



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

Matheus Eduardo Lafeta Feliciano
Shaiane Boesing Rodrigues Borges

Evolução de uma ferramenta open source de gerenciamento ágil de projetos

Florianópolis
2026

Matheus Eduardo Lafeta Feliciano
Shaiane Boesing Rodrigues Borges

Evolução de uma ferramenta open source de gerenciamento ágil de projetos

Trabalho de Conclusão de Curso do Curso de Graduação em sistemas de informação do Campus Florianópolis da Universidade Federal de Santa Catarina para a obtenção do título de bacharel em sistemas de informação.
Orientador: Prof. Jean Carlo Rossa Hauck, Dr.

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Feliciano, Matheus Eduardo Lafeta
Evolução de uma ferramenta open source de gerenciamento
ágil de projetos / Matheus Eduardo Lafeta Feliciano,
Shaiane Boesing Rodrigues Borges ; orientador, Jean Carlo
Rossa Hauck, 2026.
261 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Centro Tecnológico,
Graduação em Sistemas de Informação, Florianópolis, 2026.

Inclui referências.

1. Sistemas de Informação. 2. Gerência de projetos. 3.
Software. 4. Métodos ágeis. I. Borges, Shaiane Boesing
Rodrigues. II. Hauck, Jean Carlo Rossa. III. Universidade
Federal de Santa Catarina. Graduação em Sistemas de
Informação. IV. Título.

Matheus Eduardo Lafeta Feliciano
Shaiane Boesing Rodrigues Borges

Evolução de uma ferramenta open source de gerenciamento ágil de projetos

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de “bacharel em sistemas de informação” e aprovado em sua forma final pelo Curso de Graduação em sistemas de informação.

Florianópolis, 12 de junho de 2026.

Prof. Álvaro Junio Pereira Franco, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Jean Carlo Rossa Hauck, Dr.
Orientador

Profa. Fabiane Barreto Vavassori Benitti,
Dra.
Avaliadora
Universidade Federal de Santa Catarina

Prof. Raul Sidnei Wazlawick, Dr.
Avaliador
Universidade Federal de Santa Catarina

RESUMO

Ferramentas de gestão de projetos têm se mostrado essenciais no apoio ao planejamento, execução e monitoramento de projetos, especialmente com o crescimento do uso de métodos ágeis, como o Scrum. No entanto, observa-se que poucas plataformas de gerenciamento de projetos, especialmente as open source, oferecem suporte integrado que contemple simultaneamente práticas ágeis e processos mais detalhados da gerência tradicional, como análise de riscos, controle de custos e gerenciamento preciso de cronogramas. Essas lacunas dificultam a integração entre abordagens ágeis e tradicionais e limitam o uso pedagógico e profissional dessas ferramentas. Diante desse cenário, identifica-se a necessidade de desenvolvimento e integração de funcionalidades específicas a uma ferramenta de código aberto, ampliando sua aplicabilidade no contexto educacional e corporativo. O estudo contempla a implementação de três módulos principais: um módulo dedicado à gestão de riscos, uma ferramenta para visualização do caminho crítico com base nas histórias de usuários e tarefas do projeto e um módulo de gestão orçamentária. A abordagem metodológica se caracteriza como pesquisa aplicada, envolvendo levantamento bibliográfico, análise de ferramentas correlatas disponíveis no mercado, definição de requisitos e desenvolvimento iterativo com apoio de métodos ágeis. Como resultado, os três módulos foram implementados e integrados ao ambiente ágil da ferramenta open source Taiga.io, preservando sua natureza livre e a possibilidade de implantação local. O módulo de cronograma passou a oferecer um gráfico de Gantt com identificação do caminho crítico e da folga das atividades; o módulo de gestão orçamentária permite registrar custos planejados e realizados, taxas horárias por membro e receitas, consolidando os dados em um painel financeiro; e o módulo de gestão de riscos disponibiliza o registro de riscos, a matriz de probabilidade $\times$ impacto e o acompanhamento da evolução dos riscos ao longo do projeto. Com essas adições, a ferramenta passa a reunir, em um único ambiente open source, suporte a cronograma, custos e riscos, reduzindo a lacuna entre as abordagens ágil e prescritivas identificada na análise.

Palavras-chave: Gerência de projetos, Métodos ágeis, Taiga.io, Gestão de riscos, Caminho crítico, Gestão orçamentária, Software open source.

LISTA DE FIGURAS

Figura 1 – Método do caminho crítico apresentado no Guia PMBOK 6 ^a edição. . .	21
Figura 2 – Exemplo de diagrama de Gantt	22
Figura 3 – Tabela Pxi quantitativa retirada do PMBOK	27
Figura 4 – Análise SWOT	28
Figura 5 – Backlog de projeto no Taiga	41
Figura 6 – Detalhamento da sprint no Taiga	41
Figura 7 – Registro de riscos configurado no Monday	43
Figura 8 – Gestão de cronograma no Monday	43
Figura 9 – Dashboard de gestão de custos no Monday	44
Figura 10 – Pacotes de trabalho no OpenProject	45
Figura 11 – Gráfico de Gantt do OpenProject	46
Figura 12 – Exemplo de RAID log para gestão de riscos no ProjectManager	48
Figura 13 – Gestão de cronograma com gráfico de Gantt no ProjectManager	48
Figura 14 – Card de riscos configurado no Jira	51
Figura 15 – Card de riscos configurado no Jira	51
Figura 16 – Planejamento de cronograma com <i>roadmap</i> e <i>timeline</i> no Jira	52
Figura 17 – Exemplo de estimativas de esforço e registro de horas trabalhadas no Jira	53
Figura 18 – Exemplo 1 de registro de riscos configurado no Artia	55
Figura 19 – Exemplo 2 de registro de riscos configurado no Artia	56
Figura 20 – Planejamento e acompanhamento de cronograma no Artia	56
Figura 21 – Dashboard de indicadores de um projeto no Artia	57
Figura 22 – Exemplo de registro de riscos configurado no Smartsheet	59
Figura 23 – Visualização de cronograma em gráfico de Gantt no Smartsheet	59
Figura 24 – Dashboard de custos e orçamento configurado no Smartsheet	60
Figura 25 – Visão geral de um projeto no Zoho Projects	61
Figura 26 – Gestão de cronograma com gráfico de Gantt no Zoho Projects	62
Figura 27 – Dashboard de orçamento e custos no Zoho Projects	63
Figura 28 – Protótipo Gráfico de Gantt	71
Figura 29 – Protótipo Matriz Pxi	73
Figura 30 – Visão geral da arquitetura do Taiga	77
Figura 31 – Modelo de dados do módulo de gestão de cronograma	88
Figura 32 – Modelo de dados do módulo de gestão financeira	89
Figura 33 – Modelo de dados do módulo de gestão de riscos	90
Figura 34 – Visualização da página do diagrama de Gantt	90
Figura 35 – Visualização da página do diagrama de Gantt com atividade “Catálogo de Produtos” recolhida	91
Figura 36 – Representação das barras no diagrama de Gantt	92

Figura 37 – Menu de opções e edição da página Gantt	92
Figura 38 – Demonstração de caminho crítico no diagrama de Gantt	93
Figura 39 – Representação visual da folga total de uma atividade	94
Figura 40 – Página com a lista de atividades com atributos financeiros	95
Figura 41 – Página com o detalhamento financeiro de uma atividade	96
Figura 42 – Página com detalhamento valor/hora de membro do projeto	97
Figura 43 – Página com as atividades designadas a um membro	98
Figura 44 – Página com as receitas do projeto	99
Figura 45 – Página com o detalhamento de uma receita	99
Figura 46 – Página com um resumo financeiro do projeto, primeira parte	100
Figura 47 – Página com um resumo financeiro do projeto, segunda parte	100
Figura 48 – Tela de listagem de riscos	103
Figura 49 – Tela de detalhamento de um risco	105
Figura 50 – Matriz de Probabilidade × Impacto	106
Figura 51 – Painel de evolução dos riscos do projeto	107
Figura 52 – Resultado da visualização do cronograma no gráfico de Gantt	110
Figura 53 – Resultado da visualização do caminho crítico no gráfico de Gantt	111
Figura 54 – Resultado da visualização da folga total de uma atividade	112
Figura 55 – Cadastro de horas planejadas e custo planejado na listagem de custos por atividade	113
Figura 56 – Distribuição do custo planejado por categorias no detalhamento finan- ceiro do épico 1	114
Figura 57 – Configuração do valor/hora dos integrantes do projeto	115
Figura 58 – Página de listas de atividades de um integrante do projeto	115
Figura 59 – Horas realizadas e custo realizado calculados na listagem de custos por atividade	116
Figura 60 – Entradas de trabalho no detalhamento financeiro da atividade de T2	117
Figura 61 – Listagem de receitas cadastradas no projeto	118
Figura 62 – Detalhamento de uma receita cadastrada no projeto	118
Figura 63 – Indicadores financeiros consolidados no dashboard do projeto	119
Figura 64 – Gráficos financeiros consolidados no dashboard do projeto	119
Figura 65 – Tela de cadastro de um novo risco	122
Figura 66 – Listagem principal dos riscos cadastrados	123
Figura 67 – Busca textual de riscos	123
Figura 68 – Filtragem de riscos por categoria	124
Figura 69 – Alteração de status diretamente pela listagem	124
Figura 70 – Atribuição de responsável	124
Figura 71 – Tela de detalhamento de um risco	125
Figura 72 – Definição da data de validade e justificativa do risco	126

Figura 73 – Associação das sprints ao risco	127
Figura 74 – Visão geral da matriz de riscos	128
Figura 75 – Filtragem automática da listagem após seleção de uma célula da matriz	129
Figura 76 – Busca textual aplicada sobre os riscos da matriz	130
Figura 77 – Painel de filtros avançados da matriz de riscos	130
Figura 78 – Visão geral da evolução dos riscos	132
Figura 79 – Aplicação de filtros avançados na evolução dos riscos	132
Figura 80 – Busca textual aplicada na evolução dos riscos	133
Figura 81 – Visualização individual da evolução de um risco	134
Figura 82 – Associação de avaliação a uma sprint	134
Figura 83 – Inclusão de uma nova avaliação	135
Figura 84 – Confirmação de avaliação pendente	135
Figura 85 – Confirmação para exclusão de avaliação	135

LISTA DE QUADROS

Quadro 1 – Áreas de conhecimento segundo o PMBOK 6 ^a edição.	18
Quadro 2 – Grupos de processos segundo o PMBOK 6 ^a edição.	19
Quadro 3 – Quadro EAR retirada do PMBOK	26
Quadro 4 – Quadro Pxl qualitativa retirada do PMBOK	27
Quadro 5 – Termos e sinônimos de busca	37
Quadro 6 – Matriz de funcionalidades das ferramentas analisadas	65
Quadro 7 – Serviços que compõem a arquitetura da plataforma Taiga	75
Quadro 8 – Novas tabelas introduzidas no banco de dados	87
Quadro 9 – Escala de probabilidade utilizada no módulo de riscos	102
Quadro 10 – Classificação da criticidade dos riscos	102

SUMÁRIO

1	INTRODUÇÃO	12
1.1	OBJETIVOS	13
1.1.1	Objetivo Geral	13
1.1.2	Objetivos Específicos	13
1.2	METODOLOGIA	13
1.2.1	Uso de Inteligência Artificial Generativa	15
2	FUNDAMENTAÇÃO TEÓRICA	16
2.1	GERENCIAMENTO DE PROJETOS	16
2.1.1	Projeto	16
2.1.2	Gerência de Projeto	17
2.1.3	Gerência de Cronograma	17
2.1.3.1	Definição e sequenciamento de atividades	18
2.1.3.2	Estimativa da duração	19
2.1.3.3	Elaboração do cronograma	19
2.1.3.4	Caminho crítico	20
2.1.3.5	Diagrama de Gantt	21
2.1.4	Gerência de Custos	21
2.1.4.1	Estimar custos	22
2.1.4.2	Determinar orçamento	22
2.1.5	Gerência de Riscos	23
2.1.5.1	Planejamento de riscos	23
2.1.5.2	Identificação dos riscos	24
2.1.5.2.1	<i>Métodos para coleta de informações para identificação de novos riscos</i>	<i>24</i>
2.1.5.3	Análise dos riscos	25
2.1.5.4	Monitoramento e controle dos riscos	28
2.2	TECNOLOGIAS E CONCEITOS	29
2.2.1	Gerais	29
2.2.2	Back-end	30
2.2.3	Front-end	32
3	TRABALHOS RELACIONADOS	35
3.1	DEFINIÇÃO DO PROTOCOLO DE PESQUISA	35
3.1.1	Critérios de Inclusão e Exclusão	35
3.1.2	Plataformas de Busca	36
3.1.3	Termos de Busca	36
3.1.4	Construção da String e dos Prompts de Busca	36
3.1.5	Definição da Extração dos Dados	38
3.2	EXECUÇÃO DA BUSCA	38

