UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GUILHERME STACIAKI DA LUZ

ATUALIZAÇÃO DO FRAMEWORK RAILS PARA GARANTIA DA EVOLUÇÃO DO SISTEMA DE GESTÃO DE TCC

GUARAPUAVA

2025

GUILHERME STACIAKI DA LUZ

ATUALIZAÇÃO DO FRAMEWORK RAILS PARA GARANTIA DA EVOLUÇÃO DO SISTEMA DE GESTÃO DE TCC

Rails Framework Update to Ensure the Evolution of the Thesis Management System

Projeto de 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.

Orientadora: Profª Drª Renata Luiza Stange

Coorientador: Prof. Dr. Diego Marczal

GUARAPUAVA

2025

4.0 Internacional

Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

DOM	Document Object Model
SGTCC	Sistema de Gestão de Trabalho de Conclusão de Curso
TCC	Trabalho de Conclusão de Curso
TCC 1	Trabalho de Conclusão de Curso 1
TCC 2	Trabalho de Conclusão de Curso 2
TSI	Tecnologia em Sistemas para Internet
UTFPR	Universidade Tecnológica Federal do Paraná
UX	User Experience

SUMÁRIO

1 INTRODUÇÃO

O TCC é uma atividade acadêmica essencial na finalização de cursos de graduação, no qual o acadêmico aplica seus conhecimentos adquiridos durante sua formação em torno de um tema específico. No curso de TSI da UTFPR, o TCC envolve o desenvolvimento de soluções tecnológicas, integrando pesquisa e prática (COINT, 2023).

Para otimizar o processo de gerenciamento do TCC, foi desenvolvido o SGTCC[1], uma plataforma que facilita a entrega, correção e avaliação dos trabalhos, além de reduzir a burocracia e a necessidade de documentação impressa. O SGTCC tem evoluído para atender às demandas acadêmicas e administrativas de forma mais eficiente, automatizando etapas que antes eram realizadas manualmente, como o agendamento de bancas e a assinatura de termos de compromisso. No entanto, com a rápida evolução das tecnologias web e as crescentes preocupações com a segurança e a eficiência dos sistemas, é necessário evoluir e aprimorar esses sistemas para garantir a qualidade do software.

Neste contexto, é fundamental considerar as Leis de Lehman, que afirmam que os sistemas de software em uso contínuo precisam passar por manutenção e evolução para permanecerem úteis e relevantes (LEHMAN; BELADY, 1985). Dessa forma, a necessidade de manter o SGTCC atualizado e seguro, prevenindo falhas e melhorando a qualidade do sistema, torna-se evidente. Isso não apenas se alinha com as Leis de Lehman, que destacam a inevitável evolução de sistemas de software, mas também segue as melhores práticas aplicadas no desenvolvimento de software (FOWLER, 2018). Tais práticas são essenciais para garantir a longevidade e a eficácia do sistema em um ambiente tecnológico em constante mudança.

Dessa forma, este trabalho propõe a atualização das bibliotecas e dependências do SGTCC, bem como a adequação do código para garantir a continuidade da evolução e aprimoramento do sistema. Essas ações visam não apenas corrigir possíveis vulnerabilidades, mas também melhorar a performance e compatibilidade com novas tecnologias, assegurando a capacidade de evolução do software de maneira eficaz, beneficiando tanto os usuários quanto os desenvolvedores.

1.1 Objetivos

Nesta seção, serão pontuados os objetivos do trabalho de conclusão de curso.

1.1.1 Objetivo Geral

Atualizar o framework Ruby on Rails no SGTCC, eliminando dependências descontinuadas, para garantir que o sistema continue a receber melhorias e facilite a manutenção.

[1] Disponível em https://tcc.tsi.pro.br/o-tcc. Acessado em 30 de outubro de 2024.

1.1.2 Objetivos específicos

Os objetivos específicos são listados a seguir:

1. Atualizar bibliotecas e dependências: para garantir a segurança e a continuidade do sistema por meio da atualização de todas as bibliotecas desatualizadas e dependências utilizadas, prevenindo vulnerabilidades e melhorando o desempenho.

2. Reescrever o JavaScript utilizando Hotwire: a estrutura atual da interface do sistema depende do Webpacker. Com a descontinuação dessa biblioteca, é necessário reescrever o código JavaScript, adotando uma solução mais moderna, como o Hotwire, que dispensa o uso da biblioteca descontinuada.

3. Refatorar o código: refatoração o código existente, para melhorar sua qualidade, modularidade e facilidade de manutenção, utilizando os princípios e padrões de projeto.

