de equipe possa utilizá-la. Para editar ou excluir algum item, basta clicar nos botões localizados no final de cada linha da tabela. Caso seja necessário criar um item, basta clicar no botão “NOVO”. Essa estrutura é apresentada em quase todas as tabelas do sistema, com exceção apenas do quadro Scrum.

Figura 5 - Lista de solicitações cadastradas

Fonte: Autoria Própria (2022).

Ao clicar para editar um registro, o cadastro com todas as suas respectivas informações é mostrado conforme a Figura 6. É aberto um componente do tipo modal, no caso da Figura 6 são apresentados os componentes que representam os dados do registro, sendo eles do tipo texto, data e caixa de seleção, sendo o último geralmente utilizado para carregar dados referente a outras tabelas de dados existentes no sistema.

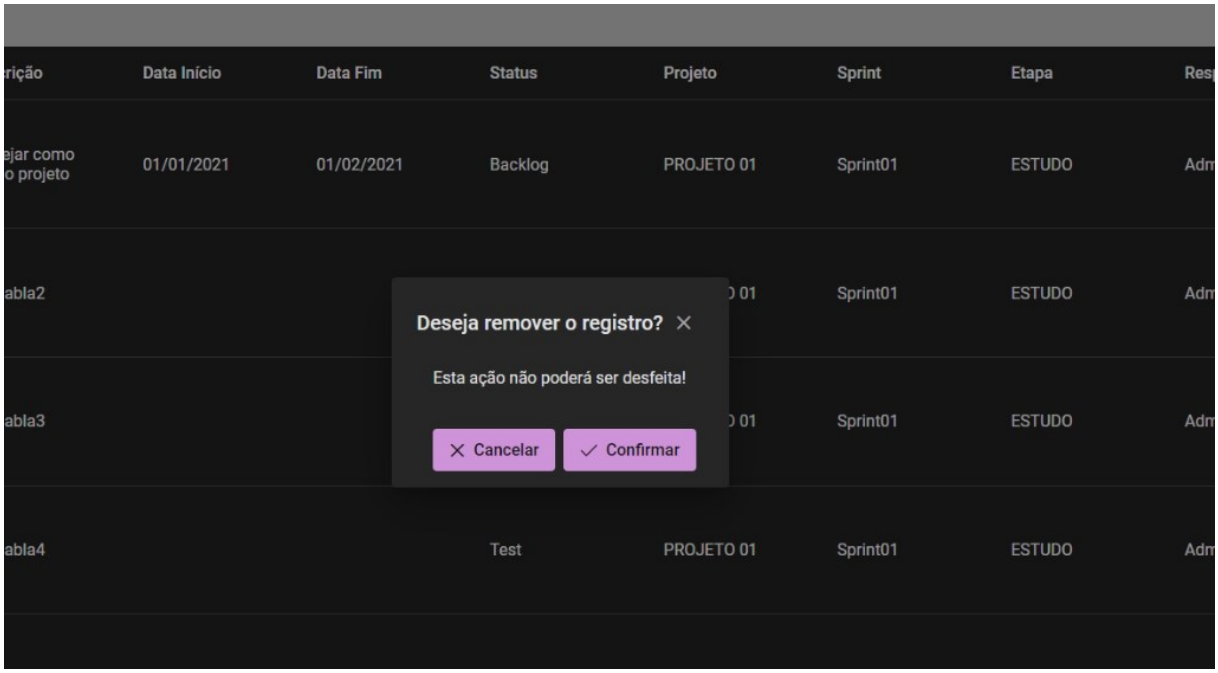
Figura 6 - Modal de detalhes de uma solicitação



Fonte: Autoria Própria (2022).

A Figura 7 demonstra a exclusão de um registro. Ao clicar para excluir um registro uma mensagem solicitando a confirmação do usuário é mostrada. Quando a tentativa de exclusão do registro for concluída com sucesso uma mensagem de aviso é mostrada. Caso ocorra algum problema com a exclusão, geralmente pelo registro ter algum vínculo com outro do sistema, uma mensagem dizendo que a exclusão não pode ser concluída é apresentada para o usuário.

Figura 7 - Aviso de confirmação de exclusão



Fonte: Autoria Própria (2022).

Na Figura 8 pode-se visualizar a principal tela do sistema, o quadro Scrum. Nesse quadro é realizada toda a movimentação do *status* das solicitações cadastradas, de acordo com a necessidade do usuário. O quadro pode ser filtrado por projeto, *sprint* e responsável, esses filtros proporcionam facilidade para que os usuários vejam os dados específicos para algum caso que estão trabalhando naquele momento, como por exemplo se um usuário é responsável por uma *sprint* ele pode filtrá-la para saber exatamente o status das solicitações que fazem parte dessa *sprint*. Cada *card* do quadro corresponde à uma solicitação e apresenta as principais informações sobre a mesma que são: nome, descrição, etapa, responsável, *sprint* e projeto. Também é possível visualizar todos os dados da solicitação ao clicar no botão com o ícone de um olho que foi destacado na Figura 8, na Figura 9 pode ser vista a tela modal com os dados da solicitação. Conforme descrito na seção 4.1 o quadro

Scrum possui as opções de *status*, quando movemos o card para outra coluna seu *status* é automaticamente alterado para *o* representado pela respectiva coluna. Essa funcionalidade proporciona agilidade e facilidade para que os usuários do projeto saibam como estão os andamentos das atividades da equipe assim que um usuário altera o status de uma solicitação.

