UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

RAUL FERREIRA DA ROCHA

ESTRATÉGIA DE NOTIFICAÇÕES NO SISTEMA DE GESTÃO DE TCC DO CURSO DE SISTEMAS PARA INTERNET: DEFINIÇÃO E IMPLEMENTAÇÃO

GUARAPUAVA

2025

RAUL FERREIRA DA ROCHA

ESTRATÉGIA DE NOTIFICAÇÕES NO SISTEMA DE GESTÃO DE TCC DO CURSO DE SISTEMAS PARA INTERNET: DEFINIÇÃO E IMPLEMENTAÇÃO

Notification Strategy in the Thesis Management System of the Internet Systems Course: Definition and Implementation

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador : Profº. Drº. Diego Marczal

Coorientador: Profª. Drª. Renata Luiza Stange

GUARAPUAVA

2025

RAUL FERREIRA DA ROCHA

ESTRATÉGIA DE NOTIFICAÇÕES NO SISTEMA DE GESTÃO DE TCC DO CURSO DE SISTEMAS PARA INTERNET: DEFINIÇÃO E IMPLEMENTAÇÃO

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 01/Dezembro/2025

Diego Marczal
Doutor
Universidade Tecnológica Federal do Paraná

Andres Jessé Porfirio
Doutor
Universidade Tecnológica Federal do Paraná

Dênis Lucas Silva
Mestre
Universidade Tecnológica Federal do Paraná

GUARAPUAVA
2025

RESUMO

O Sistema de Gerenciamento de Trabalhos de Conclusão de Curso (SGTCC) desempenha papel central na organização das etapas de entrega, avaliação e acompanhamento dos TCCs do curso de Sistemas para Internet da UTFPR – Câmpus Guarapuava. Entretanto, mesmo após diversas evoluções ao longo dos anos, o sistema ainda apresentava lacunas relacionadas à comunicação entre seus usuários, especialmente no que se refere ao acompanhamento de prazos, pendências documentais e eventos como bancas e reuniões. Diante desse cenário, este trabalho teve como objetivo definir e implementar uma estratégia de notificações automáticas capaz de aprimorar o fluxo de comunicação entre coordenadores, docentes e acadêmicos sem a necessidade de intervenção manual. A metodologia empregada envolveu o levantamento e a priorização dos requisitos por meio da análise dos fluxos do sistema e da aplicação de um formulário aos usuários, utilizando o método MoSCoW para classificação. A partir disso, foram modeladas regras de notificação, templates de mensagens e fluxos de envio, integrados ao SGTCC utilizando Ruby on Rails, Active Job e Solid Queue, com arquitetura assíncrona e banco de dados dedicado exclusivamente ao módulo de notificações. O desenvolvimento contemplou desde a configuração do ambiente até a estruturação de serviços, modelos e hooks responsáveis pela geração e processamento das notificações. Como resultados, foram implementadas notificações de atualização de calendário, pendências de assinatura, prazos de envio, agendamento e confirmação de bancas, registros de reuniões, submissão de termos e outras situações críticas para o andamento do TCC. A suíte de testes também foi ampliada, garantindo maior confiabilidade às funcionalidades implementadas. Conclui-se que a estratégia de notificações desenvolvida contribuiu para melhorar a organização, a transparência e o acompanhamento das atividades do TCC, além de estabelecer bases estruturais para futuras extensões, como a integração com outros canais de comunicação.

Palavras-chave: notificações; sistemas acadêmicos; gestão de tcc; automação; ruby on rails.

ABSTRACT

The Thesis Management System (SGTCC) plays a central role in organizing the stages of submission, evaluation, and monitoring of final papers in the Internet Systems course at UTFPR – Guarapuava. However, even after several improvements over the years, the system still presented gaps related to communication among its users, particularly regarding the monitoring of deadlines, document pendencies, and academic events such as meetings and examination boards. In this context, this work aimed to define and implement an automatic notification strategy capable of improving the communication flow among coordinators, faculty member, and students without requiring manual intervention. The methodology included identifying and prioritizing requirements through the analysis of system workflows and the application of a user questionnaire, employing the MoSCoW method for classification. Based on these findings, notification rules, message templates, and delivery flows were modeled and integrated into the SGTCC using Ruby on Rails, Active Job, and Solid Queue, with an asynchronous architecture supported by a database dedicated exclusively to the notification module. The development process ranged from environment configuration to the structuring of services, models, and hooks responsible for generating and processing notifications. As a result, notifications were implemented for calendar updates, signature pendencies, submission deadlines, scheduling and confirmation of examination boards, meeting records, term submissions, and other critical situations for the progress of the final paper process. The test suite was also expanded, ensuring greater reliability of the implemented functionalities. It is concluded that the developed notification strategy contributes to improving organization, transparency, and the monitoring of TCC activities, while also establishing structural foundations for future extensions, such as the integration with additional communication channels.

Keywords: notifications; academic systems; tcc management; automation; ruby on rails.

LISTA DE ABREVIATURAS E SIGLAS

Siglas

CD	Entrega Contínua
CI	Integração Contínua
SGTCC	Sistema de Gerenciamento de Trabalho de Conclusão de Curso
SI	Sistemas para Internet
SMS	Serviço de Mensagens Curtas, do inglês *Short Message Service*
TCC	Trabalho de Conclusão de Curso
UTFPR	Universidade Tecnológica Federal do Paraná
UX	Experiência do usuário, do inglês *User Experience*

SUMÁRIO

1 INTRODUÇÃO

A realização de um Trabalho de Conclusão de Curso (TCC) constitui requisito obrigatório para a conclusão e obtenção do diploma em diversos cursos de graduação, representando o momento em que o estudante aplica os conhecimentos adquiridos ao longo do curso em uma ou mais áreas de formação (COINT, 2023). Esse processo, no entanto, envolve a produção e o gerenciamento de múltiplos documentos, o que pode dificultar sua organização e acompanhamento. Com o intuito de centralizar e facilitar esse gerenciamento, foi desenvolvido o Sistema de Gerenciamento de Trabalho de Conclusão de Curso (SGTCC), sistema que busca simplificar as etapas de entrega e avaliação dos trabalhos, além de reduzir a necessidade de documentos impressos. Assim como ocorre com diversas aplicações, o SGTCC encontra-se em constante evolução, visando atender às necessidades dos usuários e tornar mais ágeis tarefas como a assinatura de documentos, o agendamento de bancas e a avaliação de defesas. Contudo, mesmo após as atualizações implementadas por Ferreira (2015), Silva (2019), Lima (2023) e Luz (2024), o sistema ainda apresenta oportunidades de aprimoramento.

O cumprimento dos prazos durante o desenvolvimento do TCC é essencial para o bom andamento do processo, uma vez que atrasos podem gerar prejuízos tanto para os discentes quanto para os docentes. Para os estudantes, o não cumprimento de prazos pode resultar na reprovação em etapas do TCC e no consequente adiamento da conclusão do curso. Para os professores, pode comprometer o planejamento das atividades, dificultar a avaliação adequada dos trabalhos e gerar sobrecarga de tarefas. Além disso, os atrasos afetam diretamente a organização das bancas. Outro fator que impacta o gerenciamento do TCC é a ausência de assinaturas em documentos, o que impede sua validação formal e o registro adequado dos trâmites. Tanto estudantes quanto docentes podem ser responsáveis pela não entrega de documentos ou pela falta de assinaturas dentro dos prazos estabelecidos.

Diante desses desafios, esta proposta visa oferecer uma solução que mantenha todos os envolvidos no processo de gerenciamento do TCC devidamente informados sobre prazos e pendências, a fim de minimizar os prejuízos mencionados e prevenir outros que possam surgir em decorrência do não cumprimento das etapas. Para isso, podem ser adotadas diferentes estratégias de notificação, cujo objetivo é comunicar os usuários de maneira eficiente, promovendo maior engajamento. Essas estratégias podem ser implementadas por meio de notificações automáticas por e-mail, mensagens em aplicativos de comunicação ou lembretes via Serviço de Mensagens Curtas, do inglês *Short Message Service* (SMS).

O desenvolvimento de uma estratégia de notificação para o SGTCC envolve desafios técnicos e funcionais. Entre eles, destaca-se a definição dos tipos de notificação mais adequados a cada situação, de modo a equilibrar efetividade e evitar o excesso de comunicações que possam ser ignoradas pelos usuários. Além disso, é necessário garantir a integridade e a confiabilidade das informações transmitidas, assegurando que os dados sobre prazos e pendências estejam sempre atualizados e sincronizados com o sistema. Outro aspecto relevante

diz respeito à privacidade dos usuários e à conformidade com as normas de proteção de dados, especialmente no envio de informações por e-mail ou SMS. Do ponto de vista técnico, a implementação requer integração com serviços externos de envio de mensagens e o desenvolvimento de uma lógica eficiente para o disparo automático das notificações, de forma escalável e sustentável. Também devem ser considerados aspectos de usabilidade, assegurando que as mensagens sejam compreensíveis e que efetivamente contribuam para a gestão eficiente do TCC.

Dessa forma, este trabalho propôs definir e implementar uma estratégia de notificação no SGTCC, com o objetivo de garantir que todos os participantes do processo de TCC estejam cientes dos prazos e atualizações, assegurando o cumprimento adequado de todas as etapas previstas.

1.1 Objetivos

Neste trecho, são descritos os objetivos que orientaram o desenvolvimento deste trabalho, tanto em nível geral quanto específico.

1.1.1 Objetivo geral

Formular e implementar uma estratégia de notificações automáticas de prazos e pendências no sistema SGTCC, visando aprimorar o acompanhamento e a gestão do processo de TCC.

1.1.2 Objetivos específicos

- Identificar os pontos do processo de TCC que demandam notificações automáticas, com base na análise do fluxo de atividades e nas necessidades dos usuários do SGTCC.
- Definir os tipos, os canais e os momentos adequados para o envio das notificações, considerando os diferentes perfis de usuários (discentes, docentes e coordenadores).
- Elaborar a estratégia de notificações automáticas, detalhando a lógica de acionamento, a frequência, a personalização e o conteúdo das mensagens.
- Desenvolver e integrar a funcionalidade de notificações ao sistema SGTCC, utilizando os recursos disponíveis no *framework* Ruby on Rails.

1.2 Justificativa

Com o objetivo de facilitar o gerenciamento dos TCCs do curso de Sistemas para Internet (SI) do campus Guarapuava da Universidade Tecnológica Federal do Paraná (UTFPR), foi desenvolvido o SGTCC. Apesar de sua utilidade e importância para o curso, o sistema ainda apresenta oportunidades de aprimoramento, especialmente no que se refere à comunicação proativa com seus usuários.

Uma das principais lacunas apontadas pelos usuários é a ausência de notificações que os informem sobre prazos e pendências relacionadas ao processo do TCC. Essa falta de comunicação pode ocasionar atrasos na entrega de documentos e na realização de bancas de defesa, o que pode, consequentemente, gerar reprovação em etapas do TCC, retrabalho e até atrasos na colação de grau dos discentes. Outro problema recorrente diz respeito à não realização de assinaturas dentro dos prazos estabelecidos, o que pode comprometer a emissão de documentos oficiais, gerando entraves burocráticos. Soma-se a isso a inexistência de avisos sobre eventos importantes do processo, como o agendamento, a alteração ou o cancelamento de bancas, bem como sobre mudanças nos prazos, o que pode gerar confusão e desorganização entre os participantes. Além disso, os envolvidos no processo não dispõem de uma maneira simples e clara de acompanhar o tempo decorrido desde o início do TCC e o tempo restante para a conclusão de suas etapas, o que frequentemente leva ao acúmulo de tarefas próximo ao encerramento dos prazos.

Para suprir essas demandas, a formulação e implementação de uma estratégia de notificação se apresentam como uma solução eficaz, já amplamente utilizada em diversas aplicações. Desse modo, o desenvolvimento dessa funcionalidade no SGTCC visa manter os usuários constantemente informados sobre prazos, pendências e atualizações do processo, bem como sobre a criação, alteração e cancelamento de bancas de defesa. A definição da estratégia adotada foi fundamentada nas necessidades e nas experiências relatadas pelos próprios usuários, além de considerar práticas consolidadas em outras aplicações que enfrentam desafios semelhantes.