1.2 Justificativa

No campo da Engenharia de Software, o presente trabalho se enquadra na categoria de manutenção de software, compreendendo três principais tipos de manutenção: preventiva, corretiva e perfectiva. A manutenção preventiva refere-se à atualização de tecnologias com o intuito de garantir a qualidade, principalmente nos aspectos relacionados à segurança e disponibilidade. Segundo Sommerville (2011), a refatoração é uma forma de manutenção preventiva, pois visa melhorar a estrutura do software, reduzindo sua complexidade e tornando-o mais compreensível. Já a manutenção corretiva envolve a correção de erros que podem surgir com a atualização, assegurando o funcionamento correto das aplicações. Por último, a manutenção perfectiva considera melhorias contínuas no código, como a refatoração e a aplicação de princípios de design de software, como o SOLID, para melhorar a legibilidade, escalabilidade e eficiência do sistema.

A ausência de atualizações, isto é, de uma manutenção preventiva, em um software pode resultar em diversos problemas significativos, como um aumento na exposição a ciberataques, uma vez que falhas de segurança conhecidas não são corrigidas. Além disso, o desempenho do sistema tende a piorar, devido a incompatibilidades com novas tecnologias e atualizações de infraestrutura, o que pode causar lentidão e falhas frequentes. Por fim, a manutenção do sistema também se torna mais complexa e onerosa, já que a resolução de problemas em um software desatualizado demanda mais tempo e recursos, além de aumentar o risco de interrupções nos serviços (SOMMERVILLE, 2011).

Considerando que o SGTCC lida com informações críticas, que envolvem todo o processo de gestão dos trabalhos de conclusão de curso, a falta de atualizações pode resultar em riscos de segurança dos dados e falhas na continuidade do serviço. Além disso, um código não

otimizado dificulta a manutenção e a escalabilidade do sistema, impactando negativamente a experiência dos usuários.

2 O SGTCC

O SGTCC começou seu desenvolvimento em 2015, com o intuito de transformar a gestão das atividades de TCC do curso de TSI em um processo digital, visando simplificar todo o processo e centralizar as informações e regulamentos em um único sistema (FERREIRA, 2015). Este sistema serviu de base para o desenvolvimento do sistema que é utilizado atualmente para a gestão dos trabalhos de conclusão de curso de TSI, o SGTCC.

Posteriormente, em 2019, foi dado continuidade ao projeto com a adequação do sistema ao processo do TCC de TSI, junto com a incorporação de assinatura eletrônica em documentos, removendo o uso de papel em todo o processo, tornando digital toda a gestão do processo. Também foram aplicadas diversas melhorias nos módulos existentes do sistema, tais como a criação de tipos de usuários, agendamento de defesas com documentos relacionados ao TCC e avaliações. Também foram feitas mudanças na área pública com informações mais relevantes, como trabalhos realizados e histórico de orientações, sendo adicionado também estatísticas para o professor responsável (SILVA, 2019).

Outras contribuições para o projeto, foram realizadas no segundo semestre do ano letivo de 2023, onde os alunos da disciplina de Desenvolvimento para Web 5 do curso de TSI aplicaram alterações em diversas partes do sistema, como a criação de novas funcionalidades, correção de *bugs*, atualizações de bibliotecas e de documentações do projeto. Ao todo, foram 2739 linhas de código adicionadas e 1012 removidas, somando um total de 1623 *commits* e 238 PRs[1].

Ainda em 2023, o sistema recebeu outras melhorias para o seu funcionamento, com destaque na otimização de suas telas. Foram aplicadas técnicas de UX Design focadas na estética e funcionalidade do sistema, tendo foco na revisão da interface gráfica a fim de torná-la mais agradável e uma reorganização de elementos de design para aprimorar a usabilidade e a eficiência do mesmo (LIMA, 2023).

Atualmente o sistema conta com as seguintes áreas:

- Área pública: composta por uma seção disponível para qualquer pessoa na internet, sem a necessidade de autenticação. Nela contém uma breve descrição do termo do TCC e seus objetivos gerais, além de outras informações como bancas de TCC do período corrente, calendário com as atividades necessárias e a listagem de TCCs aprovados.

[1] Uma pull request (PR) é uma proposta para mesclar as alterações de um branch em outro. Em uma pull request, os colaboradores podem revisar e discutir o conjunto de alterações proposto antes de integrá-las à base de código principal. As pull requests exibem as diferenças, ou comparações, entre o conteúdo no branch de origem e aquele no branch de destino. Disponível em https://docs.github.com/pt/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests?platform=linux. Acesso em 05 de novembro de 2024.

- Área de membro externo: esta área é disponibilizada para convidados e instituições externas terem acesso às bancas de defesa, das quais fazem parte, e as informações sobre estas, tais como acesso a documentos pendentes como aos documentos já assinados.

- Área acadêmica: nesta área é possível acompanhar todo o avanço e desempenho do discente referente ao desenvolvimento do TCC. Neste módulo, encontram-se as atividades essenciais para o andamento do TCC, juntamente com os documentos relacionados à orientação, como os pendentes de assinatura e os já assinados. Além disso, há informações sobre a banca de defesa, incluindo o local, a data e o horário da apresentação.

