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

DEPARTAMENTO DE COMPUTAÇÃO
CIÊNCIA DA COMPUTAÇÃO

BRUNO FONSECA DO CARMO

UMA PROPOSTA DE SISTEMA DE INFORMAÇÃO PARA CONTROLE E SUPORTE A BANCAS DE TCC DE GRADUAÇÃO

ALEGRE - ES
JULHO DE 2023

UMA PROPOSTA DE SISTEMA DE INFORMAÇÃO PARA CONTROLE E SUPORTE A BANCAS DE TCC DE GRADUAÇÃO

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

BRUNO FONSECA DO CARMO

Orientador
GIULIANO PRADO DE MORAIS GIGLIO

Universidade Federal do Espírito Santo

ALEGRE - ES

JULHO DE 2023

BRUNO FONSECA DO CARMO

UMA PROPOSTA DE SISTEMA DE INFORMAÇÃO PARA CONTROLE E SUPORTE A BANCAS DE TCC

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 19 de julho de 2023.

COMISSÃO EXAMINADORA

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

Prof. Dr. Bruno Vilela Oliveira
Universidade Federal do Espírito Santo

Prof. Dr. Dayan de Castro Bissoli
Universidade Federal do Espírito Santo

SUMÁRIO

1 Introdução **18**

1.1 Problema e sua Importância 19

1.2 Objetivos 20

1.2.1 Objetivo Geral 20

1.2.2 Objetivos específicos 21

2 Revisão de Literatura **22**

2.1 Sistemas relacionados 22

2.1.1 Gestor de TCC: Desenvolvimento de Sistema para Gestão de Trabalhos Acadêmicos 22

2.1.2 Projeto e Desenvolvimento de um Sistema para Gerenciamento de Trabalhos de Conclusão de Curso 22

2.1.3 Sistema Web para Gestão de Trabalhos de Conclusão de Curso 23

2.1.4 SGTCC: Sistema Gerenciador de Trabalho de Conclusão de Curso 23

2.1.5 WebTCC - Plataforma para controle e acompanhamento de trabalhos finais de curso 23

2.1.6 Desenvolvimento de um Sistema para o Gerenciamento do Processo de Trabalho de Conclusão de Curso do Curso de Tecnologia em Sistemas para Internet da UTFPR Campus Guarapuava 24

2.1.7 Sistema de Controle de Submissão de Propostas de Trabalho de Conclusão de Curso 24

2.1.8 GTFURB: Sistema para Gestão de Trabalho de Conclusão de Curso do Departamento de Sistemas E Computação Da Furb 24

2.1.9 TCC Online: Sistema de Gerenciamento de Trabalhos de Conclusão de Curso 25
2.1.10 Sistema Gerenciador de Bancas de TCC – Gbancas 25
2.2 Tecnologias Envolvidas 26
2.2.1 Spring Framework 26
2.2.2 Spring Boot 26
2.2.3 Spring Data JPA 27
2.2.4 Spring Security 27
2.2.5 JSON 28
2.2.6 Angular 28
2.2.7 TypeScript 28
2.2.8 PostgreSQL 29
2.2.9 Git 29
2.2.10 GitHub 29

3 Metodologia 30
3.1 Metodologia do desenvolvimento 30
3.2 Análise dos requisitos 33
3.2.1 Descrição dos usuários 34
3.2.2 Requisitos funcionais 34
3.2.3 Requisitos não funcionais 37
3.2.4 Regras de negócio 38
3.2.5 Casos de uso 39
3.2.6 Diagrama de classe 41
3.3 Modelagem do banco de dados 42
3.3.1 Descrição dos dados 44

4 Desenvolvimento 45

4.1 Arquitetura do sistema 45
4.1.1 API 45
4.1.2 REST 46
4.1.3 Desenho da arquitetura 49
4.2 Backend 49
4.2.1 Estrutura do Projeto 50
4.2.2 Conexão com o banco de dados 51
4.2.3 Endpoints 53
4.2.4 Facades 54
4.2.5 Services 55
4.2.6 Entidades 56
4.2.7 Segurança 57
4.2.7.1 Autenticação 60
4.2.7.2 Autorização 61
4.3 Frontend 63
4.3.1 Estrutura do projeto 63
4.3.2 Comunicação com o servidor 64
4.3.3 Navegação 67
4.4 Interfaces 71
4.4.1 Tela de Login 71
4.4.2 Tela Inicial 71
4.4.3 Telas de Áreas de TCC 73
4.4.4 Telas de Professor 73
4.4.5 Telas de Aluno 75
4.4.6 Telas de Usuário 76
4.4.7 Telas de Importação 78

4.4.8 Telas de Modelo de Avaliação 81
4.4.9 Telas de Modelo de Documento 83
4.4.10 Telas de Projeto de TCC 84
4.4.11 Telas de Agenda 90
4.4.12 Telas de Avaliação 93

5 Testes e Validações 98
5.1 Inspeções 98
5.2 Teste de Desenvolvimento 99
5.3 Teste de Usuário 102
5.4 Teste de Usabilidade 103
5.5 Avaliação dos Testes Realizados 104
5.5.1 Satisfação dos Testes 105
5.5.2 Resultados Alcançados 105
5.5.3 Análise Global dos Testes 105

6 Considerações Finais e Conclusões 107
6.1 Contribuição científica da proposta deste trabalho 108
6.2 Trabalhos Futuros 109

Referências 111

Apêndice A – Descrição das tabelas do banco de dados 115

Anexo A – Termo de Compromisso de Desenvolvimento de TCC 131

LISTA DE FIGURAS

Figura 1 Exemplo de um JSON 28

Figura 2 Processo *Scrum* 31

Figura 3 O modelo incremental 33

Figura 4 Casos de uso referente ao administrador 40

Figura 5 Casos de uso referente ao professor de TCC, orientador, membros da banca e aluno orientando 40

Figura 6 Diagrama de classe 41

Figura 7 Modelo físico do banco de dados 43

Figura 8 Arquitetura do sistema 49

Figura 9 Estrutura do Projeto - Backend 50

Figura 10 Propriedades de configuração do banco de dados 51

Figura 11 Exemplo de repository 52

Figura 12 Exemplo de controller 53

Figura 13 Exemplo de facade 54

Figura 14 Exemplo de service 55

Figura 15 Exemplo de entidade 56

Figura 16 Exemplo de login 58

Figura 17 Exemplo do token decodificado 59

Figura 18 Exemplo do refresh token decodificado 59

Figura 19 Implementação da autenticação 61

Figura 20 Implementação da autorização 62

Figura 21 Exemplo de endpoint com autorização 62

Figura 22 Estrutura do Projeto - Frontend 63

Figura 23 BaseService 65

Figura 24 Classe Environment 65

Figura 25 Exemplo de uma service 66

Figura 26 AuthInterceptor 67

Figura 27 Módulo da agenda 69

Figura 28 Módulo de roteamento da agenda 69

Figura 29 Módulo de roteamento geral da aplicação 70

Figura 30 Tela de login 71

Figura 31 Tela Inicial 72

Figura 32 Menu Administrador 72

Figura 33 Menu Aluno 72

Figura 34 Tela de listagem de áreas de TCC 73

Figura 35 Tela de cadastro de área de TCC 73

Figura 36 Tela de listagem de professores 74

Figura 37 Tela de cadastro de professor 74

Figura 38 E-mail de criação do professor 75

Figura 39 Tela de listagem de alunos 75

Figura 40 Tela de cadastro de aluno 76

Figura 41 E-mail de criação do aluno 76

Figura 42 Tela de listagem de usuários 77

Figura 43 Modal de edição de usuário 77

Figura 44 Modal de desativação do usuário 78

Figura 45 Modal de importação 78

Figura 46 Planilha de importação de professores 79

Figura 47 Planilha de importação de alunos 79

Figura 48 Modal de erros na importação 80

Figura 49 Planilha de importação de professores com erros 80

Figura 50 Planilha de importação de alunos com erros 80

Figura 51 Tela de listagem de modelos de avaliação 81

Figura 52 Tela de cadastro de modelo de avaliação 81

Figura 53 Tela de edição do modelo de avaliação 82

Figura 54 Modal de cadastro de itens do modelo de avaliação 83

Figura 55 Tela de listagem de modelos de documento 83

Figura 56 Tela de cadastro de modelo de documento 84

Figura 57 Tela de listagem de projeto de TCC 84

Figura 58 Tela de cadastro de projeto de TCC 85

Figura 59 Tela de cadastro dos documentos do projeto de TCC 85

Figura 60 Tela de cadastro das observações do projeto de TCC 86

Figura 61 Tela de cadastro dos anexos gerais do projeto de TCC 86

Figura 62 Tela de agendamento da apresentação do projeto de TCC 87

Figura 63 Modal de cancelamento/reprovação de um projeto de TCC 87

Figura 64 Modal de confirmação da solicitação de membro à banca - Qualificação 88

Figura 65 Modal de solicitação de membro à banca 88

Figura 66 E-mail de solicitação para membro de banca 89

Figura 67 Tela de listagem de projetos de TCC - Membro de Banca 89

Figura 68 Planilha de exportação de projeto de TCC 90

Figura 69 Tela de listagem de agendas de apresentação 90

Figura 70 Tela de cadastro de agenda de apresentação - parte 1 91

Figura 71 Tela de cadastro de agenda de apresentação - parte 2 91

Figura 72 Tela de cadastro das disponibilidades 92

Figura 73 Tela da minha agenda 93

Figura 74 Tela da avaliação do projeto de TCC 94

Figura 75 Tela de documentos de TCC na etapa de avaliação 94

Figura 76 Tela de observação do documento de TCC na etapa de avaliação . . 95

Figura 77 Tela de anexos gerais de TCC na etapa de avaliação 95

Figura 78 Modal de avaliação do TCC 96

Figura 79 Modal de listagem das avaliações do TCC 96

Figura 80 Tela de avaliação de projeto de TCC finalizada 97

LISTA DE TABELAS

Tabela 1 Usuários do sistema 34

Tabela 2 Descrição de dados da entidade agenda_apresentacao 44

Tabela 3 Tabela de inspeções 98

Tabela 4 Resultado dos testes de desenvolvimento 100

Tabela 5 Resultado dos testes de usabilidade 103

Tabela 6 Descrição de dados da entidade agenda_apresentacao_restricao . . 115

Tabela 7 Descrição de dados da entidade aluno 116

Tabela 8 Descrição de dados da entidade anexo_geral 117

Tabela 9 Descrição de dados da entidade apresentacao 118

Tabela 10 Descrição de dados da entidade area_tcc 118

Tabela 11 Descrição de dados da entidade membro_banca 119

Tabela 12 Descrição de dados da entidade modelo_aspecto_avaliacao 119

Tabela 13 Descrição de dados da entidade modelo_avaliacao 120

Tabela 14 Descrição de dados da entidade modelo_documento_tipo_tcc 120

Tabela 15 Descrição de dados da entidade modelo_documento 121

Tabela 16 Descrição da entidade modelo_item_avaliacao_tipo_professor 121

Tabela 17 Descrição de dados da entidade modelo_item_avaliacao_tipo_tcc .. 121

Tabela 18 Descrição de dados da entidade modelo_item_avaliacao 122

Tabela 19 Descrição de dados da entidade professor_area_tcc 122

Tabela 20 Descrição de dados da entidade professor_disponibilidade 123

Tabela 21 Descrição de dados da entidade professor 123

Tabela 22 Descrição de dados da entidade projeto_tcc_aluno 124

Tabela 23 Descrição de dados da entidade projeto_tcc_aspecto_avaliacao 124

Tabela 24 Descrição de dados da entidade projeto_tcc_avaliacao 125

Tabela 25 Descrição de dados da entidade projeto_tcc_doc_visualizado_por .. 125

Tabela 26 Descrição de dados da entidade projeto_tcc_nota 126

Tabela 27 Descrição de dados da entidade projeto_tcc_situacao 126

Tabela 28 Descrição de dados da entidade projeto_tcc 127

Tabela 29 Descrição de dados da entidade role 127

Tabela 30 Descrição de dados da entidade user_role 128

Tabela 31 Descrição de dados da entidade user 128

Tabela 32 Descrição de dados da entidade versao_tcc_observacao 129

Tabela 33 Descrição de dados da entidade versao_tcc 130

LISTA DE SIGLAS

TCC	Trabalho de Conclusão de Curso
UFES	Universidade Federal do Espírito Santo
HTML	*HyperText Markup Language*
CSS	*Cascading Style Sheets*
SGTCC	Sistema Gerenciador de Trabalho de Conclusão de Curso
PDF	*Portable Document Format*
JSP	*Java Server Pages*
HTTP	*Hypertext Transfer Protocol*
PHP	*Hypertext Preprocessor*
IDE	Ambiente de desenvolvimento integrado
JPA	*Java Persistence API*
JSON	*JavaScript Object Notation*
SPA	*Single Page Applications*
SGBDOR	Sistema de Gerenciamento de Banco de Dados de Objeto Relacional
SGBD	Sistema de Gerenciamento de Banco de Dados
CI/CD	Continuous Integration/Continuous Delivery
PO	*Product Owner*
UML	*Unified Modeling Language*
RF	Requisitos Funcionais
RNF	Requisitos Não Funcionais
MER	Modelo Entidade Relacionamento
REST	*Representational State Transfer*
API	*Application Programming Interface*
HTTP	Hypertext Transfer Protocol

URI	Uniform Resource Identifier
XML	*Extensible Markup Language*
JWT	JSON Web Token
CSV	*Comma-Separated Values*

RESUMO

O Trabalho de Conclusão de Curso (TCC) representa uma das etapas mais significativas durante a graduação, sendo essencial para a conclusão do curso. No entanto, essa fase envolve diversos processos que requerem atenção por parte das pessoas envolvidas, incluindo o aluno, o professor de TCC, o orientador e os membros da banca. Se essa devida atenção não for concedida, pode comprometer a aprovação do aluno no TCC. Visando resolver este problema, e outros existentes, o presente trabalho apresenta uma proposta de desenvolvimento de um sistema web para servir de suporte, com o objetivo de auxiliar todos os atuantes no processo de TCC, e assim tornar este processo menos complexo e mais produtivo. Para o desenvolvimento do sistema, foi utilizado no *backend* o framework Spring, e no *frontend* a plataforma Angular, e para o armazenamento de dados, o banco de dados PostgreSQL. Portanto, espera-se que, com a utilização do sistema desenvolvido, atinja-se o seu objetivo de flexibilizar e dar mais celeridade ao processo de TCC.

Palavras-chave: Trabalho de Conclusão de Curso; Sistema Web; Spring Framework; Angular; Java.

1 INTRODUÇÃO

Na atualidade é notável o impacto que a tecnologia da informação traz para as diferentes áreas, as beneficiando com diferentes estratégias, entre elas, pode-se citar a automatização nos diferentes processos de organização, melhorando sua eficácia e sua eficiência (MORAES; SILVA; ADVERS, 2021).

Além disso, a implementação de um sistema de informação em uma organização promove o aumento da sua produtividade, reduzindo a incidência de erros ou a necessidade de retrabalhos, auxiliando os gestores nas tomadas de decisão e consequentemente diminuindo os custos da organização (ALBERTIN; ALBERTIN, 2008). Desta forma, é possível automatizar um processo que é totalmente manual, no qual não exista nenhuma tecnologia implementada que auxilie em sua execução. Em nível acadêmico, dentre as diversas necessidades e aplicações, pode-se citar sua contribuição no processo de controle e suporte à bancas de TCC (Trabalho de Conclusão de Curso).

O TCC é o trabalho que completa a formação do discente, produzido nas etapas finais do ensino superior, sendo este, composto por várias etapas, as quais são imprescindíveis para que o aluno se prepare para o mercado de trabalho. O TCC agrega na formação de novos profissionais em diferentes áreas, contribuindo para o desenvolvimento da ciência e tecnologia, e consequentemente, da sociedade (GUIMARÃES; SOBRINHO, 2020).

Para a elaboração do TCC, é necessária a observância de algumas dimensões, como: conhecimento em metodologia, planejamento de pesquisa e a condução da orientação. A orientação é realizada pelos professores, os quais devem fornecer a orientação como auxílio e trabalho conjunto, no planejamento e desempenho da pesquisa. O orientador deve nortear o orientando nos aspectos técnico-científicos e também nos aspectos éticos (MEDEIROS et al., 2015).

Cada instituição define como ocorre o processo de execução de um TCC. Sendo as-

sim, o presente trabalho se baseia inicialmente nos processos de execução do TCC do Departamento de Computação da Universidade Federal do Espírito Santo (UFES) - Campus de Alegre, e a partir destes, avaliar a possibilidade de ampliar aos demais cursos de graduação. O processo supracitado consiste em duas disciplinas, que se constituem em duas etapas de TCC, que são: disciplina de Trabalho de Conclusão de Curso I (TCC 1) e a segunda, Trabalho de Conclusão de Curso II (TCC 2). O TCC 1 é a construção de um pré-projeto, sua apresentação e validação (que se constitui numa qualificação do mesmo), e o TCC 2 é finalização do projeto em si com sua defesa (UFES, 2012).

O TCC é avaliado por uma banca examinadora, composta pelo professor orientador, e por, no mínimo, mais dois membros convidados (UFES, 2012). É de suma relevância que o orientando conheça os membros de sua banca, dessa forma, o discente conhecerá as possíveis cobranças, e ficará mais preparado para entregar um trabalho de boa qualidade (LOPES et al., 2020).

De acordo com UFES (2012), o fluxo de execução do TCC I inicia-se com um graduando à procura de um professor orientador, o qual poderá aceitar ou não a proposta do graduando. Caso aceite, será preenchido e assinado um Termo de Compromisso de Desenvolvimento de TCC, disposto no Anexo A, e entregue ao professor da disciplina de TCC 1, o qual divulgará o calendário das apresentações dos pré-projetos. No TCC II, o professor regente conduzirá as etapas finais do TCC, que, dentre outras coisas, fará a divulgação do calendário de defesa de TCC para que os graduandos finalizem e preparem suas apresentações que, após isto, enviará as cópias do seu trabalho ao seu orientador e a banca de avaliação, e por fim defender seu TCC (UFES, 2012).

Assim que o graduando defender seu TCC, caso seja aprovado, apenas será necessário entregar a versão final encadernada de seu trabalho, caso contrário, se for sua primeira reprovação, poderá corrigir seu TCC e defendê-lo novamente até cinco dias úteis antes do término do semestre letivo, por outro lado se não for sua primeira reprovação, este graduando deverá cursar a disciplina novamente (UFES, 2012).

1.1 Problema e sua Importância

De acordo com Clemente e Santos (2015), existem alguns aspectos que são negativos na elaboração do TCC, como: falha na observação do cronograma do trabalho, falta de comunicação entre orientador e orientando, incompatibilidade de horários, distância

geográfica, dificuldade em encontrar material bibliográfico, custos, limitação de tempo na agenda do professor, entre outros.

Ainda se tratando sobre as dificuldades, Malheiros e Oliveira (2004) afirma em seu trabalho que, os discentes entrevistados relataram que suas maiores dificuldades foram em custos, tempo e na procura de um orientador.

Além desses aspectos negativos, existem também alguns regulamentos que podem dificultar o processo do TCC. Alguns passos precisam ser cumpridos para que o mesmo chegue à banca, sendo estes, possíveis complicadores do processo, fazendo com que os alunos necessitem estar atentos aos e-mails, ou até mesmo buscar informações em sua instituição sobre as etapas no processo do TCC, tais como: professores disponíveis para orientá-los, temas disponíveis, cronogramas de entrega, apresentação, entre outros.

Em vista disso, o presente trabalho propõe o desenvolvimento de um sistema a fim de auxiliar no controle e suporte a bancas de TCC, com o propósito de diminuir ou até mesmo resolver os problemas relatados anteriormente.

1.2 Objetivos

Nesta seção são apresentados os objetivos do presente trabalho.

1.2.1 Objetivo Geral

Este trabalho tem como objetivo desenvolver um sistema de informação web de controle e suporte à bancas de TCC, que terá como base o processo de TCC do Departamento de Computação da UFES - Campus Alegre, projetado para ter todas as etapas de um TCC acadêmico aderente a qualquer curso de graduação. Pretende-se tornar o processo mais produtivo e menos complexo para os agentes envolvidos, que são os professores de TCC, professores orientadores, professores da banca e orientandos, além de possibilitar interfaces responsivas e mais intuitivas, como critérios de usabilidade satisfatórios.

1.2.2 Objetivos específicos

Além dos objetivos gerais citados anteriormente, planeja-se atingir os seguintes objetivos específicos:

1. Ajudar na organização das orientações dos alunos em TCC, quanto ao processo de construção à sua defesa;
2. Dar suporte logístico aos professores supervisores de TCC na administração de documentos, comunicação e agendamentos necessários;
3. Gerenciar as tarefas previstas dos membros das bancas de TCC;
4. Proporcionar informações sobre todo o processo de TCC;
5. Realizar todo o gerenciamento da implementação do sistema, desde o levantamento de requisitos, sua análise, implementação das estruturas arquiteturais, até a garantia da qualidade, usando técnicas de testes aderentes ao tipo de sistema proposto;
6. Aplicar tecnologias mais atuais que as encontradas em sistemas similares, aplicando técnicas de vanguarda no desenvolvimento proposto.

2 REVISÃO DE LITERATURA

Neste capítulo são apresentadas as fundamentações teóricas que embasam no desenvolvimento deste trabalho.

2.1 Sistemas relacionados

Existem diversos sistemas na literatura que auxiliam desde a gestão do TCC até o controle de bancas do mesmo. Nos tópicos a seguir, são evidenciados alguns deles, a fim de realizar uma análise comparativa entre a proposta deste trabalho com os trabalhos publicados, objetivando, assim, levantar a contribuição científica do mesmo, nos resultados alcançados. É importante ressaltar que todos os sistemas identificados e analisados nesta pesquisa foram desenvolvidos em ambientes acadêmicos.

2.1.1 Gestor de TCC: Desenvolvimento de Sistema para Gestão de Trabalhos Acadêmicos

Esse sistema foi desenvolvido com o objetivo de facilitar a comunicação entre os envolvidos dos projetos, a produtividade dos trabalhos e a geração de documentação do processo. As linguagens de programação que foram utilizadas são Python com o *framework* Django, devido às vantagens que esses oferecem, como a estabilidade, confiabilidade, a grande comunidade colaborativa atuante e a facilidade sintática e ortográfica (ALVES; URQUIZA; ROLAND, 2016).

2.1.2 Projeto e Desenvolvimento de um Sistema para Gerenciamento de Trabalhos de Conclusão de Curso

Esse sistema promove o cadastro, a redefinição de senha e autenticação do usuário, a listagem de todos os trabalhos cadastrados e detalhamento, adição de novos

trabalhos, edição e remoção de trabalhos atualmente cadastrados, geração de atas, bloqueio e desbloqueio de usuários. As linguagens utilizadas para o desenvolvimento desse sistema são HTML, CSS e Javascript (SILVA, 2017).

2.1.3 Sistema Web para Gestão de Trabalhos de Conclusão de Curso

Esse sistema fornece funcionalidades que auxiliam em diversos aspectos na produção de um TCC, podendo destacar a possibilidade de cadastro de assinatura digital e informações pessoais, a submissão e visualização de propostas de trabalho, o controle de frequência, a submissão de trabalho, o cadastro do regulamento das bancas, o cadastro das bancas, a listagem do resultado das bancas e a emissão dos documentos importantes. Esse sistema foi desenvolvido com a utilização das tecnologias HTML, CSS e JavaScript, foi utilizado também o PHP, que é uma linguagem de programação especialmente adequada para desenvolvimento web (BIRCK, 2021).

2.1.4 SGTCC: Sistema Gerenciador de Trabalho de Conclusão de Curso

O sistema SGTCC (Sistema Gerenciador de Trabalho de Conclusão de Curso) tem como funcionalidade o cadastro de papéis de usuários, o cadastro de usuários e temas, a moderação de temas, o cadastro, edição e exclusão de TCC's, a troca de orientador e coorientador, a composição de banca e a emissão de relatórios. Para o desenvolvimento do SGTCC foram utilizadas linguagens como PHP, HTML, Javascript e CSS, além de utilizar a biblioteca jQuery para requisições no servidor (PAZ et al., 2015).

2.1.5 WebTCC - Plataforma para controle e acompanhamento de trabalhos finais de curso

Essa plataforma possibilita o professor de TCC de acompanhar um determinado trabalho e, se necessário, promover a garantia da qualidade dos mesmos. O WebTCC também possibilita o professor de TCC realizar operações de inclusão, exclusão e alteração de alunos, orientadores, trabalhos de finais de curso, como também, gerenciar perguntas e respostas mais frequentes para um TCC de um dado curso, gerenciar

propostas de TCC e gerar relatório em PDF das reuniões realizadas. Para a construção desta plataforma, foram utilizadas as tecnologias: HTML, CSS, Vuetify, Vue.js, JavaScript, PHP, Laravel e MySQL (FREITAS; SILVA, 2020).

2.1.6 Desenvolvimento de um Sistema para o Gerenciamento do Processo de Trabalho de Conclusão de Curso do Curso de Tecnologia em Sistemas para Internet da UTFPR Campus Guarapuava

Esse sistema oferece um maior controle de professores e acadêmicos nas atividades do TCC, com benefícios na resolução de dúvidas e feedbacks, uma vez que as informações referentes a cada processo de TCC estão armazenadas no mesmo sistema. Para a construção desse sistema, foram utilizadas as ferramentas HTML, CSS3, Bootstrap, Javascript, jQuery, Ruby, framework Rails, Scrum (FERREIRA, 2015).

2.1.7 Sistema de Controle de Submissão de Propostas de Trabalho de Conclusão de Curso

Esse sistema permite que o aluno se cadastre e envie suas propostas de modo simplificado. O sistema possui ferramentas autoexplicativas, fazendo com que os professores possam gerenciar e avaliar as propostas sem dificuldades. Este sistema foi construído a partir da tecnologia JSP que auxilia na criação de sistemas web utilizando a linguagem de programação Java. Para o banco de dados, foi utilizado o MySQL, porque possui consistência, alta performance, confiabilidade e é gratuito. O Web Service escolhido foi REST devido a facilidade de transferência de dados utilizando protocolo HTTP. No desenvolvimento *mobile* foi escolhido Android, por ser um sistema que utiliza Java para programação e possui multi-thread, o que facilita a transição de informações (SIQUEIRA, 2014).

2.1.8 GTFURB: Sistema para Gestão de Trabalho de Conclusão de Curso do Departamento de Sistemas E Computação Da Furb

Esse sistema possibilita o seu acesso através da utilização de um navegador web, por meio de uma interface simples e intuitiva que permite aos usuários acompanharem todos os processos do desenvolvimento de TCC. O GTFURB também permite que o

coordenador tenha facilidade na organização e formação das bancas examinadoras. A linguagem de programação utilizada foi Java para plataforma web e o banco de dados MySQL. Foi utilizado o framework Hibernate para realizar a comunicação e as transações no banco de dados e o framework PrimeFaces, auxiliando no desenvolvimento das páginas web (HENSCHEL, 2020).

2.1.9 TCC Online: Sistema de Gerenciamento de Trabalhos de Conclusão de Curso

Esse sistema promove facilidade na organização da elaboração de TCCs através de seus requisitos funcionais, os quais são: cadastro de usuários, notificação ao usuário sobre o sucesso do cadastro via e-mail, edição de dados pessoais do usuário, criação, edição e finalização de projetos, realizar convites a outros usuários para integrar a um projeto, criar novas tarefas, atribuí-las a usuários específicos, editá-las e fechá-las, além disso, é possível criar novas versões para o projeto, anexar arquivos, e inserir comentários nessas versões. Para o desenvolvimento do sistema, foram utilizadas as seguintes tecnologias: PHP, banco de dados MySQL, HTML, CSS, template AdminLTE, Gravatar (serviço para inserção de avatares em sites), sistema de controle de versão de arquivos Git, e para hospedagem do código foi utilizado o GitHub (CABRAL, 2017).

2.1.10 Sistema Gerenciador de Bancas de TCC – Gbancas

Esse sistema auxilia o coordenador a gerenciar todas as etapas de formação das bancas, como, o cadastramento dos alunos, seus respectivos trabalhos, orientadores e todos os componentes necessários para formação da banca, facilitando o processo de emissão de comprovante de participação dos professores e da ata de defesa dos trabalhos, contribuindo para o melhor desempenho das tarefas, facilitando e agilizando a atividade realizada. Para a construção deste sistema, foram utilizadas as seguintes ferramentas: Java, por meio da IDE NetBeans, Android Studio (ambiente de desenvolvimento para a codificação da aplicação mobile), MySql WorkBench (sistema gerenciador de banco de dados) (GOMES, 2016).

2.2 Tecnologias Envolvidas

Para que haja o controle de bancas de TCC, foi desenvolvido um software dividido em duas camadas, o *backend* e o *frontend*. O *backend* é uma estrutura responsável pela operação do sistema, o qual trata todas as requisições realizadas pelo *frontend*, sendo este último o responsável pela parte visual do sistema, na qual haverá a interação do usuário.

Nos próximos tópicos, apresentam-se as tecnologias utilizadas no sistema proposto.

2.2.1 Spring Framework

Trata-se de um framework *open-source*, que fornece um modelo abrangente de programação e configuração para aplicativos em Java, baseado nos padrões de projeto: inversão de controle e injeção de dependência. Criado por Rod Johnson e descrito em seu livro "*Expert One-on-One: JEE Design e Development*", este framework oferece diversos módulos para serem utilizados de acordo com a necessidade do projeto, como por exemplo, módulos voltados para desenvolvimento web, persistência de dados, acesso remoto, segurança, entre outros (SPRING, 2022).

2.2.2 Spring Boot

Desde sua criação, o ecossistema Spring cresceu muito, fornecendo funcionalidades já desenvolvidas, as quais são de grande auxílio ao desenvolvedor, exigindo apenas que este adicione as dependências necessárias ao seu projeto. Contudo, com este crescimento, surgiram alguns problemas, como por exemplo: a configuração do projeto já não é tão simples quanto antes, pois quanto mais módulos adicionados mais dependências eram necessárias em sua configuração (BOAGLIO, 2017).

O projeto Spring Boot além de resolver estas questões, ainda nos apresenta um novo modelo de desenvolvimento, mais simples e direto, sem propor novas soluções para problemas já resolvidos, mas sim alavancando as tecnologias existentes presentes no ecossistema Spring de modo a aumentar significativamente a produtividade do desenvolvedor (DEVMEDIA, 2015).

De acordo com SPRING (2022) dentre as características do Spring Boot, destacam-se:

- Criação de aplicativos Spring autônomos.
- Incorporação de servidores WEBs.
- Fornecimento de *starter's* opinativos.
- Configuração automática de bibliotecas Spring e de terceiros.
- Funcionalidades para ambiente de produção como métricas, *health checks* e configurações externalizadas.
- Dispensar a necessidade de configuração XML e geração de código.

2.2.3 Spring Data JPA

O Spring Data é um projeto de alto nível que unifica e facilita o acesso à diferentes tipos de persistência de dados, seja um banco de dados relacional ou NoSQL. Sendo assim, o Spring Data JPA é um integrante desta família Spring Data, no qual facilita a implementação de repositórios baseados em JPA. Necessita apenas do desenvolvedor criar as interfaces de repositórios e o Spring se encarrega de criar toda a implementação automaticamente (SPRING, 2022).

2.2.4 Spring Security

É o padrão para segurança de aplicativos baseados em Spring, providenciando um sistema de autenticação e autorização de alto nível e altamente customizável (SPRING, 2022).

De acordo com SPRING (2022), suas principais características, são:

- Sistema de autenticação.
- Sistema de autorização.
- Proteção contra ataques.
- Integração com a Servlet API.
- Integração opcional com o Spring Web MVC.
- Fácil instalação através de um starter Spring Boot.

2.2.5 JSON

Especificado por Douglas Crockford em 2000, o JSON é basicamente um formato de troca de dados entre sistemas, caracterizado por sua simplicidade e facilidade de leitura e escrita, o que fez se tornar uma estrutura muito popular (JSON, 2022).

Além disso, considerado um formato totalmente independente de linguagem de programação, por utilizar convenções familiares aos programadores da família C de linguagens, o JSON tem sido altamente adotado no desenvolvimento de APIs. Principalmente pelo fato de ser altamente produtivo em desenvolvimento de serviços e em aplicações distribuídas (JSON, 2022).

Esta estrutura é constituída em duas estruturas, seja ela um conjunto de pares nome/-valor ou uma lista de valores, como demonstra a Figura 1.

Figura 1: Exemplo de um JSON

```
{
    "name": "John",
    "age": 22,
    "phones": ["111", "222", "333"]
}
```

Fonte: Do autor

2.2.6 Angular

É uma plataforma de desenvolvimento, construída em TypeScript pela equipe Angular do Google. Basicamente, esta plataforma é utilizada para construção de SPA, ou seja, aplicativos de página única, que são aplicações cuja funcionalidade está concentrada em uma única página, logo, apenas o conteúdo principal é atualizado na página de forma assíncrona, mantendo toda a estrutura da página estática e sem a necessidade de recarregar toda a página ou redirecionar para uma nova página (GOOGLE, 2022).

