UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE

DEPARTAMENTO DE COMPUTAÇÃO
CIÊNCIA DA COMPUTAÇÃO

RÔMULO BORGES DE ALMEIDA

PROPOSTA DE SISTEMA DE GERENCIAMENTO DE FASES E ARTEFATOS DE UM PROJETO DE SOFTWARE

ALEGRE - ES
AGOSTO 2022

PROPOSTA DE SISTEMA DE GERENCIAMENTO DE FASES E ARTEFATOS DE UM PROJETO DE SOFTWARE

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

por

RÔMULO BORGES DE ALMEIDA

Orientador
GIULIANO PRADO DE MORAIS GIGLIO

Universidade Federal do Espírito Santo

ALEGRE - ES

AGOSTO 2022

RÔMULO BORGES DE ALMEIDA

SISTEMA DE GERENCIAMENTO DE FASES E ARTEFATOS DE UM PROJETO DE SOFTWARE

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 09 de agosto de 2022.

COMISSÃO EXAMINADORA

Prof. MSc. Giuliano Prado de Morais Giglio
Universidade Federal do Espírito Santo
Orientador

Prof. DSc. Clayton Vieira Fraga Filho
Universidade Federal do Espírito Santo

Prof. DSc. Marcelo Otone Aguiar
Universidade Federal do Espírito Santo

SUMÁRIO

1 Introdução **15**

1.1 O problema e sua importância 17

1.2 Objetivos 17

1.2.1 Objetivo geral 17

1.2.2 Objetivos específicos 18

2 Revisão da literatura **19**

2.1 Softwares similares 19

2.1.1 Trello 19

2.1.2 Jira 20

2.1.3 Especificidades da aplicação com softwares similares 21

2.2 Arquitetura do sistema 22

2.2.1 API 23

2.2.2 REST 23

2.2.3 Projeto da arquitetura 25

2.3 Tecnologias utilizadas 26

2.3.1 Angular 26

2.3.2 TypeScript 26

2.3.3 JSON 26

2.3.4 Spring Framework 27

2.3.5 Java 27

2.3.6 PostgreSQL 28

2.3.7 Uso das tecnologias na arquitetura proposta 28

3 Metodologia 30

3.1 Metodologia de desenvolvimento 30

3.2 Elicitação e análise de requisitos 31

3.2.1 Descrição dos usuários 32

3.2.2 Requisitos funcionais 32

3.2.3 Casos de uso 36

3.2.4 Modelo de domínio 42

3.3 Modelagem de banco de dados 43

3.3.1 Descrição de dados 44

3.4 Garantia de qualidade 45

4 Desenvolvimento da aplicação 46

4.1 Modelo lógico 46

4.2 Backend 47

4.2.1 Estrutura do projeto 47

4.2.2 Conexão com banco de dados 48

4.2.3 Endpoints 49

4.2.4 Façades 50

4.2.5 Services 51

4.2.6 Entidades 53

4.2.7 Segurança 54

4.3 Frontend 56

4.3.1 Estrutura do frontend 57

4.3.2 Comunicação com servidor 58

4.3.3 Navegação 61

4.4 Interfaces 62
4.4.1 Tela de projetos 63
4.4.2 Requisitos funcionais e não funcionais 66
4.4.3 Artefatos 67
4.4.4 Descrição de tabelas do banco de dados 70
4.4.5 Geração de relatório das entidade 72
4.4.6 Testes 74
4.4.7 Geração de relatório de incidentes de testes funcionais 76
4.4.8 Sprints 78

5 Testes e garantia de qualidade 80
5.1 Teste unitário 80
5.2 Teste de desenvolvimento 82
5.3 Teste de usabilidade 82

6 Resultados alcançados e considerações finais 84
6.1 Melhorias e trabalhos futuros 85
6.1.1 Usuário do sistema 85
6.1.2 Projeto de software 85
6.1.3 Requisitos 85
6.1.4 Descrição de tabelas do banco de dados 86
6.1.5 Análise de riscos 86
6.1.6 Método ágil 86
6.1.7 Relatórios 87
6.1.8 Testes 87
6.1.9 Garantia de qualidade 87
6.1.10 Integração com plataformas de hospedagem de código fonte 87

Referências 89

Apêndice A – Cenários de caso de uso 92

Apêndice B – Descrição dos dados 146

LISTA DE FIGURAS

Figura 1 Trello 20

Figura 2 Jira 21

Figura 3 Arquitetura cliente-servidor 24

Figura 4 Arquitetura do sistema 25

Figura 5 Processo Scrum 30

Figura 6 O processo de elicitação e análise de requisitos 31

Figura 7 Diagrama de Caso de Uso 1 37

Figura 8 Diagrama de Caso de Uso 2 37

Figura 9 Modelo de domínio 42

Figura 10 Modelo entidade-relacionamento 44

Figura 11 Modelo lógico 46

Figura 12 Estrutura do projeto *backend* 47

Figura 13 Informações da conexão com banco de dados 48

Figura 14 Exemplo de repository 49

Figura 15 Exemplo de controller 50

Figura 16 Exemplo de facade 51

Figura 17 Exemplo de service 52

Figura 18 Exemplo de entidade 53

Figura 19 Exemplo de login via backend 54

Figura 20 Exemplo de JWT decodificado 55

Figura 21 JWT no header da requisição 56

Figura 22 Exemplo de rota que exige autorização 56

Figura 23 Estrutura do frontend 57

Figura 24 BaseStore 58

Figura 25 Arquivo de *environment* 59

Figura 26 Exemplo de *store* 59

Figura 27 Interceptor para *token* de acesso 60

Figura 28 TokenStorageService 60

Figura 29 Arquivo de configuração do roteamento 61

Figura 30 Exemplo de navegação entre rotas 62

Figura 31 Implementação do DataService 62

Figura 32 Tela de listagem de projetos vazia 63

Figura 33 Modal de novo projeto 63

Figura 34 Tela de listagem de projetos com registro 64

Figura 35 Modal informativa sobre legenda 64

Figura 36 Legenda da tela de *overview* do projeto 65

Figura 37 Tela de *overview* do projeto 65

Figura 38 Tela de listagem de requisitos 66

Figura 39 Tela de cadastro de requisitos 66

Figura 40 Tela de edição de requisito 67

Figura 41 Tela de listagem de modelos conceituais de dados 68

Figura 42 Tela de cadastro de artefatos 68

Figura 43 Tela de edição de artefatos 69

Figura 44 Tela de listagem de entidades do banco de dados 70

Figura 45 Tela de cadastro de entidades do banco de dados 71

Figura 46 Alerta de formulário de atributos 71

Figura 47 Tela de cadastro de entidade do banco de dados com atributos 72

Figura 48 Geração de relatório de entidades 73

Figura 49 Listagem dos relatórios de entidades 73

Figura 50 Exemplo de relatório de entidades 74

Figura 51 Tipos de teste do módulo "Testes" 74

Figura 52 Listagem de testes unitários 75

Figura 53 Cadastro de teste unitário 75

Figura 54 Edição de teste unitário 76

Figura 55 Listagem de testes funcionais 76

Figura 56 Geração de relatório de incidentes de testes funcionais 77

Figura 57 Listagem de relatório de incidentes de testes funcionais 77

Figura 58 Exemplo de relatório de incidentes de testes funcionais 78

Figura 59 Tela de listagem de sprints 78

Figura 60 Tela de cadastro de sprints 79

Figura 61 Tela de edição de sprint 79

Figura 62 Exemplo de teste unitário 81

Figura 63 Resultados dos testes unitários 81

Figura 64 Resultados testes de desenvolvimento 82

Figura 65 Resultados testes de usabilidade 83

LISTA DE TABELAS

Tabela 1 Descrição dos usuários 33

Tabela 2 Descrição de dados da entidade Projeto 45

Tabela 3 Descrição da estrutura do projeto *backend* 48

Tabela 4 Descrição da estrutura do projeto *frontend* 58

Tabela 5 Tipo de artefato e seu arquivo exigido 70

Tabela 6 Descrição de dados da entidade "arquivo" 146

Tabela 7 Descrição de dados da entidade "artefato" 146

Tabela 8 Descrição de dados da entidade "atributo" 147

Tabela 9 Descrição de dados da entidade "entidade" 147

Tabela 10 Descrição de dados da entidade "etapa_projeto" 147

Tabela 11 Descrição de dados da entidade "historico_acao_projeto" 148

Tabela 12 Descrição de dados da entidade "projeto" 148

Tabela 13 Descrição de dados da entidade "relatorio" 148

Tabela 14 Descrição de dados da entidade "requisito" 149

Tabela 15 Descrição de dados da entidade "role" 149

Tabela 16 Descrição de dados da entidade "sprint" 149

Tabela 17 Descrição de dados da entidade "sprint_entidade" 149

Tabela 18 Descrição de dados da entidade "sprint_requisito" 150

Tabela 19 Descrição de dados da entidade "sprint_teste" 150

Tabela 20 Descrição de dados da entidade "sprint_versao_artefato" 150

Tabela 21 Descrição de dados da entidade "teste" 150

Tabela 22 Descrição de dados da entidade "users" 151

Tabela 23 Descrição de dados da entidade "versao_artefato" 151

LISTA DE SIGLAS

GP	Gerenciamento de Projetos
MVC	Model-View-Controller
API	Applications Protocolo Interface
HTTP	HyperText Transfer Protocol
REST	REpresentational State Transfer
URI	Uniform Resource Identifier
JSON	JavaScript Object Notation
JPA	Java Persistence API
UC	Caso de uso
RS	Resposta do sistema
FA	Fluxo alternativo
EV	Evento de sistema
MSG	Mensagem
FE	Fluxo de exceção
PO	Product Owner

RESUMO

Este trabalho apresenta uma proposta de desenvolvimento de uma aplicação web para a gestão de fases e artefatos de um projeto de *software*, fornecendo versionamento para artefatos produzidos em ferramentas externas como por exemplo casos de uso, diagrama de classes, modelo lógico de dados, protótipos e fornecendo gerenciamento dentro do próprio sistema para requisitos, testes, descrições de tabelas e *sprints*. O objetivo do trabalho foi criar um sistema que torne o gerenciamento de artefatos de *software* mais acessível e flexível, permitindo uma visão mais ampla do processo de desenvolvimento de *software*. Para o desenvolvimento da proposta foram utilizadas no *backend* tecnologias como Java, Spring Framework e banco de dados PostgreSQL. Já para o desenvolvimento do *frontend* do sistema foi utilizado o Angular.

Palavras-chave: Artefatos; Fase; Projeto de software; Java; Spring Framework; Angular.

1 INTRODUÇÃO

A adoção de *softwares* e da computação no mundo é cada vez maior e permeia quase todos os ramos da sociedade moderna. Empresas de diversos tamanhos dependem de aplicações para automatizar seus processos. Governos interagem com cidadãos através de sistemas de informação, seja para coletar impostos, divulgar iniciativas ou realizar eleições (VALENTE, 2020). Portanto, é de suma importância o desenvolvimento e manutenção de *softwares* que atendam as realidades atuais.

O desenvolvimento de sistemas de *software* vem aumentando a cada ano e a velocidade desse crescimento foi recentemente acelerada. Devido a pandemia do COVID-19, diversas empresas foram obrigadas a alterar a estrutura de seu trabalho em função de *lockdowns* e do *home-office*. Com isso, para continuarem a conduzir seus negócios, muitas empresas investiram na introdução de novos produtos, no aumento da produtividade e na redução de custos. Tendo em vista tal fato, o setor de *software* no Brasil apresentou um crescimento expressivo de 28,7% no ano de 2020, onde o segmento de IoT cresceu com uma taxa de 8% em comparação ao ano de 19 (ABES, 2021). Enquanto isso, no mesmo período, o mercado doméstico de serviços e o PIB sofreram queda (IBGE, 2021).

Dentre os diversos projetos de desenvolvimento de *software*, nem todos chegam ao seu fim. Muitos fracassam ou são cancelados por não atenderem totalmente aos requisitos dos clientes ou por excederem o prazo e orçamento estimados. Apesar de não haver uma explicação simples para isso, as deficiências nos requisitos são consideradas uma das principais causas para tal (ESPINDOLA; MAJDENBAUM; AUDY, 2004).

Todo esse desenvolvimento de *software* da atualidade traz atenção também para a manutenção dos mesmos. Tendo essa importância por conta da crescente quantidade de *softwares* em funcionamento nas empresas e por precisarem continuar funcionando sem serem substituídos. Com isso, essa necessidade de manutenção traz

consigo diversos problemas, originados tanto da própria administração da organização quanto do perfil dos clientes ou técnicas não suficientes para a construção do *software* (PADUELLI, 2007). Tendo em vista tais pontos apresentados, torna-se clara a necessidade de utilização de alguma técnica para o auxílio no processo de desenvolvimento de *software*.

O Gerenciamento de Projetos (GP) é um conjunto de atividades que apoiam na execução e organização das tarefas de desenvolvimento de *software* (ARAUJO, 2017). A falta ou a aplicação insuficiente do processo de GP é um dos principais fatores que ocasionam problemas nos projetos conduzidos pelas organizações que operam na indústria de *software* (GONÇALVES, 2017). Portanto, a aplicação correta desse processo é um dos meios pelo qual as organizações conseguem sustentar a entrega do *software* no prazo e com custo previsto. Com isso, *softwares* que apoiam no processo de GP são muito úteis para as organizações, uma vez que permitem uma melhor condução e visibilidade do processo geral.

De acordo com PMI (2017) as fases de um projeto são:

- Iniciação - onde é identificada a viabilidade do projeto, seu orçamento e cronograma;
- Planejamento - onde é definido o escopo do projeto e as ações necessárias para alcançar os objetivos;
- Execução - onde é executado o trabalho definido no Plano de Gerenciamento do Projeto;
- Monitoramento e Controle - onde é realizada análise, acompanhamento e controle do progresso e desempenho do projeto;
- Encerramento - onde são realizados os processos para finalizar todas as atividades de todos os grupos de processos.

Através de um sistema de gerenciamento de artefatos de *software*, pode-se fornecer ao usuário ferramentas para gerenciar as diversas etapas do desenvolvimento de um *software*, sendo elas: Especificação de requisitos; Modelagem de análise; Monitoramento e controle; Implementação; Testes; Análise de riscos; Implantação. Dessa forma, o usuário pode ter um maior controle de cada etapa ao manter diferentes versões de seus artefatos.

A vantagem de usar uma aplicação *web* para a realização do gerenciamento dos artefatos do desenvolvimento de um *software* está na facilidade do versionamento dos artefatos através dos repositórios fornecidos para tal, além de melhor supervisão de cada etapa e do andamento total do projeto. Além disso, soma-se como vantagem da aplicação *web* o fato do *software* poder ser acessado via navegador independente da plataforma onde o usuário se encontra, sem a necessidade de instalação do *software* em cada dispositivo cliente, o que torna a tarefa de entrega de atualizações aos usuários bem mais prática e rápida (ESPINHA, 2022).

1.1 O problema e sua importância

Com a evolução da tecnologia e com a informatização de diversos negócios, sistemas cada vez mais complexos surgem e trazem com eles mais complexidade e desafios para o gerenciamento dos projetos. Com isso, os gerentes de projeto necessitam de ferramentas cada vez mais robustas para ajudá-los no processo de gerenciamento das diversas etapas do desenvolvimento de uma aplicação de *software* (KOSTALOVA; TETREVOVA; SVEDIK, 2015).

Pela análise dos *softwares* levantados, notou-se que estes sistemas dão pouco foco às fases que o gerente precisa passar para ter um projeto de *software*. Portanto, o sistema proposto visa proporcionar ao público alvo(gerentes de projeto, analistas de requisitos) uma experiência em que ele consiga controlar todos os artefatos de *software* através de versionamento e acompanhar todo o processo produtivo do ciclo de vida de um *software*, de modo que ele possa tomar melhores decisões com base em versões de seus artefatos e no andamento do projeto.

1.2 Objetivos

Nesta seção serão apresentados os objetivos da aplicação *web* a ser desenvolvida.

1.2.1 Objetivo geral

O objetivo deste trabalho é o desenvolvimento de uma aplicação *web* norteadora para o gerenciamento de artefatos de projeto de *software* dentro das diversas etapas do desenvolvimento.

1.2.2 Objetivos específicos

Além dos objetivos gerais e cumprir com a proposta do trabalho, pretende-se atingir os seguintes objetivos específicos:

- Permitir o gerenciamento de artefatos de *software* de forma menos complexa que das ferramentadas levantadas;
- Permitir uma visão mais ampla do processo de desenvolvimento de *software*;
- Propiciar informações detalhadas de cada etapa do desenvolvimento para tomadas de decisão;
- Proporcionar aos alunos de graduação um sistema para a organização e controle das versões dos artefatos de seus projetos de *software* desenvolvidos ao longo do curso de graduação e de suas disciplinas;
- Aplicar tecnologia mais aderente ao projeto em relação aos sistemas levantados, com conteúdo extracurricular que agrega uma pesquisa adicional a conteúdos de formação profissional do autor.

2 REVISÃO DA LITERATURA

Ao decorrer desta seção serão apresentados os princípios que fundamentam o desenvolvimento do sistema.

2.1 Softwares similares

Durante a revisão da literatura, alguns *softwares* semelhantes foram levantados, a fim de analisar as funcionalidades oferecidas e não oferecidas, além de extrair requisitos para o sistema proposto. A seguir estão alguns dos sistemas encontrados.

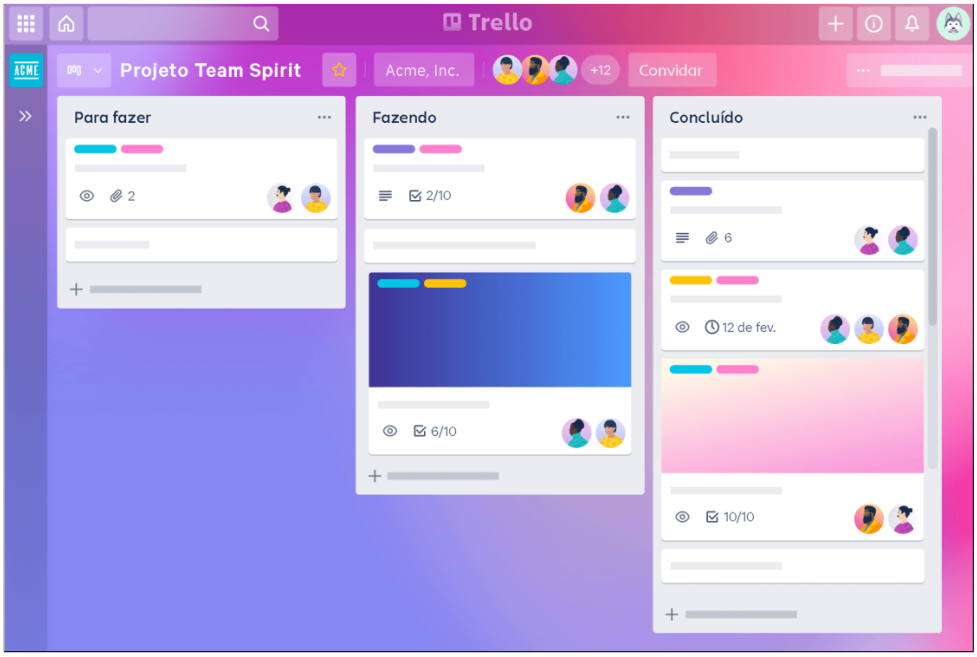
2.1.1 Trello

O Trello permite a criação de *boards*, que são estruturas que contém listas. As listas são organizadas de forma linear e são compostas por *cards*. Uma lista pode representar um conjunto de tarefas do projeto ou até mesmo uma fase do mesmo e dentro dela é possível adicionar *cards* de modo a organizar o projeto. Em uma lista de tarefas a serem feitas por exemplo, um *card* representaria cada uma dessas tarefas.

Em cada *card* é possível adicionar uma descrição, marcadores, anexos e data de entrega por exemplo. Além disso, o Trello permite a criação de equipes, de forma que o acesso a certas informações sejam restringidas ou liberadas para as pessoas. Com essa função de equipes e membros, é possível por exemplo que seja criada uma notificação em um *card* para que todos os membros ou observadores daquele *card* sejam avisados quando a data de entrega estiver se aproximando. Além disso, o Trello também permite a adição dos chamados *power-ups*, que permitem a integração da ferramenta com outros aplicativos, como Evernote, Jira, Slack, entre outros (TRELLO, 2022). Vale ressaltar que o Trello é uma ferramenta que não compreende todo o processo de desenvolvimento de um *software*, sendo ele uma ferramenta que fornece

um auxílio e um foco maior na parte de gerenciamento de estados e atribuições das tarefas. A figura 1 apresenta a ferramenta Trello.

Figura 1: Trello



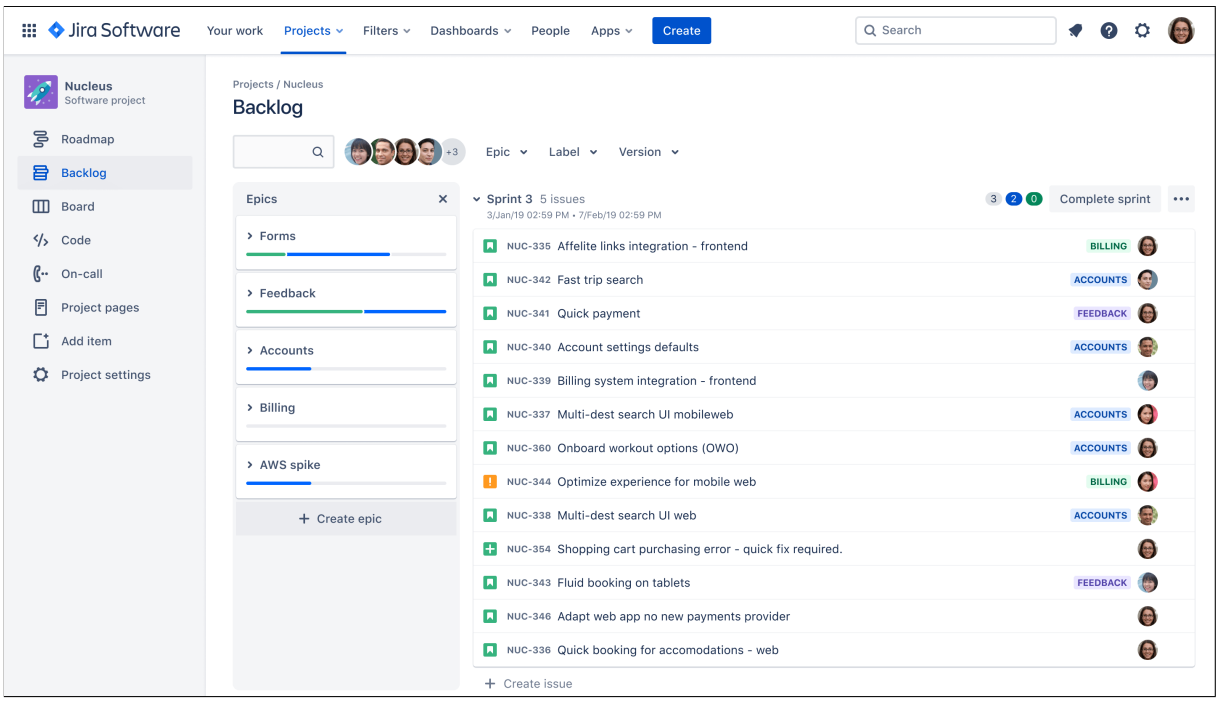
Fonte: Trello (2022)

2.1.2 Jira

O Jira permite o gerenciamento de projetos ágeis através de um conjunto de ferramentas. Em um primeiro momento, é possível organizar equipes seguindo a estrutura de equipes do Scrum. Portanto, é possível atribuir as responsabilidades de *Product Owner*, *Scrum Master* e organizar a equipe Scrum com os desenvolvedores que realizarão a codificação e desenvolvimento do sistema.

O *software* permite também a manutenção de listas de *tasks* e *bugs*. Dessa forma, é oferecida uma estrutura para a priorização desses itens de maneira a atingir os objetivos do projeto. Além disso, o Jira permite a conexão com outros aplicativos como Slack e Github, de forma a tornar um ambiente integrado (ATLASSIAN, 2022). A figura 2 apresenta a ferramenta Jira.

Figura 2: Jira



Fonte: Atlassian (2022)

Como ilustrado na figura 2, o software permite a estruturação do *Backlog* através da criação de *Epic*, *Story* e *Task*. Para cada um desses, é possível adicionar uma descrição, anexos, classificar a prioridade, atribuir a alguém da equipe e vincular com outros artefatos. Estes podem ser organizados através de *Sprints*, onde é possível definir a duração do ciclo, seu objetivo e observar os artefatos de acordo com seus status: a fazer, em progresso e concluído.

2.1.3 Especificidades da aplicação com softwares similares

As ferramentas citadas são utilizadas por diversas organizações para o gerenciamento de projetos, mas não são ferramentas que fornecem um apoio e orientação quanto aos passos a serem seguidos durante um processo de desenvolvimento de *software* e nem são tidas como repositórios de artefatos. Elas permitem que o usuário realize e gerencie as etapas de um projeto de *software* mas sem guiá-lo e orientá-lo por cada etapa desse projeto.

Diferentemente dos *softwares* citados até então, o sistema proposto neste trabalho fornece ao seu usuário uma orientação sobre quais passos devem ser seguidos durante o processo de desenvolvimento de *software* e se já está no momento desse passo ser

seguido.

Ao se cadastrar um projeto, este estará inicialmente na fase de "Especificação de Requisitos". Neste estado, o usuário poderá somente realizar cadastro de artefatos relacionados a essa fase: requisitos funcionais e não funcionais; modelo conceitual de dados. As fases seguintes a essa (Modelagem de análise; Monitoramento e controle; Implementação; Testes; Análise de riscos; Implantação) estarão bloqueadas, até que o usuário realize alguma atividade relacionada a fase imediatamente anterior a que ele deseja utilizar. Dessa forma, o usuário terá uma visão de progresso e orientação da fase a ser seguida.

2.2 Arquitetura do sistema

A arquitetura de *software* representa a estrutura de dados e os componentes necessários para construir um sistema computacional (PRESSMAN; MAXIM, 2021). Ela descreve a estrutura técnica, as limitações e características de cada componente, assim como as interfaces entre eles. A arquitetura é o arcabouço do sistema e por isso torna-se o plano de mais alto nível da construção de cada novo sistema (KRAFZIG; BANKE; SLAMA, 2004). Ela é importante pois afeta o desempenho e a robustez da aplicação, assim como a capacidade de distribuição e manutenibilidade da mesma (SOMMERVILLE, 2011).

Segundo o Instituto de Gestão e Tecnologia - IGTI (2019) atualmente existem vários padrões de arquitetura utilizados na criação da solução de *software*, sendo alguns deles os seguintes:

- Arquitetura em camadas: organiza o sistema em um conjunto de camadas, podendo estas serem desconstruídas em diversos serviços;
- Arquitetura cliente-servidor: organiza serviços combinando dados do cliente e do servidor. Neste caso, os clientes acessam os servidores para utilizar os serviços por eles disponibilizados;
- Arquitetura Model-View-Controller (MVC): organiza o sistema de acordo com três camadas (Modelo, Visão e Controle) de forma a separar as representações e interações internas dos dados da forma como eles são apresentados ao usuário;

- Arquitetura de microsserviços: organiza o sistema em múltiplos serviços e componentes, de modo a possibilitar a escalabilidade e independência dos módulos.

O sistema proposto nesse trabalho utilizará a arquitetura REST em seu desenvolvimento, que permite construir aplicações *web* distribuídas de forma rápida, uma vez que seu esforço de aprendizado é pequeno e permite o aproveitamento de infraestrutura *web* existente (FERREIRA; KNOP, 2017).

Nas subseções seguintes são apresentados conceitos relacionados a arquitetura do sistema que será desenvolvido.

2.2.1 API

Uma *Applications Protocol Interface* (API) é um conjunto de protocolos e definições que são usados no desenvolvimento e integração de *softwares*. Ela permite que uma solução ou serviço se comunique com outros serviços sem saber como estes foram implementados, o que simplifica o desenvolvimento das aplicações e gera economia de tempo e dinheiro (RedHat, 2017).

De modo geral, as APIs são vistas como contratos, que representam um acordo entre a parte que se beneficiará e usará esse contrato e a parte que o implementará de fato. Com isso, quem implementa e disponibiliza a API consegue autorizar acesso a seus recursos sem abrir mão da segurança e do controle dos mesmos.

No âmbito das aplicações *web*, as APIs são, em geral remotas, o que significa que os recursos manipulados pela API estão em algum lugar fora do computador que realiza a solicitação. As APIs *web* normalmente utilizam o protocolo *HyperText Transfer Protocol* (HTTP) para as solicitações e fornecem uma definição da estrutura de suas mensagens de resposta, que geralmente possuem o formato XML ou JSON, que apresentam os dados de forma simplificada de forma que facilita a manipulação pelas aplicações (RedHat, 2017).

2.2.2 REST

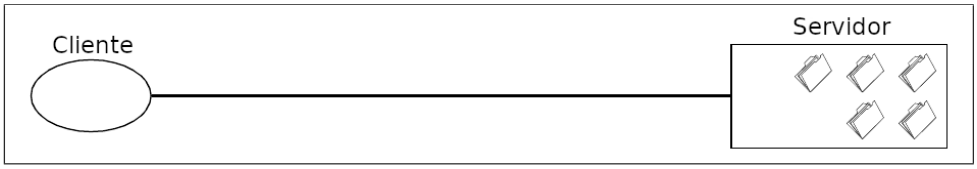
O REpresentational State Transfer (REST) é um estilo arquitetural que define as restrições fundamentais dos componentes, conectores e dados e que devem ser usadas na criação de serviços web (FIELDING, 2000).