2 MATERIAIS E MÉTODOS

Neste capítulo estão descritos os materiais, ferramentas e métodos utilizados para formular e implementar uma estratégia de notificações para o SGTCC.

2.1 Materiais

Dentre os materiais descritos a seguir, estão incluídas ferramentas e tecnologias utilizadas para o planejamento e desenvolvimento:

- **Ruby on Rails**: Já utilizado para o desenvolvimento do SGTCC, o *Ruby on Rails* é um *framework full-stack* de código aberto para a linguagem *Ruby*. Ele facilita a criação de aplicações web robustas e organizadas, adotando convenções que reduzem a necessidade de configurações manuais, promovendo o uso de boas práticas de desenvolvimento e acelerando o processo de codificação (RAILS, 2025).

- **Git e GitHub**: O *Git* é um sistema de controle de versão distribuído que permite acompanhar e gerenciar mudanças no código-fonte ao longo do tempo, facilitando a colaboração entre desenvolvedores e garantindo a integridade do projeto (GIT, 2025). Já o *GitHub* é uma plataforma baseada na nuvem que utiliza o Git para hospedar e versionar repositórios, permitindo colaboração em tempo real, integração contínua e automação de fluxos de trabalho (GITHUB, 2025). A escolha dessas ferramentas deve-se ao suporte robusto a versionamento, ao histórico detalhado de alterações e à integração nativa com pipelines de automação, essenciais para acompanhar as evoluções no módulo de notificações, além da utilização prévia dessas tecnologias para o gerenciamento do SGTCC.

- **Figma**: Ferramenta de design de interface baseada na web que permite colaboração em tempo real entre designers e desenvolvedores. Com recursos de prototipagem, componentes reutilizáveis e compartilhamento facilitado, o Figma é utilizado para a criação e validação visual de interfaces (FIGMA, 2025). Foi escolhido por permitir a rápida criação de protótipos das telas relacionadas às notificações.

- **Docker**: Plataforma de virtualização leve que permite empacotar aplicações e suas dependências em *containers*, garantindo consistência entre ambientes de desenvolvimento, testes e produção. Facilita a portabilidade, escalabilidade e automação de processos de implantação (DOCKER, 2025). A utilização do Docker para padronizar o ambiente de desenvolvimento já estava presente desde os estágios anteriores do desenvolvimento, garantindo reprodutibilidade nos testes e evitando inconsistências entre diferentes máquinas.

- **SQLite**: Banco de dados relacional leve, simples e amplamente utilizado em aplicações que necessitam de um armazenamento local rápido e sem a complexidade de um servidor dedicado. Por ser baseado em arquivos, o SQLite apresenta baixo custo de manutenção e excelente desempenho para operações de leitura e escrita em cenários de menor escala. No contexto deste trabalho, foi utilizado como banco de dados secundário especificamente para o módulo de notificações, armazenando as notificações geradas e a fila de jobs do ActiveJob. Sua escolha se deve à necessidade de isolar o armazenamento e o processamento de notificações do banco principal do SGTCC, garantindo maior independência, menor impacto no desempenho geral do sistema e simplificando a configuração do ambiente de testes.
- **ClickUp**: Plataforma de gerenciamento de projetos que centraliza tarefas, cronogramas, documentos e comunicação da equipe em um único ambiente. Sua flexibilidade e ampla capacidade de personalização ajudam a estruturar e acompanhar as etapas do projeto, facilitando a organização, o cumprimento de prazos e a produtividade da equipe envolvida (CLICKUP, 2025). Foi utilizado para planejar, acompanhar e documentar o progresso do desenvolvimento das funcionalidades, garantindo maior consistência e controle sobre as diversas etapas do desenvolvimento.

2.2 Métodos

Para formular e implementar um sistema de notificações no SGTCC são adotados os seguintes passos:

2.2.1 Levantamento e Priorização de Requisitos

Para formular a estratégia de notificações do SGTCC, inicialmente identifica-se quais situações do sistema demandam comunicação aos usuários. Esse processo tem início com uma análise das funcionalidades existentes, listando-se todas as ações que potencialmente podem gerar notificações, de forma a construir um pré-levantamento das situações notificáveis.

Com o objetivo de validar esse levantamento e complementar possíveis lacunas, aplica-se um formulário aos usuários do SGTCC, incluindo docentes e discentes do curso de Tecnologia em Sistemas para Internet da UTFPR – Câmpus Guarapuava. As respostas obtidas permitem identificar novas demandas de notificação, além de contribuir para a priorização das já mapeadas.

A organização e priorização dos requisitos são realizadas utilizando o Método MoSCoW, técnica amplamente empregada para classificação de itens em projetos de software. O método categoriza cada notificação em quatro grupos — *Must Have*, *Should Have*, *Could Have* e *Won't*

Have — permitindo definir sua importância relativa e orientar a execução do desenvolvimento conforme os recursos disponíveis e o impacto esperado no sistema.

Originalmente desenvolvido por Dai Clegg, o Método MoSCoW destaca-se por sua simplicidade e aplicabilidade em diferentes contextos, auxiliando na tomada de decisões e proporcionando uma visão estruturada das prioridades do projeto. A Figura 1 apresenta um resumo visual da técnica (SEBRAE, 2022).

Figura 1 – Método MoSCoW
Fonte: Sebrae (2022).

2.2.2 Processo de Desenvolvimento

Após levantar e priorizar os requisitos de notificação, inicia-se o processo de desenvolvimento. Essa etapa começa com a subdivisão dos requisitos em tarefas menores e individuais a serem implementadas. A gestão durante todo o processo de desenvolvimento é realizada por meio do *ClickUp*, permitindo organizar as tarefas por ordem de importância e por seu estado atual, facilitando o acompanhamento do desenvolvimento.

O fluxo de trabalho Git é organizado por meio do *GitFlow*, uma metodologia que divide as ramificações (branches) do código em diferentes tipos, de acordo com a função desempenhada por cada uma, melhorando a organização e facilitando a colaboração durante o processo de desenvolvimento. As divisões de *branches* são as seguintes (ATLASSIAN, 2025):

- **Main**: Uma das duas ramificações principais, armazena o histórico do lançamento oficial, contendo um histórico simplificado das alterações do projeto.
- **Develop**: A outra ramificação principal, serve como uma ramificação para a integração de recursos, mantendo um histórico completo das alterações do projeto.

- **Feature**: Ramificações temporárias para a implementação de recursos, criadas a partir da última versão da *branch develop*, sendo integradas novamente à *branch develop* depois de concluídas.
- **Release**: Ramificação intermediária entre a principal e a de desenvolvimento, criada quando há recursos suficientes para um lançamento, contendo apenas atualizações de segurança ou relacionadas ao lançamento. Ao fim, integra-se à *branch main*.
- **Hotfix**: Ramificação utilizada para correções rápidas de lançamentos em produção, criada a partir da *branch main* e mesclada a ela após a correção.

Visando facilitar o processo de implementação após o desenvolvimento, utiliza-se o conceito de Integração Contínua (CI)/Entrega Contínua (CD), uma automatização do processo de desenvolvimento, desde a codificação até a implementação, agilizando o lançamento de novos recursos e correções, tornando o produto mais responsivo às necessidades dos usuários. CI refere-se à automatização de testes realizados ao mesclar *branches* na *branch* principal, evitando conflitos de código, identificando erros ou problemas de segurança e verificando a qualidade do código. Já o CD prepara e testa o código para ser implementado em produção, automatizando o provisionamento da infraestrutura e o processo de lançamento das aplicações, empacotando todos os elementos necessários para implantação em qualquer ambiente (GitLab Inc., 2025).

Com base nos requisitos priorizados, são elaboradas Histórias de Usuário que descrevem as necessidades do ponto de vista dos diferentes perfis do sistema, como discentes, docentes e coordenadores. Essas histórias servem como guia para o desenvolvimento das funcionalidades de notificação, garantindo que o sistema atenda às expectativas e aos fluxos reais de uso do SGTCC.

3 ANÁLISE E PROJETO

Este capítulo descreve as etapas de análise e projeto do sistema de notificações proposto para o SGTCC, abordando a descrição do sistema, o levantamento e a análise dos requisitos, bem como a definição da arquitetura de notificação. O objetivo principal é garantir que as funcionalidades implementadas atendam efetivamente às necessidades dos usuários, promovendo uma comunicação eficiente sobre prazos e eventos relevantes ao processo de TCC.

3.1 Descrição do SGTCC

No curso de SI, o TCC é dividido em três partes principais, cada uma com entregas distintas e bancas de defesa: elas são Proposta, Projeto e Monografia. Além disso, o processo é dividido em duas unidades curriculares, ocupando dois semestres letivos. Nesse contexto de tempo limitado, o cumprimento de prazos e a boa gestão do processo tornam-se importantes (COINT, 2023).

Com base nisso, o SGTCC teve seu desenvolvimento iniciado em 2015, com o objetivo de tornar digital a gestão das atividades referentes ao TCC do Curso de SI do Campus Guarapuava da UTFPR, visando centralizar as informações e regulamentos, além de simplificar os processos necessários (FERREIRA, 2015).

O projeto teve continuidade em 2019, sendo reestruturado e contando com a melhoria nos módulos do sistema, como a criação de tipos de usuários, o *upload* de documentos relacionados ao TCC, o cadastro de reuniões realizadas e o agendamento de defesas. Além disso, ocorreu a implementação de assinaturas eletrônicas, visando eliminar o uso de papel e tornar toda a gestão digital. (SILVA, 2019).

Durante o segundo semestre de 2023, o sistema teve outras contribuições realizadas pelos alunos da disciplina Desenvolvimento para Web 5 do curso de SI, incluindo a criação de novas funcionalidades, a correção de *bugs* e atualizações nas bibliotecas do projeto.

Contudo, ainda em 2023, outras melhorias foram realizadas, principalmente visando a otimização das telas do sistema. Foram aplicadas técnicas de Experiência do usuário, do inglês *User Experience* (UX) design, com foco na estética e funcionalidade, reorganizando e tornando a usabilidade mais agradável. Muitas alterações foram feitas com base em questionários realizados com os usuários do sistema. Após essas alterações, o resultado de um novo questionário mostrou que houve uma grande melhoria na usabilidade do sistema (LIMA, 2023).

O desenvolvimento do SGTCC continua em andamento. Atualmente está em curso um projeto para a atualização do *Framework Rails* e de outras dependências do sistema, além de uma adequação no código para garantir a continuidade do sistema, corrigindo possíveis vulnerabilidades e assegurando que possa continuar a evoluir (LUZ, 2024).

As áreas implementadas atualmente são as seguintes:

- Área pública: Pode ser acessada por qualquer pessoa, essa área conta com informações gerais sobre o TCC e as atividades de TCC do período corrente.
- Área do membro externo: Disponível para instituições externas e convidados, essa área conta com acesso às bancas e documentos, relacionados aos trabalhos aos quais o membro externo faz parte.
- Área acadêmica: Tendo acesso às informações sobre bancas e acesso aos documentos e atividades dos discentes.
- Área do orientador: Área onde o docente pode monitorar e registrar todas as atividades relacionadas aos trabalhos que o mesmo orienta, além de informações sobre as bancas que irá participar.
- Área do Professor de TCC 1: O professor responsável pela disciplina de TCC 1, nesta área, verificar os prazos das entregas, agendar bancas de proposta e projeto, acompanhar as entregas feitas e ter acessos as informações de todos os alunos matriculados na disciplina.
- Área do Professor responsável pelo TCC: Responsável por gerenciar todos os processos relacionados as disciplinas de TCC 1 e TCC 2, podendo fazer cadastro de todos os outros tipos de usuários, definindo calendários, cadastrando atividades e agendando bancas de todos os tipos.