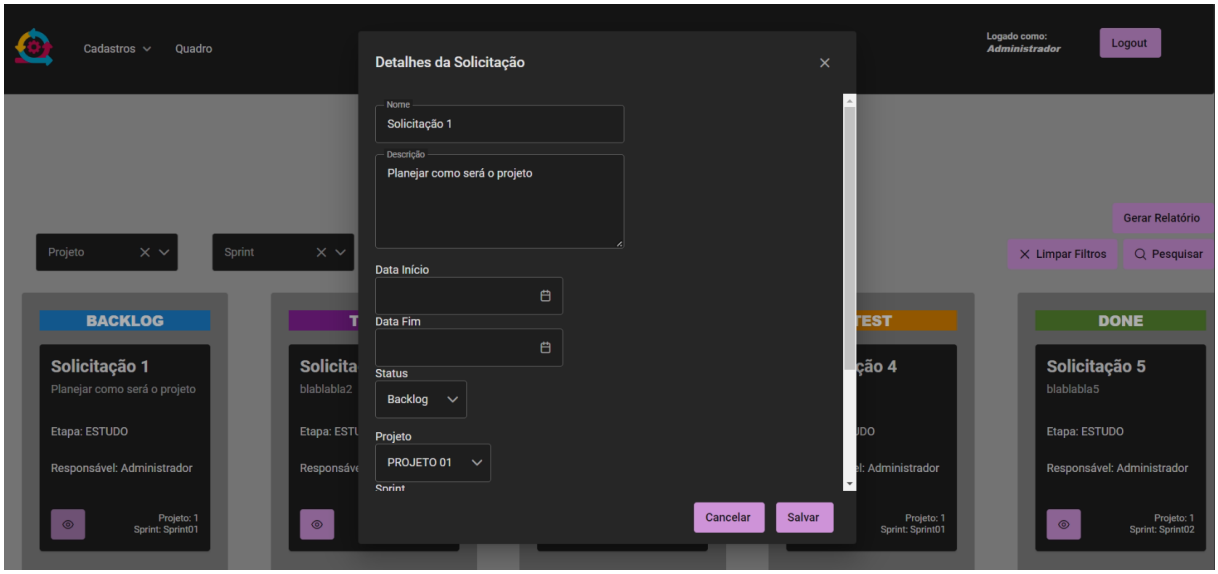
Ter essas informações sobre as solicitações acessíveis nesse formato visual proporcionam uma visão geral ao responsável pelas atividades ou a equipe inteira.

Figura 8 - Quadro Scrum

Fonte: Autoria Própria (2022).

Como pode ser observado na Figura 9, além da visualização dos ados o usuário também pode editar os valores, é normal que no meio da execução de uma atividade o usuário precise trocar a etapa ou descrever alguma informação que julga importante que a equipe saiba no card, sendo assim ao abrir a visualização do card ele poderá fazer essas modificações.

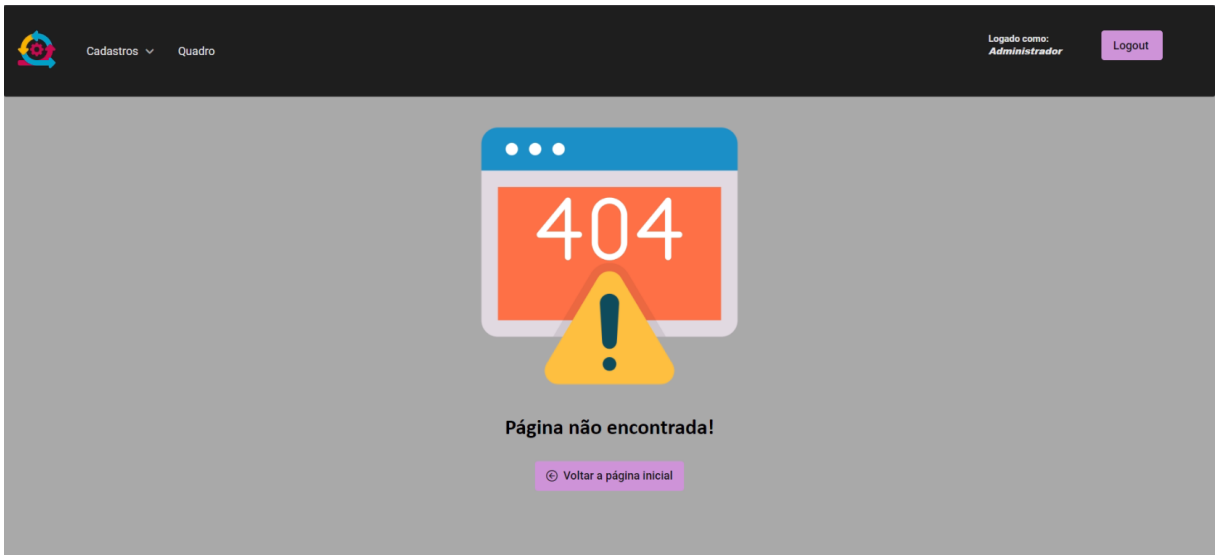
Figura 9 - Modal com os detalhes de um card



Fonte: Autoria Própria (2022).

Além das funcionalidades básicas do sistema, também foram implementadas soluções para auxiliar os usuários durante a navegação entre as páginas do sistema. Quando um usuário tentar acessar uma página não existente, ele será redirecionado a uma página específica, com intuito de que não ficar perdido na utilização do sistema, sendo representada pela Figura 10.

Figura 10 - Tela de página não encontrada



Fonte: Autoria Própria (2022).

Quando um usuário tentar acessar uma tela em que o mesmo não tenha permissão, ou seja, o cargo configurado para esse usuário não permite o acesso, ele

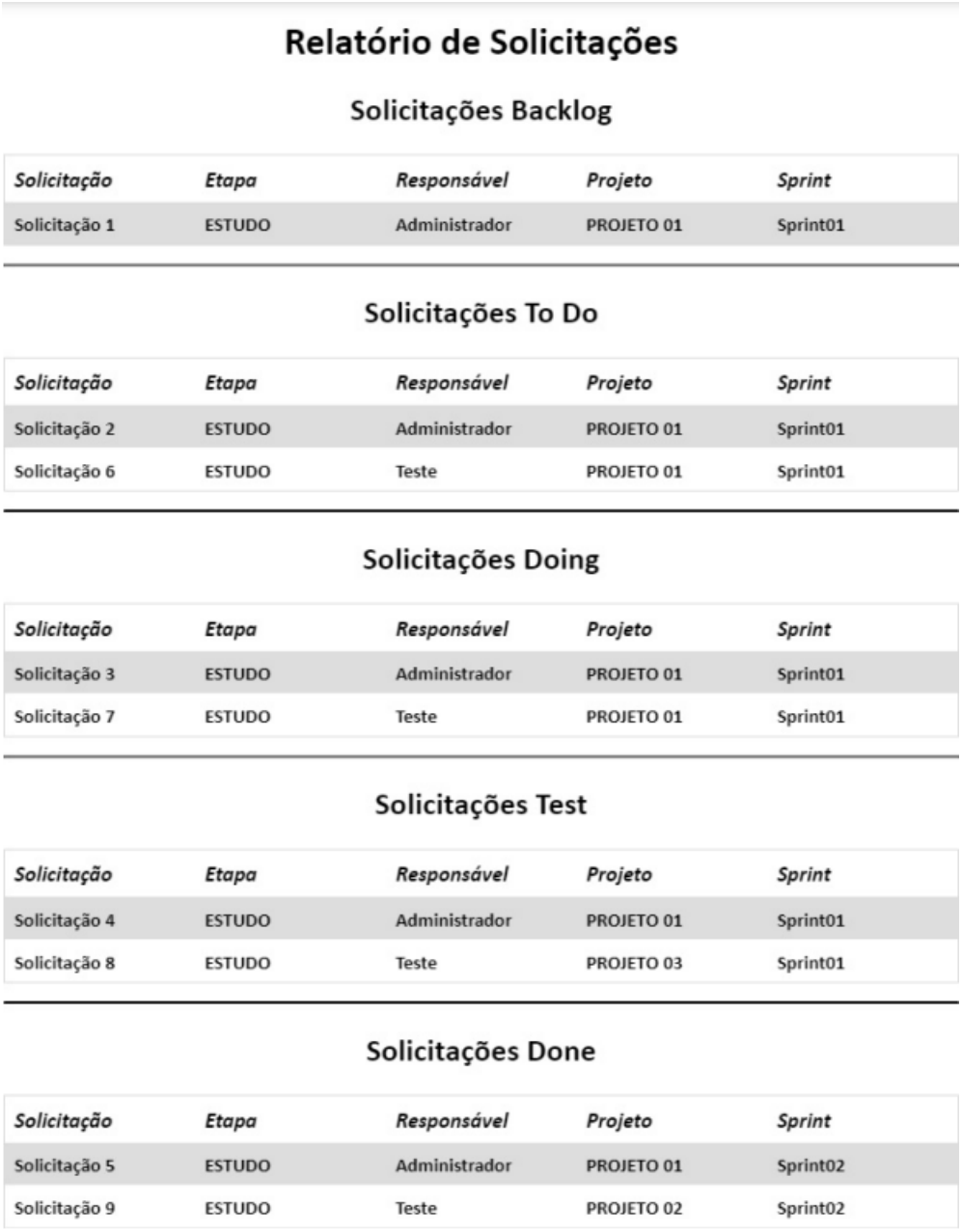
será redirecionado para uma tela que o informará que não pode acessar esta, conforme destacado na Figura 11.