Segundo Fielding (2000) os princípios orientadores e de restrição da arquitetura REST são:

1. Interface uniforme: com a aplicação do princípio da generalização da Engenharia de *Software*, é possível simplificar a arquitetura do sistema e melhorar a visibilidade de suas interações. Para obter uma interface REST uniforme, é necessário seguir quatro restrições: identificação de recursos; manipulação de recursos através de representações; mensagens auto descritivas; hypermídia como motor de estado da aplicação;

2. Cliente-servidor: esse *design pattern* impõe a separação de responsabilidades, o que permite os componentes do lado do cliente e servidor evoluírem independentemente. Ao separar as responsabilidades de interface(cliente) das de manipulação dos dados(servidor), há um ganho de portabilidade da interface entre plataformas e de escalabilidade, por simplificar os componentes do servidor. A figura 3 apresenta essa arquitetura.

Figura 3: Arquitetura cliente-servidor



Fonte: Adaptado de Fielding (2000)

3. Sem estado: cada requisição do cliente ao servidor deve conter toda a informação necessária para que este entenda e complete a requisição, sem ter como vantagem dados de contexto previamente armazenados;

4. Armazenamento em cache: toda resposta deve implícita ou explicitamente ser marcada como possível de ser armazenada em cache ou não, de forma que a aplicação cliente possa reusar a informação posteriormente dentro de um período determinado de tempo;

5. Sistema em camadas: permite que a arquitetura seja composta por camadas hierárquicas que restringem o comportamento dos componentes, de tal modo que cada componente não possa enxergar além da camada imediata com a qual ele está interagindo;

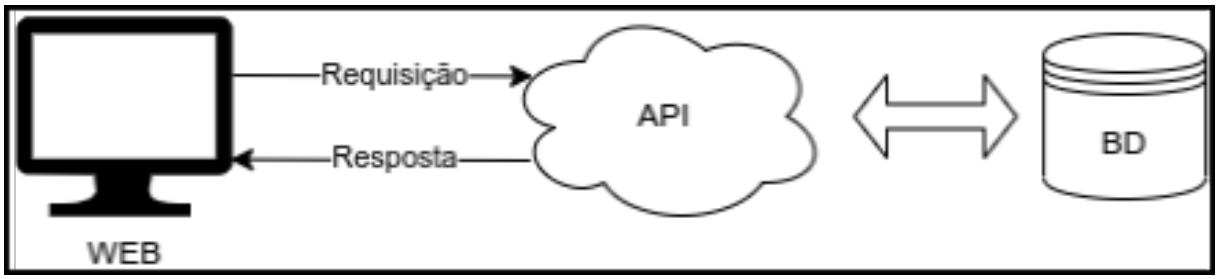
Como chave da abstração da informação no REST temos os chamados recursos, que são qualquer informação que possa ser nomeada, como um documento, imagem ou coleção de recursos. Para acessar tais recursos, são usados os métodos de recurso, que servem para performar operações sobre os recursos. Para tal, os verbos HTTP são comumente utilizados, de forma a definir a finalidade da requisição. Os verbos mais comuns são:

- *GET*: método para obter a informação identificada pela *Uniform Resource Identifier* (URI) da requisição;
- *POST*: método utilizado para requisitar ao servidor que inclua a informação enviada como novo recurso;
- *PUT*: método usado para que o servidor atualize o recurso identificado pela URI com as informações que estão sendo enviadas pela requisição;
- *DELETE*: método usado para deletar o recurso identificado pela URI.

2.2.3 Projeto da arquitetura

A arquitetura do sistema seguirá o conceito em que o *backend* e *frontend* são separados. No caso do sistema proposto, o *backend* será a API REST desenvolvida com Spring e fará a comunicação com o banco de dados. Já o *frontend* será uma aplicação *web* desenvolvida com o *framework* Angular.

Figura 4: Arquitetura do sistema



Fonte: Do autor

Como ilustrado na Figura 4, o *backend* é representado pela API REST e fará a comunicação com a base de dados. Já a aplicação *web* se comunica com a API através de requisições HTTP, que quando recebidas serão tratadas junto ao banco de dados e ao final uma resposta será emitida para a aplicação *web*.

2.3 Tecnologias utilizadas

O sistema proposto será dividido em duas camadas: uma conhecida como *frontend*, que será responsável pela interface do usuário e pelas requisições aos dados; uma segunda conhecida como *backend*, que será responsável por atender às requisições do usuário, manipular e validar as regras de negócio e se comunicar com o banco de dados.

A seguir são apresentadas as tecnologias que serão utilizadas para o desenvolvimento das camadas citadas.

2.3.1 Angular

O Angular é uma plataforma de desenvolvimento, construída com Typescript e liderada por uma equipe do Google. Essa plataforma inclui um *framework* baseado em componentes(blocos menores com *template* HTML, estilização própria e códigos em Typescript) que são usados para construir as páginas. Além disso, o Angular possui uma vasta quantidade de bibliotecas integradas para permitir a navegação entre páginas, o gerenciamento de formulário e a comunicação cliente-servidor (Angular, 2021).

2.3.2 TypeScript

O TypeScript é um superset da linguagem de programação JavaScript que traz a tipagem estática para o JavaScript. Com isso, é possível descobrir erros durante a implementação do código, além de facilitar a manutenção e entendimento do mesmo. Antes de ser executado, o código TypeScript é convertido para JavaScript, permitindo que ele possa seja executado em qualquer lugar onde o JavaScript rode (TypeScript, 2022).

2.3.3 JSON

O JSON é uma sintaxe que se baseia na sintaxe JavaScript e que é usada para armazenar, transportar e serializar objetos. O JSON é comumente utilizado quando dados são enviados de um servidor para uma página *web*, onde um programa JavaScript pode facilmente converter dados do JSON para objetos JavaScript nativos

(W3Schools, 2022).

2.3.4 Spring Framework

O Spring é um ecossistema de desenvolvimento que torna a criação de aplicações Java mais fáceis, ao fazer uso de módulos independentes. Dentre seus diversos módulos, temos os seguintes que serão utilizados no desenvolvimento do sistema proposto:

- Spring Boot: É um *framework* que nasceu a partir do Spring Framework e veio para facilitar as configurações inicias de um projeto. Com esse *framework*, os desenvolvedores podem criar aplicações com mais agilidade, se preocupando menos com as configurações do Spring. Por ser uma extensão do Spring Framework, ele tem tudo que existe no outro, de forma embutida. Além disso, ele traz consigo um servidor embarcado e com as configurações iniciais prontas;
- Spring Data: É um módulo do Spring que facilita o acesso aos dados por parte da aplicação. Dentro deste módulo há o Spring Data JPA, que torna fácil a implementação de repositórios baseados em JPA, uma vez que o desenvolvedor escreve suas interfaces do repositório e o Spring fornece automaticamente a implementação e métodos de busca customizados, sem a necessidade de escrever consultas para cada banco de dados.
- Spring Security: É um *framework* de autenticação e controle de acesso altamente customizável que foca em prover autenticação, autorização e armazenamento de senhas para aplicações Java. Além disso, ele fornece proteção contra diversos ataques.

2.3.5 Java

O Java é uma linguagem de programação relacionada ao C++ que foi lançada pela primeira vez pela Sun Microsystems em 1995. A motivação original para a criação da linguagem era a necessidade de se ter uma linguagem que fosse independente da plataforma(isto é, neutra em relação à arquitetura) (SCHILDT, 2019).

Na mesma época em que o Java estava sendo criado, surgia um fator que teve um papel crucial no futuro do Java: a *World Wide Web*. Ela impulsionou o Java para a

vanguarda do *design* de linguagens de programação, pois a *Web* também demandava programas portáteis (SCHILDT, 2019).

2.3.6 PostgreSQL

O PostgreSQL é um banco de dados objeto-relacional poderoso e *open source* que usa e estende a linguagem SQL combinada com várias outras funcionalidades para armazenar de forma segura e escalar as mais complicadas cargas de trabalho (PostgreSQL, 2022).

De acordo com PostgreSQL (2022), o PostgreSQL possui diversas funcionalidades para ajudar os desenvolvedores a construir suas aplicações e aos administradores de banco de dados a proteger a integridade dos dados e criar um ambiente tolerante a falhas. Além disso, por ser livre e *open source*, o PostgreSQL é altamente extensível e permite o usuário definir seus próprios tipos de dados e funções customizadas por exemplo.

2.3.7 Uso das tecnologias na arquitetura proposta

As interações com as tecnologias envolvidas começam do lado do cliente, onde este acessa a aplicação. Em um cenário comum de uso, como por exemplo a listagem de requisitos funcionais, o fluxo ocorre da maneira descrita abaixo.

1. O usuário clica na opção de listagem de requisitos funcionais, o que faz com que o Angular carregue o componente relacionado a essa listagem;
2. O componente envia uma requisição, com o *body* no formato JSON contendo o identificador do projeto e um texto identificando o tipo do requisito;
3. A aplicação no *backend* desenvolvida com Spring e Java recebe a requisição e consulta o banco de dados PostgreSQL para obter todos os requisitos funcionais cadastrados para aquele projeto;
4. O *backend* retorna no *body* da resposta um JSON contendo todos os requisitos funcionais encontrados;
5. O componente de listagem de requisitos recebe a resposta e carrega os requisitos funcionais em tela.

Vale ressaltar que a descrição do fluxo não foi realizada por meio de um diagrama de sequência devido a clareza e simplicidade das interações, presentes nas especificações de casos de uso feitas.

3 METODOLOGIA

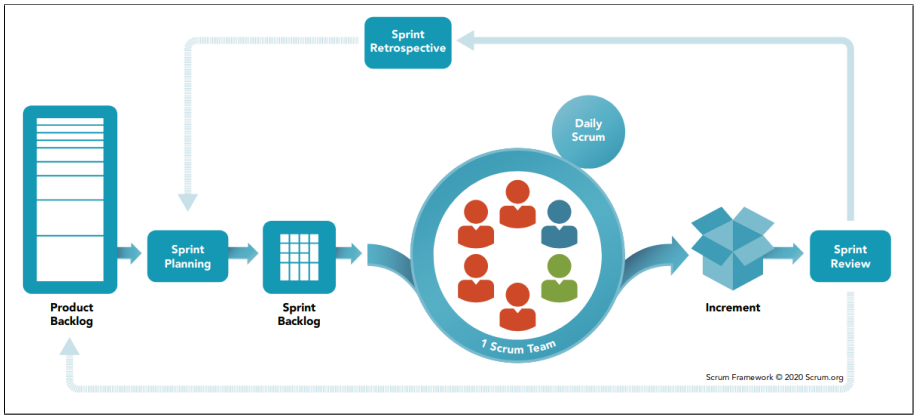
A pesquisa realizada no trabalho foi de caráter exploratório e qualitativo, onde partindo do levantamento e análise dos requisitos, foi utilizada a metodologia de desenvolvimento descrita nas subseções seguintes, visando a implementação do sistema.

3.1 Metodologia de desenvolvimento

A abordagem de desenvolvimento utilizada no projeto é a metodologia de desenvolvimento ágil conhecida como Scrum. Os processos de desenvolvimento ágeis são concebidos para produzir, rapidamente, *softwares* úteis, onde eles não são desenvolvidos como uma única unidade, mas como uma série de incrementos, sendo cada incremento uma nova funcionalidade do sistema. Com isso, os processos de especificação, projeto e implementação são intercalados (SOMMERVILLE, 2011).

A abordagem Scrum é um método ágil cujo foco está no gerenciamento do desenvolvimento iterativo, ao invés das abordagens técnicas específicas da engenharia de *software* ágil. Esta abordagem também não determina o uso de práticas de programação, como programação em pares e desenvolvimento *test-first*.

Figura 5: Processo Scrum



Fonte: Scrum (2020)

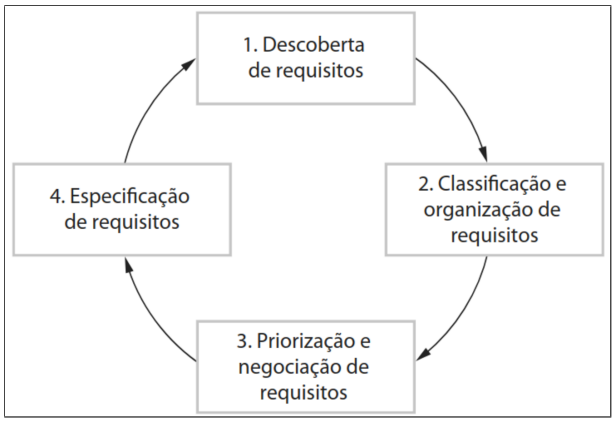
Como apresentado na figura 5, o Scrum é composto por uma série de eventos, entre eles: *Sprint*, *Sprint Planning*, *Daily Scrum* e *Sprint Review*. A *sprint* é um *container* para todos os outros eventos, onde as ideias são transformadas em artefatos. Já a unidade de planejamento onde o trabalho a ser feito é avaliado é chamada de *Sprint Planning*, que dá origem ao *Sprint Backlog* que contém os itens do *Product Backlog* a serem desenvolvidos na *sprint*. Com isso, os recursos para seu desenvolvimento são selecionados e então o *software* é implementado. Durante o período da *sprint*, são realizadas reuniões diárias rápidas, chamadas *Daily Scrum*, para o acompanhamento do *status* da equipe. Ao final de cada *sprint* é realizada a chamada *Sprint Review*, onde são revisados os itens do *Sprint Backlog* e entregues aqueles que foram finalizados.Com isso, o produto é decomposto em partes menores que são mais facilmente gerenciáveis e compreensíveis (SOMMERVILLE, 2011).

3.2 Elicitação e análise de requisitos

Com o estudo inicial de viabilidade do projeto concluído, o próximo estágio do processo de engenharia de requisitos é a elicitação e análise de requisitos. Segundo Pressman e Maxim (2021), a elicitação de requisitos combina elementos de solução de problemas, negociação, elaboração e especificação. Uma parte importante desse levantamento é entender as metas de negócio, isso é, os objetivos de longo prazo que o sistema deve alcançar.

A figura 6 apresenta um modelo de elicitação e análise de requisitos.

Figura 6: O processo de elicitação e análise de requisitos



Fonte: Sommerville (2011)

1. Descoberta de requisitos: é a atividade na qual os requisitos de domínio e da

documentação são descobertos;

2. Classificação e organização de requisitos: essa atividade toma como base uma coleção de requisitos não estruturados, agrupa os relacionados e os organiza em grupos coerentes;

3. Priorização e negociação de requisitos: nessa atividade ocorre a priorização de requisitos e a resolução de conflitos por meio da negociação de requisitos;

4. Especificação de requisitos: nessa atividade os requisitos são documentados e inseridos no próximo ciclo.

A figura 6 destaca que a elicitação e análise de requisitos é um ciclo contínuo, onde cada ciclo começa com a descoberta dos requisitos e termina com sua documentação (SOMMERVILLE, 2011). Dessa forma, os requisitos do sistema foram levantados através de diversas conversas e reuniões com o orientador, o qual idealizou a proposta baseada na experiência pregressa e necessidade didática (além da técnica) para o uso desta ferramenta no ensino do fluxo de projeto de *software* a alunos de computação em geral.

3.2.1 Descrição dos usuários

O sistema contará com alguns usuários do sistema, que sob a perspectiva da modelagem são entendidos como atores. Esses atores representam algum papel que o usuário exercerá no sistema ao desempenhar os casos de uso. Dessa maneira, as permissões e acesso as funcionalidades foram definidas de acordo com os casos de uso exercidos pelos atores.

Dentro do sistema, haverá um ator Analista de Requisitos que exercerá o papel de analista de requisitos do projeto. Além dele, haverá um ator para o Gerente de Projetos, que exercerá o papel de gerente de projeto do projeto em questão.

Essa relação pode ser observada na tabela 1:

3.2.2 Requisitos funcionais

Os requisitos funcionais são declarações dos serviços que o sistema deve fornecer e como ele deve reagir a entradas específicas. Eles dependem do tipo de *software* que

Tabela 1: Descrição dos usuários

Tipo do usuário	Descrição
Analista de requisitos	Representa o analista de requisitos
Gerente de projetos	Representa o gerente de projetos

Fonte: Do autor

vai ser desenvolvido, de seus usuários e da abordagem da organização ao escrever os requisitos (SOMMERVILLE, 2011). A seguir estão descritos os requisitos funcionais do sistema.

- RF01 - Manter projetos - O gerente de projetos poderá incluir e alterar projetos.
- RF02 - Consultar projetos - O ator poderá consultar projetos.
- RF03 - Manter requisito funcional e não funcional - O ator poderá incluir, alterar, excluir e consultar os requisitos funcionais(casos de uso) e não funcionais.
- RF04 - Manter modelo conceitual de dados - O ator poderá realizar *upload* de arquivos de prévia e base do modelo e posteriormente visualizá-lo, atualizá-lo, excluí-lo ou alterar sua versão. O ator poderá também realizar o *download* dos arquivos fornecidos em cada versão.
- RF05 - Manter casos de uso - O ator poderá realizar *upload* de arquivos de prévia e base do caso de uso e posteriormente visualizá-lo, atualizá-lo, excluí-lo ou alterar sua versão. O ator poderá também realizar o *download* dos arquivos fornecidos em cada versão.
- RF06 - Manter diagrama de classes - O ator poderá realizar *upload* de arquivos de prévia e base do modelo e posteriormente visualizá-lo, atualizá-lo, excluí-lo ou alterar sua versão. O ator poderá também realizar o *download* dos arquivos fornecidos em cada versão.
- RF07 - Manter diagrama de sequência - O ator poderá realizar *upload* de arquivos de prévia e base do modelo e posteriormente visualizá-lo, atualizá-lo, excluí-lo ou alterar sua versão. O ator poderá também realizar o *download* dos arquivos fornecidos em cada versão.
- RF08 - Manter modelo lógico de dados - O ator poderá realizar *upload* de arquivos de prévia e base do modelo e posteriormente visualizá-lo, atualizá-lo,

excluí-lo ou alterar sua versão. O ator poderá também realizar o *download* dos arquivos fornecidos em cada versão.

- RF09 - Realizar cálculo de estimativas - O ator poderá realizar cálculos de estimativas por Ponto de Função e/ou Pontos por Casos de Uso.
- RF10 - Manter gráfico de Gantt - O ator poderá realizar *upload* de arquivos de prévia e base do gráfico e posteriormente visualizá-lo, atualizá-lo, excluí-lo ou alterar sua versão. O ator poderá também realizar o *download* dos arquivos fornecidos em cada versão.
- RF11 - Manter protótipos - O ator poderá realizar *upload* de arquivos de prévia e base dos protótipos das interfaces e posteriormente visualizá-lo, atualizá-lo, excluí-lo ou alterar sua versão. O ator poderá também realizar o *download* dos arquivos fornecidos em cada versão.
- RF12 - Manter código do projeto de software - O ator poderá realizar *upload* do código do projeto, podendo ser a pasta compactada ou o arquivo de geração do projeto. Posteriormente ele poderá visualizá-lo, atualizá-lo, excluí-lo ou alterar sua versão. O ator poderá também realizar o *download* do arquivo fornecido em cada versão.
- RF13 - Manter modelo físico de dados - O ator poderá realizar *upload* de arquivos de prévia e base do modelo físico de dados e posteriormente visualizá-lo, atualizá-lo, excluí-lo ou alterar sua versão. O ator poderá também realizar o *download* dos arquivos fornecidos em cada versão.
- RF14 - Manter descrição de tabelas de banco de dados - O ator poderá incluir, alterar, excluir e consultar cada entidade do banco de dados, incluindo seus atributos e descrição dos mesmos.
- RF15 - Gerar relatório das descrições de tabelas de banco de dados - O ator poderá gerar relatório contendo as informações das descrições de tabelas de banco de dados.
- RF16 - Manter modelo de arquitetura - O ator poderá realizar *upload* de arquivos de prévia e base do modelo de arquitetura e posteriormente visualizá-lo, atualizá-lo, excluí-lo ou alterar sua versão. O ator poderá também realizar o *download* dos arquivos fornecidos em cada versão.

- RF17 - Manter testes unitários - O ator poderá incluir, alterar, excluir e consultar as unidades testadas, que são compostas por: caso de teste; objetivos; procedimentos adotados; ocorrência de teste; data do teste e solução adotada para a ocorrência.
- RF18 - Manter testes de cobertura - O ator poderá incluir, alterar, excluir e consultar as unidades testadas, que são compostas por: caso de teste; objetivos; procedimentos adotados; ocorrência de teste; data do teste e solução adotada para a ocorrência.
- RF19 - Manter testes de integração - O ator poderá incluir, alterar, excluir e consultar as unidades testadas, que são compostas por: caso de teste; objetivos; procedimentos adotados; ocorrência de teste; data do teste e solução adotada para a ocorrência.
- RF20 - Manter testes de regressão - O ator poderá incluir, alterar, excluir e consultar as unidades testadas, que são compostas por: caso de teste; objetivos; procedimentos adotados; ocorrência de teste; data do teste e solução adotada para a ocorrência.
- RF21 - Manter testes de carga - O ator poderá incluir, alterar, excluir e consultar as unidades testadas, que são compostas por: caso de teste; objetivos; procedimentos adotados; ocorrência de teste; data do teste e solução adotada para a ocorrência.
- RF22 - Manter testes funcionais - O ator poderá incluir, alterar, excluir e consultar as unidades testadas, que são compostas por: caso de teste; objetivos; procedimentos adotados; ocorrência de teste; data do teste e solução adotada para a ocorrência.
- RF23 - Gerar relatório de incidentes de testes funcionais - O ator poderá gerar relatório contendo as informações dos testes funcionais cadastrados.
- RF24 - Manter análise de riscos - O ator poderá realizar *upload* do arquivo da análise de risco, podendo ser um documento de texto por exemplo. Posteriormente ele poderá visualizá-lo, atualizá-lo, excluí-lo ou alterar sua versão. O ator poderá também realizar o *download* do arquivo fornecido em cada versão.
- RF25 - Manter telas do sistema - O ator poderá realizar *upload* de arquivos de prévia e base de cada tela do sistema e posteriormente visualizá-la, atualizá-la,

excluí-la ou alterar sua versão. O ator poderá também realizar o *download* dos arquivos fornecidos em cada versão.

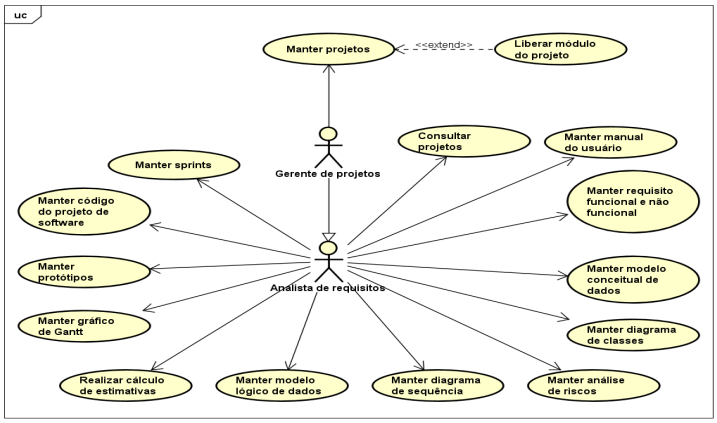
- RF26 - Manter manual do usuário - O ator poderá realizar *upload* de arquivos de prévia e base do manual do usuário e posteriormente visualizá-la, atualizá-la, excluí-la ou alterar sua versão. O ator poderá também realizar o *download* dos arquivos fornecidos em cada versão.
- RF27 - Manter help do sistema - O ator poderá realizar *upload* de arquivos de prévia e base do help do sistema e posteriormente visualizá-la, atualizá-la, excluí-la ou alterar sua versão. O ator poderá também realizar o *download* dos arquivos fornecidos em cada versão.
- RF28 - Manter sprints - O ator poderá incluir, alterar, excluir e consultar sprints, que são compostas por: data de início; data de término; versão e os itens do backlog, que são os artefatos cadastrados no projeto.
- RF29 - Liberar módulo do projeto - O gerente de projetos poderá liberar etapas que estão bloqueadas, mesmo que a etapa anterior a ela ainda não tenha sido totalmente realizada, ou seja, ainda não possui registro para todos os seus itens.

3.2.3 Casos de uso

De acordo com Pressman e Maxim (2021), um caso de uso representa o *software* do ponto de vista do usuário, e apresenta uma jornada sobre como esse usuário(desempenhando algum dos papéis) irá interagir com o sistema sob um conjunto de condições específicas.

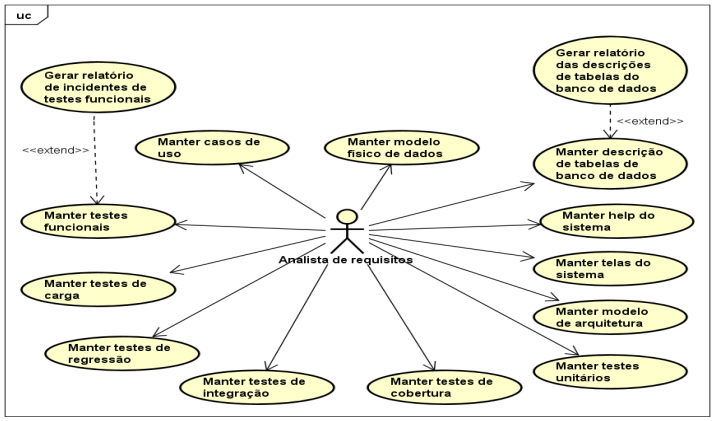
Devido a quantidade de casos de uso, o diagrama foi dividido em dois. As figuras 7 e 8 exibem os casos de uso do sistema proposto.

Figura 7: Diagrama de Caso de Uso 1



Fonte: Do autor

Figura 8: Diagrama de Caso de Uso 2



Fonte: Do autor

Conforme as figuras 7 e 8 ilustram, o ator Gerente de projetos pode executar qualquer caso de uso. Além disso, pode-se observar também que o ator Analista de requisitos não pode executar os casos de uso "Manter projetos" e "Liberar módulos" sendo esses exclusivos do Gerente de projetos.

Devido a quantidade de casos de uso, abaixo estão descritos dois cenários: "Manter projetos" e "Manter sprints", pois o primeiro é o cenário onde surge um projeto e o segundo onde os artefatos cadastrados nas diversas fases do projeto são agrupados em entregas. No Apêndice A são apresentadas as descrições detalhadas dos demais casos de uso e na lista de siglas o significado das abreviações. Vale ressaltar que não foi utilizado um diagrama de atividades devido a clareza e simplicidade das interações, presentes nas especificações de casos de uso feitas.

UC1 - Manter projetos

Ator(es): Gerente de projetos

Pré-condição:

FA1 - Não há.

FA2 - Pelo menos um projeto de software cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os projetos de software cadastrados. [FA1] [FA2]

Fluxo alternativo:

FA1 - Cadastrar projeto de software

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a modal de novo projeto de software.
3. [EV] O ator informa o nome do projeto.
4. [EV] O ator informa a descrição do projeto.
5. [EV] O ator confirma o cadastro.
6. [RS] O sistema exibe a mensagem [MSG001].
7. [RS] O sistema atualiza a listagem de projetos de software.

FA2 - Editar projeto de software

1. [EV] O ator seleciona um projeto de software na listagem.
2. [RS] O sistema exibe as informações detalhadas do projeto de software.
3. [EV] O ator altera o nome do projeto de software.
4. [EV] O ator altera a descrição do projeto de software.
5. [EV] O ator confirma a edição.
6. [RS] O sistema exibe a mensagem [MSG002].

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo projeto de software é inserido na listagem de projetos.

FA2 - Um projeto de software terá seu registro alterado na listagem de projetos.

Mensagens:

MSG001 - Projeto criado com sucesso!

MSG002 - Projeto atualizado com sucesso!

UC28 - Manter sprint

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Método ágil" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos uma sprint cadastrada.

Fluxo principal:

1. [RS] O sistema exibe a tela com as sprints cadastradas. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar sprint

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [RS] O sistema carrega as fases do projeto para o componente de seleção.
4. [EV] O ator informa a data de início da sprint.
5. [EV] O ator informa a data de término da sprint.
6. [EV] O ator informa a versão da sprint.
7. [EV] O ator informa a fase do projeto.

8. [EV] O ator informa o artefato do projeto. [FA5]
9. [EV] O ator adiciona de um a vários artefatos no backlog da sprint. [FE1]
10. [EV] O ator confirma o cadastro.
11. [RS] O sistema exibe a mensagem [MSG001].
12. [RS] O sistema atualiza a listagem de sprints.

FA2 - Editar sprint

1. [EV] O ator seleciona uma sprint na listagem.
2. [RS] O sistema exibe as informações detalhadas da sprint.
3. [RS] O sistema carrega as fases do projeto para o componente de seleção.
4. [EV] O ator altera a data de início da sprint.
5. [EV] O ator altera a data de término da sprint.
6. [EV] O ator altera a versão da sprint.
7. [EV] O ator altera o backlog da sprint. [FE2]
8. [EV] O ator confirma a edição.
9. [RS] O sistema exibe a mensagem [MSG002].

FA3 - Excluir sprint

1. [EV] O ator seleciona uma sprint na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de sprints.

FA4 - Buscar sprints

1. [EV] O ator pode informar qualquer um dos seguintes filtros: versão da sprint; número da entrega da sprint; intervalo de data de início da sprint.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de sprints com os resultados encontrados.

FA5 - Informar versão do artefato

1. [EV] O ator informa a versão do artefato.
2. Retorna ao passo 8 do FA1.

Fluxo de exceção:

FE1 - O ator adiciona um item já adicionado ao backlog.