Estando em constante evolução desde seu início, o SGTCC recebeu diversas mudanças para tentar cumprir melhor seu objetivo: gerenciar os trabalhos de TCC. Uma parte dessa evolução foi a implementação de novas funcionalidades, buscando atender melhor às necessidades dos usuários. Dentre os problemas ainda existentes no SGTCC, uma lacuna muito aparente para os usuários é a falta de avisos e lembretes sobre os prazos e sobre afazeres necessários para o processo de TCC.

Deste ponto de vista e pela tamanha importância do cumprimento de prazos no processo de TCC, surge a necessidade de informar todos os envolvidos quanto aos prazos, avisos importantes, bem como o tempo restante para a conclusão do TCC. Com o objetivo de suprir essa necessidade e minimizar os prejuízos causados pela perda de prazos e pela ausência de assinatura de documentos, faz-se necessário o planejamento, a escolha e a implementação de estratégias de notificação.

3.2 Levantamento dos requisitos

O levantamento de requisitos foi conduzido a partir de duas abordagens complementares. Primeiramente, realizou-se a análise dos fluxos operacionais já existentes no sistema

SGTCC, com o objetivo de mapear as etapas do processo, identificar pontos críticos e compreender a sequência de ações atualmente exigidas de discentes, docentes e responsáveis pelas bancas. Os resultados provenientes da análise do fluxo interno do TCC foram organizados na Tabela 1, servindo como base para a definição das funcionalidades e melhorias propostas.

Tabela 1 – Levantamento de requisitos a partir da análise de fluxos do SGTCC.

Requisito	Descrição
Cadastro e alteração de calendário	Notificar todos os usuários cadastrados no calendário atual após um período determinado da criação ou alteração do mesmo.
Assinatura de documentos pendentes	Enviar aviso imediato a todos que deveriam assinar um documento específico e ainda não realizaram a assinatura.
Confirmação de assinatura de documentos	Notificar imediatamente os demais envolvidos quando um dos participantes realizar uma assinatura.
Prazos de envio de proposta/projeto/monografia	Enviar lembretes sobre prazos de envio para orientadores e alunos, com possibilidade de definir quantidade e intervalo dos avisos, diferenciando perfis de usuários.
Envio de proposta/projeto/monografia	Avisar imediatamente orientadores, professores responsáveis e demais envolvidos sobre o envio desses documentos.
Agendamento e alteração de banca (avaliadores)	Enviar notificação aos avaliadores após um tempo definido da última alteração no agendamento, solicitando confirmação de disponibilidade.
Agendamento e alteração de banca (aluno e curso)	Notificar o aluno que irá defender e todos do curso após confirmação dos avaliadores, considerando a última alteração.
Registros dos apontamentos da banca	Avisar imediatamente o acadêmico quando houver registros de apontamentos pela banca.
Avisos sobre a evolução do tempo do TCC	Notificar o aluno ao longo do desenvolvimento do TCC, seguindo uma linha do tempo pré-definida.
Ciência na reunião	Notificar o acadêmico para ciência dos registros feitos pelo orientador em reunião, solicitando concordância.

Após a identificação de requisitos a partir da análise de fluxos do SGTCC, aplicou-se um formulário eletrônico elaborado no Google Forms [1], intitulado “Prioridade e Preferência de Notificações – Sistema SGTCC”. O instrumento utilizado para a coleta de informações foi direcionado aos estudantes e professores da UTFPR – Campus Guarapuava envolvidos no processo de TCC, a fim de compreender quais eventos do sistema demandam notificações, qual a prioridade de cada um deles e por quais meios de comunicação os usuários preferem ser informados. Essas informações são essenciais para o desenvolvimento de um sistema de notificações eficaz, personalizado e alinhado às necessidades reais dos seus diferentes tipos de usuários.

A seguir, apresentam-se de forma resumida as principais questões e suas respectivas finalidades:

[1] Disponível em https://forms.gle/5eHwmktweuRHNt1i8

- **Identificação do papel do usuário:** busca determinar se quem está respondendo é acadêmico, professor ou outro perfil. Essa informação é essencial para compreender as diferenças de necessidades entre os grupos e personalizar as notificações de acordo com cada função.
- **Perfis utilizados no sistema:** permite identificar quais papéis o usuário desempenha (avaliador, orientador, coordenador etc.), contribuindo para entender a sobreposição de funções e os diferentes contextos de uso do sistema.
- **Classificação de eventos pelo método MoSCoW:** solicita que os participantes classifiquem os eventos conforme sua prioridade (*Must, Should, Could, Won't Have*). Essa classificação auxilia na priorização dos requisitos e na definição do que deve ser implementado inicialmente.
- **Preferência de meios de notificação:** tem como objetivo levantar os canais preferidos de comunicação (e-mail, WhatsApp, Discord, notificações internas, entre outros), orientando o desenvolvimento de integrações e funcionalidades adequadas às expectativas dos usuários.
- **Sugestões de novas notificações:** oferece espaço para que os participantes proponham notificações adicionais, classificando-as segundo o método MoSCoW. Essa questão contribui para a descoberta de novos requisitos funcionais.
- **Comentários gerais e sugestões:** permite coletar feedbacks qualitativos sobre o sistema e suas funcionalidades, fornecendo subsídios para melhorias na interface, na frequência de notificações e na experiência do usuário.

3.3 Análise de Requisitos

Os resultados obtidos por meio do formulário indicaram uma maior participação dos docentes no processo de levantamento e priorização dos requisitos do sistema, correspondendo a 85% das respostas (12 participantes). Esse dado sugere um elevado nível de interesse e envolvimento por parte dos professores, possivelmente em razão de sua atuação direta nas etapas de orientação, avaliação e validação dos trabalhos de conclusão de curso, nas quais as notificações exerciam papel fundamental para o acompanhamento das atividades.

Em contrapartida, observou-se um baixo engajamento dos discentes, com apenas 2 respostas, representando 15% do total. Essa diferença pode ter indicado uma menor percepção da importância do sistema de notificações por parte dos estudantes, ou ainda dificuldades de acesso e familiaridade com o processo de levantamento de requisitos.

Com base nos resultados do formulário, observou-se que o e-mail foi apontado como o meio de comunicação preferencial para o recebimento de notificações, por ser considerado um

canal oficial, confiável e mais adequado para comunicações formais. Além disso, identificou-se interesse na implementação de notificações internas no próprio sistema, bem como no uso do WhatsApp como meio secundário de aviso, conforme ilustrado na Figura 2. Também se verificaram diferenças nas preferências entre discentes e docentes, tanto em relação à frequência quanto aos tipos de notificações desejados, refletindo as distintas necessidades e rotinas de cada grupo de usuários.

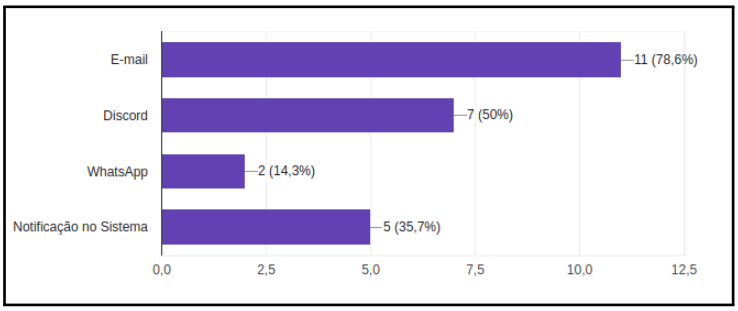


Figura 2 – Preferência de canal de notificação dos usuários.
Fonte: Autoria própria..

Outro ponto abordado no formulário, além do levantamento de requisitos, foi a classificação dos requisitos com base no Método MoSCoW, onde cada usuário classificou todos os requisitos por importância do seu ponto de vista, reunindo essas respostas na Tabela 2 que apresenta os requisitos classificados segundo a metodologia MoSCoW, a qual define as prioridades de implementação em um sistema de notificações para o processo de TCC. Observa-se que a maioria dos requisitos que são classificadas como Must representam, do ponto de vista dos usuários, aspectos essenciais para o funcionamento do sistema. Esses requisitos representam o núcleo (core) da aplicação, voltado à automação das comunicações e ao controle de prazos e documentos — funcionalidades indispensáveis para garantir o andamento adequado das etapas do TCC. Além disso, nota-se que os requisitos classificados como Should e Could correspondem a funcionalidades complementares que, embora não sejam essenciais para o funcionamento inicial do sistema, agregam valor à experiência dos usuários e à eficiência do processo. Essas funcionalidades podem ser incorporadas em versões futuras, ampliando o escopo do sistema de notificações para contemplar aspectos como acompanhamento do progresso do TCC e maior transparência na comunicação entre os participantes. Por fim, o único requisito classificado como Won't refere-se a uma funcionalidade considerada de baixa prioridade no momento, podendo ser reavaliada conforme a evolução das necessidades institucionais e a maturidade do sistema.

Requisito	Descrição detalhada	Classificação MoSCoW
Cadastro e alteração de calendário	Todos cadastrados no calendário atual devem ser notificados após um tempo da criação ou alteração do calendário.	Must
Assinatura de documentos pendentes	Enviar aviso imediato a todos que deveriam assinar determinado documento e ainda não assinaram.	Must
Prazos de envio de proposta/projeto/monografia	Notificar orientadores e alunos sobre os prazos de envio. Definir número de lembretes e intervalos, diferenciando os tipos de usuários.	Must
Envio de proposta/projeto/monografia	Avisar orientador, professor de TCC 1 (quando for proposta ou projeto) e professor responsável.	Must
Agendamento e alteração de banca (avaliadores)	Notificar membros avaliadores após algum tempo da última alteração no agendamento, pedindo confirmação de disponibilidade para a data marcada.	Must
Agendamento e alteração de banca (aluno e curso)	Notificar o aluno e todos do curso após confirmação dos avaliadores, considerando a última alteração.	Should
Registros dos apontamentos da banca	Enviar notificação imediata ao acadêmico quando houver registro de apontamentos pela banca.	Should
Avisos sobre a evolução do tempo do TCC	Notificar o aluno ao longo da linha do tempo do TCC, de acordo com intervalos definidos.	Could
Ciência na reunião	Notificação enviada ao acadêmico para ciência dos registros feitos pelo orientador, indicando concordância com as informações registradas na reunião.	Could
Termo de Desistência de Orientação de TCC pelo Professor Orientador	Notificação enviada ao Coordenador do TCC quando um professor orientador iniciar um processo de desistência de orientação, para que o coordenador possa analisar e deferir ou não o requerimento.	Could
Termo de Substituição de Orientação de TCC	Notificação enviada ao Coordenador do TCC quando o orientador e o aluno orientado tiverem assinado o Termo para a Substituição de Orientação do TCC, para que o coordenador possa analisar e deferir ou não o requerimento.	Could
Termo de Solicitação de Extensão do Prazo de TCC	Notificar o Coordenador do TCC quando tanto o orientador como o aluno orientado assinarem o termo para extensão de prazo do TCC, para que o coordenador possa analisar e deferir ou não o requerimento.	Could
Confirmação de assinatura de documentos	Notificar imediatamente os demais envolvidos quando alguém realizar uma assinatura.	Won't

Tabela 2 – Requisitos de notificação do sistema com cores de classificação MoSCoW.
Fonte: Autoria própria.

3.3.1 Histórias de usuário

As histórias de usuário a seguir foram elaboradas com base nos requisitos de notificação identificados no levantamento. Elas descrevem as necessidades do ponto de vista dos diferentes tipos de usuários do SGTCC (discentes, docentes e coordenadores).

Feature: Cadastro e alteração de calendário (Must)

Como usuário (discente, docente ou coordenador), **quero** ser notificado quando houver uma nova inserção ou modificação no calendário acadêmico do TCC **para** acompanhar alterações importantes nos prazos e me organizar de forma adequada.

Feature: Assinatura de documentos pendentes (Must)

Como usuário que precisa assinar documentos no sistema, **quero** receber uma notificação imediata via e-mail **para** ser lembrado da pendência e evitar atrasos na tramitação dos documentos.