2.2.7 TypeScript

Desenvolvida em 2012 pela Microsoft, o TypeScript é uma linguagem de programação *open-source* fortemente tipada que se baseia em JavaScript. Assim, por ser tipada, o TypeScript promove uma definição dos objetos e variáveis, fazendo com que tenha

uma boa legibilidade e compreensão do código, e consequentemente, melhorando a manutenibilidade e escrita do código.

De acordo com Stackoverflow (2018), TypeScript foi considerada pelo público a 4ª linguagem mais amada, além disso, está entre as 15 linguagens mais populares, segundo O'Grady (2019).

2.2.8 PostgreSQL

PostgreSQL é um poderoso Sistema de Gerenciamento de Banco de Dados de Objeto Relacional (SGBDOR) de código aberto, que demonstra uma boa economia e alto desempenho oferecido pelo SGBD, garantindo estabilidade e segurança quando se trata de intenso fluxos de dados, e isso não o torna extramamente caro. De acordo com UOL (2020), o Detran do Ceará conseguiu economizar em torno de 1,7 milhão de reais após realizar a migração para o PostgreSQL.

2.2.9 Git

Projetado e desenvolvido por Linus Torvalds, Git é um sistema de controle de versão distribuído gratuito e de código aberto. Trazendo como grande diferencial, o seu modelo de ramificação, que permite criar várias ramificações locais independentes uma das outras, sem a necessidade de acesso ao repositório remoto. Além de tudo, o Git é uma ótima ferramenta para trabalho em equipe em um projeto de desenvolvimento, no qual é possível vários desenvolvedores trabalharem no mesmo código, ao mesmo tempo, e quando houver a necessidade de mesclagem de código é muito simples realizar (GIT, 2022).

2.2.10 GitHub

GitHub é uma plataforma de hospedagem de código-fonte e arquivos com controle de versão utilizando o Git. Mundialmente usado, chega a ter mais de 83 milhões de desenvolvedores, mais de 4 milhões de organizações e mais de 200 milhões de repositórios. Sendo possível também, automatizar a compilação, os testes e as pipelines de implantação, por meio do GitHub Actions, o qual se trata de uma plataforma de integração contínua e entrega contínua (CI/CD) (GITHUB, 2022).

3 METODOLOGIA

Para construção do sistema proposto, foi realizado inicialmente um levantamento dos principais requisitos no processo de elaboração de TCC até sua defesa na UFES - Campus de Alegre. A etapa de levantamento de requisitos foi realizada junto ao professor orientador, o qual foi utilizado o documento "Trabalho de Conclusão de Curso: Regulamento e Normas para Elaboração", no qual se encontram os regulamentos e normas para elaboração do TCC (UFES, 2012). Desta forma, pode se caracterizar a pesquisa deste trabalho com o propósito exploratório e abordagem qualitativa.

Através dos requisitos levantados, o sistema foi implementado utilizando a metodologia de desenvolvimento descrita nas subseções seguintes.

3.1 Metodologia do desenvolvimento

Segundo Schwaber e Sutherland (2013), a metodologia *Scrum* é utilizada desde 1990, sendo muito adotada na gerência de trabalhos em produtos complexos, o qual é possível empregar em vários processos e técnicas. É indubitável a eficácia e agilidade relativa as práticas de gerenciamento de um produto utilizando o *Scrum*, de modo que é factível dar prosseguimento a melhora do produto, equipe e ambiente de trabalho (SCHWABER; SUTHERLAND, 2013).

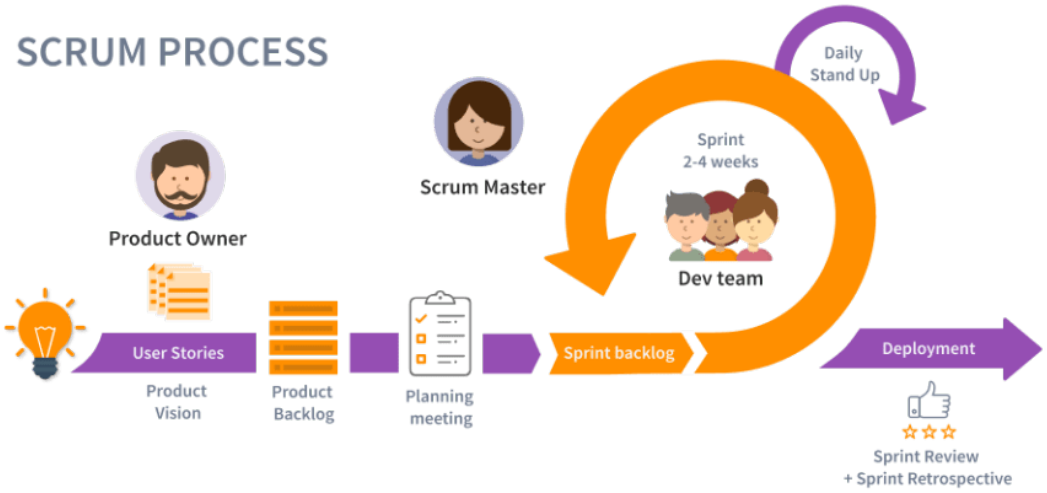
De acordo com Schwaber e Sutherland (2013), o *Scrum* possui três pilares importantes, sendo eles:

- Transparência: Boa visibilidade sobre as atividades do projeto, e comum entendimento sobre estes aspectos.
- Inspeção: Inspecionar os artefatos *Scrum* e o progresso em direção ao objetivo, a fim de detectar variações ou problemas indesejados.

- Adaptação: Etapa após a inspeção em casos de desvios ou problemas que impactam o objetivo, sendo necessário ajustar o processo ou o material que está sendo produzido.

A equipe *Scrum*, ou também conhecida com Time *Scrum*, são as pessoas responsáveis pela concepção, construção e testes do produto final do projeto. Esta equipe é formada por três papeis: o PO (*Product Owner* - Proprietário do produto), o *Scrum Master* (Mestre do *Scrum*) e o *Development Team* (Time de desenvolvimento). O PO não é apenas responsável em entender o cliente, mas também garantir a entrega máxima do Time *Scrum*, mantendo a integridade conceitual das funcionalidades, *bugs* e melhorias, e assim estabelecendo uma visão definida para o projeto. o *Scrum Master* é responsável em garantir que os fluxos e definições do *Scrum* estão sendo seguidos respeitando seus valores. E por último mas não menos importante, o *Development Team*, que são os profissionais responsáveis em realizar a entrega de um incremento do produto. O Time *Scrum* deve ser multifuncional, de modo que a equipe se autogerencie, a fim de determinar a melhor maneira de realizar o trabalho e consiga atingir a meta estabelecida pelo PO (SCHWABER; SUTHERLAND, 2013).

Figura 2: Processo *Scrum*



Fonte: Tuleap (2022)

Assim como ilustrado na Figura 2, o processo do *Scrum* pode ser definido nas seguintes etapas:

1. Definição do *Backlog*: Listar todas as entregas, marcos e objetivos do projeto.

2. *Planning Meeting*: Realizar uma reunião de planejamento antes de iniciar o clico da *Sprint*, desmembrando as atividades presentes no *Backlog*, e formando o *Sprint Backlog*.

3. *Sprint*: Ciclo de 2 à 4 semanas de desenvolvimento das atividades levantadas no *Sprint Backlog*.

4. *Daily Stand Up*: Realizar reuniões diárias de acompanhamento das tarefas. Em cenários que as equipes se encontram presencialmente, muito das vezes estas reuniões podem ser realizadas em pé, para não estender muito sua duração.

5. Fim da *Sprint*: Ao fim de cada *Sprint*, realizar a entrega dos itens finalizados, e assim concluir com uma reunião de retrospectiva (*Sprint Review*).

6. Realizar o planejamento de uma nova *Sprint*.

Em vista dos fatos abordados sobre o *Scrum*, suas vantagens e bons resultados, optou-se trabalhar com esta metodologia para o gerenciamento do projeto em questão, porém por se tratar de uma metodologia adotada para um trabalho em equipe, foi necessário realizar algumas adaptações. Sendo assim, pode-se considerar que a equipe *Scrum* neste trabalho será composta pelo professor orientador e o aluno orientando, sendo que o orientador terá como responsabilidade o papel do PO, por deter maior conhecimento do negócio, auxiliando nas definições dos itens do *Backlog* e suas prioridades, já o aluno orientando terá como responsabilidades o papel de *Scrum Master* e do *Development Team*, para isso, o aluno como *Scrum Master* será responsável em garantir que as regras do *Scrum* estão sendo seguidas, e como *Development Team*, terá como responsabilidade a parte de codificação e testes do sistema, no qual transformará o item do *Backlog* em um produto funcional.

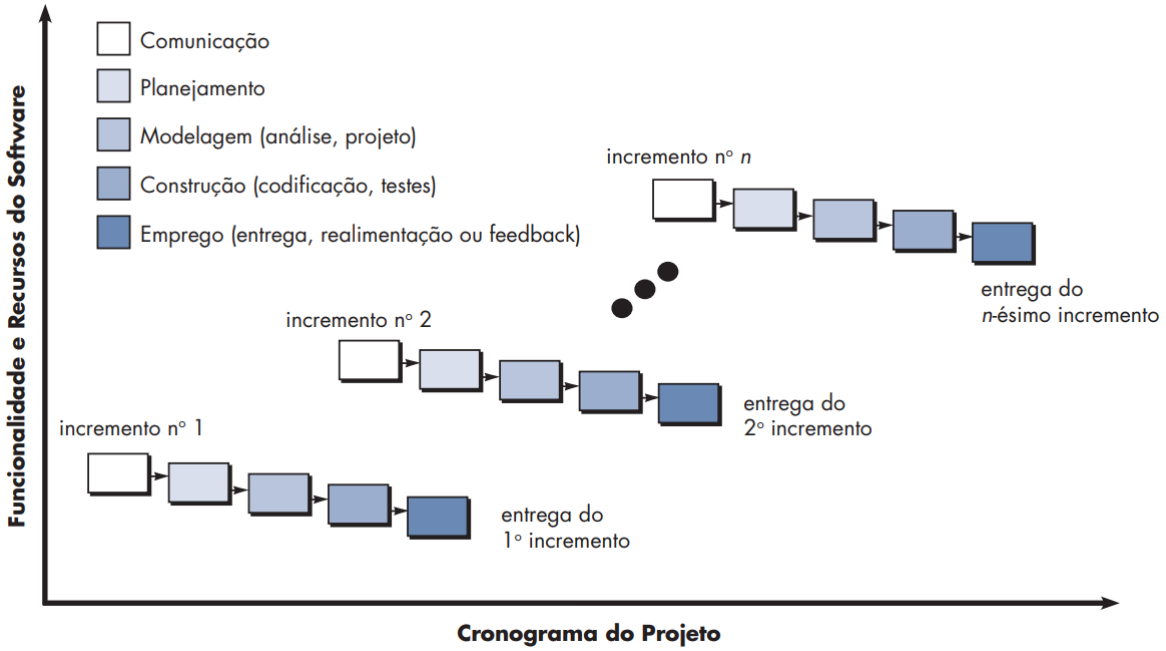
Agora no que se trata aos processos realizados pela metodologia *Scrum*, para o desenvolvimento do sistema deste trabalho, após as definições dos itens do *Backlog*, o PO juntamente com a equipe, irá alocar as atividades para serem realizadas em uma *Sprint*, que terá um período de 2 semanas. No decorrer da *Sprint*, não ocorrerá reuniões diárias, pois tanto o *Scrum Master* e o *Development Team* se refletem na mesma pessoa, logo esse processo diário será mais uma análise da situação dos itens levantados para desenvolvimento na *Sprint*. Ao final de cada Sprint, será feito uma reunião juntamente com o PO do projeto, para revisar os itens feitos e os possíveis casos de impedimentos que ocorreram no decorrer da *Sprint*, além disso, nessa mesma reunião serão levantados os próximos itens da *Sprint* seguinte, reiniciando o processo.

3.2 Análise dos requisitos

A qualidade no processo de elicitação e análise de requisitos é de suma importância, para que os requisitos sejam claros, inequívocos, de fácil compreensão, completos e consistentes. Pois um erro de concepção pode acarretar em um custo, e dependendo do momento de surgimento deste erro, o custo pode ser maior ainda, por exemplo, o custo de um erro na fase de análise é menor que o custo de um erro na fase de implementação, e o custo do erro na fase de implementação é menor que o custo de um erro da fase de implantação, o qual este pode ter um valor extremamente alto (MACHADO, 2018).

O processo de elicitação de requisitos pode ser realizado por meio de várias técnicas, por exemplo, entrevistas, questionários, *brainstorms*, leitura de documentos, observação e análise (etnografia), diagramas UML, prototipação, entre outros (SOMMERVILLE, 2015). Portanto, para a elicitação e análise dos requisitos deste trabalho, as técnicas utilizadas, foram: entrevistas, leituras de documentos e diagramas UML.

Figura 3: O modelo incremental



Fonte: Pressman (2011)

Em alguns casos, no início de um projeto, os requisitos de um *software* podem estar razoavelmente bem definidos, contudo, nem sempre o uso de um processo puramente linear é utilizado, dado que seja necessário o rápido fornecimento de uma determi-

nada funcionalidade no sistema, no qual este nem esteja finalizado por completo. Desta maneira, tais funcionalidades previamente entregues, podem ser melhoradas e expandidas em versões posteriores do *software*. Com isso, visando atender essa necessidade, é comumente utilizado o modelo de processo incremental, ilustrado na Figura 3, a qual mostra um processo projetado para ser desenvolvido com entregas incrementais (PRESSMAN, 2011).

Em virtude disso, optou-se para o desenvolvimento do sistema proposto, o modelo incremental, pois apesar do processo de desenvolvimento já se iniciar com alguns requisitos levantados, será necessário sempre voltar na etapa de planejamento para revalidar e implementar novas funcionalidades, ou até então melhorar ou expandir funcionalidades já entregues, gerando assim um incremento, ou seja, uma nova versão do *software*.

3.2.1 Descrição dos usuários

No ponto de vista da modelagem, os atores dos sistemas são aqueles que interagem com o sistema ou algum artefato produzido pelo sistema, durante a execução de uma determinada atividade, rotina ou função. No caso do sistema proposto, os atores, também identificados como usuários do sistema, estão descritos na tabela 1.

Tabela 1: Usuários do sistema

Usuários	**Descrição**
Supervisores de TCC	Professores da disciplina de TCC
Orientadores	Professores orientadores de alunos matriculas em TCC
Alunos orientados	Alunos matriculados em TCC
Professores membros da banca	Professores que serão membros da banca de uma apresentação de TCC
Administrador	Usuário responsável pelo gerenciamento do sistema

Fonte: Do autor

3.2.2 Requisitos funcionais

De acordo com Sommerville (2015), os Requisitos Funcionais (RF) representam o que um *software* faz, em termos de tarefas e serviços. Eles dependem do tipo de *software*,

os possíveis usuários e a maneira pela qual foi adotada sua escrita. Ou seja, é basicamente as definições sobre o que um sistema deve fornecer, como reagir, e como comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem definir ações que o sistema não deve fazer.

Após a finalização da etapa de elicitação e análise de requisitos, foram definidos dois processos principais no sistema, a qualificação e a defesa. A qualificação se refere a etapa introdutória do TCC, o planejamento do trabalho, esta etapa ocorre durante a disciplina de TCC 1, no caso do processo do Departamento de Computação, em que foi baseado. A defesa, se refere a etapa de realização das atividades propostas no planejamento do trabalho, ou seja, tudo aquilo que foi planejado na primeira etapa de qualificação, será feito nesta etapa que, da mesma forma que a etapa anterior, ocorre numa disciplina específica no processo de referência.

Desta forma, os requisitos funcionais (RF) foram divididos em: Requisitos Funcionais dos Processos de Qualificação e Defesa e Requisitos Funcionais Específicos.

Requisitos Funcionais dos Processos de Qualificação e Defesa:

- RF01 - Manter cadastro de alunos matriculados: O professor de TCC poderá incluir, editar, excluir e importar os alunos matriculados em TCC.
- RF02 - Manter cadastro de projeto de TCC: O professor orientador poderá incluir, editar, cancelar projetos de TCC, os projetos de TCC devem ser incluídos na situação "Em andamento".
- RF03 - Visualizar projetos de TCC cadastrados: O professor de TCC poderá visualizar os projetos de TCC cadastrados.
- RF04 - Anexar documento de TCC: O professor orientador e o aluno orientando poderão anexar o documento de TCC em PDF ou DOCX.
- RF05 - Manter agenda de apresentações do TCC: O professor de TCC poderá incluir, editar e excluir agendas disponíveis de apresentação de TCC.
- RF06 - Marcar apresentação: O professor orientador irá definir dentre os períodos de tempo disponíveis, quando será a defesa do TCC, alterando a situação do projeto de TCC em questão, sinalizando que o mesmo deverá ser apresentado.
- RF07 - Solicitar os membros à banca: O professor orientador poderá solicitar aos professores cadastrados no sistema, a participação na banca, com isso o

sistema enviará um e-mail notificando os professores solicitados a confirmarem.

- RF08 - Manter cadastro de agenda de disponibilidade: Os professores deverão informar quais os dias e horários estarão disponíveis para participar nas bancas de TCC.
- RF09 - Confirmar a solicitação de participação em banca de TCC: Os professores membros de banca, irão confirmar o convite de participação para uma banca de TCC eletiva.
- RF10 - Iniciar sessão para inclusão de avaliação de TCC: O professor de TCC poderá iniciar o processo de avaliação de TCC, e disponibilizar a ficha de avaliação para ele, para o professor orientador, e para os professores membros da banca, caso haja.
- RF11 - Postar observações sobre a defesa do TCC: O professor de TCC, o professor orientador e os membros da banca poderão, após a defesa de TCC, postar o documento com as alterações sugeridas.
- RF12 - Analisar a correção: O orientador, professor de TCC e os membros da banca, poderão analisar a correção, dizendo se atende ou não as alterações solicitadas.
- RF13 - Fornecer as notas: O orientador, professor de TCC e os membros da banca, poderão fornecer as notas.
- RF14 - Postar situação do aluno orientado: O professor de TCC irá disponibilizar a nota final do aluno orientado e informar se está aprovado ou reprovado em TCC.
- RF15 - Solicitar assinatura dos membros: O professor de TCC irá postar os documentos para assinatura.

Requisitos Funcionais Específicos:

- RF16 - Manter cadastro de professores: O administrador e o professor de TCC poderão incluir, editar, excluir e importar via CSV professores de TCC, orientadores ou membros da banca.
- RF17 - Editar dados pessoais: Todos os usuários do sistema poderão editar seus próprios dados pessoais.

- RF18 - Emitir relatório de TCC's defendidos por período: Todos os professores poderão emitir relatório de TCC's defendidos por período, contendo as informações do título do TCC, do aluno orientado, professor orientador, membros da banca, ano/período, se é qualificação ou defesa, e a situação do TCC.
- RF19 - Emitir relatório de TCC's por situação: Todos os professores poderão emitir relatório de TCC's por situação.
- RF20 - Incluir administrador: Um administrador poderá incluir outros administradores.
- RF21 - Manter documentos em geral: O professor de TCC, orientador e membro da banca, poderão anexar documentos em geral.
- RF22 - Manter cadastro de modelos de documento: O professor de TCC poderá cadastrar modelos de documentos utilizados no processo de qualificação ou defesa de TCC.
- RF23 - Manter cadastro de modelos de avaliação: O professor de TCC poderá cadastrar modelos de avaliação utilizados no processo de qualificação ou defesa de TCC.
- RF24 - Manter cadastro de áreas de TCC: O professor de TCC e o administrador poderão cadastrar áreas de TCC, sendo uma área de TCC definida por faculdade e curso.

3.2.3 Requisitos não funcionais

Requisitos Não Funcionais (RNF) são aspectos que não estão diretamente relacionados às funcionalidades específicas oferecidas pelo sistema aos usuários. Eles englobam propriedades do sistema, tais como confiabilidade, tempo de resposta e utilização de recursos. Esses requisitos são igualmente cruciais para assegurar a eficiência, segurança e experiência do usuário, sendo fatores essenciais para o êxito do projeto (SOMMERVILLE, 2015).

A seguir, estão os tópicos que apresentam os requisitos não funcionais definidos para o sistema proposto:

- RNF01 - Armazenamento Seguro de Senhas: O sistema deve assegurar o armazenamento seguro das senhas dos usuários, através do uso de técnicas de criptografia.
- RNF02 - Controle de Acesso: Apenas usuários autenticados devem ter acesso aos dados e funcionalidades do sistema.
- RNF03 - Usabilidade e Interface Amigável: A interface do usuário deve ser intuitiva e amigável, permitindo uma navegação fácil e sem dificuldades.
- RNF04 - Responsividade: O sistema deve ser responsivo, adaptando-se a diferentes tamanhos de tela.
- RNF05 - Compatibilidade com Navegadores Modernos: O sistema deve ser compatível com diferentes navegadores modernos, como Chrome, Firefox e Edge.
- RNF06 - *Lazy-Loading* para Otimização dos Módulos: O sistema deve adotar a técnica de *Lazy-Loading* do Angular para carregar módulos somente quando acessados, otimizando o tempo de carregamento inicial da aplicação.
- RNF07 - Facilidade de Manutenção e Atualização: O sistema deve ser projetado e desenvolvido visando à facilidade de manutenção e atualizações futuras, com código-fonte e estrutura bem organizados.

3.2.4 Regras de negócio

De acordo com Leite e Leonardi (1998), regras de negócio são, basicamente, definições de como as operações de um negócio são realizadas em uma organização, ditando suas maneiras e requisitos necessários para obter seu sucesso.

Portanto, as regras de negócio (RN) do sistema proposto são:

- RN01 - Cadastro de projetos de TCC: Só é possível cadastrar um projeto de TCC com a definição do(s) aluno(s) a ser orientado, e este não deve estar associado a outro projeto de TCC em andamento.
- RN02 - Confirmar solicitação à banca: Só é possível confirmar sua participação à banca, após ter definido sua disponibilidade.

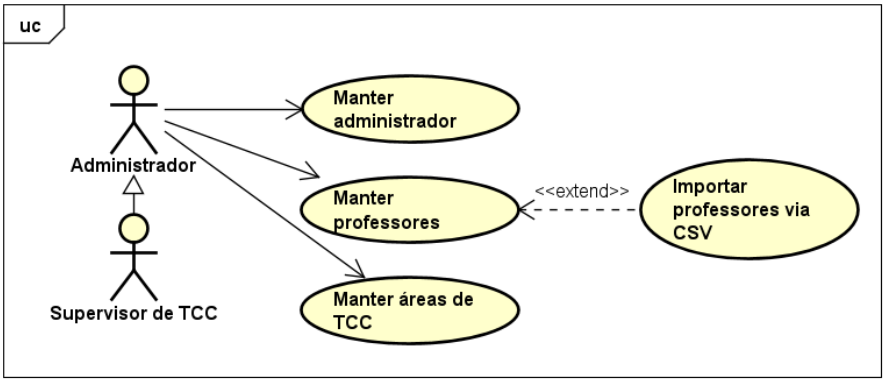
- RN03 - Agendamento de apresentação: Só é possível marcar a apresentação de TCC dentro da agenda disponível dos professores participantes da banca, e também dentro da agenda de apresentações definida pelo professor de TCC, além disso, é necessário que todos os professores solicitados tenham confirmado sua participação à banca.
- RN04 - Visualização da nota final: O aluno não poderá visualizar sua nota final, até o seu fechamento.
- RN05 - Aprovação em TCC: O aluno será aprovado apenas se a sua nota final estiver igual ou maior a média definida em sua faculdade.
- RN06 - Sessão de avaliação de TCC: A sessão só pode ser iniciada após o agendamento da apresentação estiver feito e o documento final do TCC estiver anexado.
- RN07 - Lançamento da nota final: A nota final do TCC só poderá ser lançada quando todos os professores necessários tiverem concluído a avaliação do trabalho.
- RN08 - Edição no modelo de avaliação: É importante replicar as edições nos modelos de avaliação para as avaliações em andamento. Isso significa que ao adicionar ou remover um item de avaliação, ele deve ser adicionado ou removido nas avaliações associadas ao modelo. Da mesma forma, se houver uma edição no item, essas mudanças devem ser feitas nas avaliações correspondentes, sem alterar o valor já informado.

3.2.5 Casos de uso

Segundo Sommerville (2015) os casos de uso representam todas as possíveis interações descritas nos requisitos. Os atores são representados por bonecos de palito, podendo ser pessoas ou sistemas. As classes de interação são as elipses. E por fim as linhas são as interações entre os atores e as classes de interação.

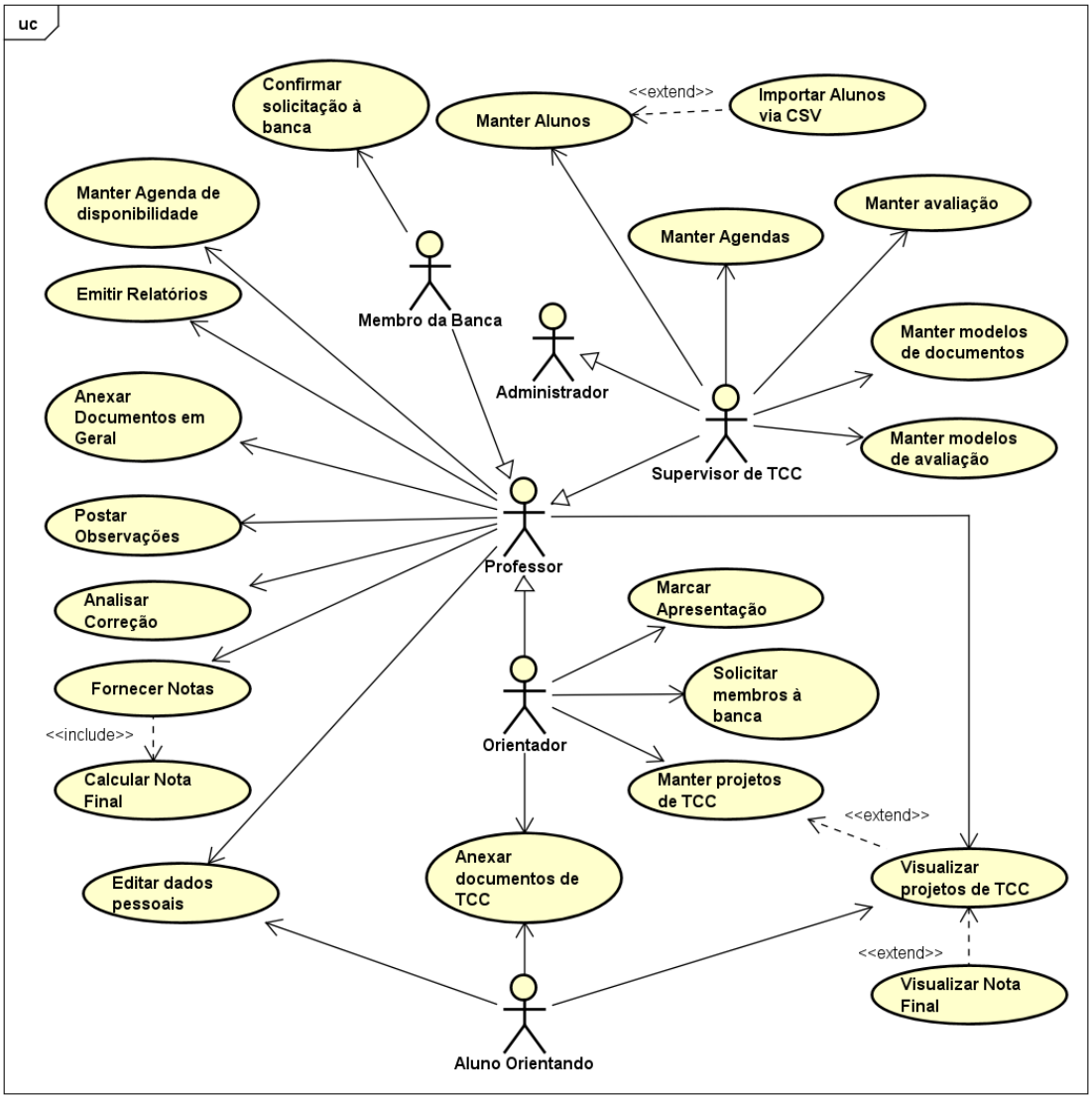
Os diagramas de casos de uso foram divididos em dois diagramas, um sendo os casos de uso do administrador e o outro sendo os casos de uso dos restantes dos usuários, sendo assim, as Figuras 4 e 5 se referem aos casos de uso definidos para o desenvolvimento do sistema proposto.

Figura 4: Casos de uso referente ao administrador



Fonte: Do autor

Figura 5: Casos de uso referente ao professor de TCC, orientador, membros da banca e aluno orientando

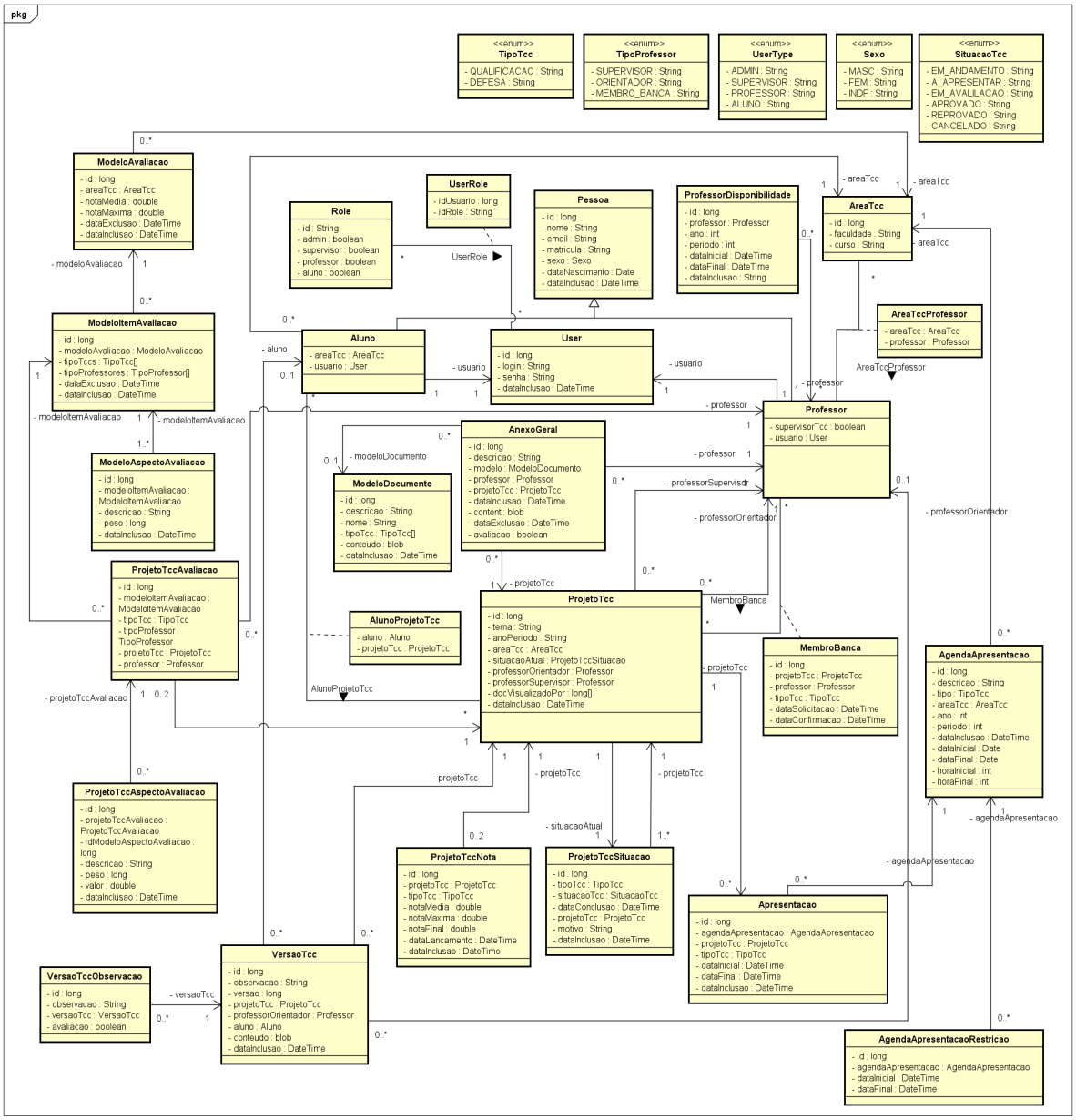


Fonte: Do autor

3.2.6 Diagrama de classe

Um diagrama de classe representa a estrutura estática de um sistema, apresentando suas classes, atributos, operações e suas associações. Ou seja, as classes são definições gerais dos tipos de objeto do sistema, seus atributos são suas características e suas operações são suas ações ou métodos realizados pela classe. A associação é um link que indica algum relacionamento entre duas classes (SOMMERVILLE, 2015). Sendo assim, a Figura 6 mostra o diagrama de classes construído no etapa de análise e levantamento de requisitos.