3.2.1	Aplicação das Strings de Busca	39
3.2.2	Processo de Seleção dos Resultados	39
3.2.3	Consolidação dos Itens Encontrados	39
3.3	RESULTADOS	39
3.3.1	Taiga	40
3.3.2	Monday	42
3.3.3	OpenProject	44
3.3.4	ProjectManager	46
3.3.5	Jira	49
3.3.6	Artia	54
3.3.7	Smartsheet	57
3.3.8	Zoho Projects	60
3.4	DISCUSSÃO	64
4	PROPOSTA DE SOLUÇÃO	67
4.1	SELEÇÃO DA FERRAMENTA	67
4.2	LEVANTAMENTO DE REQUISITOS	68
4.2.1	Épico E1 – Gestão de Cronograma	68
4.2.1.1	Integração entre o planejamento ágil e a gestão de cronograma	69
4.2.2	Épico E2 – Gestão Financeira	70
4.2.3	Épico E3 – Gestão de Riscos	70
4.3	ANÁLISE DE REQUISITOS	70
4.3.1	Histórias do Épico E1 – Gestão de Cronograma	71
4.3.2	Histórias do Épico E2 – Gestão Financeira	71
4.3.3	Histórias do Épico E3 – Gestão de Riscos	72
4.4	CONSIDERAÇÕES FINAIS	72
5	DESENVOLVIMENTO DAS NOVAS FUNCIONALIDADES DO TAIGA	74
5.1	AMBIENTE DE DESENVOLVIMENTO E FERRAMENTAS UTILIZADAS	74
5.2	ARQUITETURA DO SISTEMA	75
5.3	CONFIGURAÇÃO DO AMBIENTE DE DESENVOLVIMENTO	77
5.4	EXECUÇÃO DA APLICAÇÃO EM UMA NOVA INSTALAÇÃO	83
5.4.1	Pré-requisitos	83
5.4.2	Passo a passo de execução	84
5.5	IMPLEMENTAÇÃO DAS FUNCIONALIDADES PROPOSTAS	85
5.5.1	Visão geral das alterações no projeto do sistema	86
5.5.1.1	Modelo de dados	87
5.5.2	Épico E1 - Gestão de Cronograma	87
5.5.3	Épico E2 – Gestão Financeira	94
5.5.4	Épico E3 – Gestão de Riscos	102

5.6	TESTES	109
5.6.1	Épico E1 – Gestão de Cronograma	109
5.6.2	Épico E2 – Gestão Financeira	111
5.6.3	Épico E3 – Gestão de Riscos	120
5.7	MODIFICAÇÕES NO CÓDIGO ORIGINAL E MANUTENÇÃO FUTURA .	134
6	CONCLUSÃO	140
6.1	CONSIDERAÇÕES FINAIS	140
6.2	TRABALHOS FUTUROS	141
	REFERÊNCIAS	143
	APÊNDICE A – ANÁLISE RESTANTE DAS HISTÓRIAS DE	
	USUÁRIO	146
	APÊNDICE B – DETALHAMENTO DAS MODIFICAÇÕES NO	
	CÓDIGO ORIGINAL	149
B.1	ÉPICO E1 - GESTÃO DE CRONOGRAMA	149
B.1.1	Back-end	149
B.1.2	Front-end	161
B.1.3	Front-end-Next	172
B.2	ÉPICO E2 - GESTÃO FINANCEIRA	181
B.2.1	Back-end	181
B.2.2	Front-end	191
B.2.3	Front-end-Next	203
B.3	ÉPICO E3 - GESTÃO DE RISCOS	211
B.3.1	Back-end	211
B.3.2	Front-end	225
B.3.3	Front-end-Next	241
	APÊNDICE C – DISPONIBILIZAÇÃO DO CÓDIGO-FONTE . .	249
	APÊNDICE D – ARTIGO CIENTÍFICO EM FORMATO SBC . .	250

1 INTRODUÇÃO

As práticas de gerenciamento de projetos evoluíram seguindo caminhos distintos, o que fez com que métodos ágeis e abordagens prescritivas nem sempre conversassem de forma efetiva. Enquanto os modelos de ciclo de vida iterativos priorizam adaptação contínua, ciclos curtos e flexibilidade operacional, os modelos prescritivos baseiam-se em previsibilidade, planejamento detalhado e análise estruturada. Essa separação metodológica cria dificuldades práticas quando se busca integrar, em um mesmo ambiente, tanto a agilidade necessária para lidar com mudanças constantes quanto o rigor analítico indispensável para estimativas, controles e tomada de decisão.

De um lado, métodos ágeis impulsionam ciclos curtos, colaboração contínua e adaptação rápida, mas frequentemente carecem de instrumentos para análises aprofundadas. Em sentido oposto, práticas consolidadas de gerenciamento prescritivas, como as sistematizadas no Guia PMBOK (PMI, 2017), oferecem rigor metodológico, mas não acompanham com a mesma agilidade as iterações rápidas típicas de equipes que atuam em contextos dinâmicos e iterativos. Essa diferença de ênfases reflete-se diretamente nas ferramentas de gerenciamento de projetos: ferramentas orientados à agilidade priorizam quadros visuais e ciclo de vida iterativos, enquanto soluções prescritivas priorizam modelos de planejamento estruturado, raramente oferecendo suporte integrado aos dois modelos.

Essa assimetria entre abordagens revela uma lacuna metodológica amplamente discutida na literatura. Vierlboeck et al. (2019) apontam a ausência de modelos consolidados que integrem práticas orçamentárias ao desenvolvimento ágil, evidenciando limitações na dimensão financeira do planejamento iterativo. Dingsøyr e Petit (2021) destacam a carência de abordagens maduras para gestão de riscos em contextos que combinam desenvolvimento ágil e práticas prescritivas, reforçando que incertezas estruturais permanecem pouco tratadas em modelos de ciclo de vida iterativos. Bloss et al. (2025), por sua vez, evidenciam a importância de análises de caminho crítico baseadas em modelos probabilísticos para aumentar a confiabilidade do planejamento — um elemento que métodos ágeis não incorporam de forma nativa. Esses estudos convergem ao demonstrar que, embora amplamente adotadas, métodos ágeis não contemplam integralmente elementos essenciais da gerência de projetos, resultando em práticas fragmentadas e na necessidade de utilizar múltiplas ferramentas paralelas para cobrir diferentes dimensões do planejamento.

Atualmente existem diversas ferramentas de gerenciamento ágil de projetos disponíveis, como por exemplo Taiga, OpenProject, Jira e ProjectManager. Dentre essas ferramentas, o Taiga.io, open source e orientado a métodos ágeis, apresenta-se como um caso representativo dessa limitação. Apesar de sua simplicidade e capacidade de apoiar ciclos de vida iterativos, a ferramenta não oferece recursos nativos para uma gerência de projetos mais abrangente, carecendo de funcionalidades como gestão de riscos, cronograma e orçamento. A expansão dessas capacidades dentro de um ambiente ágil como o Taiga.io

possibilita integrar práticas prescritivas de gerenciamento em uma única ferramenta, contribuindo para a convergência entre abordagens iterativas e prescritivas. Além disso, por se tratar de uma solução livre e passível de implantação local, sua evolução também amplia as possibilidades de utilização da ferramenta em contextos acadêmicos, permitindo que conceitos relacionados à gestão de riscos, cronograma e orçamento sejam explorados em conjunto em atividades de ensino e aprendizagem. Dessa forma, o Taiga.io foi escolhido como objeto de estudo deste trabalho, sendo evoluído com funcionalidades voltadas à gestão de riscos, cronograma e orçamento.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Evoluir uma ferramenta open source a partir de um estudo das limitações presentes em ferramentas de gerenciamento de projetos. A evolução da ferramenta contempla a implementação de funcionalidades prescritivas como gestão de riscos, análise de caminho crítico e gestão orçamentária, integrando-as ao ambiente ágil da ferramenta e ampliando sua aplicabilidade em contextos educacionais e profissionais.

1.1.2 Objetivos Específicos

- Realizar um estudo analítico sobre ferramentas de gerenciamento de projetos para identificar como tratam riscos, cronograma e orçamento.
- Identificar, a partir desse estudo, quais aspectos dessas práticas não são contemplados pelas ferramentas estudadas e selecionar uma ferramenta para evolução.
- Implementar um módulo de gestão de riscos que permita acompanhar de forma contínua os riscos do projeto.
- Desenvolver um módulo de análise do caminho crítico capaz de estruturar atividades, dependências e calcular a duração mínima do projeto.
- Implementar um módulo de gestão orçamentária que possibilite estimar custos e organizar orçamentos.

1.2 METODOLOGIA

Este trabalho caracteriza-se como uma pesquisa aplicada, de enfoque tecnológico, pois tem como principal resultado o desenvolvimento e a integração de novos módulos de software à uma ferramenta open source, voltados à gestão de riscos, à gestão de cronograma e à gestão orçamentária em projetos que utilizam métodos ágeis.

A fim de alcançar os objetivos propostos neste trabalho, o seu desenvolvimento foi dividido nas seguintes etapas:

Estudo da fundamentação teórica: levantamento bibliográfico sobre gerência de projetos, métodos ágeis, gestão de riscos, gestão de cronograma, caminho crítico e gestão de custos, com base em guias de referência e na literatura especializada, de forma a estabelecer a base conceitual que orienta as decisões de projeto e as funcionalidades a serem implementadas.

Análise das ferramentas de gerência de projeto de mercado: mapeamento e análise comparativa de ferramentas de gerenciamento de projetos amplamente utilizadas no mercado, identificando como tratam riscos, cronograma e orçamento, bem como suas limitações, com o objetivo de compreender como esses aspectos são abordados e quais práticas predominam no mercado atual.

Escolha da ferramenta: nesta etapa será selecionada, com base nos estudos e na análise comparativa das ferramentas de gerenciamento de projetos realizados na etapa anterior, a ferramenta que será tomada como objeto de estudo e de eventual aprimoramento, considerando critérios como aderência a métodos ágeis, natureza open source, possibilidade de implantação on-premise sem custos de licença.

Definição das funcionalidades a serem implementadas: especificação dos requisitos funcionais a partir da fundamentação teórica e da análise comparativa das ferramentas, organizando-os em épicos, histórias de usuário e critérios de aceitação relacionados à gestão de cronograma, gestão financeira e gestão de riscos.

Estudo do código da ferramenta: análise da arquitetura e do código-fonte da ferramenta escolhida, abrangendo camadas de back-end, front-end e modelo de dados, a fim de identificar pontos de extensão adequados, padrões de projeto já adotados e integrações necessárias para inclusão dos novos módulos.

Implementação das funcionalidades: desenvolvimento iterativo dos módulos de gestão de cronograma, gestão financeira e gestão de riscos, seguindo práticas de métodos ágeis, com uso de controle de versão, ambientes de desenvolvimento e teste reprodutíveis e integração progressiva com os componentes existentes da ferramenta.

Avaliação das funcionalidades implementadas: realização de testes exploratórios ao longo do desenvolvimento, complementados por testes funcionais documentados a partir dos critérios de aceitação definidos nas histórias de usuário, para verificar se as funcionalidades implementadas atendem aos requisitos levantados, se se integram adequadamente aos fluxos de trabalho do Taiga.io e se contribuem para reduzir as lacunas identificadas na etapa de análise.

Refinamento da produção textual: revisão e consolidação da documentação gerada ao longo do projeto, com atualização e aprimoramento contínuo do texto do Trabalho de Conclusão de Curso, incluindo a fundamentação teórica, a descrição detalhada da solução proposta, o relato da implementação e a discussão dos resultados obtidos.

1.2.1 Uso de Inteligência Artificial Generativa

Na elaboração deste trabalho, foram utilizadas ferramentas de inteligência artificial generativa como apoio ao desenvolvimento e à redação técnica, especificamente Codex com o modelo Codex-5.3, Antigravity com o modelo Gemini 3.1 Pro (High) e Claude Opus 4.8. Essas ferramentas auxiliaram na análise de código, investigação de erros, compreensão de estruturas do projeto, revisão textual e organização de trechos da documentação. Seu uso ocorreu de forma supervisionada e declarada, em conformidade com as orientações de integridade acadêmica da RN nº 217/2025/CUn da Universidade Federal de Santa Catarina, do CNPq e da Sociedade Brasileira de Computação, especialmente quanto à transparência no uso de sistemas automatizados e à responsabilidade dos autores pelo conteúdo apresentado.

Cabe destacar que todas as decisões de projeto, testes funcionais, interpretações dos resultados, seleção de trechos incorporados ao texto e verificação do código permaneceram sob responsabilidade humana. As respostas geradas pelas ferramentas de IA foram avaliadas criticamente antes de qualquer incorporação ao trabalho, com verificação no código-fonte, nos arquivos do projeto, na execução da aplicação ou nas fontes bibliográficas correspondentes. Dessa forma, o uso de IA ocorreu como instrumento auxiliar, sem substituição da autoria, da análise técnica ou da responsabilidade acadêmica pelo conteúdo apresentado.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos, as tecnologias empregadas e os temas de gerência de projetos que fundamentam o desenvolvimento deste trabalho, estabelecendo a base conceitual utilizada nas seções seguintes.

2.1 GERENCIAMENTO DE PROJETOS

2.1.1 Projeto

Segundo o Guia PMBOK (PMI, 2017, p. 4) “Projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado único”. Isso significa que, em sua natureza, um projeto necessariamente gera um resultado final em algum momento do tempo. E este resultado, por sua vez, deve ser único. Destacam-se, portanto, duas características essenciais presentes em qualquer projeto: a temporariedade e a unicidade dos resultados.

A primeira característica a ser destacada é a temporariedade. Projetos possuem início e término claramente definidos. “Temporário não significa necessariamente que um projeto tem curta duração” (PMI, 2017, p. 542), mas sim que seu ciclo de vida é limitado. O encerramento pode ocorrer quando os objetivos são atingidos, quando se conclui que não poderão ser cumpridos, quando os recursos se esgotam ou quando a necessidade que justifica o projeto deixa de existir (PMI, 2017).