Feature: Prazos de envio de proposta/projeto/monografia (Must)

Como aluno ou orientador, **quero** receber lembretes sobre os prazos de envio da proposta, projeto ou monografia **para** garantir o cumprimento dos prazos definidos pela coordenação.

Feature: Envio de proposta/projeto/monografia (Must)

Como orientador ou docente responsável, **quero** ser notificado quando um aluno submeter sua proposta, projeto ou monografia **para** acompanhar a evolução dos entregáveis e validar o envio dentro do prazo.

Feature: Agendamento e alteração de banca (avaliadores) (Must)

Como membro avaliador de uma banca, **quero** ser notificado após alterações no agendamento **para** confirmar minha disponibilidade e evitar conflitos de agenda.

Feature: Agendamento e alteração de banca (aluno) (Should)

Como discente, **quero** ser notificado após a confirmação da banca pelos avaliadores **para** me preparar com antecedência.

Feature: Registros dos apontamentos da banca (Should)

Como aluno avaliado, **quero** ser notificado quando os avaliadores registrarem apontamentos **para** ter ciência imediata das observações e iniciar os ajustes necessários.

Feature: Avisos sobre a evolução do tempo do TCC (Could)

Como aluno em processo de TCC, **quero** receber notificações periódicas sobre a evolução do tempo do meu TCC **para** manter o acompanhamento do cronograma e evitar atrasos.

Feature: Ciência na reunião (Could)

Como discente participante de reuniões com o orientador, **quero** ser notificado para confirmar ciência dos registros feitos pelo orientador **para** validar as informações discutidas e manter um histórico formalizado das reuniões.

Feature: Termo de Desistência de Orientação de TCC pelo Professor Orientador (Could)

Como coordenado do TCC, **quero** ser notificado um professor orientador tenha assinado o Termo de Desistência de Orientação **para** poder analisar e deferir ou não o requerimento e então tomar as ações necessárias.

Feature: Termo de Substituição de Orientação de TCC (Could)

Como coordenado do TCC, **quero** ser notificado tanto o professor orientador quanto o aluno orientado tenham assinado o Termo de Substituição de Orientação **para** poder analisar e deferir ou não o requerimento e então tomar as ações necessárias.

Feature: Termo de Solicitação de Extensão do Prazo de TCC (Could)

Como coordenado do TCC, **quero** ser notificado tanto o professor orientador quanto o aluno orientado tenham assinado o Termo de Extensão de Prazo de TCC **para** poder analisar e deferir ou não o requerimento e então tomar as ações necessárias.

Feature: Confirmação de assinatura de documentos (Won't)

Como participante envolvido em um processo de assinatura, **quero** ser notificado quando outro membro realizar uma assinatura **para** acompanhar o progresso do fluxo de documentos em tempo real.

Figura 3 – Botão de confirmação de bancas para o professor responsável na página de bancas de defesa.

Fonte: Autoria própria..

3.3.2 Protótipos de Telas

Os protótipos das telas foram iniciados, com as adaptações necessárias nas interfaces já existentes no TCC, para que o sistema possa suportar novas interações.

A Figura 3 apresenta a interface destinada ao responsável pelas bancas, cujo objetivo principal é a gestão e confirmação das defesas de TCC. Nessa tela, são exibidas informações essenciais de cada banca, como discente, orientador, local, data e horário, além de um indicador visual de status temporal da defesa. O elemento central desta funcionalidade é o botão de confirmação, que permite ao responsável validar oficialmente a realização da banca, alterando o estado do registro para "Confirmada". Esse processo garante maior controle institucional, prevenindo inconsistências no agendamento e assegurando que apenas bancas devidamente homologadas avancem no fluxo do sistema, como no envio de notificações e habilitação das etapas subsequentes. Dessa forma, a funcionalidade contribui para a padronização e confiabilidade do gerenciamento das bancas de TCC.

Outra tela a ser desenhada foi uma interface onde o responsável pelo TCC pode gerenciar as regras de cada evento notificável, podendo ser acessada pelo menu lateral utilizando a opção configurações de notificações, conectado como um usuário com permissão de professor responsável. A Figura 4 apresenta um recorte dessa interface que apresenta o contexto onde o professor responsável pode escolher quais notificações estão habilitadas. Já a Figura 5 apresenta o restante da mesma tela, onde é possível modificar a frequência, o intervalo entre envios, o tempo de espera entre o acontecimento do evento no sistema e o envio, além de poder ativar ou desativar notificações de algum evento. Essas configurações permitem que o responsável mantenha as notificações úteis e não invasivas, mantendo a funcionalidade útil para todos os usuários do SGTCC.

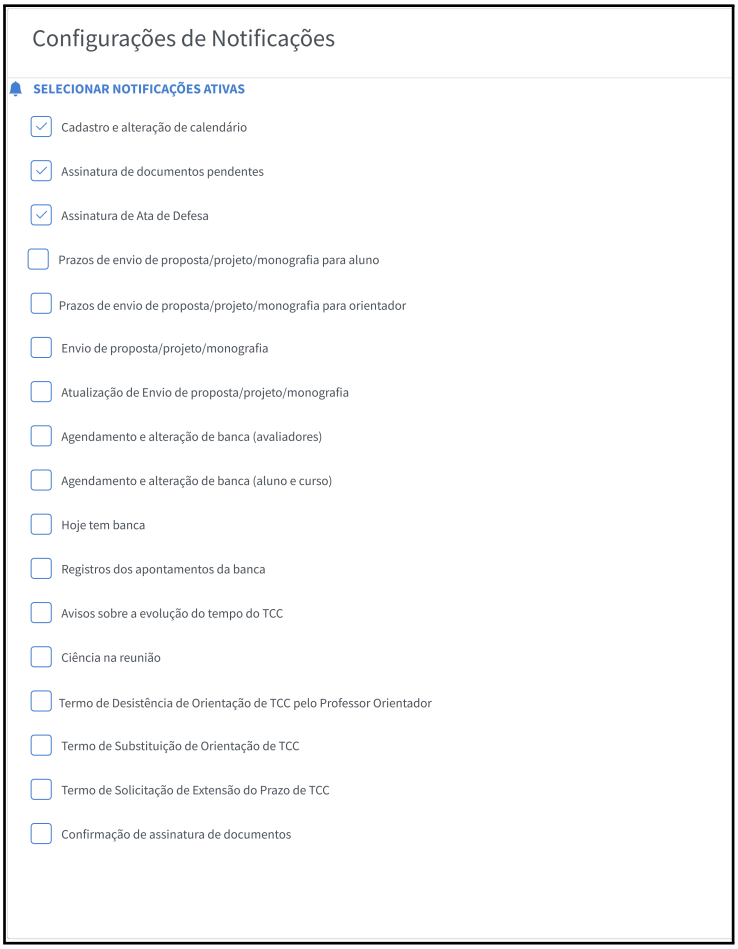


Figura 4 – Tela de configuração de notificações ativas pelo professor responsável.
Fonte: Autoria própria.

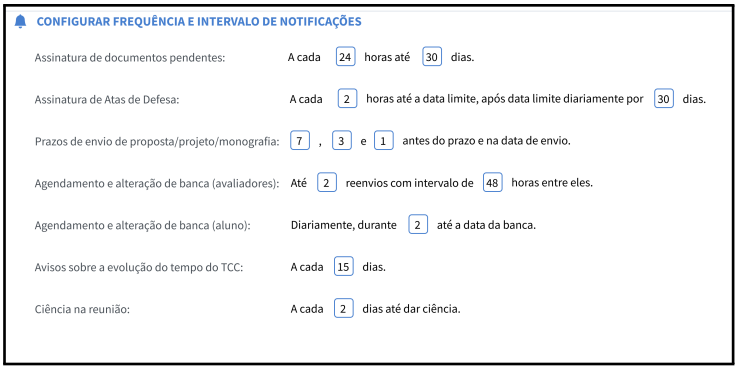


Figura 5 – Tela de configuração de regras de notificação pelo professor responsável.
Fonte: Autoria própria.

4 ESTRATÉGIAS DE NOTIFICAÇÃO DO SGTCC

A implementação foi organizada a partir do planejamento inicial, que contemplou a formulação dos modelos de notificação, a definição dos modelos de e-mail e a definição do fluxo de notificações.

O objetivo deste capítulo é apresentar de forma clara o processo de construção da solução, evidenciando as decisões técnicas, os recursos empregados e a maneira como as notificações foram estruturadas no sistema. A seguir, a seção de definição detalha os elementos que serviram de base para a implementação.

4.1 Definição

A partir do levantamento de requisitos apresentado na seção 3.2, foi possível estruturar um conjunto de notificações alinhado às demandas reais dos usuários e às particularidades do fluxo de trabalho do SGTCC. Essa etapa de definição teve como foco transformar os requisitos funcionais em regras práticas e automatizáveis, descrevendo para cada evento: o gatilho (*trigger*), os destinatários, o momento do envio e a estratégia de reenvio ou cancelamento. Cada notificação foi formalizada em um formato padronizado, conforme o modelo descrito a seguir:

Estrutura de definição de notificação

Evento: identifica a ação que dispara a notificação (ex: criação, atualização ou confirmação de dados);

Alvo: define os usuários que receberão a mensagem, podendo incluir acadêmicos, orientadores, avaliadores, coordenadores ou membros externos à instituição;

Momento do envio: determina o intervalo entre o evento e o envio da notificação, permitindo ajustes para evitar redundâncias;

Reenvio: indica se há necessidade de lembrete ou tentativa posterior;

Observações: campo destinado a exceções ou regras adicionais de comportamento.

A adoção desse formato padronizado permitiu que as notificações fossem registradas de maneira uniforme e facilmente mantidas por meio de arquivos de configuração. Durante a etapa de definição, também foram especificados os **modelos de e-mail** associados a cada notificação, os **gatilhos do sistema** (*hooks*) que disparam os eventos e o **fluxo lógico de envio**, detalhados nas subseções seguintes.

Essas definições asseguraram que a implementação seguisse um modelo previsível, reduzindo inconsistências e facilitando futuras expansões — como a integração de novos canais de comunicação (ex: notificações internas, WhatsApp ou SMS).

4.1.1 Modelos de notificação

As regras de notificação definem o comportamento operacional do sistema, descrevendo *quando*, *para quem* e *como* cada mensagem deve ser enviada. Elas foram formuladas com base na análise dos fluxos de trabalho do SGTCC e nas respostas do questionário aplicado aos usuários, apresentados na seção 3.2. A seguir, são apresentadas todas as regras implementadas, acompanhadas de uma breve explicação sobre a motivação e o comportamento de cada uma.

Cadastro e alteração de calendário

Evento: Criação ou atualização do calendário acadêmico;

Alvo: Todos os usuários cadastrados no calendário atual;

Momento do envio: 1h após a última alteração;

Reenvio: Não é necessário;

Observações: Professor responsável pode desativar; evitar notificações desnecessárias.

Essa notificação é essencial para garantir que alunos e docentes estejam sempre cientes de mudanças nos prazos e eventos do calendário acadêmico. O envio é programado com um pequeno atraso para evitar disparos múltiplos durante edições consecutivas do calendário.

Assinatura de documentos pendentes

Evento: Documento não assinado a mais de 24h;

Alvo: Quem estiver com assinatura pendente no documento;

Momento do envio: Após 24h da criação do documento;

Reenvio: A cada 24h até a assinatura ser realiza ou até 30x;

Observações: Documentos que geram essa notificação (TCO. TEP, TCAI, TSO).

Essa notificação tem como objetivo acelerar o processo de validação documental, evitando atrasos na tramitação dos termos obrigatórios. O reenvio diário foi projetado para reforçar a necessidade da assinatura até que o fluxo esteja completo.