Figura 6: Diagrama de classe



Fonte: Do autor

É importante lembrar que, apesar dos diagramas de classe serem parecidos com modelos semânticos de dados, eles não são para representar a estrutura de tabelas de banco de dados, pois a UML supõe um processo de desenvolvimento orientado a objetos, modelando os dados utilizando os objetos e seus relacionamentos.

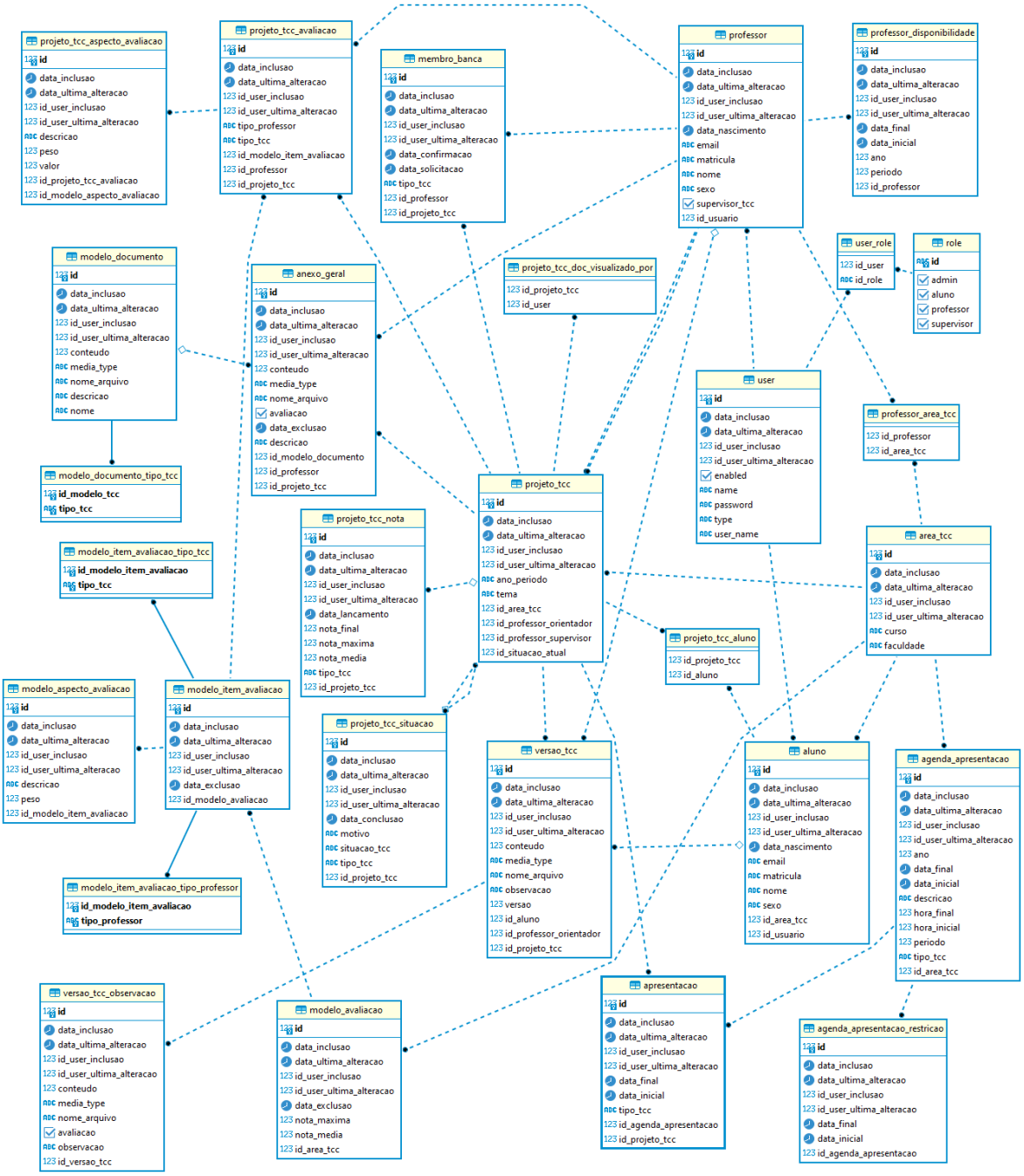
3.3 Modelagem do banco de dados

O Modelo Entidade Relacionamento (MER), desenvolvido por Peter Chen, e publicado em seu artigo em 1976, é um modelo para estruturação de tabelas de um banco de dados relacional, com o objetivo de, facilitar o projeto de banco de dados, especificando sua estrutura lógica geral (LAENDER; FLYNN, 1993). Desta forma, a Figura 7 mostra um modelo de dados físico levantado em cima dos requisitos elicitados.

De acordo com Heuser (2009), o modelo de dados conceitual pode ser definido como um modelo de dados abstrato, que identifica as necessidades de estruturação independentemente de implementação, capturando formalmente os requisitos de informação de um banco de dados. Por meio deste modelo, é descrito as entidades importantes e seus respectivos relacionamentos, explorando os conceitos do domínio do problema o qual será tratado pela implementação do sistema. Já o modelo físico, representado na Figura 7, contém as definições detalhadas da estrutura física do banco de dados. Ele engloba informações cruciais, como os tipos de dados utilizados, os tamanhos dos campos, os relacionamentos, entre outras informações. Sendo assim essas especificações fornecem as bases para a implementação concreta do banco de dados.

O modelo exibido na Figura 7 foi gerado pelo DBeaver, utilizando sua ferramenta de criação de diagramas de Entidade-Relacionamento. O DBeaver é uma solução aberta de gerenciamento de banco de dados e cliente SQL que proporciona uma interface gráfica unificada e intuitiva (DBEAVER, 2023).

Figura 7: Modelo físico do banco de dados



Fonte: Do autor

Por meio da elicitação e análise dos conceitos do domínio do problema, o processo de modelagem do banco de dados, é de fato, fundamental no desenvolvimento de um sistema, pois descreve todas as entidades do banco e seus respectivos relacionamentos, oferecendo uma maior visibilidade e compreensão do problema.

3.3.1 Descrição dos dados

Através do modelo de dados apresentado anteriormente, é demonstrado suas entidades, seus relacionamentos e seus atributos, com seus respectivos tipos, porém não é informado uma descrição sobre as colunas de cada tabela. Desta forma, se fez necessário a descrição de cada entidade, detalhando seus atributos por meio de uma tabela, que contempla o nome do atributo, seu tipo, uma coluna informando se é chave primária ou estrangeira, e uma breve descrição.

Sendo assim, a tabela 2 detalha sobre os atributos e suas características da entidade agenda_apresentacao.

Tabela 2: Descrição de dados da entidade agenda_apresentacao

agenda_apresentacao			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
ano	Inteiro		Ano
data_final	Data		Data limite máximo
data_inicial	Data		Data limite mínimo
descricao	Texto		Descrição
hora_final	Inteiro		Hora limite máximo
hora_inicial	Inteiro		Hora limite mínimo
periodo	Inteiro		Período
tipo_tcc	Texto		Tipo do TCC
id_area_tcc	Inteiro	FK	Identificador da área de TCC

Fonte: Do autor

São apresentadas no Apêndice A a descrição das demais tabelas do modelo de dados.

4 DESENVOLVIMENTO

Após a conclusão da fase de levantamento e análise de requisitos, bem como a realização de estudos sobre as tecnologias escolhidas para a implementação do projeto, destacando principalmente Angular, Spring Framework e PostgreSQL, deu-se início à fase de desenvolvimento do sistema.

4.1 Arquitetura do sistema

Segundo Pressman (2011), a arquitetura de um sistema é sua estrutura, seus componentes, suas propriedades e suas estruturas de dados. Nela que se define as relações entre os padrões de projeto, estilos arquiteturais e estruturais do *software* (PRESSMAN, 2011).

Projetar a arquitetura de um *software* é de suma importância, pois dependendo de qual arquitetura escolhida, pode-se influenciar na performance, qualidade, facilidade de manutenção e escalabilidade de um sistema, desta forma, esta decisão tem um grande impacto no sucesso do projeto (SOMMERVILLE, 2015).

Com isso, para o sistema proposto, definiu-se utilizar a arquitetura REST (Representational State Transfer - Transferência de Estado Representacional), pois se trata de uma arquitetura que permite construir aplicações *web* de forma simples e rápida, com facilidade em sua execução e alto aproveitamento da infraestrutura *web* (TOTVS, 2020).

Nas próximas subseções será abordado com mais detalhes os conceitos relacionados a arquitetura do sistema proposto.

4.1.1 API

API (Interface de Programação de Aplicações) é basicamente uma interface definida por meio de rotinas e padrões estabelecidos por uma aplicação qualquer, para que ou-

tras aplicações possam utilizar de suas funcionalidades. Ou seja, é uma interface de comunicação que um sistema pode oferecer para que outros utilizem de suas funcionalidades, dados ou recursos, sem a necessidade de conhecimento de como tais funções foram implementadas. Porém para sua utilização, será necessário saber como utilizar tais serviços e como fará a interação com estes serviços (ORENSTEIN, 2000).

4.1.2 REST

REST (Transferência de Estado Representacional), é um estilo de arquitetura de *software* que é definido um conjunto de restrições a serem utilizadas para a criação de *web services* (serviços web), no qual estes permitem a interoperabilidade entre sistemas de plataformas diferentes. Desenvolvido no ano de 2000 por Roy Fielding, a arquitetura REST adota um paradigma cliente-servidor, em que as requisições partem inicialmente do cliente e são respondidas pelo servidor (FIELDING, 2000).

As restrições definidas nesta arquitetura tem como objetivo determinar a forma pelo qual padrões como HTTP (Protocolo de Transferência de Hipertexto) e URI (Identificador Uniforme de Recurso) deveriam ser modelados. Os *web services* quando são desenvolvidos utilizando o padrão arquitetural REST, são chamados de Web API ou simplesmente API REST, desta forma, se tais *web services* respeitam todos essas restrições em sua implementação, este é considerado RESTful (SALVADORI et al., 2015).

Segundo (FIELDING, 2000), as restrições específicas que compõe o estilo REST são:

- *Client-server* (Cliente-servidor): Em arquiteturas deste tipo, sua principal característica é a separação de dois interesses, sendo eles as preocupações da interface que interage com o usuário (cliente) e as preocupações de armazenamento de dados (servidor), assim melhorando a portabilidade do cliente em várias plataformas e melhorando a escabilidade do servidor, simplificando seus componentes.

- *Stateless* (Sem estado): Esta restringe uma interação na arquitetura cliente-servidor, em que a comunicação deve ser sem estado por natureza, de forma que cada requisição realizada pelo cliente possua todas as informações necessárias para que o servidor consiga entender a requisição realizada, assim, o estado da sessão é mantido inteiramente no cliente.

- *Cache*: Esta restrição define que as respostas das requisições realizadas pelo cliente sejam rotuladas implícita ou explicitamente se podem ser armazenadas em cache ou não, sendo assim, se uma resposta puder ser armazenada em cache, o cliente terá o direito de armazená-la e reutilizar dos dados desta resposta em requisições equivalentes posteriores.

- *Uniform Interface* (Interface Uniforme): Ter uma interface uniforme entre seus componentes, significa basicamente propiciar uma arquitetura desacoplada e simplificada, como por exemplo, o protocolo utilizado em APIs REST é o HTTP e uma das uniformidades de interface seria considerar a semântica deste protocolo. Considerar tal semântica é utilizar seus códigos de *status* e verbo, que serão abordados com mais detalhes nos próximos tópicos. Segundo seu modelo arquitetural, REST é definido por algumas restrições de interface, sendo elas: identificação e manipulação de recursos, mensagens auto descritivas, e hipermídia.

- *Layered System* (Sistema em camadas): Define uma arquitetura composta por varios níveis hierárquicos, restringindo seus componentes de tal forma que nenhum deles possa enxergar outros componentes do seu mesmo nível hierárquico.

- *Code-on-demand* (Código sob demanda): REST dá a possibilidade que uma funcionalidade do cliente seja estendida baixando e executando um código, como por exemplo um código em javascript. Isso faz com que melhore a extensibilidade do sistema, porém reduz sua visibilidade e, portanto, esta restrição é considerada opcional no REST.

No protocolo HTTP, cada requisição está atrelada a um verbo, sendo este, responsável por indicar a ação a ser realizada por determinado recurso. No REST os verbos mais utilizados, são:

- GET: As requisições GET são utilizadas para buscar dados em uma API.

- POST: Para enviar dados, geralmente utilizado para operações de criação.

- PUT: Assim como POST, o PUT é possível enviar dados, porém geralmente ele é utilizado para operações de atualização.

- DELETE: Utilizado para remoção de um certo registro.

- PATCH: Utilizado para atualização de um registro, mas sem a necessidade de enviar toda a informação sobre aquele registro.

Assim como, cada requisição está atrelada a um verbo HTTP, ela também está atrelada a um código de resposta, estes códigos padronizam a maneira como o cliente é informado sobre o resultado de suas requisições. Segue as classes das respostas e alguns exemplos:

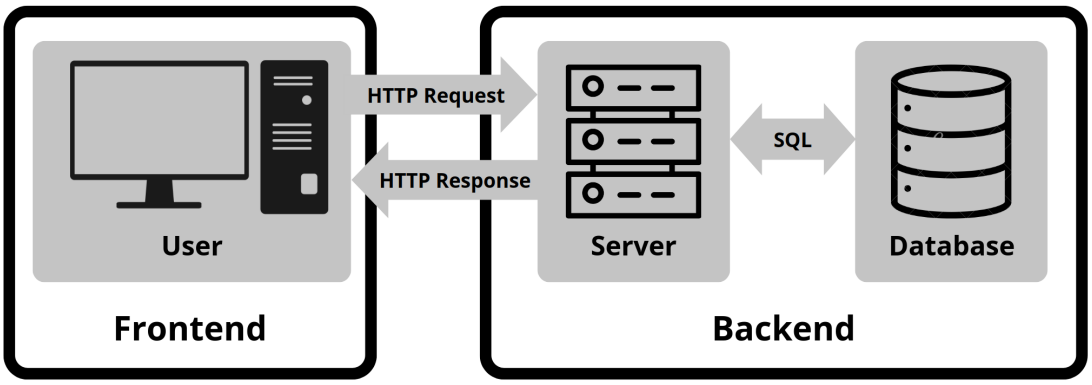
1. Respostas de informação (100-199):
    - 100: *Continue* (Prosseguir)
    - 101: *Switching Protocol* (Protocolo de comutação)
    - 102: *Processing* (Processando)
2. Respostas de sucesso (200-299):
    - 200: OK
    - 201: *Created* (Criado)
    - 202: *Accepted* (Aceito)
3. Redirecionamentos (300-399):
    - 300: *Multiple Choice* (Múltipla escolha)
    - 301: *Moved Permanently* (Movido permanentemente)
    - 302: *Found* (Encontrado)
4. Erros do cliente (400-499):
    - 400: *Bad Request* (Requisição inválida)
    - 401: *Unauthorized* (Não autorizado)
    - 402: *Payment Required* (Pagamento Requerido)
5. Erros do servidor (500-599):
    - 500: *Internal Server Error* (Erro do servidor interno)
    - 501: *Not Implemented* (Não implementado)
    - 502: *Bad Gateway* (Gateway inválido)

Devido a sua eficácia e legibilidade, a transição de dados usada como padrão em uma API REST é o JSON, embora existam outros formatos, como o XML, o JSON é o mais utilizado. Além disso, um dos principais motivos em que o JSON é considerado uma das estruturas mais utilizadas, é sua compatibilidade simples entre linguagens e frameworks existentes, tanto no *frontend* quanto no *backend* (PEZOA et al., 2016).

4.1.3 Desenho da arquitetura

Para o desenvolvimento do sistema proposto, a arquitetura definida seguirá o conceito comumente utilizado em sistemas com o *frontend* e *backend* separados. Assim, o *frontend* uma aplicação web desenvolvida em Angular e o *backend* uma aplicação Java desenvolvida em Spring Boot.

Figura 8: Arquitetura do sistema



Fonte: Do autor

Assim como ilustra a Figura 8, o *frontend* é apresentado como a aplicação web acessada pelo usuário, em que este irá realizar alguma ação, que resultará em uma HTTP *Request* (Requisição HTTP) ao *Server* (Servidor), assim a requisição será recebida e processada pelo servidor *backend*, que se comunicará com o *Database* (Banco de Dados) e por fim retornará uma HTTP *Response* (Resposta HTTP) ao *frontend*.

4.2 Backend

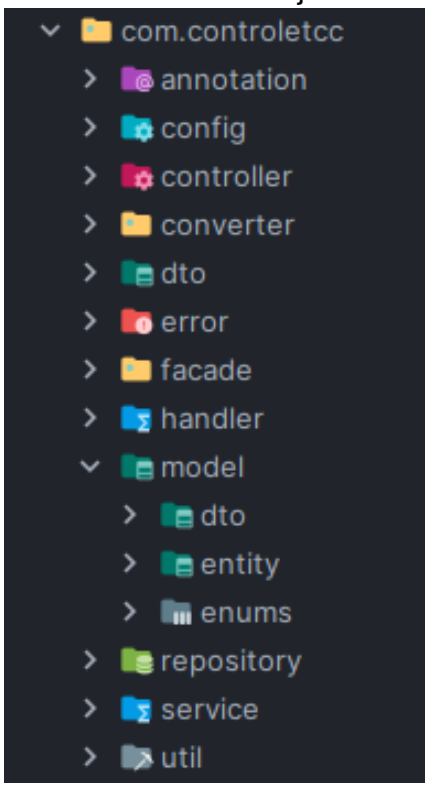
O *backend* desempenha um papel fundamental no desenvolvimento da aplicação, sendo responsável por processar a lógica de negócios, gerenciar os dados e fornecer funcionalidades e respostas aos usuários. Nesta seção, será abordado detalhadamente o *backend* da aplicação, incluindo seu projeto, estruturação, comunicação com

o banco de dados, tratamentos de segurança e tecnologias utilizadas.

4.2.1 Estrutura do Projeto

Na Figura 9, é possível observar a estruturação dos pacotes Java do projeto, e nos próximos tópicos, serão apresentadas suas respectivas definições.

Figura 9: Estrutura do Projeto - Backend



Fonte: Do autor

Estruturação dos pacotes Java:

- annotation: Interfaces Java de anotações.
- config: Classes de configuração do projeto, como autenticação e autorização.
- controller: Classes responsáveis em tratar as requisições HTTP.
- converter: Classes de conversão de tipos de dados durante o mapeamento de objetos.
- dto: Classes para transferência de dados.
- error: Exceções específicas do projeto para tratamento de erros.

- facade: Classes responsáveis por gerenciar as services.
- handler: Classes responsáveis no tratamento de eventos do projeto, como exceções.
- model: Pacote que contém as classes que representam as tabelas do banco.
  - dto: Classes de transferência de dados que refletem as entidades do banco.
  - entity: Classes que representam das entidades do banco.
  - enums: Enumeradores Java.
- repository: Interfaces responsáveis pela comunicação com o banco de dados.
- service: Classes responsáveis pelas regras de negócio.
- util: Classes utilitárias.

4.2.2 Conexão com o banco de dados

Através da biblioteca Spring Data JPA, foi realizada a configuração do banco de dados PostgreSQL. Inicialmente, foi criado o banco de dados localmente, e, em seguida, foram definidas as propriedades de configuração do Spring Data JPA no arquivo "application.properties", conforme apresentado na Figura 10. Tais configurações são necessárias para que o Spring Data JPA seja capaz de estabelecer a conexão com o banco de dados.

Figura 10: Propriedades de configuração do banco de dados

```
spring.datasource.url=jdbc:postgresql://localhost:5432/controletccdb
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.jpa.database-platform=postgres
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

Fonte: Do autor

A partir disso, foram implementados os *repositories* que estendem a interface JpaRepository do Spring Data JPA. Essa interface fornece diversos métodos prontos para realizar interações com o banco de dados, como inclusões, exclusões e buscas. Além disso, é possível construir métodos específicos de busca de forma simples usando essa interface, o que reduz consideravelmente a quantidade de código necessário.

Para utilizar essa interface, basta estendê-la no seu *repository*, fornecendo a classe da entidade e o tipo da chave primária dessa entidade (SPRING, 2022). A Figura 11 mostra um exemplo de implementação do repository da entidade “ModeloAvaliacao”.

Figura 11: Exemplo de repository

```
public interface ModeloAvaliacaoRepository extends JpaRepository<ModeloAvaliacao, Long> {

    Bruno Fonseca
    @Query(value = """
            SELECT distinct
                ma.id as id,
                concat(at.faculdade, ' - ', at.curso) as descricaoAreaTcc
            FROM ModeloAvaliacao ma
            JOIN ma.areaTcc at
                WHERE ma.dataExclusao is null
                AND at.id in :idAreaTccList
                AND (:id is null or ma.id = :id)
                AND (:idAreaTcc is null or at.id = :idAreaTcc)
            """
    )
    Page<ModeloAvaliacaoProjection> search(List<Long> idAreaTccList,
                                           Long id,
                                           Long idAreaTcc,
                                           Pageable pageable);

    1 usage  Bruno Fonseca
    boolean existsByAreaTccIdAndDataExclusaoNullAndIdNot(Long idAreaTcc, Long id);

    1 usage  Bruno Fonseca
    boolean existsByAreaTccIdAndDataExclusaoNull(Long idAreaTcc);

    1 usage  Bruno Fonseca
    ModeloAvaliacao getModeloAvaliacaoByAreaTccIdAndDataExclusaoNull(Long idAreaTcc);

}
```

Fonte: Do autor

Na Figura 11, podemos observar a criação de quatro métodos de consulta ao banco de dados. Três desses métodos são gerenciados pelo JpaRepository, o que simplifica a construção da consulta através do nome do método. Por exemplo, no método “getModeloAvaliacaoByAreaTccIdAndDataExclusaoNull”, é realizada uma consulta na entidade “ModeloAvaliacao” para buscar o registro que contenha o ID da área de TCC informado como parâmetro e cuja data de exclusão seja nula. No entanto, o primeiro método difere dos demais ao ser definido usando a anotação “@Query”. Essa abordagem se torna necessária quando é exigida uma busca mais complexa, que não pode ser expressa apenas pelo nome do método.

4.2.3 Endpoints

Os *endpoints* desempenham um papel crucial no processamento das requisições e no fornecimento de respostas às aplicações clientes. Ao receber as requisições HTTP, o servidor *backend* aciona os *endpoints* correspondentes, que são responsáveis por processar as informações solicitadas. Após o processamento adequado, o servidor envia uma resposta personalizada à aplicação cliente, contendo os dados ou resultados relevantes para a requisição realizada. Os *endpoints* são identificados por meio de Identificadores Uniformes de Recursos (URIs), que são sequências de caracteres que especificam a localização e o nome de um recurso na web (TECHTARGET, 2021).

No projeto em questão, foram implementadas classes que adotam a abordagem REST para expor os *endpoints*. Essas classes, denominadas *controllers*, possuem a responsabilidade de gerenciar as requisições e respostas de cada *endpoint*, utilizando o formato JSON como padrão para a troca de dados. A Figura 12 mostra um exemplo da *controller* referente ao “AnexoGeral”.

Figura 12: Exemplo de controller

```
@RestController
@RequestMapping("api/anexos-gerais")
@RequiredArgsConstructor
public class AnexoGeralController {
    private final AnexoGeralFacade anexoGeralFacade;

    Bruno Fonseca
    @PreAuthorize("hasAuthority('anexo-geral.read')")
    @GetMapping("{id}")
    public AnexoGeralDTO getById(@PathVariable Long id) { return anexoGeralFacade.getById(id); }

    Bruno Fonseca *
    @PreAuthorize("hasAuthority('anexo-geral.read')")
    @PostMapping("search/{idProjetoTcc}/{onlyAvaliacao}")
    public ListResponse<AnexoGeralProjection> search(
            @PathVariable Long idProjetoTcc,
            @PathVariable boolean onlyAvaliacao,
            @RequestBody AnexoGeralGridOptions options
    ) throws BusinessException {
        return anexoGeralFacade.search(idProjetoTcc, onlyAvaliacao, options);
    }

    Bruno Fonseca
    @PreAuthorize("hasAuthority('anexo-geral.create')")
    @PostMapping
    public AnexoGeralDTO insert(@RequestBody AnexoGeralDTO anexoGeralDTO) throws Exception {
        return anexoGeralFacade.insert(anexoGeralDTO);
    }

    Bruno Fonseca
    @PreAuthorize("hasAuthority('anexo-geral.delete')")
    @DeleteMapping("{id}")
    public void delete(@PathVariable Long id) throws Exception {
        anexoGeralFacade.delete(id);
    }

    Bruno Fonseca
    @PreAuthorize("hasAuthority('anexo-geral.read')")
    @GetMapping("download/{id}")
    public ArquivoDTO download(@PathVariable Long id) { return anexoGeralFacade.download(id); }
}
```

Fonte: Do autor

4.2.4 Facades

O padrão *facade* visa simplificar o acesso a um conjunto de interfaces em um subsistema, fornecendo uma única e intuitiva classe de alto nível. Isso reduz a complexidade do sistema, dividindo-o em subsistemas menores e minimizando a comunicação e dependência entre eles. Ao adotar a *facade*, as interações complexas entre os objetos são unificadas em uma classe coesa, tornando o uso e compreensão do subsistema mais fácil (GAMMA et al., 1995).

Figura 13: Exemplo de facade

```
@Component
@RequiredArgsConstructor
@Transactional(rollbackFor = BusinessException.class)
@Slf4j
public class AnexoGeralFacade {
    private final AnexoGeralService anexoGeralService;

    Bruno Fonseca
    public AnexoGeralDTO getById(Long id) {
        var anexoGeral = anexoGeralService.getById(id);
        return ModelMapperUtil.map(anexoGeral, AnexoGeralDTO.class);
    }

    Bruno Fonseca *
    public ListResponse<AnexoGeralProjection> search(
            Long idProjetoTcc,
            boolean onlyAvaliacao,
            AnexoGeralGridOptions options
    ) {
        return anexoGeralService.search(idProjetoTcc, onlyAvaliacao, options);
    }

    Bruno Fonseca
    public AnexoGeralDTO insert(AnexoGeralDTO anexoGeralDTO) throws Exception {
        var anexoGeral = ModelMapperUtil.map(anexoGeralDTO, AnexoGeral.class);
        anexoGeral = anexoGeralService.insert(anexoGeral);
        return ModelMapperUtil.map(anexoGeral, AnexoGeralDTO.class);
    }

    Bruno Fonseca
    public void delete(Long id) throws Exception {
        anexoGeralService.delete(id);
    }

    1 usage  Bruno Fonseca
    public ArquivoDTO download(Long id) {
        Arquivo anexoGeral = anexoGeralService.getById(id);
        return ModelMapperUtil.map(anexoGeral, ArquivoDTO.class);
    }
}
```

Fonte: Do autor

A Figura 13 mostra a *facade* referente à entidade “AnexoGeral”. Nessa *facade*, as informações recebidas da *controller* são processadas e encaminhadas para serem tratadas pela *service* correspondente. Um exemplo é o método “getById”, que é utilizado para consultar o ID do anexo desejado. Ao chamar esse método na *controller*, a *facade* aciona a *service* apropriada, que busca os dados do anexo geral com base no ID fornecido. Por fim, as informações são mapeadas e retornadas a *controller* para que ela possa responder ao cliente de forma adequada.

4.2.5 Services

Responsável por coordenar a interação entre as operações de negócio e as operações de persistência de dados, o que implica o uso de repositórios para recuperar e armazenar entidades ou agregados do domínio, assegurando que a persistência seja executada em conformidade com as regras de negócio estabelecidas (EVANS; SZPOTON, 2015).

Figura 14: Exemplo de service

```
@Service
@RequiredArgsConstructor
@Transactional(rollbackFor = BusinessException.class)
@Slf4j
public class AnexoGeralService {
    private final AnexoGeralRepository anexoGeralRepository;

    Bruno Fonseca
    public AnexoGeral getById(@NonNull Long id) { return anexoGeralRepository.getReferenceById(id); }

    Bruno Fonseca *
    public ListResponse<AnexoGeralProjection> search(
            @NonNull Long idProjetoTcc, boolean onlyAvaliacao, AnexoGeralGridOptions options
    ) {
        if (StringUtil.isNullOrBlank(options.getOrderByField())) {
            options.setOrderByField("id");
            options.setOrderByDirection(OrderByDirection.DESC);
        }
        var page = anexoGeralRepository.search(options.getId(), options.getDescricao(),
                options.getTipoTcc(), idProjetoTcc, onlyAvaliacao, options.getPageable());
        return new ListResponse<>(page.getContent(), page.getTotalElements());
    }

    Bruno Fonseca
    public AnexoGeral insert(@NonNull AnexoGeral anexoGeral) throws BusinessException {
        anexoGeral.setId(null);
        anexoGeral.setDataExclusao(null);
        validate(anexoGeral);
        return anexoGeralRepository.save(anexoGeral);
    }

    Bruno Fonseca
    public void delete(@NonNull Long id) throws BusinessException {
        var anexoGeral = anexoGeralRepository.getReferenceById(id);
        anexoGeral.setDataExclusao(LocalDateTime.now());
        anexoGeralRepository.save(anexoGeral);
    }

    1 usage  Bruno Fonseca
    private void validate(AnexoGeral anexoGeral) throws BusinessException {...}
}
```

Fonte: Do autor

A Figura 14 ilustra a *service* relacionada à entidade “AnexoGeral”. Na *service*, as informações são enviadas pela *facade* e passam por processos de validação, tratamento, persistência e/ou consulta, com o objetivo de atender às solicitações específicas da respectiva *facade*. Um exemplo é o método “insert” da *service*, que recebe o objeto “anexoGeral” da *facade* correspondente, realiza a validação, persiste os dados no banco de dados e retorna o registro inserido por meio da sua *repository*. Esse fluxo garante que as informações sejam adequadamente processadas e armazenadas, seguindo as regras e procedimentos definidos.

4.2.6 Entidades

Foi adotado o uso do Spring Data JPA para realizar o mapeamento das entidades do banco de dados. Essa abordagem utiliza anotações para mapear a estrutura de uma tabela em uma classe Java, permitindo a definição do nome da tabela, o nome das colunas, os relacionamentos e outras funcionalidades. Um exemplo prático de como uma tabela do banco pode ser mapeada em uma classe Java é ilustrado na Figura 15.

Figura 15: Exemplo de entidade

```java
@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Table(name = "anexo_geral")
public class AnexoGeral extends Arquivo {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

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

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "id_modelo_documento")
    private ModeloDocumento modeloDocumento;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "id_professor", nullable = false)
    private Professor professor;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "id_projeto_tcc", nullable = false)
    private ProjetoTcc projetoTcc;

    @Column(name = "data_exclusao")
    private LocalDateTime dataExclusao;