Figura 11 - Tela de usuário sem permissão de acesso

Fonte: Autoria Própria (2022).

O usuário terá a opção de gerar um relatório com as solicitações presentes no quadro Scrum, todas as informações filtradas no quadro serão utilizadas na geração do relatório, um exemplo de relatório pode ser visualizado na Figura 12. Neste estão listados separadamente cada *status* e dentro de cada status as solicitações com esse *status*. Para cada solicitação são demonstrados nome, etapa, responsável, projeto e *sprint*.

Figura 12 - Relatório de Solicitações



Relatório de Solicitações

Solicitações Backlog

Solicitação	Etapa	Responsável	Projeto	Sprint
Solicitação 1	ESTUDO	Administrador	PROJETO 01	Sprint01

Solicitações To Do

Solicitação	Etapa	Responsável	Projeto	Sprint
Solicitação 2	ESTUDO	Administrador	PROJETO 01	Sprint01
Solicitação 6	ESTUDO	Teste	PROJETO 01	Sprint01

Solicitações Doing

Solicitação	Etapa	Responsável	Projeto	Sprint
Solicitação 3	ESTUDO	Administrador	PROJETO 01	Sprint01
Solicitação 7	ESTUDO	Teste	PROJETO 01	Sprint01

Solicitações Test

Solicitação	Etapa	Responsável	Projeto	Sprint
Solicitação 4	ESTUDO	Administrador	PROJETO 01	Sprint01
Solicitação 8	ESTUDO	Teste	PROJETO 03	Sprint01

Solicitações Done

Solicitação	Etapa	Responsável	Projeto	Sprint
Solicitação 5	ESTUDO	Administrador	PROJETO 01	Sprint02
Solicitação 9	ESTUDO	Teste	PROJETO 02	Sprint02

Fonte: autoria Própria (2022).

4.4 Implementação do Sistema

Para iniciar o desenvolvimento do sistema, foi definida uma estrutura de pacotes para manter o projeto organizado e conciso. O sistema foi dividido em dois projetos, sendo que o primeiro é uma API REST que contém *o back-end* da aplicação e o segundo contém a camada visual, ou seja, o *front-end* da aplicação.

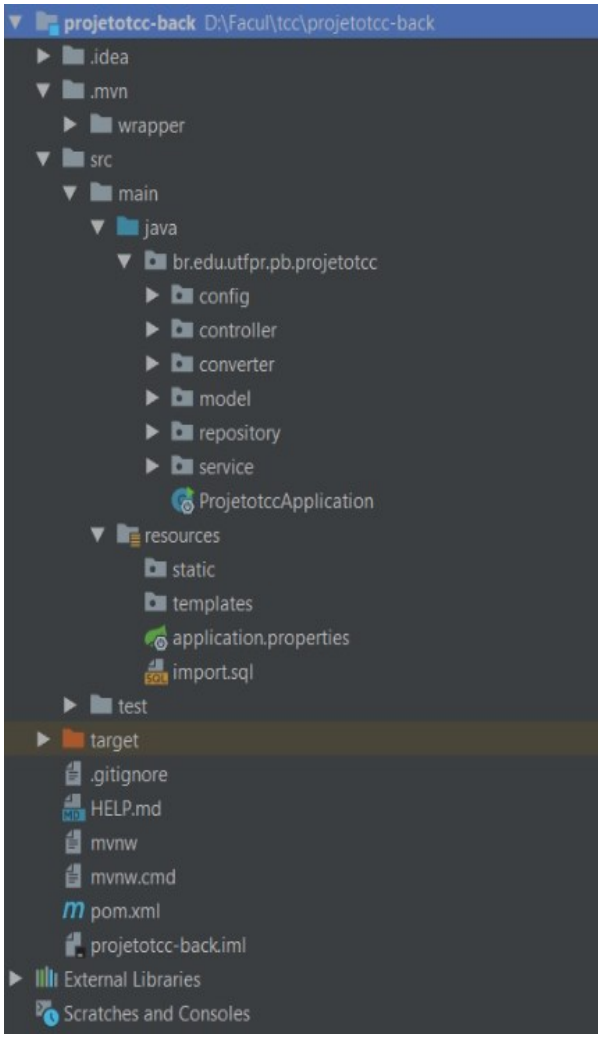
4.4.1 Projeto *Back-end*

Na Figura 13 é possível visualizar a estrutura utilizada durante o desenvolvimento do projeto, em que foi utilizado a linguagem de programação Java com o Spring Framework.

As principais pastas apresentadas são:

a) *Model*: contém todas as classes de persistência do sistema
b) *Repository*: possui todas as classes que herdam as características do Spring JPA, para realizar a persistência de dados no banco.
c) *Controller*: possui os controladores de cada classe do sistema, que é o responsável pela comunicação com o *front-end* por meio de requisições HTTP.
d) *Service*: contém todas as classes de comunicação entre o *controller* e o *repository*, como por exemplo os métodos de salvar, editar e remover.