Assinatura de Ata de Defesa

Evento: Documento não assinado após 2:30 h começa notificar;

Alvo: Quem estiver com assinatura pendente no documento;

Momento do envio: Após duas horas da defesa;

Reenvio: A cada duas horas até assinar ou até a data limite para assinatura. Depois a cada dia até 30x;

Observações: Sem observações.

A Ata de Defesa é um documento sensível e com prazos rígidos. Por isso, a notificação é configurada para ter uma frequência de envio mais alta nas primeiras horas após a banca, reduzindo o risco de atrasos e garantindo o registro formal imediato do resultado.

Prazos de envio para o aluno

Evento: Proximidade de prazos de entrega;

Alvo: Discentes que não realizaram o envio;

Momento do envio: 7, 3 e 1 dia antes do prazo. No dia também;

Reenvio: Reenvio por prazo pré definido;

Observações: Sem observações.

Essa notificação atua como lembrete preventivo para os alunos, reforçando o cumprimento dos prazos de entrega de cada atividade. O envio escalonado (7, 3 e 1 dia antes) foi definido com base em boas práticas de gestão de prazos em ambientes acadêmicos.

Prazos de envio para o orientador

Evento: Proximidade de prazos de entrega;

Alvo: Docentes com orientações no semestre corrente;

Momento do envio: 7, 3 e 1 dia antes do prazo. No dia também;

Reenvio: Reenvio por prazo pré definido;

Observações: Cuidar caso o orientador tenha mais de um orientado no mesmo semestre.

Similar à notificação anterior, essa regra visa manter o orientador ciente do progresso de seus orientandos, permitindo o acompanhamento proativo. A diferenciação por perfil de usuário evita redundâncias e melhora a clareza das mensagens.

Envio de documento

Evento: Submissão do envio feita no sistema;

Alvo: Orientador do discente que realizou o envio, Professor Responsável do TCC e Professor de TCC1 (quando entrega de TCC1);

Momento do envio: Imediato;

Reenvio: Não é necessário;

Observações: Sem observações.

Trata-se de uma notificação disparada automaticamente logo após o envio de um documento. Sua função é manter os professores informados sobre novas submissões, agilizando o processo de avaliação e validação.

Atualização de envio de documento

Evento: Submissão do envio feita no sistema;

Alvo: Orientador do discente que realizou o envio, Professor Responsável do TCC e Professor de TCC1 (quando entrega de TCC1);

Momento do envio: Imediato;

Reenvio: Não é necessário;

Observações: Sem observações.

Complementar à regra anterior, essa notificação informa sobre alterações em documentos já enviados. Ela garante rastreabilidade e comunicação transparente entre aluno e equipe docente, evitando confusões com versões antigas.

Agendamento e alteração de banca (avaliadores)

Evento: Criação ou alteração de agendamento de banca;

Alvo: Docentes e membros externos listados como avaliadores na banca;

Momento do envio: 1h após a criação ou alteração da banca;

Reenvio: Até 2 reenvios, intervalo de 48h;

Observações: Cuidar para não criar novos eventos quando houver alteração antes do envio.

Essa notificação é fundamental para o gerenciamento de bancas avaliadoras, garantindo que todos os participantes recebam confirmação formal do agendamento, garantindo que não haja uma banca em que um avaliador não tenha ciência desse compromisso. O atraso intencional de uma hora evita duplicidades caso o professor realize ajustes sucessivos.

Confirmação do agendamento e/ou alteração de banca (aluno e avaliadores)

Evento: Após confirmação dos avaliadores;

Alvo: Aluno defensor da banca e docentes e membros externos listados como avaliadores na banca;

Momento do envio: Imediato;

Reenvio: 24h antes da banca;

Observações: Confirmação das bancas pelo Professor responsável.

Essa regra fecha o ciclo de comunicação da banca, garantindo que todos os envolvidos estejam cientes da confirmação final. O reenvio programado 24 horas antes funciona como lembrete de compromisso e reforça a organização do evento.

Registros dos apontamentos da banca

Evento: Após a defesa se houver inclusão de apontamentos pela banca;

Alvo: Aluno defensor da banca;

Momento do envio: Imediato;

Reenvio: Não é necessário;

Observações: Sem observações.

Essa notificação foi projetada para promover transparência no processo avaliativo, informando ao aluno que os apontamentos realizados pela banca já estão disponíveis no sistema para consulta.

Ciência na reunião

Evento: Registro de reunião pelo orientador;

Alvo: Aluno orientado;

Momento do envio: Imediato;

Reenvio: A cada dois dias até dar a ciência.;

Observações: Sem observações.

Tem função de assegurar que o aluno confirme o recebimento e leitura das anotações de reuniões. O reenvio periódico serve como lembrete até que a ciência seja registrada no sistema.

Submissão de termos

Evento: Submissão de termo no sistema;

Alvo: Professor responsável;

Momento do envio: Imediato;

Reenvio: Não é necessário.;

Observações: Termos que geram essa notificação (TDO, TSO, TEP).

Essa notificação reforça a rastreabilidade documental no sistema, garantindo que o professor responsável seja imediatamente informado sobre novas submissões de termos institucionais que demandam a análise e validação formal.

4.1.2 Templates de E-mail

Os modelos de e-mail foram elaborados para padronizar a comunicação do sistema com os usuários, garantindo clareza e consistência das informações enviadas. A estrutura básica dos e-mails segue o padrão:

Modelo de e-mail

Assunto: identifica de forma direta o tipo de notificação;

Saudação: personalizada com o nome do destinatário;

Corpo da mensagem: contém o conteúdo específico da notificação;

Rodapé: Atenciosamente,

Sistema de Gerenciamento de TCC (SGTCC)

Universidade Tecnológica Federal do Paraná - Câmpus Guarapuava.

A seguir, são apresentados os modelos para cada tipo de notificação.

Cadastro e alteração de calendário

Assunto: [SGTCC] Atualização no calendário acadêmico de TCC1/2 ANO/PERÍODO

Corpo da mensagem:

Um novo calendário acadêmico foi cadastrado/alterado em `<DATA>`.

Por favor, consulte o sistema para verificar os prazos atualizados.

Assinatura de documentos pendentes

Assunto: [SGTCC] Assinatura pendente no documento

Corpo da mensagem:

O documento `<NOME_DO_DOCUMENTO>`, criado em `<DATA>`, ainda não foi assinado.

Sua assinatura é necessária para dar continuidade ao processo.

Assinatura pendente de Ata de Defesa

Assunto: [SGTCC] Assinatura pendente da Ata de Defesa

Corpo da mensagem:

A Ata de Defesa referente ao TCC `<TITULO_DO_TCC>` precisa ser assinada com urgência.

O prazo de assinatura é reduzido devido à relevância do documento.

Prazos de envio(acadêmico)

Assunto: [SGTCC] Lembrete de prazo para envio de <TITULO_DA_ATIVIDADE>

Corpo da mensagem:

O prazo final para envio de <TITULO_DA_ATIVIDADE> é <DATA>.

Faltam <X> dias para o encerramento. Não deixe para a última hora!

Prazos de envio(orientador)

Assunto: [SGTCC] Lembrete de prazo para envio de atividades de seus orientandos

Corpo da mensagem:

O prazo final para envio de <TITULO_DA_ATIVIDADE> de seus orientandos é <DATA>.

Acompanhe o progresso no sistema para evitar atrasos.

Envio de atividade

Assunto: [SGTCC] Novo envio de <TITULO_DA_ATIVIDADE>

Corpo da mensagem:

O acadêmico <NOME_DO_ACADÊMICO> enviou sua <TIPO_DOCUMENTO> em <DATA>.

O documento já está disponível para avaliação.

Atualização de envio de atividade

Assunto: [SGTCC] Atualização no envio de <TITULO_DA_ATIVIDADE>

Corpo da mensagem:

O acadêmico <NOME_DO_ACADÊMICO> atualizou seu envio de <TITULO_DA_ATIVIDADE>. A nova versão está disponível no sistema.

Agendamento e alteração de banca (avaliadores)

Assunto: [SGTCC] Agendamento de banca para avaliação

Corpo da mensagem:

Você foi designado como avaliador na banca do aluno <NOME_DO_ACADÊMICO>,com TCC intitulado <TITULO_DO_TCC>. Data: <DATA_BANCA>

Horário: <HORARIO>

Local: <LOCAL>

Por favor, confirme sua disponibilidade.

Agendamento e alteração de banca (aluno)

Assunto: [SGTCC] Agendamento de banca confirmado

Corpo da mensagem:

Sua banca de defesa foi agendada com os seguintes detalhes: Data: `<DATA_BANCA>`

Horário: `<HORARIO>`

Local: `<LOCAL>`

Confira os detalhes no sistema.

Registros dos apontamentos da banca

Assunto: [SGTCC] Apontamentos da banca disponíveis

Corpo da mensagem:

Os apontamentos registrados pela banca do seu TCC já estão disponíveis no sistema.

Acesse sua área para consultá-los.

Ciência na reunião

Assunto: [SGTCC] Registro de reunião disponível

Corpo da mensagem:

O orientador `<NOME_ORIENTADOR>` registrou uma reunião no sistema.

Por favor, confirme a ciência das informações acessando sua área no SGTCC.

Submissão de Termos ao Professor Responável

Assunto: [SGTCC] Registro de `TITULO_DO_TERMO`

Corpo da mensagem:

Foi registrado um `<TITULO_DO_TERMO>` relacionado ao TCC de `<NOME_DO_ALUNO>`, intitulado `<TITULO_DO_TCC>`.

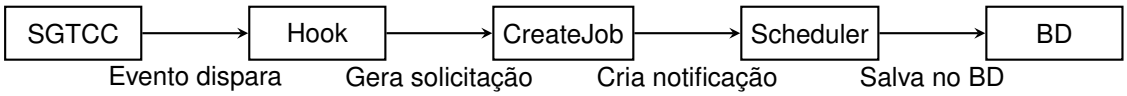
Consulte o sistema para verificar os detalhes.

4.1.3 Fluxo de Envio

Os diagramas de sequência apresentados nas Figuras 6 e 7 ilustram o fluxo de criação de uma notificação no SGTCC, detalhando as interações entre os componentes responsáveis pela construção, armazenamento e disparo da notificação. A Figura 6 descreve o processo de criação e armazenamento da notificação que se inicia quando ocorre um evento interno, como, por exemplo, a criação de uma banca de defesa. Esse evento é encaminhado

ao módulo de *hook* correspondente, que atua como um ponto de extensão responsável por reagir automaticamente às alterações relevantes no sistema. Ao receber o evento, o *hook* aciona o componente `CreateJob`, que executa, de maneira assíncrona, uma requisição ao `SchedulerService`, serviço responsável por estruturar os dados da notificação, aplicar regras de negócio e determinar os metadados necessários para a etapa posterior de envio. Após essa etapa, o `SchedulerService` salva a notificação no banco de dados (SQLite), onde permanece armazenada até o momento de processamento pelos serviços responsáveis pelo envio.

A Figura 7 descreve o fluxo completo de envio das notificações previamente criadas e armazenadas. Esse processo é assíncrono e ocorre no *container workers*, separado da aplicação *web*, envolvendo componentes que operam de forma periódica por meio de *jobs* agendados. O fluxo tem início quando o ActiveJob aciona periodicamente o `SchedulerPollerJob`, que executa consultas no banco de dados a fim de localizar notificações pendentes e elegíveis para envio. Após recuperar os registros, o `SchedulerPollerJob` envia essas informações ao `ProcessorService`, o qual aplica a lógica central do sistema de notificações, incluindo validações de prazos, regras específicas do fluxo acadêmico e também a verificação de condições que possam impedir o envio. Para isso, o `ProcessorService` consulta o `StopChecker`, que determina se existem motivos para cancelar o envio de alguma notificação. Caso o envio seja permitido, o `ProcessorService` aciona o `DispatchJob`, responsável por criar uma nova tarefa assíncrona para realizar o envio propriamente dito. Esse *job* é então encaminhado ao componente `Dispatcher`, que o direciona ao mensageiro correto a partir do tipo da notificação (por e-mail, no sistema, etc.). Em seguida, o `Dispatcher` chama o `NotificationMailer`, componente encarregado de gerar e enviar o e-mail ao destinatário. Finalmente, o `NotificationMailer` envia a mensagem ao usuário, completando o ciclo de notificação. Esse fluxo demonstra como o sistema separa as etapas de criação e envio, permitindo que não haja sobrecarga de responsabilidades em poucos serviços.