    @Column(name = "avaliacao")
    private boolean avaliacao;
}
```

Fonte: Do autor

Na Figura apresentada, podem ser observadas as anotações utilizadas no Spring Data JPA para configurar a estrutura de uma tabela do banco de dados. Algumas dessas anotações são as seguintes:

- “@Entity”: Aplicada a uma classe para indicar que ela representa uma entidade no banco de dados.
- “@Table”: Utilizada para definir o nome da tabela.
- “@Id”: Define a chave primária da tabela.
- “@Column”: Permite definir uma coluna na tabela, especificando seu nome, se é obrigatória, se é única, entre outras propriedades.

Além disso, para declarar relacionamentos, são utilizadas anotações como:

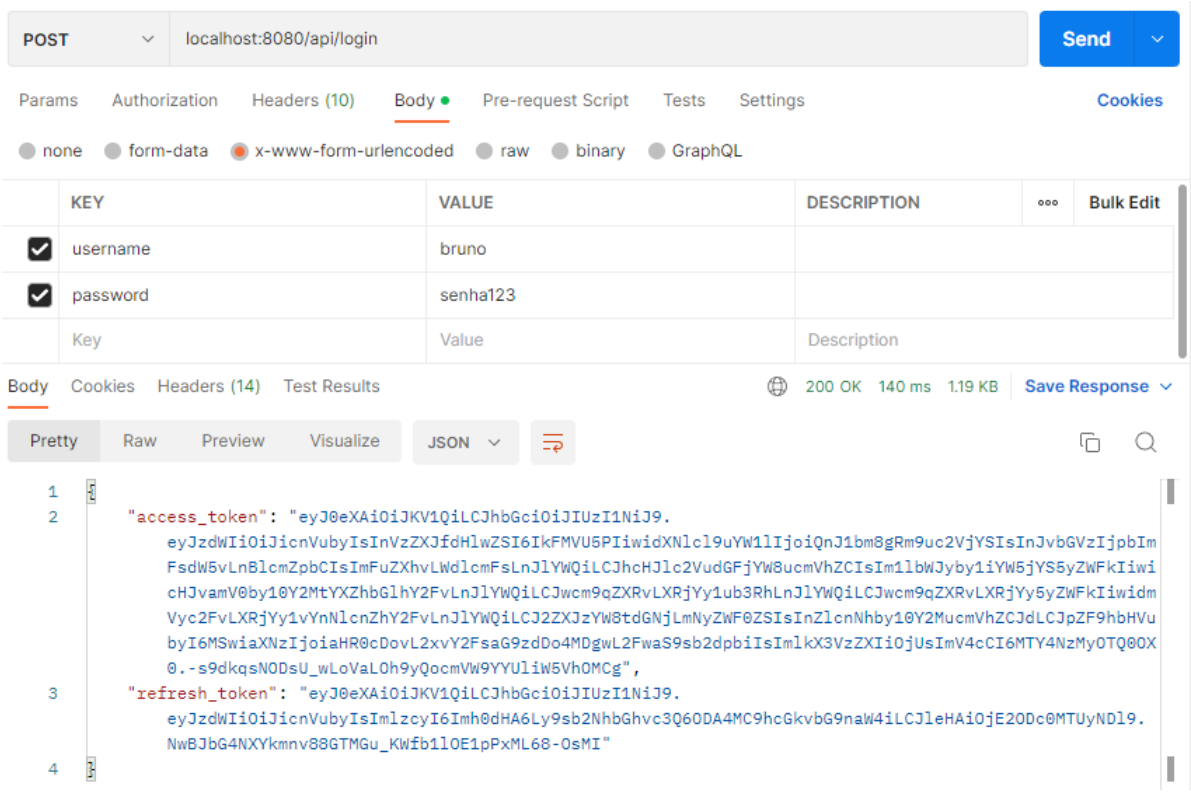
- “@ManyToOne”: Indica um relacionamento de muitos para um.
- “@OneToOne”: Indica um relacionamento de um para um.
- “@OneToMany”: Indica um relacionamento de um para muitos.

Caso seja necessário personalizar ainda mais esses relacionamentos, pode-se utilizar a anotação “@JoinColumn”. Ela permite definir propriedades adicionais, como o nome da coluna que será utilizada para mapear o relacionamento, se é obrigatória ou não, entre outras.

4.2.7 Segurança

Para garantir a segurança do sistema desenvolvido, foi implementado um processo de autenticação e autorização por meio do Spring Security (SPRING, 2022). A fim de restringir o acesso aos recursos somente a usuários autenticados e autorizados, foi implementado o JWT (JSON Web Token), uma solução amplamente reconhecida para a transmissão segura e eficiente de informações entre diferentes partes. De acordo com Jones, Bradley e Sakimura (2015), essa abordagem emprega um formato de token compacto, que permite a troca de dados confiáveis sem a necessidade de informações redundantes ou excessivas.

Figura 16: Exemplo de login



Fonte: Do autor

A Figura 16 mostra um exemplo de login do usuário “bruno”. Após o login ser realizado com sucesso, é retornado um JSON contendo o *token* e *refresh_token* da autenticação. Basicamente, o *token* é utilizado para autenticar e autorizar o usuário em cada solicitação subsequente. Ele possui uma duração curta e contém informações relevantes sobre o usuário. Por outro lado, o *refresh_token* é utilizado para obter um novo *token* quando o atual expirar. Esse *token* possui uma duração maior, evitando que o usuário precise efetuar o login novamente.

O JWT consiste em três partes: *header*, *payload* e *signature*.

- *Header* (Cabeçalho): Contém informações sobre o tipo de *token* e o algoritmo de assinatura.
- *Payload* (Carga útil): Carrega as informações transmitidas, como dados do usuário e outras informações relevantes.
- *Signature* (Assinatura): É gerada usando o cabeçalho, a carga útil e uma chave secreta. A assinatura garante a integridade dos dados no *token*.

Figura 17: Exemplo do token decodificado

Encoded PASTE A TOKEN HERE

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJicnVubyIsInVzZXJfdHlwZSI6IkFMVU5PIiwidXNlcl9uYW1lIjoiQnJ1bm8gRm9uc2VjYSIsInJvbGVzIjpbImFsdW5vLnBlcmZpbCIsImFuZXhvLWdlcmFsLnJlYWQiLCJhcHJlc2VudGFjYW8ucmVhZCIsIm1lbWJyby1iYW5jYS5yZWFkIiwicHJvamV0by10Y2MtYXZhbGlhY2FvLnJlYWQiLCJwcm9qZXRvLXRjYy1ub3RhLnJlYWQiLCJwcm9qZXRvLXRjYy5yZWFkIiwidmVyc2FvLXRjYy1vYnNlcnZhY2FvLnJlYWQiLCJ2ZXJzYW8tdGNjLmNyZWF0ZSIsInZlcnNhby10Y2MucmVhZCJdLCJpZF9hbHVuYnlI6MSwiaXNzIjoiaHR0cDovL2xvY2FsaG9zdDo4MDgwL2FwaS9sb2dpbiIsImlkX3VzZXIiOjUsIm V4cCI6MTY4NzMyOTQ0OX0.-s9dkqsNODsU_wLoVaLOh9yQocmVW9YYUliW5VhOMCg

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "typ": "JWT",
  "alg": "HS256"
}
```


PAYLOAD: DATA


```
{
  "sub": "bruno",
  "user_type": "ALUNO",
  "user_name": "Bruno Fonseca",
  "roles": [
    "aluno.perfil",
    "anexo-geral.read",
    "apresentacao.read",
    "membro-banca.read",
    "projeto-tcc-avaliacao.read",
    "projeto-tcc-nota.read",
    "projeto-tcc.read",
    "versao-tcc-observacao.read",
    "versao-tcc.create",
    "versao-tcc.read"
  ],
  "id_aluno": 1,
  "iss": "http://localhost:8080/api/login",
  "id_user": 5,
  "exp": 1687329449
}
```


VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

Fonte: Do autor

Figura 18: Exemplo do refresh token decodificado

Encoded PASTE A TOKEN HERE

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJicnVubyIsImlzcyI6Imh0dHA6Ly9sb2NhbGhvc3Q6ODA4MC9hcGkvbG9naW4iLCJleHAiOjE2ODc0MTUyNDl9.NwBJbG4NXYkmnv88GTMGu_KWfb1lOE1pPxML68-OsMI

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "typ": "JWT",
  "alg": "HS256"
}
```


PAYLOAD: DATA


```
{
  "sub": "bruno",
  "iss": "http://localhost:8080/api/login",
  "exp": 1687415249
}
```


VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

Fonte: Do autor

Nas Figuras 17 e 18, é possível observar suas decodificações. Suas definições são apresentadas abaixo:

- “sub”: Representa o assunto do *token*, ou seja, o identificador do usuário para o qual o *token* foi emitido. No caso deste exemplo, o valor é “bruno”.
- “user_type”: Indica o tipo de usuário ao qual o *token* pertence. Valores possíveis: “ADMIN”, “SUPERVISOR”, “PROFESSOR” ou “ALUNO”.
- “user_name”: Contém o nome completo do usuário associado ao *token*. No exemplo, o valor é “Bruno Fonseca”.
- “roles”: É uma lista de strings que representa as permissões concedidas ao usuário.
- “id_aluno”: Contém o identificador único do aluno relacionado ao *token*.
- “iss”: Indica o emissor do *token*, ou seja, a entidade que emitiu o *token*.
- “id_user”: Representa o identificador único do usuário relacionado ao *token*.
- “exp”: Especifica o tempo de expiração do *token*, representado em forma de *timestamp*.

4.2.7.1 Autenticação

A autenticação é um processo fundamental que permite o reconhecimento do usuário em um sistema. Durante esse processo, o usuário é identificado por meio do fornecimento de seu nome de usuário e senha no momento do login. Assim, a Figura 19 ilustra como foi feito a implementação da autenticação no *backend*.

Nesta Figura, o código apresenta uma classe chamada “CustomAuthenticationFilter” que estende um filtro de autenticação padrão do Spring Security. Essa classe personalizada lida com a autenticação de um usuário com base em um nome de usuário e senha fornecidos na solicitação. Ela usa um “AuthenticationManager” para autenticar o usuário e um “TokenService” para gerar um *token* de autenticação. Quando a autenticação é bem-sucedida, a classe retorna o *token* em formato JSON na resposta HTTP.

Figura 19: Implementação da autenticação

```
public class CustomAuthenticationFilter extends UsernamePasswordAuthenticationFilter {
    private final AuthenticationManager authenticationManager;
    private final TokenService tokenService;

    Bruno Fonseca *
    @Override
    public Authentication attemptAuthentication(
            HttpServletRequest request,
            HttpServletResponse response
    ) throws AuthenticationException {
        var username = request.getParameter( name: "username");
        var password = request.getParameter( name: "password");
        var authenticationToken = new UsernamePasswordAuthenticationToken(username, password);
        return authenticationManager.authenticate(authenticationToken);
    }

    4 usages  Bruno Fonseca *
    @Override
    protected void successfulAuthentication(
            HttpServletRequest request,
            HttpServletResponse response,
            FilterChain chain,
            Authentication authentication
    ) throws IOException {
        var user = (User) authentication.getPrincipal();
        var tokenMap = tokenService.getTokenMap(request, user.getUsername());
        response.setContentType(MediaType.APPLICATION_JSON_VALUE);
        new ObjectMapper().writeValue(response.getOutputStream(), tokenMap);
    }
}
```

Fonte: Do autor

4.2.7.2 Autorização

A autorização é um processo essencial que ocorre após a autenticação, sendo responsável por determinar as ações que um usuário autenticado está autorizado a realizar dentro de um sistema. Enquanto a autenticação tem como foco verificar a identidade do usuário, a autorização concentra-se nas permissões concedidas a ele. A Figura 20 ilustra a implementação da autorização, enquanto a Figura 21 apresenta um exemplo de *endpoint* com permissão de acesso.

Com mais detalhes, o código apresentado na Figura 20 implementa um filtro de autorização personalizado para uma API. Esse filtro tem a função de verificar se uma requisição possui um *token* de autenticação válido no cabeçalho de autorização. Caso o *token* seja válido, o filtro realiza a autenticação do usuário, adiciona as informações de autenticação ao contexto de segurança e permite o acesso à API. Por outro lado, caso o *token* seja inválido, a requisição é tratada como não autenticada, e uma resposta de erro é retornada.

Figura 20: Implementação da autorização

```
public class CustomAuthorizationFilter extends OncePerRequestFilter {
    private final TokenService tokenService;

    Bruno Fonseca *
    @Override
    protected void doFilterInternal(
            HttpServletRequest request,
            @NonNull HttpServletResponse response,
            @NonNull FilterChain filterChain
    ) throws ServletException, IOException {
        if (request.getServletPath().equals(SecurityConstants.API_LOGIN) || request.getServletPath().equals(SecurityConstants.API_REFRESH_TOKEN)) {
            filterChain.doFilter(request, response);
        } else {
            var authorizationHeader = request.getHeader(HttpHeaders.AUTHORIZATION);
            if (authorizationHeader != null && authorizationHeader.startsWith(SecurityConstants.TOKEN_PREFIX)) {
                try {
                    var token = authorizationHeader.substring(SecurityConstants.TOKEN_PREFIX.length());
                    var algorithm = Algorithm.HMAC256(SecurityConstants.SECRET.getBytes());
                    var verifier = JWT.require(algorithm).build();
                    var decodedJWT = verifier.verify(token);
                    var username = decodedJWT.getSubject();
                    var user = tokenService.getUserEnabled(username);
                    var userLogged = new UserLogged(user.getId(), user.getName(), user.getType());
                    var authenticationToken = new UsernamePasswordAuthenticationToken(userLogged, credentials: null, user.getAuthorities());
                    SecurityContextHolder.getContext().setAuthentication(authenticationToken);
                    filterChain.doFilter(request, response);
                } catch (Exception e) {
                    errorAuthenticationToken(response, e);
                }
            } else {
                filterChain.doFilter(request, response);
            }
        }
    }

    2 usages  Bruno Fonseca
    public static void errorAuthenticationToken(HttpServletResponse response, Exception e) throws IOException {
        log.error("Error logging in: {}", e.getMessage());
        response.setHeader( name: "error", e.getMessage());
        response.setStatus(HttpStatus.FORBIDDEN.value());
        var errorMap = new HashMap<String, String>();
        errorMap.put("error_message", e.getMessage());
        response.setContentType(MediaType.APPLICATION_JSON_VALUE);
        new ObjectMapper().writeValue(response.getOutputStream(), errorMap);
    }
}
```

Fonte: Do autor

Figura 21: Exemplo de endpoint com autorização

```
@PreAuthorize("hasAuthority('projeto-tcc.read')")
@GetMapping("{id}")
public ProjetoTccDTO getById(@PathVariable Long id) { return projetoTccFacade.getById(id); }
```

Fonte: Do autor

Já na Figura 21, podemos observar o código que apresenta a anotação “@PreAuthorize” no *endpoint getById* da entidade “ProjetoTcc”. Essa anotação impõe uma restrição de acesso ao método, permitindo exclusivamente que usuários com a autoridade “projeto-tcc.read” tenham acesso a ele. Isso significa que apenas usuários autorizados terão permissão para chamar esse método e obter os detalhes de um projeto de TCC com base em seu ID. Se um usuário sem essa autoridade tentar acessar o método, uma exceção de acesso não autorizado será lançada. Essa anotação oferece um controle de acesso granular, restringindo o acesso somente a usuários com as permissões adequadas.

4.3 Frontend

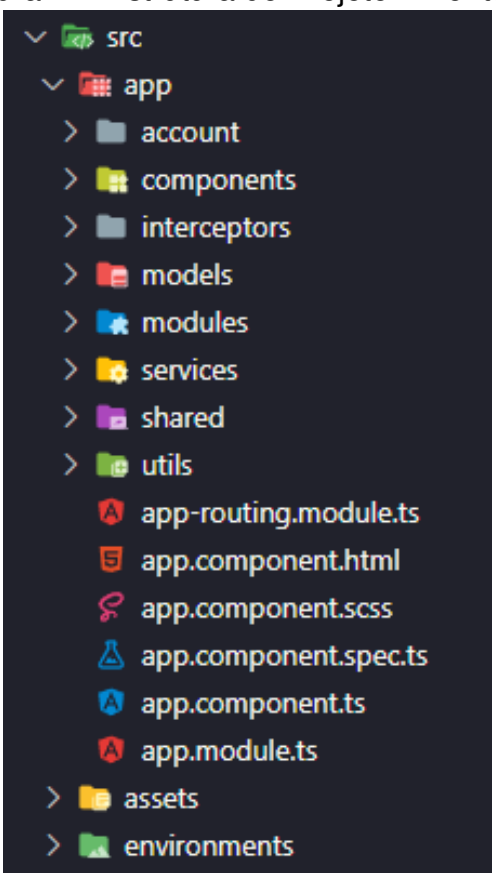
O *frontend* do projeto é a camada que possibilita a interação do usuário com o sistema, por meio de interfaces gráficas. Neste sistema em particular, optou-se pelo desenvolvimento do *frontend* utilizando o Angular, um *framework* que oferece suporte para a criação de *Single-Page Applications* (SPAs). Isso significa que a aplicação é concentrada em uma única página, na qual apenas o conteúdo principal é atualizado, proporcionando uma experiência de navegação mais fluida e rápida para o usuário.

Nos próximos tópicos, serão abordados maiores detalhes da implementação do *frontend* do projeto. Serão discutidos aspectos como a estruturação do projeto, comunicação com o servidor, navegabilidade e as interfaces desenvolvidas.

4.3.1 Estrutura do projeto

Na Figura 22, é possível observar a estruturação das pastas do projeto, e nos próximos tópicos, serão apresentadas suas respectivas definições.

Figura 22: Estrutura do Projeto - Frontend



Fonte: Do autor

Estruturação das pastas:

- account: Contém os arquivos necessários para configuração de autenticação e autorização.
- components: Contém os componentes que são reutilizados no projeto.
- interceptors: Configuração dos interceptadores de requisição.
- models: Classes de comunicação com o servidor *backend*.
- modules: Contém as páginas do sistema, separadas em módulos.
- services: Classes responsáveis pela comunicação com o servidor *backend*.
- shared: Contém as configurações de componentes compartilhados dentro do projeto.
- utils: Classes utilitárias.
- assets: Contém recursos estáticos que são utilizados pela aplicação.
- environments: Contém as classes de configuração relacionadas ao ambiente de execução da aplicação.

4.3.2 Comunicação com o servidor

A comunicação com o servidor é estabelecida por meio de requisições HTTP. Para simplificar esse processo, foi desenvolvida uma classe abstrata chamada “BaseService”, que fornece métodos prontos para interagir com o servidor. A Figura 23 ilustra essa classe e os métodos definidos nela, os quais correspondem aos verbos REST, criados para serem utilizados pelas classes *services* que herdam dessa *base*.

Além disso, é importante observar que na classe “BaseService” são necessários alguns parâmetros, incluindo o nome da *controller* que será utilizado para acessar os recursos desejados. Nesse sentido, o nome da *controller* é concatenado à *baseUrl* definida no arquivo de configuração de ambiente chamado *environment*, resultando na formação da URI que identifica o recurso a ser utilizado. A Figura 24 apresenta as informações contidas no arquivo *environment*.

Figura 23: BaseService

```
export abstract class BaseService {
    protected baseController = '';

    protected constructor(
        private controller: string,
        private httpClient: HttpClient
    ) {
        this.baseController = `${environment.baseUrl}/${this.controller}`;
    }

    protected getUrl(action: string): string {
        return `${this.baseController}/${action}`;
    }

    protected get<T>(path: string): Observable<T> {
        return <Observable<T>>this.httpClient.get(this.getUrl(path), { responseType: 'text' })
            .pipe(map(r => this._parseJson(r)));
    }

    protected post<T>(path: string, body: any): Observable<T> {
        return <Observable<T>>this.httpClient.post(this.getUrl(path), body, { responseType: 'text' })
            .pipe(map(r => this._parseJson(r)));
    }

    protected put<T>(path: string, body: any): Observable<T> {
        return <Observable<T>>this.httpClient.put(this.getUrl(path), body, { responseType: 'text' })
            .pipe(map(r => this._parseJson(r)));
    }

    protected patch<T>(path: string, body?: any): Observable<T> {
        return <Observable<T>>this.httpClient.patch(this.getUrl(path), body, { responseType: 'text' })
            .pipe(map(r => this._parseJson(r)));
    }

    protected del<T>(path: string): Observable<T> {
        return <Observable<T>>this.httpClient.delete(this.getUrl(path));
    }

    private _parseJson(response: any): any {
        return response ? JSON.parse(response, dateParser) : null;
    }
}
```

Fonte: Do autor

Figura 24: Classe Environment

```
export const environment = {
    baseUrl: 'http://localhost:8080/api',
    production: true,
};
```

Fonte: Do autor

Conforme mencionado anteriormente, todos os serviços de comunicação com o servidor são derivados da classe “BaseService”. A Figura 25 exemplifica essa herança, demonstrando como ela contribui para simplificar e tornar o código mais legível.

Figura 25: Exemplo de uma service

```
export class ModeloAvaliacaoService extends BaseService {

    constructor(private httpClient: HttpClient) {
        super('modelo-avaliacoes', httpClient);
    }

    getById(id: number): Observable<ModeloAvaliacaoModel> {
        return this.get(`${id}`);
    }

    search(options: ModeloAvaliacaoOptionsModel): Observable<ListResponseModel<ModeloAvaliacaoGridModel>> {
        return this.post('search', options);
    }

    insert(modeloAvaliacao: ModeloAvaliacaoModel): Observable<ModeloAvaliacaoModel> {
        return this.post('', modeloAvaliacao);
    }

    update(modeloAvaliacao: ModeloAvaliacaoModel): Observable<ModeloAvaliacaoModel> {
        return this.put(`${modeloAvaliacao.id}`, modeloAvaliacao);
    }

    delete(id: number): Observable<any> {
        return this.del(`${id}`);
    }

}
```

Fonte: Do autor

No entanto, os *endpoints* desenvolvidos estão protegidos por permissões, o que requer a adição do *token* de autenticação no cabeçalho das requisições. Para realizar essa tarefa, foi criada uma classe *interceptor* que implementa a interface *HttpInterceptor* do Angular. Essa classe permite executar uma ação na requisição interceptada. A Figura 26 apresenta a implementação desse *interceptor*.

Inicialmente, é verificado se a requisição é um login ou um *refreshToken*. Se for o caso, a requisição é enviada diretamente para o próximo manipulador. Caso contrário, o *interceptor* verifica se o cabeçalho da requisição contém um *token* de autorização. Se não houver, ele adiciona um *token* de acesso fornecido pelo serviço de autenticação. Em seguida, a requisição é enviada para o próximo manipulador.

Se ocorrer um erro de autenticação (status 403 ou 401), o *interceptor* tenta renovar o *token* de acesso. Para isso, realiza uma solicitação ao serviço de autenticação e, se a renovação for bem-sucedida, o novo *token* é salvo e a requisição original é reenviada. Caso ocorram outros erros, mensagens correspondentes serão exibidas para informar o usuário sobre o problema e ajudá-lo a solucioná-lo. Essa abordagem garante que as requisições sejam feitas com as devidas permissões e evita acesso não autorizado aos recursos do sistema.

Figura 26: AuthInterceptor

```
export class AuthInterceptor implements HttpInterceptor {

    constructor(
        private httpClient: HttpClient,
        private authService: AuthService,
        private alertService: AlertService
    ) { }

    intercept(request: HttpRequest<unknown>, next: HttpHandler): Observable<HttpEvent<unknown>> {
        if(request.url.includes('api/login') || request.url.includes('api/token/refresh') ) {
            return next.handle(request);
        }
        const req = !request.headers.has('Authorization') ? request.clone({
            setHeaders: {
                Authorization: `Bearer ${this.authService.accessToken}`
            }
        }) : request.clone();
        return next.handle(req).pipe(catchError((err: HttpErrorResponse) => {
            if (err.status === 403 || err.status === 401) {
                return this.httpClient.get(`${environment.baseUrl}/token/refresh`, {
                    headers: {
                        Authorization: `Bearer ${this.authService.refreshToken}`
                    }
                }).pipe(
                    switchMap((response: any) => {
                        this.authService.saveToken(<TokenModel>response);
                        return next.handle(request.clone({
                            setHeaders: {
                                Authorization: `Bearer ${this.authService.accessToken}`
                            }
                        }));
                    }),
                    catchError((err: HttpErrorResponse) => {
                        if(err.status === 403 || err.status === 401) {
                            this.alertService.showError(
                                'Sua sessão expirou ou você não tem acesso a esta funcionalidade',
                                'Erro', () => {
                                setTimeout(() => this.authService.logout(), 100);
                            });
                        } else if (err.status === 400 || err.status === 500) {
                            this.alertService.showError(err.error);
                        }
                        return throwError(() => err);
                    })
                );
            } else if (err.status === 400 || err.status === 500) {
                this.alertService.showError(err.error);
            } else if (err.status === 0 || err.status === 405) {
                this.alertService.showError('Servidor não está respondendo, tente novamente mais tarde');
            } else if (err.status === 404) {
                this.alertService.showError('404 - Not Found');
            }
            return throwError(() => err)
        }));
    }
}
```

Fonte: Do autor

4.3.3 Navegação

Nesta seção, será abordado a implementação da navegabilidade do sistema. No entanto, antes disso, é importante compreender a estrutura de modularização adotada e como foi aplicado o *lazy loading modules* (módulos de carregamento lento).

No Angular, um NgModule desempenha um papel fundamental na organização e modularização de um aplicativo. Ele é uma classe decorada que agrupa componentes relacionados, serviços, diretivas, pipes e outros artefatos em um contexto lógico dentro de um aplicativo Angular (GOOGLE, 2022).

Por padrão, os NgModules são carregados de forma imediata. Isso significa que, ao carregar a aplicação, todos os NgModules são carregados, independentemente de serem necessários imediatamente ou não. No entanto, em aplicações maiores com várias rotas, é uma prática recomendada utilizar o *lazy loading* (carregamento lento) como um padrão de design. Esse recurso permite carregar os NgModules somente quando eles são necessários. A utilização do *lazy loading* ajuda a manter o tamanho inicial dos pacotes menores, o que, por sua vez, contribui para reduzir os tempos de carregamento (GOOGLE, 2022).

A modularização no Angular é uma prática fundamental para a organização e estruturação eficiente do código de um aplicativo. Ao dividir o projeto em módulos independentes e coesos, é possível melhorar significativamente a manutenibilidade, reusabilidade e escalabilidade do sistema. Além disso, a modularização permite um desenvolvimento mais ágil e facilita a colaboração entre os membros da equipe. Com uma arquitetura modular bem definida, é mais fácil identificar e isolar problemas, realizar atualizações e implementar novos recursos. Em resumo, a modularização no Angular é uma estratégia poderosa para tornar o código mais organizado, flexível e sustentável ao longo do ciclo de vida do projeto (GOOGLE, 2022).

No sistema desenvolvido, foram estabelecidos três módulos principais:

- Agendas: Este módulo inclui as funcionalidades relacionadas à agenda.
- Cadastros: Neste módulo, estão disponíveis as funcionalidades referentes aos registros e cadastros gerais do sistema.
- Projetos de TCC: Esse módulo abrange as funcionalidades relacionadas diretamente ao gerenciamento dos projetos de TCC.

Cada módulo consiste em dois NgModule distintos. O primeiro NgModule abrange os componentes declarados no projeto, assim como os importados necessários para seu correto funcionamento. O nome desse NgModule é definido pela concatenação do nome do módulo em questão com a palavra “Module”.

Por outro lado, o segundo NgModule é responsável pelo roteamento das telas, sendo seu nome definido pela concatenação do nome do módulo com a palavra “RoutingModule”.

Nas Figuras 27 e 28, são apresentados os dois NgModules correspondentes ao módulo de Agendas: “AgendaModule” e “AgendaRoutingModule”, respectivamente.

Na Figura 27, são apresentados no “declarations” (declarações), os componentes declarados naquele módulo, ou seja, as telas desenvolvidas referentes a Agenda. Já nos “imports” (importações), são definidas as importações necessárias para o funcionamento deste módulo.

Figura 27: Módulo da agenda

```
@NgModule({
    declarations: [
        FormAgendaApresentacoesComponent,
        ListAgendaApresentacaoComponent,
        MinhaAgendaComponent
    ],
    imports: [
        CommonModule,
        AgendasRoutingModule,
        AppMaterialModule,
        SharedComponentModule,
        ReactiveFormsModule,
        NgxMaskModule.forChild(),
        CalendarModule.forRoot({
            provide: DateAdapter,
            useFactory: adapterFactory,
        })
    ]
})
export class AgendasModule { }
```

Fonte: Do autor

Já na Figura 28, são exibidas as rotas e seus componentes correspondentes, juntamente com suas rotas filhas. Nesse contexto, o módulo RouterModule é importado para fornecer recursos de roteamento, enquanto o RouterModule é exportado para permitir que outros módulos importem e utilizem as rotas definidas nesse módulo.

Figura 28: Módulo de roteamento da agenda

```
const routes: Routes = [
    {
        path: 'agenda-apresentacoes',
        children: [
            {
                path: '',
                component: ListAgendaApresentacaoComponent,
                data: { breadcrumb: 'Agenda Geral de Apresentações - ' + USER_TYPE[getUserType()] }
            },
            {
                path: 'form',
                component: FormAgendaApresentacoesComponent,
                data: { breadcrumb: 'Cadastro' }
            }
        ],
        data: { breadcrumb: 'Agenda de Apresentações' }
    },
    {
        path: 'minha-agenda',
        children: [
            {
                path: '',
                component: MinhaAgendaComponent,
                data: { breadcrumb: 'Minha Agenda' }
            }
        ]
    },
];

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

Fonte: Do autor

Portanto, foi implementado o recurso de *lazy loading* no AppRoutingModule, utilizando os três módulos criados, como pode ser observado na Figura 29. Nesse sentido, quando a rota “agendas” é acessada, o Angular irá carregar de forma assíncrona o módulo AgendasModule utilizando a função “import()”.