Figura 13 - Estrutura do projeto back-end



Fonte: Autoria Própria (2022).

Na Listagem 1 é apresentada a classe abstrata utilizada por todos os *controllers*. Essa classe contém os principais métodos utilizados para a manipulação de dados no sistema, são eles:

a) findAll(): responsável por buscar todos os registros de uma determinada tabela.
b) save(): é o responsável por inserir ou atualizar um registro.
c) findOne(): responsável por buscar um determinado registro do banco de dados
d) delete(): responsável por remover um registro.

A utilização dessa classe visa a reutilização de código e boas práticas de programação, visto que todos os formulários terão métodos semelhantes.

Listagem 1 - Classe abstrata de CRUD

```
public abstract class CrudController <T, ID extends Serializable>{
	protected abstract CrudService<T, ID> getService();
@ApiOperation(value = "Retorna uma lista de objetos.")
	@ApiResponses( value = {
		@ApiResponse(code = 200, message = "Lista retornada com sucesso!"),
		@ApiResponse(code = 401, message = "Você não tem permissão para acessar"
					+ "esse recurso!"),
		@ApiResponse(code = 403, message = "O recurso solicitado não está disponível "
					+ "para usuários não autenticados!"),
		@ApiResponse(code = 404, message = "O recurso solicitado não foi encontrado!")
	})
	@GetMapping
	public List<T> findAll(){
		return getService().findAll();
	}
	@GetMapping("page")
	public Page<T> findAll(@RequestParam int page,
						@RequestParam int size,
						@RequestParam(required = false) String order,
						@RequestParam(required = false) Boolean asc){
		PageRequest pageRequest = PageRequest.of(page, size);
		if (order != null && asc != null) {
			pageRequest = PageRequest.of(page, size, asc ? Sort.Direction.ASC :
Sort.Direction.DESC, order );
		}
		return getService().findAll(pageRequest);
	}
	@GetMapping("{id}")
	public T findOne(@PathVariable ID id) {
		return getService().findOne(id);
	}
	@PostMapping
	public T save(@RequestBody @Valid T entity) {
		return getService().save(entity);
	}
	@GetMapping("exists/{id}")
	public boolean exists(@PathVariable ID id) {
		return getService().exists(id);
	}
	@GetMapping("count")
	public long count() {
		return getService().count();
	}
	@DeleteMapping("{id}")
	public void delete(@PathVariable ID id) {
		getService().delete(id);
	}
}
```

Fonte: Autoria Própria (2022).

A Listagem 2 apresenta as configurações de segurança utilizadas pelo *framework* Spring Security para definir quais URLs cada perfil de usuário pode acessar. Nela pode-se observar que algumas funcionalidades do sistema só podem ser acessadas por usuários que tem o perfil de acesso “ADMIN”. As funcionalidades cujas URLs não estão apresentadas na Listagem 2, não possuem uma limitação de acesso por perfil de usuário, ou seja, todos os usuários têm acesso desde que autenticados.

Listagem 2 - Métodos responsável pela definição de acessos URLs

```
@Override
        public void configure(HttpSecurity http) throws Exception {
                http.authorizeRequests()
                .antMatchers(HttpMethod.GET, "/projeto/**").hasAnyRole("ADMIN", "USER")
                                .antMatchers("/projeto/**").hasRole("ADMIN")
                                .antMatchers("/cargo/**").hasRole("ADMIN")
                                .antMatchers("/**").authenticated()
                                .antMatchers("/**").permitAll()
                                .anyRequest().permitAll();
        }
```

Fonte: Autoria Própria (2022).

Na Listagem 3 é apresentado o método *findAllByStatus* que é responsável por preencher o quadro Scrum com as solicitações e direcioná-las para suas respectivas colunas baseado no seu *status* atual. Já o método *findAllByResponsavel* é responsável por fornecer os dados para o *dashboard* da tela inicial do sistema, que mostra os dados do usuário autenticado.

Listagem 3 - Métodos responsáveis pelos cards do quadro scrum.

```
        @GetMapping("busca-status/{status}")
         public List<Solicitacao> findAllByStatus(@PathVariable("status") String status) {
                  return solicitacaoService.findByStatus(status);
         }

         @GetMapping("filtra-quadro")
         public List<Solicitacao> findSolicitacaoByProjetoAndSprintAnAndResponsavel(
         @RequestParam(required = false) Long projetoId,
         @RequestParam(required = false) Long sprintId,
         @RequestParam(required = false) Long responsavelId){
         return (solicitacaoService.findSolicitacaoByProjetoAndSprintAnAndResponsavel
         (projetoId,sprintId,responsavelId));
         }

         @GetMapping("resp")
         public List<Solicitacao> findAllByResponsavel(@RequestParam("responsavelId") Long
responsavelId) {
                  return solicitacaoService.findAllByResponsavel(responsavelId); }
```

Fonte: Autoria Própria (2022).

Ainda na Listagem 3, é possível visualizar o método *findSolicitacaoByProjetoAndSprintAnAndResponsavel que* é utilizado para filtrar o quadro dependendo de quais filtros o usuário está utilizando, podendo buscar por projeto, *sprint* ou responsável.

A Listagem 4 demonstra como o método *findSolicitacaoByProjetoAndSprintAnAndResponsavel* é implementado no *repository* utilizando uma *query* com JPQL para a filtragem.

Listagem 4 - Método com SQL puro no repository

```
@Query("SELECT s FROM Solicitacao s WHERE (:projetoId is null or s.projeto.id = :projetoId) and (:sprintId
is null" + " or s.sprint.id = :sprintId) and (:responsavelId is null or s.responsavel.id = :responsavelId)")
   List<Solicitacao> findSolicitacaoByProjetoAndSprintAnAndResponsavel(
         @Param("projetoId") Long projetoId,
         @Param("sprintId") Long sprintId,
         @Param("responsavelId") Long responsavelId);
```

Fonte: Autoria Própria (2022).