A segunda característica é a unicidade dos resultados. Uma entrega é definida como qualquer produto, resultado ou capacidade verificável e única. Essa entrega pode ser um produto, serviço ou resultado de pesquisa, podendo ainda combinar mais de um desses elementos. Mesmo quando atividades semelhantes se repetem em diferentes projetos, cada entrega mantém seu caráter exclusivo em função de suas características específicas (PMI, 2017).

Além dessas duas características citadas, um projeto também tem o papel de promover mudanças. Isso significa que ele existe para levar uma organização de uma situação atual para uma situação futura desejada. Essa mudança pode acontecer de forma imediata ou em etapas, até que o objetivo seja alcançado. Assim, quando um projeto é concluído com sucesso, ele representa a transformação planejada que a organização buscava (PMI, 2017).

Por último, todo projeto deve gerar valor. Segundo o PMBOK (PMI, 2017), o valor pode ser algo tangível, como aumento do lucro, novos equipamentos ou maior participação no mercado, ou intangível, como reputação, reconhecimento da marca ou alinhamento com a estratégia da empresa. Isso mostra que o resultado de um projeto vai além da entrega em si: ele precisa trazer benefícios reais para a organização e para as partes interessadas envolvidas.

2.1.2 Gerência de Projeto

Gerenciar projetos é planejar a forma pela qual um projeto será executado dentro do tempo estimado, a fim de obter o resultado desejado com os recursos disponíveis. O Guia PMBOK (PMI, 2017) define o gerenciamento de projetos como a aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto para atender aos seus requisitos.

Para realizar isso, algumas práticas essenciais costumam ser seguidas: identificar os requisitos do projeto; considerar as diferentes necessidades, preocupações e expectativas das partes interessadas; estabelecer e manter a comunicação ativa; gerenciar recursos; e equilibrar restrições muitas vezes conflitantes, como escopo, cronograma, custo, qualidade, recursos e risco (PMI, 2017).

O Guia PMBOK (2017) organiza essas práticas em dez áreas do conhecimento e cinco grupos de processos que, respectivamente, representam os campos de atuação essenciais para uma gestão de projetos e as fases lógicas desse gerenciamento. As dez áreas de conhecimento estão apresentadas no Quadro 1, enquanto os cinco grupos de processos são detalhados no Quadro 2.

Nas seções a seguir, são aprofundados os conceitos de cronograma, custo e risco, que constituem os principais focos de análise deste estudo.

2.1.3 Gerência de Cronograma

A gerência de cronogramas envolve não só a estimativa de tempo de cada tarefa do projeto, mas também a ordem em que elas acontecem e suas respectivas dependências. Além disso, é uma das principais ferramentas de interesse das partes interessadas, já que é o principal caminho pela qual estimativas de prazos são feitas.

A sexta edição do Guia PMBOK afirma que o cronograma do projeto fornece um plano detalhado que representa como e quando o projeto vai entregar os produtos, serviços e resultados definidos no escopo do projeto, e serve como ferramenta de comunicação, gerenciamento de expectativas das partes interessadas e como base para a emissão de relatórios de desempenho (PMI, 2017).

De acordo com Barcaui, o controle do cronograma deve ser realizado de forma contínua, analisando causas, impactos e possíveis desvios ao longo do projeto. Além disso, a responsabilidade pela preparação das estimativas deve recair sobre aqueles que executam as atividades do projeto, pois são eles que detêm o conhecimento prático necessário para avaliar o esforço necessário (Barcaui, 2012).

O Guia PMBOK 6ª edição apresenta o gerenciamento de cronograma em seis processos: (1) planejar como o cronograma será gerenciado, (2) separar o trabalho em atividades específicas, (3) ordenar essas atividades e estabelecer suas dependências, (4) estimar as durações de cada atividade, (5) consolidar todas as informações em um modelo

Quadro 1 – Áreas de conhecimento segundo o PMBOK 6ª edição.

Áreas de conhecimento	Descrição resumida
Gerenciamento da Integração do Projeto	Coordena os elementos do projeto de forma unificada e garante que os objetivos sejam alcançados de maneira integrada.
Gerenciamento do Escopo do Projeto	Define o que será entregue e assegura que todo o trabalho necessário — e apenas ele — seja realizado.
Gerenciamento do Cronograma do Projeto	Planeja, define e controla o tempo e a sequência das atividades.
Gerenciamento dos Custos do Projeto	Estima, orça e controla os gastos do projeto para garantir que o orçamento seja respeitado.
Gerenciamento da Qualidade do Projeto	Assegura que o produto, serviço ou resultado atenda aos padrões e requisitos estabelecidos.
Gerenciamento dos Recursos do Projeto	Planeja e gerencia a equipe e demais recursos físicos necessários à execução do projeto.
Gerenciamento das Comunicações do Projeto	Garante o fluxo adequado e eficiente de informações entre todas as partes envolvidas.
Gerenciamento dos Riscos do Projeto	Identifica, analisa e responde a eventos incertos que possam afetar o desempenho do projeto.
Gerenciamento das Aquisições do Projeto	Administra contratos e aquisições externas de bens e serviços necessários ao projeto.
Gerenciamento das Partes Interessadas do Projeto	Identifica as partes interessadas e promove seu engajamento ao longo do ciclo de vida do projeto.

Fonte: Adaptado de PMI (2017).

e linha de base do cronograma e (6) monitorar o andamento e controlar as mudanças (PMI, 2017).

Barcaui também aborda o gerenciamento do tempo em oito pontos que se relacionam com a visão do PMBOK, mas trazem um foco mais prático: (1) definição das atividades, (2) sequenciamento das atividades, (3) estimativa de recursos, (4) estimativa da duração, (5) elaboração do cronograma, (6) controle do cronograma, (7) análise de valor agregado e (8) corrente crítica (Barcaui, 2012).

Este estudo dará maior atenção às etapas de definição e sequenciamento das atividades, estimativa da duração e elaboração do cronograma.

2.1.3.1 Definição e sequenciamento de atividades

Primeiramente, deve ser realizada a definição das atividades. Esse processo trata de dividir pacotes de entrega maiores em itens menores, que possam ser mais facilmente mensuráveis, atribuídos a responsáveis e executados. O nível de detalhamento deve ser

Quadro 2 – Grupos de processos segundo o PMBOK 6ª edição.

Grupos de processos	Descrição resumida
Iniciação	Define e autoriza formalmente o início do projeto ou de uma de suas fases.
Planejamento	Estabelece os objetivos do projeto e desenvolve o plano para alcançá-los.
Execução	Coloca o plano em prática, realizando o trabalho definido para atingir os objetivos.
Monitoramento e Controle	Acompanha o progresso, compara o desempenho planejado com o real e implementa ajustes quando necessário.
Encerramento	Finaliza todas as atividades e formaliza a conclusão do projeto ou de uma fase.

Fonte: Adaptado de PMI (2017).

suficiente para permitir a estimativa de duração, do custo e das dependências (Barcaui, 2012, p.26). Já o sequenciamento, trata de organizar essas atividades de forma lógica, isto é, considerando as dependências entre elas, o que permite estabelecer a ordem adequada de execução (Barcaui, 2012, p.35).

2.1.3.2 Estimativa da duração

Estimar trata-se de calcular o tempo necessário para concluir cada atividade com os recursos disponíveis (PMI, 2017) e baseando-se nas informações mais confiáveis acessíveis no momento. Um fator muito importante para realizar estimativas é compreender quais são os recursos humanos disponíveis e, se possível, aproveitar de dados históricos de produtividade para fazer estimativas mais realistas. É necessário ter clara a distinção entre duração e esforço. Segundo Barcaui (Barcaui, 2012), esforço é a quantidade efetiva de trabalho que será despendida por um recurso, enquanto duração é o intervalo de tempo de calendário necessário para concluir a atividade, considerando jornadas de trabalho, restrições e interrupções.

2.1.3.3 Elaboração do cronograma

Tendo sido definidas as atividades, suas sequências lógicas, dependências e estimativas de duração, é possível elaborar o cronograma a partir da análise das atividades, sequências, durações, recursos e restrições, gerando um modelo de cronograma. O PMBOK resume este passo como o processo de “analisar sequências de atividades, durações, requisitos de recursos e restrições para criar o modelo de cronograma”, cujo benefício é produzir datas planejadas para as atividades (PMI, 2017). É importante reforçar que este processo ocorre ao longo do projeto, pois é iterativo e exige revisões do cronograma. Existem séries de ferramentas e métodos aplicáveis nesse momento, como caminho crítico,

otimização de recursos, antecipações/esperas e compressão. Barcaui reforça que “determinar a programação de um projeto não é uma atividade simples. Na verdade, é uma combinação de arte e ciência” (Barcaui, 2012, p. 82), isto porque a ciência identifica o caminho crítico e as folgas, enquanto a arte garante, mesmo com todas as incertezas, que o cronograma seja cumprido.

2.1.3.4 Caminho crítico

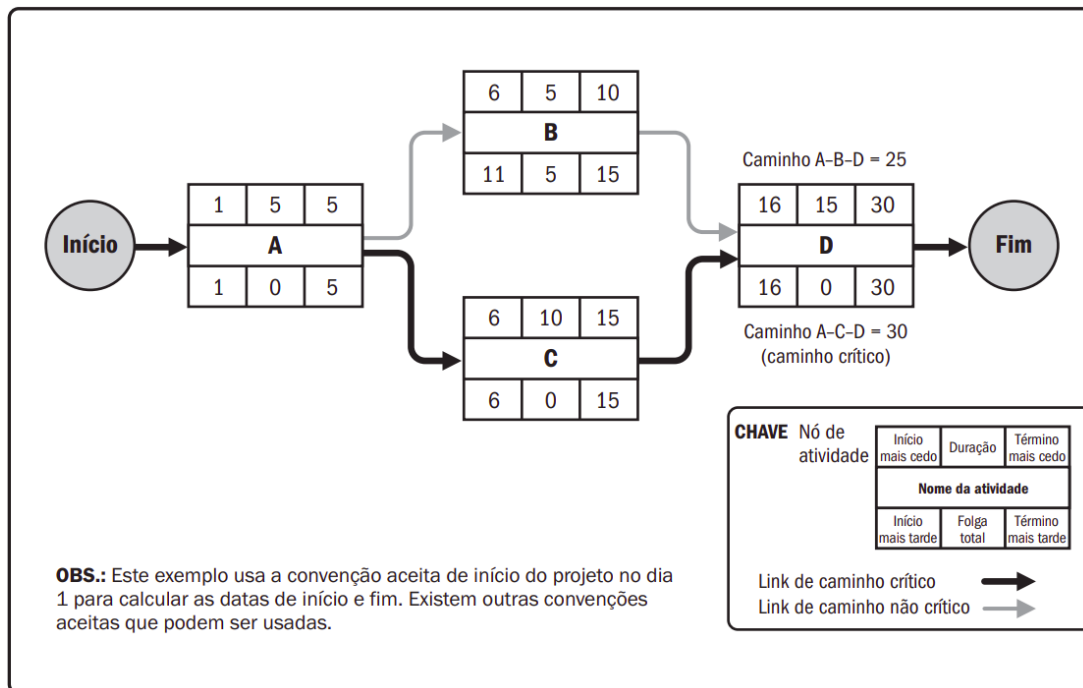
O método do caminho crítico é apresentado no PMBOK como a técnica que é usada para estimar a duração mínima do projeto e que calcula início e término mais cedo e mais tarde para todas as atividades, identificando o caminho mais longo que determina a menor duração possível do projeto; nesse caminho, a folga total costuma ser geralmente zero (PMI, 2017).

O guia afirma que as datas calculadas não são o cronograma final, mas uma indicação dos períodos de tempo dentro dos quais a atividade poderia ser executada, abordando conceitos como folga total, folga livre, folga positiva ou negativa, e que a análise de folga negativa ajuda a recolocar o cronograma de volta aos trilhos (PMI, 2017). Em complemento, Barcaui lista o método do caminho crítico entre as técnicas analíticas mais usadas e introduz a ideia de datas mais cedo/mais tarde como conceitos fundamentais para o cálculo e para a identificação da criticidade (Barcaui, 2012).

Para compreender o funcionamento do método, é importante entender alguns conceitos fundamentais. O início mais cedo (IC) representa o momento mais próximo em que uma atividade pode começar, considerando a conclusão das atividades predecessoras. O término mais cedo (TC) é a data mais próxima em que a atividade pode ser finalizada, caso inicie em seu IC. Já o início mais tarde (IT) e o término mais tarde (TT) indicam os limites máximos em que a atividade pode começar ou terminar sem causar atrasos no prazo total do projeto. A diferença entre o término mais tarde e o término mais cedo define a folga total (FT), que é o tempo que uma atividade pode atrasar sem impactar a conclusão do projeto. Quando essa folga é igual a zero, a atividade é considerada crítica, ou seja, qualquer atraso nela afetará diretamente o prazo final.

A Figura 1 apresenta um exemplo ilustrativo do método, destacando a sequência de atividades que compõe o caminho crítico e, portanto, determinam a menor duração possível do projeto. Cada atividade é representada por um retângulo que exibe os principais elementos utilizados nesse cálculo: início mais cedo (IC), término mais cedo (TC), duração (D), início mais tarde (IT), término mais tarde (TT) e folga total (FT). As setas mostram as dependências entre as atividades, enquanto as cores diferenciam o caminho crítico (cor preta em negrito) das demais atividades com folga (em cinza claro).