Figura 29: Módulo de roteamento geral da aplicação

```
const routes: Routes = [
    { ...
    },
    {
        path: '',
        component: NavigationComponent,
        children: [
            {
                path: 'agendas',
                loadChildren: () => import('./modules/agendas/agendas.module').then((m) => m.AgendasModule),
                data: { breadcrumb: { skip: true } },
                canActivate: [AuthGuard]
            },
            {
                path: 'cadastros',
                loadChildren: () => import('./modules/cadastros/cadastros.module').then((m) => m.CadastrosModule),
                data: { breadcrumb: { skip: true } },
                canActivate: [AuthGuard]
            },
            {
                path: 'tcc',
                loadChildren: () => import('./modules/tcc/tcc.module').then((m) => m.TccModule),
                data: { breadcrumb: { skip: true } },
                canActivate: [AuthGuard]
            },
            { ...
            },
            { ...
            },
            { ...
            }
        ],
        canActivate: [AuthGuard],
    },
    { path: '**', component: PageNotFoundComponent },
];

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

Fonte: Do autor

Essa abordagem garante que o módulo AgendasModule, juntamente com todos os seus componentes, serviços e dependências, seja carregado somente quando a rota correspondente for visitada. Essa estratégia é altamente eficiente, pois otimiza o desempenho da aplicação ao reduzir o tempo de inicialização, resultando em uma experiência de usuário mais ágil e rápida. Ao adiar o carregamento dos elementos do módulo até que sejam necessários, evita-se qualquer sobrecarga desnecessária no carregamento inicial da aplicação. Como resultado, os recursos do sistema são utilizados de forma mais eficiente, permitindo uma resposta mais rápida às interações do usuário e garantindo uma experiência mais fluída e eficiente.

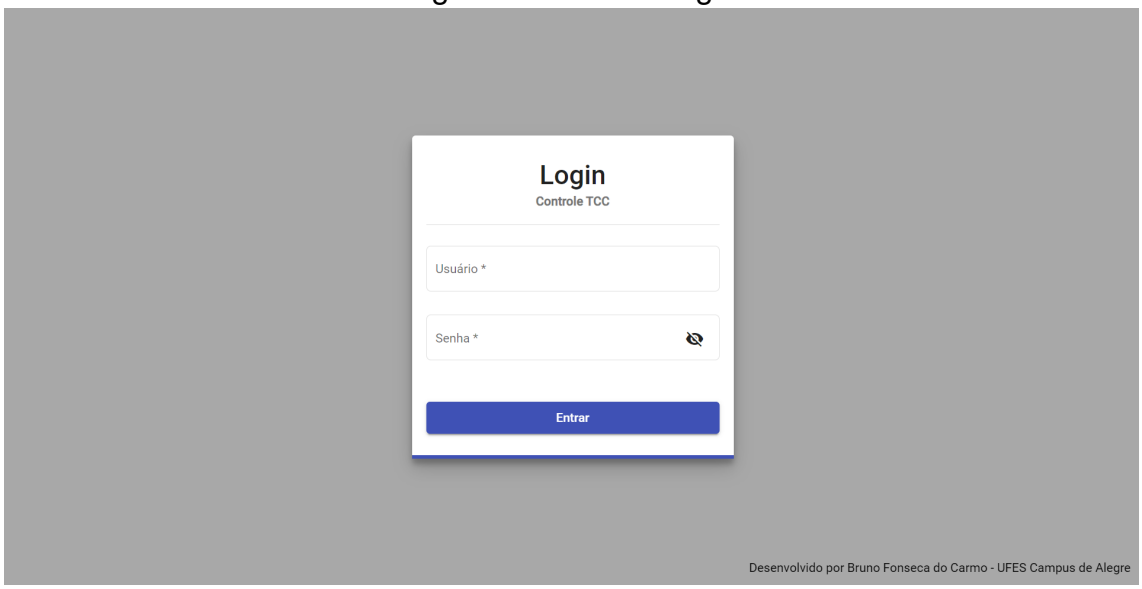
4.4 Interfaces

Nesta seção, serão apresentadas as telas do sistema.

4.4.1 Tela de Login

A tela de login é a primeira interface de acesso ao sistema, proporcionando ao usuário a possibilidade de autenticação por meio do fornecimento de seu nome de usuário e senha. Essa tela é ilustrada na Figura 30. Após uma autenticação bem-sucedida, o usuário é redirecionado para a tela inicial do sistema. No entanto, se ocorrer uma falha na autenticação, uma mensagem de erro será exibida, informando sobre o problema de autenticação.

Figura 30: Tela de login

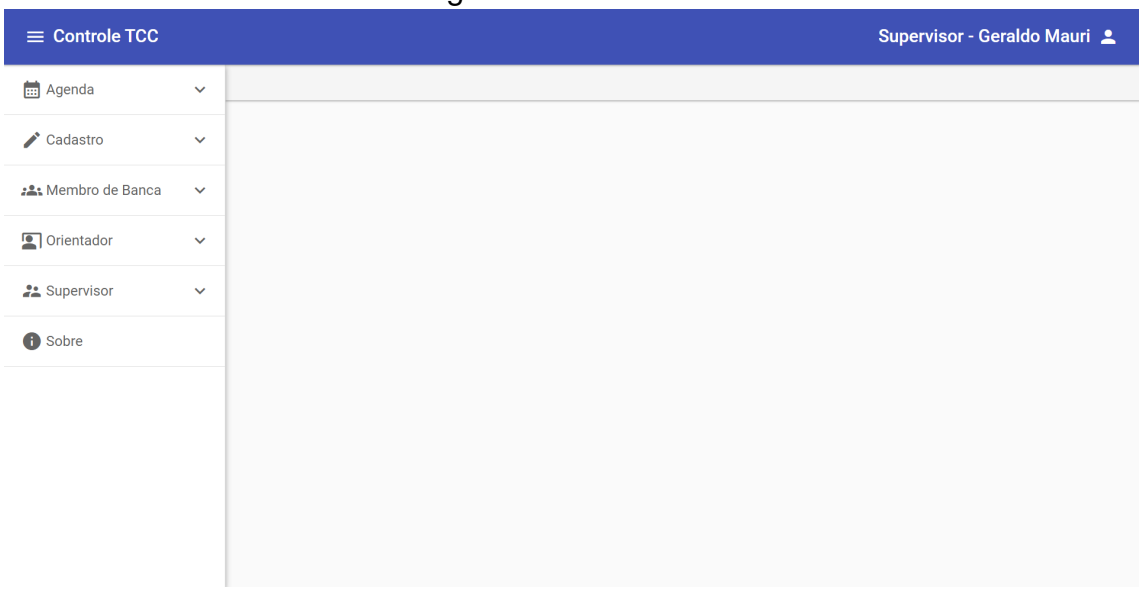


Fonte: Do autor

4.4.2 Tela Inicial

Após realizar o login com êxito, o sistema utiliza o *token* de autenticação para identificar o tipo de usuário e direcioná-lo para a tela inicial correspondente. Na Figura 31, é possível visualizar a tela inicial de um usuário professor supervisor.

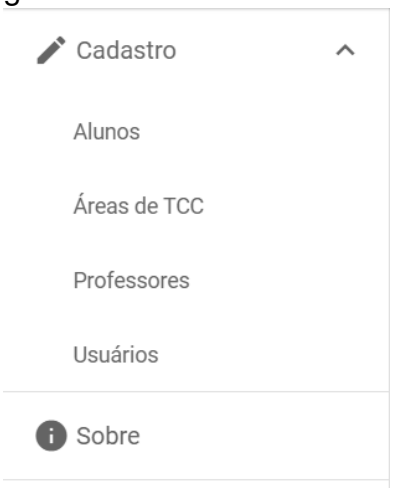
Figura 31: Tela Inicial



Fonte: Do autor

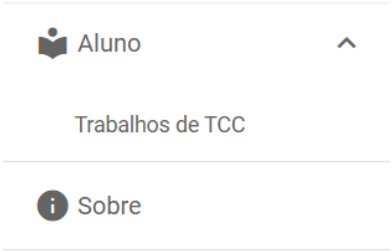
Já as Figuras 32 e 33 exibem os menus destinados aos usuários administradores e alunos, respectivamente.

Figura 32: Menu Administrador



Fonte: Do autor

Figura 33: Menu Aluno

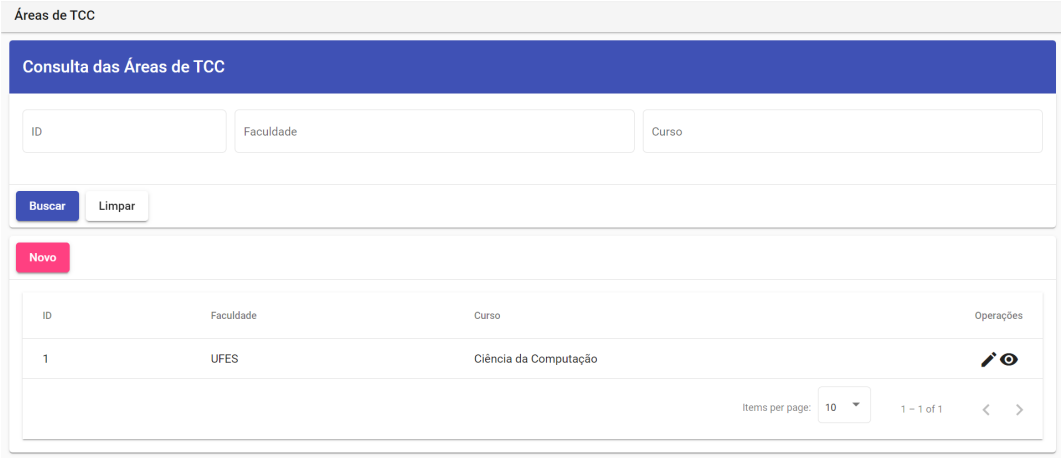


Fonte: Do autor

4.4.3 Telas de Áreas de TCC

Na Figura 34, é apresentada a tela de consulta das áreas de TCC, na qual é possível editar e visualizar as informações de cada área por meio dos ícones localizados na última coluna da tabela. Além disso, a inserção de uma nova área de TCC pode ser realizada acessando o botão “Novo”, que redireciona para a tela ilustrada na Figura 35.

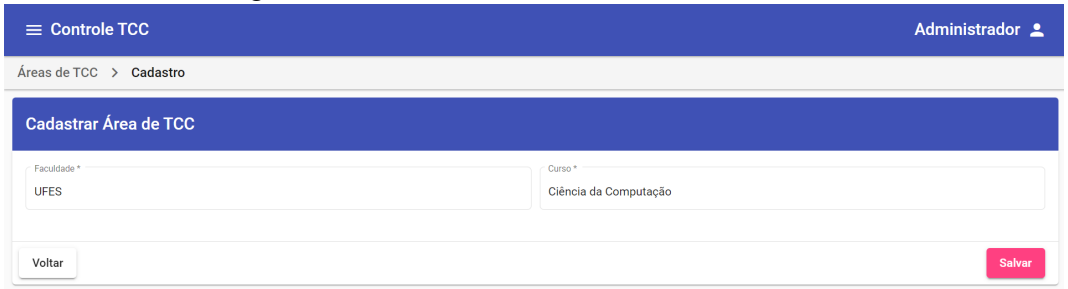
Figura 34: Tela de listagem de áreas de TCC



Fonte: Do autor

Na tela de cadastro e edição de áreas de TCC, representada na Figura 35, os usuários têm a capacidade de registrar uma nova área de TCC e também podem utilizar essa mesma tela para editar as informações de uma área de TCC existente.

Figura 35: Tela de cadastro de área de TCC



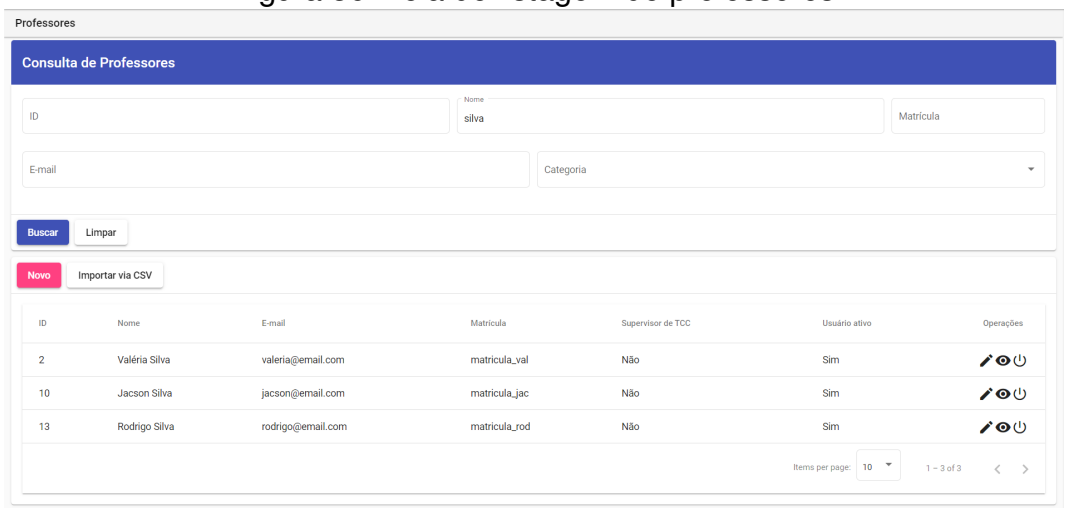
Fonte: Do autor

4.4.4 Telas de Professor

A Figura 36 ilustra a tela de listagem de professores, oferecendo a opção de filtrar a busca com base nos campos apresentados. Além disso, nessa mesma tela, são dis-

ponibilizados botões interativos que permitem executar diversas ações. É possível redirecionar para a tela de inclusão de um novo professor, realizar importação de dados por meio de arquivos CSV, editar ou visualizar informações de um professor existente e, por fim, inativar o usuário associado a esse professor, resultando no desligamento de seu acesso ao sistema.

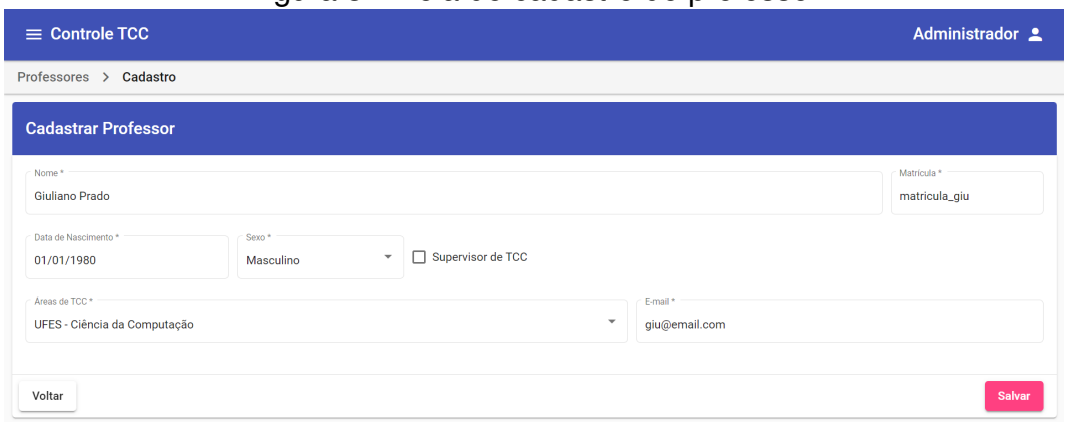
Figura 36: Tela de listagem de professores



Fonte: Do autor

A tela de cadastro de professores, representada na Figura 37, desempenha um papel fundamental ao permitir a realização das operações de inclusão e edição. É por meio dessa interface que os dados pertinentes ao professor são inseridos e atualizados. Desta forma, é possível definir se o professor em questão é um supervisor ou não de TCC.

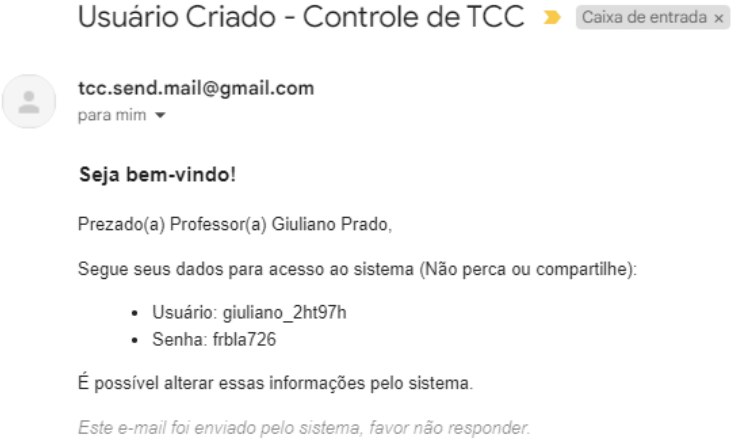
Figura 37: Tela de cadastro de professor



Fonte: Do autor

Após o cadastro de um novo professor, o sistema realiza a geração de um usuário e senha exclusivos para o mesmo. Essas informações são enviadas de maneira automatizada para o endereço de e-mail fornecido durante o processo de cadastro. Esse e-mail, apresentado na Figura 38, contém todas as informações necessárias para que o professor possa acessar o sistema.

Figura 38: E-mail de criação do professor

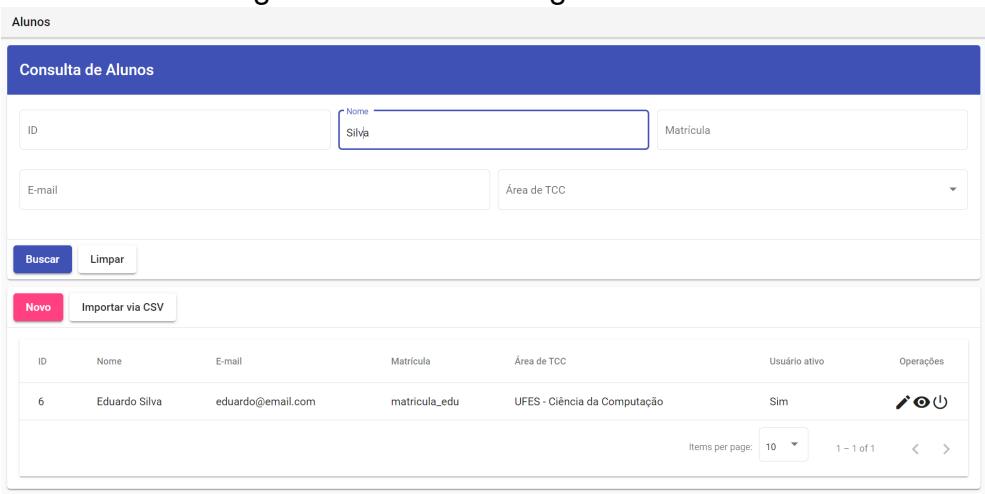


Fonte: Do autor

4.4.5 Telas de Aluno

A Figura 39 exibe a interface de listagem de alunos, que proporciona uma variedade de recursos, como filtros de busca e várias ações, incluindo inclusão, edição e importação de alunos, bem como a possibilidade de desativar um usuário vinculado a um aluno.

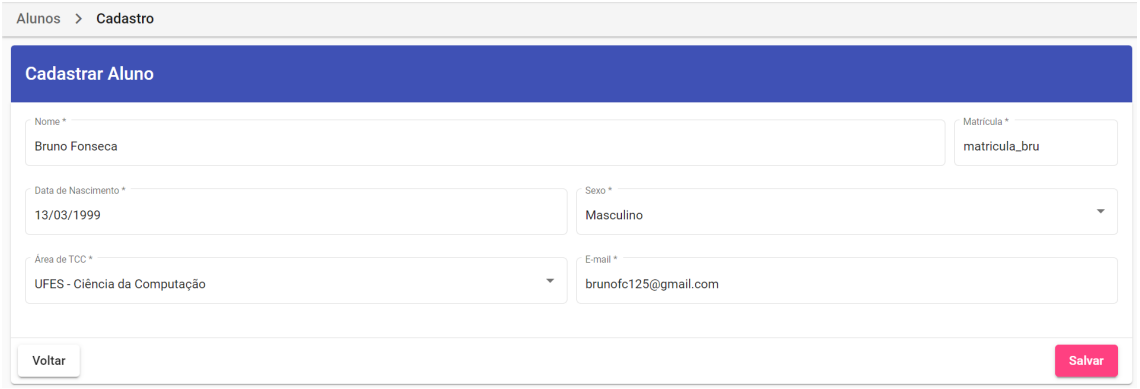
Figura 39: Tela de listagem de alunos



Fonte: Do autor

Na tela de cadastro exibida na Figura 40, é possível realizar a inclusão e edição de dados de um aluno.

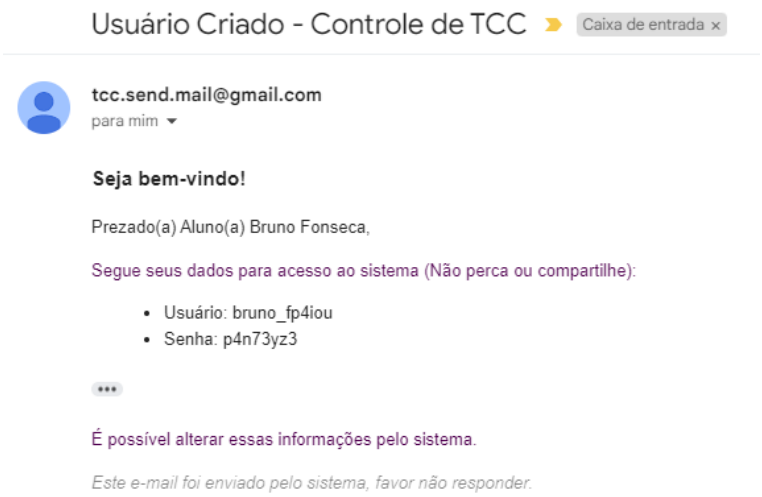
Figura 40: Tela de cadastro de aluno



Fonte: Do autor

Da mesma forma que acontece para o professor, a inclusão do aluno também envolve o envio de um e-mail com as informações necessárias para acessar o sistema. O e-mail correspondente é exemplificado na Figura 41.

Figura 41: E-mail de criação do aluno

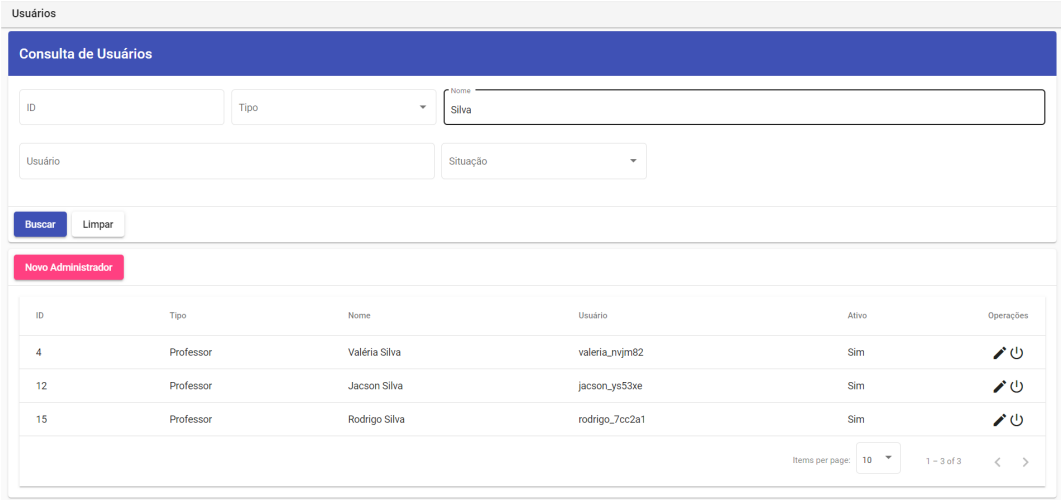


Fonte: Do autor

4.4.6 Telas de Usuário

Para o gerenciamento de usuários, a tela mostrada na Figura 39 exibe informações sobre os usuários do sistema. Nessa tela, é possível adicionar um novo administrador, editar e ativar/inativar usuários existentes.

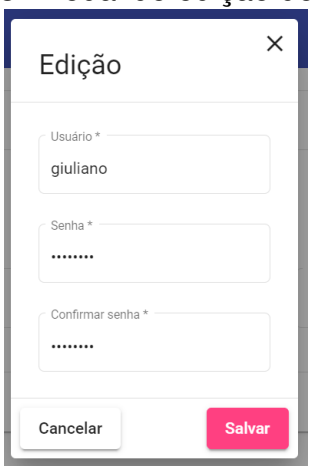
Figura 42: Tela de listagem de usuários



Fonte: Do autor

Para a inclusão e alteração de um usuário, é exibida a modal representada na Figura 43. Essa mesma modal é utilizada tanto na edição dos usuários dos professores e alunos, como também permite que o próprio usuário logado altere suas credenciais de segurança.

Figura 43: Modal de edição de usuário



Fonte: Do autor

Ao realizar a desativação de um usuário, uma modal é exibida, conforme ilustrado na Figura 44, na qual é solicitado a confirmação da ação atual. Já no processo de ativação, a mesma modal é apresentada, porém solicitando a confirmação específica para a ativação do usuário.

Figura 44: Modal de desativação do usuário

Fonte: Do autor

4.4.7 Telas de Importação

Para as importações de alunos e professores, o mesmo fluxo ocorre no sistema, desta forma a Figura 45 mostra a modal utilizada em ambas entidades. Este modal, contém um botão de *Download* do modelo, que ao clicar é realizado o *download* do modelo de importação.

Figura 45: Modal de importação

Fonte: Do autor

A Figura 46 apresenta a planilha que tem como objetivo importar os dados dos professores. Essa planilha possui colunas distintas, cada uma representando um atributo relevante. Esses atributos incluem informações como nome, e-mail, sexo, matrícula, data de nascimento, supervisão de TCC e áreas de TCC do professor. Algumas colunas apresentam opções pré-definidas para determinados campos, como por exemplo, o sexo que pode ser “MASC”, “FEM” ou “INDF”.

Figura 46: Planilha de importação de professores

	A	B	C	D	E	F	G
1	Nome	E-mail	Sexo (MASC/FEM/INDF)	Matrícula	Data de nascimento (dd/MM/yyyy)	Supervisor de TCC (S/N)	ID's Áreas de TCC (Lista separada por '-', Consulte o Sistema)
2	Edmar Hell	edmar@email.com	MASC	matricula_edm	02/01/80	N	1
3	Geraldo Mauri	geraldo@email.com	MASC	matricula_gmi	03/01/80	S	1
4	Juliana Pinheiro	juliana@email.com	FEM	matricula_jul	04/01/80	N	1
5	Antonio Junior	antonio@email.com	MASC	matricula_ant	05/01/80	N	1
6	Clayton Fraga	clayton@email.com	MASC	matricula_cla	06/01/80	N	1
7	Dayan Bissoli	dayan@email.com	MASC	matricula_day	07/01/80	N	1
8	Helder Mendes	helder@email.com	MASC	matricula_hel	08/01/80	N	1
9	Jacson Silva	jacson@email.com	MASC	matricula_jac	09/01/80	N	1
10	Larice Andrade	larice@email.com	FEM	matricula_lar	10/01/80	N	1
11	Paulo Souza	paulo@email.com	MASC	matricula_pso	11/01/80	N	1
12	Rodrigo Silva	rodrigo@email.com	MASC	matricula_rod	12/01/80	N	1
13	Simone Costa	simone@email.com	FEM	matricula_sim	13/01/80	N	1

Fonte: Do autor

Da mesma forma, a Figura 47 apresenta uma planilha destinada à importação dos dados dos alunos, com uma organização semelhante. A planilha é composta por colunas que representam os atributos relevantes para a entidade aluno, tais como nome, e-mail, sexo, data de nascimento, matrícula e área de TCC. Essa estrutura padronizada da planilha facilita a manipulação e o tratamento dos dados durante o processo de importação, resultando em uma operação mais eficiente e precisa.

Figura 47: Planilha de importação de alunos

	A	B	C	D	E	F
1	Nome	E-mail	Sexo (MASC/FEM/INDF)	Data de nascimento (dd/MM/yyyy)	Matricula	ID Área de TCC (Consulte o Sistema)
2	Romulo Borges	romulo@email.com	MASC	01/01/99	matricula_rom	1
3	Gabriel Dias	gabriel@email.com	MASC	02/01/99	matricula_gab	1
4	Felipe Camatta	felipe@email.com	MASC	03/01/99	matricula_fel	1
5	Alcebíades Gama	alcebiades@email.com	MASC	04/01/99	matricula_alc	1
6	Eduardo Silva	eduardo@email.com	MASC	05/01/99	matricula_edu	1
7	Maria Sobreira	maria@email.com	FEM	06/01/99	matricula_mar	1

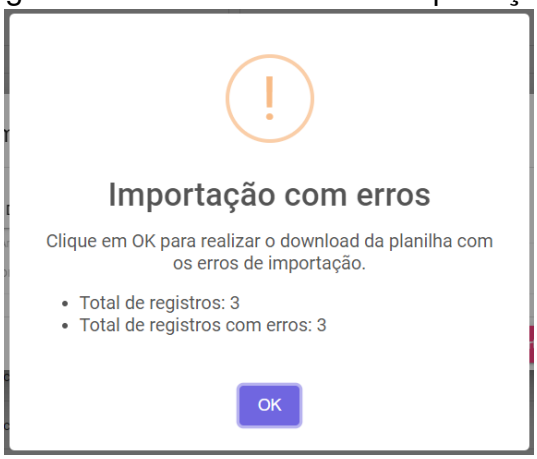
Fonte: Do autor

É fundamental destacar que o sistema oferece a opção de baixar modelos de planilhas apenas com o cabeçalho preenchido. No entanto, é importante ressaltar que os dados contidos nas planilhas das Figuras 46 e 47 são exemplos para facilitar a compreensão do processo de importação.

Ambas as planilhas seguem um formato universalmente reconhecido e amplamente utilizado: o CSV (Comma-Separated Values - Valores Separados por Vírgula), que permite a representação dos dados em formato de texto, utilizando vírgulas como separadores entre as colunas. Esse formato é suportado por diferentes aplicativos de planilhas eletrônicas e bancos de dados, o que facilita a integração e a interoperabilidade do sistema com outras ferramentas (SHAFRANOVICH, 2005).

A abordagem de importação simplificada oferece uma maneira mais fácil de identificar e corrigir erros antes do processo de importação. Caso ocorra alguma importação incorreta de planilha, será exibido uma modal, apresentada na Figura 48, após a importação com erros. Essa modal apresentará o número total de registros, bem como o total de registros com erros, permitindo também o *download* da planilha importada juntamente com os erros relatados.

Figura 48: Modal de erros na importação



Fonte: Do autor

A seguir, nas Figuras 49 e 50, apresentam-se as planilhas com os erros de importação. Durante o processo de importação, caso ocorra algum erro, uma coluna de erros é adicionada, relatando as falhas encontradas em cada linha.

Na Figura 49, pode-se observar que o professor da linha 2 não pôde ser importado, devido à existência de outro professor com a mesma matrícula. Além disso, o professor da linha 3 também não pôde ser importado devido a um e-mail inválido, juntamente com outros erros exibidos.

Figura 49: Planilha de importação de professores com erros

	A	B	C	D	E	F	G	H
1	Nome	E-mail	Sexo (M	Matrícula	Data de nas	Sup	ID's	Erros
2	Edmar Hell	edmar@email.com	MASC	matricula_edm	02/01/1980	N	1	Já existe outro professor com esta matrícula
3	Marcos Barbosa	marcos				N		Erro na coluna Sexo (MASC/FEM/INDF): Valor inválido, deveria ser MASC/FEM/INDF Erro na coluna Data de nascimento (dd/MM/yyyy): Valor inválido, deveria seguir o formato dd/MM/yyyy ou dd/MM/yy Erro na coluna ID's Áreas de TCC (Lista separada por '-', Consulte o Sistema): Valor inválido, deveria ser uma lista separada por "-" do tipo numérico Matrícula não informada E-mail inválido Sexo não informado Data de nascimento não informada Áreas de TCC não informadas
4	Vitoria Silva	vitoria		matricula_vis		N	1	Erro na coluna Sexo (MASC/FEM/INDF): Valor inválido, deveria ser MASC/FEM/INDF Erro na coluna Data de nascimento (dd/MM/yyyy): Valor inválido, deveria seguir o formato dd/MM/yyyy ou dd/MM/yy E-mail inválido Sexo não informado Data de nascimento não informada

Fonte: Do autor

Da mesma forma, na planilha de importação de alunos, ilustrada na Figura 50, ocorreram erros de importação devido à ausência da informação de sexo, e-mail inválido e formato incorreto da data de nascimento, entre outros erros.

Figura 50: Planilha de importação de alunos com erros

	A	B	C	D	E	F	G
1	Nome	E-mail	Sexo (M	Data de nas	Matricula	ID Á	Erros
2	Romulo Borges	romulo@email.com	MASC	01/01/1999	matricula_rom	1	Já existe outro aluno com esta matrícula
3	Julia Rodrigues	julia					Erro na coluna Data de nascimento (dd/MM/yyyy): Valor inválido, deveria seguir o formato dd/MM/yyyy ou dd/MM/yy Matrícula não informada E-mail inválido Sexo não informado Data de nascimento não informada Área de TCC não informada
4	Guilherme Araujo	teste			matricula_gui	1	Erro na coluna Sexo (MASC/FEM/INDF): Valor inválido, deveria ser MASC/FEM/INDF Erro na coluna Data de nascimento (dd/MM/yyyy): Valor inválido, deveria seguir o formato dd/MM/yyyy ou dd/MM/yy E-mail inválido Sexo não informado Data de nascimento não informada

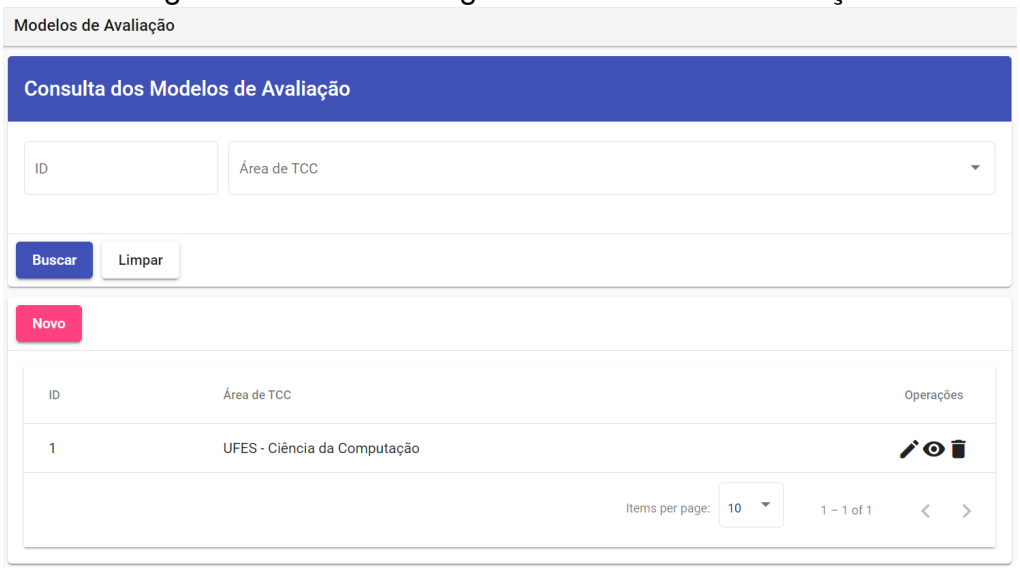
Fonte: Do autor

4.4.8 Telas de Modelo de Avaliação

Para tornar o processo de avaliação mais abrangente e personalizável, foram desenvolvidas funcionalidades de modelos de avaliação. Esses modelos permitem a parametrização das avaliações de acordo com a área do TCC, o tipo de TCC (Qualificação ou Defesa) e o tipo de professor (Supervisor, Orientador ou Membro de Banca). Nas configurações desses modelos, são definidos os aspectos de avaliação específicos de cada tipo de TCC e professor. As telas correspondentes a essas funcionalidades podem ser visualizadas nas Figuras 51, 52, 54 e 53.

A Figura 51 mostra a tela de listagem dos modelos de avaliação, na qual é possível redirecionar para a tela de cadastro, editar um modelo existente, visualizá-lo ou excluí-lo.

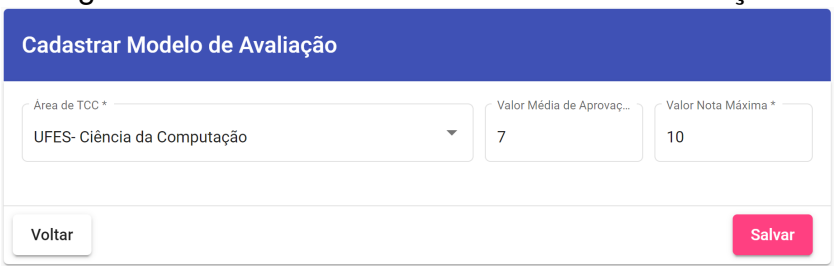
Figura 51: Tela de listagem de modelos de avaliação



Fonte: Do autor

Na Figura 52, exibe a tela de cadastro de um novo modelo de avaliação, na qual é informado qual sua área de TCC, o valor médio e máximo da nota para aprovação.

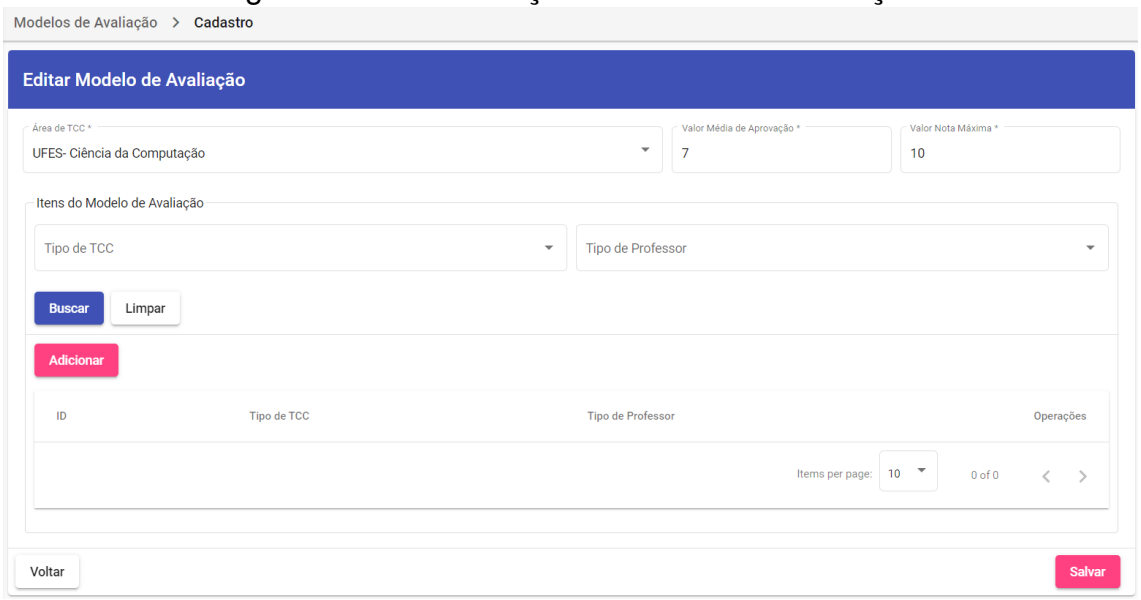
Figura 52: Tela de cadastro de modelo de avaliação



Fonte: Do autor

Após a inclusão bem-sucedida de um modelo de avaliação, o estado da tela é alterado para o modo de edição e exibe-se a funcionalidade de cadastro dos itens desse modelo. Esses itens correspondem ao modelo de avaliação específico para o tipo de TCC e ao tipo de professor. A Figura 53 ilustra a tela de edição.

Figura 53: Tela de edição do modelo de avaliação



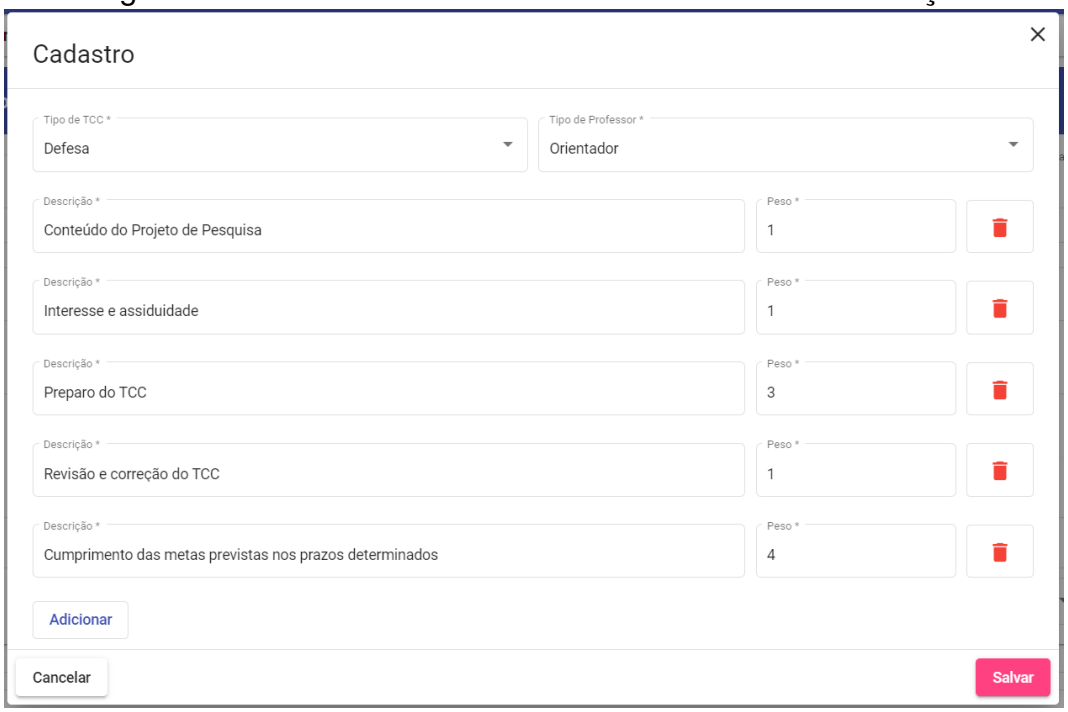
Fonte: Do autor

Na tela de edição, conforme mencionado anteriormente, são exibidas opções para realizar o cadastro dos itens no modelo de avaliação. É possível buscar os itens por tipo de TCC e tipo de professor, adicionar um novo item, bem como editar, visualizar ou excluir itens existentes. Ao clicar no botão “Adicionar”, uma modal de cadastro para esse item é exibida.

Essa modal, ilustrada na Figura 54, apresenta opções para selecionar o tipo de TCC e o tipo de professor. Além disso, permite definir os aspectos de avaliação específicos para esse professor e tipo de TCC. Para cada aspecto de avaliação, são exibidos campos de descrição e peso, acompanhados de um botão de exclusão, caso seja necessário remover o aspecto.

Se houver avaliações em andamento vinculadas ao modelo de avaliação que está sendo editado, quaisquer alterações feitas afetarão essas avaliações. Isso significa que ao editar um aspecto, como o peso ou a descrição, essas alterações também serão refletidas nas avaliações relacionadas. Da mesma forma, ao adicionar um novo aspecto, ele será incluído nas avaliações relacionadas. Por último, se um aspecto for excluído, essa exclusão também será aplicada na avaliação associada.

Figura 54: Modal de cadastro de itens do modelo de avaliação

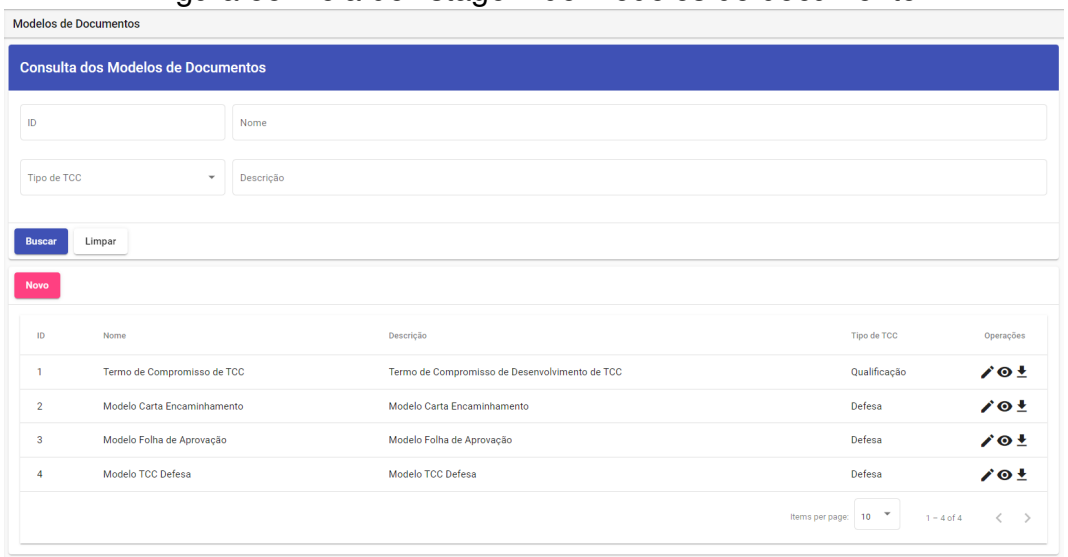


Fonte: Do autor

4.4.9 Telas de Modelo de Documento

A Figura 55 apresenta a interface de listagem dos modelos de documentos. Nessa tela, os usuários têm a capacidade de editar, visualizar ou baixar um modelo de documento específico, além de realizar buscas com filtros personalizados e redirecionar para a tela de cadastro.

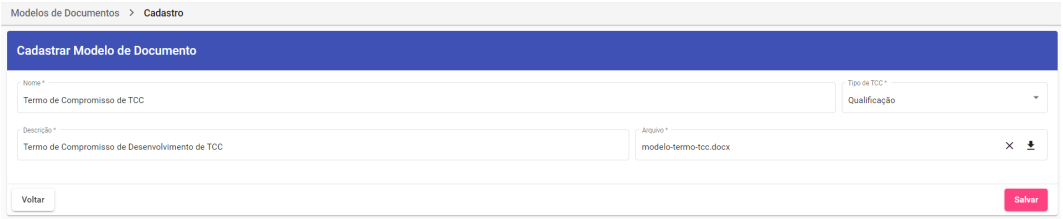
Figura 55: Tela de listagem de modelos de documento



Fonte: Do autor

A Figura 56 apresenta a tela de cadastro de um modelo de documento. Nessa tela, os usuários têm a capacidade de cadastrar um novo modelo de documento, fornecendo informações como o nome do modelo, o tipo de TCC ao qual se refere, uma breve descrição e o anexo correspondente. Além de permitir o cadastro, essa tela também possibilita a edição de modelos já existentes.

Figura 56: Tela de cadastro de modelo de documento

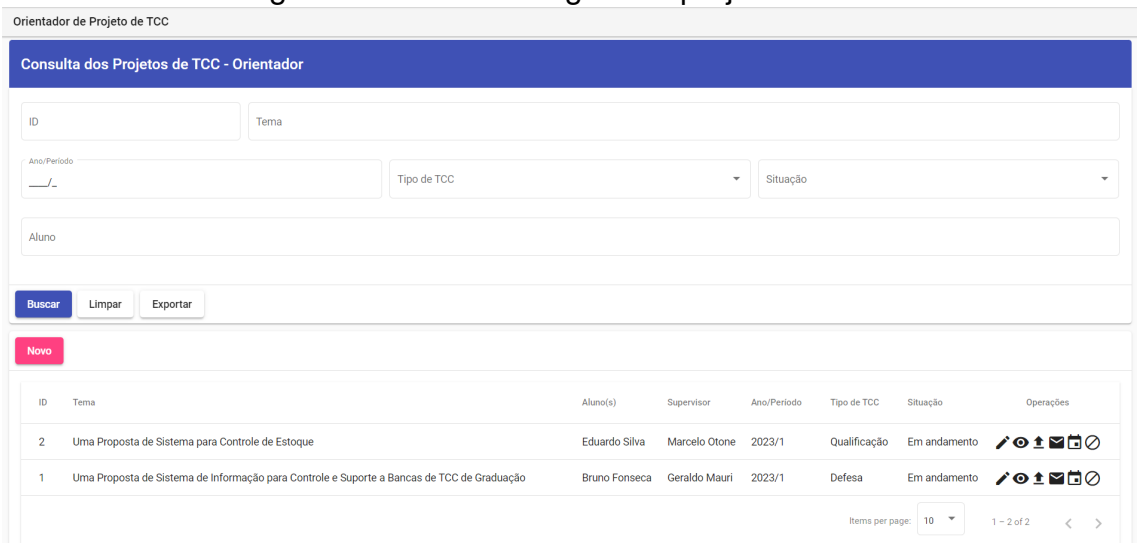


Fonte: Do autor

4.4.10 Telas de Projeto de TCC

A fim de facilitar a busca de TCC's, a Figura 57 apresenta a tela de listagem dos projetos disponíveis. Nessa tela, além de realizar pesquisas específicas, é também possível executar uma variedade de ações relacionadas aos projetos de TCC. Essas ações incluem edição, visualização, anexação de documentos, listagem e convite de membros para a banca, agendamento de apresentações, cancelamento ou reprovação de projetos, entre outras opções. Cada uma dessas ações será descrita com mais detalhes nesta seção.

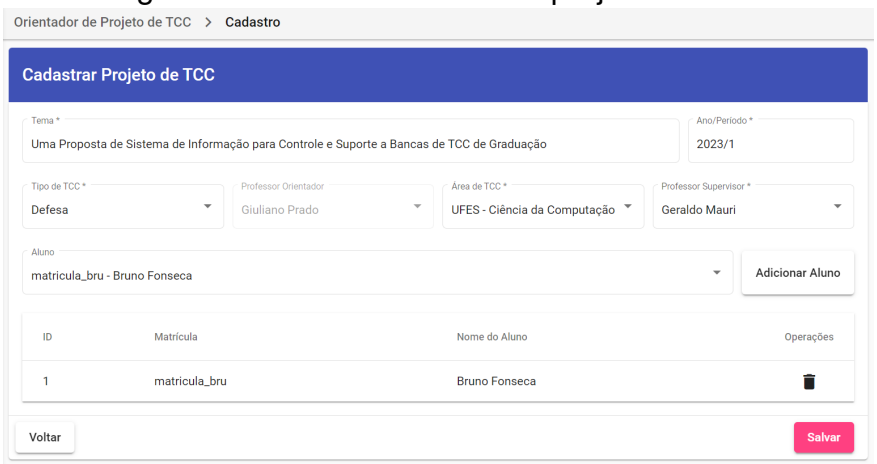
Figura 57: Tela de listagem de projeto de TCC



Fonte: Do autor

A Figura 58 apresenta a tela de cadastro e edição de projetos de TCC. Nessa tela, é possível tanto cadastrar um novo projeto de TCC quanto editar projetos já existentes. Os campos disponíveis para preenchimento incluem tema, ano/período, tipo de TCC, área de TCC, supervisor e os alunos envolvidos. Foi implementada a funcionalidade de cadastrar vários alunos para um mesmo projeto de TCC, visando atender não apenas o Departamento de Computação da UFES - Campus de Alegre, mas também outros departamentos. Além disso, é importante ressaltar que o professor orientador de um novo projeto de TCC será automaticamente o professor logado no sistema.