4.4.2 Projeto *Front-end*

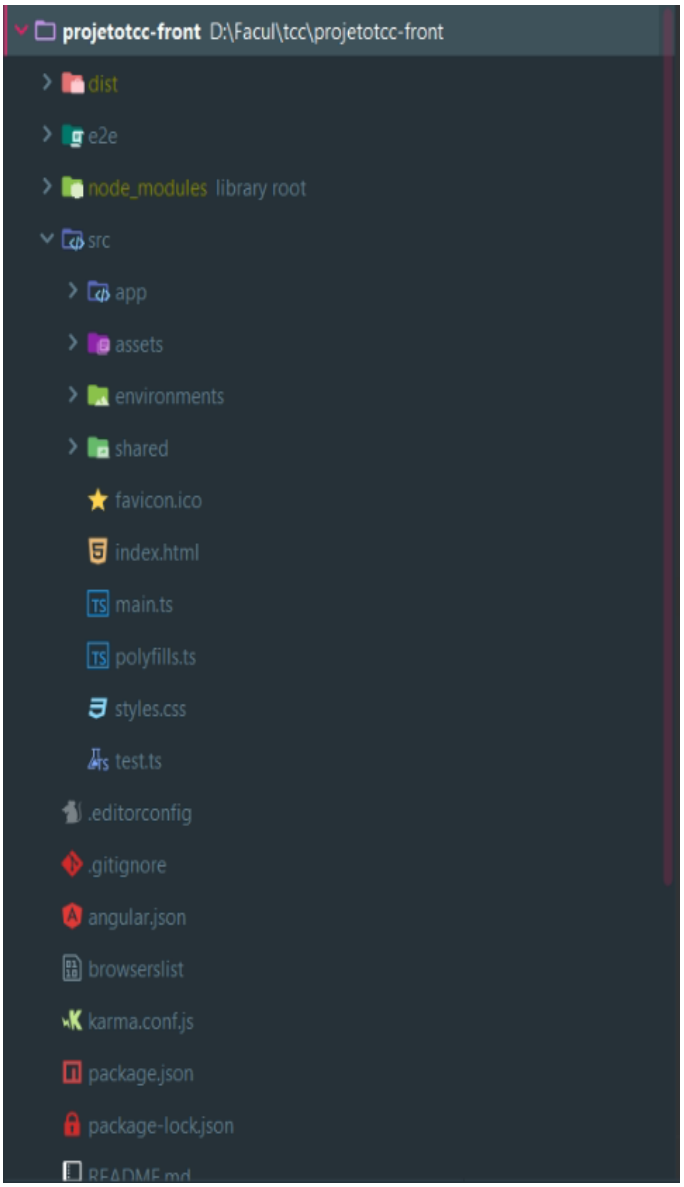
Na Figura 14 é apresentada a estrutura que segue o padrão do Angular CLI, ao criar um projeto. O Angular CLI é uma ferramenta *open source* desenvolvida pelo próprio time do Angular e é utilizado para facilitar a criação de componentes, classes,

services e outros. Utilizando uma série de comandos do Angular CLI não é necessário fazer a criação de todos esses citados acima manualmente.

- Pasta app: ficam todos os arquivos responsáveis pelas funcionalidades do sistema, sendo eles: typescript, html e css.
- Pasta assets: é ficam armazenadas todas as imagens utilizadas no sistema.
- Pasta environments: ficam os arquivos de configurações da API, como por exemplo, o endereço IP e porta que está sendo executado o servidor.

Os demais arquivos são os arquivos padrões do Angular, que são responsáveis por toda a configuração do projeto.

Figura 14 - Estrutura do projeto front-end



Fonte: Autoria Própria (2022).

Na sequência do texto serão apresentados os principais componentes desenvolvidos para aplicação cliente. Na Listagem 5 é apresentado o Solicitação*Component* que possui os métodos básicos, que também são encontrados em outros *Components*, que são os responsáveis pelo controle de inserção, edição, exclusão e busca de registros no *front-end*. Os principais métodos são:

a) carregarCombos(): responsável por buscar os registros de outras entidades que tem ligação direta com a solicitação.
b) findAll(): responsável por buscar os registros cadastrados das solicitações
c) findAllPaged(): busca os registros cadastrados mas os serializa para que possam ser carregados separadamente por página, ideal para listas grandes.
d) newEntity(): responsável por abrir a tela de inserção de um novo registro, para que o usuário possa cadastrá-lo.
e) edit(): responsável por pegar o registro escolhido pelo usuário e abrir a tela de edição para que o usuário possa alterar os dados desse registro.
f) save(): responsável por fazer as chamadas para que o registro em questão, tanto um novo como um sendo editado seja enviado para o *back-end* para ser salvo.
g) delete(): responsável por fazer as chamadas necessárias para a exclusão do registro escolhido pelo usuário.
h) cancel(): cancela a inserção ou edição de um registro.

Listagem 5 – Solicitacao Component

```
export class SolicitacaoComponent implements OnInit {

  @ViewChild('dt', null) dataTable: Table;
  statuss = Object.keys(StatusOpt).map(key => ({ label: StatusOpt[key], value: StatusOpt[key] }));
  solicitacoes: Solicitacao[];
  solicitacaoEdit = new Solicitacao();
  showDialog = false;
  responsaveis: Usuario[];
  projetos: Projeto[];
  etapas: Etapa[];
  sprints: Sprint[];
  totalRecords: number;
  maxRecords = 10;
  currentPage = 1;

  constructor(private solicitacaoService: SolicitacaoService,
            private confirmationService: ConfirmationService,
            private projetoService: ProjetoService,
            private etapaService: EtapaService,
            private sprintService: SprintService,
            private usuarioService: UsuarioService,
            private messageService: MessageService
            ) {
}

  ngOnInit() {
    this.carregarCombos();
    this.findAll();
  }

  carregarCombos() {
    this.projetoService.findAll().subscribe(e => this.projetos  = e );
    this.etapaService.findAll().subscribe(e => this.etapas  = e );
    this.sprintService.findAll().subscribe(e => this.sprints  = e );
    this.usuarioService.findAll().subscribe(e => this.responsaveis  = e );
  }
  findAll() {
    this.solicitacaoService.findAll().subscribe( e => this.solicitacoes = e);
  }
```

```
findAllPaged(page: number, size: number) {
   this.solicitacaoService.findPageable(page, size).subscribe(e => {
     this.solicitacoes = e.content;
     this.totalRecords = e.totalElements;
   });
 }
 lazyLoad(event: LazyLoadEvent) {
   const pageNumber = event.first / event.rows;
   this.currentPage = pageNumber;

   this.maxRecords = event.rows;

   setTimeout( () => {
     this.findAllPaged(this.currentPage, this.maxRecords);
   }, 250);
 }

 newEntity() {
   this.solicitacaoEdit = new Solicitacao();
   this.solicitacaoEdit.projeto = this.projetos[0];
   this.solicitacaoEdit.sprint = this.sprints[0];
   this.solicitacaoEdit.etapa = this.etapas[0];
   this.solicitacaoEdit.responsavel = this.responsaveis[0];
   this.solicitacaoEdit.status = this.statuss[0].value;
   this.showDialog = true;
 }

 save() {
   this.solicitacaoService.save(this.solicitacaoEdit).subscribe( e => {
       this.solicitacaoEdit = new Solicitacao();
       this.findAll();
       this.showDialog = false;
       this.messageService.add({severity: 'success', summary: 'Confirmado',
         detail: 'Registro salvo com sucesso!'});
     },
     error => {
       this.messageService.add({severity: 'error', summary: 'Erro',
         detail: 'Falha ao salvar o registro!'});
     }
   );
 }

 cancel() {
   this.showDialog = false;
   this.solicitacaoEdit = new Solicitacao();
 }
```

```
edit(solicitacao: Solicitacao) {
    this.solicitacaoEdit = Object.assign({}, solicitacao);
    this.showDialog = true;
  }

  delete(solicitacao: Solicitacao) {
    this.confirmationService.confirm({
      message: 'Esta ação não poderá ser desfeita!',
      header: 'Deseja remover o registro?',
      acceptLabel: 'Confirmar',
      rejectLabel: 'Cancelar',
      accept: () => {
        this.solicitacaoService.delete(solicitacao.id).subscribe(() => {
          this.dataTable.reset();
          this.messageService.add({severity: 'success', summary: 'Confirmado',
            detail: 'Registro removido com sucesso!'});
        }, error => {
          this.messageService.add({severity: 'error', summary: 'Erro',
            detail: 'Falha ao remover o registro!'});
        });
      }
    });
  }
}
```