1. [RS] O sistema exibe a mensagem [MSG004].
2. Retorna ao passo 9 do FA1.

FE2 - O ator adiciona um item já adicionado ao backlog.

1. [RS] O sistema exibe a mensagem [MSG004].
2. Retorna ao passo 7 do FA2.

Pós-condição:

FA1 - Uma nova sprint é inserida na listagem de sprints.

FA2 - Uma sprint terá seu registro alterado na listagem de sprints.

FA3 - Uma sprint terá seu registro removido na listagem de sprints.

FA4 - As sprints encontradas são exibidas na listagem de sprints.

Mensagens:

MSG001 - Sprint cadastrada com sucesso

MSG002 - Sprint atualizada com sucesso

MSG003 - Sprint deletada com sucesso

MSG004 - Esse artefato já está no backlog

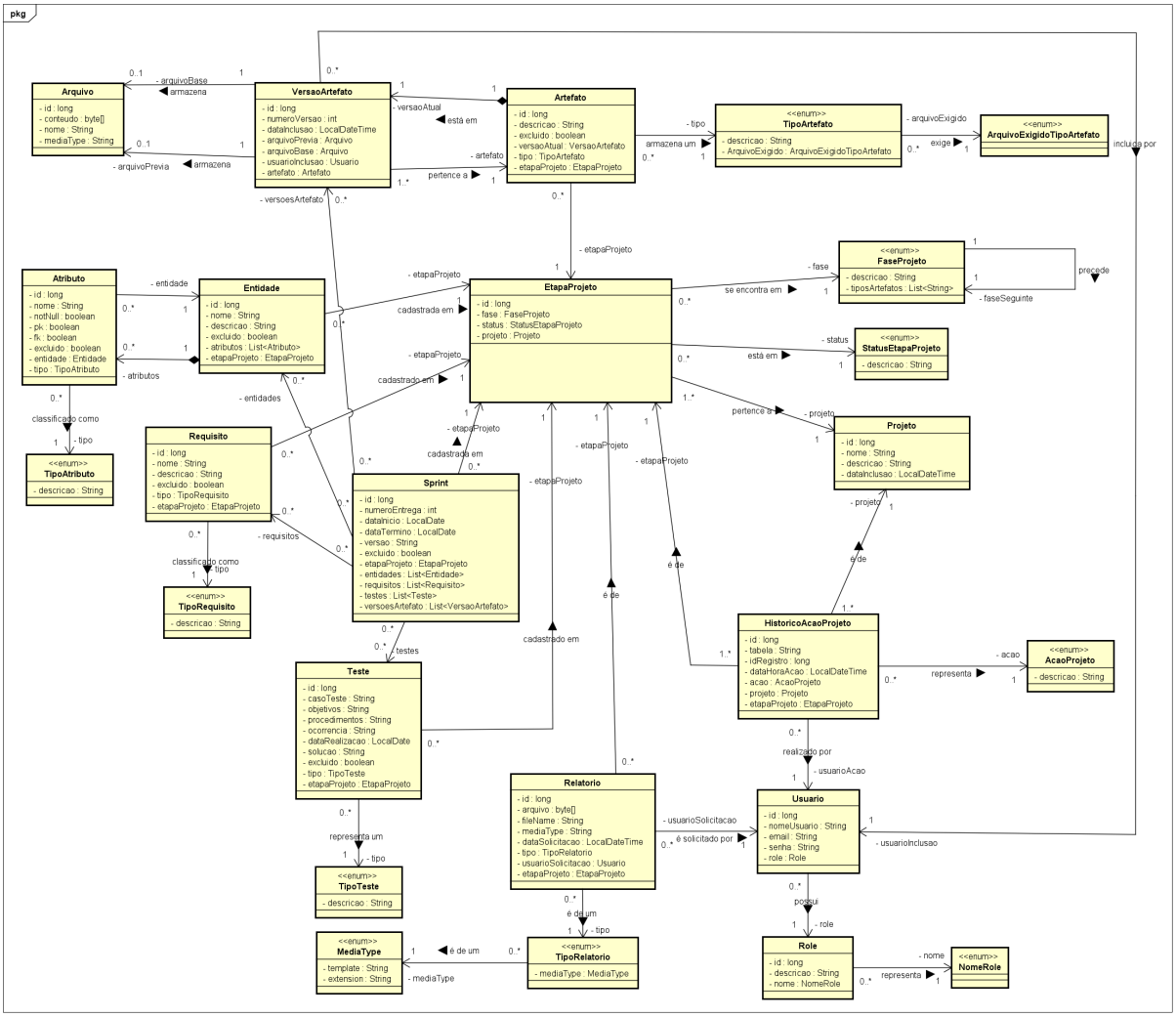
3.2.4 Modelo de domínio

Um diagrama de classes descreve a estrutura estática do sistema, em um grau mais elevado de abstração do que o alcançado pelo código-fonte, mas mostrando classes, interfaces e relacionamentos (ALMEIDA, 2007).

Segundo Sommerville (2011), os diagramas de classe são utilizados no desenvolvimento de um modelo de sistema orientado a objetos com o propósito de exibir as classes e suas associações. Dessa forma, uma classe de objeto pode ser imaginada como um tipo de objeto do sistema e uma associação pode ser vista como um link entre classes que possuam algum relacionamento.

A figura 9 exibe o modelo de domínio do sistema proposto.

Figura 9: Modelo de domínio



Fonte: Do autor

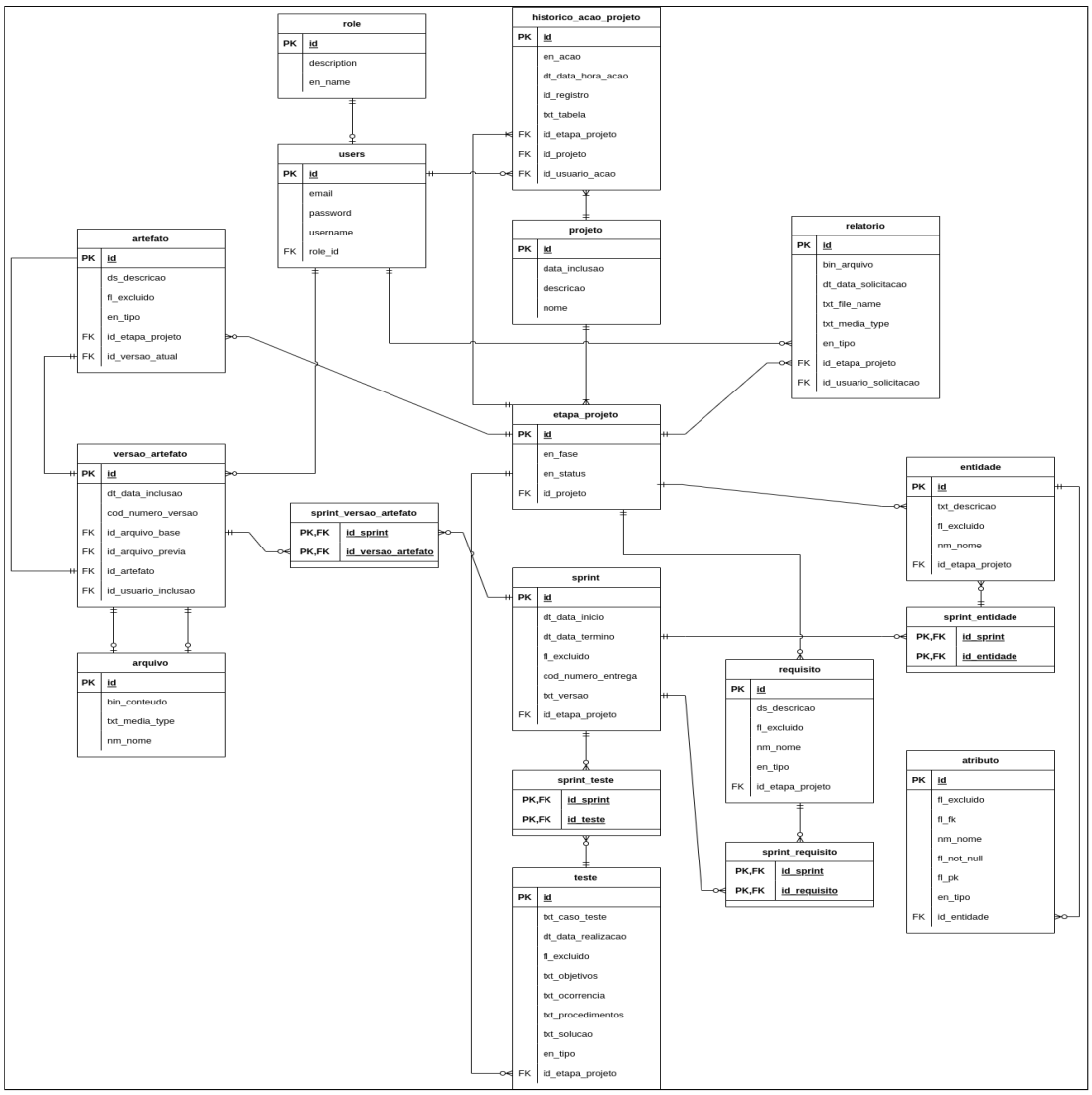
Na figura 9 é importante observar que a classe chamada Artefato representa todos os artefatos de projeto que possuem as seguintes características: representam artefatos gerados de forma externa ao sistema, são versionados e armazenados como figuras ou arquivos complexos. Dessa forma, os artefatos são diferenciados pelo atributo tipo, que identifica qual tipo de artefato está sendo armazenado em determinado objeto. Além disso, é importante destacar que as classes que representam os enumeradores não possuem seus valores no diagrama, devido a quantidade de valores representados.

3.3 Modelagem de banco de dados

O projeto de banco dados descreve a realidade de forma abstrata, de forma que as informações podem ser representadas fisicamente em uma base de dados. Para a criação da modelagem conceitual de um projeto, é comum a utilização do modelo entidade-relacionamento. Esse modelo tem como base a utilização de um conjunto de objetos chamados "entidades" e relacionamentos entre esses objetos para descrever algo do mundo real. Uma entidade é algo do mundo real, podendo representar algo físico ou conceitual. Além disso, as entidades podem ser classificadas como fracas(sua existência está condicionada à presença de alguma outra entidade) ou forte(caso não dependa de outra entidade para existir) (DANTAS; CÓRDULA; ARAÚJO, 2016).

A figura 10 mostra o modelo entidade-relacionamento do sistema proposto:

Figura 10: Modelo entidade-relacionamento



Fonte: Do autor

De forma similar à figura 9, é importante notar na figura 10 que os artefatos que possuem características idênticas, são armazenados no banco de dados em uma mesma tabela chamada artefato, cuja versão é controlada por relacionamentos com a tabela versao_artefato e cujo tipo é controlado por um atributo chamado "en_tipo", que representa um valor do enumerador de tipos de artefato.

3.3.1 Descrição de dados

Como o modelo entidade-relacionamento não aborda o tipo dos atributos, foi necessária uma descrição mais detalhada destes. Os atributos serão descritos em tabelas que representam a entidade da qual eles fazem parte, onde a primeira coluna é o nome do

atributo, a segunda é o tipo que ele armazena, a terceira indica se ele é chave primária ou estrangeira e a última fornece uma breve descrição dele.

Um exemplo da descrição dos dados se encontra na tabela 2 logo a seguir.

Tabela 2: Descrição de dados da entidade Projeto

projeto			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	identificador
nome	texto		Nome do Projeto
data_inclusao	data		Data de inclusão do Projeto

Fonte: Do autor

As descrições para as demais entidades se encontram no Apêndice B.

3.4 Garantia de qualidade

A qualidade de um projeto compreende o nível de atendimento às funções e particularidades especificadas no modelo de requisitos. Para a garantia da qualidade do *software* desenvolvido é importante a realização de testes, também conhecidos como verificação e validação, que asseguram que o *software* implementa corretamente determinada função e que este pode ser rastreado segundo os requisitos do cliente (PRESSMAN; MAXIM, 2021).

Tendo em vista a garantia da qualidade, foram realizados alguns tipos de teste no sistema, sendo eles:

- Teste de desenvolvimento: realizado pelo programador durante o desenvolvimento de determinada *feature*;
- Teste de usabilidade: realizado com o intuito de melhorar a experiência do usuário;
- Teste unitário: realizado com o intuito de testar cada unidade do *software*, componente ou classe por exemplo.

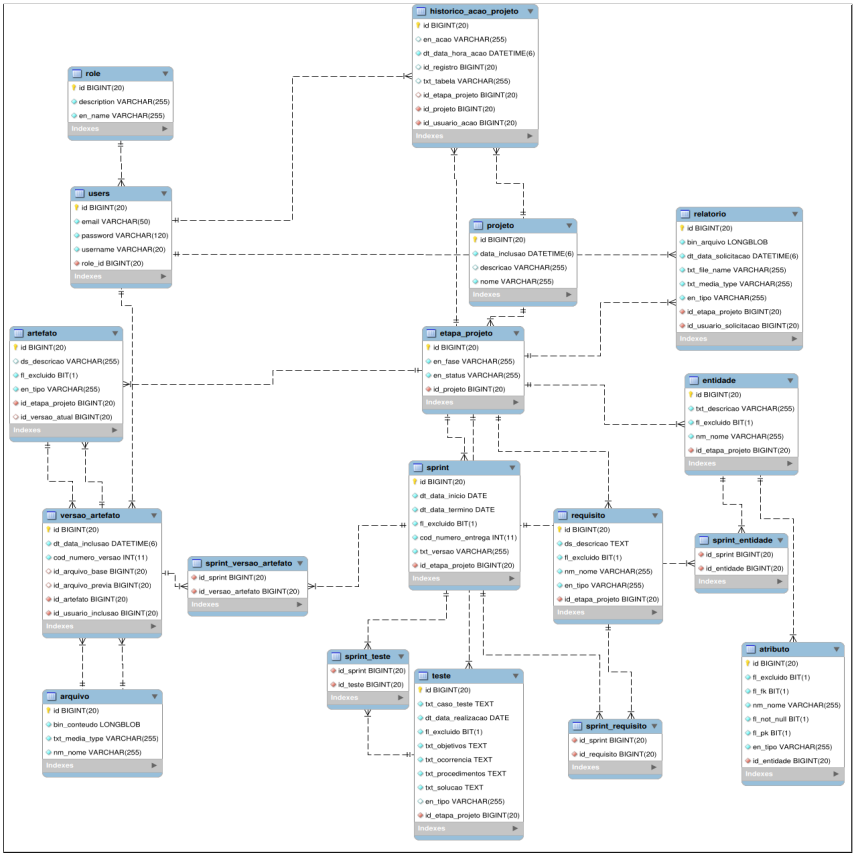
4 DESENVOLVIMENTO DA APLICAÇÃO

O desenvolvimento do sistema foi realizado a partir das modelagens propostas e dos estudos sobre os *frameworks* Spring e Angular, dentre outros estudos.

4.1 Modelo lógico

O modelo lógico do banco de dados, exibido na figura 11, foi elaborado a partir do levantamento dos requisitos e das funcionalidades do sistema.

Figura 11: Modelo lógico



Fonte: Do autor

É importante ressaltar que as tabelas "artefato" e "versao_artefato" são tabelas genéricas utilizadas por todos os tipos de artefatos que possuem versionamento dos arquivos enviados na plataforma.

Ainda sobre o modelo lógico representado na figura 11, este foi desenhado utilizando a ferramenta MySQL Workbench, que é uma ferramenta gráfica para trabalhar com bancos de dados MySQL e que permite a criação de modelos a partir de *schemas* do banco de dados (ORACLE, 2022).

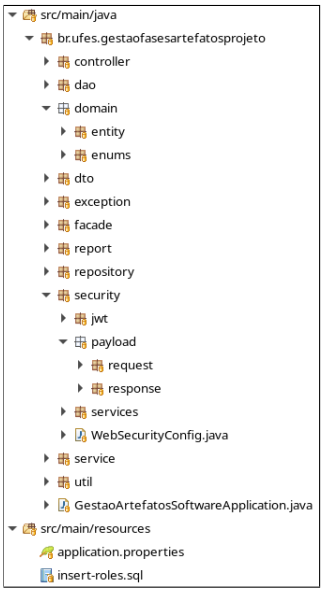
4.2 Backend

Conforme apresentado na figura 4 na sessão 2.2.3, o servidor *backend* é o responsável pelas regras de negócio, interface com banco de dados e atendimento das requisições do *frontend*. Para realizar esta última tarefa, são disponibilizadas APIs que são desacopladas do *frontend*, o que permite que diferentes sistemas se comuniquem e sejam atendidos.

4.2.1 Estrutura do projeto

A figura 12 exibe a estrutura dos pacotes da aplicação Java desenvolvida para o *backend*, e a tabela 3 as descreve.

Figura 12: Estrutura do projeto *backend*



Fonte: Do autor

Tabela 3: Descrição da estrutura do projeto *backend*

Pacote	Função
controller	Contém as classes responsáveis por atender as requisições HTTP
dao	Contém classes responsáveis por realizar comunicação com banco de dados
domain/entity	Contém as entidades do negócio mapeadas com o banco de dados
domain/enums	Contém os enumeradores das regras de negócio
dto	Contém as classes utilizadas na transferência de dados
exception	Contém as implementações de exceções específicas do negócio
facade	Contém as classes responsáveis por orquestrar as services
report	Contém as classes responsáveis pela geração de relatórios
repository	Contém classes responsáveis por realizar comunicação com banco de dados
security	Contém as classes responsáveis pela autenticação e segurança
service	Contém as classes responsáveis pelas regras de negócio
util	Contém as classes utilitárias

Fonte: Do autor

4.2.2 Conexão com banco de dados

Para a conexão do servidor *backend* com o banco de dados PostgreSQL, foi utilizada a biblioteca Spring Data JPA, em conjunto com o *driver* específico do PostgreSQL. Com isso, os dados da conexão ficam armazenados no arquivo "application.properties", como exibido na figura 13, o que torna mais fácil a alteração da conexão e da configuração de conexões diversas para perfis diferentes.

Figura 13: Informações da conexão com banco de dados

```
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:postgresql://localhost:5432/gestaoartefatosdb
spring.datasource.username=postgres
spring.datasource.password=romulo.123
```

Fonte: Do autor

Para usufruir dessa conexão com o banco de dados, foram implementadas interfaces que estendem as funcionalidades providas pela interface JpaRepository do Spring

Data JPA. Essa interface JpaRepository reduz significativamente a quantidade de código necessário para implementar uma camada de acesso a dados, e tudo que ela precisa é receber a classe de domínio que ela deve gerenciar e o tipo do id dessa classe Spring (2022). A figura 14 mostra um exemplo de *repository* implementada para a entidade Arquivo.

Figura 14: Exemplo de repository

```
@Repository
public interface RoleRepository extends JpaRepository<Role, Long> {

    List<Role> findByIdIn(List<Long> id) throws Exception;

    Optional<Role> findOneByName(RoleName name) throws Exception;

}
```

Fonte: Do autor

Pela figura 14, é possível ressaltar também os *Query Methods* presentes no Spring Data JPA, onde é possível derivar uma *query* a partir do nome do método. Na *repository* da figura 14, existe o método "findByIdIn" que recebe uma lista de ID do tipo *Long* e encontra (*find*) todos os registros de Role cujo ID estão (*in*) na lista de ID fornecida.

4.2.3 Endpoints

Por padrão, o Spring Boot utiliza o servidor Apache Tomcat, que é um *container* servlet. Um servlet é basicamente um módulo de *software* executado em um servidor *web* para atender as requisições das aplicações clientes. Com isso, o servidor *backend* recebe as requisições HTTP, as processa e devolve uma reposta à aplicação cliente (PARASCHIV, 2017).

Para tal finalidade, o servidor foi desenvolvido utilizando a abordagem REST, visando torná-lo uma API com *endpoints* para serem consumidos por aplicações clientes. Um *endpoint* é, de maneira resumida, uma URI referente a um recurso da aplicação.

A disponibilização desses *endpoints* foi feita por meio de *controllers*, que são as classes responsáveis por expor as informações e ações para as aplicações clientes. Além disso, toda informação transferida entre aplicação cliente e *controller*, foi feita por meio do formato JSON.

Figura 15: Exemplo de controller

```
@RestController
@RequestMapping("api/projetos")
public class ProjetoController extends BaseController {

    @Autowired
    private ProjetoFacade projetoFacade;

    @GetMapping("/{id}")
    @PreAuthorize("hasAnyRole('ROLE_GERENTE_PROJETOS', 'ROLE_ANALISTA_REQUISITOS')")
    public ResponseEntity<?> getById(@PathVariable("id") Long id) throws Exception {
        try {
            return ResponseEntity.ok(projetoFacade.getById(id));
        } catch (BusinessException e) {
            return badRequest(e);
        } catch (Exception e) {
            return internalServerError(e);
        }
    }

    @GetMapping
    @PreAuthorize("hasAnyRole('ROLE_GERENTE_PROJETOS', 'ROLE_ANALISTA_REQUISITOS')")
    public ResponseEntity<?> getProjetosListagem() throws Exception {
        try {
            return ResponseEntity.ok(projetoFacade.getProjetosListagem());
        } catch (BusinessException e) {
            return badRequest(e);
        } catch (Exception e) {
            return internalServerError(e);
        }
    }

    @PostMapping
    @PreAuthorize("hasRole('ROLE_GERENTE_PROJETOS')")
    public ResponseEntity<?> insert(@RequestBody NovoProjetoDTO newProjetoDTO) {
        try {
            return ResponseEntity.status(HttpStatus.CREATED).body(projetoFacade.insert(newProjetoDTO));
        } catch (BusinessException e) {
            return badRequest(e);
        } catch (Exception e) {
            return internalServerError(e);
        }
    }

    @PutMapping
    @PreAuthorize("hasRole('ROLE_GERENTE_PROJETOS')")
    public ResponseEntity<?> update(@RequestBody AtualizacaoProjetoDTO atualizacaoProjetoDTO) {
        try {
            return ResponseEntity.ok(projetoFacade.update(atualizacaoProjetoDTO));
        } catch (BusinessException e) {
            return badRequest(e);
        } catch (Exception e) {
            return internalServerError(e);
        }
    }

}
```

Fonte: Do autor

4.2.4 Façades

Segundo Gamma et al. (2000), as *façades* têm como responsabilidade fornecer uma interface simplificada para um conjunto complexo de classes. Nesse sentido, foram criadas *façades* para cada entidade que as *controllers* gerenciam. Dessa forma, as *façades* fornecem uma interface simplificada para as *controllers*, além de orquestrarem uma ou mais *services* de modo que toda a regra de negócio para determinada ação seja executada.

Figura 16: Exemplo de facade

```java
@Component
public class ProjetoFacade {

    @Autowired
    private ProjetoService projetoService;

    @Autowired
    private EtapaProjetoService etapaProjetoService;

    @Autowired
    private HistoricoAcaoProjetoService historicoAcaoProjetoService;

    public ProjetoDTO getById(Long id) throws Exception {
        return ModelMapperUtil.map(projetoService.getById(id), ProjetoDTO.class);
    }

    public List<ProjetoListagemDTO> getProjetosListagem() throws Exception {
        return projetoService.getProjetosListagem();
    }

    public ProjetoDTO insert(NovoProjetoDTO novoProjetoDTO) throws Exception {
        var projeto = projetoService.iniciarProjeto(novoProjetoDTO);

        historicoAcaoProjetoService.incluirAcao(projeto, AcaoProjeto.INICIO_PROJETO, null);

        iniciarPrimeiraEtapa(projeto);
        iniciarEtapaMetodoAgil(projeto);

        return ModelMapperUtil.map(projeto, ProjetoDTO.class);
    }

    private void iniciarPrimeiraEtapa(Projeto projeto) throws Exception {
        var etapaProjeto = etapaProjetoService.iniciarEtapa(projeto, FaseProjeto.getFaseInicial());
        historicoAcaoProjetoService.incluirAcao(projeto, AcaoProjetoUtil.getAcaoByOperacaoAndArtefato(TipoOperacao.INICIO_FASE, FaseProjeto.getFaseInicial().name()), etapaProjeto);
    }

    private void iniciarEtapaMetodoAgil(Projeto projeto) throws Exception {
        var etapaProjetoMetodoAgil = etapaProjetoService.iniciarEtapa(projeto, FaseProjeto.METODO_AGIL);
        historicoAcaoProjetoService.incluirAcao(projeto, AcaoProjetoUtil.getAcaoByOperacaoAndArtefato(TipoOperacao.INICIO_FASE, FaseProjeto.METODO_AGIL.name()), etapaProjetoMetodoAgil);
    }

    public ProjetoDTO update(AtualizacaoProjetoDTO atualizacaoProjetoDTO) throws Exception {
        var projeto = projetoService.update(atualizacaoProjetoDTO);

        historicoAcaoProjetoService.incluirAcao(projeto, AcaoProjeto.ATUALIZACAO_PROJETO, null);

        return ModelMapperUtil.map(projeto, ProjetoDTO.class);
    }

}
```

Fonte: Do autor

A figura 16 mostra a *façade* responsável pela entidade Projeto. Os parâmetros de seus públicos métodos vêm diretamente da *controller* e são destinados para a *service* que deve tratá-los.

Como exemplo, o método *insert* recebe uma instância de NovoProjetoDTO - objeto de transferência de dados com as informações para criação de novo projeto - e a passa para a *service* de projeto para iniciar um novo projeto. Com o projeto iniciado, o método incluirAcao da *service* de histórico de ações do projeto é chamado, para registrar no histórico o início do projeto. Após isso, dois métodos são chamados: iniciarPrimeiraEtapa e iniciarEtapaMetodoAgil, que iniciam a primeira etapa de um projeto e também iniciam a etapa de método ágil, registrando ambas ações também no histórico de ações do projeto. Por fim, o projeto que foi cadastrado é mapeado para o ProjetoDTO e retornado para a *controller*.

4.2.5 Services

Segundo Evans (2016), as *services* têm a responsabilidade de gerir as regras de negócio. Além disso, elas são responsáveis por se comunicar com a camada *Repository* - que faz a persistência dos dados.

Figura 17: Exemplo de service

```java
@Service
public class ProjetoService {

    @Autowired
    private ProjetoRepository projetoRepository;

    public Projeto getById(Long id) throws Exception {
        if (id == null) {
            throw new BusinessException("ID não fornecido para obtenção do Projeto");
        }

        return projetoRepository.getById(id);
    }

    public List<ProjetoListagemDTO> getProjetosListagem() throws Exception {
        return projetoRepository.getProjetosListagem();
    }

    public Projeto iniciarProjeto(NovoProjetoDTO novoProjetoDTO) throws Exception {
        validateInsert(novoProjetoDTO);

        var projeto = createFromDTO(novoProjetoDTO);
        return projetoRepository.save(projeto);
    }

    public Projeto update(AtualizacaoProjetoDTO atualizacaoProjetoDTO) throws Exception {
        validateUpdate(atualizacaoProjetoDTO);

        var projeto = getById(atualizacaoProjetoDTO.getId());
        if (projeto == null) {
            throw new BusinessException("Não foi possível obter um projeto com o id " + atualizacaoProjetoDTO.getId());
        }

        projeto.setNome(atualizacaoProjetoDTO.getNome());
        projeto.setDescricao(atualizacaoProjetoDTO.getDescricao());

        return projetoRepository.save(projeto);
    }

    private Projeto createFromDTO(NovoProjetoDTO novoProjetoDTO) throws Exception {
        var projeto = new Projeto();

        projeto.setNome(novoProjetoDTO.getNome());
        projeto.setDescricao(novoProjetoDTO.getDescricao());
        projeto.setDataInclusao(LocalDateTime.now());

        return projeto;
    }

    private void validateInsert(NovoProjetoDTO novoProjetoDTO) throws Exception {
        List<String> errors = new ArrayList<>();

        if (novoProjetoDTO == null) {
            errors.add("Nenhuma informação para criação de um novo projeto foi fornecida");
        } else {
            if (StringUtil.isNullOrBlank(novoProjetoDTO.getNome())) {
                errors.add("Nome do projeto não foi fornecido");
            }
        }

        if (!errors.isEmpty()) {
            throw new BusinessException(errors);
        }
    }

    private void validateUpdate(AtualizacaoProjetoDTO atualizacaoProjetoDTO) throws Exception {
        List<String> errors = new ArrayList<>();

        if (atualizacaoProjetoDTO == null) {
            errors.add("Nenhuma informação para atualização do projeto foi fornecida");
        } else {
            if (atualizacaoProjetoDTO.getId() == null) {
                errors.add("ID do projeto não foi fornecido");
            }

            if (StringUtil.isNullOrBlank(atualizacaoProjetoDTO.getNome())) {
                errors.add("Nome do projeto não foi fornecido");
            }
        }

        if (!errors.isEmpty()) {
            throw new BusinessException(errors);
        }
    }

}
```

Fonte: Do autor

A figura 17 mostra a *service* responsável pela entidade Projeto. Os seus métodos públicos recebem os parâmetros diretamente da *façade* e realizam alguns tratamentos de regra de negócio antes de passá-los para a camada *Repository* realizar a persis-

tência.

Como exemplo, o método *insert* recebe uma instância de NovoProjetoDTO e primeiramente realiza uma validação com o método validateInsert - que verifica se o objeto recebido não é nulo e se foi fornecido um nome para o projeto. Caso um objeto nulo tenha sido fornecido ou o nome do projeto não tenha sido fornecido, uma exceção é levantada e a inserção do projeto é interrompida. Caso contrário, uma instância de projeto é criada através do método createFromDTO e logo depois é persistida pelo método *save* da *repository* de projeto.