Legenda:
Hook = `Hooks::ExaminationBoard`
CreateJob = `Notifications::CreateJob`
Scheduler = `Notifications::SchedulerService`
BD = Banco de Dados `SQLite` para notificações e *jobs*

Figura 6 – Diagrama de sequência — Processo de criação de notificação

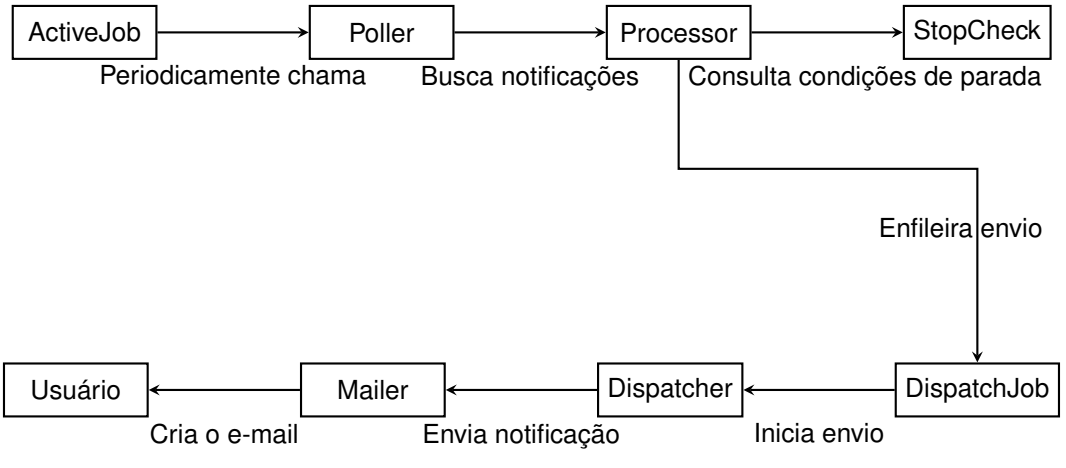


Figura 7 – Diagrama de sequência — Processo de envio de notificações

4.2 Implementação das funcionalidades

O processo de implementação das funcionalidades iniciou-se após a conclusão do planejamento arquitetural descrito na seção anterior, tendo como objetivo principal integrar o sistema de notificações ao SGTCC de forma modular, assíncrona e independente do fluxo principal da aplicação. Para alcançar esse propósito, foi necessário adaptar a estrutura existente do sistema, incorporando novos componentes responsáveis pela criação, agendamento, envio e controle das notificações.

A implementação foi conduzida de maneira incremental, partindo da configuração do ambiente de desenvolvimento e do mecanismo de filas, seguida pela modelagem das classes de notificação, pela definição dos serviços responsáveis pelo processamento assíncrono e pela integração com os eventos internos da aplicação. Cada etapa foi validada por meio de testes automatizados, garantindo a confiabilidade e a consistência do sistema diante de diferentes cenários operacionais.

Nos tópicos a seguir, são detalhadas as etapas do processo de implementação, desde a configuração do ambiente e das filas de execução até a arquitetura de serviços, a integração com os eventos do sistema e a estratégia de testes utilizada para assegurar o correto funcionamento das funcionalidades desenvolvidas.

4.2.1 Configuração do ambiente de desenvolvimento

Apesar do SGTCC já ser uma aplicação em pleno funcionamento, foram necessárias alterações estruturais no ambiente de desenvolvimento para integrar o sistema de notificações de forma assíncrona, utilizando o *framework* `Active Job`[1] do Ruby on Rails em conjunto com o adaptador `Solid Queue`. Essas modificações visaram garantir isolamento, paralelismo e tolerância a falhas durante o processamento das notificações. As modificações envolveram ajustes na configuração do Docker, na definição de dependências Ruby via *Gemfile*, e na criação de arquivos auxiliares para orquestrar o enfileiramento de tarefas assíncronas.

No Docker, o serviço da aplicação foi configurado para incluir o processo de enfileiramento do Solid Queue em execução paralela ao servidor web, garantindo que os *jobs* pudessem ser processados em segundo plano. O arquivo `docker-compose.yml` passou a incluir um serviço adicional responsável pelo processamento das filas, conforme o exemplo simplificado a seguir:

Listing 4.1 – Trecho adicionado ao arquivo docker-compose.yml com serviço de filas Solid Queue.

```
1  workers:
2    <<: *app
3    command: >
4      bash -c "
5        bundle exec bin/jobs
6        "
7    environment:
8      RAILS_ENV: development
```

Essa separação permite que o serviço `workers` seja reiniciado ou escalado independentemente do servidor web, prática comum em arquiteturas que utilizam *filas distribuídas*. O serviço `workers` é responsável por executar o comando `bin/jobs`, que inicializa o processo de monitoramento e execução das filas configuradas pelo Solid Queue. Esse comando é mantido em execução contínua dentro do contêiner, funcionando de forma independente do servidor web, mas conectado ao mesmo banco de filas.

No arquivo Gemfile, foram adicionadas dependências específicas para suportar o processamento assíncrono (*gem solid_queue*) e a integração com o SQLite(*gem sqlite3*). O motivo da escolha da `gem solid_queue` está detalhado em subseção 4.2.3. Essa decisão simplificou a infraestrutura, permitindo que todo o processamento ocorresse no próprio banco SQLite configurado para a fila.

[1] Disponível em https://guides.rubyonrails.org/active_job_basics.html. Acessado em 05 de novembro de 2025.

Para definir a estrutura e o comportamento das filas, foi criado o arquivo **config/-queue.yml**, no qual são especificadas as filas disponíveis, suas prioridades e a quantidade de *workers* alocados para cada tipo de tarefa:

Listing 4.2 – Configuração dos `dispatchers` (Despachantes) e `workers` (Trabalhadores) no arquivo config/queue.yml.

```
1 default: &default
2   dispatchers:
3     - polling_interval: 1
4       batch_size: 500
5   workers:
6     - queues: "*"
7       threads: 3
8       processes: 1
9       polling_interval: 0.1
```

Nesta configuração, o Solid Queue utiliza um único *dispatcher* com intervalo de varredura de um segundo e tamanho máximo de lote (`batch size`) de 500 tarefas por ciclo. Esse componente é responsável por identificar novas tarefas enfileiradas no banco de dados e distribuí-las para os *workers*. Os *workers* são configurados para processar todas as filas (`queues: "*"`) com três *threads* de execução concorrentes por processo, garantindo paralelismo e eficiência no tratamento de notificações, e-mails e demais tarefas assíncronas. O parâmetro `polling_interval: 0.1` assegura uma checagem contínua da fila, com latência mínima entre a criação de uma tarefa e sua execução efetiva. A configuração contida no arquivo `config/queue.yml` permite que o Solid Queue funcione sem dependências externas, dispensando a utilização de sistemas adicionais como Redis ou Sidekiq. Todo o controle das filas ocorre dentro da própria aplicação Rails, utilizando o banco de dados para persistência e controle de concorrência.

O SGTCC utiliza o banco de dados PostgreSQL para armazenar as informações da aplicação. Contudo, para a implementação do sistema de notificações, optou-se por uma arquitetura de múltiplos bancos de dados, adicionando um segundo repositório exclusivo para o gerenciamento das filas de execução.

Conforme detalhado na subseção 4.2.3, o `Solid Queue` foi configurado para operar sobre um banco de dados `SQLite` dedicado, isolado do banco principal da aplicação. Essa decisão técnica foi fundamental para evitar situações de *database locking* e garantir que operações intensivas em escrita — como a criação, atualização e remoção de tarefas assíncronas — não interferissem no desempenho do PostgreSQL.

Além de reduzir a possibilidade de contenção de recursos, o uso de um banco secundário também simplifica a manutenção do sistema, permitindo que o módulo de notificações seja escalado ou reiniciado independentemente do restante da aplicação.

A configuração da coexistência entre os dois bancos foi definida no arquivo `config/database.yml`, conforme o exemplo a seguir:

Listing 4.3 – Configuração de múltiplos bancos de dados no arquivo config/database.yml.

```
default: &default
  adapter: postgresql
  encoding: utf8
  pool: 5
  username: <%= ENV['db.username'] %>
  password: <%= ENV['db.password'] %>
  host: <%= ENV['db.host'] %>
  migrations_paths: db/migrate

development:
  primary:
    <<: *default
    database: postgres

  queue:
    adapter: sqlite3
    database: storage/queue_development.sqlite3
    migrations_paths: db/queue_migrate
    pool: 5
    timeout: 15000
    properties:
      journal_mode: WAL
... # continuação da configuração do banco de dados em outros ambientes
```

No exemplo acima, o ambiente `development` define duas conexões: a conexão `primary`, responsável pelos dados centrais da aplicação, e a conexão `queue`, utilizada exclusivamente pelo Solid Queue. O modo `WAL` (*Write-Ahead Logging*) foi habilitado no SQLite para melhorar o desempenho de gravação concorrente, assegurando integridade e eficiência no processamento de tarefas assíncronas.

4.2.2 Estruturação das classes de notificação

Para dar início ao desenvolvimento, foi projetada uma arquitetura flexível e extensível, capaz de acomodar novos tipos de eventos, conteúdos e regras de disparo, sem necessidade de alterações estruturais no código. Essa flexibilidade decorre da separação explícita entre três modelos principais — *Notification*, *NotificationTemplate* e *NotificationRule* — cada um responsável por um aspecto independente da gestão de notificações. Essa divisão reduz o acoplamento entre as partes e permite que novos comportamentos sejam adicionados apenas por meio de configurações e registros no banco de dados, sem que seja necessário modificar as classes existentes. Por exemplo, para introduzir um novo tipo de notificação no sistema, basta criar um novo *template* e definir suas regras correspondentes, sem editar os serviços ou *jobs* já existentes. De forma semelhante, é possível alterar textos, periodicidades, destinatários ou condições de envio apenas modificando dados persistidos, preservando o restante da arquitetura.

O modelo *Notification* é responsável por registrar cada instância de notificação gerada no sistema. Ele armazena informações como o destinatário (recipient), o tipo da notificação (notification_type), os dados específicos do evento (data) e o estado atual (status, scheduled_at, sent_at). Essa modelagem permite acompanhar o ciclo de vida completo de uma notificação, desde o agendamento até o envio e eventuais reenvios.

O modelo *NotificationTemplate* define a estrutura textual das notificações, contendo campos como *key*, *subject* e *body*, que servem de base para mensagens de e-mail e alertas exibidos no sistema. Essa separação entre dados e conteúdo textual facilita a customização dos modelos de mensagem, permitindo que o administrador altere títulos e textos sem a necessidade de alterar o código-fonte.

Já o modelo *NotificationRule* define as regras de disparo, como o número de dias ou horas de antecedência em relação ao evento, o número máximo de tentativas de reenvio e o intervalo entre elas. Essas regras são associadas a um *template* e permitem configurar notificações flexíveis, adaptando-se a diferentes cenários do sistema, como prazos de atividades ou confirmação de bancas.

4.2.3 Processamento assíncrono com Active Job e Solid Queue

Durante o desenvolvimento do sistema de notificações, tornou-se necessário lidar com operações assíncronas — tarefas que não devem ser executadas no mesmo ciclo da requisição web, como o envio de e-mails, o agendamento de mensagens futuras e a execução de verificações recorrentes. Para atender a essa necessidade, optou-se pela utilização do Active Job, *framework* nativo do Ruby on Rails que fornece uma interface padronizada para a criação, enfileiramento e execução de tarefas em segundo plano. O Active Job abstrai a lógica de comunicação entre a aplicação e o mecanismo de fila, permitindo que o mesmo código seja compatível com diferentes adaptadores.