Fonte: Autoria Própria (2022).

No *component* do QuadroScrum, o usuário fará a movimentação dos *cards* para atualizar os status das solicitações, o método responsável por essa atualização é o *drop()*, que é demonstrado pela Listagem 6. Quando um *card* é solto em uma coluna o seu *status* é automaticamente alterado para o representado pela coluna de destino.

Listagem 6 - Método de atualização dos cards no movimento

```
drop(event: CdkDragDrop<Object[]>) {
    if (event.previousContainer === event.container) {
      moveItemInArray(event.container.data, event.previousIndex, event.currentIndex);
    } else {
      transferArrayItem(
        event.previousContainer.data,
        event.container.data,
        event.previousIndex,
        event.currentIndex,
      );
    }
    this.containerStatus = event.container.element.nativeElement.getAttribute('id');
    this.solicitacaoEdit = Object.assign({},this.solicitacoes.find(solicitacao => solicitacao.id ===
event.container.data[event.currentIndex]['id']))
    this.solicitacaoEdit.status = this.containerStatus;
    this.save();
  }
```

Fonte: Autoria Própria (2022).

Como pode ser observado na Listagem 6, quando o *card* é solto na coluna destino o sistema atualiza o registro no banco de dados alterando o status da solicitação em questão e chamando o método *save* para que a requisição de alteração seja enviada ao servidor.

No primeiro momento em que o quadro é aberto ele é carregado pelo método findSolicitacoes que está demonstrado na Listagem 7. Esse método utiliza o resultado da requisição HTTP do *service* de solicitação que traz os objetos que são as solicitações. Após isso esses objetos são manipulados através do método *filter* (metodo padrão de *array)* que filtra a lista conforme um parâmetro passado a ela, nesse caso filtramos o status atual para separar em diferentes listas as solicitações, agrupando-as por status.

Listagem 7 – findSolicitacoes, método de busca dos cards

```
findSolicitacoes() {
   this.solicitacaoService.findAll().subscribe( e => {
     this.solicitacoes = e;
     this.solicitacoesBacklog = this.solicitacoes.filter(solicitacao => solicitacao.status === "Backlog");
     this.solicitacoesTodo = this.solicitacoes.filter(solicitacao => solicitacao.status === "To Do");
     this.solicitacoesDoing = this.solicitacoes.filter(solicitacao => solicitacao.status === "Doing");
     this.solicitacoesTest = this.solicitacoes.filter(solicitacao => solicitacao.status === "Test");
     this.solicitacoesDone = this.solicitacoes.filter(solicitacao => solicitacao.status === "Done");
   }
   );
  }
```

Fonte: Autoria Própria (2022).

O quadro também pode ser filtrado por projeto, *sprint*, e responsável, o método responsável por fazer esse filtro é o *findSolicitacoesFiltrado*, conforme destacado na Listagem 8. Esse método irá fazer uma chamada ao método *findFiltro* da classe solicitacaoService, que espera um projeto, sprint e responsável como parâmetros para que o servidor faça uma pesquisa no banco de dados buscando todas as solicitações que estão tem como valor esses filtros. Os filtros passados podem ser nulos, ou seja, é possível filtrar apenas por um ou dois dos parâmetros. Após ter o retorno do banco de dados o sistema irá fazer uma separação novamente usando o método *filter* do *array*, agrupando novamente as solicitações por status, mas dessa vez apenas as solicitações filtradas com o que o usuário escolheu.

Listagem 8 - Método para busca dos cards pelos filtros

```
findSolicitacoesFiltrado(projetoId: number, sprintId: number, responsavelId: number){
   this.solicitacaoService.findFiltro(projetoId,sprintId,responsavelId).subscribe( e => {
      this.solicitacoes = e;
      this.solicitacoesBacklog = this.solicitacoes.filter(solicitacao => solicitacao.status === "Backlog");
      this.solicitacoesTodo = this.solicitacoes.filter(solicitacao => solicitacao.status === "To Do");
      this.solicitacoesDoing = this.solicitacoes.filter(solicitacao => solicitacao.status === "Doing");
      this.solicitacoesTest = this.solicitacoes.filter(solicitacao => solicitacao.status === "Test");
      this.solicitacoesDone = this.solicitacoes.filter(solicitacao => solicitacao.status === "Done");
     }
    );
  }
```

Fonte: Autoria Própria (2022).

A Listagem 9 demonstra a lógica do método *findFiltro,* que faz uma chamada no *solicitacaoService*. Como mencionado anteriormente com as suas respectivas validações para que algum dos parâmetros de filtro possa ser nulos e mesmo assim retornar uma lista filtrada ao usuário.

Listagem 9 - Método que faz a lógica dos filtros

```
findFiltro(projetoId: number, sprintId: number, responsavelId: number) {
     let url = `${this.getUrl()}/filtra-quadro`;
     if (projetoId != null) {
       url += `?projetoId=${projetoId}`;
     }
     if (projetoId == null && sprintId != null) {
       url += `?sprintId=${sprintId}`;
     }
     else if(sprintId != null){
       url += `&sprintId=${sprintId}`;
     }
     if (projetoId == null && sprintId == null && responsavelId != null) {
       url += `?responsavelId=${responsavelId}`;
     }
     else if(responsavelId != null){
       url += `&responsavelId=${responsavelId}`;
     }
     if(projetoId == null && sprintId == null && responsavelId == null){
       return this.findAll();
     }
     return this.http.get<Solicitacao[]>(url);
  }
```

Fonte: Autoria Própria (2022).