Figura 1 – Método do caminho crítico apresentado no Guia PMBOK 6ª edição.



Fonte: PMI (2017).

2.1.3.5 Diagrama de Gantt

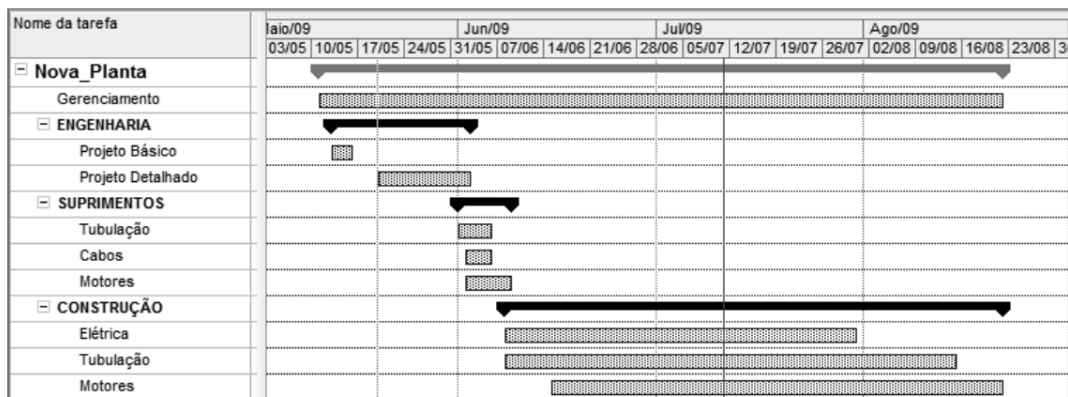
O diagrama de Gantt, também denominado gráfico de barras, é uma forma de representação visual do cronograma em que atividades, pacotes de trabalho ou entregas são dispostos em uma escala temporal, geralmente por meio de barras horizontais que indicam suas datas de início, término e duração, conforme representado na figura 2. Essa representação facilita a compreensão da distribuição das atividades ao longo do tempo, podendo ser utilizada tanto em uma visão resumida, com níveis superiores do projeto, quanto em uma visão detalhada, com atividades específicas. Além de apoiar a comunicação do cronograma, o gráfico de Gantt pode contribuir para o controle do projeto ao permitir a visualização de dependências, marcos, atividades críticas e comparações entre o planejado e o executado. (Barcaui, 2012)

2.1.4 Gerência de Custos

O conceito de custo refere-se ao valor que se paga para produzir algo, seja um serviço ou um bem. No âmbito de projetos, pode ser compreendido como o valor necessário para a utilização de recursos — humanos, equipamentos, materiais ou insumos — no processo de criação de um produto, serviço ou resultado. Um gasto somente se caracteriza como custo no momento em que o recurso é efetivamente utilizado na execução de uma atividade (Barbosa et al., 2019).

O gerenciamento de custos, por sua vez, inclui um conjunto de processos que

Figura 2 – Exemplo de diagrama de Gantt



Fonte: Barcaui (2012, p.120).

buscam estimar, orçar e controlar os custos do projeto, a fim de garantir que ele seja concluído dentro do orçamento aprovado (PMI, 2017). Dessa forma, o gerenciamento de custos mostra-se essencial na gestão de projetos, pois estabelece as bases para a tomada de decisões viáveis no contexto econômico do empreendimento.

2.1.4.1 Estimar custos

Estimar custos é realizar uma previsão dos recursos financeiros que serão necessários para executar as atividades do projeto. Essa estimativa considera, entre outros, o uso de pessoas, materiais e equipamentos. Conforme define o PMBOK, trata-se de uma avaliação quantitativa dos custos prováveis dos recursos necessários para concluir uma atividade (PMI, 2017).

É fundamental que essas estimativas sejam confiáveis, pois servem de base para a definição do orçamento e para o controle da saúde financeira do projeto. Barbosa et al. destacam que as estimativas devem ser elaboradas com base nas informações conhecidas no momento e precisam levar em conta os riscos envolvidos (Barbosa et al., 2019). Os autores complementam que a precisão dessas estimativas tende a aumentar à medida que o projeto avança e novas informações são conhecidas.

Para que a estimativa de custos seja realizada, é necessário que o projeto já tenha definido seu escopo, atividades e recursos necessários, pois são esses elementos que fornecem a base para calcular os valores envolvidos. Além disso, recomenda-se o uso de informações históricas e dados de projetos anteriores, que ajudam a dar maior realismo às previsões (Barbosa et al., 2019).

2.1.4.2 Determinar orçamento

Determinar o orçamento é transformar as estimativas de custos feitas anteriormente em um valor total para o projeto. Nesse momento, é possível visualizar quanto será

necessário gastar em cada parte do trabalho. Segundo o PMBOK, o principal benefício dessa etapa é a definição da linha de base de custos, que serve para acompanhar o andamento financeiro e medir o desempenho do projeto (PMI, 2017).

O desempenho do projeto, nesse contexto, é medido pela comparação de valores planejados com os valores gastos em cada fase. O orçamento aprovado constitui justamente essa linha de base de custos inicial, que passa a servir como referência de acompanhamento (Barbosa et al., 2019). Portanto, a determinação do orçamento é uma etapa essencial na gerência de custos, pois define os limites financeiros dentro dos quais o projeto deve ser conduzido.

2.1.5 Gerência de Riscos

Quando se fala de risco, usualmente ele é considerado apenas como algo negativo. O PMBOK, porém, define risco como um evento ou condição incerta que, caso se concretize, pode gerar impactos negativos ou positivos. O livro Gerenciamento de Riscos apresenta um bom exemplo: a variação do câmbio pode trazer consequências positivas ou negativas — a depender da apreciação ou depreciação da moeda e da exposição do projeto (receitas ou custos em moeda estrangeira) (PMI, 2017; Salles Jr., 2010).

Riscos possuem três componentes: o evento, a probabilidade de ocorrer e o impacto. O evento descreve a origem do risco e seus possíveis efeitos; a probabilidade indica a chance de ocorrência; o impacto expressa o quanto a consequência afetará o projeto, caso o risco se concretize. Em geral, o efeito é tratado qualitativamente, enquanto o impacto é medido quantitativamente (Salles Jr., 2010).

2.1.5.1 Planejamento de riscos

O planejamento de riscos deve ser feito na abertura do projeto, avaliando tamanho e complexidade para que o estudo seja proporcional. Depois de elaborado, requer manutenção durante todo o ciclo de vida, com revisões periódicas ou quando houver mudanças relevantes no contexto. No planejamento, podem constar: categorias de riscos, definições de probabilidade e impacto e a matriz de probabilidade $\times$ impacto. Outros itens podem ser incluídos, mas, para o contexto deste trabalho, a explicação desses elementos é suficiente (PMI, 2017).

O processo de gerenciamento de riscos contempla fases que podem ser divididas em identificação de riscos, análise dos riscos e monitoramento dos riscos. Na fase de identificação são realizadas dinâmicas a fim de mapear o máximo número de riscos possíveis, em análise é feita a categorização quantitativa e qualitativa desses riscos e no monitoramento, é feito o acompanhamento da evolução e resolução desses riscos.

2.1.5.2 Identificação dos riscos

Identificar os Riscos é o processo de levantar os riscos individuais e as fontes do risco geral do projeto, registrando suas características de forma clara e uniforme para apoiar a análise e o planejamento de respostas. O principal resultado é a documentação estruturada dos riscos no registro (com formatação consistente — por exemplo, causa–evento–efeito), incluindo, quando cabível, responsáveis preliminares e respostas iniciais a serem refinadas nos processos seguintes. Trata-se de um processo contínuo ao longo do ciclo de vida, pois novos riscos podem surgir e o nível de risco agregado pode se alterar; a cadência dessas revisões é definida no plano de gerenciamento dos riscos (PMI, 2017).

A identificação deve ser colaborativa e abrangente, envolvendo gerente do projeto, equipe, especialistas de risco (se houver), usuários, clientes, outras áreas (operações, jurídico, suprimentos), partes interessadas e especialistas externos. Além de ampliar a cobertura, essa participação fortalece o senso de responsabilidade da equipe sobre os riscos e respectivas ações de resposta. Recomenda-se incentivar que todas as partes interessadas contribuam com entradas de risco, garantindo visão multifacetada e reduzindo vieses na etapa de priorização subsequente (PMI, 2017).

2.1.5.2.1 Métodos para coleta de informações para identificação de novos riscos

Com as informações do projeto organizadas, passa-se à identificação de novas incertezas, examinando de forma sistemática múltiplas dimensões: cada item do escopo, cada atividade do cronograma e cada recurso do orçamento são fontes potenciais de risco. Todas as áreas de conhecimento podem originar riscos — integração (ambiente), escopo (definição insuficiente), prazos (metas apertadas/inviáveis), custos (orçamentos restritos), recursos humanos (doenças, absenteísmo, produtividade, demissão, capacitação), comunicações (perdas/ineficácia), qualidade (falhas/inadequação) e aquisições (baixo desempenho, não conformidade, atrasos). Para apoiar esse levantamento, recomenda-se o uso de técnicas de dinâmica de grupo, como brainstorming, brainwriting, Delphi e SWOT, envolvendo a equipe para ampliar a cobertura e reduzir vieses (Salles Jr., 2010).

Brainstorming e Brainwriting

Brainstorming é uma reunião da equipe facilitada, geralmente pelo gerente de projetos, para gerar ideias livremente e identificar riscos, baseada no princípio de “pegar carona na ideia do outro”. Exige preparação (seleção de participantes, distribuição prévia de informações) e obedece à regra “não ao não”: nenhuma ideia é criticada durante a coleta, preservando o fluxo criativo. As contribuições são registradas integralmente e, ao final, obtém-se uma lista de riscos (Salles Jr., 2010).

Brainwriting com o mesmo objetivo, substitui o debate aberto pela produção escrita: cada participante anota os riscos sob sua ótica, as folhas circulam em rodadas para que

outros complementem, e ao final a equipe consolida as entradas e seleciona os itens que comporão a lista final. Essa abordagem tende a ampliar a participação e reduzir inibições (Salles Jr., 2010).

Técnica Delphi

Procedimento estruturado, remoto e anônimo para identificar riscos e buscar convergência de opinião. Um facilitador seleciona os participantes (stakeholders do projeto), distribui as informações e recebe, individual e anonimamente, listas de riscos. Em seguida, consolida as entradas (unificando descrições equivalentes) e redistribui a lista para revisão e complementos. O processo ocorre em rodadas sucessivas (tipicamente 2–4) até a saturação, ou seja, quando não surgem novidades relevantes. O anonimato reduz viés hierárquico e pressão de pares; uma variação inclui especialistas emitindo recomendações técnicas específicas sobre os riscos do projeto (Salles Jr., 2010).

2.1.5.3 Análise dos riscos

Categorias de riscos

A Estrutura Analítica de Riscos (EAR) organiza, de forma hierárquica, todas as fontes de risco do projeto (nível 0) em grandes categorias (nível 1 — técnico, de gerenciamento, comercial e externo) e em subcategorias (nível 2 — p.ex., escopo, requisitos, estimativas; comunicação, recursos; fornecedores e subcontratos; legislação e câmbio). Essa hierarquia atua como checklist e como mecanismo de categorização por causa raiz, ampliando a cobertura na identificação, facilitando priorização e alocação de responsáveis por especialidade (conforme ao quadro 3) (PMI, 2017).

Definições de probabilidade e impacto dos riscos

As definições de probabilidade e impacto devem ser ajustadas ao contexto e ao apetite/limiares de risco da organização, podendo usar escalas corporativas ou próprias; o número de níveis reflete o detalhamento desejado (3 para simples, 5 para granular). As escalas devem cobrir prazo, custo e desempenho/qualidade e atender tanto a ameaças quanto a oportunidades. O quadro 4, proveniente do PMBOK, apenas ilustra um exemplo de tais escalas. Em conjunto, essas definições fundamentam a matriz probabilidade × impacto e orientam a priorização no registro de riscos (PMI, 2017).

Matriz de probabilidade impacto

Ameaças e oportunidades são tratadas na mesma matriz, interpretando o impacto como negativo para ameaças e positivo para oportunidades. A escala pode ser descritiva

Quadro 3 – Quadro EAR retirada do PMBOK

EAR NÍVEL 0	EAR NÍVEL 1	EAR NÍVEL 2
0. TODAS AS FONTES DE RISCO DO PROJETO	1. RISCO TÉCNICO	1.1 Definição do escopo
		1.2 Definição dos requisitos
		1.3 Estimativas, premissas e restrições
		1.4 Processos técnicos
		1.5 Tecnologia
		1.6 Interfaces técnicas
		Etc.
	2. RISCO DE GERENCIAMENTO	2.1 Gerenciamento de projetos
		2.2 Gerenciamento de portfólio
		2.3 Gerenciamento de operações
		2.4 Organizações
		2.5 Recursos
		2.6 Comunicações
		Etc.
	3. RISCO COMERCIAL	3.1 Termos e condições do contato
		3.2 Aquisição interna
		3.3 Fornecedores e prestadores de serviços
		3.4 Subcontratos
		3.5 Estabilidade do cliente
		3.6 Parcerias e joint ventures
		Etc.
	4. RISCO EXTERNO	4.1 Legislação
		4.2 Taxas de câmbio
		4.3 Local/Instalações
		4.4 Meio ambiente/clima
		4.5 Concorrência
		4.6 Regulamentação
		Etc.

Fonte: PMI (2017).