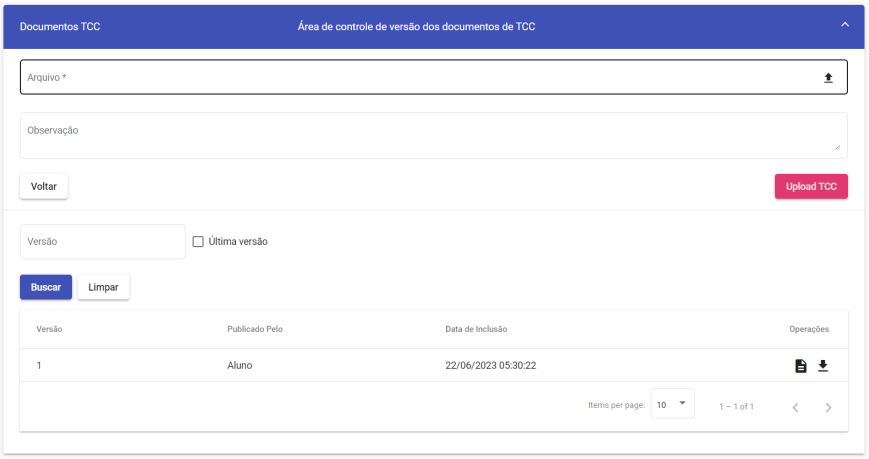
Figura 58: Tela de cadastro de projeto de TCC



Fonte: Do autor

Uma funcionalidade foi criada para anexar documentos em um TCC, oferecendo controle de versão. Os usuários podem adicionar novos documentos com observações relevantes, buscar versões anteriores, acessar observações anteriores e baixar os documentos. Essa funcionalidade está disponível na tela mostrada na Figura 59.

Figura 59: Tela de cadastro dos documentos do projeto de TCC



Fonte: Do autor

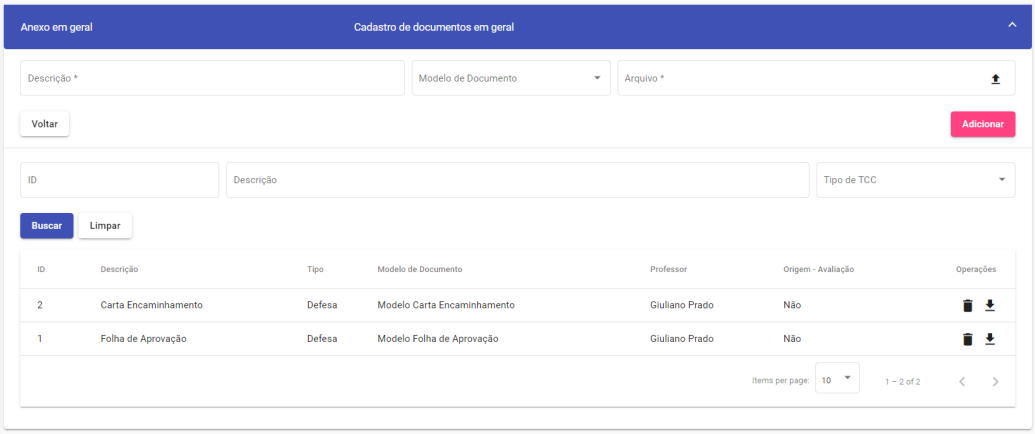
Assim, na Figura 60, é apresentada a tela em que é possível visualizar ou adicionar novas observações em uma versão de TCC. É também possível anexar um documento específico a uma observação, vinculando-a àquela versão em particular. Dessa forma, o professor terá a capacidade de cadastrar observações, enquanto o aluno somente poderá visualizá-las, tendo a opção de baixar os documentos relacionados.

Figura 60: Tela de cadastro das observações do projeto de TCC

Fonte: Do autor

Assim como há uma tela de cadastro de versões de documentos de TCC, também existe uma tela dedicada ao cadastro de documentos em geral. Essa tela é especialmente útil para documentos como “Carta de Encaminhamento” e “Folha de Aprovação”, entre outros. Na Figura 61, é possível visualizar essa tela, na qual os usuários podem anexar documentos, fornecendo informações como descrição, associação com o modelo de documento específico e o próprio arquivo anexo. Além disso, a tela permite a realização de buscas de documentos através de filtros, como ID, descrição e tipo de TCC relacionado.

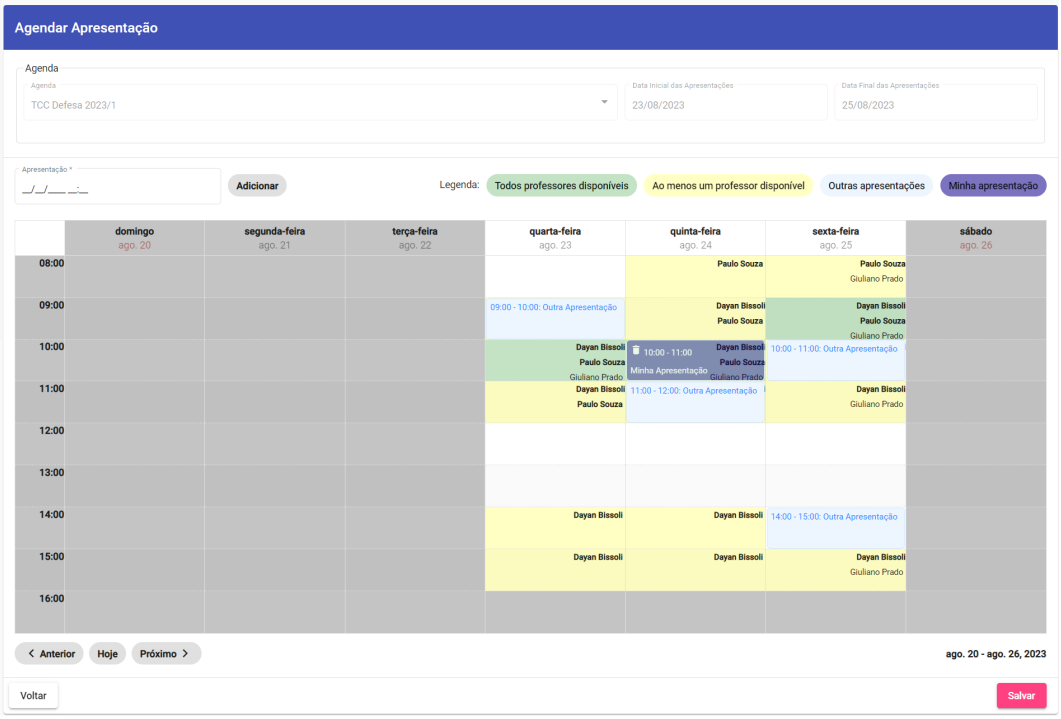
Figura 61: Tela de cadastro dos anexos gerais do projeto de TCC



Fonte: Do autor

A Figura 62 exibe a tela de agendamento da apresentação. Nessa tela, é possível selecionar a agenda desejada e visualizar os limites disponíveis para definição da apresentação. Além disso, é apresentada a disponibilidade dos professores nos diferentes dias e horários, bem como quais horários já estão ocupados com outras apresentações.

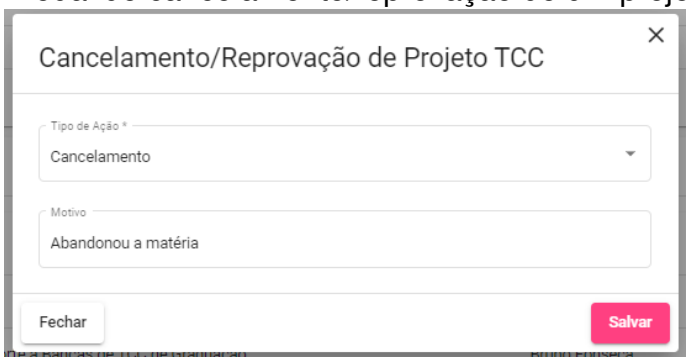
Figura 62: Tela de agendamento da apresentação do projeto de TCC



Fonte: Do autor

Foi desenvolvida uma modal para realizar o cancelamento ou reprovação de TCC. Para utilizar essa funcionalidade, basta informar o tipo de ação desejada (cancelamento ou reprovação) e fornecer uma descrição do motivo que justifica essa ação. A Figura 63 exibe a interface gráfica dessa modal.

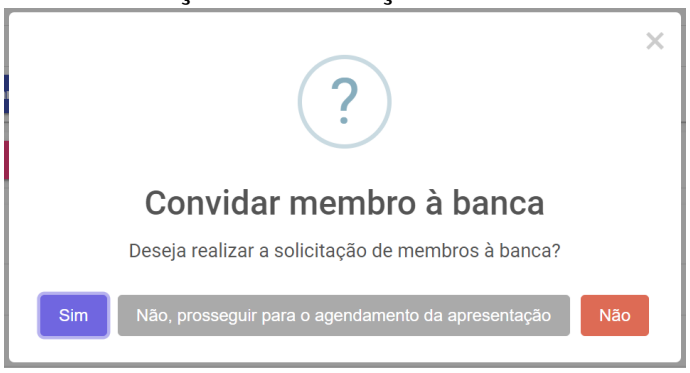
Figura 63: Modal de cancelamento/reprovação de um projeto de TCC



Fonte: Do autor

Foi desenvolvida uma funcionalidade que permite o convite de membros para compor a banca de avaliação. Essa funcionalidade consiste em um procedimento capaz de enviar convites por e-mail aos professores selecionados. No entanto, antes de enviar o convite, há um procedimento adicional caso o projeto de TCC esteja em fase de "Qualificação". Nesse caso, é exibida uma janela de confirmação, na qual o usuário pode escolher se deseja ou não convidar o professor selecionado, além de ser oferecida a opção de redirecionamento para a tela de agendamento da apresentação sem convidar membros à banca. A Figura 64 ilustra essa janela de confirmação.

Figura 64: Modal de confirmação da solicitação de membro à banca - Qualificação



Fonte: Do autor

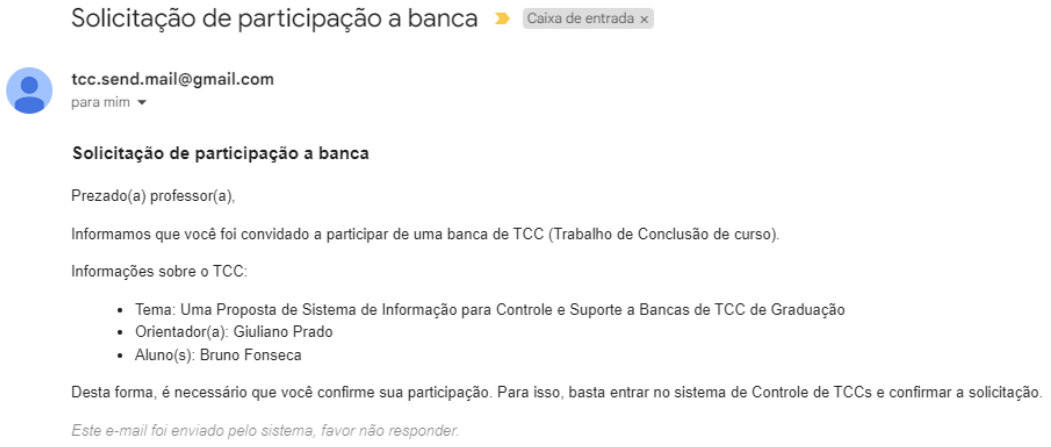
Conforme mencionado anteriormente, foi desenvolvida uma modal (conforme ilustrado na Figura 65) com o objetivo de enviar convites por e-mail aos professores, solicitando sua confirmação para participar da banca de avaliação do TCC. Essa modal apresenta um campo para selecionar o professor desejado, além de uma tabela que lista os professores já contatados, exibindo se eles confirmaram ou não a solicitação. Também é possível excluir uma solicitação específica caso necessário.

Figura 65: Modal de solicitação de membro à banca

Fonte: Do autor

O e-mail de solicitação à banca de avaliação pode ser visualizado na Figura 66. Neste e-mail, é apresentada uma mensagem que informa o convite, além de fornecer informações relevantes sobre o projeto de TCC ao qual o convite se refere, como o tema, o orientador e os alunos envolvidos.

Figura 66: E-mail de solicitação para membro de banca


Solicitação de participação a banca Caixa de entrada ×

tcc.send.mail@gmail.com
para mim

Solicitação de participação a banca

Prezado(a) professor(a),

Informamos que você foi convidado a participar de uma banca de TCC (Trabalho de Conclusão de curso).

Informações sobre o TCC:

- Tema: Uma Proposta de Sistema de Informação para Controle e Suporte a Bancas de TCC de Graduação
- Orientador(a): Giuliano Prado
- Aluno(s): Bruno Fonseca

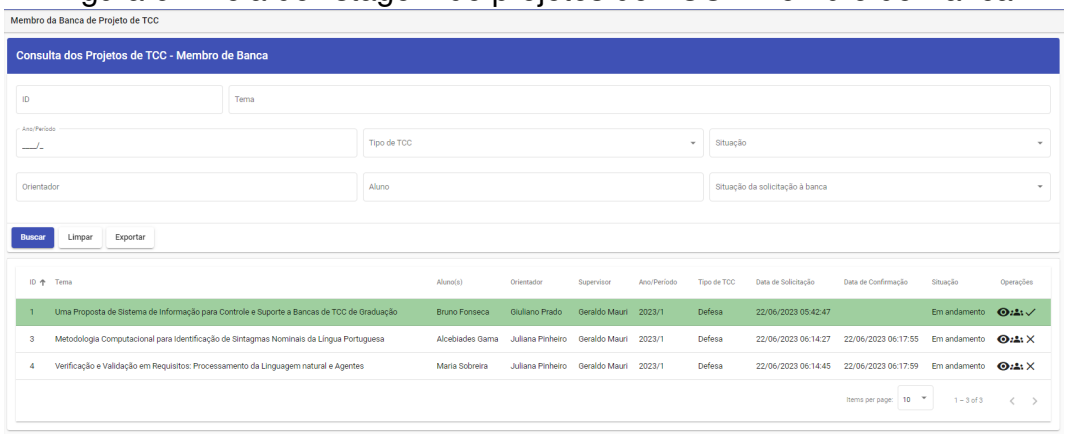
Desta forma, é necessário que você confirme sua participação. Para isso, basta entrar no sistema de Controle de TCCs e confirmar a solicitação.

Este e-mail foi enviado pelo sistema, favor não responder.

Fonte: Do autor

Após o envio do e-mail, o professor convidado terá a oportunidade de visualizar os projetos de TCC que aguardam sua confirmação. Para isso, a Figura 67 ilustra a mesma tela de listagem de projetos de TCC mencionada anteriormente na Figura 57. No entanto, nesta tela específica, os projetos de TCC que aguardam a confirmação do professor solicitado são destacados. É possível confirmar ou cancelar a solicitação de participação diretamente nesta tela.

Figura 67: Tela de listagem de projetos de TCC - Membro de Banca


Membro da Banca de Projeto de TCC

Consulta dos Projetos de TCC - Membro de Banca

ID | Tema | Ano/Período | Tipo de TCC | Situação | Orientador | Aluno | Situação da solicitação à banca

Buscar | Limpar | Exportar

ID	Tema	Aluno(s)	Orientador	Supervisor	Ano/Período	Tipo de TCC	Data de Solicitação	Data de Confirmação	Situação	Operações
1	Uma Proposta de Sistema de Informação para Controle e Suporte a Bancas de TCC de Graduação	Bruno Fonseca	Giuliano Prado	Geraldo Mauri	2023/1	Defesa	22/06/2023 05:42:47		Em andamento	
3	Metodologia Computacional para Identificação de Sintagmas Nominais da Língua Portuguesa	Alcebiades Gama	Juliana Pinheiro	Geraldo Mauri	2023/1	Defesa	22/06/2023 06:14:27	22/06/2023 06:17:55	Em andamento	
4	Verificação e Validação em Requisitos: Processamento da Linguagem natural e Agentes	Maria Sobreira	Juliana Pinheiro	Geraldo Mauri	2023/1	Defesa	22/06/2023 06:14:45	22/06/2023 06:17:59	Em andamento	

Items per page: 10 1 – 3 of 3

Fonte: Do autor

Além disso, em relação às funcionalidades do projeto de TCC, é possível exportar os dados em uma planilha a partir da tela de listagem dos projetos. Essa planilha, apresentada pela Figura 68, no formato CSV, contém informações essenciais, como ID, Tema, Alunos, Orientador, Supervisor, Membros da Banca, Ano/Período, Etapa do TCC e Situação do TCC. Durante a exportação, os dados podem ser filtrados de acordo com as opções disponíveis na tela de listagem. No entanto, é importante destacar que, ao contrário da exibição paginada na tela, a consulta na planilha inclui todos os registros que correspondem ao filtro selecionado, sem limitações de paginação.

Figura 68: Planilha de exportação de projeto de TCC

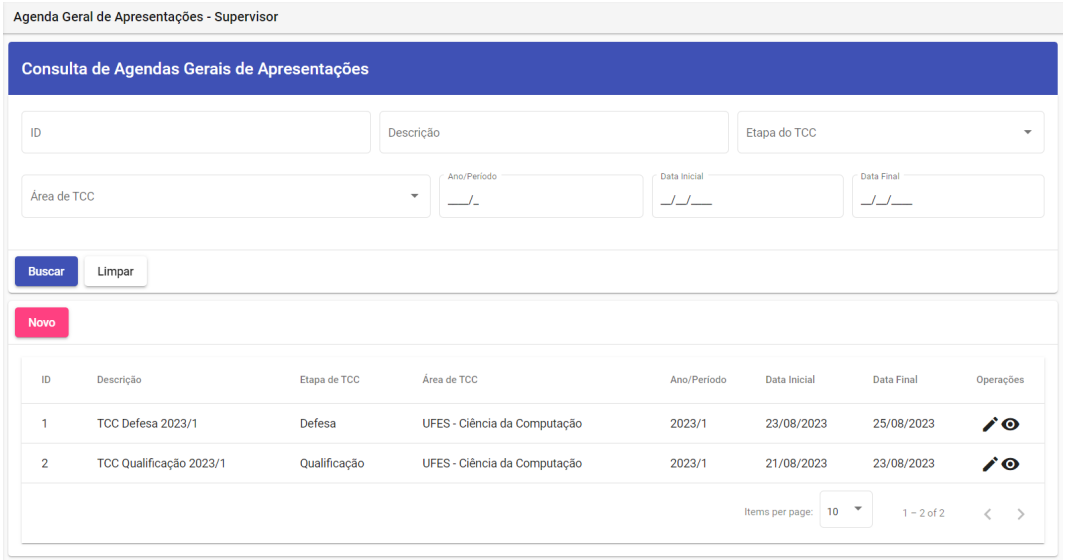
ID	Tema	Alunos	Orientador	Supervisor	Membros da Banca	Ano/Período	Etapa de TCC	Situação do TCC
1	Uma Proposta de Sistema de Informação para Controle e Suporte a Bancas de TCC de Graduação	Bruno Fonseca	Giuliano Prado	Geraldo Mauri	Dayan Bissoli, Edmar Hell	2023/1	Defesa	Em avaliação
2	Uma Proposta de Sistema para Controle de Estoque	Eduardo Silva	Giuliano Prado	Marcelo Otone		2023/1	Qualificação	Cancelado
5	Uma Proposta de Sistema para Controle de Produto	Eduardo Silva	Giuliano Prado	Marcelo Otone		2023/1	Qualificação	Em andamento

Fonte: Do autor

4.4.11 Telas de Agenda

Nesta seção, serão apresentadas e descritas as telas relacionadas à agenda da apresentação. A Figura 69 mostra a tela de listagem das agendas, na qual é possível realizar buscas utilizando filtros. Para cada item na tabela, há a opção de redirecionar para a tela de edição ou visualização. Além disso, há a opção de acessar a tela de cadastro de uma nova agenda.

Figura 69: Tela de listagem de agendas de apresentação

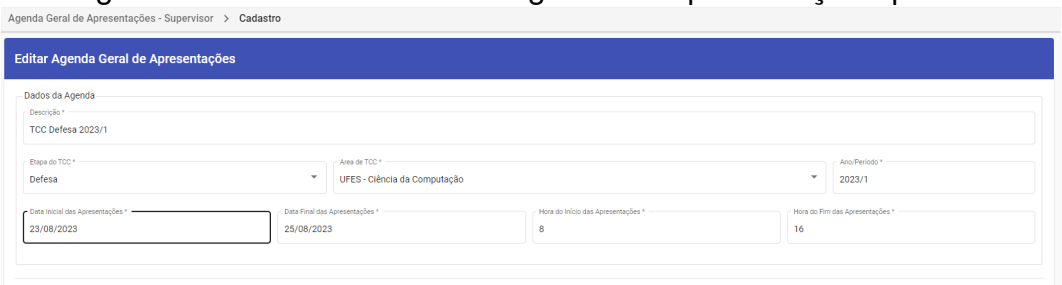


Fonte: Do autor

A tela de cadastro por ser extensa, suas informações foram divididas em duas Figuras: a Figura 70, que representa o formulário da agenda, e a Figura 71, que mostra o cadastro das restrições das agendas.

A Figura 70 apresenta a seção inicial do cadastro, na qual é possível definir uma descrição para a agenda, o tipo de TCC ao qual se refere, a área de TCC e o ano/período correspondente. Além disso, são exibidos os campos relacionados às agendas de apresentações, incluindo as datas e horários limite para as apresentações.

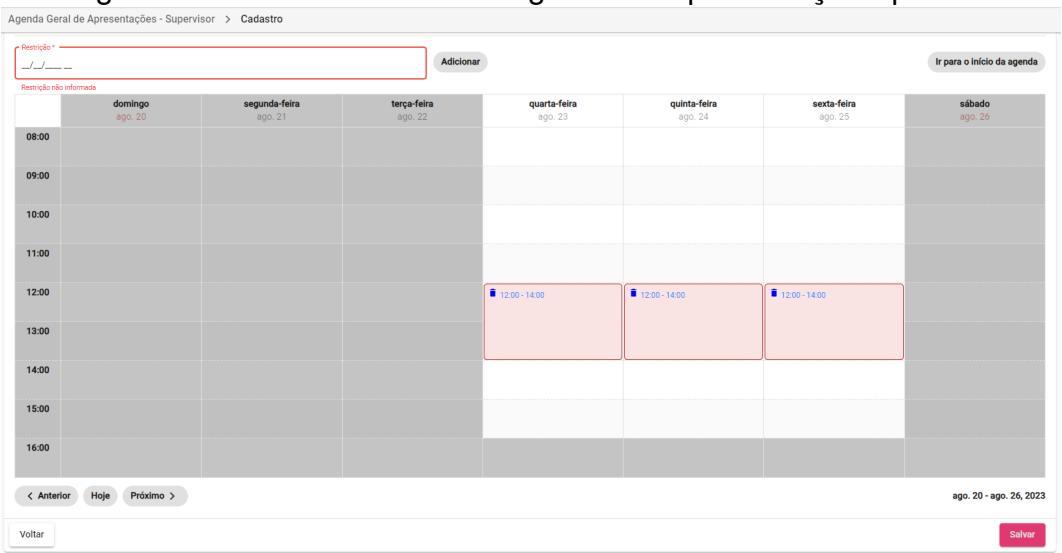
Figura 70: Tela de cadastro de agenda de apresentação - parte 1



Fonte: Do autor

Com base nas informações fornecidas na tela anterior, a agenda de apresentação é gerada na tela ilustrada na Figura 71, levando em consideração os limites informados. Essa abordagem garante que a agenda seja construída de forma a bloquear automaticamente a parte que ficaria fora dos limites estabelecidos. Além disso, há a possibilidade de cadastrar restrições no espaço disponível, o qual as apresentações não poderão ser agendadas.

Figura 71: Tela de cadastro de agenda de apresentação - parte 2



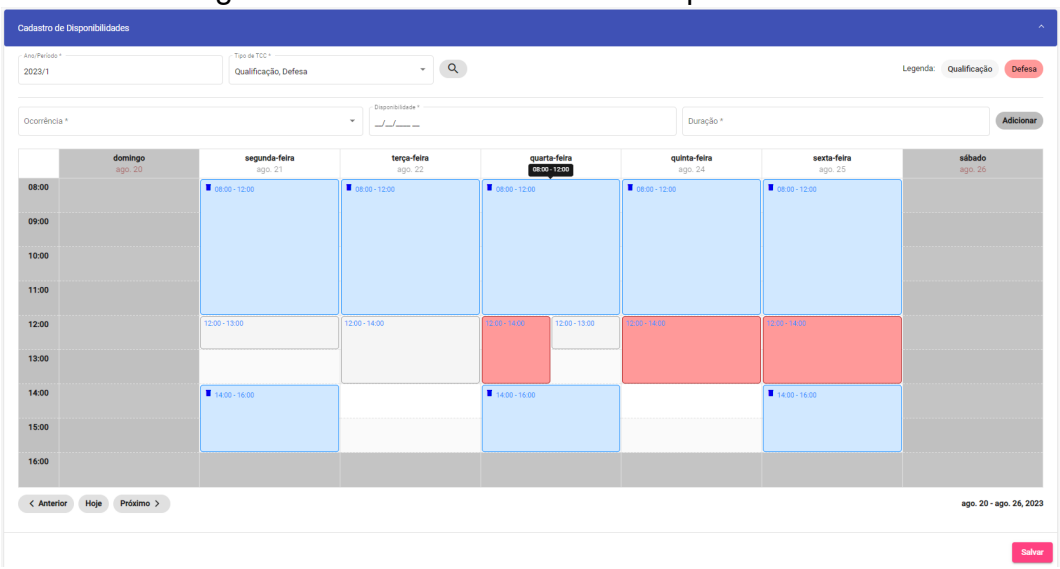
Fonte: Do autor

Para agendar as apresentações de um projeto de TCC, é necessário realizar o cadastro de uma agenda específica. Além disso, os professores envolvidos precisam registrar suas disponibilidades. Essa funcionalidade é ilustrada na Figura 72, que apresenta uma interface intuitiva para esse propósito. Nessa tela, as restrições de qualificação e defesa são destacadas com cores distintas: cinza para as restrições de qualificação e vermelho para as restrições de defesa.

Ao cadastrar as disponibilidades, os professores podem selecionar o tipo de TCC (Qualificação ou Defesa), podendo ser um ou ambos. A tela de cadastro também permite definir a frequência das disponibilidades, como todas as segundas-feiras, todos os dias úteis ou todos os dias, entre outras opções, oferecendo flexibilidade aos professores. Além disso, é necessário informar a data e a duração da ocorrência específica.

Dessa forma, as disponibilidades dos professores são visualizadas como blocos de cor azul na interface, facilitando a identificação e a gestão do cronograma de apresentações.

Figura 72: Tela de cadastro das disponibilidades



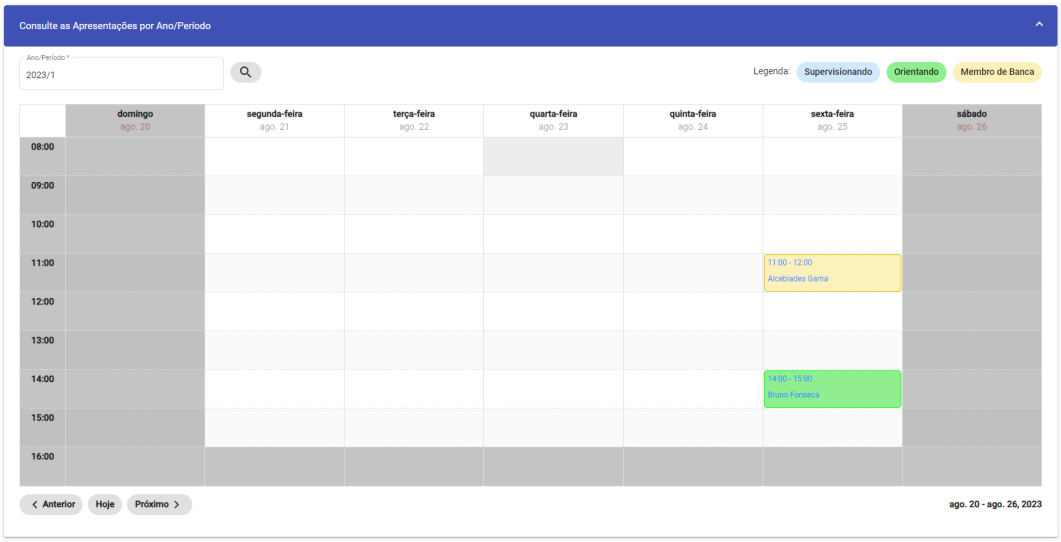
Fonte: Do autor

Existem diversos processos que envolvem agendas, especialmente quando se trata da organização de datas e horários. Com o objetivo de proporcionar uma visualização abrangente, foi desenvolvida uma tela específica para exibir todas as apresentações relacionadas ao professor logado, independentemente de seu papel como supervisor, orientador ou membro de banca. Essa tela de visualização, representada na Figura 73, oferece a possibilidade de buscar as agendas de apresentação pelo ano/período,

apresentando todas as atividades vinculadas ao professor em questão.

Uma característica importante é a representação visual das apresentações por meio de cores distintas, o que facilita a associação entre o professor e cada atividade. Adicionalmente, uma legenda explicativa é exibida, esclarecendo o significado de cada cor utilizada.

Figura 73: Tela da minha agenda



Fonte: Do autor

4.4.12 Telas de Avaliação

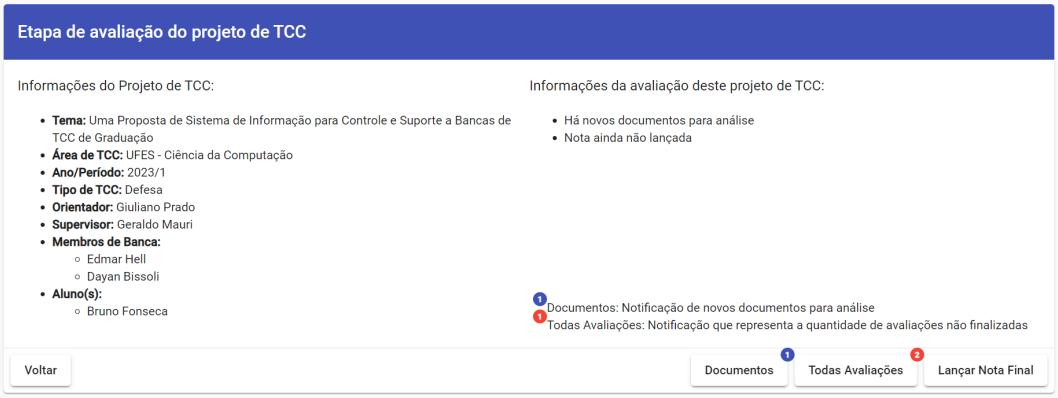
Nesta seção, serão apresentadas as telas referentes à avaliação do projeto de TCC. A primeira tela, mostrada na Figura 74, exibe a avaliação em andamento, apresentando informações gerais sobre o projeto de TCC em questão, tais como: tema, área de TCC, ano/período, orientador, supervisor, membros da banca e alunos. Além disso, são exibidas informações relevantes sobre a avaliação, como a presença de novos documentos para análise, se todas as avaliações foram concluídas e se a nota já foi lançada.

Essa tela é acessível a todos os usuários do sistema que estejam envolvidos no projeto de TCC em questão. No entanto, as ações exibidas variam para cada tipo de usuário. Por exemplo, uma opção “Avaliar” é mostrada para os tipos de professores que foram configurados no modelo de avaliação associado a esse projeto de TCC. Além disso, é exibido um botão “Documentos” que redireciona para uma área contendo os documentos relacionados ao processo de avaliação, incluindo a versão final

do TCC e outros documentos necessários para a avaliação. Esse botão é destacado quando há uma versão do TCC ainda não visualizada pelo usuário logado.

Para o supervisor de TCC tem uma opção adicional chamada “Todas Avaliações”. Ao clicar nesse botão, uma modal é exibida com a lista de todas as avaliações, indicando se foram realizadas ou não. O botão é destacado para chamar a atenção e exibe a quantidade de avaliações restantes a serem concluídas.

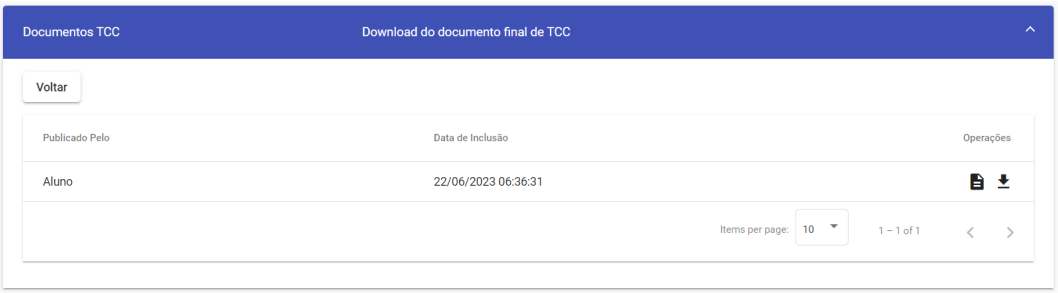
Figura 74: Tela da avaliação do projeto de TCC



Fonte: Do autor

A Figura 75 mostra a tela com a versão mais recente do documento de TCC, na qual é possível realizar o *download* e, se necessário, redirecionar para uma tela de observações. Essa funcionalidade permite ao professor adicionar comentários adicionais, caso deseje.

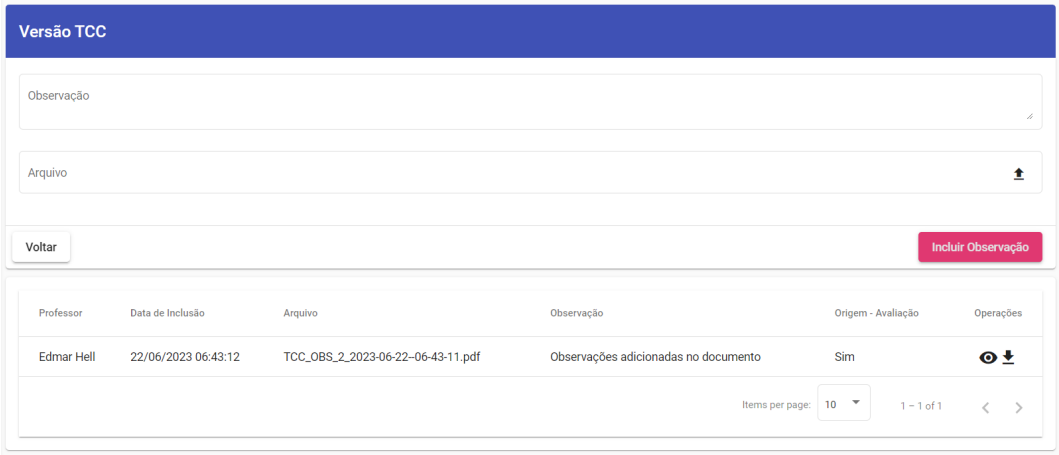
Figura 75: Tela de documentos de TCC na etapa de avaliação



Fonte: Do autor

A Figura 76 ilustra a tela de observações mencionada anteriormente. Nessa tela, o professor tem a capacidade de anexar novos documentos relacionados a observações específicas, permitindo que o aluno visualize e faça as correções necessárias, quando aplicável.

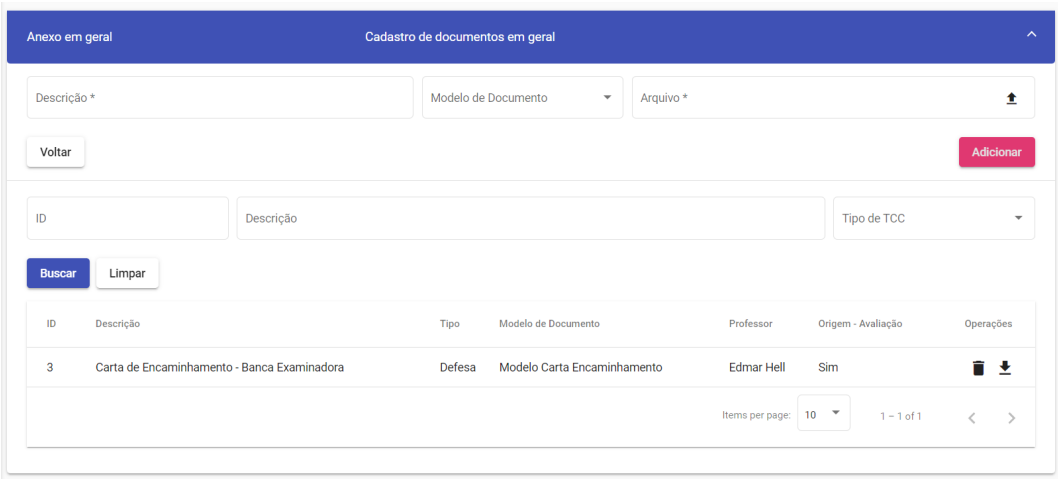
Figura 76: Tela de observação do documento de TCC na etapa de avaliação



Fonte: Do autor

A tela exibida na Figura 77 apresenta funcionalidades relacionadas aos anexos em geral, que são relevantes para o processo de avaliação. Nessa tela, é possível realizar o *upload* de novos documentos nesse contexto, além de efetuar o *download* de anexos já cadastrados, excluí-los e realizar uma busca por meio de filtros. Ao cadastrar um novo anexo, a tela solicita informações como descrição, o tipo de documento ao qual se refere e o próprio arquivo anexado. Vale ressaltar que a definição do tipo de documento é opcional nesse caso, podendo ou não ser especificado.

Figura 77: Tela de anexos gerais de TCC na etapa de avaliação



Fonte: Do autor

Quando um professor clica no botão “Avaliar”, uma janela modal é aberta, carregando os aspectos definidos no modelo de avaliação associado ao projeto de TCC em questão. Nessa janela, os aspectos são apresentados, solicitando ao professor que atribua um valor a cada um deles. Também são exibidos o valor máximo e a média da nota necessária para aprovação, previamente configurados no modelo. Além disso, com base

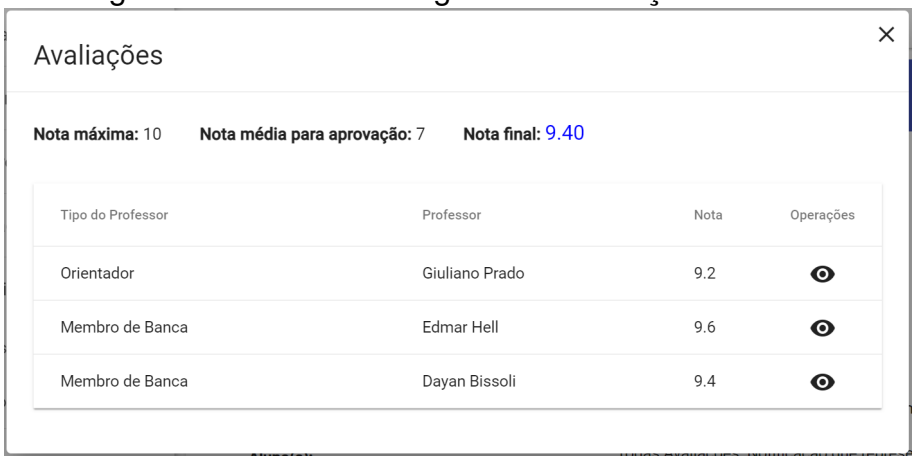
nos valores inseridos, é realizado o cálculo da nota final específica para a avaliação desse professor, levando em consideração as notas atribuídas a cada aspecto.

Figura 78: Modal de avaliação do TCC

Fonte: Do autor

Sendo assim, o professor supervisor poderá visualizar todas as avaliações realizadas. Para esse fim, foi desenvolvida uma modal (apresentada na Figura 79), que exibe uma lista dos tipos de avaliação, juntamente com o respectivo professor responsável e a nota atribuída por ele. Além disso, nesta tela também são apresentadas informações sobre a nota máxima possível, a nota média e a nota final do projeto de TCC em questão. A nota final é calculada com base na média das notas atribuídas pelos professores.

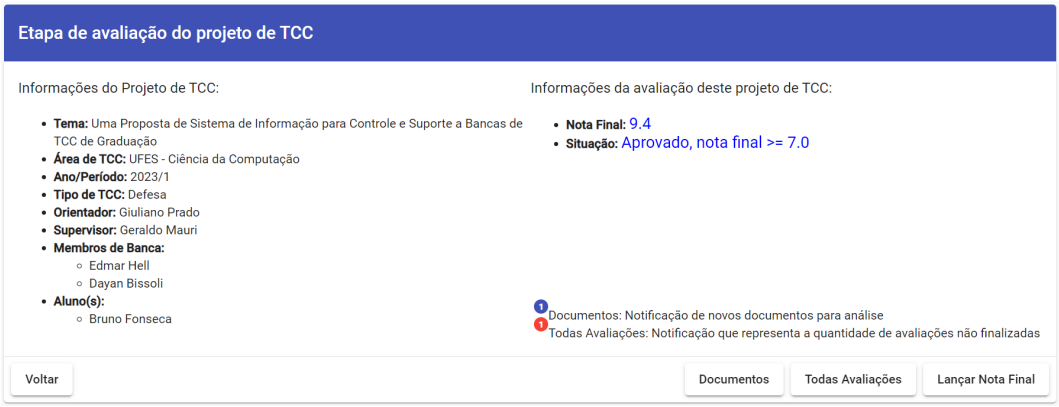
Figura 79: Modal de listagem das avaliações do TCC



Fonte: Do autor

Após a conclusão das avaliações realizadas por todos os professores envolvidos, o professor supervisor poderá lançar a nota final, atualizando assim a situação do projeto de TCC. Com isso, será possível liberar aos alunos associados a este projeto as informações sobre sua situação e o valor obtido na nota final. A Figura 80 apresenta a tela atualizada após a finalização das avaliações.

Figura 80: Tela de avaliação de projeto de TCC finalizada



Fonte: Do autor

5 TESTES E VALIDAÇÕES

O objetivo do teste é demonstrar a capacidade de um programa em cumprir sua finalidade e identificar possíveis falhas antes de sua utilização efetiva. Durante a fase de testes do software, o programa é executado utilizando dados simulados. Os resultados desses testes são minuciosamente analisados em busca de erros, anomalias ou quaisquer informações relacionadas aos atributos não funcionais do programa. É importante ressaltar que a realização desses testes contribui para a melhoria contínua da qualidade e confiabilidade do software (SOMMERVILLE, 2015).

5.1 Inspeções

As inspeções são fundamentais para garantir a qualidade de um projeto. Essas atividades de controle de qualidade envolvem uma análise minuciosa do software, sua documentação e os registros do processo, com o objetivo de identificar erros, omissões e garantir a conformidade com os padrões de qualidade estabelecidos. Essas verificações são essenciais para assegurar que os produtos ou serviços desenvolvidos atendam às expectativas e requisitos estabelecidos (SOMMERVILLE, 2015). A seguir, é apresentado a tabela 3, que relata os erros encontrados e suas soluções.

Tabela 3: Tabela de inspeções

Falha	**Solução**
Na importação via planilha, ao colocar uma planilha com colunas a menos, resultava um erro de importação.	Foi adicionado um tratamento de erros para quando ocorrer este erro em específico, considerar um erro no registro referente aquela planilha.

Falha	**Solução**
Ao tentar excluir um modelo de avaliação, notou-se que o mesmo não estava sendo excluído.	Foi corrigido o parâmetro do método http DELETE na *controller*, que estava com o nome incorreto.
Notou-se que era possível editar os membros da banca mesmo após a finalização de um TCC.	Foi retirado essa opção quando o TCC estiver em situação final.
Estava dando erro quando o usuário administrador tentava cadastrar uma área de TCC.	Foi adicionado a permissão para realizar este cadastro.
Foi identificado um problema em que era possível excluir um modelo de avaliação mesmo quando havia projetos de TCC em processo de avaliação. Essa situação resultava em um erro nas telas de avaliação.	Foi adicionado uma validação para garantir a integridade dos projetos de TCC em avaliação. Agora, não é mais possível excluir o modelo de avaliação associado a um projeto enquanto este estiver em processo de avaliação.

Fonte: Do autor

5.2 Teste de Desenvolvimento

De acordo com Sommerville (2015) os testes de desenvolvimento desempenham um papel fundamental na garantia da qualidade do software ao longo do processo de desenvolvimento. Essas atividades são conduzidas pela equipe responsável pela criação do sistema e têm como objetivo identificar e corrigir erros, assegurando que o software cumpra os requisitos e funcione conforme o esperado.