4.2.6 Entidades

O mapeamento e controle das classes e atributos que representam as abstrações do domínio do problema são feitas através da biblioteca Spring Data JPA. Para que o mapeamento entre a classe e o banco de dados aconteça, as entidades devem possuir a anotação @Entity. Além disso, outras anotações podem ser utilizadas para manter controle dos atributos e relacionamentos entre entidades.

Um exemplo de entidade é mostrado na figura 18.

Figura 18: Exemplo de entidade

```
@Entity
@Table(name = "PROJETO")
public class Projeto {

    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "nome", nullable = false)
    private String nome;

    @Column(name = "descricao")
    private String descricao;

    @Column(name = "data_inclusao", nullable = false, updatable = false)
    private LocalDateTime dataInclusao;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
```

```java
    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }

    public LocalDateTime getDataInclusao() {
        return dataInclusao;
    }

    public void setDataInclusao(LocalDateTime dataInclusao) {
        this.dataInclusao = dataInclusao;
    }

}
```

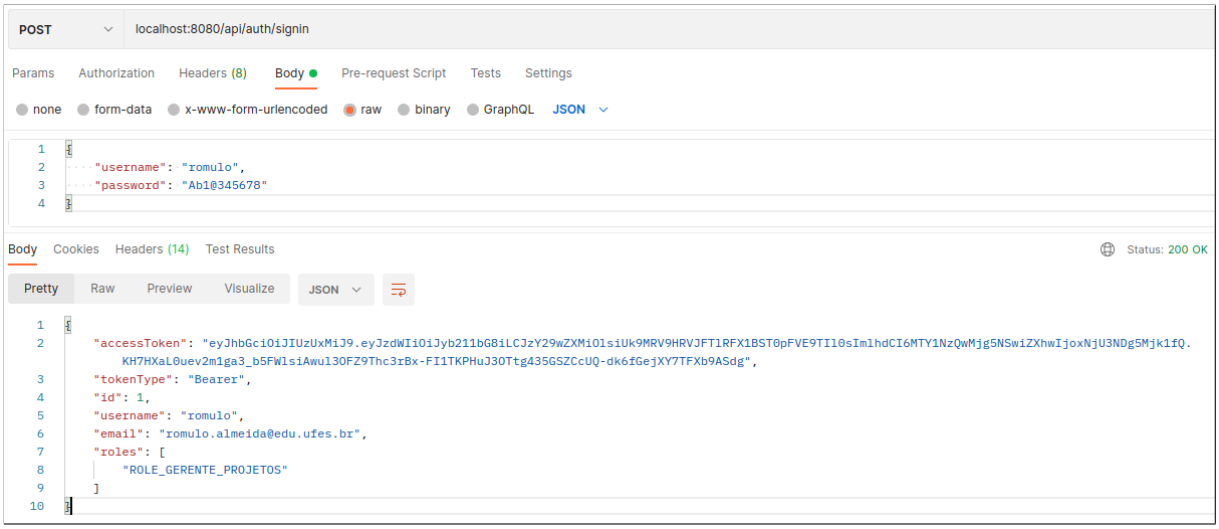
Fonte: Do autor

4.2.7 Segurança

A implementação da autenticação e autorização no servidor foi feita por meio do Spring Security. Esse *framework* permite que determinados recursos do servidor sejam disponibilizados somente para usuários autenticados e com as permissões necessárias. Para a realização da autenticação no servidor, foi utilizado o JSON Web Token (JWT), que é um padrão aberto (RFC 7519) que define uma maneira compacta de transmitir informações entre duas partes de forma segura através de um objeto JSON.

A figura 19 mostra um exemplo de login do usuário "romulo".

Figura 19: Exemplo de login via backend

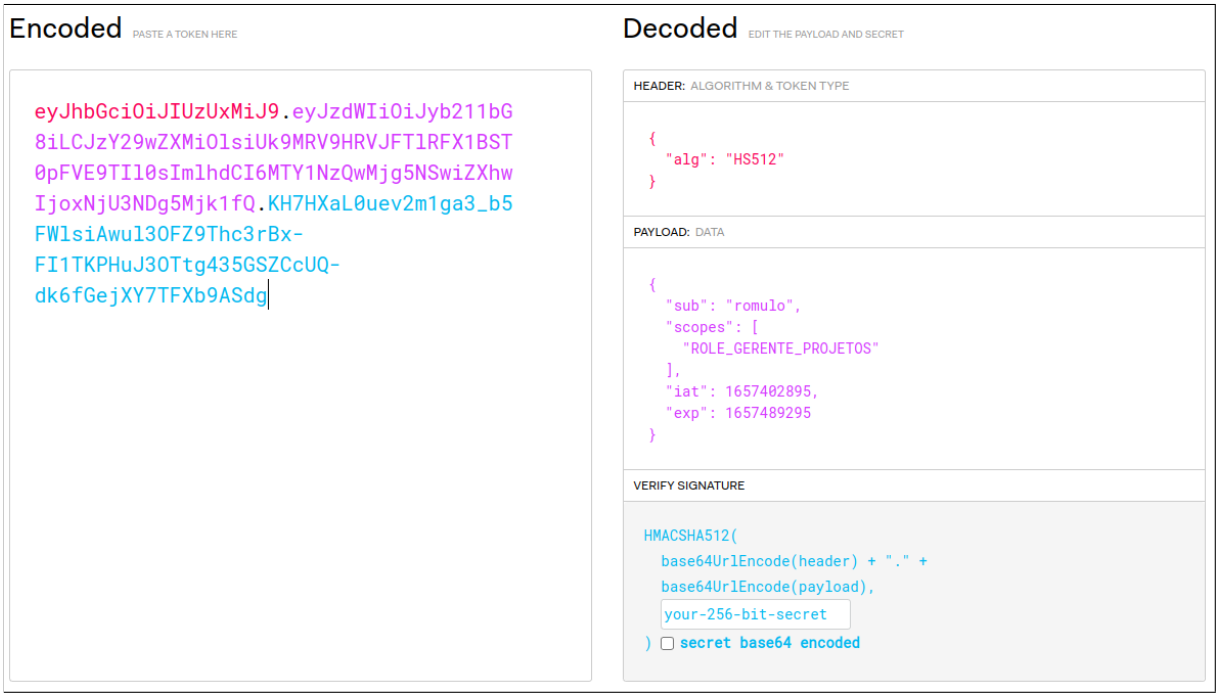


Fonte: Do autor

Ao realizar um login com sucesso, é retornado um JSON contendo o *access token* e

algumas informações do usuário, como seu *e-mail*, id e permissão (role). O *access token* é um JWT, e pode ser decodificado para interpretação de suas informações, como exibido na figura 20.

Figura 20: Exemplo de JWT decodificado



Fonte: Do autor

Como mostrado na figura 20, o JWT é composto por três partes: *header*, *payload* e *signature*. A primeira parte, que é o *header*, é onde fica a informação do algoritmo de assinatura utilizado, que no caso da figura 20 é o Sha512. Já na segunda parte, *payload*, é onde ficam as informações de usuário e dados adicionais, como por exemplo o *username* e as *roles* que ele possui (*scopes*). Na terceira e última parte, ficam as informações de assinatura, que são utilizadas para verificar se a mensagem não foi alterada durante o caminho.

Para acessar algum recurso ou realizar alguma operação que necessite de autenticação ou autorização específica, o usuário deve informar no *header* da requisição HTTP o seu *access token*, para que a *controller* realize a verificação do JWT e então libere o recurso desejado. A figura 21 mostra um JWT no *header* da requisição.

Figura 21: JWT no header da requisição

	KEY	VALUE
☑	Authorization ⓘ	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJyb211bG8iL

Fonte: Do autor

Para rotas em que são necessárias autorizações específicas (rotas com a anotação @PreAuthorize), igual à rota mapeada pelo método *getById* mostrado na figura 22, o *access token* fornecido deve possuir a *role* exigida na anotação @PreAuthorize dentro da sua lista "*scopes*".

Figura 22: Exemplo de rota que exige autorização

```
@RestController
@RequestMapping("api/projetos")
public class ProjetoController extends BaseController {

    @Autowired
    private ProjetoFacade projetoFacade;

    @GetMapping("/{id}")
    @PreAuthorize("hasAnyRole('ROLE_GERENTE_PROJETOS', 'ROLE_ANALISTA_REQUISITOS')")
    public ResponseEntity<?> getById(@PathVariable("id") Long id) throws Exception {
        try {
            return ResponseEntity.ok(projetoFacade.getById(id));
        } catch (BusinessException e) {
            return badRequest(e);
        } catch (Exception e) {
            return internalServerError(e);
        }
    }
```

Fonte: Do autor

4.3 Frontend

O *frontend* da aplicação foi desenvolvido utilizando o *framework* Angular. Com ele foi construída uma *single-page application* (SPA), que é uma aplicação de página única. Nesse tipo de aplicação, todas as funcionalidades existem em uma única página HTML, que é alterada de acordo com as funcionalidades da aplicação que o usuário acessa.

Além disso o Angular utiliza componentes para construir as aplicações. Os componentes são classes construídas com TypeScript que possuem a anotação @Component e eles servem como alicerces de uma aplicação. Cada componente pode ser compostos por três partes: uma classe TypeScript com as lógicas e funções de programação; um *template* HTML que instrui o Angular a como desenhar esse componente e um

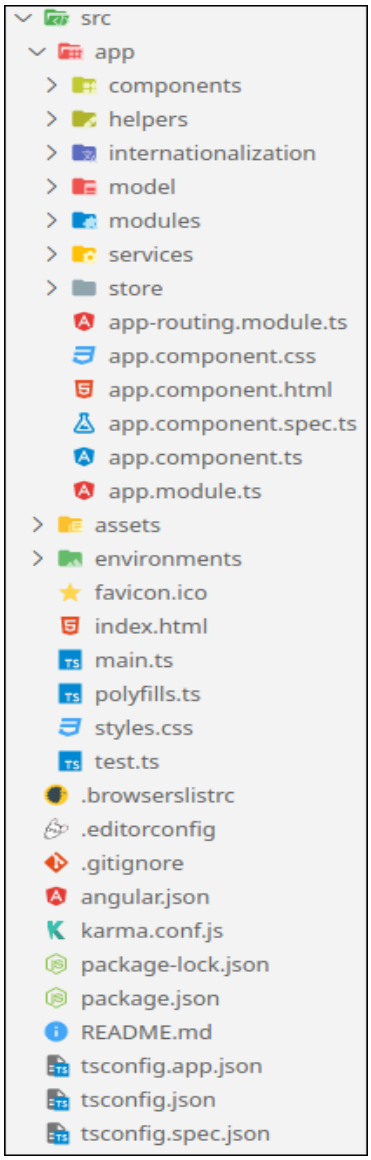
arquivo CSS de estilo para estilização do componente.

Com essa ideia de componentes, o Angular oferece um forte encapsulamento das funcionalidades desenvolvidas, além de facilitar a leitura do código e o reuso.

4.3.1 Estrutura do frontend

A figura 23 mostra a estrutura do projeto Angular e a tabela 4 descreve as pastas.

Figura 23: Estrutura do frontend



Fonte: Do autor

Tabela 4: Descrição da estrutura do projeto *frontend*

Pasta	Função
components	Contém componentes feitos para serem reutilizados
helpers	Contém classes utilizadas para facilitar atividades comuns no sistema
internationalization	Contém classes utilizadas para definição de liguagem dos componentes
model	Contém classes de dados utilizadas em comunicações com o servidor backend
modules	Contém as páginas do sistema, separadas em módulos por tópico
services	Contém classes utilizadas para realizar atividades dentro do sistema
store	Contém classes responsáveis pela comunicação com o servidor backend
app-routing.module.ts	Arquivo responsável pelo mapeamento de páginas em links
app.module.ts	Arquivo de configuração do módulo principal da aplicação
environments	Contém as classes que possuem as informações do ambiente, como URL do servidor
index.html	Página principal da aplicação, onde as funcionalidades são renderizadas

Fonte: Do autor

4.3.2 Comunicação com servidor

A comunicação com o servidor *backend* é feita por *stores*, que são responsáveis por realizar as requisições HTTP, utilizando a biblioteca HttpClient presente no Angular. Todas as *stores* do sistema estendem de uma classe chamada "BaseStore", que é exibida na figura 24.

Figura 24: BaseStore

```
export abstract class BaseStore {

    private _controllerUrl: string;

    constructor(controller: string) {
        this._controllerUrl = environment.appUrl + controller;
    }

    getBaseUrl(path?: string) {
        return this._controllerUrl + (path || '');
    }

}
```

Fonte: Do autor

Como mostrado na figura 24, a "BaseStore" exige em seu construtor o *path* da *controller* que a *store* deseja se comunicar, para que seja concatenado com a URL do servidor definida no arquivo de *environment*. Além disso, ela define o método "getBaseUrl" que retorna a URL da *controller* concatenada com o *path* passado por parâmetro. A figura 25 mostra o arquivo de *environment* onde está a URL do servidor, já a figura 26 mostra um exemplo de *store* que estende essas funcionalidades.

Figura 25: Arquivo de *environment*

```
export const environment = {
  production: false,
  appUrl: 'http://localhost:8080/',
  authApi: 'http://localhost:8080/api/auth/'
};
```

Fonte: Do autor

No objeto exportado no arquivo *environment* da figura 25 ficam informações como a URL do servidor e do serviço de autenticação.

Figura 26: Exemplo de *store*

```
@Injectable()
export class ProjetoStore extends BaseStore {

    constructor(private http: HttpClient) {
        super('api/projetos/');
    }

    getById(id: number) {
        return this.http.get<ProjetoModel>(this.getBaseUrl(`${id}`));
    }

    getProjetosListagem() {
        return this.http.get<ProjetoListagemModel[]>(this.getBaseUrl());
    }

    insert(novoProjetoModel: NovoProjetoModel): Observable<ProjetoModel> {
        return this.http.post<ProjetoModel>(this.getBaseUrl(), novoProjetoModel);
    }

    update(atualizacaoProjetoModel: AtualizacaoProjetoModel): Observable<ProjetoModel> {
        return this.http.put<ProjetoModel>(this.getBaseUrl(), atualizacaoProjetoModel);
    }

}
```

Fonte: Do autor

Como mostra a *store* de exemplo da figura 26, todos os métodos que realizam comunicação com o servidor utilizam a biblioteca HttpClient do Angular, fazendo uso do método referente ao verbo HTTP desejado. Dessa forma, é possível informar *Headers*, *Query params* e etc. Um exemplo de *headers* nas requisições é para informar o *access token* para acesso a recursos protegidos no servidor backend. A implementação dessa funcionalidade foi implementada no sistema *frontend* por meio de um *interceptor*, como mostrado na figura 27.

Figura 27: Interceptor para *token* de acesso

```
const TOKEN_HEADER_KEY = 'Authorization';

@Injectable()
export class AuthInterceptor implements HttpInterceptor {

    constructor(private tokeStorageService: TokenStorageService) {}

    intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
        let authReq = req;
        const token = this.tokeStorageService.getToken();

        if (token != null) {
            authReq = req.clone({ headers: req.headers.set(TOKEN_HEADER_KEY, 'Bearer ' + token) });
        }

        return next.handle(authReq);
    }

}


export const authInterceptorProviders = [
    { provide: HTTP_INTERCEPTORS, useClass: AuthInterceptor, multi: true }
];
```

Fonte: Do autor

A figura 27 mostra o *interceptor* implementado que implementa a interface HttpInterceptor definida pelo Angular e sobrescreve o método *intercept*, que permite realizar alguma ação com a requisição interceptada. No método *intercept* em questão, a requisição HTTP é interceptada para que em seu *headers* seja colocado o token de acesso obtido pelo "TokenStorageService" no *header* "Authorization". O "TokenStorageService" é exibido na figura 28.

Figura 28: TokenStorageService

```
@Injectable()
export class TokenStorageService {

    signOut() {
        sessionStorage.clear();
    }

    saveToken(token: string) {
        sessionStorage.removeItem(TOKEN_KEY);
        sessionStorage.setItem(TOKEN_KEY, token);
    }

    getToken(): string | null {
        return sessionStorage.getItem(TOKEN_KEY);
    }

    saveUser(user: any) {
        sessionStorage.removeItem(USER_KEY);
        sessionStorage.setItem(USER_KEY, JSON.stringify(user));
    }

    getUser(): JwtResponseModel | null {
        const userStr = sessionStorage.getItem(USER_KEY);

        if (userStr == null) {
            return null;
        }

        return JSON.parse(userStr);
    }

    hasRole(role: string) {
        const user = this.getUser();

        return (user != null && !!user.roles) ? user.roles.some(roleName => roleName == role) : false;
    }

}
```

Fonte: Do autor

A figura 28 mostra a classe que gerencia no *frontend* o *token* obtido pela API de login

exibida na figura 19. O *token* obtido é armazenado no *session storage* do navegador, ficando disponível enquanto a aba em que o usuário está usando está aberta.

4.3.3 Navegação

A partir da listagem de projetos, que funciona como uma interface principal, o usuário tem acesso a todas as telas do sistema, que funcionam de acordo com o projeto escolhido.

O controle da navegação para essas páginas foi feito utilizando a biblioteca *Router* do Angular. Para tal, as rotas e os respectivos componentes que elas devem renderizar foram definidos no arquivo "app-routing.module.ts". O conteúdo desse arquivo é mostrado na figura 29.

Figura 29: Arquivo de configuração do roteamento

```
const routes: Routes = [
  {path: '', redirectTo: 'signup', pathMatch: 'full'},
  {path: 'signup', component: SignUpComponent},
  {path: 'signin', component: SignInComponent},
  {path: 'lista-projetos', component: ProjetoListComponent},
  {path: 'projeto-overview/:uuid', component: ProjetoOverviewComponent},
  {path: 'lista-artefatos/:uuid', component: ArtefatoListComponent},
  {path: 'form-artefato/:uuid', component: ArtefatoFormComponent},
  {path: 'lista-requisitos/:uuid', component: RequisitoListComponent},
  {path: 'form-requisito/:uuid', component: RequisitoFormComponent},
  {path: 'lista-testes/:uuid', component: TesteListComponent},
  {path: 'form-teste/:uuid', component: TesteFormComponent},
  {path: 'lista-entidades/:uuid', component: EntidadeListComponent},
  {path: 'form-entidade/:uuid', component: EntidadeFormComponent},
  {path: 'lista-relatorios/:uuid', component: RelatorioListComponent},
  {path: 'lista-sprints/:uuid', component: SprintListComponent},
  {path: 'form-sprint/:uuid', component: SprintFormComponent},
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Fonte: Do autor

A figura 29 apresenta as definições de rota da aplicação, onde existe um vetor armazenado no objeto *routes* em que cada *path* está mapeada para um *component* específico da aplicação. Para que a navegação de uma rota para outra ocorra, é necessário que o componente da rota atual dispare um evento de navegação e informe os dados que ele quer passar para a próxima rota. Um exemplo dessa navegação é exibido na figura 30.

Figura 30: Exemplo de navegação entre rotas

```
const uuidRota = this.dataService.setItem({"idProjeto": id});
this.router.navigate(['projeto-overview', uuidRota]);
```

Fonte: Do autor

A figura 30 apresenta a navegação para a tela de *overview* de um projeto, que está disponível na rota /projeto-overview. Ao navegar para essa rota, o componente "ProjetoOverviewComponent" é carregado, conforme exibido na figura 29. Além disso, é importante destacar na figura 30 que para realizar a passagem de valores entre rotas, é utilizado o "DataService", que armazena o objeto passado por parâmetro no método "setItem" no *session storage* e retorna um UUID único, que é utilizado na rota da página. A implementação do "DataService" é mostrado na figura 31.

Figura 31: Implementação do DataService

```
@Injectable({
    providedIn: 'root'
})
export class DataService {

    private dataMap: Map<string, any> = new Map<string, any>();

    setItem(data: any): string | null {
        if (data == null) {
            return null;
        }

        const uuid = uuidv4();

        this.dataMap.set(uuid, data);

        return uuid;
    }

    getItem(uuid: string | null): any {
        if (!uuid) {
            return null;
        }