- Área do orientador: contém as informações das principais atividades para a continuidade do TCC, como o monitoramento das datas de entregas de cada etapa realizada pelo acadêmico. Ademais, apresenta-se uma seção específica para reuniões, onde é possível registrar informações que ficam disponíveis para o aluno, além de permitir o acesso às informações referentes às bancas das quais o professor orientador é membro avaliador.

- Área do professor da disciplina de TCC 1: nesta área o professor da disciplina de TCC 1 tem acesso a todos os discentes matriculados, podendo agendar bancas de defesa de propostas e projetos, acompanhar as entregas feitas por cada estudantes na disciplina e verificar prazos relacionados ao calendário em andamento.

- Área do responsável pelo TCC: aqui encontra-se o maior número de funcionalidades, uma vez que é possível gerenciar o andamento de processos relacionados ao TCC 1 e TCC 2. A área permite fazer o cadastramento de professores orientadores, acadêmicos, professor de TCC 1 , membros externos e outros professores responsáveis pela administração do sistema. Também é possível definir o calendário de um semestre e cadastrar novas atividades que integrariam as matérias de TCC 1 e TCC 2. No sistema, o professor responsável tem acesso a uma seção para administrar as bancas de TCC, podendo visualizar e agendar bancas, selecionando o estudante e os professores que avaliarão o trabalho, além de definir a data e o tipo da banca também.

Atualmente, estão em andamento outros projetos que envolvem melhorias no SGTCC, um deles propõe melhorias no código do sistema e o outro melhorias na interface gráfica e funcionalidades.

3 MATERIAIS E MÉTODOS

Para atualizar o sistema e alinhá-lo aos padrões atuais, é necessário um conjunto de ferramentas e práticas que facilitem o processo de atualização. Este capítulo apresenta os recursos utilizados e a metodologia adotada para garantir uma transição eficiente e segura.

3.1 Ferramentas

A atualização do sistema será conduzida com o suporte de diversas ferramentas, abrangendo desde o planejamento até a implementação e o gerenciamento do código-fonte. Essas tecnologias foram selecionadas para otimizar o tempo, melhorar a colaboração e garantir a estabilidade do projeto.

- **Ruby On Rails**: *Framework web* completo e robusto que acelera o desenvolvimento de aplicações web utilizando a linguagem Ruby. Ele facilita a criação de código estruturado, escalável e de fácil manutenção, seguindo convenções que reduzem a necessidade de configuração e promovem boas práticas de desenvolvimento (RAILS, 2025a).
- **Docker**: Plataforma de virtualização leve que permite criar, empacotar e executar aplicações em containers. Esses containers garantem a execução consistente e isolada do software, independentemente do ambiente. Com isso, o Docker facilita a implantação, escalabilidade e portabilidade das aplicações (DOCKER, 2025).
- **ClickUp**: Plataforma de gerenciamento de projetos que centraliza tarefas, documentos e comunicação. Flexível e personalizável, ajuda equipes a organizar fluxos de trabalho, acompanhar projetos e automatizar processos (CLICKUP, 2025).
- **Git**: Sistema de controle de versão distribuído que permite rastrear alterações no código-fonte, facilitando a colaboração entre desenvolvedores e garantindo a integridade do histórico de desenvolvimento (GIT, 2025).
- **Github**: Plataforma baseada em nuvem que utiliza Git para hospedar repositórios, permitindo o versionamento de código, colaboração em equipe e automação de fluxos de trabalho (GITHUB, 2025).
- **Postgresql**: Banco de dados relacional open-source conhecido por sua estabilidade, segurança e desempenho avançado. Suporta consultas complexas, extensibilidade e transações robustas, sendo ideal para aplicações escaláveis (POSTGRESQL, 2025).
- **Hotwire**: Ferramenta para desenvolvimento web rápido que minimiza o uso de JavaScript, permitindo atualizações dinâmicas na interface diretamente do servidor. Ele melhora a experiência do usuário sem comprometer o desempenho (HOTWIRE, 2025).

3.2 Processo de atualização

A atualização de sistemas desenvolvidos em Ruby on Rails requer um planejamento cuidadoso para garantir compatibilidade, estabilidade e segurança. A equipe mantenedora do framework disponibiliza um guia oficial para auxiliar nesse processo, mas recomenda-se seguir boas práticas adicionais para minimizar impactos na aplicação (RAILS, 2025b).

3.2.1 Recomendações Gerais para Atualização do Rails

Uma das principais recomendações para garantir uma transição segura é possuir uma cobertura robusta de testes automatizados. Esses testes asseguram que a aplicação continua funcionando corretamente após cada etapa do processo. Caso a cobertura seja insuficiente, será necessário realizar verificações manuais em todas as funcionalidades impactadas.

O Rails exige versões mínimas específicas do Ruby para cada release. Por isso, recomenda-se atualizar primeiro o Ruby para a versão mais recente suportada antes de prosseguir com a atualização do framework.