(muito baixo–muito alto) ou numérica; neste último caso, costuma-se calcular uma pontuação multiplicando probabilidade por impacto ($P \times I$) e classificar por faixas de prioridade. A figura 3 ilustra um exemplo de matriz e esquema de pontuação (PMI, 2017).

Análise Swot

A análise SWOT consiste em categorizar riscos com base em fatores externos e internos a organização/projeto, forças (strengths) ou fraquezas (weakness), internos, oportunidade (Opportunities) ou ameaças (Threats), externos (Salles Jr., 2010).

- **Forças:** Recursos e habilidades que a equipe possui, a fim de aproveitar melhor oportunidades e reduzir ameaças. São fatos.

Quadro 4 – Quadro Pxl qualitativa retirada do PMBOK

ESCALA	PROBABILIDADE	+/- IMPACTO SOBRE OBJETIVO DO PROJETO		
		TEMPO	CUSTO	QUALIDADE
Muito alto	>70%	>6 meses	>US\$ 5 milhões	Impacto muito significativo sobre a funcionalidade geral
Alto	51 - 70%	3 - 6 meses	US1M – US 5M	Impacto significativo sobre a funcionalidade geral
Médio	31 - 50%	1 - 3 meses	US\$ 501.000 - US\$ 1 milhão	Algum impacto em áreas funcionais essenciais
Baixo	11 - 30%	1 - 4 semanas	US\$ 100.000 - US\$ 500.000	Impacto secundário sobre a funcionalidade geral
Muito baixo	1 - 10%	1 semana	<US\$ 100.000	Impacto secundário sobre funções secundárias
Nulo	<1%	Sem mudança	Sem mudança	Nenhuma mudança em funcionalidade

Fonte: PMI (2017).

Figura 3 – Tabela Pxl quantitativa retirada do PMBOK

		Ameaças					Oportunidades					
		Muito baixo	Baixo	Moderado	Alto	Muito alto	Muito alto	Alto	Moderado	Baixo	Muito baixo	
Probabilidade	Muito alta 0.90	0.05	0.09	0.18	0.36	0.72	0.72	0.36	0.18	0.09	0.05	Muito alta 0.90
	Alta 0.70	0.04	0.07	0.14	0.28	0.56	0.56	0.28	0.14	0.07	0.04	Alta 0.70
	Média 0.50	0.03	0.05	0.10	0.20	0.40	0.40	0.20	0.10	0.05	0.03	Média 0.50
	Baixa 0.30	0.02	0.03	0.06	0.12	0.24	0.24	0.12	0.06	0.03	0.02	Baixa 0.30
	Muito baixa 0.10	0.01	0.01	0.02	0.04	0.08	0.08	0.04	0.02	0.01	0.01	Muito baixa 0.10
		Impacto negativo					Impacto positivo					

Fonte: PMI (2017).

- **Fraquezas:** Deficiências que atrapalham no aproveitamento de oportunidades, dificultando a capacidade de minimizar ameaças. São fatos que devem ser superados.
- **Oportunidades:** Riscos com consequências potencialmente positivas para o projeto.
- **Ameaças:** Riscos com consequências potencialmente negativas para o projeto.

Na representação gráfica acima, é feito a interseção de fatores internos (Forças/Fraquezas) com fatores externos (Oportunidades/Ameaças), formando quatro quadrantes que orientam estratégias de ação:

	Oportunidades	Ameaças
Forças		
Fraquezas		

Figura 4 – Análise SWOT

- Oportunidades/Forças: A equipe está preparada para tirar proveito da oportunidade;
- Oportunidades/Fraquezas: A equipe não está preparadas para alavancar a oportunidade;
- Ameaças/Forças: A equipe está preparada para enfrentar a ameaça;
- Ameaças/Fraquezas: A equipe não está preparada para enfrentar a ameaça.

2.1.5.4 Monitoramento e controle dos riscos

O ato de monitorar os riscos em um projeto é fundamental para o projeto evoluir de maneira saudável e cautelosa. O monitoramento deve ser constante ao longo de sua vida, havendo um acompanhamento dos riscos já levantados juntamente com a identificação e análise de novos riscos que podem ser identificados. Esse estudo auxilia na tomada de decisões com base nas informações levantadas até o momento, fornecendo uma base para decisões mais seguras e alinhadas aos objetivos do projeto (PMI, 2017).

Durante o processo de monitoramento de riscos, é usado informações de desempenho do projeto para determinar se (PMI, 2017):

- As respostas a riscos implementadas são efetivas;
- O nível geral de risco do projeto sofreu alterações;
- O status dos riscos individuais identificados do projeto sofreu alterações;
- Surgiram novos riscos individuais do projeto;
- A abordagem do gerenciamento dos riscos ainda é a apropriada;

- As premissas do projeto ainda são válidas;
- As políticas e os procedimentos de gerenciamento dos riscos estão sendo seguidos;
- As reservas de contingência para custos ou cronograma devem ser modificadas;
- A estratégia do projeto ainda é válida.

Para realizar o monitoramento, são analisados documentos produzidos ao longo do projeto, alguns exemplos são: Registro das questões, para procurar informações se quaisquer questões pendentes foram atualizadas e necessitam de alguma manutenção no risco; Registro dos riscos, contendo o a descrição detalhada de todos os riscos identificados até o momento; Relatório de riscos, incluindo dados da exposição atual do riscos, assim como as estratégia de resposta ao risco definida anteriormente (PMI, 2017).

Durante o processo, podem haver auditorias e reuniões dos riscos. As auditorias de riscos avaliam a eficácia do gerenciamento de riscos; cabe ao gerente do projeto garantir sua realização na frequência prevista no plano, podendo ocorrer nas revisões rotineiras ou em reuniões específicas, com formato e objetivos definidos previamente. As revisões de riscos periódicas avaliam e documentam a eficácia das respostas, identificam novos riscos, reavaliam os atuais, encerram os defasados e registram questões e lições aprendidas, ocorrendo em reuniões de status ou dedicadas conforme o plano.

Como resultado do monitoramento de riscos, pode haver solicitações de mudança em componentes do plano de gerenciamento do projeto, como as linhas de base de custos e do cronograma. Também podem ser atualizados o registro de premissas, o registro de questões, o registro de lições aprendidas, o registro de riscos e o relatório de riscos.

2.2 TECNOLOGIAS E CONCEITOS

2.2.1 Gerais

Protocolo de Transferência de Hipertexto (HTTP)

O Protocolo de Transferência de Hipertexto (HTTP) é um protocolo que organiza a troca de informações entre cliente e servidor, no modelo requisição-resposta. Ele permite solicitar páginas, arquivos e dados de APIs e receber as respectivas respostas. HTTPS é o mesmo protocolo operando com uma camada de segurança, que protege os dados em trânsito (MDN Web Docs, 2025c).

Interface de Programação de Aplicações (API)

Uma Interface de Programação de Aplicações (API) é um contrato de interface que especifica como sistemas de software se comunicam, formatos de dados e regras de acesso. No contexto da Web, APIs são expostas pelo back-end e consumidas por

clientes via HTTP/HTTPS. Por analogia, funciona como um “garçom” que recebe pedidos (requisições), encaminha-os ao serviço apropriado e entrega as respostas; formalmente, trata-se de uma interface padronizada entre cliente e servidor (MDN Web Docs, 2025f).

Frameworks

No lado do cliente, frameworks e bibliotecas fornecem uma base de componentes, padrões e ferramentas para construir interfaces de forma previsível, manutenível e consistente. Eles organizam a aplicação em componentes reutilizáveis, tratam roteamento no navegador, auxiliam no gerenciamento de estado e padronizam tarefas comuns (formulários, testes), reduzindo código acidental e divergência de estilos (MDN Web Docs, 2025d). No contexto deste trabalho, o front-end legado do Taiga utiliza AngularJS; no back-end, emprega-se Django (fora do escopo da referência citada, centrada em frameworks do lado do cliente).

Docker

Docker empacota aplicações em contêineres, tornando-as portáteis e reproduzíveis entre máquinas. As imagens são definidas por Dockerfile e executadas pelo Docker Engine; quando há vários serviços, o Docker Compose descreve tudo em um arquivo, simplificando a construção, execução e integração entre os componentes (aplicação, banco, broker, servidor web) (Docker Inc, 2025).

2.2.2 Back-end

No contexto de sistemas web, o back-end é a camada de servidor responsável pela lógica de negócio, acesso a dados e integrações, expondo funcionalidades por meio de APIs sobre protocolos como HTTP/HTTPS. Também abrange autenticação e autorização, validações, orquestração de processos e tarefas assíncronas, contribuindo para a segurança, disponibilidade e desempenho da aplicação (MDN Web Docs, 2025e).

A seguir, apresentam-se as tecnologias de back-end utilizadas pelo Taiga, necessárias ao desenvolvimento deste trabalho.

Python

Python é uma linguagem de alto nível, interpretada e de tipagem dinâmica, criada por Guido van Rossum no final de 1989 e lançada publicamente em 1991. Possui múltiplos paradigmas de programação: orientação a objetos, imperativo e funcional, detém coleta automática de lixo e uma biblioteca padrão abrangente, além de ecossistema de pacotes mantido via *pip*/PyPI (Python Software Foundation, 2025).

Amplamente adotada no mercado, no contexto deste trabalho, Python compõe a camada de backend do Taiga, implementada com o framework web Django, que é

responsável pelo processamento de dados e pelas respostas às requisições dos clientes. Dessa forma, é necessária a implementação de novos módulos, seguindo o mesmo padrão de projeto definido pelo Taiga para atingir os objetivos deste trabalho.

Web Server Gateway Interface

O Web Server Gateway Interface (WSGI) é uma especificação definida pela PEP 3333 (2010) que padroniza a interface entre servidores web e aplicações (e middlewares) escritas em Python. A PEP 3333 atualiza a PEP 333 para o ecossistema do Python 3 e não fornece uma implementação; em vez disso, descreve o contrato que deve ser seguido para garantir portabilidade e desacoplamento entre servidor e aplicação. Diversos servidores compatíveis implementam essa especificação — como Gunicorn (utilizado neste trabalho), uWSGI e Waitress — enquanto frameworks como Flask e Django expõem a aplicação como um objeto WSGI (callable), permitindo que qualquer servidor aderente execute a mesma aplicação sem modificações (Eby, 2010).

Gunicorn

O Gunicorn (Green Unicorn) é um servidor HTTP WSGI para sistemas do tipo Unix. Ele implementa o lado servidor da especificação WSGI, recebendo requisições e encaminhando-as ao callable WSGI exposto pela aplicação (por exemplo, em Django, Flask ou Pyramid). Sua arquitetura prefork utiliza um processo árbitro que coordena múltiplos workers, cada um executando a aplicação em processo separado. Esse modelo favorece isolamento de falhas, melhor aproveitamento de múltiplos núcleos e escalabilidade por paralelismo de processos. Na prática, isso permite atender alto volume de requisições com baixo acoplamento entre servidor e aplicação, mantendo configuração simples e footprint contido (Chesneau, 2024).

Django

Django é um framework web em Python voltado à produtividade, segurança e organização do código. Segue o padrão MVT (Model–View–Template), que separa dados, lógica e apresentação, facilitando manutenção e evolução. Traz recursos prontos para tarefas comuns (autenticação, formulários, painel administrativo e roteamento), o que acelera o desenvolvimento com menos código e boas práticas incorporadas. Para execução em produção, o Django expõe uma aplicação compatível com WSGI e integra-se a servidores como o Gunicorn, que recebem as requisições da web e as repassam para a aplicação (Foundation, 2025).

Celery

Celery é uma fila de tarefas distribuída que usa um message broker (serviço que intermedia a troca, enfileira e encaminha mensagens entre a aplicação e os workers), para trafegar mensagens, permitindo processamento assíncrono em tempo quase real e agendamento de tarefas (via Celery Beat). Os workers executam as tarefas fora do fluxo da requisição web, aliviando a aplicação principal (Solem, 2023).

Banco de Dados

Banco de dados é o conjunto de dados organizados e gerenciados por um sistema gerenciador de banco de dados (SGBD), que permite armazenar, consultar, atualizar e remover informações com persistência, controle de concorrência e segurança. Entre os modelos mais utilizados está o relacional, que organiza os dados em tabelas relacionadas por chaves e utiliza SQL, oferecendo transações e propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Há também famílias não relacionais (por exemplo, documentos, chave-valor, colunar e grafos), apropriadas para certos cenários de flexibilidade de esquema e escala horizontal (Microsoft, 2025).

No Taiga, adota-se um banco de dados relacional, especificamente o PostgreSQL, amplamente utilizado no mercado e compatível com frameworks Python como o Django. O PostgreSQL provê integridade referencial, índices avançados e suporte a JSONB (útil para dados semiestruturados), o que favorece manutenibilidade e confiabilidade no armazenamento das informações da aplicação (The PostgreSQL Global Development Group, 2025).

2.2.3 Front-end

O front-end é a camada de apresentação que executa no cliente (navegador), responsável por renderizar a interface, lidar com a interação do usuário e consumir APIs do back-end sobre HTTP/HTTPS. Geralmente envolve HTML, CSS e JavaScript (com utilização de frameworks que aumentam o desempenho e facilitam seu desenvolvimento), uso de Web APIs do navegador, além de preocupações como acessibilidade, desempenho de carregamento e responsividade (MDN Web Docs, 2025h; Sanity, 2024).

A seguir, apresentam-se as tecnologias de front-end utilizadas pelo Taiga, necessárias ao desenvolvimento deste trabalho.