        return this.dataMap.get(uuid);
    }

}
```

Fonte: Do autor

Além do método "setItem" é possível observar também na figura 31 o método "getItem" que dado um UUID retorna o objeto presente no *session storage* mapeado para essa chave.

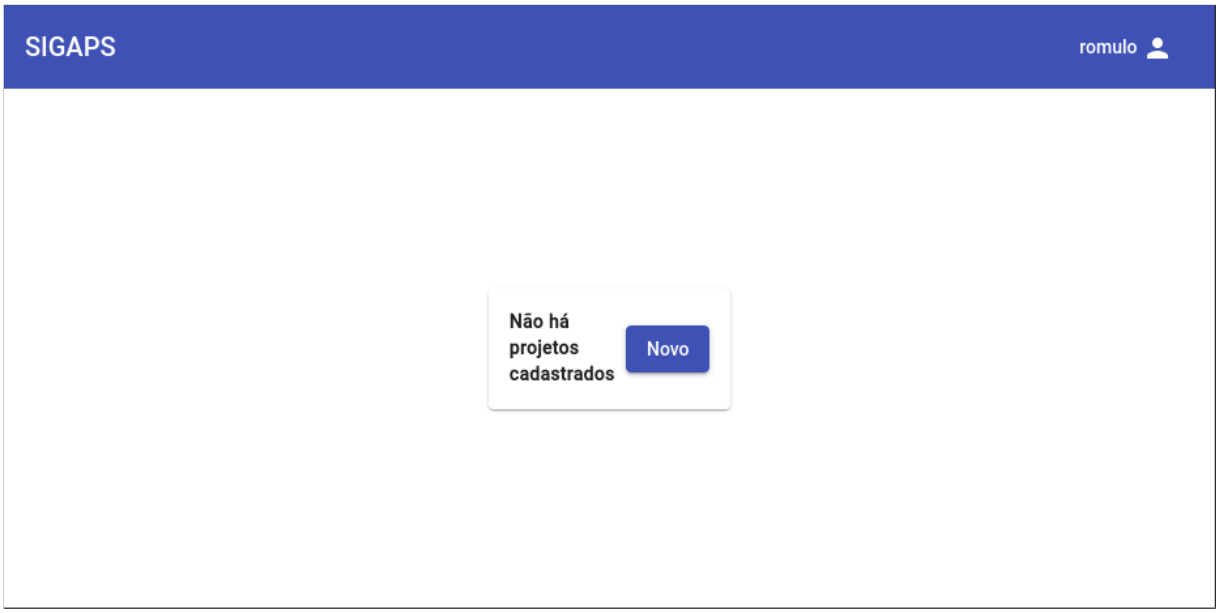
4.4 Interfaces

Nesta seção serão apresentadas as telas do sistema.

4.4.1 Tela de projetos

O sistema permite que o gerente de projetos realize o cadastro de projetos.

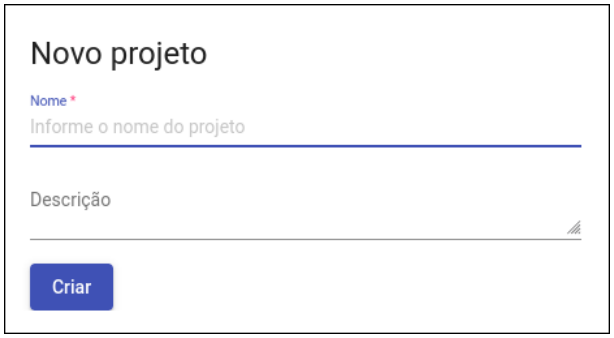
Figura 32: Tela de listagem de projetos vazia



Fonte: Do autor

A figura 32 mostra a tela inicial de um usuário logado no sistema, que é a listagem de projetos, que fica disponível na rota /lista-projetos. No caso da figura 32, ainda não há projetos cadastrados o que leva a exibição do botão "Novo" para o gerente de projetos, de forma que um projeto possa ser adicionado. A figura 33 mostra a modal que é exibida quando o usuário clica no botão "Novo".

Figura 33: Modal de novo projeto

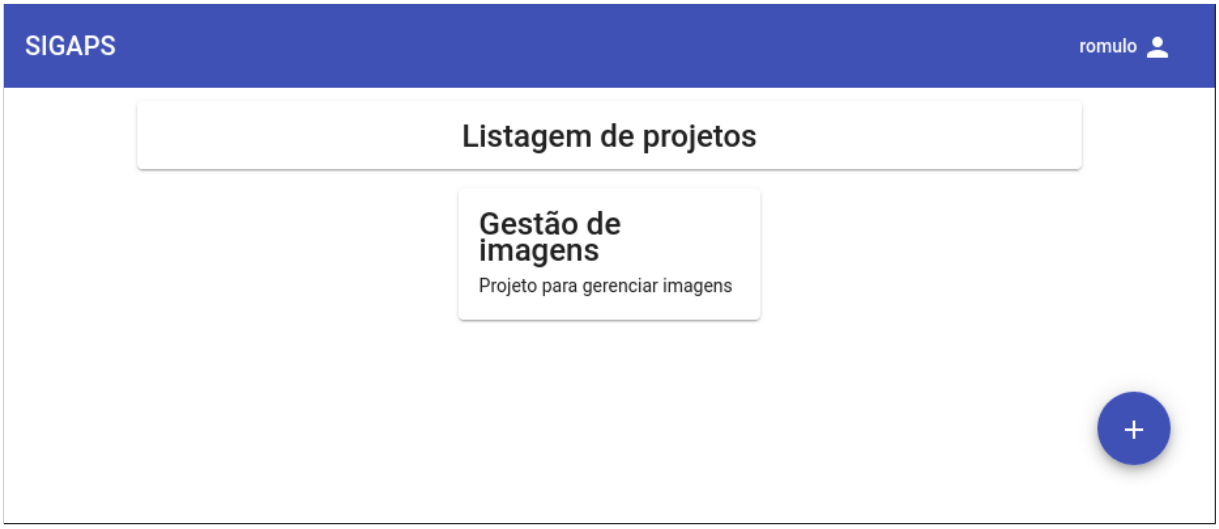


Fonte: Do autor

Conforme exibido na figura 33, ao criar um novo projeto o usuário informa um nome

(campo obrigatório) e uma descrição para o projeto. A figura 34 mostra a tela de listagem de projetos com registros cadastrados.

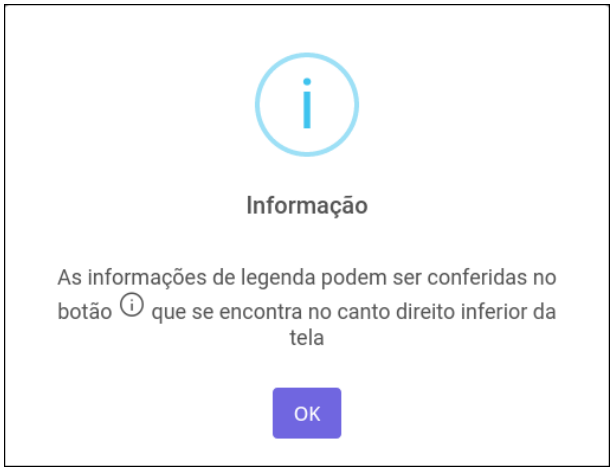
Figura 34: Tela de listagem de projetos com registro



Fonte: Do autor

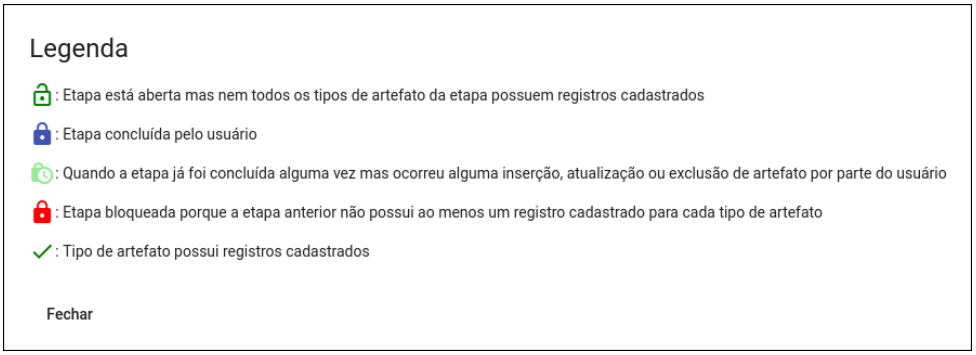
Ao clicar em algum projeto listado, assim como o da figura 34, o usuário é direcionado para a tela de *overview* do projeto, presente na rota /projeto-overview. Ao entrar na tela é exibida ao usuário uma modal, informando que as informações de legenda da tela podem ser visualizadas por meio de uma modal, que é aberta ao clicar no botão de informações no canto inferior direito da tela. As figuras 35 e 36 mostram a modal informativa sobre a legenda e a legenda, respectivamente.

Figura 35: Modal informativa sobre legenda



Fonte: Do autor

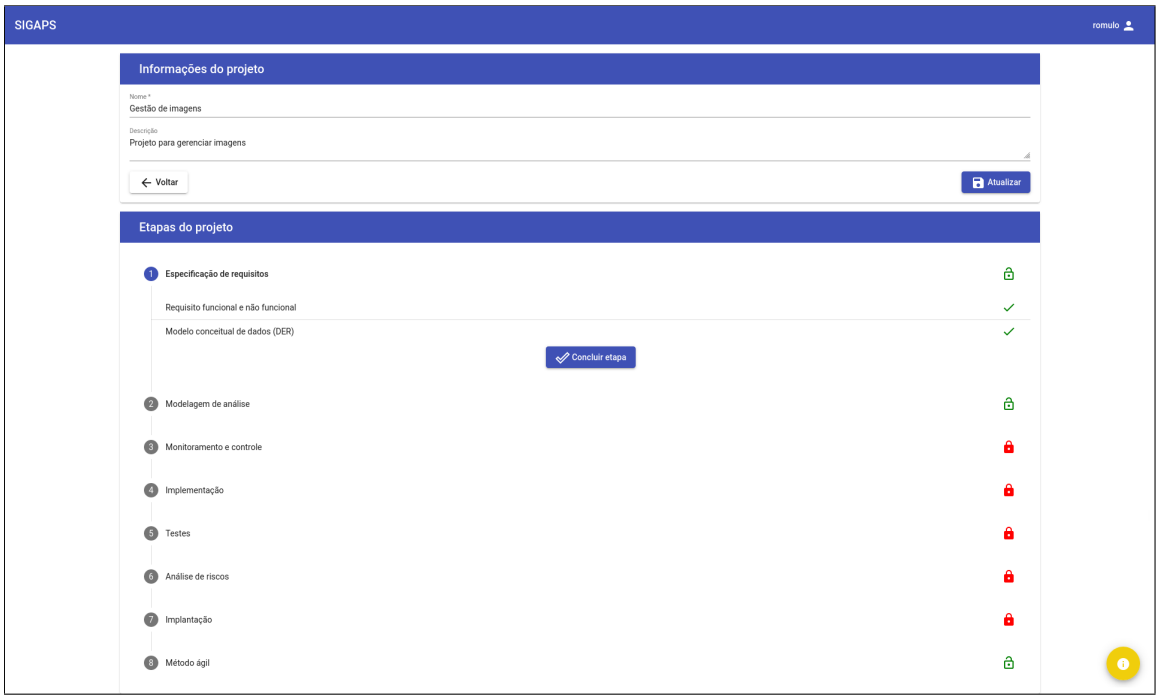
Figura 36: Legenda da tela de *overview* do projeto



Fonte: Do autor

Na tela de *overview* do projeto o usuário pode cadastrar os artefatos de cada fase do projeto. Quando um projeto é cadastrado, os módulos (que representam as fases) de "Especificação de requisitos" e "Método ágil" são automaticamente iniciados. Assim que o usuário conclui uma fase, ou seja, cadastra ao menos um registro para cada tipo de artefato da fase, a fase seguinte é liberada e um botão para concluir a fase anterior é liberado. As fases do projeto são liberadas de forma sequencial para dar ao usuário, sobretudo o aluno de graduação, que o trabalho realizado em uma fase do projeto depende e influencia em outras etapas deste. A figura 37 mostra a tela de *overview* do projeto.

Figura 37: Tela de *overview* do projeto

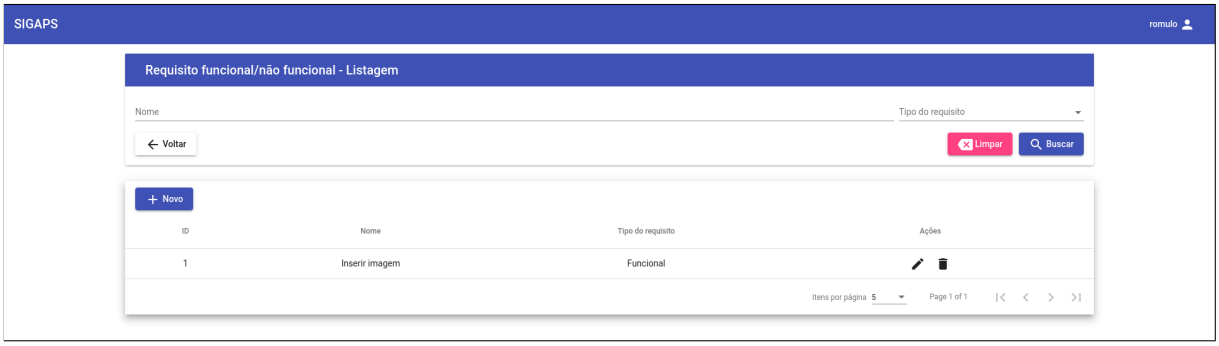


Fonte: Do autor

4.4.2 Requisitos funcionais e não funcionais

A tela de listagem de requisitos está disponível na rota /lista-requisitos e pode ser acessada pelo módulo "Especificação de requisitos" na tela de *overview* do projeto. Essa tela permite a busca paginada de requisitos por nome e tipo (funcional ou não funcional) e é exibida na figura 38.

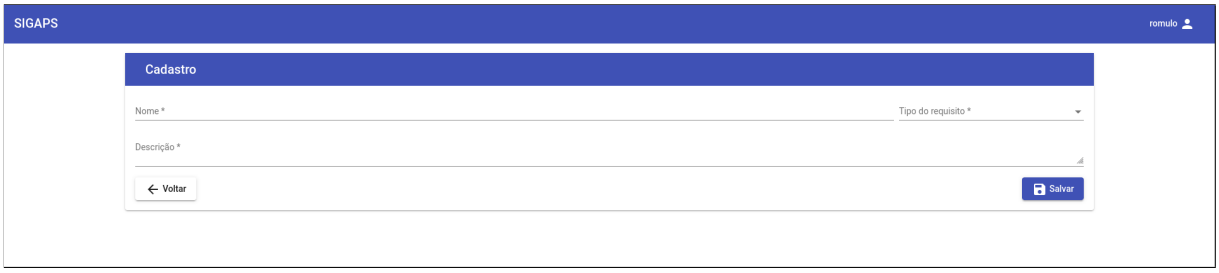
Figura 38: Tela de listagem de requisitos



Fonte: Do autor

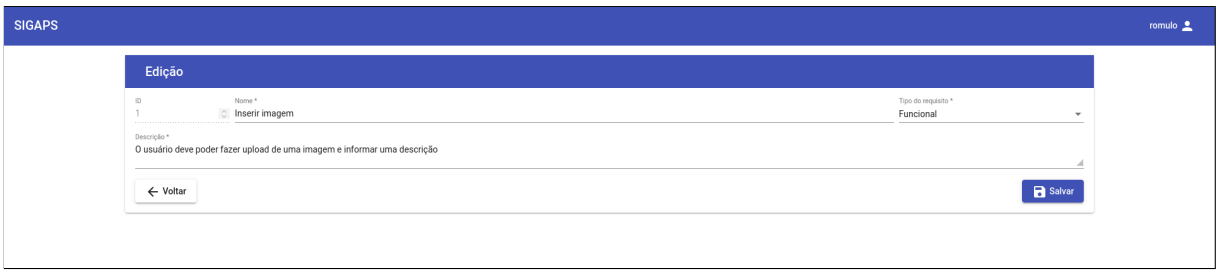
Como mostra a figura 38 também é possível excluir e editar requisitos já cadastrados. Ao excluir um requisito, acontece uma exclusão lógica no banco de dados e o registro deixa de aparecer na tela de listagem. Já quando o usuário clica no botão de editar, ele é direcionado para a mesma tela de cadastro de um novo requisito, mas em modo de edição. As figuras 39 e 40 mostram as telas de cadastro e edição de requisito, respectivamente.

Figura 39: Tela de cadastro de requisitos



Fonte: Do autor

Figura 40: Tela de edição de requisito



Fonte: Do autor

Como mostram as figuras 39 e 40, para todo requisito é obrigatório informar o seu nome, uma descrição e seu tipo (funcional ou não funcional). De acordo com Pressman e Maxim (2021), a especificação de requisitos pode assumir diferentes significados para cada pessoa, podendo ser um documento escrito, um conjunto de modelos gráficos, conjunto de cenários de uso ou até mesmo a combinação desses fatores. Tendo isso em vista, o formulário de cadastro requisitos desenvolvido visa fornecer uma forma geral básica para essa especificação, com o nome do requisito e sua descrição.

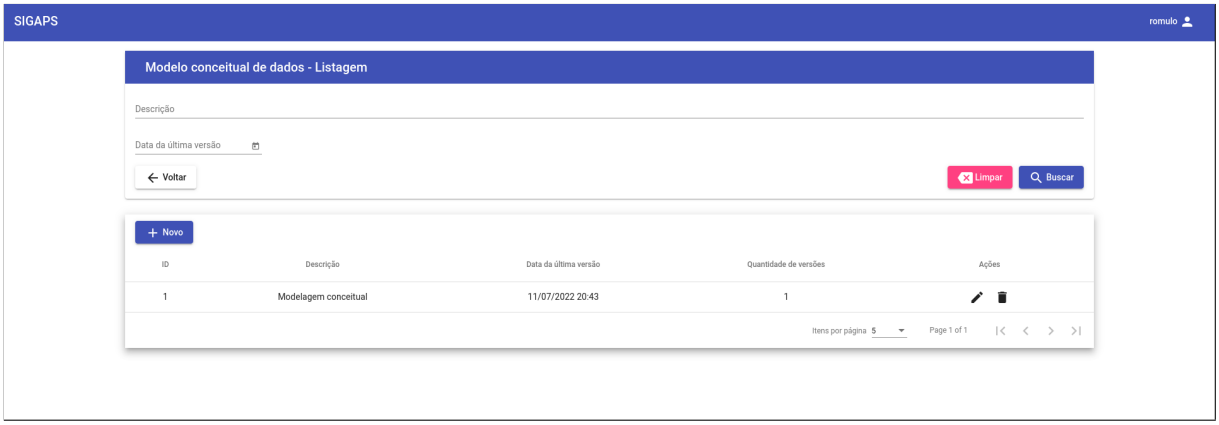
4.4.3 Artefatos

Durante o processo de desenvolvimento de um software é comum que existam mudanças nos diagramas de classe, diagrama de casos de uso, modelo conceitual de dados, protótipos, entre outros. Dessa forma, é interessante que o usuário possa submeter cada uma das versões produzidas desses artefatos, de modo a manter o histórico dessas versões. Tendo em vista essa característica, o sistema desenvolvido permite que o usuário realize *upload* do arquivo gerado na ferramenta em que utilizou para fazer seu diagrama de classes por exemplo, assim como um arquivo de prévia deste. Desse modo, cada vez que o usuário realiza um *upload* de um novo arquivo desse diagrama de classes, uma nova versão é gerada, mantendo a versão anterior em um histórico.

Devido a semelhança entre alguns tipos de artefato do sistema, que é a de manter o versionamento de arquivos externos fornecidos pelo usuário, foram criados componentes reutilizáveis de listagem e cadastro, de modo que o desenvolvimento ficasse mais fácil para todos os tipos de artefato que tinham os mesmos requisitos. Para exemplificar os componentes reutilizáveis, foi utilizado o tipo de artefato "Modelo con-

ceitual de dados" que pode ser acessado pelo módulo "Especificação de requisitos" na tela de *overview* do projeto. A figura 41 mostra a tela de listagem.

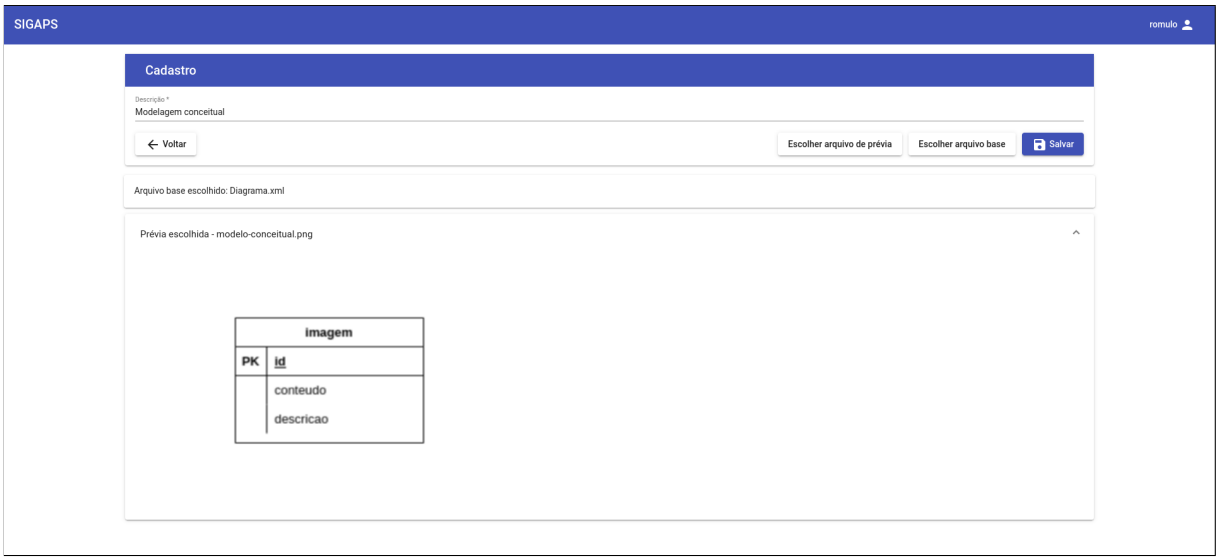
Figura 41: Tela de listagem de modelos conceituais de dados



Fonte: Do autor

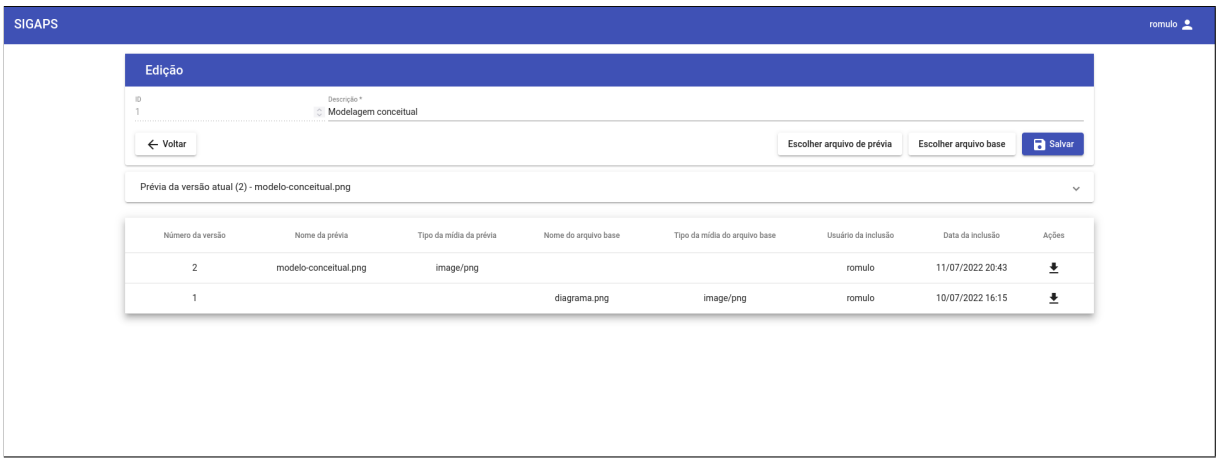
Na tela de listagem mostrada na figura 41 é possível realizar a busca de artefatos a partir de sua descrição e da data de sua última versão, isso é, a última vez que o usuário realizou *upload* de um arquivo base ou de prévia. Além disso, o usuário pode cadastrar um novo artefato, editar ou excluir um existente. As figuras 42 e 43 mostram as telas de cadastro e edição de artefatos.

Figura 42: Tela de cadastro de artefatos



Fonte: Do autor

Figura 43: Tela de edição de artefatos



Fonte: Do autor

Como mostra a figura 42, ao cadastrar um artefato o usuário deve fornecer uma descrição e pelo menos um arquivo base (arquivo exportado pela ferramenta externa que ele utilizou para desenvolver o artefato) ou um arquivo de prévia da versão (que é uma imagem PNG ou JPEG e pode ser visualizada na tela). Dependendo do tipo do artefato que está sendo cadastrado, o formulário pode solicitar somente o arquivo de prévia, somente o base ou qualquer um dos dois. A tabela 5 mostra os tipos de artefatos e os arquivos que são solicitados no formulário.

Ao realizar uma edição em um artefato cadastrado e inserir um novo arquivo de prévia ou base, uma nova versão é gerada contendo esses arquivo fornecidos. Dessa forma, o histórico das versões do artefato é mantido em uma tabela, que pode ser visualizada na figura 43. Além disso, para toda versão já cadastrada do artefato o usuário pode realizar o download dos arquivos fornecidos, sendo que a versão atual sempre tem o seu arquivo de prévia (se presente) exibido logo acima da tabela de versões do artefato.

Tabela 5: Tipo de artefato e seu arquivo exigido

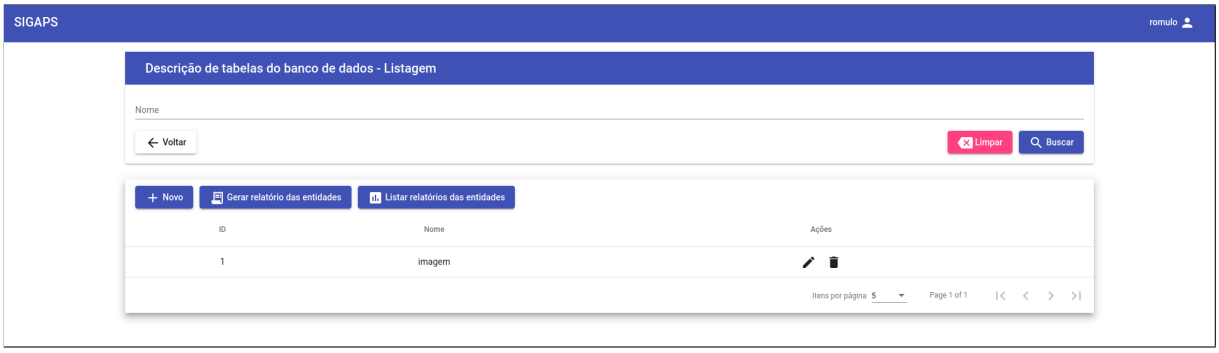
Tipo artefato	Arquivo exigido
Modelo conceitual de dados	Arquivo de prévia ou base
Diagrama de classes	Arquivo de prévia ou base
Diagrama de sequência	Arquivo de prévia ou base
Modelo lógico de dados	Arquivo de prévia ou base
Gráfico de Gantt	Arquivo de prévia ou base
Protótipo	Arquivo de prévia ou base
Código do projeto	Arquivo base
Modelo físico de dados	Arquivo de prévia ou base
Modelo de arquitetura	Arquivo de prévia ou base
Análise de riscos	Arquivo base
Tela do sistema	Arquivo de prévia ou base
Manual do usuário	Arquivo de prévia ou base

Fonte: Do autor

4.4.4 Descrição de tabelas do banco de dados

A tela de listagem de entidades do banco de dados está disponível na rota /lista-entidades e pode ser acessada pelo módulo "Implementação" na tela de *overview* do projeto. Essa tela permite a busca paginada de entidade do banco de dados por nome e é exibida na figura 44.

Figura 44: Tela de listagem de entidades do banco de dados

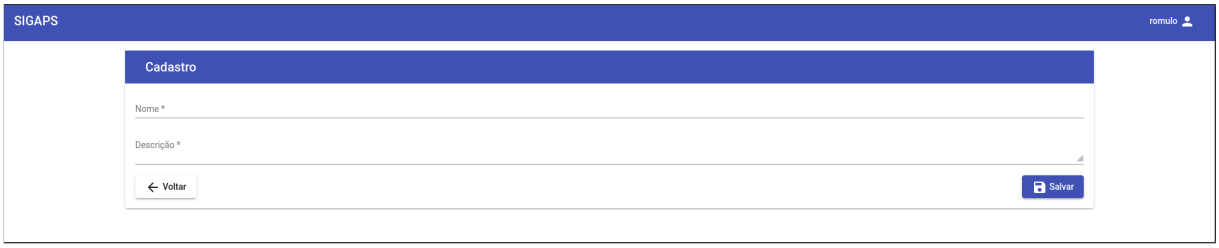


Fonte: Do autor

Como mostra a figura 44, é possível adicionar, editar e excluir entidades. Além disso,

é possível também gerar um relatório de todas as entidade já cadastradas no projeto, inclusive as excluídas.

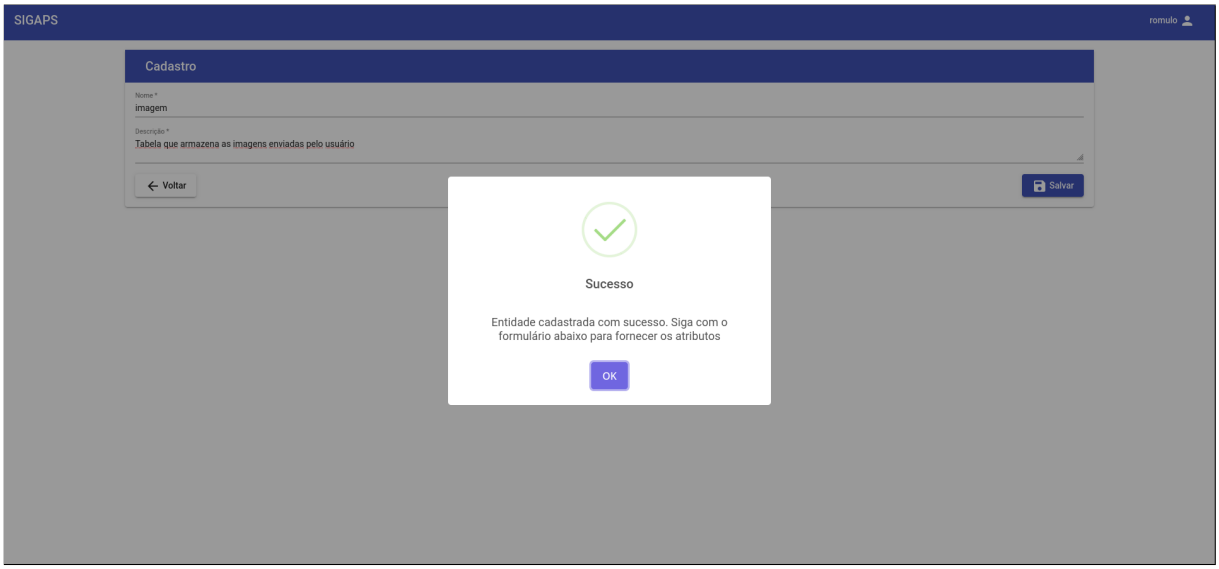
Figura 45: Tela de cadastro de entidades do banco de dados



Fonte: Do autor

Ao entrar na tela de cadastro de entidade (figura 45), é solicitado que o usuário informe um nome e uma descrição para a entidade. Ao informar e salvar essa entidade, o usuário recebe um alerta informando que o formulário para cadastro dos atributos da entidade foi liberado. A figura 46 mostra esse alerta.