O desenvolvimento do processamento de tarefas assíncronas iniciou-se com o adaptador SideKiq, uma biblioteca que gerencia as tarefas em segundo plano, porém foram encontradas algumas dificuldades ao utilizá-lo, a primeira foi um aumento das dependências do projeto, pois o SideKiq necessita de um servidor para executar as tarefas e outro servidor para rodar o Redis[2], um armazenamento de estrutura de dados em memória, o qual é utilizado para armazenar os serviços em segundo plano, outro problema encontrado foi a utilização do banco de dados da aplicação para armazenar as notificações, o que pode acarretar em atrasos nas requisições dos usuários, pelo fato do banco estar sendo utilizado por uma tarefa assíncrona durante o processamento de uma notificação.

Pensando em contornar esses problemas foi realizada a mudança de adaptador de tarefas em segundo plano, a nova escolha foi o Solid Queue[3], uma solução projetada para oferecer um mecanismo de enfileiramento leve, eficiente e totalmente integrado à aplicação, sem necessidade de dependências externas, outro ponto que baseou essa mudança foi que esse é o adaptador padrão do Rails a partir da versão 8, além desses motivos o Solid Queue também facilita trabalhar com outro banco de dados para as notificações e os processos assíncronos, a tecnologia de banco de dados escolhida para esse banco secundário foi o SQLite, por se tratar de um banco de dados embutido, leve e por não precisar de um servidor.

4.2.4 Arquitetura de Serviços (Service Objects)

Para manter a lógica de negócios organizada, testável e aderente ao Princípio da Responsabilidade Única (SRP), a arquitetura do sistema de notificações foi dividida em um conjunto de *Service Objects* (Objetos de Serviço) e *Jobs* (Tarefas) especializados. Cada classe tem um propósito único, evitando que Models (como `Notification`) ou Jobs (como `DispatchJob`) acumulem responsabilidades. A arquitetura de serviços é composta pelos seguintes componentes:

- **`Notifications::CreateJob` (Job de Criação):** Este é um *job* do Active Job. Sua única função é receber os dados brutos de um evento (tipo de notificação, destinatário, etc.) e chamar o `SchedulerService`. Esta classe é o ponto-chave da arquitetura para resolver problemas de concorrência: os *Hooks* (executados pelo processo web do Puma) apenas enfileiram este job — uma operação de escrita muito rápida na tabela `solid_queue_jobs` — em vez de chamar o `SchedulerService` diretamente. Isso transfere a lógica de criação da notificação (que envolve múltiplas leituras e escritas nas tabelas `notifications` e `notification_templates`) para o processo *worker* do Solid Queue. Ao garantir que apenas o processo *worker* escreva

[2] Disponível em https://redis.io/. Acessado em 06 de novembro de 2025.

[3] Disponível em https://github.com/rails/solid_queue. Acessado em 06 de novembro de 2025.

no banco de dados SQLite, o conflito de `database is locked` com o processo `web` é eliminado.

- **`Notifications::SchedulerService` (Serviço de Agendamento):** Responsável por *criar* ou *atualizar* o registro da `Notification` no banco de dados. Ele recebe os dados do `CreateJob`, consulta o `NotificationTemplate` e o `NotificationRule` associados para calcular o `scheduled_at` (horário de envio). Ele define o status inicial como `pending` (para envio imediato) ou `scheduled` (para envio futuro) e persiste o registro. Ele **não** enfileira o envio.

- **`Notifications::SchedulerPollerJob` (Job de Sondagem/Recorrente):** Este é um job recorrente (definido no `recurring.yml` do Solid Queue para rodar a cada 15 minutos). Sua função é consultar o banco de dados por notificações "enviáveis"(usando o escopo `Notification.pending_to_send`, que busca status `pending` ou `scheduled` com `scheduled_at` no passado). Para cada notificação encontrada, ele invoca o `ProcessorService`.

- **`Notifications::ProcessorService` (O "Cérebro"):** Este é o serviço de lógica mais complexo. Ele decide *se* uma notificação encontrada pelo *Poller* deve realmente ser enviada. Ele executa as seguintes verificações:

  - Verifica se a notificação já foi finalizada (enviada, falhada ou cancelada).
  - Chama o `Notifications::StopChecker` para verificar se a condição de parada foi atingida (ex: o documento já foi assinado?).
  - Verifica se o número de tentativas (`attempts`) excedeu o `max_attempts`.
  - Se todas as verificações passarem, ele enfileira o `Notifications::DispatchJob` para o envio.
  - Após enfileirar, ele atualiza o status da notificação (incrementa `attempts` e define como `sent` se for envio único, ou `scheduled` para o futuro se for uma notificação insistente).

- **`Notifications::StopChecker` (Serviço de Verificação):** Um serviço altamente especializado chamado pelo `ProcessorService`. Ele contém a lógica de negócios para determinar se uma notificação "insistente"(como um lembrete de assinatura pendente) deve ser interrompida. Ele se conecta ao banco de dados principal (PostgreSQL) para verificar o estado de outros modelos (ex: `Signature.status == true`).

- **`Notifications::DispatchJob` (Job de Despacho):** Um job simples, sua única tarefa é receber um `notification_id` e chamar o `Dispatcher`. O uso de um job aqui isola a tentativa de envio (que pode falhar devido a problemas de rede

ou API) da lógica de processamento, permitindo que o Solid Queue use suas próprias políticas de retentativa para falhas de infraestrutura.

- **`Notifications::Dispatcher` (Serviço de Entrega):** O componente final. Ele lê o atributo `channel` do template da notificação (ex: 'email') e invoca o método apropriado, como `NotificationMailer.generic_email(...)`.

Esta arquitetura desacoplada garante que cada componente possa ser modificado e testado de forma independente, além de resolver os desafios de concorrência do banco de dados SQLite.

4.2.5 Integração com Eventos do Sistema (Hooks)

A integração do sistema de notificações com a lógica de negócios principal da aplicação é realizada através de *Hooks* (Ganchos) e *callbacks* do Active Record.

A lógica é implementada da seguinte forma:

- **Callbacks nos Modelos:** Os modelos principais da aplicação (ex: `Document`, `AcademicActivity`, `ExaminationBoard`) utilizam callbacks como `after_commit` que são chamadas após a criação ou atualização de um objeto daquele modelo. O uso de `after_commit` é crucial, pois garante que a transação do banco de dados principal (PostgreSQL) foi concluída com sucesso antes de tentar criar uma notificação (que escreve no banco SQLite).
- **Módulos de Hooks:** Os callbacks não contêm lógica de negócios. Em vez disso, eles chamam um método de classe em um módulo de Hook dedicado, como `Notifications::Hooks::Documents.document_created(self)`.
- **Tradução da Lógica:** Dentro do módulo de Hook (ex: `Notifications::Hooks::Documents`), o método `document_created` recebe o objeto do modelo (ex: um `Document`). A função do hook é transformar esse objeto de negócios em parâmetros de notificação: quem deve ser notificado (`recipient`), qual o tipo de notificação (`notification_type`) e quais dados (`data`) o e-mail precisará.
- **Enfileiramento do Job:** A ação final do hook é chamar o `Notifications::CreateJob.perform_later(...)`, passando os parâmetros necessários para a criação da notificação.
- **Reutilização de Código:** Um módulo `Notifications::HookHelpers` foi criado para compartilhar lógicas comuns entre os diferentes hooks, como o método `schedule_notification` (um encapsulador para `CreateJob.perform_later`) e `event_key` (para gerar chaves de evento únicas).

Essa abordagem mantém os modelos da aplicação livres de regras complexas de notificação, centralizando a decisão de quando e quem notificar nos módulos de Hook. Dessa forma, o processamento pesado — como criação de registros, cálculos de agendamento ou chamadas externas — é deslocado para o sistema de jobs em segundo plano, reduzindo o acoplamento e preservando o desempenho do processo web.

4.2.6 Estratégia de Testes e Validação

Garantir a confiabilidade de um sistema de notificações assíncrono e distribuído (entre dois bancos de dados e múltiplos serviços) exige uma estratégia de testes robusta. A abordagem utilizada foi a de **testes de unidade** com RSpec, focando em isolar e validar cada componente da arquitetura. A cobertura de testes foi dividida da seguinte forma:

- **Testes de Modelo (Models):**
  - **`Notification`:** Testa as validações, os escopos (especialmente `pending_to_send`, validando sua lógica de tempo) e os callbacks, garantindo que são chamados corretamente.
  - **`NotificationTemplate`** / **`NotificationRule`:** Testam as associações e validações básicas.
- **Testes de Callback (Models):**
  - Testes de integração de baixo nível (ex: em `spec/models/document_spec.rb`) que usam o *matcher* `have_enqueued_job(Notifications::CreateJob)`.
  - Esses testes confirmam que, ao criar ou atualizar um modelo (ex: `create(:document)`), o callback `after_commit` é disparado corretamente e enfileira o `CreateJob`.
  - "Dublês"(Stubs) são usados para isolar o callback testado de outros callbacks (ex: `allow(document).to receive(:create_signatures)`) para evitar "ruído"nos testes.
- **Testes de Job (Jobs):**
  - Utilizam o `ActiveJob::TestHelper` e seus *matchers* (ex: `have_enqueued_job`).
  - **`CreateJob`:** Testa se o método `perform` chama corretamente o `Notifications::SchedulerService` com os argumentos que recebeu.

  - **SchedulerPollerJob:** Testa se o `perform` consulta o escopo `Notification.pending_to_send` (que é "dublado") e chama o `Notifications::ProcessorService` para cada notificação retornada.
  - **DispatchJob:** Testa se o `perform` encontra a notificação, chama `Notifications::Dispatcher.new(notification).call` e relevanta exceções para acionar os *retries* do Solid Queue.

- **Testes de Serviço (Services):**
  - **SchedulerService:** Testa todos os caminhos lógicos: criação de notificações `pending` (para agora), `scheduled` (com regras `days_before` ou `hours_after`), e a lógica de idempotência (não sobrescrever uma notificação `sent` ou `failed`).
  - **ProcessorService:** O teste mais complexo. "Dubla"(stubs) o `StopChecker` e usa `have_enqueued_job` para verificar a lógica:
    * Testa se os *guards* (verificações iniciais) funcionam (ex: `return if notification.sent?`).
    * Testa se o job é cancelado se `StopChecker.met?` for `true`.
    * Testa se o job falha se `attempts >= max_attempts`.
    * Testa a lógica de "envio único"(marca como `sent` após enfileirar).
    * Testa a lógica "insistente"(marca como `scheduled` para o futuro).
  - **Dispatcher:** "Dubla"(stubs) o `NotificationMailer` e verifica se `deliver_later` é chamado para o canal `email`.

- **Testes de Mailer (Mailers):**
  - Em `spec/mailers/notification_mailer_spec.rb`, testa o método `generic_email`.
  - Verifica se os cabeçalhos (`to`, `subject`, `from`) estão corretos.
  - Verifica se a interpolação das variáveis (ex: `%<academic_name>`) está funcionando no assunto e no corpo.
  - Verifica se o fallback para chaves ausentes (`Hash.new { ... }`) impede o `KeyError`.
  - Verifica se ambas as partes, `text/plain` (com `<br>` convertido para `n`) e `text/html` (com `<br>`), são geradas.

Esta estratégia de testes de unidade granulares permite que o sistema seja refatorado com segurança e que qualquer falha no fluxo de notificação possa ser identificada e corrigida.

4.3 Resultados obtidos

A implementação do sistema de notificações no SGTCC resultou em uma arquitetura isolada do fluxo principal da aplicação, permitindo o envio automatizado de comunicações aos usuários de forma assíncrona e confiável. Com a inclusão do *framework* `Active Job` e do adaptador `Solid Queue`, foi possível estabelecer um fluxo de processamento independente do servidor web, garantindo maior desempenho, escalabilidade e tolerância a falhas durante a execução das tarefas de notificação.