No sistema desenvolvido, foi adotada uma abordagem em que os testes de desenvolvimento ocorreram de forma contínua e iterativa. A cada nova funcionalidade desenvolvida, foram realizados testes em paralelo, não apenas após a conclusão, mas também durante a fase de construção. Isso permitiu uma validação constante do código em desenvolvimento, assegurando a qualidade ao longo de todo o processo. A tabela 4 mostra os resultados dos testes realizados.

Tabela 4: Resultado dos testes de desenvolvimento

Falhas	**Teste realizado**	**Motivo (erro)**	**Solução**
Ao editar algum projeto de TCC, estava perdendo a informação da sua situação atual, e seu orientador.	Edição de Projeto de TCC	Estava realizando dois *updates* em uma mesma chamada.	Foi alterado para realizar apenas um *update* na chamada.
Ao salvar a agenda de apresentação, a etapa do TCC não estava salvando.	Cadastro de agendas de apresentações	Faltou adicionar o @Enumerated na coluna do banco.	Foi adicionado o @Enumerated na coluna do banco.
Ao entrar na tela de visualização de apresentação, era possível editar a apresentação.	Agendamento de apresentação	Faltou desabilitar o componente de apresentação quando a tela estivesse em estado de visualização	Foi adicionado o código para desabilitar o componente de apresentação quando a tela estivesse em estado de visualização.
Ao anexar um documento no componente de anexo, e depois retirar o documento, não era possível adicionar o mesmo documento.	Anexo de documento	Quando era retirado o documento do componente, permanecia o nome dele, e assim não pegava o *change* do componente quando tentava anexar o mesmo documento.	Foi adicionado um código que ao remover o documento, era removido também todas as informações do documento daquele componente.

Falhas	Teste realizado	Motivo (erro)	Solução
Ao desativar um usuário, o mesmo conseguia logar no sistema.	Login no sistema	Na autenticação, não estava consultando a informação se estava ativo ou não.	Foi adicionado na autenticação, a verificação se o usuário em questão estava ativo.
Os ícones e fontes não funcionavam quando não havia internet.	Execução do sistema local sem internet	As informações de ícones e fontes eram carregadas por um link externo.	Foi adicionado os arquivos referentes aos ícones e fontes no próprio projeto.
Não estava sendo possível redirecionar para tela de documentos a partir da tela de projeto de TCC.	Acesso a tela de documentos	As rotas para acesso desta tela por meio da tela de projeto de TCC estava definida incorretamente.	Foi corrigido as rotas referentes a essas telas, definindo que a tela de documentos também é uma tela filha da tela de projeto de TCC.
Não estava sendo alterado o ano/-período quando o projeto de TCC era iniciado a etapa de defesa.	Iniciar etapa de defesa	Não foi incrementado o ano/período do projeto de TCC em questão.	Foi adicionado o incremento do ano/período ao iniciar a etapa de defesa.
Não estava sendo exibindo o professor supervisor na tela de projeto de TCC no modo de edição ou visualização.	Editar/Visualizar projeto de TCC	Não estava sendo carregado o campo com as informações de supervisor.	Foi adicionado a chamada do método para carregar as informações de supervisor.

Fonte: Do autor

5.3 Teste de Usuário

O teste de usuário é uma fase essencial do processo de teste, na qual os usuários experimentam um sistema ou produto de software para fornecer *feedback* sobre sua usabilidade, funcionalidade e adequação às suas necessidades. Esse *feedback* direto dos usuários desempenha um papel fundamental no aprimoramento contínuo do software, garantindo uma experiência satisfatória e um produto final confiável que atenda às expectativas dos usuários (SOMMERVILLE, 2015).

Existem três tipos de testes de usuário no desenvolvimento de software:

- Teste alfa: Os usuários trabalham diretamente com a equipe de desenvolvimento nas instalações dos desenvolvedores para identificar e corrigir problemas antes do lançamento oficial.
- Teste beta: Uma versão preliminar do software é disponibilizada aos usuários externos, que fornecem feedback sobre usabilidade e relatam problemas para que sejam corrigidos antes do lançamento.
- Teste de aceitação: Os clientes testam o sistema para determinar se ele atende aos requisitos e está pronto para ser implantado em seu ambiente, verificando se todas as funcionalidades necessárias estão presentes.

A realização desses testes é crucial para aprimorar a qualidade do software, garantir a satisfação dos usuários e evitar problemas futuros. Nesse sentido, considerando que o orientador deste projeto é considerado um usuário do sistema desenvolvido, foram conduzidas reuniões de validação e testes em várias funcionalidades, caracterizando, portanto, um teste alfa.

No teste alfa, foram levantados os seguintes *feedbacks*:

- Ao acessar a tela de projeto de TCC, o supervisor não deve visualizar a coluna referente ao supervisor.
- Alterar a palavra “ações” na coluna das tabelas para “operações”.
- Ajustar as agendas para que sejam intervalos de 1 hora.
- Adicionar a possibilidade de cadastrar ocorrências na tela de disponibilidades.

- Adicionar a opção de remover uma ocorrência completa na tela de disponibilidades, caso seja necessário.
- Incluir nos cabeçalhos das planilhas de importação os padrões aceitáveis ou uma breve descrição que auxilie no preenchimento das colunas.
- Retirar os campos de CPF e RG de todo o sistema.
- Ao editar um modelo de avaliação, editar também os itens de avaliação ativos.
- Adicionar a média final na tela de visualização das avaliações pelo supervisor.
- Adicionar um *checkbox* informando se o professor logado já realizou a avaliação.
- Adicionar informação no processo de avaliação, notificando que existem novos documentos para análise e quantos professores faltam para avaliar.

5.4 Teste de Usabilidade

Segundo Rubin e Chisnell (2011), o teste de usabilidade é um procedimento fundamental, no qual participantes representativos desempenham um papel crucial na avaliação detalhada do nível de usabilidade de um produto. Ao levar em consideração critérios específicos, eles fornecem *feedbacks*, permitindo identificar pontos fortes e áreas de melhoria, e contribuindo para aprimorar a experiência do usuário de forma significativa. Esse processo desempenha um papel essencial no desenvolvimento de produtos mais intuitivos, eficientes e satisfatórios para os usuários finais.

A seguir é apresentado a tabela 5, que relata os pontos levantados em reuniões realizadas com o professor orientador.

Tabela 5: Resultado dos testes de usabilidade

Defeitos e observações	**Solução**
A validação da importação de dados costumava ser realizada de forma incremental, exigindo que o usuário fizesse várias tentativas até resolver todos os problemas.	Foram feitas alterações na validação da importação. Agora, quando o usuário importar a planilha e ocorrerem erros, todos os erros referentes a essa importação serão exibidos.

Defeitos e observações	Solução
Títulos não muito intuitivos.	Foram alterados os títulos para torná-los mais intuitivos.
Ao invés do professor informar suas restrições na agenda, seria melhor informar suas disponibilidades.	Foi alterado para os professores cadastrarem suas disponibilidades ao invés de suas restrições na agenda.
Não possibilitar professor confirmar solicitação de convite à banca, caso este não tenha cadastrado suas disponibilidades.	Foram implementadas validações adicionais para impedir essa ação.
Não é permitir agendar a apresentação caso os membros da banca não estejam cadastrados ou não tenham confirmado sua participação.	Foram implementadas validações adicionais para impedir essa ação.
Destacar as linhas dos projetos de TCC que possuem convites pendentes para os membros da banca.	Foi adicionada uma verificação para que, quando a data de confirmação estiver vazia, a linha mude de cor.
Colocar no agendamento de apresentação o nome dos membros da banca em negrito.	Foi implementada uma verificação para destacar em negrito os nomes dos membros de banca na tela de agendamento da apresentação.
Quando não parametrizado a avaliação para algum tipo de professor, não exibir o botão de avaliação.	Foi adicionado uma verificação, que o botão de avaliação só será visível para os professores que estão parametrizados.

Fonte: Do autor

5.5 Avaliação dos Testes Realizados

Nesta seção, será apresentado uma avaliação sobre os resultados obtidos por meio dos testes realizados. Será feita uma análise global dos testes, discutindo a sua satisfação, os resultados alcançados e a capacidade de atestar a qualidade do sistema com base nessas avaliações.

5.5.1 Satisfação dos Testes

Durante a fase de testes, o sistema passou por avaliações realizadas pelo desenvolvedor, que por ser um aluno, é também um usuário do sistema, assim como pelo orientador, que também desempenha o papel de usuário.

O orientador participou de algumas reuniões de testes para fornecer *feedback* valioso. Além disso, durante a etapa de desenvolvimento, os testes foram realizados por um dos usuários finais, uma vez que o desenvolvedor também é um aluno. Adicionalmente, os testes de usuário e usabilidade foram realizados em conjunto com o orientador deste projeto, que também atua como usuário final. Dessa forma, o usuário final do sistema esteve ativamente envolvido tanto no desenvolvimento quanto nos testes, tanto pelo aluno quanto pelo professor orientador, resultando em testes satisfatórios e bem-sucedidos neste projeto.

Por ter esta abordagem, permitiu-se obter uma perspectiva interna e externa sobre a satisfação em relação à utilização do sistema.

5.5.2 Resultados Alcançados

Durante os testes realizados no sistema, foram identificados alguns erros que foram prontamente tratados durante o processo de desenvolvimento. Essa abordagem permitiu que as funcionalidades fossem validadas e o sistema funcionasse de maneira adequada.

Além disso, o professor orientador ofereceu sugestões importantes para melhorar a usabilidade e o funcionamento do sistema. Essas sugestões foram cuidadosamente consideradas e implementadas, resultando em melhorias significativas na experiência do usuário e no desempenho do sistema como um todo.

Através desse processo iterativo de identificação de erros e incorporação de sugestões, foi possível aprimorar a qualidade do sistema, garantindo sua confiabilidade e eficiência no cumprimento de suas funcionalidades.

5.5.3 Análise Global dos Testes

A análise global dos testes realizados no sistema de controle e suporte a bancas de TCC evidencia resultados satisfatórios em termos de eficiência, confiabilidade e ade-

quação às necessidades dos usuários. Através dos testes, foi possível atestar a qualidade do sistema, proporcionando um ambiente eficiente e confiável para a realização das bancas de TCC. Os resultados obtidos validam o desenvolvimento do sistema, destacando a sua relevância e contribuição para a gestão do processo de trabalhos de conclusão de curso.

6 CONSIDERAÇÕES FINAIS E CONCLUSÕES

O desenvolvimento do sistema revelou-se uma solução abrangente e potencialmente eficiente para auxiliar no processo de construção e defesa dos trabalhos de conclusão de curso. Ao proporcionar uma organização aprimorada aos alunos, o sistema contribui para que eles tenham uma visão clara e estruturada das etapas envolvidas na elaboração do TCC, desde o cadastro do projeto até a apresentação final. Além disso, os professores de TCC também se beneficiam com ferramentas de administração da disciplina, o que possibilita uma gestão menos complexa de todo o processo de desenvolvimento do TCC.

O sistema também desempenha um papel fundamental no suporte aos professores orientadores, fornecendo uma plataforma interativa que facilita a comunicação e a troca de informações com seus orientandos. Dessa forma, os orientadores podem acompanhar de maneira mais precisa o progresso dos alunos, oferecendo observações e orientações mais direcionadas.

O sistema possui uma série de recursos que podem auxiliar os membros das bancas de TCC em suas tarefas. Além das funcionalidades, como visualização dos trabalhos em andamento, realização de avaliações e inserção de comentários, o sistema também oferece diversas outras ferramentas complementares. Por exemplo, os membros das bancas podem acessar as apresentações agendadas, bem como materiais de referência relevantes. Essa variedade de recursos pode auxiliar os membros das bancas em sua preparação para as defesas, possivelmente resultando em um processo mais consistente.

A disponibilidade de informações abrangentes sobre todo o processo do Trabalho de Conclusão de Curso (TCC) é um dos benefícios significativos do sistema. Os usuários têm acesso a dados atualizados e relevantes, incluindo cronogramas das etapas, materiais de referência e a possibilidade de exportar relatórios. Essa ampla base de conhecimento pode auxiliar na compreensão clara e aprofundada das fases do TCC, o

que pode fornecer um suporte valioso tanto para estudantes quanto para professores.

Além disso, o desenvolvimento desse sistema proporcionou ao autor uma valiosa oportunidade para aplicar os conhecimentos adquiridos ao longo da graduação de forma prática. Assim também, durante esse processo, foi possível explorar tecnologias atuais, como Spring Boot, Angular e TypeScript, que não foram amplamente abordadas durante o curso de Ciência da Computação. Essa experiência permitiu ao autor aperfeiçoar e praticar habilidades nessas tecnologias específicas, complementando o conhecimento prévio e ampliando a sua bagagem profissional. Como resultado, o autor se sente mais preparado de maneira abrangente para enfrentar os desafios do mercado de trabalho.

6.1 Contribuição científica da proposta deste trabalho

Tendo como base os trabalhos encontrados na literatura, citados na seção 2.1, é possível levantar algumas características, tais como: tecnologias utilizadas, o seu foco principal e especificidade quanto sua implantação. Desta forma, o sistema proposto no presente trabalho, se difere aos referenciados nessas características relatadas.

Ao que se refere a tecnologia, o presente trabalho, se baseia em tecnologias mais modernas e mais adotadas em novos projetos no mercado atual, assim, pode-se citar os trabalhos de Birck (2021), Paz et al. (2015), Freitas e Silva (2020) e Cabral (2017), no qual adotaram o PHP, que apesar de ser uma tecnologia ainda muito utilizada no mercado atualmente, possui alternativas modernas que são mais simples, mais adotadas e com mais vantagens. Segundo Agarwal (2022), algumas desvantagens do PHP, são: não é tão seguro, possui tipagem fraca, após a utilização de mais recursos causam baixo desempenho na aplicação, apresenta baixa qualidade no tratamento de erros, entre outras. Assim como PHP, foram encontrados trabalhos que também utilizaram JSP, como Siqueira (2014), que de acordo com Isvy (2012) JSP não é muito bem documentado e não existem muitas melhorias nos últimos anos.

O foco do sistema proposto refere-se à interação do orientador e seu orientando, e também nos processos realizados pelo professor supervisor de TCC e dos professores da banca de TCC, de uma forma que possa gerenciar todas as etapas referentes ao desenvolvimento até sua defesa. Sendo assim, alguns dos trabalhos referenciados na seção 2.1, demonstram um foco divergente ao do sistema proposto por este trabalho.

Os trabalhos de Alves, Urquiza e Roland (2016), Silva (2017), Paz et al. (2015) e Cabral (2017), denotam um grande foco na interação do orientador e seu orientando, abordando aspectos como a definição do tema, a seleção do professor que atuará como orientador e, em alguns casos, até mesmo o gerenciamento da evolução do documento de TCC. Além disso, o trabalho de Silva (2017) e Paz et al. (2015) objetiva as etapas existentes no processo inicial do TCC, abrangendo a fase de escolha do tema e a busca por um professor orientador.

Já o trabalho de Cabral (2017) tem como intuito a gestão do projeto de TCC, que se refere ao controle de versão do documento de TCC, mas também controle das tarefas a serem realizadas. Por fim, o trabalho de Gomes (2016), possui um maior foco nas etapas na qual se referem à banca de TCC, bem como seu agendamento, geração de relatório em cima dos agendamentos, obtenção das notas dos professores da banca, entre outros.

Quanto a especificidade da implantação, ou seja, se o sistema seria específico a alguma instituição ou não, o presente trabalho tem como base o processo de TCC do Departamento de Computação da UFES - Campus de Alegre, porém projetado para atender todas as etapas de um TCC acadêmico relativo a qualquer curso de graduação. O processo realizado no departamento citado é praticado em muitos cursos de graduação, em instituições públicas e privadas e, o recorte desse processo por base já se consiste na prática da maioria dos cursos, o que não inviabiliza as adequações feitas para serem aderentes à quase totalidade dos cursos que praticam a atividade de TCC. Desta maneira, os trabalhos de Silva (2017), Birck (2021), Freitas e Silva (2020), Ferreira (2015), Siqueira (2014), Henschel (2020) e Gomes (2016), têm como objetivo o desenvolvimento de um sistema que será implantado em uma instituição específica, porém em alguns destes trabalhos, apresentam possíveis futuras modificações que adequariam o sistema para atender outras instituições.

6.2 Trabalhos Futuros

No projeto em questão, foram identificados diversos requisitos funcionais que foram discutidos na seção 3.2.2. Devido à complexidade de implementação destes requisitos e restrição de tempo, alguns outros requisitos funcionais foram considerados adiados, a fim de serem atendidos em melhorias futuras, que são:

- Emitir declaração de participação: O sistema irá emitir a declaração de partici-

pação após todos os documentos serem assinados por todos os membros da banca.

- Emitir fichas de avaliação: O sistema deve permitir ao professor de TCC emitir as fichas de avaliação do orientador e dos examinadores.
- Emitir carta de encaminhamento à banca examinadora: O sistema deve permitir ao professor de TCC e orientador emitir a carta de encaminhamento à banca examinadora.
- Emitir termo de compromisso de desenvolvimento de TCC: O sistema deve permitir ao professor de TCC e orientador emitir o termo de compromisso de desenvolvimento de TCC.
- Emitir termo de inserção de co-orientador de TCC: O sistema deve permitir ao orientador emitir o termo de inserção de co-orientador de TCC.
- Emitir certificado de orientador e examinador de monografia: O sistema deve permitir ao professor de TCC emitir o certificado de orientador e examinador de monografia.
- Emitir certificado de aprovação da monografia: O sistema deve permitir ao professor de TCC emitir o certificado de aprovação da monografia.

Todos esses requisitos dizem respeito à emissão de arquivos, seja um arquivo parametrizado ou não. No entanto, a falta da implementação desses requisitos foi tratada por meio da funcionalidade de anexação de documentos à parte em um projeto de TCC. Dessa forma, para atender a esses requisitos, o usuário responsável deverá realizar o *upload* dos documentos correspondentes em seus respectivos projetos de TCC.

Além disso, como parte dos planos futuros, há a intenção de implantar o sistema no Departamento de Computação da UFES - Campus de Alegre. Inicialmente, ele será disponibilizado para utilização exclusiva nesse departamento, com o objetivo de posteriormente expandir sua aplicação para outros departamentos da UFES - Campus de Alegre e, potencialmente, também em outras instituições.