O processo de atualização deve ocorrer de forma incremental, seguindo as etapas abaixo:

1. Certificar-se de que todos os testes estão passando na versão atual do Rails.
2. Atualizar para a versão mais recente dentro da mesma versão principal, resolvendo avisos de depreciação.
3. Avançar para a versão mais recente da próxima versão secundária, aplicando os ajustes necessários.
4. Repetir esse processo até atingir a versão desejada.

O Rails fornece um comando para auxiliar na migração: `bin/rails app:update`. Esse comando sugere modificações nos arquivos do projeto para torná-los compatíveis com a nova versão. Após executá-lo, é essencial revisar as mudanças aplicadas e resolver possíveis conflitos. Além disso, o Rails cria um arquivo `config/initializers/new_framework_defaults_X_Y.rb` para permitir a ativação gradual de novas configurações. Após validar a atualização, esse arquivo pode ser removido e a configuração `config.load_defaults` deve ser ajustada para refletir a versão utilizada.

3.2.2 Recomendações para Atualização do Rails 6 para 7

A atualização do Rails da versão 6 para 7 introduz mudanças significativas, incluindo melhorias de desempenho, novos padrões de convenção e atualizações de segurança. Para garantir uma transição segura e eficiente, recomenda-se seguir um processo estruturado.

1. Preparação para a atualização: antes de iniciar a migração, é essencial garantir que a aplicação esteja estável na versão mais recente do Rails 6. Isso inclui:

    a) Certificar-se de que todos os testes automatizados estão passando e que há cobertura suficiente para validar as principais funcionalidades.

    b) Atualizar o Ruby para a versão mínima exigida pelo Rails 7.

    c) Resolver todos os avisos de depreciação identificados na versão 6.

2. Etapas para atualização: após garantir a estabilidade da aplicação na versão 6, a atualização pode ser realizada de forma gradual:

    a) Atualizar a gem `rails` no arquivo `Gemfile` para a versão 7 e executar:

    ```
    bundle update rails
    bin/rails app:update
    ```

    b) Revisar as alterações sugeridas pelo comando `bin/rails app:update` e ajustá-las conforme necessário.

    c) Ajustar arquivos de configuração, incluindo `config/application.rb` e `config/initializers/new_framework_defaults_7_0.rb`, ativando as novas configurações gradualmente.

    d) Testar a aplicação para identificar possíveis regressões e corrigir eventuais problemas.

    e) Atualizar dependências e gems para garantir compatibilidade com a nova versão.

3. Principais Mudanças na Versão 7: essa versão do Rails introduz várias melhorias importantes, entre elas:

    a) Introdução do Hotwire, que permite maior interatividade sem necessidade de JavaScript excessivo.

    b) Uso nativo do `zeitwerk` como carregador de código padrão.

    c) Novas estratégias de gerenciamento de conexões com o banco de dados.

d) Melhorias no Active Record, incluindo novas otimizações de cache.

4. Finalização da Atualização: após validar todas as mudanças e garantir o funcionamento correto da aplicação, recomenda-se:

    a) Remover o arquivo `new_framework_defaults_7_0.rb`.

    b) Atualizar o `config.load_defaults` para a nova versão.

Seguindo essas diretrizes, a migração do Rails 6 para 7 pode ser realizada de forma segura, garantindo a estabilidade e o desempenho da aplicação.

3.2.3 Recomendações para Atualização do Rails 7 para 8

O Rails não disponibiliza um guia específico para a atualização entre versões, mas é importante estar atento às principais mudanças introduzidas no Rails 8.0 para garantir uma transição segura. Antes de aplicar a atualização, recomenda-se seguir os passos gerais: garantir uma boa cobertura de testes, revisar avisos de depreciação na versão 7 e verificar a compatibilidade das dependências.

1. **Remoção de funcionalidades obsoletas**: Algumas configurações e métodos foram eliminados no Rails 8.0, exigindo ajustes no código. Entre elas:

    - `config.read_encrypted_secrets` foi removido do Railties, uma vez que o uso de credenciais criptografadas passou a ser totalmente gerenciado pelo sistema de `config.credentials`.
    - O argumento `model: nil` em `form_with` deixou de ser suportado, exigindo que formulários utilizem um modelo explícito ou forneçam `url` diretamente.
    - Configurações depreciadas no Active Record foram eliminadas, incluindo:
        - `config.active_record.commit_transaction_on_non_local_return`, que alterava o comportamento de transações, mas agora foi removido para evitar comportamento inesperado.
        - `config.active_record.allow_deprecated_singular_associations_name`, usado para permitir nomes singulares em associações, foi eliminado para incentivar práticas mais consistentes.

2. **Aprimoramentos de segurança**:

    - Foi introduzido um tempo limite padrão de 1 segundo para `Regexp.timeout`, reduzindo vulnerabilidades a ataques de negação de serviço (ReDoS) causados por expressões regulares malformadas.

- Agora, `config.active_record.default_timezone` aceita a opção `:local`, permitindo que as operações de banco de dados utilizem o fuso horário local do sistema em vez do UTC.