Embora na metodologia proposta inicialmente previsse a divisão dos requisitos em tarefas menores e independentes, acompanhadas de revisões contínuas pela equipe por meio do *GitFlow* e da gestão no ClickUp, essa abordagem não pôde ser integralmente seguida durante o desenvolvimento. A principal dificuldade esteve relacionada à incompatibilidade de horários entre o autor e os orientadores, o que dificultou a realização de revisões frequentes, reuniões de alinhamento e validações incrementais ao longo do processo.

Diante desse cenário, a estratégia de desenvolvimento precisou ser adaptada. Em vez de uma divisão refinada em pequenas tarefas com ciclos constantes de revisão, o trabalho foi conduzido de maneira mais concentrada: os requisitos previamente levantados foram analisados e implementados de forma contínua, com validações realizadas à medida que a funcionalidade completas foi concluída. Essa mudança permitiu manter o progresso do projeto apesar das limitações de comunicação.

Assim, o desenvolvimento acabou ocorrendo em ciclos mais longos e com maior autonomia por parte do autor, priorizando a entrega funcional da implementação no sistema para posterior avaliação conjunta. Essa adaptação do processo metodológico não impediu o cumprimento dos objetivos definidos, mas representou uma diferença relevante entre o planejamento inicial e a execução prática do trabalho.

O principal resultado obtido foi a criação de um módulo de notificações completamente desacoplado da lógica principal do sistema, operando de maneira autônoma sobre um banco de dados secundário (`SQLite`). Essa separação eliminou os problemas de bloqueio (*database locking*) anteriormente observados durante operações simultâneas de leitura e escrita, além de permitir que o processamento de mensagens e o envio de e-mails ocorressem sem impacto perceptível no desempenho do SGTCC.

As funcionalidades implementadas permitem que o sistema:

- Gere notificações automáticas baseadas em eventos do sistema, como criação de documentos, prazos de envio e agendamento de bancas;
- Agende notificações futuras conforme regras configuráveis de antecedência e reenvio;
- Interrompa notificações de lembrete de forma automática, quando a ação esperada (como uma assinatura pendente) é concluída;

- Enfileire e processe os envios em segundo plano, evitando travamentos e sobrecarga do servidor principal;

Além disso, a integração dos *hooks* aos modelos principais da aplicação tornou o processo de geração de notificações transparente e automatizado. A criação ou atualização de registros relevantes (como documentos, prazos ou bancas) passa a acionar imediatamente o enfileiramento das notificações correspondentes, sem a necessidade de intervenção manual.

A partir dos testes realizados, foi possível confirmar que o sistema consegue lidar com múltiplas notificações simultâneas, mantendo consistência nos registros e enviando os e-mails corretamente de acordo com as regras estabelecidas. O comportamento esperado foi validado para diferentes cenários — como notificações insistentes, notificações únicas e notificações com múltiplos destinatários —, demonstrando a robustez e a previsibilidade do mecanismo de agendamento e envio.

Resumidamente, o desenvolvimento resultou em uma melhoria na capacidade do SGTCC de se comunicar com seus usuários, automatizando processos que antes eram manuais e centralizando a gestão de notificações. A arquitetura construída possibilita, ainda, a expansão para novos tipos de aviso e a inclusão futura de outros canais de comunicação, como mensagens instantâneas, sem necessidade de reestruturação do sistema existente.

4.3.1 Métricas

O processo de desenvolvimento das notificações foi desenvolvido em 17 *commits* criando 1 PR (*Pull Request*), modificando 80 arquivos, adicionando 2832 linhas e removendo 49.

A ampliação dos testes automatizados foi diretamente relacionada às funcionalidades de notificações desenvolvidas neste trabalho. Antes das implementações, o sistema possuía 812 testes automatizados, distribuídos entre modelos, serviços e controladores já existentes. O conjunto de testes criado para validar o novo módulo de notificações adicionou mais 45 testes, totalizando 857 testes ao final da implementação.

Os testes produzidos abrangem geração de notificações, callbacks de modelos, agendamento, despacho, regras de repetição, envio de e-mails, verificação de *payloads*, interpretação de templates, falhas de interpolação, execução de *jobs* assíncronos e fluxos de processamento periódico. Esses cenários representam 100% dos fluxos introduzidos por este trabalho. Em termos de verificações (assertivas), o número total passou de 974 para 1033, um acréscimo de 59 verificações decorrentes da validação dos comportamentos esperados.

Em resumo, este trabalho contribuiu diretamente com:

- **45 novos testes** especificamente relacionados ao sistema de notificações;
- **59 novas verificações**;
- validação completa de todos os fluxos síncronos e assíncronos introduzidos.

Essas métricas demonstram que as funcionalidades desenvolvidas foram incorporadas ao sistema com garantia de qualidade, rastreabilidade e comportamento validado, refletindo diretamente nos objetivos deste trabalho.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo principal o desenvolvimento e a implementação de uma estratégia de notificações automáticas para o sistema SGTCC, considerando as reais necessidades dos usuários envolvidos nos processos de TCC. A partir do levantamento de requisitos com discentes e docentes, foi possível identificar os principais pontos de falha na comunicação sobre prazos, eventos e pendências, que frequentemente resultavam em atrasos ou retrabalho.

Com base nessas informações, foi projetada e implementada uma arquitetura de notificações integrada ao sistema existente, com foco na clareza, na confiabilidade e na pontualidade das mensagens. A solução proposta tem como fundamento o uso do *framework* `Active Job` e do adaptador `Solid Queue`, que possibilitaram o processamento assíncrono de tarefas, garantindo o envio das notificações de forma eficiente, sem comprometer o desempenho da aplicação principal.

Com a implementação do módulo de notificações esperam-se resultados significativos para o SGTCC ao decorrer do tempo. O sistema passou a ser capaz de gerar e enviar notificações automáticas de acordo com eventos internos, como a criação de documentos, prazos de entrega e agendamento de bancas, além de permitir agendamentos futuros e reenvios baseados em regras configuráveis. A arquitetura adotada também eliminou problemas de concorrência e bloqueio de banco de dados, assegurando maior estabilidade e fluidez nas operações.

Do ponto de vista técnico, a principal contribuição deste trabalho está na automação e na comunicação entre os usuários do sistema. Antes da implementação, o acompanhamento de prazos e pendências dependia majoritariamente de ações manuais, o que resultava em esquecimentos e atrasos. Com a nova estrutura, docentes, discentes e demais envolvidos passam a receber lembretes e avisos automáticos, reduzindo a probabilidade de falhas humanas e melhorando a organização e a transparência no acompanhamento das etapas do TCC.

A utilização de *Service Objects*, *Jobs* e *Hooks* contribuiu para uma arquitetura mais modular, organizada e aderente aos princípios de boas práticas de engenharia de software, como o Princípio da Responsabilidade Única (SRP)[1] e a separação de preocupações. Essa abordagem tornou o código mais legível, testável e preparado para expansões futuras, além de facilitar a manutenção e o trabalho colaborativo dentro da equipe de desenvolvimento do SGTCC.

Entre as limitações identificadas, destaca-se que a versão atual do sistema realiza notificações exclusivamente por e-mail. Apesar de a arquitetura desenvolvida permitir a inclusão de outros canais, como notificações via WhatsApp ou alertas internos na própria plataforma, essas funcionalidades ainda não foram implementadas. A estrutura atual, baseada em serviços e *jobs* desacoplados, permite que novos canais sejam adicionados sem alterar o fluxo existente: bastaria criar novos serviços responsáveis pelo envio via API do WhatsApp ou pelo disparo de

[1] Definido por Robert C. Martin em *Agile Software Development: Principles, Patterns, and Practices*.

notificações internas, reutilizando o mesmo mecanismo de agendamento, enfileiramento e reenvio. A incorporação desses dois novos canais poderia ampliar o alcance das notificações e aumentar a taxa de engajamento dos usuários, pois a preferência de meios de comunicação difere entre os diferentes tipos de usuários.

Outra limitação diz respeito ao uso do banco de filas baseado em `SQLite`, que, embora leve e eficiente, pode não ser o mais adequado para cenários de produção com alta demanda. No entanto, esse não é o caso do SGTCC, cujo volume de uso é moderado e controlado. Dessa forma, o `SQLite` se apresenta como uma solução ideal para o contexto institucional atual, oferecendo simplicidade, baixo custo operacional e desempenho suficiente para o volume de tarefas processadas pelo sistema.

Como trabalhos a complementar o desenvolvimento dessa funcionalidade, propõe-se a ampliação do sistema de notificações com a integração de novos canais de comunicação, especialmente via API do WhatsApp e notificações em tempo real dentro da própria aplicação.

Concluindo, o sistema de notificações desenvolvido representa um avanço significativo na comunicação e na eficiência operacional do SGTCC. A solução automatiza processos essenciais, reduz erros decorrentes de falha humana e estabelece uma base sólida para futuras melhorias. Dessa forma, contribui diretamente para a modernização do sistema e para a evolução contínua da gestão acadêmica na instituição.

REFERÊNCIAS

ATLASSIAN. **Gitflow Workflow**. 2025. Acesso em: 4 jun. 2025. Disponível em: https://www.atlassian.com/br/git/tutorials/comparing-workflows/gitflow-workflow.

CLICKUP. **ClickUp Documentation**. 2025. Documentação oficial da plataforma de gerenciamento de projetos. Disponível em: https://help.clickup.com. Acesso em: 05 jun. 2025.

COINT. **Normas Operacionais Complementares do Trabalho de Conclusão de Curso do Curso Superior de Tecnologia em Sistemas para Internet - Câmpus Guarapuava**. 2023. Disponível em: https://tcc.tsi.pro.br/uploads/attached_document/file/2/normas-operacionais-complementares-do-TCC-TSI-GP-2023-1.pdf. Acesso em: 23 abr. 2025.

DOCKER. **Docker Documentation**. 2025. Documentação oficial. Disponível em: https://docs.docker.com. Acesso em: 06 jun. 2025.

FERREIRA Érico 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 Campus Guarapuava.** 2015. Monografia (TCC) — Universidade Tecnológica Federal do Paraná, 2015.

FIGMA. **Figma – Design, Prototype, and Collaborate All in the Browser**. 2025. Documentação oficial. Disponível em: https://www.figma.com. Acesso em: 06 jun. 2025.

GIT. **Git – Distributed Version Control System**. 2025. Documentação oficial. Disponível em: https://git-scm.com. Acesso em: 05 jun. 2025.

GITHUB. **GitHub Docs**. 2025. Documentação oficial. Disponível em: https://docs.github.com. Acesso em: 05 jun. 2025.

GitLab Inc. **O que é CI/CD?** 2025. Acesso em: 4 jun. 2025. Disponível em: https://about.gitlab.com/pt-br/topics/ci-cd/.

LIMA, A. C. D. **Projeto e implementação de interface baseada na experiêcia do usuário para um sistema de gerenciamento de trabalho de conclusão de curso**. 2023. Monografia (TCC) — Universidade Tecnológica Federal do Paraná, 2023.

LUZ, G. S. da. **Atuallização do Framework Rails para garantia de evolução do Sistema de Gestão de TCC**. 2024. Monografia (Projeto de TCC) — Universidade Tecnológica Federal do Paraná, 2024.

RAILS, R. on. **Ruby on Rails Guides**. 2025. Documentação oficial. Disponível em: https://guides.rubyonrails.org. Acesso em: 05 jun. 2025.

SEBRAE. **Metodologia MoSCoW: Como priorizar requisitos de projetos de forma estratégica**. 2022. Acessado em: 12 de junho de 2025. Disponível em: https://sebrae.com.br/Sebrae/Portal%20Sebrae/Arquivos/ebook_sebrae_metodologia_moscow.pdf.

SILVA, R. G. A. **Aperfeiçoamento do Sistema de Gestão de processos de Trabalho de Conclusão de curso de Tecnologia em Sistemas para Internet Da UTFPR Campus Guarapuava**. 2019. Monografia (TCC) — Universidade Tecnológica Federal do Paraná, 2019.