REFERÊNCIAS

AGARWAL, P. *Advantages and Disadvantages of PHP*. 2022. Disponível em: <https://www.geeksforgeeks.org/advantages-and-disadvantages-of-php/>. Acesso em: 10 jul. 2022. Citado na página 108.

ALBERTIN, A. L.; ALBERTIN, R. M. d. M. Benefícios do uso de tecnologia de informação para o desempenho empresarial. *Revista de Administração Pública*, v. 42, n. 2, p. 275–302, 2008. Disponível em: <https://doi.org/10.1590/S0034-76122008000200004>. Citado na página 18.

ALVES, L. C.; URQUIZA, M. F.; ROLAND, C. E. de F. Gestor de tcc: desenvolvimento de sistema para gestão de trabalhos acadêmicos. *Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica*, v. 7, n. 1, 2016. Citado 2 vezes nas páginas 22 e 109.

BIRCK, R. H. Sistema web para gestão de trabalhos de conclusão de curso. 2021. Citado 3 vezes nas páginas 23, 108 e 109.

BOAGLIO, F. *Spring Boot: Acelere o desenvolvimento de microsserviços*. [S.l.]: Casa do Código, 2017. Citado na página 26.

CABRAL, L. L. Tcc online: sistema de gerenciamento de trabalhos de conclusão de curso. Universidade Federal Fluminense, 2017. Citado 3 vezes nas páginas 25, 108 e 109.

CLEMENTE, F. A. S.; SANTOS, L. C. B. Desmistificando o trabalho de conclusão de curso (tcc) da graduação. *Revista Educação-UNG-Ser*, v. 10, n. 2, p. 23–39, 2015. Citado na página 19.

DBEAVER. *About DBeaver*. 2023. Disponível em: <https://dbeaver.io>. Acesso em: 21 jul. 2023. Citado na página 42.

DEVMEDIA. *Spring Boot: simplificando o Spring*. 2015. Disponível em: <https://www.devmedia.com.br/spring-boot-simplificando-o-spring/31979>. Acesso em: 10 jun. 2022. Citado na página 26.

EVANS, E.; SZPOTON, R. *Domain-driven design*. [S.l.]: Helion, 2015. Citado na página 55.

FERREIRA, É. D. *Desenvolvimento de um sistema para o gerenciamento do processo de trabalho de conclusão de curso do curso de Tecnologia em Sistemas para Internet da UTFPR Câmpus Guarapuava*. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2015. Citado 2 vezes nas páginas 24 e 109.

FIELDING, R. *Representational State Transfer (REST)*. 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>. Acesso em: 14 jun. 2022. Citado na página 46.

FREITAS, F. M.; SILVA, J. H. C. d. Webtcc-plataforma para controle e acompanhamento de trabalhos finais de curso. 2020. Citado 3 vezes nas páginas 24, 108 e 109.

GAMMA, E. et al. *Design patterns: elements of reusable object-oriented software*. [S.l.]: Pearson Deutschland GmbH, 1995. Citado na página 54.

GIT. *Git–distributed-even-if-your-workflow-isnt*. 2022. Disponível em: <https://git-scm.com/about>. Acesso em: 12 jun. 2022. Citado na página 29.

GITHUB. 2022. Disponível em: <https://github.com/>. Acesso em: 26 jun. 2022. Citado na página 29.

GOMES, V. Sistema gerenciador de bancas de tcc–gbancas. *Revista do COMINE*, v. 1, n. 1, 2016. Citado 2 vezes nas páginas 25 e 109.

GOOGLE. *What is Angular?* 2022. Disponível em: <https://angular.io/guide/what-is-angular>. Acesso em: 12 jun. 2022. Citado 3 vezes nas páginas 28, 67 e 68.

GUIMARÃES, J. d. C.; SOBRINHO, F. D. d. S. Fatores facilitadores e dificultadores à construção do tcc. *Revista Brasileira de Administração Científica*, v. 11, n. 3, p. 82–99, Jul. 2020. Disponível em: <http://doi.org/10.6008/CBPC2179-684X.2020.003.0006>. Citado na página 18.

HENSCHEL, B. E. Gtfurb: Sistema para gestao de trabalho de conclusao de curso do departamento de sistemas e computac ao da furb. *Disponıvel em: http://dsc. inf. furb. br/arquivos/tccs/monografias/20181brian- edward- henschelmonografia. pdf. Acesso*, v. 5, 2020. Citado 2 vezes nas páginas 25 e 109.

HEUSER, C. A. *Projeto de banco de dados: Volume 4 da Série Livros didáticos informática UFRGS*. [S.l.]: Bookman Editora, 2009. Citado na página 42.

ISVY, M. *Spring MVC: from JSP and Tiles to Thymeleaf*. 2012. Disponível em: <https://spring.io/blog/2012/10/30/spring-mvc-from-jsp-and-tiles-to-thymeleaf>. Acesso em: 10 jul. 2022. Citado na página 108.

JONES, M.; BRADLEY, J.; SAKIMURA, N. *Json web token (jwt)*. [S.l.], 2015. Citado na página 57.

JSON. *Introdução ao JSON*. 2022. Disponível em: <https://www.json.org/json-pt.html>. Acesso em: 12 jun. 2022. Citado na página 28.

LAENDER, A. H.; FLYNN, D. J. A semantic comparison of the modelling capabilities of the er and niam models. In: SPRINGER. *International Conference on Conceptual Modeling*. [S.l.], 1993. p. 242–256. Citado na página 42.

LEITE, J. C. S. do P.; LEONARDI, M. C. Business rules as organizational policies. In: IEEE. *Proceedings Ninth International Workshop on Software Specification and Design*. [S.l.], 1998. p. 68–76. Citado na página 38.

LOPES, E. F. B. et al. A relação entre orientador e orientando no processo de produção científica. *Brazilian Journal of Development*, v. 6, n. 1, p. 3854–3868, 2020. Citado na página 19.

MACHADO, F. N. R. *Análise e Gestão de Requisitos de Software–Onde nascem os sistemas*. [S.l.]: Saraiva Educação SA, 2018. Citado na página 33.

MALHEIROS, R. C.; OLIVEIRA, V. de N. Facilidades e dificuldades na elaboração de trabalhos de conclusão de curso. *ConScientiae Saúde*, Universidade Nove de Julho, n. 3, p. 65–72, 2004. Citado na página 20.

MEDEIROS, B. C. et al. Dificuldades do processo de orientação em trabalhos de conclusão de curso (tcc): um estudo com os docentes do curso de administração de uma instituição privada de ensino superior. *Holos*, v. 5, p. 242–255, 2015. Citado na página 18.

MORAES, L. R. D. de; SILVA, R. A. F. da; ADVERS, P. F. O avanço da tecnologia da informação e seus reflexos na contabilidade. *TCC-Ciências Contábeis*, 2021. Citado na página 18.

O'GRADY, S. *The RedMonk Programming Language Rankings: June 2019*. 2019. Disponível em: <https://redmonk.com/sogrady/2019/07/18/language-rankings-6-19/>. Acesso em: 12 jun. 2022. Citado na página 29.

ORENSTEIN, D. *Application Programming Interface*. 2000. Disponível em: <https://www.computerworld.com/article/2593623/application-programming-interface.html>. Acesso em: 13 jun. 2022. Citado na página 46.

PAZ, W. J. d. et al. Sgtcc: Sistema de gerência de trabalho de conclusão de curso. Insitituto Federal de Educação, Ciência e Tecnologia de Goiás, 2015. Citado 3 vezes nas páginas 23, 108 e 109.

PEZOA, F. et al. Foundations of json schema. In: *Proceedings of the 25th international conference on World Wide Web*. [S.l.: s.n.], 2016. p. 263–273. Citado na página 49.

PRESSMAN, R. Engenharia de software, 7a. *Edição. Rio de Janeiro: McGraw*, 2011. Citado 3 vezes nas páginas 33, 34 e 45.

RUBIN, J.; CHISNELL, D. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. [S.l.]: John Wiley & Sons, 2011. Citado na página 103.

SALVADORI, I. L. et al. Desenvolvimento de web apis restful semânticas baseadas em json. 2015. Citado na página 46.

SCHWABER, K.; SUTHERLAND, J. La guía de scrum. *Scrumguides. Org*, v. 1, p. 21, 2013. Citado 2 vezes nas páginas 30 e 31.

SHAFRANOVICH, Y. *Common format and MIME type for comma-separated values (CSV) files*. [S.l.], 2005. Citado na página 79.

SILVA, M. R. R. d. Projeto e desenvolvimento de um sistema para gerenciamento de trabalhos de conclusão de curso. Universidade Federal de Uberlândia, 2017. Citado 2 vezes nas páginas 23 e 109.

SIQUEIRA, R. C. d. Sistema de controle de submissão de propostas de trabalho de conclusão de curso. Universidade Tecnológica Federal do Paraná, 2014. Citado 3 vezes nas páginas 24, 108 e 109.

SOMMERVILLE, I. Software engineering 10th edition. *ISBN-10*, v. 137035152, p. 18, 2015. Citado 9 vezes nas páginas 33, 34, 37, 39, 41, 45, 98, 99 e 102.

SPRING. *Spring Framework*. 2022. Disponível em: <https://spring.io/projects/spring-framework>. Acesso em: 10 jun. 2022. Citado 4 vezes nas páginas 26, 27, 52 e 57.

STACKOVERFLOW. *Developer Survey Results*. 2018. Disponível em: <https://insights.stackoverflow.com/survey/2018#technology-most-loved-dreaded-and-wanted-languages>. Acesso em: 12 jun. 2022. Citado na página 29.

TECHTARGET. *What is an API endpoint?* 2021. Disponível em: <https://www.techtarget.com/searchapparchitecture/definition/API-endpoint>. Acesso em: 20 mai. 2023. Citado na página 53.

TOTVS, E. *Arquitetura REST: Saiba o que é e seus diferenciais*. 2020. Disponível em: <https://www.totvs.com/blog/developers/rest/>. Acesso em: 15 mai. 2022. Citado na página 45.

TULEAP. *Understanding Agile Scrum in 10 minutes*. 2022. Disponível em: <https://www.tuleap.org/agile/agile-scrum-in-10-minutes>. Acesso em: 20 jun. 2022. Citado na página 31.

UFES. *Trabalho de Conclusão de Curso: Regulamento e Normas para Elaboração*. 2012. Disponível em: <https://computacao.alegre.ufes.br/sites/computacao.alegre.ufes.br/files/field/anexo/trabalho_de_conclusao_de_curso_-_regulamento_e_normas_para_elaboracao.pdf#overlay-context=regulamentos>. Acesso em: 20 abr. 2022. Citado 2 vezes nas páginas 19 e 30.

UOL. *Conheça o banco de dados PostgreSQL*. 2020. Disponível em: <https://meunegocio.uol.com.br/blog/conheca-o-banco-de-dados-postgresql/>. Acesso em: 12 jun. 2022. Citado na página 29.

APÊNDICE A – DESCRIÇÃO DAS TABELAS DO BANCO DE DADOS

Tabela 6: Descrição de dados da entidade agenda_apresentacao_restricao

agenda_apresentacao_restricao			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
data_final	Data/Hora		Data limite máximo
data_inicial	Data/Hora		Data limite mínimo
id_agenda_apresentacao	Inteiro	FK	Identificador da agenda de apresentação

Fonte: Do autor

Tabela 7: Descrição de dados da entidade aluno

aluno			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
data_nascimento	Data		Data de nascimento
email	Texto		E-mail
matricula	Texto		Matrícula
nome	Texto		Nome
sexo	Texto		Sexo
id_area_tcc	Inteiro	FK	Identificador da área de TCC
id_usuario	Inteiro	FK	Identificador do usuário

Fonte: Do autor

Tabela 8: Descrição de dados da entidade anexo_geral

anexo_geral			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
conteudo	Blob		Conteúdo do arquivo
media_type	Texto		Tipo de mídia
nome_arquivo	Texto		Nome do arquivo
avaliacao	Booleano		Documento da etapa de avaliação
data_exclusao	Data/Hora		Data da exclusão
descricao	Texto		Descrição
id_modelo_documento	Inteiro	FK	Identificador do modelo de documento
id_professor	Inteiro	FK	Identificador do professor
id_projeto_tcc	Inteiro	FK	Identificador do projeto de TCC

Fonte: Do autor

Tabela 9: Descrição de dados da entidade apresentacao

apresentacao			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
data_final	Data/Hora		Data limite máximo
data_inicial	Data/Hora		Data limite mínimo
tipo_tcc	Texto		Tipo do TCC
id_agenda_apresentacao	Inteiro	FK	Identificador da agenda de apresentação
id_projeto_tcc	Inteiro	FK	Identificador do projeto de TCC

Fonte: Do autor

Tabela 10: Descrição de dados da entidade area_tcc

area_tcc			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
curso	Texto		Curso
faculdade	Texto		Faculdade

Fonte: Do autor

Tabela 11: Descrição de dados da entidade membro_banca

membro_banca			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
data_confirmacao	Data/Hora		Data da confirmação
data_solicitacao	Data/Hora		Data da solicitação
tipo_tcc	Texto		Tipo do TCC
id_professor	Inteiro	FK	Identificador do professor
id_projeto_tcc	Inteiro	FK	Identificador do projeto de TCC

Fonte: Do autor

Tabela 12: Descrição de dados da entidade modelo_aspecto_avaliacao

modelo_aspecto_avaliacao			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
descricao	Texto		Descrição
peso	Inteiro		Peso da nota
id_modelo_item_avaliacao	Inteiro	FK	Identificador do item do modelo de avaliação

Fonte: Do autor

Tabela 13: Descrição de dados da entidade modelo_avaliacao

modelo_avaliacao			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
data_exclusao	Data/Hora		Data da exclusão
nota_maxima	Real		Nota máxima
nota_media	Real		Nota média para aprovação
id_area_tcc	Inteiro	FK	Identificador da área de TCC

Fonte: Do autor

Tabela 14: Descrição de dados da entidade modelo_documento_tipo_tcc

modelo_documento_tipo_tcc			
Atributo	Tipo de dado	(PK/FK)	Descrição
id_modelo_tcc	Inteiro	PK, FK	Identificador do modelo de documento
tipo_tcc	Texto	PK	Tipo do TCC

Fonte: Do autor

Tabela 15: Descrição de dados da entidade modelo_documento

modelo_documento			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
conteudo	Blob		Conteúdo do arquivo
media_type	Texto		Tipo de mídia
nome_arquivo	Texto		Nome do arquivo
descricao	Texto		Descrição
nome	Texto		Nome do modelo

Fonte: Do autor

Tabela 16: Descrição da entidade modelo_item_avaliacao_tipo_professor

modelo_item_avaliacao_tipo_professor			
Atributo	Tipo de dado	(PK/FK)	Descrição
id_modelo_item_avaliacao	Inteiro	PK, FK	Identificador do item do modelo de avaliação
tipo_professor	Texto	PK	Tipo do professor

Fonte: Do autor

Tabela 17: Descrição de dados da entidade modelo_item_avaliacao_tipo_tcc

modelo_item_avaliacao_tipo_tcc			
Atributo	Tipo de dado	(PK/FK)	Descrição
id_modelo_item_avaliacao	Inteiro	PK, FK	Identificador do item do modelo de avaliação
tipo_tcc	Texto	PK	Tipo do TCC

Fonte: Do autor

Tabela 18: Descrição de dados da entidade modelo_item_avaliacao

modelo_item_avaliacao			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
data_exclusao	Data/Hora		Data da exclusão
id_modelo_avaliacao	Inteiro	FK	Identificador do modelo de avaliação

Fonte: Do autor

Tabela 19: Descrição de dados da entidade professor_area_tcc

professor_area_tcc			
Atributo	Tipo de dado	(PK/FK)	Descrição
id_professor	Inteiro	FK	Identificador do professor
id_area_tcc	Inteiro	FK	Identificador da área de TCC

Fonte: Do autor

Tabela 20: Descrição de dados da entidade professor_disponibilidade

professor_disponibilidade			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
data_final	Data/Hora		Data limite máximo
data_inicial	Data/Hora		Data limite mínimo
ano	Inteiro		Ano
periodo	Inteiro		Período
id_professor	Inteiro	FK	Identificador do professor

Fonte: Do autor

Tabela 21: Descrição de dados da entidade professor

professor			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
data_nascimento	Data		Data de nascimento
email	Texto		E-mail
matricula	Texto		Matrícula
nome	Texto		Nome
sexo	Texto		Sexo
supervisor_tcc	Booleano		Supervisor de TCC
id_usuario	Inteiro	FK	Identificador do usuário

Fonte: Do autor

Tabela 22: Descrição de dados da entidade projeto_tcc_aluno

projeto_tcc_aluno			
Atributo	Tipo de dado	(PK/FK)	Descrição
id_projeto_tcc	Inteiro	FK	Identificador do projeto de TCC
id_aluno	Inteiro	FK	Identificador do aluno

Fonte: Do autor

Tabela 23: Descrição de dados da entidade projeto_tcc_aspecto_avaliacao

projeto_tcc_aspecto_avaliacao			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
descricao	Texto		Descrição
id_modelo_aspecto_avaliacao	Inteiro		Identificador do aspecto do modelo de avaliação
peso	Inteiro		Peso da nota
valor	Real		Valor do aspecto
id_projeto_tcc_avaliacao	Inteiro	FK	Identificador da avaliação de um projeto de TCC

Fonte: Do autor

Tabela 24: Descrição de dados da entidade projeto_tcc_avaliacao

projeto_tcc_avaliacao			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
tipo_professor	Texto		Tipo do professor
tipo_tcc	Texto		Tipo do TCC
id_modelo_item_avaliacao	Inteiro	FK	Identificador do item do modelo de avaliação
id_professor	Inteiro	FK	Identificador do professor
id_projeto_tcc	Inteiro	FK	Identificador do projeto de TCC

Fonte: Do autor

Tabela 25: Descrição de dados da entidade projeto_tcc_doc_visualizado_por

projeto_tcc_doc_visualizado_por			
Atributo	Tipo de dado	(PK/FK)	Descrição
id_projeto_tcc	Inteiro	FK	Identificador do projeto de TCC
id_user	Inteiro		Identificador do usuário

Fonte: Do autor

Tabela 26: Descrição de dados da entidade projeto_tcc_nota

projeto_tcc_nota			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
data_lancamento	Data/Hora		Data de lançamento da nota final
nota_final	Real		Valor da nota final
nota_maxima	Real		Nota máxima
nota_media	Real		Nota média para aprovação
tipo_tcc	Texto		Tipo do TCC
id_projeto_tcc	Inteiro	FK	Identificador do projeto de TCC

Fonte: Do autor

Tabela 27: Descrição de dados da entidade projeto_tcc_situacao

projeto_tcc_situacao			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
data_conclusao	Data/Hora		Data de conclusão
motivo	Texto		Motivo da situação
situacao_tcc	Texto		Situação
tipo_tcc	Texto		Tipo do TCC
id_projeto_tcc	Inteiro	FK	Identificador do projeto de TCC

Fonte: Do autor

Tabela 28: Descrição de dados da entidade projeto_tcc

projeto_tcc			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
ano_periodo	Texto		Ano/Período
tema	Texto		Tema do TCC
id_area_tcc	Inteiro	FK	Identificador da área de TCC
id_professor_orientador	Inteiro	FK	Identificador do prof. orientador
id_professor_supervisor	Inteiro	FK	Identificador do prof. supervisor
id_situacao_atual	Inteiro	FK	Identificador da situação atual

Fonte: Do autor

Tabela 29: Descrição de dados da entidade role

role			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Texto	PK	Identificador
admin	Booleano		Permissão de administrador
aluno	Booleano		Permissão de aluno
professor	Booleano		Permissão de professor
supervisor	Booleano		Permissão de supervisor

Fonte: Do autor

Tabela 30: Descrição de dados da entidade user_role

user_role			
Atributo	Tipo de dado	(PK/FK)	Descrição
id_user	Inteiro	FK	Identificador do usuário
id_role	Texto	FK	Identificador da permissão

Fonte: Do autor

Tabela 31: Descrição de dados da entidade user

user			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
enabled	Booleano		Habilitado
name	Texto		Nome do usuário
password	Texto		Senha do usuário
type	Texto		Tipo do usuário
user_name	Texto		Nome de login para acessar

Fonte: Do autor

Tabela 32: Descrição de dados da entidade versao_tcc_observacao

versao_tcc_observacao			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
conteudo	Blob		Conteúdo do arquivo
media_type	Texto		Tipo de mídia
nome_arquivo	Texto		Nome do arquivo
avaliacao	Booleano		Documento da etapa de avaliação
observacao	Texto		Observação
id_versao_tcc	Inteiro	FK	Identificador da versão do TCC

Fonte: Do autor

Tabela 33: Descrição de dados da entidade versao_tcc

versao_tcc			
Atributo	Tipo de dado	(PK/FK)	Descrição
id	Inteiro	PK	Identificador
data_inclusao	Data/Hora		Data da inclusão
data_ultima_alteracao	Data/Hora		Data da última alteração
id_user_inclusao	Inteiro		ID Usuário - Inclusão
id_user_ultima_alteracao	Inteiro		ID Usuário - Última alteração
conteudo	Blob		Conteúdo do arquivo
media_type	Texto		Tipo de mídia
nome_arquivo	Texto		Nome do arquivo
observacao	Texto		Observação
versao	Inteiro		Versão do documento
id_aluno	Inteiro	FK	Identificador do aluno
id_professor_orientador	Inteiro	FK	Identificador do prof. orientador
id_projeto_tcc	Inteiro	FK	Identificador do projeto de TCC

Fonte: Do autor

ANEXO A – TERMO DE COMPROMISSO DE DESENVOLVIMENTO DE TCC

UFES Centro de Ciências Agrárias

dcomp Departamento de Computação

TERMO DE COMPROMISSO DE DESENVOLVIMENTO DE TCC

Firmam o compromisso de desenvolvimento do TCC, o discente regularmente matriculado no Curso de <indicar o curso>, _____, matrícula nº._____ na condição de orientando, e o Professor _____ do Departamento de Computação, da Universidade Federal do Espírito Santo, na condição de orientador na área _____.

Orientando e orientador declaram estarem cientes das normas para elaboração do TCC, descritas no Manual de TCC do Curso de <indicar o curso>.

Alegre , ES, ____ de _____ de _____.

_____ _____

Nome do Orientando
Matrícula

Nome do Orientador
Matrícula