HTML

O HTML (HyperText Markup Language) define a estrutura e a semântica do conteúdo na Web, organizando textos, imagens e mídias em elementos (como títulos, parágrafos, listas e links) que podem ser interpretados pelos navegadores (MDN Web Docs, 2025b).

CSS

O CSS (Cascading Style Sheets) descreve a apresentação visual do conteúdo — tipografia, cores, espaçamentos, temas, layouts responsivos e animações —, contribuindo para legibilidade e experiência de uso. Ao separar conteúdo (HTML) de apresentação (CSS), favorece manutenção, reutilização e acessibilidade (W3C, 2025).

DOM

O DOM (Document Object Model) é a representação em árvore do documento carregado pelo navegador, composta por nós (elementos, atributos e textos) e uma interface padronizada para leitura e modificação desse conteúdo. O DOM é construído pelo navegador a partir da análise do HTML e pode ser modificado dinamicamente por scripts (MDN Web Docs, 2025a).

JavaScript

JavaScript é a linguagem de programação padronizada pela especificação ECMAScript, criada em 1995 para execução no navegador. No front-end, complementa HTML e CSS ao prover interatividade e lógica de aplicação no cliente, tendo se consolidado como peça central da Web moderna (MDN Web Docs, 2025g).

Tecnicamente, permite manipular o DOM, tratar eventos, realizar comunicações assíncronas, além de acessar diversas Web APIs. O ecossistema inclui frameworks e bibliotecas como React, Angular e Vue auxiliam a estruturar interfaces, gerenciar estado, organizar rotas e melhorar a produtividade e a manutenibilidade.

Node.js é um ambiente de execução (runtime) que permite executar JavaScript no servidor (fora do navegador). Baseado no motor V8, viabiliza a criação de serviços, scripts e APIs em JavaScript, mantendo a mesma linguagem no front-end e no back-end. Em conjunto, o npm (Node Package Manager) atua como gerenciador de pacotes do ecossistema, facilitando a instalação e o gerenciamento de dependências, além da execução de scripts que apoiam o ciclo de desenvolvimento (OpenJs Foundation, s.d.), (Npmjs, s.d.).

Angular

Angular é uma plataforma e framework para construção de aplicações web de página única (SPA). Baseado em componentes e templates, adota TypeScript como linguagem principal e oferece recursos nativos como injeção de dependências, roteamento, módulos, formularização (Template-Driven e Reactive Forms), HttpClient, detecção de mudanças e um toolchain integrado via Angular CLI (geração de código, testes, lint, build e bundling). O ecossistema incorpora o uso de RxJS (Reactive Extensions for JavaScript) para programação reativa, favorecendo composição de fluxos assíncronos e maior previsibilidade na interface.

É importante distinguir Angular (2+) de AngularJS (1.x): o primeiro reescreve a base em TypeScript e componentes, enquanto o segundo é anterior e organiza a aplicação em controllers e scopes. No contexto deste trabalho, o Taiga utiliza um frontend legado em AngularJS (1.x), historicamente combinado com CoffeeScript e Pug, o que explica a presença desses artefatos no repositório; ainda assim, os conceitos de separação de responsabilidades e componentes inspiram a estruturação moderna adotada por frameworks atuais.

Pug

Pug é um motor de templates para JavaScript/Node que usa sintaxe por indentação e gera HTML. Facilita páginas dinâmicas com interpolação, parciais e layouts reutilizáveis, integrando-se facilmente ao Express. Por questões de marca registrada, o projeto Jade foi renomeado para Pug; em projetos legados (como o Taiga), é comum encontrar arquivos com extensão `.jade`, que correspondem a templates do ecossistema Jade/Pug (Pugjs.org, s.d.).

CoffeeScript

CoffeeScript é uma linguagem que compila para JavaScript, oferecendo sintaxe mais concisa e recursos como funções com seta (`->/=>`), compreensões, classes e indentação significativa, com o objetivo de tornar o código mais legível e produtivo. O resultado da compilação é JavaScript padrão, o que permite interoperar com qualquer biblioteca ou framework do ecossistema e executar o código em qualquer ambiente onde JavaScript executa (navegador ou Node.js) (Coffeescript.org, s.d.).

3 TRABALHOS RELACIONADOS

Este capítulo analisa ferramentas correlatas que embasam e complementam o estudo, com foco nas funcionalidades de gerência de cronograma, riscos e custos em sistemas de gerenciamento de projetos. A análise explora como essas funcionalidades são implementadas em diferentes ferramentas, o que permite identificar boas práticas, lacunas e oportunidades de melhoria que fundamentam a proposta de desenvolvimento apresentada.

3.1 DEFINIÇÃO DO PROTOCOLO DE PESQUISA

A busca pelas ferramentas envolve a identificação e análise de sistemas de gerenciamento de projetos que apresentam funcionalidades de gerenciamento de cronograma, riscos e custos, fornecendo base para a definição das funcionalidades a serem desenvolvidas. Para garantir a abrangência e a consistência metodológica da pesquisa, definem-se termos e expressões-chave que representam os principais conceitos do estudo. Com base nesses termos, constroem-se strings de busca otimizadas, aplicadas em diferentes motores de busca, permitindo a identificação de ferramentas relevantes, atuais e alinhadas às práticas de gestão ágil de projetos. Além disso, estabelecem-se critérios de inclusão e exclusão para refinar e filtrar os resultados obtidos.

3.1.1 Critérios de Inclusão e Exclusão

Os critérios de inclusão e exclusão têm o propósito de garantir coerência metodológica e relevância prática às ferramentas identificadas durante a pesquisa. Esses critérios orientam o processo de filtragem dos resultados, assegurando que apenas soluções alinhadas aos objetivos deste estudo — isto é, ferramentas com funcionalidades voltadas à gerência de cronograma, riscos e custos — sejam consideradas para análise detalhada.

São incluídas na pesquisa as ferramentas que: (a) adotam métodos ágeis de gerenciamento, como *Scrum*, *Kanban* ou *Scrumban*; (b) oferecem módulos ou recursos relacionados a pelo menos uma das três áreas principais de interesse (cronograma, riscos ou custos); (c) estão disponíveis em versão gratuita, de teste ou de código aberto; e (d) possuem site ativo, o que permite validação das informações e análise funcional.

Por outro lado, são excluídas as ferramentas que: (a) não apresentam características específicas de gerenciamento de projetos, limitando-se a controle de tarefas ou comunicação; (b) não oferecem acesso público, documentação técnica ou demonstração funcional; ou (c) se encontram descontinuadas ou inativas há longos períodos, comprometendo sua aplicabilidade e relevância.

Esses critérios são aplicados de forma consistente em todas as plataformas de busca utilizadas, assegurando comparabilidade entre os resultados e homogeneidade na amostra final de ferramentas analisadas.

3.1.2 Plataformas de Busca

As buscas são realizadas em diferentes plataformas, com o objetivo de diversificar as fontes de informação e aumentar a confiabilidade dos resultados obtidos. A utilização de múltiplos mecanismos permite validar e complementar os dados coletados, uma vez que cada plataforma possui critérios próprios de indexação e classificação de conteúdo.

As plataformas selecionadas são o Google¹, o ChatGPT² e o Gemini³. O Google é utilizado como principal fonte, por reunir ampla cobertura de sites institucionais, artigos técnicos e repositórios de código. As pesquisas são realizadas em modo de navegação anônima, de modo a reduzir o impacto de histórico e personalização nos resultados.

O ChatGPT e o Gemini são empregados como ferramentas complementares, permitindo a identificação de soluções emergentes e comparativos de mercado por meio de respostas sintetizadas a partir de múltiplas fontes verificadas.

Durante o processo de coleta, os resultados das três plataformas passam por duas etapas de filtragem. A primeira consiste na leitura dos títulos e descrições das páginas retornadas, para eliminar links irrelevantes ou duplicados. A segunda envolve a análise das páginas principais das ferramentas, verificando a presença de recursos relacionados à gestão de cronograma, custos e riscos.

3.1.3 Termos de Busca

Os termos de busca representam a base conceitual para a formulação das estratégias de pesquisa utilizadas nesta etapa. Eles são definidos a partir das principais áreas de interesse do estudo — cronograma, custos e riscos — associadas ao contexto de gerência ágil de projetos.

A organização dos termos segue uma estrutura de categorias conceituais, que agrupam os eixos temáticos centrais da pesquisa, seus respectivos termos principais e sinônimos mais recorrentes na literatura e em documentações técnicas. Essa abordagem visa garantir abrangência sem perda de precisão, permitindo a recuperação de resultados que utilizem diferentes denominações para o mesmo conceito.

O Quadro 5 apresenta os principais termos e sinônimos utilizados na formulação das strings de busca.

3.1.4 Construção da String e dos Prompts de Busca

A partir dos termos definidos na etapa anterior, são elaboradas strings de busca com o objetivo de combinar os conceitos principais e seus sinônimos por meio de operadores booleanos. Essa técnica permite estruturar a pesquisa de forma que os resultados retornem ferramentas que abordem simultaneamente os eixos temáticos definidos, como a gerência

¹ <https://www.google.com/>

² <https://chat.openai.com>

³ <https://gemini.google.com>

Quadro 5 – Termos e sinônimos de busca

Categoria	Termo principal	Termos relacionados / sinônimos
Project management tool	Project management	Project tracking, project planning, project software
Approaches	Agile	Scrum, Kanban, Scrumban, Agile methodologies
Features	Risk management, Schedule, Cost management	Risk control, risk monitoring, risk mitigation, timeline, time tracking, Gantt chart, milestone planning, budgeting, expense tracking, financial control, cost estimation

Fonte: Elaborado pelo autor (2025).

de cronograma, riscos e custos em contextos ágeis. A formulação das strings segue a lógica de interseção e variação de termos. O operador AND conecta os conceitos centrais, enquanto o operador OR agrupa sinônimos e variações linguísticas de um mesmo termo.

Com o objetivo de priorizar plataformas e softwares de gerenciamento de projetos, foram incorporados termos direcionadores como *tool*, *software* e *platform*, utilizados em conjunto com os termos principais de busca do quadro 5, que orientam os mecanismos de busca a retornarem resultados referentes a produtos e sistemas.

A busca é realizada diretamente no mecanismo de pesquisa do Google, em modo de navegação anônima, a fim de minimizar a interferência de histórico e personalização de resultados. A string de busca aplicada segue a seguinte estrutura:

("project management tool"OR "project management software"OR "agile platform") AND ("agile"OR "kanban"OR "scrum"OR "scrumban") AND (risk management OR risk control OR schedule OR time management OR cost OR budget)

Para garantir consistência entre as diferentes plataformas utilizadas, a mesma lógica da string é também convertida em um prompt textual, aplicado em motores de busca baseados em inteligência artificial, como o ChatGPT e o Gemini. Prompt adaptado:

Find project management tools, project management software, and agile platforms that implement Agile, Kanban, Scrum, or Scrumban practices and include features related to risk management, risk control, scheduling, time management, cost management, or budget management. For each solution, provide its official website and a brief summary of its main features.

Essa versão mantém a estrutura lógica da string original, mas em formato semântico, possibilitando interpretação contextual mais ampla pelos mecanismos de busca baseados em inteligência artificial.

3.1.5 Definição da Extração dos Dados

Para cada ferramenta identificada, é extraído um conjunto padronizado de informações, definido previamente para assegurar uniformidade, comparabilidade e consistência entre as análises. Para isso, adota-se um modelo estruturado que orienta o processo e garante que todas as ferramentas sejam avaliadas segundo os mesmos critérios, permitindo análises equivalentes e reproduzíveis.

1. **Licenciamento:** identificar o tipo de licença oferecida.
2. **Funcionalidades Principais:** listar os módulos e recursos disponibilizados, com foco em funcionalidades de cronograma, gestão de risco, orçamento, documentação e colaboração.
3. **Modelo de Cobrança e Limitações:** descrever os planos existentes, as restrições da versão gratuita e os recursos acessíveis exclusivamente em versões pagas.
4. **Recursos de Gestão de Riscos:** verificar se a ferramenta oferece suporte para identificação, categorização, análise qualitativa, análise quantitativa e monitoramento contínuo de riscos.
5. **Recursos de Planejamento do Cronograma:** avaliar a presença de funcionalidades como gráficos de Gantt, cronogramas, linhas do tempo, dependências, sprints e backlog.
6. **Recursos de Gestão de Custos:** identificar mecanismos de apoio ao planejamento orçamentário, acompanhamento de custos planejados, custos realizados e métricas relacionadas.
7. **Observações Relevantes:** registrar limitações observadas, dificuldades de uso, recursos parcialmente disponíveis e qualquer outro aspecto pertinente identificado durante a análise.

3.2 EXECUÇÃO DA BUSCA

A execução da busca segue o protocolo definido na Seção 3.1 e utiliza, de forma combinada, consultas tradicionais na web e consultas mediadas por modelos de linguagem. O objetivo é identificar ferramentas de software relacionadas à gestão de riscos, cronograma e custos, selecionando apenas soluções pertinentes ao escopo deste estudo. Esta seção descreve como foram aplicadas as estratégias de busca, como foi realizada a filtragem dos resultados e como foram consolidadas as ferramentas identificadas.

3.2.1 Aplicação das Strings de Busca

No Google, em janela anônima, a busca padrão foi executada e as primeiras 5 páginas de resultados retornados percorridas manualmente, observando títulos, descrições e links associados a possíveis ferramentas de software.