Na Listagem 10 pode ser visualizado o método *limparFiltros*. Tendo o quadro filtrado, o usuário pode limpar os filtros e voltar a ter todas as solicitações aparentes no quadro, o método *limparFiltros* é o responsável por fazer essa ação, alterando o

valor selecionado pelo usuário nos filtros para nulo e buscando novamente todas as solicitações utilizando o método *findSolicitacoes*.

Listagem 10 - Método que limpa os filtros e recarrega os cards

```
limparFiltros(){
    this.selectedProjeto = null;
    this.selectedSprint = null;
    this.selectedResponsavel = null;
    this.carregarCombos();
    this.findSolicitacoes();
  }
```

Fonte: Autoria Própria (2022).

Também no quadro o usuário pode gerar um relatório trazendo todas as solicitações mostradas no quadro. A Figura 12 exibe o exemplo de um relatório gerado. Na Listagem 11 os métodos *gerarRelatorio e createPDF* são os responsáveis por gerar o relatório, assim como o método *print()* apresentado na Listagem 12.

Listagem 11 – Métodos que geram o relatório

```
gerarRelatorio() {
   const title = 'Relatório de Solicitações';
   this.createPDF(title,this.solicitacoesBacklog, this.solicitacoesTodo, this.solicitacoesDoing,
this.solicitacoesTest, this.solicitacoesDone);
  }
  private createPDF(title: string, solicitacoesBacklog: Solicitacao[],
                solicitacoesTodo: Solicitacao[],
                solicitacoesDoing: Solicitacao[],
                solicitacoesTest: Solicitacao[],
                solicitacoesDone: Solicitacao[],): void {
   this.gerarPdf.clear();
   const factory = this.resolver.resolveComponentFactory(PdfPageComponent);
   const componentRef = this.gerarPdf.createComponent(factory);
   componentRef.instance.title = title;
   componentRef.instance.solicitacoesBacklog = solicitacoesBacklog;
   componentRef.instance.solicitacoesTodo = solicitacoesTodo;
   componentRef.instance.solicitacoesDoing = solicitacoesDoing;
   componentRef.instance.solicitacoesTest = solicitacoesTest;
   componentRef.instance.solicitacoesDone = solicitacoesDone;
   componentRef.instance.emitter.subscribe(() => {
    const config = {
     filename: 'relatório.pdf',
     margin: 1,
    };
    this.print(componentRef.location.nativeElement, config);
    componentRef.destroy();
   });
  }
```

Fonte: Autoria Própria (2022).

Listagem 12 - Método responsável por gerar o PDF do relatório

```
private print(content: any, config: any): void {
    console.log("teste print.");
    html2pdf()
      .set(config)
      .from(content)
      .toPdf()
      .outputPdf('dataurlnewwindow');
  }
```

Fonte: Autoria Própria (2022).

No método *gerarRelatorio* é passado como parâmetro o título e então é realizada a chamada ao método *createPDF* que é responsável por preencher o relatório com os dados do quadro. E o método *print* utiliza os dados dos outros métodos e gera o *download* do arquivo em formato *.pdf*.

O Principal*Component* que é o *component* responsável por criar o *dashboard* com as informações do usuário na página inicial do sistema, dentro dele temos o método *findSolicitacoes,* destacado na Listagem 13, que irá buscar as solicitações vinculadas ao usuário e utilizar os respectivos dados para fazer a criação do gráfico, nos *chartOptions* são passadas as características dos gráficos.

Listagem 13 - Método que preenche os dados do dashboard

```
findSolicitacoes(){
   this.solicitacaoService.findSolicitacoesUsuario(this.usuarioId).subscribe( e => {
    this.solicitacoes = e;
    this.solicitacoesBacklog = this.solicitacoes.filter(solicitacao => solicitacao.status === "Backlog");
    this.solicitacoesTodo = this.solicitacoes.filter(solicitacao => solicitacao.status === "To Do");
    this.solicitacoesDoing = this.solicitacoes.filter(solicitacao => solicitacao.status === "Doing");
    this.solicitacoesTest = this.solicitacoes.filter(solicitacao => solicitacao.status === "Test");
    this.solicitacoesDone = this.solicitacoes.filter(solicitacao => solicitacao.status === "Done");
    this.data = {
     labels: ['Backlog','To Do','Doing','Test','Done'],
     datasets: [
       {data: [this.solicitacoesBacklog.length,
               this.solicitacoesTodo.length,
               this.solicitacoesDoing.length,
               this.solicitacoesTest.length,
               this.solicitacoesDone.length],
        backgroundColor: [
          cores.corBacklog,
          cores.corToDo,
          cores.corDoing,
          cores.corTest,
          cores.corDone,
        ],
        hoverBackgroundColor: [
          cores.corBacklog,
          cores.corToDo,
          cores.corDoing,
          cores.corTest,
          cores.corDone,
        ]
       } ]
    };
    this.chartOptions = {
     legend: {
       display: true,
       position: 'bottom',
       labels: {
         fontColor: "white",
         usePointStyle: true
       }, }
    }  });
  }
```

Fonte: Autoria Própria (2022).

Na Listagem 14 são apresentados os métodos *findProjetos* e *findSprints*, que são responsáveis por buscar os dados de projetos e *sprints* do usuário autenticado e mostrá-los no *dashboard*.

Listagem 14 – Métodos que trazem os dados de projeto e sprints

```
findProjetos(){
   this.projetosAtivos = 0;
   this.projetoService.findAll().subscribe(e=>{
    this.projetos = e;

    for(var projeto of this.projetos){
     if(projeto.responsavel.id == this.usuarioId && projeto.dataFim == null){
      this.projetosAtivos++;
     }
    }
   })
  }
  findSprints(){
   this.sprintsAtivos = 0;
   this.sprintService.findAll().subscribe(e=>{
    this.sprints = e;
    for(var sprint of this.sprints){
     if(sprint.responsavel.id == this.usuarioId && sprint.dataFim == null){
      this.sprintsAtivos++;
     }
    }
   })
  }
findProjetos(){
   this.projetosAtivos = 0;
   this.projetoService.findAll().subscribe(e=>{
    this.projetos = e;
    for(var projeto of this.projetos){
     if(projeto.responsavel.id == this.usuarioId && projeto.dataFim == null){
      this.projetosAtivos++;
     }
    }
   })
  }
  findSprints(){
   this.sprintsAtivos = 0;
   this.sprintService.findAll().subscribe(e=>{
    this.sprints = e;
    for(var sprint of this.sprints){
     if(sprint.responsavel.id == this.usuarioId && sprint.dataFim == null){
      this.sprintsAtivos++;
     }
    }
   })
  }
```

Fonte: Autoria Própria (2022).