3. **Mudanças na inicialização e configuração**:

   - O arquivo `config/application.rb` foi atualizado para simplificar a inicialização da aplicação, tornando-a mais modular e alinhada com as melhores práticas.
   - O suporte a `config.load_defaults 7.0` foi removido, tornando obrigatório definir as configurações da versão 8.0 explicitamente.

4. **Melhorias no Active Record**:

   - O método `pluck` agora pode ser usado diretamente em relações associadas, simplificando consultas a atributos específicos.
   - Foi adicionado suporte para ordenação por múltiplas colunas usando `order(:column1, :column2)` sem precisar especificar direções individualmente.

5. **Etapas recomendadas para a atualização**:

   - **Revisão de dependências**: Certifique-se de que todas as gems e bibliotecas utilizadas no projeto são compatíveis com a nova versão.
   - **Execução do comando de atualização**: Rodar `bin/rails app:update` para ajustar arquivos de configuração conforme necessário.
   - **Testes e validação**: Após a atualização, execute testes automatizados e manuais para garantir o funcionamento correto da aplicação.
   - **Monitoramento em produção**: Após a implantação, acompanhe logs e métricas para detectar possíveis problemas e agir rapidamente.

4 RESULTADOS PRELIMINARES

Neste capítulo, apresentam-se os resultados preliminares do levantamento e análise das necessidades de atualização do SGTCC, visando garantir a segurança, compatibilidade e desempenho do sistema. A análise inicial focou na identificação das bibliotecas essenciais para o funcionamento do sistema, categorizando-as conforme sua aplicação em ambiente de produção, desenvolvimento e testes. Além disso, discute-se a descontinuação do Webpacker e suas implicações para a integração com frameworks modernos, como Vue.js.

Nos tópicos a seguir, detalha-se o levantamento inicial das bibliotecas utilizadas, a comparação entre versões e as adaptações necessárias para garantir a continuidade e eficiência do sistema.

4.1 Levantamento inicial da necessidade de atualização

Desde o seu início, o SGTCC passou por diversas mudanças e aprimoramentos, tornando-se um sistema mais robusto e eficaz em seu objetivo principal de gerenciar o processo de conclusão de curso de maneira eficiente. No entanto, para garantir a continuidade dessa evolução e a incorporação de melhorias no sistema, é importante que ele seja mantido atualizado com as tecnologias mais recentes. Isso não só assegura a compatibilidade do sistema com novos padrões, mas também promove a segurança e o desempenho.

As tabelas apresentam a versão atual de cada biblioteca e a versão necessária para a atualização. A Tabela 1 lista as bibliotecas essenciais para o ambiente de produção. A Tabela 2 descreve as bibliotecas voltadas ao desenvolvimento, como ferramentas de depuração e análise de código. A Tabela 3 aborda as bibliotecas utilizadas nos testes, que são fundamentais para garantir a qualidade e a confiabilidade do sistema. Por fim, a Tabela 4 detalha as bibliotecas empregadas tanto no desenvolvimento quanto nos testes, comparando as versões atuais com as mais recentes.

A versão 7 do Rails trouxe mudanças significativas, destacando a remoção do Webpacker[1], que foi descontinuado por seus mantenedores (RAILS, 2004). Como resultado, o Webpacker se tornou obsoleto e foi substituído por soluções mais modernas para gerenciamento de JavaScript. Essas mudanças impactam diretamente o funcionamento da biblioteca Vue.js[2], uma vez que sua integração com o Rails sempre dependia do Webpacker. Com a remoção do

[1] Webpacker é um wrapper Rails que utiliza o sistema de build Webpack, oferecendo uma configuração padrão e boas práticas. O Webpack é um empacotador de módulos estáticos para aplicações JavaScript modernas. Disponível em https://guiarails.com.br/webpacker.html. Acessado em 13 de setembro de 2024.

[2] Vue.js é um Framework JavaScript para construção de interfaces de usuário. Baseado em HTML, CSS e JavaScript padrão, ele oferece um modelo de programação declarativo e baseado em componentes, facilitando o desenvolvimento de interfaces de qualquer complexidade. Disponível em https://vuejs.org/guide/introduction.html. Acessado em 24 de outubro de 2024.

Biblioteca	Versão Atual	Última Versão
bootsnap	1.17.0	1.18.4
puma	5.6.8	6.4.3
rails	6.1.7.8	7.2.1.1
uglifier	4.2.0	4.2.1
active_link_to	1.3.0	1.3.0
bootstrap	4.6.2.1	5.3.3
bootstrap4-datetime-picker-rails	4.6.2.1	5.3.3
breadcrumbs_on_rails	5.0.0	6.0.0
bundle-audit	0.9.1	0.9.2
carrierwave	2.2.6	3.0.7
carrierwave-i18n	0.3.0	3.0.0
devise	4.9.3	4.9.4
devise-i18n	1.12.0	1.12.1
exception_notification	4.0.0	5.0.0
font-awesome-sass	6.5.1	6.5.2
jquery-rails	4.4.0	5.0.0
kaminari	1.2.0	1.3.0
mini_magick	4.12.0	5.0.1
momentjs-rails	2.0.0	3.0.0
pg	1.5.4	1.5.8
rails-i18n	7.0.8	7.0.9
reek	6.1.4	6.3.0
simple_form	5.3.0	5.3.1
sassc-rails	2.0.0	2.0.1
validators	1.0.0	2.0.0
net-ldap	0.18.0	0.19.0
mechanize	2.9.1	2.12.2
jsonapi-serializer	0.12.0	0.13.0
psych	3.3.4	5.1.2