Nas plataformas baseadas em modelos de linguagem — ChatGPT e Gemini — não foi utilizada a string estruturada. Em vez disso, foi aplicado diretamente o prompt em linguagem natural definido na Seção 3.1.4. Para cada modelo, o prompt completo foi enviado, e todas as sugestões de ferramentas apresentadas na resposta inicial foram registradas.

3.2.2 Processo de Seleção dos Resultados

Após a coleta inicial em cada plataforma, foi iniciado o processo de seleção dos resultados. Foram aplicados os critérios de inclusão e exclusão definidos na Seção 3.1.1, descartando itens redundantes, artigos científicos, artigos em blogs, notícias, tutoriais e quaisquer conteúdos que não caracterizam uma ferramenta efetiva de software.

Para padronizar o volume de dados analisados, foram selecionadas até vinte ferramentas provenientes de cada fonte (Google, GPT e Gemini), totalizando sessenta entradas iniciais. Em seguida, foram comparadas manualmente todas as entradas, identificando duplicações de nomes e URLs. Quando a mesma ferramenta apareceu em mais de uma plataforma, foi preservado apenas um único registro representativo para análise posterior.

3.2.3 Consolidação dos Itens Encontrados

Com os itens selecionados, todas as ferramentas foram organizadas em uma planilha unificada contendo nome, descrição resumida e link oficial. Nesta etapa, foram removidas duplicatas remanescentes, foi padronizada a grafia dos nomes e foi verificada a disponibilidade atual de cada ferramenta. Após essa consolidação, o conjunto inicial de sessenta ferramentas passou a totalizar quarenta e duas entradas válidas.

A partir da descrição de cada ferramenta, foi avaliada sua pertinência em relação aos domínios de interesse deste trabalho, ou seja, se existem alguma funcionalidade que envolvem riscos, cronograma ou custos. Com base nessa análise, o conjunto consolidado foi reduzido para as oito ferramentas que apresentam maior aderência aos critérios definidos na Seção 3.1.5. Essas oito soluções constituem o conjunto selecionado para análise aprofundada na Seção 3.3.

3.3 RESULTADOS

Esta seção apresenta a análise detalhada das ferramentas selecionadas após o processo de execução e consolidação da busca. Cada ferramenta é descrita seguindo os critérios definidos na Seção 3.1.5, permitindo a comparação estruturada entre as soluções.

3.3.1 Taiga

Descrição Geral

O Taiga é uma plataforma de gerenciamento de projetos orientada a métodos ágeis, especialmente Scrum e Kanban. O sistema é conhecido por sua interface simples e objetiva, que favorece equipes que trabalham em iterações rápidas, oferecendo visualizações claras para backlog, sprints e fluxo de trabalho.

Licenciamento

O Taiga é open-source e pode ser utilizado gratuitamente em ambientes auto-hospedados. A versão em nuvem (Taiga Cloud) oferece um plano gratuito e planos pagos que expandem limites de projetos, armazenamento e suporte. Não há cobrança por usuário.

Funcionalidades Principais

- Backlog completo com histórias e épicos.
- Organização de sprints com quadros Kanban.
- Gráfico de burndown integrado.
- Gestão de tarefas, issues e épicos.
- Integrações com GitHub, GitLab e Bitbucket.
- Suporte a plugins e extensões.

Gestão de Riscos

O Taiga não possui um módulo nativo de gestão de riscos. Entretanto, certos elementos da plataforma podem ser utilizados como indicadores de risco, como a métrica “doses iocaine”, que aponta que um usuário está sobrecarregado ou em uma atividade fora de sua zona de conforto — equivalente a um alerta de risco no processo.

Gestão de Cronograma

Embora não possua Gantt nativo, o Taiga organiza o trabalho por meio de backlog, priorização e ciclos de sprint. Essa abordagem está alinhada a métodos ágeis, mas limita a visualização de planejamento prescritivo (ex.: baseline, dependências, caminho crítico).

Figura 5 – Backlog de projeto no Taiga

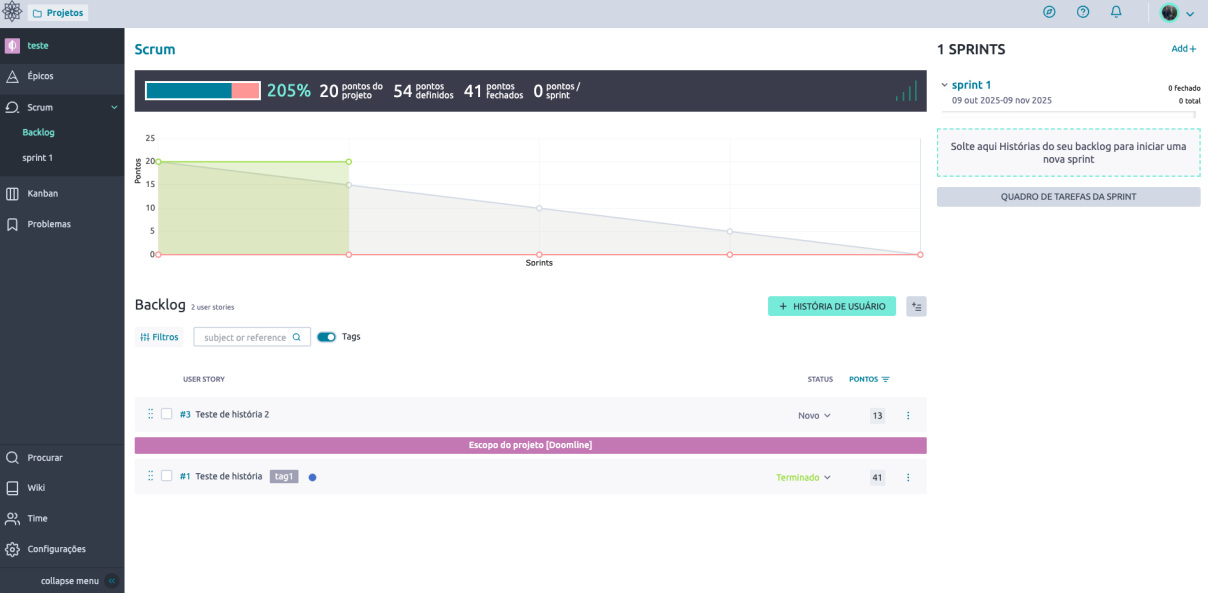
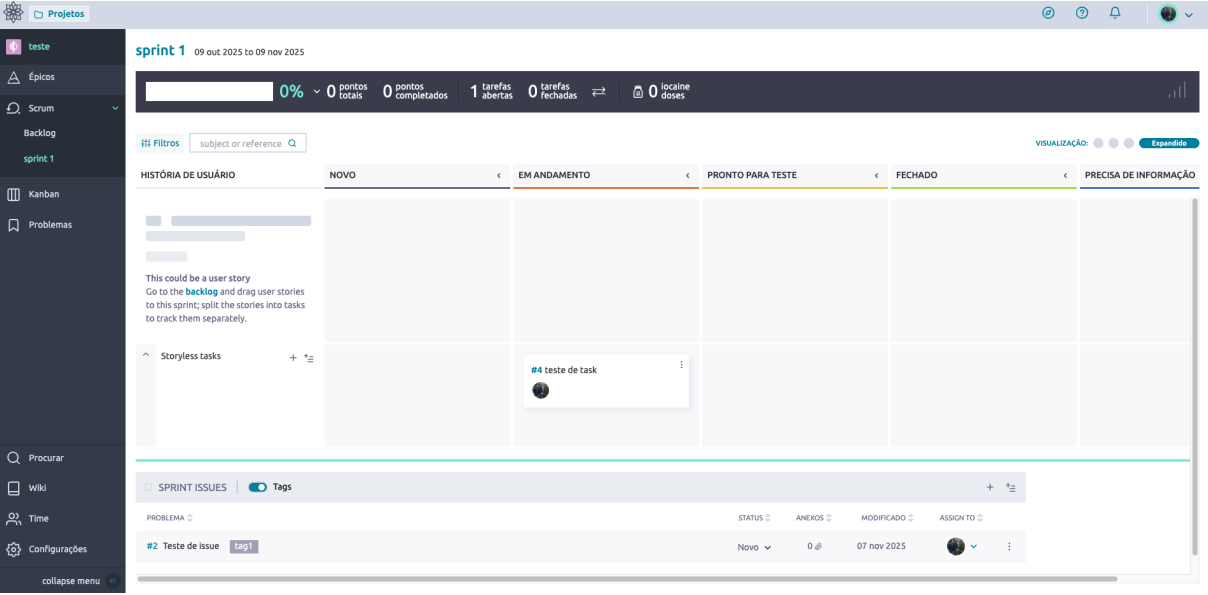


Figura 6 – Detalhamento da sprint no Taiga



Gestão de Custos

O Taiga não possui funcionalidades nativas relacionadas a custos ou orçamento. Qualquer controle financeiro depende de ferramentas externas ou modelagem manual fora da plataforma.

Observações e Limitações

O Taiga é eficiente, pela facilidade de uso e agilidade na organização dos dados de maneira estruturada, para equipes puramente ágeis, mas carece de módulos para risco, custos e cronograma de abordagens prescritivas. Sua simplicidade é tanto uma vantagem quanto uma limitação, dependendo do contexto organizacional.

3.3.2 Monday

Descrição Geral

O Monday é uma plataforma comercial de gerenciamento de projetos com foco em colaboração e visualização. Sua estrutura baseia-se em *boards* personalizáveis, que permitem modelar tarefas, cronogramas, riscos e custos de maneira integrada. A ferramenta prioriza flexibilidade e organização visual, sendo amplamente utilizada em equipes ágeis e ambientes híbridos.

Licenciamento

O Monday opera por meio de planos de assinatura pagos, com um período de teste gratuito de 14 dias. Todos os planos têm limites diferentes de funcionalidades, enquanto recursos avançados exigem assinaturas superiores. O sistema não oferece módulos nativos de gestão financeira ou risco; tais funcionalidades são configuradas manualmente.

Funcionalidades Principais

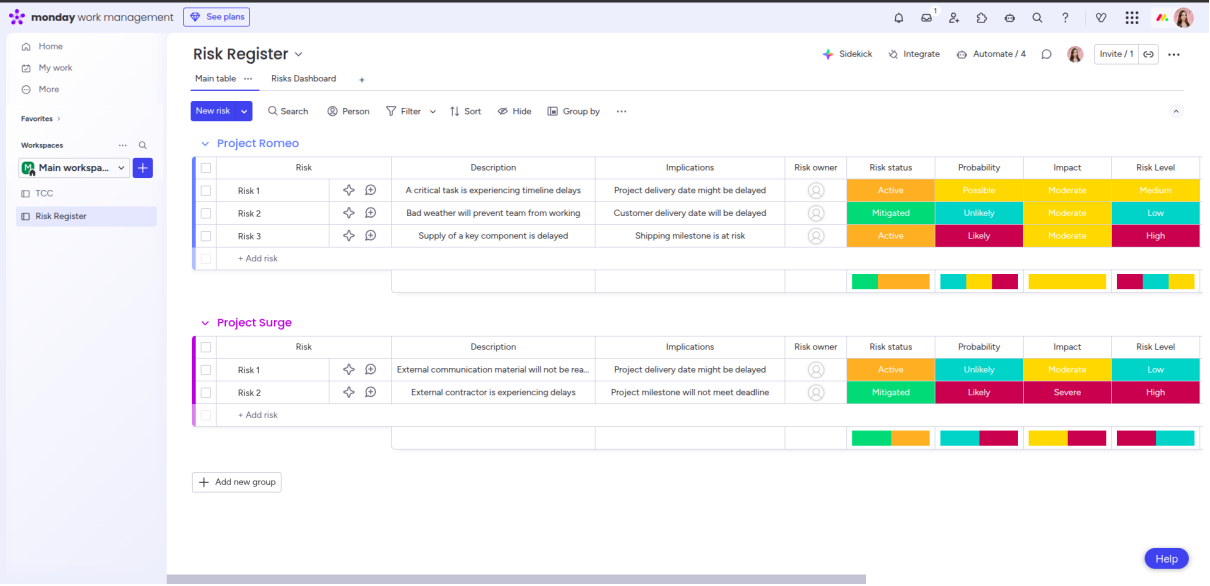
- Estrutura baseada em *boards* altamente personalizáveis.
- Dashboards com indicadores, gráficos e consolidação de dados.
- Automação de fluxos e notificações.
- Integração entre múltiplos boards para visão unificada.
- Visualizações avançadas para acompanhamento do projeto.

Gestão de Riscos

O Monday permite montar registros de riscos (*Risk Registers*) por meio de boards específicos. Cada risco é configurado como um item com campos de probabilidade, impacto, criticidade (calculada via fórmulas), tipo, status e estratégias de mitigação.

A ferramenta não oferece matriz de probabilidade e impacto nativa, tampouco técnicas de análise qualitativa ou quantitativa. Todos os elementos dependem de modelagem do usuário.