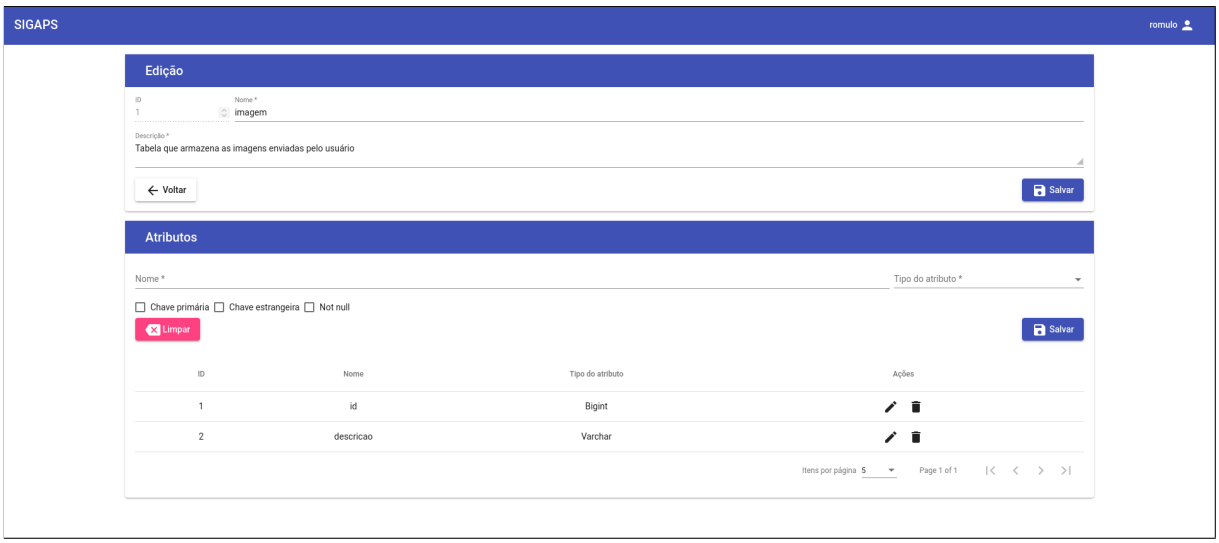
Figura 46: Alerta de formulário de atributos



Fonte: Do autor

Ao clicar em "Ok" o usuário permanece na mesma tela, mas agora com a opção de adicionar os atributos da entidade que ele está cadastrando, conforme mostra a figura 47.

Figura 47: Tela de cadastro de entidade do banco de dados com atributos



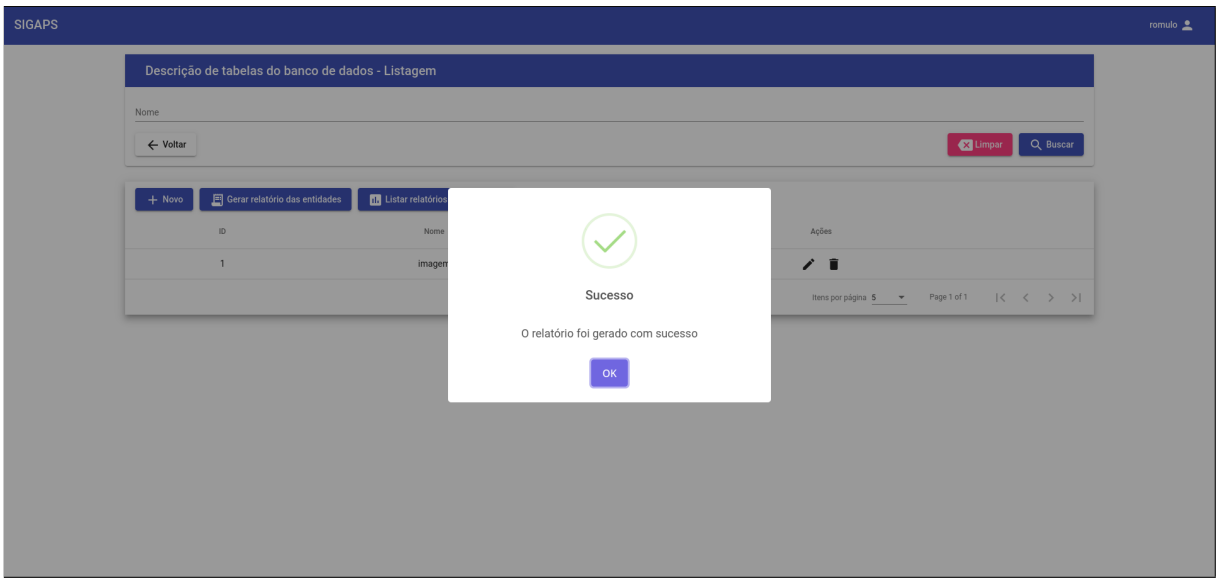
Fonte: Do autor

Como mostra a figura 47, para cadastrar um atributo o usuário deve fornecer um nome e o tipo do atributo. Além disso, o usuário pode informar se aquele atributo é *not null*, *primary key* ou *foreign key* ao marcar os campos de *checkbox*. Da mesma forma, o usuário pode também editar ou remover um atributo já cadastrado. Por fim, ao salvar um atributo é verificado se já existe algum outro para aquele mesmo nome, caso exista a operação é interrompida.

4.4.5 Geração de relatório das entidade

Como exibido na figura 44, é possível gerar relatórios das entidade já cadastradas no banco de dados. Para tal, o usuário deve clicar no botão "Gerar relatório das entidades" e esperar que a mensagem de sucesso apareça na tela, assim como mostrada na figura 48.

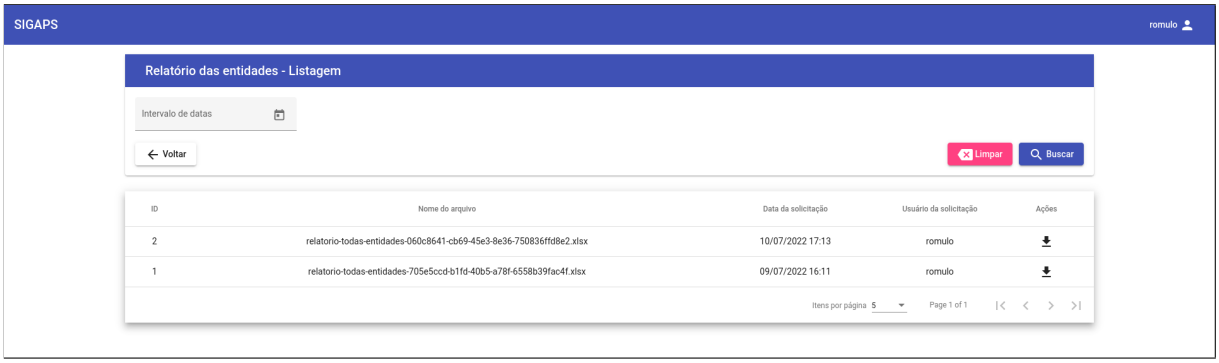
Figura 48: Geração de relatório de entidades



Fonte: Do autor

Para visualizar o relatório gerado, o usuário pode acessar a tela de listagem de relatório de entidade presente na rota /lista-relatorios através do botão "Listar relatórios das entidades" presente na tela de listagem de entidades. Na tela de listagem de relatórios é possível consultar todos os relatórios de entidade geradas e filtrar por um intervalo de datas de solicitação. Além disso, o usuário pode realizar o *download* de cada um desses relatórios. A figura 49 mostra essa tela.

Figura 49: Listagem dos relatórios de entidades



Fonte: Do autor

Ao realizar o download o usuário recebe um arquivo .xlsx contendo as informações do relatório gerado. A figura 50 mostra um exemplo de relatório gerado.

Figura 50: Exemplo de relatório de entidades

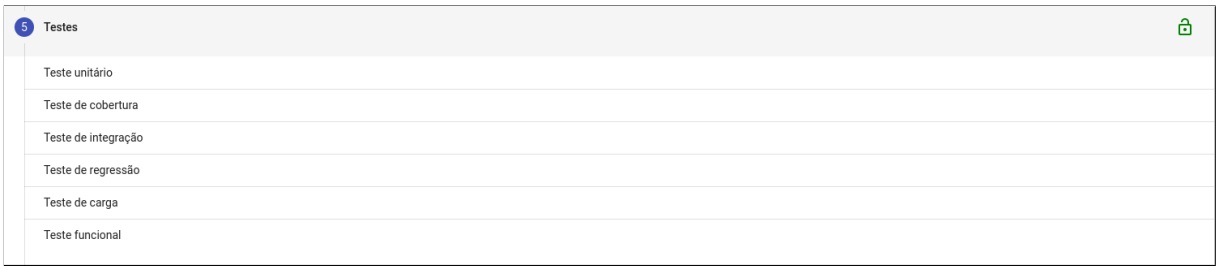
	A	B	C	D	E	F	G	H	I	J	K
1	ID entidade	Nome entidade	Descrição entidade	Entidade excluída?	ID atributo	Nome atributo	Tipo do atributo	Primary key?	Foreign key?	Not null?	Atributo excluído?
2	1	imagem	Tabela que armazena as imagens enviadas pelo usuário	Não	1	id	Bigint	Sim	Não	Sim	Não
3					2	descricao	Varchar	Não	Não	Sim	Não
4											
5											
6											
7											
8											
9											

Fonte: Do autor

4.4.6 Testes

As telas de listagem de testes estão disponíveis na rota /lista-testes e podem ser acessadas pelo módulo "Testes" na tela de *overview* do projeto. Apesar do módulo "Testes" exibir vários tipos de teste, como mostrado na figura 51, tanto a tela de listagem quanto as de cadastro e edição utilizam o mesmo componente internamente, exibindo um título diferente de acordo com o tipo de teste.

Figura 51: Tipos de teste do módulo "Testes"

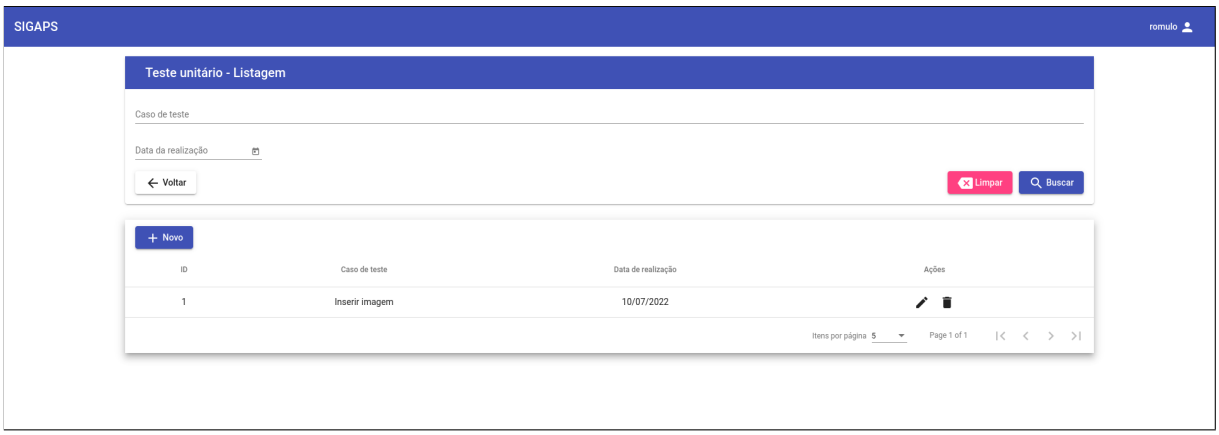


Fonte: Do autor

Como os formulários serão iguais para todos os tipos de teste exibidos na figura 51, para a demonstração da tela de listagem e cadastro será utilizado o "Teste unitário".

A tela de listagem permite a busca paginada dos testes do tipo em questão, sendo possível realizar filtro por caso de teste e data de realização do teste. A figura 52 mostra a tela de listagem de testes unitários.

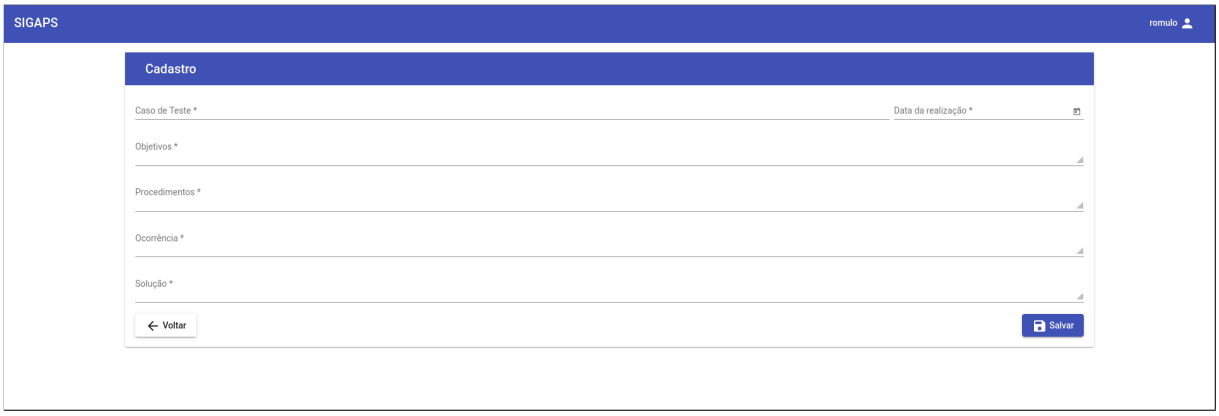
Figura 52: Listagem de testes unitários



Fonte: Do autor

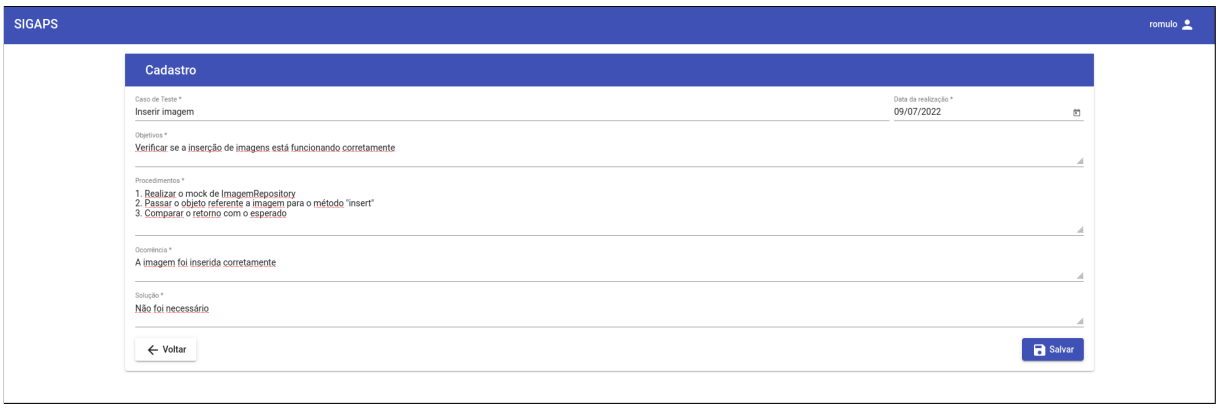
Como mostra a figura 52, é possível adicionar, editar e excluir testes. Ao clicar no botão "Novo" ou no ícone de lápis (edição) o usuário é direcionado para a mesma tela de formulário de teste, mas em modos distintos. As figuras 53 e 54 mostram o formulário de teste nos modos de cadastro e edição, respectivamente.

Figura 53: Cadastro de teste unitário



Fonte: Do autor

Figura 54: Edição de teste unitário



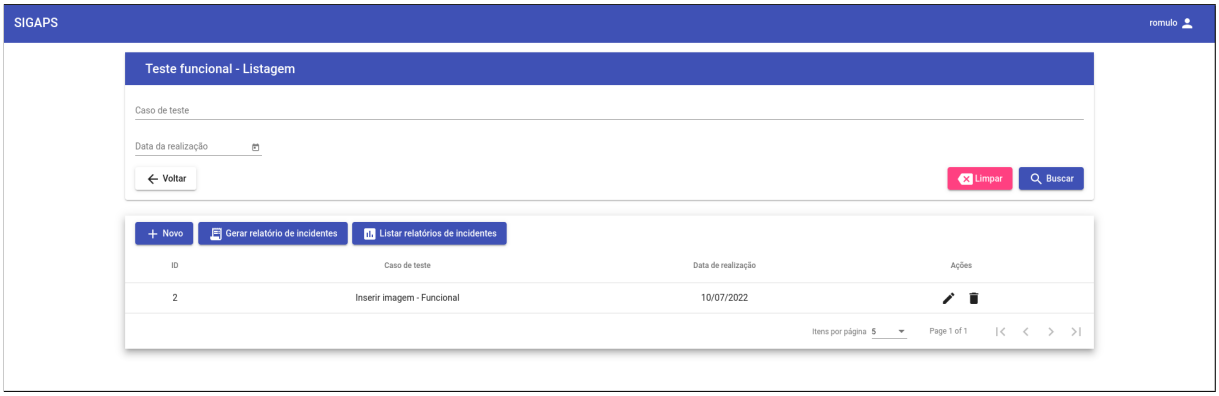
Fonte: Do autor

Como mostram as figuras 53 e 54 para todo teste é obrigatório informar os campos: caso de teste; data de realização; objetivos; procedimentos; ocorrência e solução.

4.4.7 Geração de relatório de incidentes de testes funcionais

Para os teste do tipo "Funcional", a tela de listagem exibe duas funcionalidades a mais: a geração de relatório dos testes funcionais e a listagem dos relatórios gerados. A figura 55 mostra essas duas funcionalidades na listagem de testes funcionais.

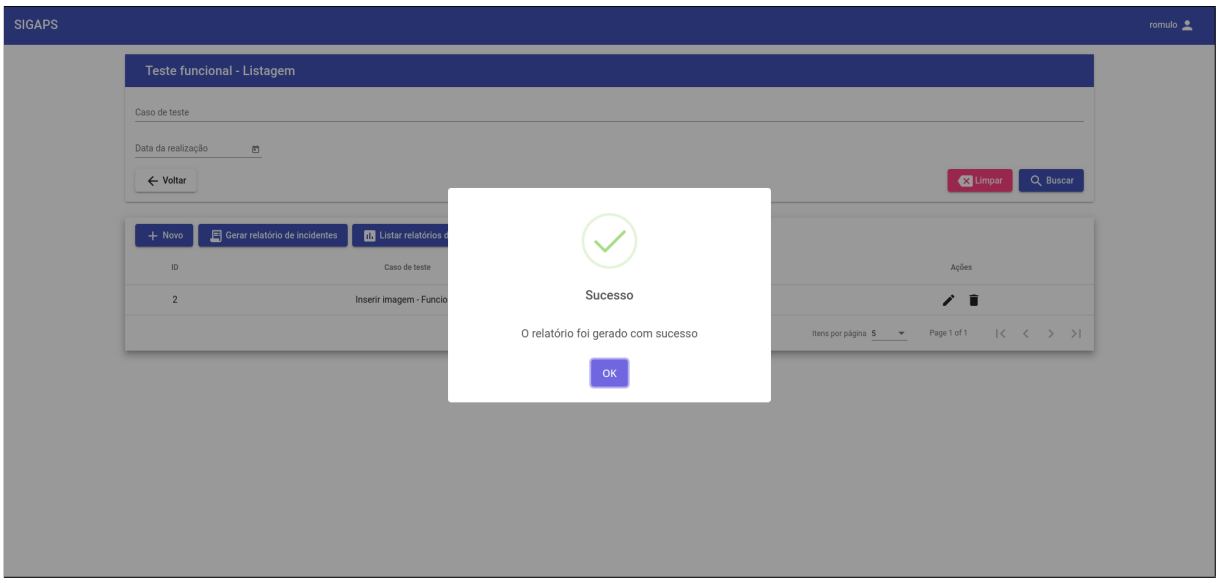
Figura 55: Listagem de testes funcionais



Fonte: Do autor

Para a geração dos relatórios o usuário deve clicar no botão "Gerar relatório de incidentes" e aguardar a mensagem de sucesso aparecer na tela, assim como mostra a figura 56.

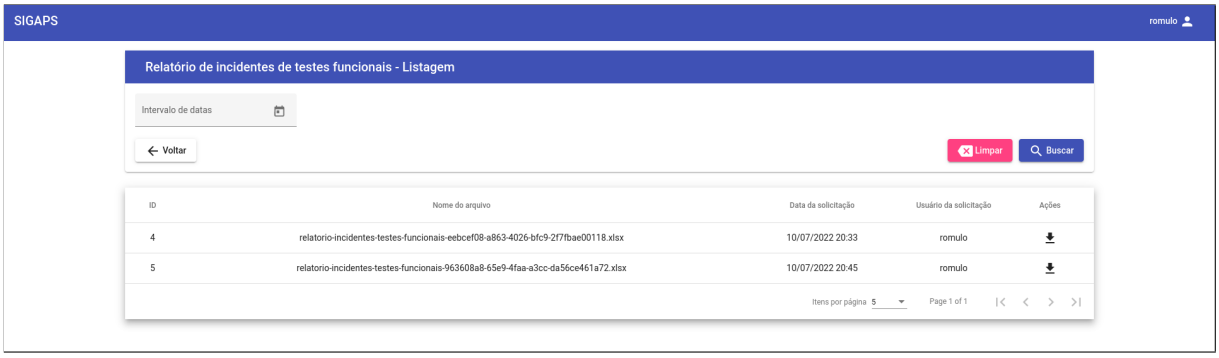
Figura 56: Geração de relatório de incidentes de testes funcionais



Fonte: Do autor

Para visualizar o relatório gerado, o usuário pode acessar a tela de listagem de relatório de incidentes presente na rota /lista-relatorios através do botão "Listar relatório de incidentes". Na tela de listagem de relatórios é possível consultar todos os relatórios de incidentes gerados e filtrar por um intervalo de datas de solicitação. Além disso, o usuário pode realizar *download* de cada um desses relatórios. A figura 57 mostra essa tela.

Figura 57: Listagem de relatório de incidentes de testes funcionais



Fonte: Do autor

Ao realizar o *download* o usuário recebe um arquivo .xlsx contendo as informações do relatório gerado.

Figura 58: Exemplo de relatório de incidentes de testes funcionais

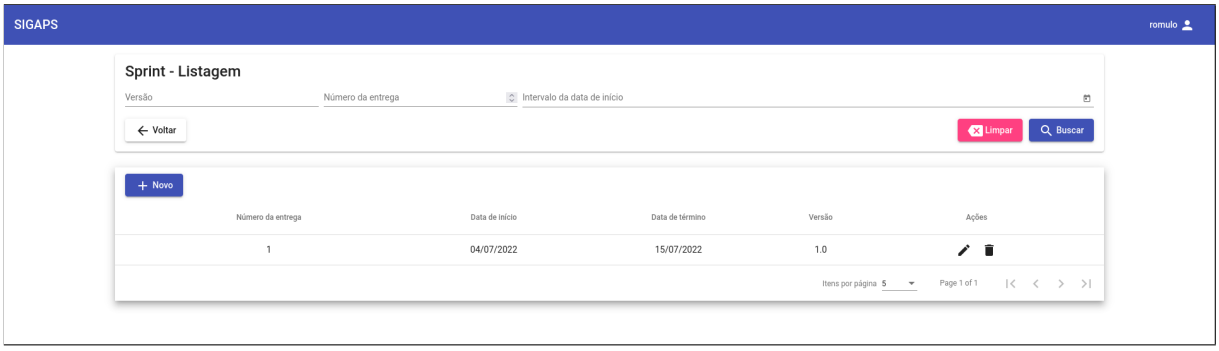
	A	B	C	D	E	F	G	H
1	ID	Caso de teste	Objetivos	Procedimentos	Ocorrência	Data de realização	Solução	Excluído?
2	2	Inserir imagem - Funcional	Verificar se a inserção de imagens está funcionando corretamente	1. Entrar na tela de listagem de imagens 2. Clicar no botão novo 3. Fazer upload da imagem 4. Informar uma descrição 5. Salvar	A imagem foi inserida corretamente	10/07/2022	Não foi necessário	Não
3								
4								
5								

Fonte: Do autor

4.4.8 Sprints

A tela de listagem de *sprints* está disponível na rota /lista-sprints e pode ser acessada pelo módulo "Método ágil" na tela de *overview* do projeto. Essa tela permite a busca paginada de *sprints* por versão, número de entrega e intervalo de data de início. A figura 59 demonstra essa tela.

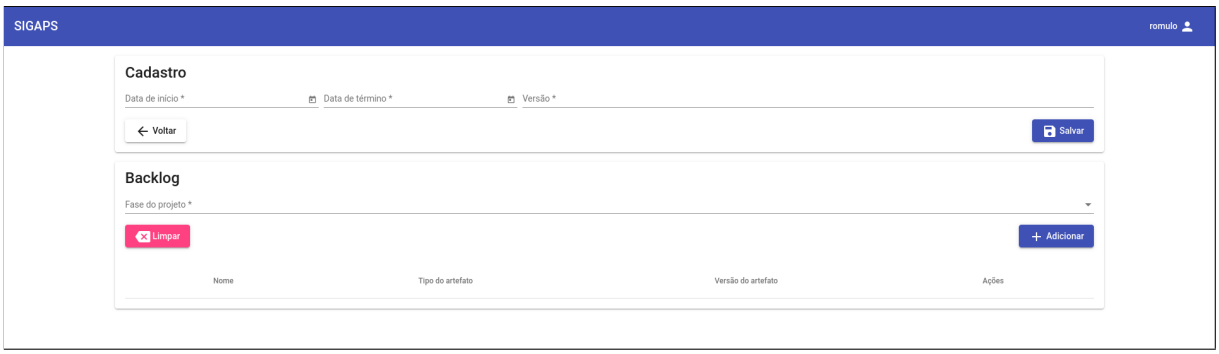
Figura 59: Tela de listagem de sprints



Fonte: Do autor

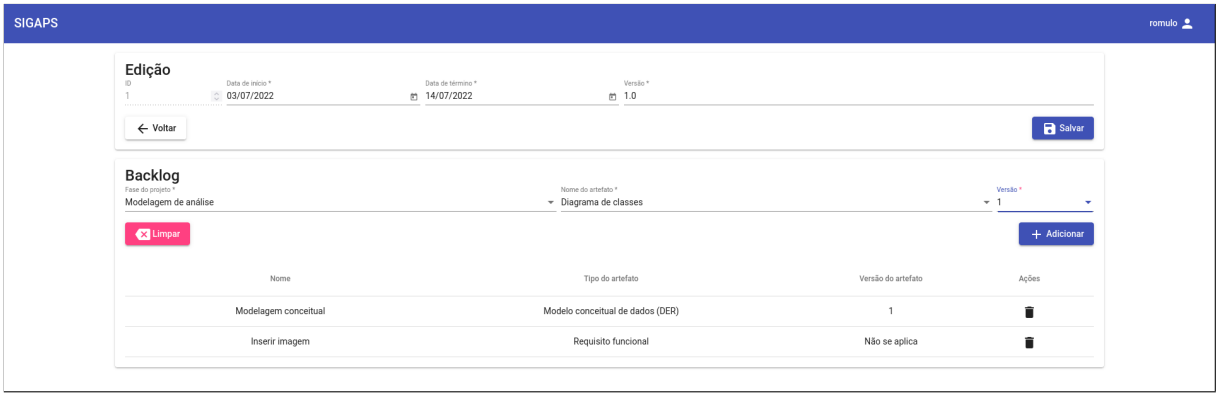
Como mostra a figura 59, na tela é possível adicionar, editar e excluir *sprints*. Ao excluir uma *sprint*, acontece uma exclusão lógica no banco de dados e o registro deixa de aparecer na listagem. Já quando o usuário clica no botão de editar, ele é direcionado para a mesma tela de cadastro de uma nova *sprint*, mas em modo de edição. As figuras 60 e 61 mostram as telas de cadastro e edição, ambas disponíveis na rota /form-sprint.

Figura 60: Tela de cadastro de sprints



Fonte: Do autor

Figura 61: Tela de edição de sprint



Fonte: Do autor

Como mostram as figuras 60 e 61, para toda *sprint* é obrigatório informar a data de início e termino, a versão e os itens do *backlog* daquela *sprint*. Para informar os itens do *backlog*, o usuário deve primeiro informar a fase do projeto em que está o artefato que ele deseja. Dessa forma, um *select* é carregado contendo todos os artefatos cadastrados naquela fase do projeto. Ao selecionar um desses artefatos, caso esse artefato tenha versionamento, um novo *select* é aberto para que o usuário selecione a versão do artefato. Caso não tenha versionamento, o *select* de versão não é carregado e o usuário já pode adicionar o artefato ao *backlog*. Um exemplo do formulário com um artefato que possui versionamento é exibido na figura 61.

5 TESTES E GARANTIA DE QUALIDADE

Devido a abordagem de desenvolvimento utilizada no trabalho, que foi o Scrum, o orientador atuou como PO. Dessa forma, os cenários de testes foram definidos com o PO e realizados pelo autor. As seções seguintes apresentam os testes realizados durante o projeto.

5.1 Teste unitário

Os testes unitários se concentram em cada unidade do *software*, componente ou classe por exemplo, conforme implementado no código-fonte. Dessa forma, esses testes utilizam intensamente técnicas de teste, com caminhos específicos na estrutura de controle dos componentes para que se garanta a cobertura completa e detecção máxima de erros (PRESSMAN; MAXIM, 2021). Prioritariamente, os métodos das classes devem ser testados individualmente. No entanto, em alguns casos é necessário realizar algumas sequências de teste (SOMMERVILLE, 2011).

Tendo em vista a importância dos testes unitários foram testadas as principais *services* do sistema, que são as classes responsáveis pela regra de negócio. As figuras 62 e 63 mostram um exemplo de teste unitário desenvolvido e as falhas encontradas por eles, respectivamente.

Figura 62: Exemplo de teste unitário