Tabela 1 – Bibliotecas para Produção

Webpacker, a compatibilidade do Vue.js com o Rails se torna um desafio, exigindo a reescrita de todo o código JavaScript existente que utiliza essa biblioteca.

Para essa reescrita, o Hotwire[3] será adotado como alternativa para desenvolver o novo código JavaScript. Essa escolha se justifica pelo fato do Hotwire ter se tornado o novo padrão para aplicações Rails, proporcionando uma abordagem mais eficiente e moderna para a criação de interfaces interativas. Dessa forma, a transição para o Hotwire não apenas atende à necessidade de reescrita, mas também alinha o desenvolvimento com as melhores práticas atuais recomendadas pelo ecossistema Rails.

[3] Hotwire é uma abordagem para desenvolvimento web que combina Turbo e Stimulus, permitindo a criação de aplicações mais rápidas e responsivas, utilizando HTML em vez de JavaScript para interatividade. O Turbo facilita a navegação rápida e as atualizações de página sem recarregar totalmente a aplicação, enquanto o Stimulus oferece uma forma leve de adicionar interatividade aos elementos HTML. Disponível em https://hotwired.dev. Acesso em 24 de outubro de 2024.

Biblioteca	Versão Atual	Última Versão
better_errors	2.9.0	2.10.0
binding_of_caller	1.0.0	1.0.1
listen	3.8.0	3.9.0
spring	2.1.1	4.2.1
spring-watcher-listen	2.0.1	2.1.0
web-console	4.1.0	4.2.0
brakeman	6.1.0	6.2.2
rubocop	1.58.0	1.67.0
rubocop-rspec	2.25.0	3.1.0
rubocop-rails	2.22.2	2.26.2
bullet	7.1.4	7.2.0
erb_lint	0.5.0	0.7.0
htmlbeautifier	1.4.2	1.4.3
solargraph	0.47.0	0.48.0
yard	0.9.36	0.9.37

Tabela 2 – Bibliotecas para Desenvolvimento

Biblioteca	Versão Atual	Última Versão
database_cleaner-active_record	2.1.0	2.2.0
guard-rspec	1.0.0	1.0.1
simplecov	0.17.1	0.22.0
simplecov-console	0.9.1	0.9.2
shoulda-matchers	5.3.0	6.4.0
capybara-screenshot	1.0.0	1.0.1

Tabela 3 – Bibliotecas para Testes

Biblioteca	Versão Atual	Última Versão
byebug	11.1.3	11.1.5
factory_bot_rails	6.4.2	6.4.3
selenium-webdriver	4.10.0	4.25.0
capybara	3.39.2	3.40.0
webdrivers	5.0.0	6.0.0
faker	3.2.2	3.5.1
rspec-rails	6.1.0	7.0.1

Tabela 4 – Bibliotecas para Desenvolvimento e Testes

4.2 Atualização Preliminar do Framework Rails

Para estimar com precisão a quantidade de páginas que precisarão ser reescritas, foi realizada uma atualização preliminar do Framework Rails. Esse processo permitiu identificar todos os trechos de código que apresentarão incompatibilidades ou problemas após a migração.

Atualmente, a aplicação conta com um total de 57 arquivos de componentes desenvolvidos em Vue.js e 177 arquivos no formato `.html.erb`, responsáveis pela construção das páginas e renderização dos componentes. Com a migração para uma arquitetura mais moderna, será necessário refatorar esses componentes.

A reescrita será feita por meio da substituição do Vue.js por ferramentas nativas mais atuais do framework, como o Hotwire. Isso resultará em uma abordagem mais alinhada às diretrizes do ecossistema Rails, garantindo maior compatibilidade, desempenho aprimorado e redução da complexidade na manutenção do código.