5. CONCLUSÃO

Neste trabalho foi desenvolvido um sistema web para a gestão de demandas de equipes com base nos conceitos da metodologia SCRUM. O projeto foi implementado utilizando as tecnologias Java com o Spring Framework para o *back-end* e o *framework* Angular para o *front-end.*

Todos os objetivos propostos ao trabalho foram alcançados, sendo o principal proporcionar a equipe que utilizará o software uma visão do andamento do projeto. Algumas melhorias foram feitas no banco de dados apresentado no trabalho de conclusão de curso 1, visando melhorar o desempenho da aplicação e também foram feitas modificações visuais para melhor experiência dos usuários.

Por não trabalhar diretamente com as tecnologias usadas, o autor foi desafiado para aprender as ferramentas e conseguir a melhor implementação do projeto. Resultando em dúvidas durante o desenvolvimento das rotinas do sistema relacionadas ao uso de alguns componentes, entretanto essas dúvidas foram sanadas por meio de pesquisas na documentação das bibliotecas, fóruns de discussão e com a ajuda do orientador.

No *front-end* foram utilizadas várias bibliotecas para se obter os resultados desejados, pois somente o *framework* Angular e os componentes do PrimeNG não conseguiram suprir todas as demandas necessárias, tais como *Bootstrap* e *Bulma*.

Com trabalhos futuros esse sistema poderá ser melhorado a fim que equipes de grande porte possam utilizar-lo para a gestão de demandas, criar diferentes controles, mais modelos de relatórios, aumentar a capacidade do *dashboard* e implementar melhorias visuais, sempre tendo a atenção para que os conceitos do SCRUM estejam presentes.

O Repositório com o código fonte da aplicação está disponível em: https://github.com/AfonsoSouza12/TCC

REFERÊNCIAS

ARTIA. **Gestão de projetos**: o que é e tudo sobre como gerenciar projetos. 2019. Disponível em: <https://artia.com/blog/gestao-de-projetos-o-que-e-para-que-serve/>. Acesso em: 18 mar. 2019.

CARNEIRO, L. B.; SILVA, A. C. C. L. M.; ALENCAR, L. H. **Scrum agile project management methodology application for workflow management**: a case study. Proceedings of the 2018 IEEE IEEM. 2018, p. 938-942.

CERVONE, H. F. Understanding agile project management methods using Scrum, **OCLC Syst. Serv. Int. Digit. Libr. Perspect**., v. 27, n.1, p. 18–22, fev. 2011.
CHING, P. M.; MUTUC, J. E. **Modeling the dynamics of an agile Scrum team in the development of a single software project**. Proceedings of the 2018 IEEE IEEM, 2018, p. 386-390

DARWISH, N. R.; RIZK, N. M. Multi-Dimensional Success Factors of Agile Software Development Projects. **International Journal of Computer Applications**, 2015, p. 118-133.

GLAIEL, Firas; MOULTON, Allen; MADNICK, Stuart. **Agile project dynamics: A system dynamics investigation of agile software development methods**. 2014. Disponível em: <https://cams.mit.edu/wp-content/uploads/2013-05.pdf>. Acesso em: 08 abr. 2019.

GUIA PMBOK. **Conhecimento em gerenciamento de projetos.** Parte 1, 6ª ed. 2017.

HARB, Y.; NOTEBOOM, C.; SARNIKAR, S. Evaluating project characteristics for selecting the best-fit agile software development methodology: a teaching case. **Journal of the Midwest Association for Information Systems**, v. 1, n. 1, 2015, p33-55.

HIGHSMITH, J.; COCKBURN, A. Agile software development: The business of innovation, **Computer***,* v. 34, n. 9, p. 120 - 127, 2001.

JAAKKOLA, H.; THALHEIM, B. **A framework for systematic change management**. In: European Conference on Software Process Improvement. Springer Berlin Heidelberg, Jun. 2014, pp. 60-72.

KENETT, R. S. Implementing Scrum using Business Process Management and Pattern analysis methodologies. **Dyn. Relationships Manag. J.**, v. 2, n. 2, p. 29–48, 2013.

KHALIL, Asif; KOTAIAH, Bonthu. **Implementation of Agile Methodology based on SCRUM Tool**. In: International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS-2017). IEEE, 2017, p. 2351-2357

MAY, J.; YORK, J. LANE, M. Play ball: bringing Scrum into the classroom, **Jourrnal Inf. Syst. Educ**., v. 27, n. 2, p. 87-93, 2016.

OLIVEIRA, E. C. B. de; ALENCAR, L. H.; COSTA, A. P. C. S. Decision process of allocating projects to project managers, **Prod. Plan. Control**, v. 29, n. 8, jun., p. 645–654, 2018.

PMI. Project Management Institute. **A guide to the project management body of knowledge**, 6 ed, 2017.

ROMANO, Breno Lisi; SILVA, Alan Delgado da. **Project management using the Scrum agile method:** a case study within a small enterprise. 12th International Conference on Information Technology - New Generations, 2015, p. 774-776.

SCHWABER, Ken; SUTHERLAND, Jeff. **The definitive guide to Scrum**: the rules of the game, The Scrum Guide Org., 2017. Disponível em: https://www.scrumguides.org/docs/. Acesso em: 20 mar. 2019.

SCRUM STUDY. **Um Guia para o conhecimento em SCRUM (GUIA SBOK™).** Edição 2013.

SCRUM. **DesenvolvimentoAgil.com.br**. 2019. Disponível em: https://www.desenvolvimentoagil.com.br/scrum/. Acesso em: 22 mar. 2019.

SCRUM.ORG. **What is Scrum?** 2019. Disponível em: https://www.scrum.org/resources/what-is-scrum. Acesso em: 18 mar. 2019.

SITEWARE. **O que é SCRUM?** 2019. Disponível em: https://www.siteware.com.br/projetos/ferramentas-para-scrum/. Acesso em: 23 mar. 2019.

ŠPUNDAK, M. Mixed agile/traditional project management methodology: reality or illusion? **Procedia - Soc. Behav. Sci.**, v. 119, p. 939–948, 2014.