```
@DisplayName("Deve inserir artefato")
@Test
void deveInserirArtefato() {
    var artefatoSalvo = new Artefato();
    artefatoSalvo.setId(1L);

    var novoArtefatoDTO = criarNovoArtefatoDTO("Artefato 1", 1L, TipoArtefato.MODELO_CONCEITUAL_DADOS);

    when(artefatoRepository.save(any(Artefato.class))).thenReturn(artefatoSalvo);

    try {
        var artefatoSalvadoService = artefatoService.insert(novoArtefatoDTO);
        assertEquals(artefatoSalvadoService.getId(), artefatoSalvo.getId());
    } catch (Exception e) {
        fail();
    }
}
```

Fonte: Do autor

Figura 63: Resultados dos testes unitários

Classe	Data escrita teste	Data execução teste	Ocorrência	Solução
TesteService	18/07/2022	18/07/2022	Consulta utilizada para a tela de listagem de testes estava permitindo buscar sem informar o tipo do teste.	O campo de tipo do teste passou a ser obrigatório para a consulta.
ArtefatoService, EntidadeService, RequisitoService, SprintService, TesteService	14/07/2022	14/07/2022	As consultas de listagem paginada apresentavam erro ao tentar consultar sem informar a página inicial ou a quantidade de registros por página.	Quando não fornecidos, página inicial fica definida como 1 e quantidade de registros por página fica definida como 50.
TesteService	18/07/2022	18/07/2022	Estava sendo permitido salvar qualquer tipo de teste sem o campo “Solução” informado.	O campo de solução passou a ser obrigatório na inserção e atualização de testes.
TesteService	18/07/2022	18/07/2022	Estava sendo permitido salvar qualquer tipo de teste com o campo “Data de realização” com valor futuro.	Foi adicionada validação para permitir somente datas com valores até a data atual (inclusive).

Fonte: Do autor

5.2 Teste de desenvolvimento

Durante o desenvolvimento é comum que a equipe de desenvolvedores do sistema atue também com um papel de testador do *software*, onde o programador que desenvolveu realiza os chamados testes de desenvolvimento no sistema ou *feature* desenvolvida, visando descobrir *bugs* (SOMMERVILLE, 2011).

No sistema desenvolvido foram realizados testes de desenvolvimento durante todo o processo de construção do sistema, sempre verificando todas as funcionalidades já desenvolvidas. A figura 64 mostra os resultados desses testes.

Figura 64: Resultados testes de desenvolvimento

Falhas	Teste realizado	Motivo (erro)	Solução
Botão para incluir projetos estava aparecendo para todos os tipos de usuário	Listagem de projetos	Não estava sendo verificado se o usuário é gerente de projetos	Foi adicionada condição para o botão só aparecer para gerentes de projetos
Overview do projeto estava considerando registro excluído como cadastrado	Overview do projeto	Não estava sendo considerada a flag de excluído na consulta de artefatos cadastrados	Foi considerada a flag de excluído na consulta dos artefatos cadastrados
Tabelas compartilhando mesma sequence de ID	Inserção de entidades diferentes	Estratégia de geração que estava sendo usada era AUTO	Estratégia de geração alterada para IDENTITY
Todas versões do artefato estavam sendo geradas como primeira versão	Inserção de artefatos	Número da versão estava sempre sendo atribuído como um	Número da versão passou a ser o número da última versão do artefato mais um
Listagem de atributos de uma descrição de entidade do banco retornava erro inesperado	Edição de descrição de entidade do banco	Não havia construtor para a projeção na query	Criado construtor específico para o retorno das informações
Estava sendo permitido cadastrar testes com data de realização futura	Inserção de teste com data de realização futura	Não havia validação da data de realização	Adicionada validação para só permitir data de realização até a data atual
Telas de listagem não estavam com os botões de paginação habilitado	Listagem de artefatos	As consultas de listagem não estavam retornando a quantidade de registros	As consultas de listagem passaram a retornar também a quantidade de registros
Relatório de incidentes dos testes funcionais não baixava corretamente	Gerar e baixar um relatório de incidentes	O blob que estava sendo utilizado para baixar o relatório estava sendo convertido incorretamente	Conversão do blob foi implementada da forma correta
Etapas do projeto com somente um tipo de artefato não exibiam esse tipo de artefato	Overview de projeto	Uso da tela de overview do projeto	Montagem do vetor de tipos de artefato das fases estava cortando elementos em posição que não deveria ser considerada

Fonte: Do autor

5.3 Teste de usabilidade

Os testes de usabilidade são técnicas de caixa-preta utilizadas para descobrir problemas e pontos de melhoria ao observar o usuário utilizando o produto (ALURA, 2022). Dessa forma, esse tipo de teste é bom para verificar por exemplo se os filtros de busca

são fáceis de usar e se atendem o usuário (VOLPATO, 2014). Portanto, os testes de usabilidade ajudam a melhorar a experiência do usuário. Dessa forma, os testes de usabilidade foram realizados em conjunto com o PO e autor.

Com os testes de usabilidade foram verificados os layouts das telas, a navegabilidade, a disposição das informações e dos botões, além da comunicação dos botões entre as telas. Por fim, foi avaliada também a performance do sistema ao executar as ações do sistema. A figura 65 mostra os resultados desses testes.

Figura 65: Resultados testes de usabilidade

Falhas	Teste realizado	Motivo (erro)	Solução
As informações das tabelas não ficavam centralizadas na coluna	Listagem de artefatos	Não estava sendo aplicado estilo para centralizar as informações das colunas	Foi aplicado estilo para centralizar as informações da coluna
Header de listagem de projetos estava seno exibido mesmo sem projetos cadastrados	Listagem de projetos	Não estava sendo verificada a quantidade de projetos cadastrados para a exibição do header	A exibição do header de listagem de projetos passou a ser exibido somente quando há algum projeto cadastrado
Telas sem botão de voltar	Listagem de requisitos	Botões para voltar para tela anterior não tinham sido implementados	Todas as telas passaram a ter um botão de voltar
Tela de login e cadastro de usuário estavam sem estilo	Login	Não havia sido definido estilo e imagem de fundo para as telas de login e cadastre-se	Foi definido estilo e imagem de fundo para as telas de login e cadastre-se
Header dos cards estavam sem estilo, dificultando a diferenciação na tela	Tela de overview de projetos	Não havia sido definido estilo para o header dos cards	Foi definido o estilo do header do card sendo o mesmo estilo do toolbar
Visualização da prévia de artefatos estava com tamanho muito grande	Cadastro de artefato	Largura e altura da imagem de prévia estava sem valor definido	Foi definido um valor fixo para largura e altura da prévia do artefato
Não havia legenda para os ícones da tela de overview do projeto	Tela de overview de projetos	Não havia sido implementada legenda para os ícones da tela	Foi criado um botão no canto inferior direito da tela para apresentar a legenda
Data de realização exibida na tela de listagem de testes mostrava hora apesar do campo não possuir	Listagem de testes	A máscara para formatação de data estava incorreta	Foi utilizada a máscara correta para formatação da data

Fonte: Do autor

6 RESULTADOS ALCANÇADOS E CONSIDERAÇÕES FINAIS

Ao final deste trabalho foi concluído o desenvolvimento da aplicação *web* norteadora para o gerenciamento de artefatos de projeto de *software* dentro do cronograma sugerido. A aplicação torna mais acessível e flexível o gerenciamento de artefatos de *software* produzidos por aplicações externas, ao possibilitar que os gerentes de projetos e analistas de requisitos armazenem as diversas versões desses artefatos, permitindo uma visão mais ampla do processo de desenvolvimento de *software*. Dessa forma, os gerentes de projetos e analistas de requisitos podem armazenar os artefatos, como por exemplo: casos de uso, modelo conceitual de dados, diagrama de classes, diagrama de sequência, requisitos funcionais, testes, entre outros, de maneira fácil e organizada por etapa de desenvolvimento. Além disso, o sistema propicia informações detalhadas dessas etapas, de modo a ajudar nas tomadas de decisão.

O desenvolvimento do projeto foi de grande valor, uma vez que permitiu o estudo e execução das etapas de desenvolvimento de um *software*. Além de permitir a escolha de ferramentas, técnicas e tecnologias que foram utilizadas.

Para o desenvolvimento da aplicação foram utilizadas tecnologias que são comumente utilizadas no cenário atual do mercado de trabalho, mas não ofertadas na grade curricular do curso de Ciência da Computação, tal como Spring Boot, Angular, TypeScript, dentre outras. Dessa forma, o desenvolvimento da aplicação permitiu a busca de conhecimento e aprendizado por parte do autor, que procurou entender mais a fundo e se aperfeiçoar em tecnologias do mercado, além de se aperfeiçoar e praticar tecnologias que já possuía algum grau de afinidade. Tudo isso transformou o desenvolvimento em uma tarefa mais árdua mas que em seu final, após muito tempo dedicado e muita busca, foi convertida em conhecimento.

6.1 Melhorias e trabalhos futuros

Os cadastros dos artefatos das etapas de desenvolvimento de um projeto de *software* foram desenvolvidos de uma maneira mais abrangente. Dessa forma, os formulários de cadastro podem não conter todas as informações sugeridas por autores da literatura. Sendo assim, as funcionalidades do sistema podem ser complementadas, de modo a aumentar a qualidade da aplicação.

Dentre os requisitos funcionais levantados (seção 3.2.2) está o "RF09 - Realizar cálculo de estimativas", que foi adiado devido a complexidade para seu desenvolvimento, o que acarretaria em um atraso no cronograma e consequentemente na entrega da aplicação.

6.1.1 Usuário do sistema

Como sugestão para o gerenciamento dos usuários do sistema, sugere-se a criação de uma tela no sistema para que o usuário possa editar as suas informações. Dessa forma, será permitido ao usuário alterar seu login, email e senha. Além disso, sugere-se a implementação de recuperação de senha por e-mail.

6.1.2 Projeto de software

Como sugestão para o gerenciamento dos projetos de *software* cadastrados, sugere-se a criação de uma tela para a visualização do histórico de todas as ações realizadas dentro do projeto. Esse histórico já é gerado e alimentado pela aplicação, porém não há uma forma do usuário visualizá-la de forma direta, somente via consultas em banco de dados. Além disso, sugere-se implementar a funcionalidade para o gerente de projetos excluir um projeto de *software*.

6.1.3 Requisitos

Como sugestão para o cadastro de requisitos funcionais e não funcionais, sugere-se a adição de campos para registrar a prioridade, a situação, os clientes, usuários e outros envolvidos que participaram do levantamento do requisito em questão. Além disso, sugere-se registrar as restrições de domínio que se aplicam ao requisito e também permitir ao usuário manter uma matriz de rastreabilidade entre requisitos funcionais e

casos de uso, de forma a tornar mais fácil a visualização dos impactos ocorridos por conta de mudanças nos requisitos.

6.1.4 Descrição de tabelas do banco de dados

No formulário de descrição das tabelas do banco de dados o usuário pode inserir os atributos da tabela em questão. Para isso, ele deve fornecer, entre outros campos, o tipo do atributo que deseja inserir. Esse tipo do atributo é fornecido por meio de um componente de seleção, onde o usuário pode informar um valor dentre alguns valores pré-definidos. Os valores presentes na versão desenvolvida da aplicação contemplam os tipos de atributos mais usuais no dia a dia, mas podem não ser suficientes para tabelas que guardem valores não tão usuais. Dessa forma, sugere-se a expansão da lista de tipos de atributo, de modo a contemplar mais casos, tornando assim a funcionalidade mais completa.

6.1.5 Análise de riscos

A funcionalidade de análise de riscos implementada permite que o usuário mantenha versões de sua análise de risco realizada em uma ferramenta externa. Uma possível melhoria nessa funcionalidade seria realizar a análise de riscos dentro da própria aplicação, permitindo que o usuário mantenha informações de risco como por exemplo: descrição, probabilidade, impacto, etapas de mitigação, contingência, estado atual, entre outras.

6.1.6 Método ágil

O módulo de método ágil desenvolvido possui a funcionalidade para manter *sprints*, que é uma característica compartilhada entre algumas metodologias ágeis. Contudo, seria interessante a implementação de funcionalidades específicas para o método ágil utilizado no desenvolvimento do projeto, de modo a contemplar características inerentes a outros métodos ágeis, como Scrum, XP, Kanban, dentre outros.

6.1.7 Relatórios

Ao solicitar a geração de um relatório, a aplicação *frontend* fica em um estado de espera, aguardando o servidor *backend* finalizar o processamento desse relatório. Desse modo, sugere-se a utilização de uma técnica para tornar o processamento do relatório em uma tarefa assíncrona, como por exemplo a utilização de fila. Dessa maneira, ao solicitar a geração de um relatório a aplicação envia a solicitação para uma fila e a aplicação *frontend* do usuário fica livre para que este realize outras tarefas, voltando a alertá-lo quando seu relatório for processado.

6.1.8 Testes

Como sugestão para o módulo de testes do sistema, sugere-se a avaliação da inclusão de um campo para anexar um relatório no formulário de cadastro de um teste. Dessa forma, o usuário poderá fornecer um relatório gerado pela ferramenta utilizada na realização dos testes.

6.1.9 Garantia de qualidade

Os testes realizados no sistema foram realizados com um caráter de desenvolvimento. Sendo assim, não foi realizada uma testagem do sistema com o público-alvo. Dessa forma sugere-se a realização desses testes, de forma a levantar melhorias e aumentar a qualidade do sistema.

De forma parecida, os testes unitários realizados no sistema se ativeram às *services* que desempenham papéis mais importantes no sistema. Por este motivo, seria interessante expandir a cobertura desses testes para mais classes e métodos, ultrapassando a camada *service* e abrangendo também a de *façade* e *repository*.

6.1.10 Integração com plataformas de hospedagem de código fonte

A funcionalidade para manter código fonte desenvolvida foi feita de forma a permitir que usuário realize o *upload* de versões do código fonte do sistema, seja por meio de arquivos fonte ou arquivos compactados. No entanto, algumas ferramentas existentes, como o Git, são comumente adotadas para essa tarefa. Dessa forma, seria interessante a integração do sistema proposto com plataformas de hospedagem de arquivos

com controle de versão, como o GitHub e GitLab.

REFERÊNCIAS

ABES. *Mercado Brasileiro de Software*: Panorama e tendências. [S.l.], 2021. Citado na página 15.

ALMEIDA, V. C. D. *Uso da linguagem OCL no contexto de diagramas de classe da UML e programas em Java*. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, Belo Horizonte, MG, jul. 2007. Citado na página 42.

ALURA. *Testes de usabilidade*. 2022. <https://www.alura.com.br/apostila-ux-usabilidade-mobile-web/usabilidade>. Acessado em: 11 jul. 2022. Citado na página 82.

Angular. *What is Angular?* 2021. <https://angular.io/guide/what-is-angular>. Acessado em: 13 fev. 2022. Citado na página 26.

ARAUJO, L. S. d. *Gerenciamento de Projetos de Software com PMBOK e SCRUM: um estudo e análise comparativa do método tradicional e ágil*. Monografia (TCC) — Universidade Federal Fluminense, Niterói-RJ, 2017. Citado na página 16.

ATLASSIAN. *Jira*. 2022. <https://www.atlassian.com/software/jira>. Acessado em: 17 ago. 2022. Citado 2 vezes nas páginas 20 e 21.

DANTAS, C. M.; CÓRDULA, F. R.; ARAÚJO, W. J. Análise da representação da informação em modelos entidade relacionamento com base em metadados. *Archeion Online*, v. 4, n. 1, p. 40–63, dez. 2016. Disponível em: <https://periodicos3.ufpb.br/index.php/archeion/article/view/32301>. Citado na página 43.

ESPINDOLA, R.; MAJDENBAUM, A.; AUDY, J. Uma análise crítica dos desafios para engenharia de requisitos em manutenção de software. In: . [S.l.: s.n.], 2004. p. 226–238. Citado na página 15.

ESPINHA, R. G. *9 Vantagens de utilizar um sistema web de gestão de atividades*. 2022. <https://artia.com/blog/9-vantagens-de-utilizar-um-sistema-web-de-gestao-de-atividades/>. Acessado em: 15 ago. 2022. Citado na página 17.

EVANS, E. *Domain-Driven Design*. [S.l.]: Alta Books, 2016. Citado na página 51.

FERREIRA, W. O.; KNOP, I. de O. Estruturação de aplicações distribuídas com a arquitetura rest. *Caderno de Estudos em Sistemas de Informação*, v. 3, n. 1, 2017. Citado na página 23.

FIELDING, R. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado), 01 2000. Citado 2 vezes nas páginas 23 e 24.

GAMMA, E. et al. *Padrões de Projeto*. [S.l.]: Bookman, 2000. Citado na página 50.

GONÇALVES, R. Q. *ENSINO DE GERENCIAMENTO DE PROJETOS DE SOFTWARE MEDIADO POR FERRAMENTAS*. Tese (Doutorado) — Universidade Federal de Santa Catarina, Florianópolis-SC, 2017. Citado na página 16.

IBGE. *Contas Nacionais Trimestrais*: Indicadores de volume e valores correntes. [S.l.], 2021. Citado na página 15.

IGTI. *Arquitetura de software: sua definição e aplicação nos negócios*. 2019. <https://www.igti.com.br/blog/arquitetura-de-software-definicao-negocios>. Acessado em: 12 dez. 2021. Citado na página 22.

KOSTALOVA, J.; TETREVOVA, L.; SVEDIK, J. Support of project management methods by project management information system. *Procedia - Social and Behavioral Sciences*, v. 210, p. 96–104, 2015. ISSN 1877-0428. Proceedings of the 4th International Conference on Leadership, Technology, Innovation and Business Management (ICLTIBM-2014). Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877042815056803>. Citado na página 17.

KRAFZIG, D.; BANKE, K.; SLAMA, D. *Enterprise SOA*: Service-oriented architecture best practices. Indianapolis: Prentice Hall, 2004. Citado na página 22.

ORACLE. *General Information*. 2022. <https://dev.mysql.com/doc/workbench/en/wb-intro.html>. Acessado em: 20 jul. 2022. Citado na página 47.

PADUELLI, M. M. *Manutenção de Software: problemas típicos e diretrizes para uma disciplina específica*. Dissertação (Mestrado) — Universidade de São Paulo, São Carlos, SP, jun. 2007. Citado na página 16.

PARASCHIV, E. *How Spring Web MVC Really Works*. 2017. <https://stackify.com/spring-mvc/>. Acessado em: 11 jul. 2022. Citado na página 49.

PMI. *Guia PMBOK®*: Um guia para o conjunto de conhecimentos em gerenciamento de projetos. Pennsylvania: PMI, 2017. Citado na página 16.

PostgreSQL. *What is PostgreSQL?* 2022. <https://www.postgresql.org/about/>. Acessado em: 13 fev. 2022. Citado na página 28.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de Software*: Uma abordagem profissional. Porto Alegre: AMGH Editora Ltda., 2021. Citado 6 vezes nas páginas 22, 31, 36, 45, 67 e 80.

RedHat. *O que é API?* 2017. <https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces>. Acessado em: 12 dez. 2021. Citado na página 23.

SCHILDT, H. *Java the complete reference*. New York: McGraw-Hill Education, 2019. Citado 2 vezes nas páginas 27 e 28.

SCRUM. *What is Scrum?* 2020. <https://www.scrum.org/resources/what-is-scrum>. Acessado em: 15 ago. 2022. Citado na página 30.

SOMMERVILLE, I. *Engenharia de Software*. São Paulo: Pearson Prentice Hall, 2011. Citado 8 vezes nas páginas 22, 30, 31, 32, 33, 42, 80 e 82.

SPRING. *Working with Spring Data Repositories*. 2022. <https://docs.spring.io/spring-data/jpa/docs/1.6.0.RELEASE/reference/html/repositories.html#repositories.create-instances>. Acessado em: 09 jul. 2022. Citado na página 49.

TRELLO. *Gerencie os projetos do time em qualquer lugar*. 2022. <https://trello.com/home>. Acessado em: 17 ago. 2022. Citado 2 vezes nas páginas 19 e 20.

TypeScript. *What is TypeScript?* 2022. <https://www.typescriptlang.org/>. Acessado em: 12 fev. 2022. Citado na página 26.

VALENTE, M. T. *Engenharia de Software Moderna*. [S.l.]: Independente, 2020. Citado na página 15.

VOLPATO, E. *Teste de usabilidade: o que é e para que serve?* 2014. <https://brasil.uxdesign.cc/teste-de-usabilidade-o-que-%C3%A9-e-para-que-serve-de3622e4298b>. Acessado em: 11 jul. 2022. Citado na página 83.

W3Schools. *What is JSON?* 2022. <https://www.w3schools.com/whatis/whatis_json.asp>. Acessado em: 13 fev. 2022. Citado na página 27.

APÊNDICE A – CENÁRIOS DE CASO DE USO

UC2 - Manter projetos

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Não há.

Fluxo principal:

1. [RS] O sistema exibe a tela com os projetos de software cadastrados. [FA1] [FA2]

Fluxo alternativo:

Não há.

Fluxo de exceção:

Não há.

Pós-condição:

Os projetos cadastrados são exibidos.

Mensagens:

Não há

UC3 - Liberar módulos

Ator(es): Gerente de projetos

Pré-condição:

Pelo menos um projeto de software cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os projetos de software cadastrados.
2. [EV] O ator seleciona um projeto de software na listagem.
3. [RS] O sistema exibe as informações detalhadas do projeto de software.
4. [EV] O ator seleciona a opção de liberar o módulo desejado.
5. [RS] O sistema solicita a confirmação da liberação.
6. [EV] O ator confirma a liberação.
7. [RS] O sistema exibe a mensagem [MSG001].
8. [RS] O sistema atualiza as informações detalhadas do projeto de software.

Fluxo alternativo:

Não há.

Fluxo de exceção:

Não há.

Pós-condição:

Um novo módulo do projeto de software é liberado.

Mensagens:

MSG001 - Etapa liberada com sucesso!

UC4 - Manter requisito funcional e não funcional

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo de "Especificação de requisitos" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um requisito funcional ou não funcional cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os requisitos cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar requisito funcional ou não funcional

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [RS] O sistema carrega os tipos de requisito na seleção.
4. [EV] O ator informa o nome do requisito.
5. [EV] O ator informa o tipo do requisito.
6. [EV] O ator informa a descrição do requisito.
7. [EV] O ator confirma o cadastro.
8. [RS] O sistema exibe a mensagem [MSG001].
9. [RS] O sistema atualiza a listagem de requisitos.

FA2 - Editar requisito funcional ou não funcional

1. [EV] O ator seleciona um requisito funcional ou não funcional na listagem.
2. [RS] O sistema exibe as informações detalhadas do requisito.
3. [EV] O ator altera o nome do requisito.
4. [EV] O ator altera o tipo do requisito.
5. [EV] O ator altera a descrição do requisito.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].

FA3 - Excluir requisito funcional ou não funcional

1. [EV] O ator seleciona um requisito funcional ou não funcional na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.

4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de requisitos.

FA4 - Buscar requisito funcional ou não funcional

1. [EV] O ator pode informar qualquer um dos filtros: nome do requisito; tipo do requisito.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de requisitos com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo requisito é inserido na listagem de requisitos.

FA2 - Um requisito terá seu registro alterado na listagem de requisitos.

FA3 - Um requisito terá seu registro removido na listagem de requisitos.

FA4 - Os requisitos encontrados são exibidos na listagem de requisitos.

Mensagens:

MSG001 - Requisito cadastrado com sucesso

MSG002 - Requisito atualizado com sucesso

MSG003 - Requisito deletado com sucesso

UC5 - Manter modelo conceitual de dados

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo de "Especificação de requisitos" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um modelo conceitual de dados cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os modelos conceituais de dados cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar modelo conceitual de dados

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição do modelo.
4. [EV] O ator informa o arquivo de prévia do modelo.
5. [EV] O ator informa o arquivo base do modelo.
6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].
8. [RS] O sistema atualiza a listagem de modelos.

FA2 - Editar modelo conceitual de dados

1. [EV] O ator seleciona um modelo conceitual de dados na listagem.
2. [RS] O sistema exibe as informações detalhadas do modelo.
3. [EV] O ator altera a descrição do modelo.
4. [EV] O ator altera o arquivo de prévia do modelo.
5. [EV] O ator altera o arquivo base do modelo.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de versões do modelo.

FA3 - Excluir modelo conceitual de dados

1. [EV] O ator seleciona um modelo conceitual de dados na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de modelos.

FA4 - Buscar modelo conceitual de dados

1. [EV] O ator pode informar qualquer um dos filtros: descrição do modelo conceitual de dados; data da última versão do modelo conceitual de dados.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de modelos com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo modelo é inserido na listagem de modelos conceituais de dados.

FA2 - Um modelo terá seu registro alterado na listagem de modelos conceituais de dados.

FA3 - Um modelo terá seu registro removido na listagem de modelos conceituais de dados.

FA4 - Os modelos encontrados são exibidos na listagem de modelos conceituais de dados.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC6 - Manter casos de uso

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo de "Modelagem de análise" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um caso de uso cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os casos de uso cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar caso de uso

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição do caso de uso.
4. [EV] O ator informa o arquivo de prévia do caso de uso.
5. [EV] O ator informa o arquivo base do caso de uso.
6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].
8. [RS] O sistema atualiza a listagem de casos de uso.

FA2 - Editar caso de uso

1. [EV] O ator seleciona um caso de uso na listagem.
2. [RS] O sistema exibe as informações detalhadas do caso de uso.

3. [EV] O ator altera a descrição do caso de uso.
4. [EV] O ator altera o arquivo de prévia do caso de uso.
5. [EV] O ator altera o arquivo base do caso de uso.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de versões do caso de uso.

FA3 - Excluir caso de uso

1. [EV] O ator seleciona um caso de uso na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de casos de uso.

FA4 - Buscar casos de uso

1. [EV] O ator pode informar qualquer um dos filtros: descrição do caso de uso; data da última versão do caso de uso.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de casos de uso com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo caso de uso é inserido na listagem de casos de uso.

FA2 - Um caso de uso terá seu registro alterado na listagem de casos de uso.

FA3 - Um caso de uso terá seu registro removido na listagem de casos de uso.

FA4 - Os casos de uso encontrados são exibidos na listagem de casos de uso.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC7 - Manter diagrama de classes

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo de "Modelagem de análise" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um diagrama de classes cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os diagramas de classes cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar diagrama de classes

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição do diagrama.
4. [EV] O ator informa o arquivo de prévia do diagrama.
5. [EV] O ator informa o arquivo base do diagrama.
6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].

8. [RS] O sistema atualiza a listagem de diagramas de classes.

FA2 - Editar diagrama de classes

1. [EV] O ator seleciona um diagrama de classes na listagem.
2. [RS] O sistema exibe as informações detalhadas do diagrama.
3. [EV] O ator altera a descrição do diagrama.
4. [EV] O ator altera o arquivo de prévia do diagrama.
5. [EV] O ator altera o arquivo base do diagrama.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de versões do diagrama.

FA3 - Excluir diagrama de classes

1. [EV] O ator seleciona um diagrama de classes na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de diagramas.

FA4 - Buscar diagrama de classes

1. [EV] O ator pode informar qualquer um dos filtros: descrição do diagrama de classes; data da última versão do diagrama de classes.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de diagramas com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo diagrama é inserido na listagem de diagramas de classes.

FA2 - Um diagrama terá seu registro alterado na listagem de diagramas de classes.

FA3 - Um diagrama terá seu registro removido na listagem de diagramas de classes.

FA4 - Os diagramas encontrados são exibidos na listagem de diagramas de classes.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC8 - Manter diagrama de sequência

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo de "Modelagem de análise" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um diagrama de sequência cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os diagramas de sequência cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar diagrama de sequência

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição do diagrama.

4. [EV] O ator informa o arquivo de prévia do diagrama.
5. [EV] O ator informa o arquivo base do diagrama.
6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].
8. [RS] O sistema atualiza a listagem de diagramas de sequência.

FA2 - Editar diagrama de sequência

1. [EV] O ator seleciona um diagrama de sequência na listagem.
2. [RS] O sistema exibe as informações detalhadas do diagrama.
3. [EV] O ator altera a descrição do diagrama.
4. [EV] O ator altera o arquivo de prévia do diagrama.
5. [EV] O ator altera o arquivo base do diagrama.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de versões do diagrama.

FA3 - Excluir diagrama de sequência

1. [EV] O ator seleciona um diagrama de sequência na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de diagramas de sequência.

FA4 - Buscar diagrama de sequência

1. [EV] O ator pode informar qualquer um dos filtros: descrição do diagrama de sequência; data da última versão do diagrama de sequência.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de diagramas com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo diagrama é inserido na listagem de diagramas de sequência.

FA2 - Um diagrama terá seu registro alterado na listagem de diagramas de sequência.

FA3 - Um diagrama terá seu registro removido na listagem de diagramas de sequência.

FA4 - Os diagramas encontrados são exibidos na listagem de diagramas de sequência.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC9 - Manter modelo lógico de dados

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo de "Modelagem de análise" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um modelo lógico de dados cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os modelos lógicos de dados cadastrados. [FA1]

[FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar modelo lógico de dados

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição do modelo.
4. [EV] O ator informa o arquivo de prévia do modelo.
5. [EV] O ator informa o arquivo base do modelo.
6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].
8. [RS] O sistema atualiza a listagem de modelos lógicos de dados.

FA2 - Editar modelo lógico de dados

1. [EV] O ator seleciona um modelo lógico de dados na listagem.
2. [RS] O sistema exibe as informações detalhadas do modelo.
3. [EV] O ator altera a descrição do modelo.
4. [EV] O ator altera o arquivo de prévia do modelo.
5. [EV] O ator altera o arquivo base do modelo.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de versões do modelo.

FA3 - Excluir modelo lógico de dados

1. [EV] O ator seleciona um modelo lógico de dados na listagem.
2. [EV] O ator seleciona a opção excluir.

3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de modelos lógicos de dados.

FA4 - Buscar modelo lógico de dados

1. [EV] O ator pode informar qualquer um dos filtros: descrição do modelo lógico dados; data da última versão do modelo lógico de dados.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de modelos com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo modelo é inserido na listagem de modelos lógicos de dados.

FA2 - Um modelo terá seu registro alterado na listagem de modelos lógicos de dados.

FA3 - Um modelo terá seu registro removido na listagem de modelos lógicos de dados.

FA4 - Os modelos encontrados são exibidos na listagem de modelos lógicos de dados.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC10 - Manter gráfico de Gantt

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo de "Monitoramento e controle" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um gráfico de Gantt cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os gráficos de Gantt cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar gráfico de Gantt

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição do gráfico.
4. [EV] O ator informa o arquivo de prévia do gráfico.
5. [EV] O ator informa o arquivo base do gráfico.
6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].
8. [RS] O sistema atualiza a listagem de gráficos de Gantt.

FA2 - Editar gráfico de Gantt

1. [EV] O ator seleciona um gráfico de Gantt na listagem.
2. [RS] O sistema exibe as informações detalhadas do gráfico.
3. [EV] O ator altera a descrição do gráfico.
4. [EV] O ator altera o arquivo de prévia do gráfico.
5. [EV] O ator altera o arquivo base do gráfico.
6. [EV] O ator confirma a edição.

7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de versões do gráfico.

FA3 - Excluir gráfico de Gantt

1. [EV] O ator seleciona um gráfico de Gantt na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de gráficos de Gantt.

FA4 - Buscar gráfico de Gantt

1. [EV] O ator pode informar qualquer um dos filtros: descrição do gráfico de Gantt; data da última versão do gráfico de Gantt.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de gráficos com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo gráfico é inserido na listagem de gráficos de Gantt.

FA2 - Um gráfico terá seu registro alterado na listagem de gráficos de Gantt.

FA3 - Um gráfico terá seu registro removido na listagem de gráficos de Gantt.

FA4 - Os gráficos encontrados são exibidos na listagem de gráficos de Gantt.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC11 - Manter protótipo

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Implementação" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um protótipo cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os protótipos cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar protótipo

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição do protótipo.
4. [EV] O ator informa o arquivo de prévia do protótipo.
5. [EV] O ator informa o arquivo base do protótipo.
6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].
8. [RS] O sistema atualiza a listagem de protótipos.

FA2 - Editar protótipo

1. [EV] O ator seleciona um protótipo na listagem.
2. [RS] O sistema exibe as informações detalhadas do protótipo.

3. [EV] O ator altera a descrição do protótipo.
4. [EV] O ator altera o arquivo de prévia do protótipo.
5. [EV] O ator altera o arquivo base do protótipo.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de protótipos.

FA3 - Excluir protótipo

1. [EV] O ator seleciona um protótipo na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de protótipos.

FA4 - Buscar protótipo

1. [EV] O ator pode informar qualquer um dos filtros: descrição do protótipo; data da última versão do protótipo.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de protótipos com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo protótipo é inserido na listagem de protótipos.

FA2 - Um protótipo terá seu registro alterado na listagem de protótipos.

FA3 - Um protótipo terá seu registro removido na listagem de protótipos.

FA4 - Os protótipos encontrados são exibidos na listagem de protótipos.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC12 - Manter código do projeto de software

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Implementação" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um código do projeto de software cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os códigos do projeto de software cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar código do projeto de software

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição do código.