Um exemplo dessa refatoração é o componente Flash Messages, responsável por exibir mensagens temporárias com diferentes estilos, dependendo do tipo da mensagem. Antes da migração, sua implementação utilizava o Vue.js, conforme mostrado abaixo, onde o componente gerenciava dinamicamente a exibição e remoção das mensagens:

```
1  <template>
2    <div>
3      <transition name="fade">
4        <div
5          v-for="(message, index) in flashMessages"
6          v-show="show"
7          :key="index"
8          :class="getAlertClass(message, index)"
9          role="alert"
10       >
11         <button
12           class="close"
13           data-dismiss="alert"
14           aria-label="Close"
15         />
16         {{ message[index + 1] }}
17       </div>
18     </transition>
19   </div>
20 </template>
21
22 <script>
23 export default {
24   name: 'FlashMessages',
25
26   props: {
27     messages: {
```

```
28        type: Array,
29        default() {
30          return [];
31        },
32      },
33    },
34
35    data() {
36      return {
37        show: true,
38        flashMessages: [],
39        types: {
40          'error': 'danger',
41          'alert': 'warning',
42          'notice': 'info',
43          'success': 'success'
44        },
45      };
46    },
47
48    mounted() {
49      this.setFlashMessages();
50      this.fadeOutMessage();
51      this.listenMessages();
52    },
53
54    methods: {
55      setFlashMessages() {
56        this.flashMessages = this.messages;
57      },
58
59      getAlertClass(message, index) {
60        const type = this.types[message[index]];
61
```

```
62        return `alert alert-${type} alert-dismissible`;
63      },
64
65      fadeOutMessage() {
66        const tenSeconds = 10000;
67
68        setTimeout(() => {
69          this.show = false;
70        }, tenSeconds);
71      },
72
73      listenMessages() {
74        this.$root.$on('add-flash-message', (data) => {
75          this.flashMessages.push(data);
76        });
77      },
78    },
79  };
80  </script>
81
82  <style scoped>
83  .fade-enter-active, .fade-leave-active {
84    transition: opacity .5s;
85  }
86
87  .fade-enter, .fade-leave-to {
88    opacity: 0;
89  }
90  </style>
```

Após a migração, o componente foi reescrito utilizando Hotwire e Stimulus, tecnologias modernas do Rails que possibilitam uma abordagem mais eficiente na manipulação do frontend. O Hotwire permite atualizações em tempo real sem a necessidade de requisições AJAX tradicionais, enquanto o Stimulus adiciona comportamento dinâmico de forma mais integrada ao Rails. Essa combinação resulta na redução da complexidade do código e melhora a manutenção. A nova versão do componente é apresentada a seguir:

```
<div
  data-controller="flash-message"
  data-target="flash-message.message"
  class="alert alert-<%= type_class(type) %> alert-dismissible"
  role="alert"
>
  <button
    type="button"
    class="close"
    aria-label="Close"
    data-action="click->flash-message#removeMessage"
  >
    <span aria-hidden="true">&times;</span>
  </button>
  <%= message %>
</div>

import { Controller } from "stimulus"

export default class extends Controller {
  static targets = ["message"]

  connect() {
    setTimeout(() => {
      this.removeMessage()
    }, 10000)
  }

  removeMessage() {
    this.messageTarget.remove();
  }
}
```

A nova implementação traz diversas melhorias na exibição das mensagens flash, substituindo o uso do Vue.js pelo Stimulus, que é mais integrado ao ecossistema Rails. Assim como

na versão anterior, os componentes são renderizados na página de layout, conforme ilustrado abaixo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, user-scalable=no,
    initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <title>
      <%= full_title(yield(:head_title)) %>
    </title>
    <%= csrf_meta_tags %>
    <%= csp_meta_tag %>
    <%= stylesheet_link_tag    'application', media: 'all',
    'data-turbolinks-track': 'reload' %>
    <%= stylesheet_link_tag 'print', media: 'print' %>
    <%= javascript_include_tag 'application',
    'data-turbolinks-track': 'reload' %>
    <%= javascript_pack_tag 'application',
    'data-turbolinks-track': 'reload' %>
    <%= favicon_link_tag %>
  </head>
  <body>
    <div id="app" class="page">
      <div class="page-main">
        <div class="container-fluid m-0">
          <div class="row">
            <%= render 'layouts/shared/header',
            current_user: current_academic %>
            <%= render 'layouts/academics/sidebar' %>
            <div class="col-md-9 col-lg-10">
              <%= render_breadcrumbs builder:
              ::BootstrapBreadcrumbsBuilder %>
              <div class="card" id="main-card">
                <div class="card-header">
```

```
33                                  <h1 class="page-title mb-3">
34                                    <%= yield(:page_title) %>
35                                  </h1>
36                                </div>
37                                <div class="card-body">
38                                  <%= render 'shared/flash_messages'%>
39                                  <%= yield %>
40                                </div>
41                              </div>
42                            </div>
43                          </div>
44                        </div>
45                      </div>
46                    </div>
47                  </body>
48                </html>
```

Na implementação anterior, a renderização e remoção das mensagens flash eram gerenciadas pelo Vue.js. O componente era renderizado através do código `<%= render 'shared/flash_messages' %>`, onde a função `render` chamava o arquivo `_flash_messages.html.erb`. Este arquivo continha apenas a seguinte estrutura:

```
<div><flash-messages :messages="<\%= flash.to_json \%>" /></div>
```

Esse componente Vue realizava as seguintes operações:

- Montava as mensagens no DOM de forma dinâmica ao ser carregado.
- Exibia a mensagem por um tempo determinado e a removia automaticamente da interface.
- Exigia a importação e inicialização do Vue.js no projeto, o que aumentava a complexidade do código.

A refatoração substitui o Vue.js pelo Stimulus, proporcionando uma abordagem mais leve e integrada ao Rails. As principais alterações incluem:

- O arquivo `_flash_messages.html.erb` agora utiliza o Stimulus para gerenciar a exibição e remoção das mensagens, com a conexão entre o HTML e o controlador sendo feita pelos atributos `data-controller` e `data-action`.

- O controlador Stimulus `flash_message_controller.js` é responsável por remover a mensagem automaticamente após 10 segundos, usando o método `connect`.
- A lógica de remoção das mensagens é aplicada diretamente no DOM, sem a necessidade de frameworks externos, o que resulta em menor tempo de carregamento e maior facilidade de manutenção.

Com essa refatoração, a aplicação apresenta melhor desempenho e o código se torna mais simples e alinhado com as melhores práticas do Rails.

5 CONSIDERAÇÕES FINAIS

Até o momento, várias etapas importantes foram realizadas como parte do processo de atualização do SGTCC. A análise inicial do sistema foi concluída, com o mapeamento das dependências descontinuadas, como o Webpacker, e a identificação dos componentes que precisam ser atualizados ou substituídos. Também foi feita a preparação do ambiente de desenvolvimento, incluindo a atualização para versões mais recentes do Ruby on Rails, garantindo que o sistema esteja alinhado com as melhores práticas tecnológicas atuais.

Em paralelo, iniciou-se a reescrita do código JavaScript, com a introdução do Hotwire como alternativa ao uso do Webpacker. Essa transição visa melhorar a integração entre o frontend e o backend, simplificando o código e otimizando as interações entre cliente e servidor. Embora ainda esteja em andamento, já foram observados ganhos preliminares em termos de performance e facilidade de manutenção.

Ao final do trabalho, espera-se que o SGTCC esteja modernizado de maneira significativa, com a remoção completa das dependências obsoletas e a implementação do Hotwire como a principal solução para gerenciar as interações dinâmicas no frontend. Isso resultará em um sistema mais rápido, seguro e fácil de manter, proporcionando uma base sólida para futuras evoluções. A adaptação final do código e os testes realizados garantirão que as novas tecnologias sejam corretamente integradas, assegurando que o sistema continue a atender às necessidades dos usuários de forma eficiente e escalável.

REFERÊNCIAS

CLICKUP. **One app for projects, knowledge, conversations and more**. 2025. https://clickup.com/. Acesso em: 13 fev. 2025.

COINT. **Normas Operacionais Complementares do Trabalho de Conclusão de Curso do Curso Superior de Tecnologia em Sistemas para Internet - Câmpus Guarapuava**. 2023. https://tcc.tsi.pro.br/uploads/attached_document/file/2/normas-operacionais-complementares-do-TCC-TSI-GP-2023-1.pdf. Acesso em: 04 nov. 2024.

DOCKER. **Accelerate how you build, share, and run applications**. 2025. https://www.docker.com. Acesso em: 13 fev. 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 Câmpus Guarapuava**. 2015 — Universidade Tecnológica Federal do Paraná, Guarapuava, PR, 2015.

FOWLER, M. **Refactoring: Improving the Design of Existing Code**. 2nd. ed. Boston, MA: Addison-Wesley, 2018.

GIT. **Free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency**. 2025. https://git-scm.com. Acesso em: 13 fev. 2025.

GITHUB. **Build and ship software on a single, collaborative platform**. 2025. https://github.com/. Acesso em: 13 fev. 2025.

HOTWIRE. **Hotwire: HTML Over The Wire**. 2025. https://hotwired.dev/. Acesso em: 13 fev. 2025.

LEHMAN, M. M.; BELADY, L. A. **Program Evolution: Processes of Software Change**. London: Academic Press, 1985.

LIMA, A. C. **Projeto e implementação de interface baseada na experiência do usuário para um sistema de gerenciamento de trabalho de conclusão de curso**. 2023 — Universidade Tecnológica Federal do Paraná, Guarapuava, PR, 2023.

POSTGRESQL. **PostgreSQL: The World's Most Advanced Open Source Relational Database**. 2025. https://www.postgresql.org/. Acesso em: 13 fev. 2025.

RAILS. **Webpacker**. 2004. https://github.com/rails/webpacker. Acesso em: 24 out. 2024.

RAILS. **Compress the complexity of modern web apps**. 2025. https://rubyonrails.org. Acesso em: 13 fev. 2025.

RAILS. **Upgrading Ruby on Rails**. 2025. https://guides.rubyonrails.org/upgrading_ruby_on_rails.html. Acesso em: 06 fev. 2025.

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 Câmpus Guarapuava**. 2019 — Universidade Tecnológica Federal do Paraná, Guarapuava, PR, 2019.

SOMMERVILLE, I. **Software Engineering**. 9th. ed. Boston, MA: Addison-Wesley, 2011.