4. [EV] O ator informa o arquivo de prévia do código.
5. [EV] O ator informa o arquivo base do código.
6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].
8. [RS] O sistema atualiza a listagem de códigos do projeto de software.

FA2 - Editar código do projeto de software

1. [EV] O ator seleciona um código do projeto de software na listagem.
2. [RS] O sistema exibe as informações detalhadas do código.
3. [EV] O ator altera a descrição do código.
4. [EV] O ator altera o arquivo de prévia do código.
5. [EV] O ator altera o arquivo base do código.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de versões do código.

FA3 - Excluir código do projeto de software

1. [EV] O ator seleciona um código do projeto de software na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de códigos do projeto de software.

FA4 - Buscar código do projeto de software

1. [EV] O ator pode informar qualquer um dos filtros: descrição do código do projeto de software; data da última versão do código do projeto de software.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de modelos com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo código é inserido na listagem de códigos do projeto de software.

FA2 - Um código terá seu registro alterado na listagem de códigos do projeto de software.

FA3 - Um código terá seu registro removido na listagem de códigos do projeto de software.

FA4 - Os códigos encontrados são exibidos na listagem de códigos do projeto de software.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC13 - Manter modelo físico de dados

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Implementação" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um modelo físico de dados cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os modelos físicos de dados cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar modelo físico de dados

1. [EV] O ator seleciona a opção novo.

2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição do modelo.
4. [EV] O ator informa o arquivo de prévia do modelo.
5. [EV] O ator informa o arquivo base do modelo.
6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].
8. [RS] O sistema atualiza a listagem de modelos físicos de dados.

FA2 - Editar modelo físico de dados

1. [EV] O ator seleciona um modelo físico de dados na listagem.
2. [RS] O sistema exibe as informações detalhadas do modelo.
3. [EV] O ator altera a descrição do modelo.
4. [EV] O ator altera o arquivo de prévia do modelo.
5. [EV] O ator altera o arquivo base do modelo.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de versões do modelo.

FA3 - Excluir modelo físico de dados

1. [EV] O ator seleciona um modelo físico de dados na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de modelos físicos de dados.

FA4 - Buscar modelo físico de dados

1. [EV] O ator pode informar qualquer um dos filtros: descrição do modelo físico de dados; data da última versão do modelo físico de dados.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de modelos com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo modelo é inserido na listagem de modelos físicos de dados.

FA2 - Um modelo terá seu registro alterado na listagem de modelos físicos de dados.

FA3 - Um modelo terá seu registro removido na listagem de modelos físicos de dados.

FA4 - Os modelos encontrados são exibidos na listagem de modelos físicos de dados.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC14 - Manter descrição de tabelas do banco de dados

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Implementação" liberado.

FA1, FA4, FA5, FA6 - Não há.

FA2, FA3 - Pelo menos uma descrição de tabelas do banco de dados.

Fluxo principal:

1. [RS] O sistema exibe a tela com as descrições de tabelas do banco de dados. [FA1] [FA2] [FA3] [FA4] [FA5]

Fluxo alternativo:

FA1 - Cadastrar descrição de tabela do banco de dados

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa o nome da tabela.
4. [EV] O ator informa a descrição da tabela.
5. [EV] O ator confirma o cadastro.
6. [RS] O sistema exibe a mensagem [MSG001].
7. [RS] O sistema habilita o formulário de cadastro de atributos.
8. [RS] O sistema inicia o FA2 na etapa 2.

FA2 - Editar descrição de tabela do banco de dados

1. [EV] O ator seleciona uma descrição de tabela do banco de dados na listagem.
2. [RS] O sistema exibe as informações detalhadas da tabela.
3. [EV] O ator altera o nome da tabela.
4. [EV] O ator altera a descrição da tabela.
5. [EV] O ator altera os atributos da tabela. [FE1]
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].

FA3 - Excluir descrição de tabela do banco de dados

1. [EV] O ator seleciona uma descrição de tabela do banco de dados na listagem.
2. [EV] O ator seleciona a opção excluir.

3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de descrições de tabela do banco de dados.

FA4 - Buscar descrição de tabela do banco de dados

1. [EV] O ator pode informar o nome da tabela.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de descrições de tabela do banco de dados com os resultados encontrados.

FA5 - Baixar relatório de descrições de tabelas do banco de dados

1. [EV] O ator seleciona a opção listar relatórios.
2. [RS] O sistema exibe a listagem com todos os relatórios gerados. [FA6]
3. [EV] O ator seleciona um relatório.
4. [EV] O ator confirma o download.
5. [RS] O sistema baixa o relatório escolhido.

FA6 - Buscar relatórios de descrições de tabelas do banco de dados gerados

1. [EV] O ator informa um intervalo de datas de solicitação.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de relatórios com os resultados encontrados.

Fluxo de exceção:

FE1 - O ator informa um nome de atributo já utilizado

1. [RS] O sistema exibe a mensagem [MSG004].
2. Retorna ao passo 5 do FA2.

Pós-condição:

FA1 - Uma nova tabela é inserida na listagem de descrições de tabela do banco de dados.

FA2 - Uma tabela terá seu registro alterado na listagem de descrições de tabela do banco de dados.

FA3 - Uma tabela terá seu registro removido na listagem de descrições de tabela do banco de dados.

FA4 - As tabelas encontradas são exibidas na listagem de descrições de tabela do banco de dados.

FA5 - Uma planilha do formato xlsx é baixada no navegador do ator.

FA6 - Os relatórios encontrados são exibidos.

Mensagens:

MSG001 - Entidade cadastrada com sucesso. Siga com o formulário abaixo para fornecer os atributos

MSG002 - Entidade atualizada com sucesso

MSG003 - Entidade deletada com sucesso

MSG004 - Já existe um atributo com esse nome

UC15 - Gerar relatório de descrições de tabelas do banco de dados

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Módulo "Implementação" liberado.

Fluxo principal:

1. [RS] O sistema exibe a tela com as tabelas cadastradas.
2. [EV] O ator seleciona a opção de gerar relatório.
3. [RS] O sistema exibe a mensagem [MSG001].

Fluxo de exceção:

Não há.

Pós-condição:

Uma planilha do formato xlsx contendo todas descrições de tabelas do banco de dados cadastradas é gerada.

Mensagens:

MSG001 - O relatório foi gerado com sucesso

UC16 - Manter modelo de arquitetura

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Implementação" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um modelo de arquitetura cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os modelos de arquitetura cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar modelo de arquitetura

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição do modelo.
4. [EV] O ator informa o arquivo de prévia do modelo.
5. [EV] O ator informa o arquivo base do modelo.
6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].

8. [RS] O sistema atualiza a listagem de modelos de arquitetura.

FA2 - Editar modelo de arquitetura

1. [EV] O ator seleciona um modelo de arquitetura na listagem.
2. [RS] O sistema exibe as informações detalhadas do modelo.
3. [EV] O ator altera a descrição do modelo.
4. [EV] O ator altera o arquivo de prévia do modelo.
5. [EV] O ator altera o arquivo base do modelo.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de versões do modelo.

FA3 - Excluir modelo de arquitetura

1. [EV] O ator seleciona um modelo de arquitetura na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de modelos de arquitetura.

FA4 - Buscar modelo de arquitetura

1. [EV] O ator pode informar qualquer um dos filtros: descrição do modelo de arquitetura; data da última versão do modelo de arquitetura.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de modelos com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo modelo é inserido na listagem de modelos de arquitetura.

FA2 - Um modelo terá seu registro alterado na listagem de modelos de arquitetura.

FA3 - Um modelo terá seu registro removido na listagem de modelos de arquitetura.

FA4 - Os modelos encontrados são exibidos na listagem de modelos de arquitetura.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC17 - Manter teste unitário

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Testes" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um teste unitário cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os testes unitários cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar teste unitário

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa o caso de teste.

4. [EV] O ator informa a data de realização.
5. [EV] O ator informa os objetivos.
6. [EV] O ator informa os procedimentos.
7. [EV] O ator informa a ocorrência.
8. [EV] O ator informa a solução.
9. [EV] O ator confirma o cadastro.
10. [RS] O sistema exibe a mensagem [MSG001].
11. [RS] O sistema atualiza a listagem de testes unitários.

FA2 - Editar teste unitário

1. [EV] O ator seleciona um teste unitário na listagem.
2. [RS] O sistema exibe as informações detalhadas do teste.
3. [EV] O ator altera o caso de teste.
4. [EV] O ator altera a data de realização.
5. [EV] O ator altera os objetivos.
6. [EV] O ator altera os procedimentos.
7. [EV] O ator altera a ocorrência.
8. [EV] O ator altera a solução.
9. [EV] O ator confirma a edição.
10. [RS] O sistema exibe a mensagem [MSG002].
11. [RS] O sistema atualiza a listagem de testes unitários.

FA3 - Excluir teste unitário

1. [EV] O ator seleciona um teste unitário na listagem.
2. [EV] O ator seleciona a opção excluir.

3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de testes unitários.

FA4 - Buscar teste unitário

1. [EV] O ator pode informar qualquer um dos filtros: caso de teste; data de realização do teste.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de testes com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo teste é inserido na listagem de testes unitários.

FA2 - Um teste terá seu registro alterado na listagem de testes unitários.

FA3 - Um teste terá seu registro removido na listagem de testes unitários.

FA4 - Os testes encontrados são exibidos na listagem de testes unitários.

Mensagens:

MSG001 - Teste cadastrado com sucesso

MSG002 - Teste atualizado com sucesso

MSG003 - Teste deletado com sucesso

UC18 - Manter teste de cobertura

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Testes" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um teste de cobertura cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os testes de cobertura cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar teste de cobertura

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa o caso de teste.
4. [EV] O ator informa a data de realização.
5. [EV] O ator informa os objetivos.
6. [EV] O ator informa os procedimentos.
7. [EV] O ator informa a ocorrência.
8. [EV] O ator informa a solução.
9. [EV] O ator confirma o cadastro.
10. [RS] O sistema exibe a mensagem [MSG001].
11. [RS] O sistema atualiza a listagem de testes de cobertura.

FA2 - Editar teste de cobertura

1. [EV] O ator seleciona um teste de cobertura na listagem.
2. [RS] O sistema exibe as informações detalhadas do teste.
3. [EV] O ator altera o caso de teste.
4. [EV] O ator altera a data de realização.
5. [EV] O ator altera os objetivos.

6. [EV] O ator altera os procedimentos.
7. [EV] O ator altera a ocorrência.
8. [EV] O ator altera a solução.
9. [EV] O ator confirma a edição.
10. [RS] O sistema exibe a mensagem [MSG002].
11. [RS] O sistema atualiza a listagem de testes de cobertura.

FA3 - Excluir teste de cobertura

1. [EV] O ator seleciona um teste de cobertura na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de testes de cobertura.

FA4 - Buscar teste de cobertura

1. [EV] O ator pode informar qualquer um dos filtros: caso de teste; data de realização do teste.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de testes com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo teste é inserido na listagem de testes de cobertura.

FA2 - Um teste terá seu registro alterado na listagem de testes de cobertura.

FA3 - Um teste terá seu registro removido na listagem de testes de cobertura.

FA4 - Os testes encontrados são exibidos na listagem de testes de cobertura.

Mensagens:

MSG001 - Teste cadastrado com sucesso

MSG002 - Teste atualizado com sucesso

MSG003 - Teste deletado com sucesso

UC19 - Manter teste de integração

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Testes" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um teste de integração cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os testes de integração cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar teste de integração

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa o caso de teste.
4. [EV] O ator informa a data de realização.
5. [EV] O ator informa os objetivos.
6. [EV] O ator informa os procedimentos.
7. [EV] O ator informa a ocorrência.
8. [EV] O ator informa a solução.

9. [EV] O ator confirma o cadastro.
10. [RS] O sistema exibe a mensagem [MSG001].
11. [RS] O sistema atualiza a listagem de testes de integração.

FA2 - Editar teste de integração

1. [EV] O ator seleciona um teste de integração na listagem.
2. [RS] O sistema exibe as informações detalhadas do teste.
3. [EV] O ator altera o caso de teste.
4. [EV] O ator altera a data de realização.
5. [EV] O ator altera os objetivos.
6. [EV] O ator altera os procedimentos.
7. [EV] O ator altera a ocorrência.
8. [EV] O ator altera a solução.
9. [EV] O ator confirma a edição.
10. [RS] O sistema exibe a mensagem [MSG002].
11. [RS] O sistema atualiza a listagem de testes de integração.

FA3 - Excluir teste de integração

1. [EV] O ator seleciona um teste de integração na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de testes de integração.

FA4 - Buscar teste de integração

1. [EV] O ator pode informar qualquer um dos filtros: caso de teste; data de realização do teste.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de testes com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo teste é inserido na listagem de testes de integração.

FA2 - Um teste terá seu registro alterado na listagem de testes de integração.

FA3 - Um teste terá seu registro removido na listagem de testes de integração.

FA4 - Os testes encontrados são exibidos na listagem de testes de integração.

Mensagens:

MSG001 - Teste cadastrado com sucesso

MSG002 - Teste atualizado com sucesso

MSG003 - Teste deletado com sucesso

UC20 - Manter teste de regressão

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Testes" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um teste de regressão cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os testes de regressão cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar teste de regressão

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa o caso de teste.
4. [EV] O ator informa a data de realização.
5. [EV] O ator informa os objetivos.
6. [EV] O ator informa os procedimentos.
7. [EV] O ator informa a ocorrência.
8. [EV] O ator informa a solução.
9. [EV] O ator confirma o cadastro.
10. [RS] O sistema exibe a mensagem [MSG001].
11. [RS] O sistema atualiza a listagem de testes de regressão.

FA2 - Editar teste de regressão

1. [EV] O ator seleciona um teste de regressão na listagem.
2. [RS] O sistema exibe as informações detalhadas do teste.
3. [EV] O ator altera o caso de teste.
4. [EV] O ator altera a data de realização.
5. [EV] O ator altera os objetivos.
6. [EV] O ator altera os procedimentos.
7. [EV] O ator altera a ocorrência.
8. [EV] O ator altera a solução.
9. [EV] O ator confirma a edição.
10. [RS] O sistema exibe a mensagem [MSG002].

11. [RS] O sistema atualiza a listagem de testes de regressão.

FA3 - Excluir teste de regressão

1. [EV] O ator seleciona um teste de regressão na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de testes de regressão.

FA4 - Buscar teste de regressão

1. [EV] O ator pode informar qualquer um dos filtros: caso de teste; data de realização do teste.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de testes com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo teste é inserido na listagem de testes de regressão.

FA2 - Um teste terá seu registro alterado na listagem de testes de regressão.

FA3 - Um teste terá seu registro removido na listagem de testes de regressão.

FA4 - Os testes encontrados são exibidos na listagem de testes de regressão.

Mensagens:

MSG001 - Teste cadastrado com sucesso

MSG002 - Teste atualizado com sucesso

MSG003 - Teste deletado com sucesso

UC21 - Manter teste de carga

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Testes" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um teste de carga cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os testes de carga cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar teste de carga

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa o caso de teste.
4. [EV] O ator informa a data de realização.
5. [EV] O ator informa os objetivos.
6. [EV] O ator informa os procedimentos.
7. [EV] O ator informa a ocorrência.
8. [EV] O ator informa a solução.
9. [EV] O ator confirma o cadastro.
10. [RS] O sistema exibe a mensagem [MSG001].
11. [RS] O sistema atualiza a listagem de testes de carga.

FA2 - Editar teste de carga

1. [EV] O ator seleciona um teste de carga na listagem.
2. [RS] O sistema exibe as informações detalhadas do teste.
3. [EV] O ator altera o caso de teste.
4. [EV] O ator altera a data de realização.
5. [EV] O ator altera os objetivos.
6. [EV] O ator altera os procedimentos.
7. [EV] O ator altera a ocorrência.
8. [EV] O ator altera a solução.
9. [EV] O ator confirma a edição.
10. [RS] O sistema exibe a mensagem [MSG002].
11. [RS] O sistema atualiza a listagem de testes de carga.

FA3 - Excluir teste de carga

1. [EV] O ator seleciona um teste de carga na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de testes de carga.

FA4 - Buscar teste de carga

1. [EV] O ator pode informar qualquer um dos filtros: caso de teste; data de realização do teste.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de testes com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo teste é inserido na listagem de testes de carga.

FA2 - Um teste terá seu registro alterado na listagem de testes de carga.

FA3 - Um teste terá seu registro removido na listagem de testes de carga.

FA4 - Os testes encontrados são exibidos na listagem de testes de carga.

Mensagens:

MSG001 - Teste cadastrado com sucesso

MSG002 - Teste atualizado com sucesso

MSG003 - Teste deletado com sucesso

UC22 - Manter teste funcional

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Testes" liberado.

FA1, FA4, FA5, FA6 - Não há.

FA2, FA3 - Pelo menos um teste funcional cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os testes funcionais cadastrados. [FA1] [FA2] [FA3] [FA4] [FA5]

Fluxo alternativo:

FA1 - Cadastrar teste funcional

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa o caso de teste.

4. [EV] O ator informa a data de realização. [FE1]
5. [EV] O ator informa os objetivos.
6. [EV] O ator informa os procedimentos.
7. [EV] O ator informa a ocorrência.
8. [EV] O ator informa a solução.
9. [EV] O ator confirma o cadastro.
10. [RS] O sistema exibe a mensagem [MSG001].
11. [RS] O sistema atualiza a listagem de testes funcionais.

FA2 - Editar teste funcional

1. [EV] O ator seleciona um teste funcional na listagem.
2. [RS] O sistema exibe as informações detalhadas do teste.
3. [EV] O ator altera o caso de teste.
4. [EV] O ator altera a data de realização. [FE2]
5. [EV] O ator altera os objetivos.
6. [EV] O ator altera os procedimentos.
7. [EV] O ator altera a ocorrência.
8. [EV] O ator altera a solução.
9. [EV] O ator confirma a edição.
10. [RS] O sistema exibe a mensagem [MSG002].
11. [RS] O sistema atualiza a listagem de testes funcionais.

FA3 - Excluir teste funcional

1. [EV] O ator seleciona um teste funcional na listagem.
2. [EV] O ator seleciona a opção excluir.

3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de testes funcionais.

FA4 - Buscar teste funcional

1. [EV] O ator pode informar qualquer um dos filtros: caso de teste; data de realização do teste.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de testes com os resultados encontrados.

FA5 - Baixar relatório de incidentes de testes funcionais

1. [EV] O ator seleciona a opção listar relatórios.
2. [RS] O sistema exibe a listagem com todos os relatórios gerados. [FA6]
3. [EV] O ator seleciona um relatório.
4. [EV] O ator confirma o download.
5. [RS] O sistema baixa o relatório escolhido.

FA6 - Buscar relatórios de incidentes de testes funcionais

1. [EV] O ator informa um intervalo de datas de solicitação.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de relatórios com os resultados encontrados.

Fluxo de exceção:

FE1 - O ator informa uma data de realização futura

1. [RS] O sistema exibe a mensagem [MSG004].
2. Retorna ao passo 4 do FA1.

FE2 - O ator informa uma data de realização futura

1. [RS] O sistema exibe a mensagem [MSG004].
2. Retorna ao passo 4 do FA2.

Pós-condição:

FA1 - Um novo teste é inserido na listagem de testes funcionais.

FA2 - Um teste terá seu registro alterado na listagem de testes funcionais.

FA3 - Um teste terá seu registro removido na listagem de testes funcionais.

FA4 - Os testes encontrados são exibidos na listagem de testes funcionais.

FA5 - Uma planilha do formato xlsx é baixada no navegador do ator.

FA6 - Os relatórios encontrados são exibidos.

Mensagens:

MSG001 - Teste cadastrado com sucesso

MSG002 - Teste atualizado com sucesso

MSG003 - Teste deletado com sucesso

MSG004 - Data de realização não pode ser futura

UC23 - Gerar relatório de incidentes de testes funcionais

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Módulo "Testes" liberado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os testes funcionais cadastrados.
2. [EV] O ator seleciona a opção de gerar relatório.
3. [RS] O sistema exibe a mensagem [MSG001].

Fluxo de exceção:

Não há.

Pós-condição:

Uma planilha do formato xlsx contendo todos os testes funcionais cadastrados é gerada.

Mensagens:

MSG001 - O relatório foi gerado com sucesso

UC24 - Manter análise de riscos

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Análise de riscos" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos uma análise de riscos cadastrada.

Fluxo principal:

1. [RS] O sistema exibe a tela com as análises de riscos cadastradas. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar análise de riscos

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição da análise.
4. [EV] O ator informa o arquivo base da análise.
5. [EV] O ator confirma o cadastro.
6. [RS] O sistema exibe a mensagem [MSG001].
7. [RS] O sistema atualiza a listagem de análises de riscos.

FA2 - Editar análise de riscos

1. [EV] O ator seleciona uma análise de riscos na listagem.
2. [RS] O sistema exibe as informações detalhadas da análise.
3. [EV] O ator altera a descrição da análise.
4. [EV] O ator altera o arquivo base da análise.
5. [EV] O ator confirma a edição.
6. [RS] O sistema exibe a mensagem [MSG002].
7. [RS] O sistema atualiza a listagem de versões da análise.

FA3 - Excluir análise de riscos

1. [EV] O ator seleciona uma análise de riscos na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de análises de riscos.

FA4 - Buscar análise de riscos

1. [EV] O ator pode informar qualquer um dos filtros: descrição da análise de riscos; data da última versão da análise de riscos.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de análises com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Uma nova análise é inserida na listagem de análises de riscos.

FA2 - Uma análise terá seu registro alterado na listagem de análises de riscos.

FA3 - Uma análise terá seu registro removido na listagem de análises de riscos.

FA4 - As análises encontradas são exibidas na listagem de análises de riscos.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC25 - Manter tela do sistema

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Implantação" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos uma tela do sistema cadastrada.

Fluxo principal:

1. [RS] O sistema exibe a tela com as telas do sistema cadastradas. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar tela do sistema

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição da tela.
4. [EV] O ator informa o arquivo de prévia da tela.
5. [EV] O ator informa o arquivo base da tela.

6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].
8. [RS] O sistema atualiza a listagem de telas do sistema.

FA2 - Editar tela do sistema

1. [EV] O ator seleciona uma tela do sistema na listagem.
2. [RS] O sistema exibe as informações detalhadas da tela.
3. [EV] O ator altera a descrição da tela.
4. [EV] O ator altera o arquivo de prévia da tela.
5. [EV] O ator altera o arquivo base da tela.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de versões da tela.

FA3 - Excluir tela do sistema

1. [EV] O ator seleciona uma tela do sistema na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de telas do sistema.

FA4 - Buscar tela do sistema

1. [EV] O ator pode informar qualquer um dos filtros: descrição da tela do sistema; data da última versão da tela do sistema.
2. [EV] O ator confirma a busca.

3. [RS] O sistema atualiza a listagem das telas com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Uma nova tela é inserida na listagem de telas do sistema.

FA2 - Uma tela terá seu registro alterado na listagem de telas do sistema.

FA3 - Uma tela terá seu registro removido na listagem de telas do sistema.

FA4 - As telas encontradas são exibidas na listagem de telas do sistema.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC26 - Manter manual do usuário

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Implantação" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um manual do usuário cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os manuais do usuário cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar manual do usuário

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.

3. [EV] O ator informa a descrição do manual.
4. [EV] O ator informa o arquivo de prévia do manual.
5. [EV] O ator informa o arquivo base do manual.
6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].
8. [RS] O sistema atualiza a listagem de manuais do usuário.

FA2 - Editar manual do usuário

1. [EV] O ator seleciona um manual do usuário na listagem.
2. [RS] O sistema exibe as informações detalhadas do manual.
3. [EV] O ator altera a descrição do manual.
4. [EV] O ator altera o arquivo de prévia do manual.
5. [EV] O ator altera o arquivo base do manual.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de versões do manual.

FA3 - Excluir manual do usuário

1. [EV] O ator seleciona um manual do usuário na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.
5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de manuais do usuário.

FA4 - Buscar manual do usuário

1. [EV] O ator pode informar qualquer um dos filtros: descrição do manual do usuário; data da última versão do manual do usuário.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de manuais com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo manual é inserido na listagem de manuais do usuário.

FA2 - Um manual terá seu registro alterado na listagem de manuais do usuário.

FA3 - Um manual terá seu registro removido na listagem de manuais do usuário.

FA4 - Os manuais encontrados são exibidos na listagem de manuais do usuário.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

UC27 - Manter help do sistema

Ator(es): Gerente de projetos, Analista de requisitos

Pré-condição:

Fluxo principal - Módulo "Implantação" liberado.

FA1, FA4 - Não há.

FA2, FA3 - Pelo menos um help do sistema cadastrado.

Fluxo principal:

1. [RS] O sistema exibe a tela com os helps do sistema cadastrados. [FA1] [FA2] [FA3] [FA4]

Fluxo alternativo:

FA1 - Cadastrar help do sistema

1. [EV] O ator seleciona a opção novo.
2. [RS] O sistema exibe a tela de cadastro.
3. [EV] O ator informa a descrição do help.
4. [EV] O ator informa o arquivo de prévia do help.
5. [EV] O ator informa o arquivo base do help.
6. [EV] O ator confirma o cadastro.
7. [RS] O sistema exibe a mensagem [MSG001].
8. [RS] O sistema atualiza a listagem de helps do sistema.

FA2 - Editar help do sistema

1. [EV] O ator seleciona um help do sistema na listagem.
2. [RS] O sistema exibe as informações detalhadas do help.
3. [EV] O ator altera a descrição do help.
4. [EV] O ator altera o arquivo de prévia do help.
5. [EV] O ator altera o arquivo base do help.
6. [EV] O ator confirma a edição.
7. [RS] O sistema exibe a mensagem [MSG002].
8. [RS] O sistema atualiza a listagem de versões do help.

FA3 - Excluir help do sistema

1. [EV] O ator seleciona um help do sistema na listagem.
2. [EV] O ator seleciona a opção excluir.
3. [RS] O sistema solicita a confirmação da exclusão.
4. [EV] O ator confirma a exclusão.

5. [RS] O sistema exibe a mensagem [MSG003].
6. [RS] O sistema atualiza a listagem de helps do sistema.

FA4 - Buscar help do sistema

1. [EV] O ator pode informar qualquer um dos filtros: descrição do help do sistema; data da última versão do help do sistema.
2. [EV] O ator confirma a busca.
3. [RS] O sistema atualiza a listagem de helps com os resultados encontrados.

Fluxo de exceção:

Não há.

Pós-condição:

FA1 - Um novo help é inserido na listagem de helps do sistema.

FA2 - Um help terá seu registro alterado na listagem de helps do sistema.

FA3 - Um help terá seu registro removido na listagem de helps do sistema.

FA4 - Os helps encontrados são exibidos na listagem de helps do sistema.

Mensagens:

MSG001 - Artefato cadastrado com sucesso

MSG002 - Artefato atualizado com sucesso

MSG003 - Artefato deletado com sucesso

APÊNDICE B – DESCRIÇÃO DOS DADOS

Tabela 6: Descrição de dados da entidade "arquivo"

arquivo			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
bin_conteudo	binário		Conteúdo do arquivo
txt_media_type	texto		Tipo de mídia do arquivo
nm_nome	texto		Nome do arquivo

Fonte: Do autor

Tabela 7: Descrição de dados da entidade "artefato"

artefato			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
ds_descricao	texto		Descrição do Artefato
en_tipo	texto		Tipo do artefato
id_etapa_projeto	inteiro	FK	Identificador da etapa do projeto
id_versao_atual	inteiro	FK	Identificador da versão do artefato
fl_excluido	booleano		Sinalizador de exclusão

Fonte: Do autor

Tabela 8: Descrição de dados da entidade "atributo"

atributo			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
nm_nome	texto		Nome do atributo
en_tipo	texto		Tipo de dado do atributo
fl_not_null	booleano		Atributo pode ser nulo ou não
fl_pk	booleano		Representa chave primária ou não
fl_fk	booleano		Representa chave estrangeira ou não
id_entidade	inteiro	FK	Identificador da entidade
fl_excluido	booleano		Sinalizador de exclusão

Fonte: Do autor

Tabela 9: Descrição de dados da entidade "entidade"

entidade			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
nm_nome	texto		Nome da Entidade
ds_descricao	texto		Descrição da Entidade
id_etapa_projeto	inteiro	FK	Identificador da etapa do projeto
fl_excluido	booleano		Sinalizador de exclusão

Fonte: Do autor

Tabela 10: Descrição de dados da entidade "etapa_projeto"

etapa_projeto			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
en_fase	texto		Fase da etapa do projeto
en_status	texto		Status da etapa do projeto
id_projeto	inteiro	FK	Identificador do projeto

Fonte: Do autor

Tabela 11: Descrição de dados da entidade "historico_acao_projeto"

historico_acao_projeto			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
en_acao	texto		Ação realizada dentro do projeto
dt_data_hora_acao	data		Data e hora da ação
id_registro	inteiro		Identificador do registro impactado na ação
txt_tabela	texto		Tabela impactada na ação
id_etapa_projeto	inteiro	FK	Identificador da etapa do projeto
id_projeto	inteiro	FK	Identificador do projeto
id_usuario_acao	inteiro	FK	Identificador do usuário da ação

Fonte: Do autor

Tabela 12: Descrição de dados da entidade "projeto"

projeto			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
nm_nome	texto		Nome do Projeto
ds_descricao	texto		Nome do Projeto
dt_data_inclusao	data		Data de inclusão do Projeto

Fonte: Do autor

Tabela 13: Descrição de dados da entidade "relatorio"

relatorio			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
bin_arquivo	binário		Conteúdo do relatório
dt_data_solicitacao	data		Data e hora da solicitação
txt_file_name	texto		Nome do arquivo do relatório
txt_media_type	texto		Tipo de mídia do arquivo do relatório
en_tipo	texto		Tipo do relatório
id_etapa_projeto	inteiro	FK	Identificador da etapa do projeto
id_usuario_solicitacao	inteiro	FK	Identificador do usuário solicitante

Fonte: Do autor

Tabela 14: Descrição de dados da entidade "requisito"

requisito			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
nm_nome	texto		Nome do requisito
ds_descricao	texto		Descrição do requisito
en_tipo	texto		Tipo do requisito
id_etapa_projeto	inteiro	FK	Identificador da etapa do projeto
fl_excluido	booleano		Sinalizador de exclusão

Fonte: Do autor

Tabela 15: Descrição de dados da entidade "role"

role			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
ds_description	texto		Descrição da role
en_name	texto		Nome da role

Fonte: Do autor

Tabela 16: Descrição de dados da entidade "sprint"

sprint			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
dt_data_inicio	data		Data de início da sprint
dt_data_termino	data		Data de término da sprint
cod_numero_entrega	inteiro		Número da entrega
txt_versao	texto		Texto da versão
id_projeto	inteiro	FK	Identificador do projeto
fl_excluido	booleano		Sinalizador de exclusão

Fonte: Do autor

Tabela 17: Descrição de dados da entidade "sprint_entidade"

sprint_entidade			
Atributo	Tipo de dado	(PK/FK)	Descrição
id_sprint	inteiro	PK e FK	Identificador da sprint
id_entidade	inteiro	PK e FK	Identificador da entidade

Fonte: Do autor

Tabela 18: Descrição de dados da entidade "sprint_requisito"

sprint_requisito			
Atributo	Tipo de dado	(PK/FK)	Descrição
id_sprint	inteiro	PK e FK	Identificador da sprint
id_requisito	inteiro	PK e FK	Identificador do requisito

Fonte: Do autor

Tabela 19: Descrição de dados da entidade "sprint_teste"

sprint_teste			
Atributo	Tipo de dado	(PK/FK)	Descrição
id_sprint	inteiro	PK e FK	Identificador da sprint
id_teste	inteiro	PK e FK	Identificador do teste

Fonte: Do autor

Tabela 20: Descrição de dados da entidade "sprint_versao_artefato"

sprint_versao_artefato			
Atributo	Tipo de dado	(PK/FK)	Descrição
id_sprint	inteiro	PK e FK	Identificador da sprint
id_versao_artefato	inteiro	PK e FK	Identificador da versão do artefato

Fonte: Do autor

Tabela 21: Descrição de dados da entidade "teste"

teste			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
dt_data_realizacao	data		Data e hora de realização do teste
txt_caso_teste	texto		Unidade testada
txt_objetivos	texto		Objetivos do teste
txt_ocorrencia	texto		Ocorrência durante o teste
txt_procedimentos	texto		Procedimentos adotados no teste
txt_solucao	texto		Solução adotada para a ocorrência
en_tipo	texto		Tipo do Teste
id_etapa_projeto	inteiro	FK	Identificador da etapa do projeto
fl_excluido	booleano		Sinalizador de exclusão

Fonte: Do autor

Tabela 22: Descrição de dados da entidade "users"

users			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
txt_email	texto		E-mail do usuário
txt_password	texto		Senha criptografada do usuário
txt_username	texto		Login do usuário
id_role	inteiro	FK	Identificador da role do usuário

Fonte: Do autor

Tabela 23: Descrição de dados da entidade "versao_artefato"

versao_artefato			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	inteiro	PK	Identificador
dt_data_inclusao	data		Data de inclusão da versão
cod_numero_versao	inteiro		Número da versão
id_arquivo_base	inteiro	FK	Identificador do arquivo base
id_arquivo_previa	inteiro	FK	Identificador do arquivo de prévia
id_artefato	inteiro	FK	Identificador do artefato
id_usuario _inclusao	inteiro	FK	Identificador do usuário da inclusão

Fonte: Do autor