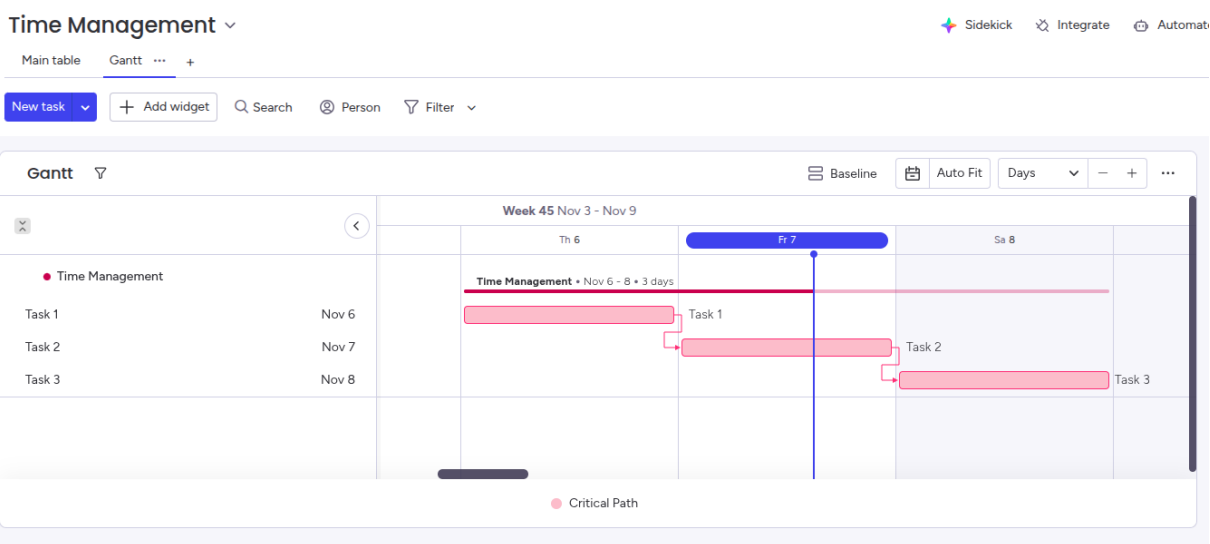
Figura 7 – Registro de riscos configurado no Monday



Gestão de Cronograma

O Monday possui visualizações de Timeline, calendário e gráfico de Gantt, com suporte a dependências, caminho crítico e baseline. As tarefas podem ser vinculadas a boards de custos e riscos por meio de conexões.

Figura 8 – Gestão de cronograma no Monday



Gestão de Custos

A plataforma não contém um módulo financeiro nativo; os custos são representados através de colunas numéricas e de fórmula. É possível criar dashboards financeiros

personalizados para controle de orçamento, custo real, custo planejado e variações.

Figura 9 – Dashboard de gestão de custos no Monday

Budget Plan

Project 1

Item	Notes	Point of con...	Status	Start Date	Resource	Travel	Actual Spend	Budgeted	Difference	On Budget	+
Task 1	Project D...		Done	May 12, '22	\$11,000	\$2,000	\$13,000	\$15,000	\$2,000	✓	
Task 2			Done	May 12, '22	\$700	\$200	\$900	\$1,000	\$100	✓	
Task 3	Resourc...		Working on it	Feb 8, '22	\$6,000	\$1,000	\$7,000	\$6,000	\$-1,000	✗	
					\$17,700 sum	\$3,200 sum	\$20,900 sum	\$22,000 sum	\$1,100 sum		

Project 2

Item	Notes	Point of con...	Status	Start Date	Resource	Travel	Actual Spend	Budgeted	Difference	On Budget	+
Task 1			Working on it	Jun 2, '22	\$1,500	\$400	\$1,900	\$3,000	\$1,100	✓	
Task 2			Stuck	Jun 14, '22	\$2,000	\$700	\$2,700	\$2,600	\$-100	✗	
Task 3			Working on it	Jun 16, '22	\$2,000	\$200	\$2,200	\$2,000	\$-200	✗	
					\$5,500 sum	\$1,300 sum	\$6,800 sum	\$7,600 sum	\$800 sum		

+ Add new group

Help

Observações e Limitações

O Monday é forte em personalização e colaboração, mas carece de formalização metodológica. Recursos de risco e custos dependem inteiramente da modelagem manual, o que limita seu uso para projetos complexos ou ambientes de engenharia e PMO.

3.3.3 OpenProject

Descrição Geral

O OpenProject é uma ferramenta de gerenciamento de projetos amplamente utilizada em ambientes corporativos e institucionais, como enunciado no seu próprio site ao listar o nome de diversas empresas e instituições, destacando-se por sua robustez e por oferecer módulos completos para planejamento, acompanhamento e documentação. É uma solução open-source com forte aderência a boas práticas de PMO (OpenProject, s.d.).

Licenciamento

Sua Community Edition é gratuita e pode ser auto-hospedada sem limites de usuários. As edições Enterprise (Cloud e On-premises) incluem suporte profissional, módulos avançados e recursos adicionais de segurança.

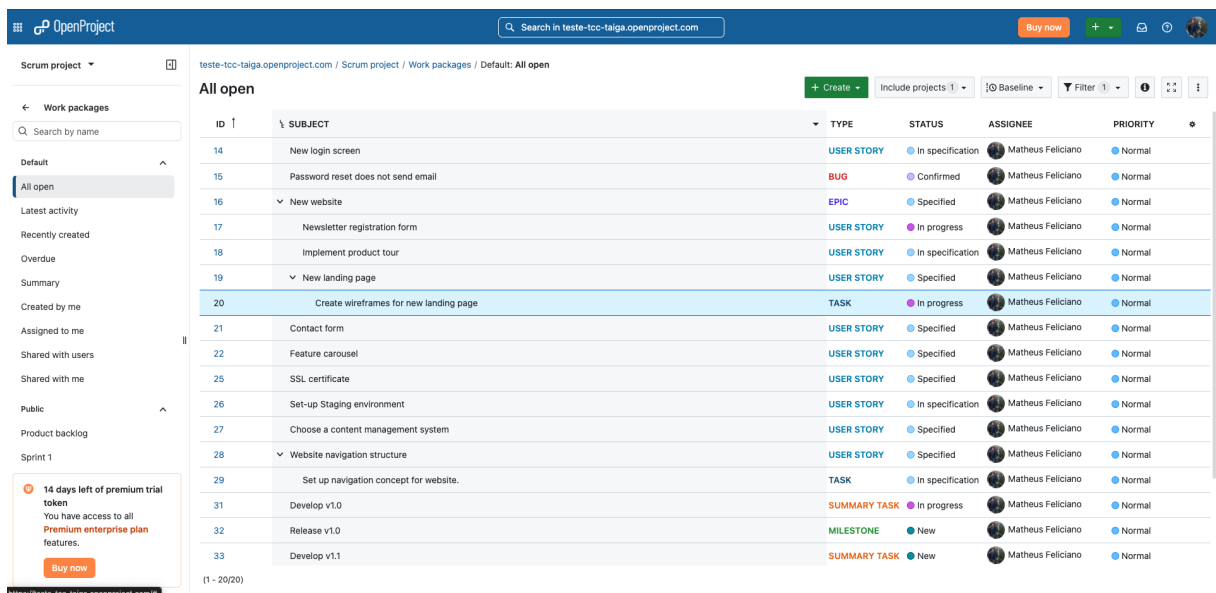
Funcionalidades Principais

- Múltiplos projetos com hierarquia.
- Backlog com épicos, histórias e tarefas.
- Roadmaps e planejamento estratégico.
- Quadros ágeis e burndown chart.
- Gantt avançado (projeto e multi-projeto).
- Team Planner para alocação de equipe.
- Agenda, atas e registro de decisões.
- Módulos de tempo, orçamento e relatórios financeiros.

Gestão de Riscos

O OpenProject não possui módulo nativo de riscos, mas é possível modelar riscos como *work packages* específicos. Na Enterprise Edition, alguns plugins adicionais oferecem funcionalidades avançadas.

Figura 10 – Pacotes de trabalho no OpenProject

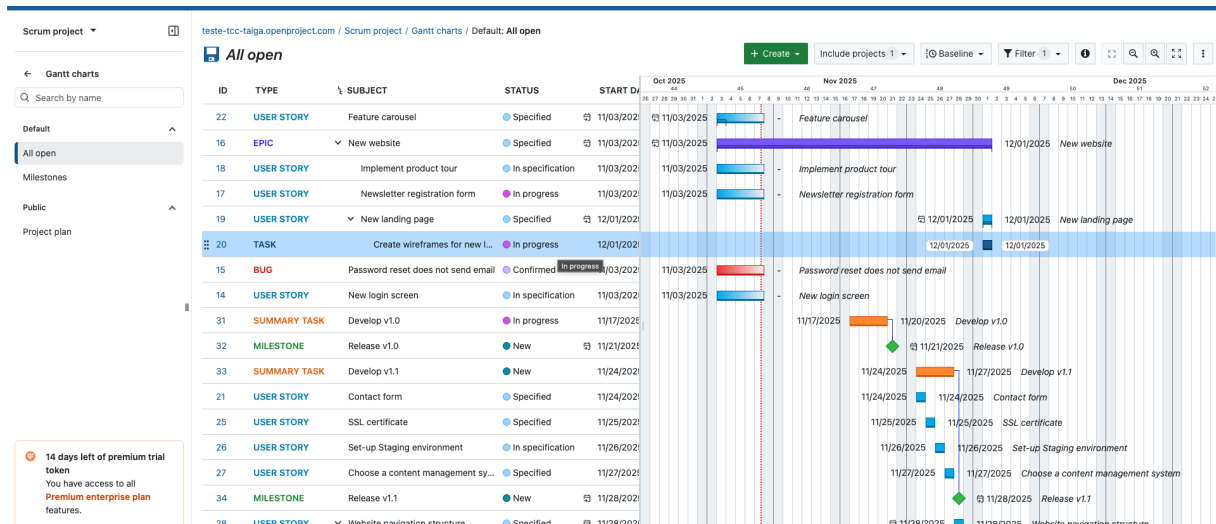


ID	SUBJECT	TYPE	STATUS	ASSIGNEE	PRIORITY
14	New login screen	USER STORY	In specification	Matheus Feliciano	Normal
15	Password reset does not send email	BUG	Confirmed	Matheus Feliciano	Normal
16	New website	EPIC	Specified	Matheus Feliciano	Normal
17	Newsletter registration form	USER STORY	In progress	Matheus Feliciano	Normal
18	Implement product tour	USER STORY	In specification	Matheus Feliciano	Normal
19	New landing page	USER STORY	Specified	Matheus Feliciano	Normal
20	Create wireframes for new landing page	TASK	In progress	Matheus Feliciano	Normal
21	Contact form	USER STORY	Specified	Matheus Feliciano	Normal
22	Feature carousel	USER STORY	Specified	Matheus Feliciano	Normal
25	SSL certificate	USER STORY	Specified	Matheus Feliciano	Normal
26	Set-up Staging environment	USER STORY	In specification	Matheus Feliciano	Normal
27	Choose a content management system	USER STORY	Specified	Matheus Feliciano	Normal
28	Website navigation structure	USER STORY	Specified	Matheus Feliciano	Normal
29	Set up navigation concept for website.	TASK	In specification	Matheus Feliciano	Normal
31	Develop v1.0	SUMMARY TASK	In progress	Matheus Feliciano	Normal
32	Release v1.0	MILESTONE	New	Matheus Feliciano	Normal
33	Develop v1.1	SUMMARY TASK	New	Matheus Feliciano	Normal

Gestão de Cronograma

A ferramenta possui um dos Ganttts mais completos entre as analisadas: suporta dependências, marcos, baseline, caminho crítico e gerenciamento multi-projeto. O planejamento é integrado a pacotes de trabalho e responsivos às atualizações.

Figura 11 – Gráfico de Gantt do OpenProject



Gestão de Custos

O OpenProject fornece módulos completos de custos, incluindo orçamento, custos planejados, custos reais, acompanhamento por horas e relatórios financeiros. A Enterprise Edition amplia a quantidade de relatórios e integrações possíveis.

Observações e Limitações

O OpenProject é a ferramenta mais completa dentre as analisadas, mas parte dos módulos avançados é paga e exclusiva da Enterprise Edition. Apesar disso, a versão gratuita já atende a grande parte das necessidades metodológicas formais.

3.3.4 ProjectManager

Descrição Geral

O ProjectManager é uma plataforma comercial de gerenciamento de projetos disponibilizada como serviço em nuvem (*SaaS*), voltada ao planejamento, acompanhamento e colaboração em equipes distribuídas. A ferramenta oferece uma combinação de visualizações como gráfico de Gantt, quadro Kanban, listas, planilhas e calendários, permitindo que o mesmo projeto seja analisado sob diferentes perspectivas, com atualização em tempo real do progresso, cargas de trabalho e custos.

Licenciamento

O ProjectManager adota um modelo de assinatura por usuário, com planos pagos diferenciados por volume de recursos, capacidade de projetos e necessidades corporativas

(como recursos avançados de segurança e suporte). A plataforma oferece um período de teste gratuito (30 dias) sem exigência inicial de cartão de crédito, após o qual é necessário aderir a um dos planos disponíveis. Os planos de entrada possuem limites em número de projetos e em certas funcionalidades avançadas, enquanto funcionalidades de portfólio, automação e relatórios mais completos tendem a estar concentradas em planos superiores.

Funcionalidades Principais

- Visualizações múltiplas de projeto (Gantt, lista, planilha, quadro Kanban e calendário), com dados sincronizados em tempo real.
- Dashboards e relatórios configuráveis para acompanhamento de status, desempenho e portfólio de projetos.
- Gestão de recursos, incluindo alocação de equipe, controle de carga de trabalho e apontamento de horas (*timesheets*).
- Automação de fluxos de trabalho, notificações e aprovações.
- Armazenamento centralizado de documentos e comentários em tarefas, com suporte a colaboração em tempo real e aplicativo móvel.

Gestão de Riscos

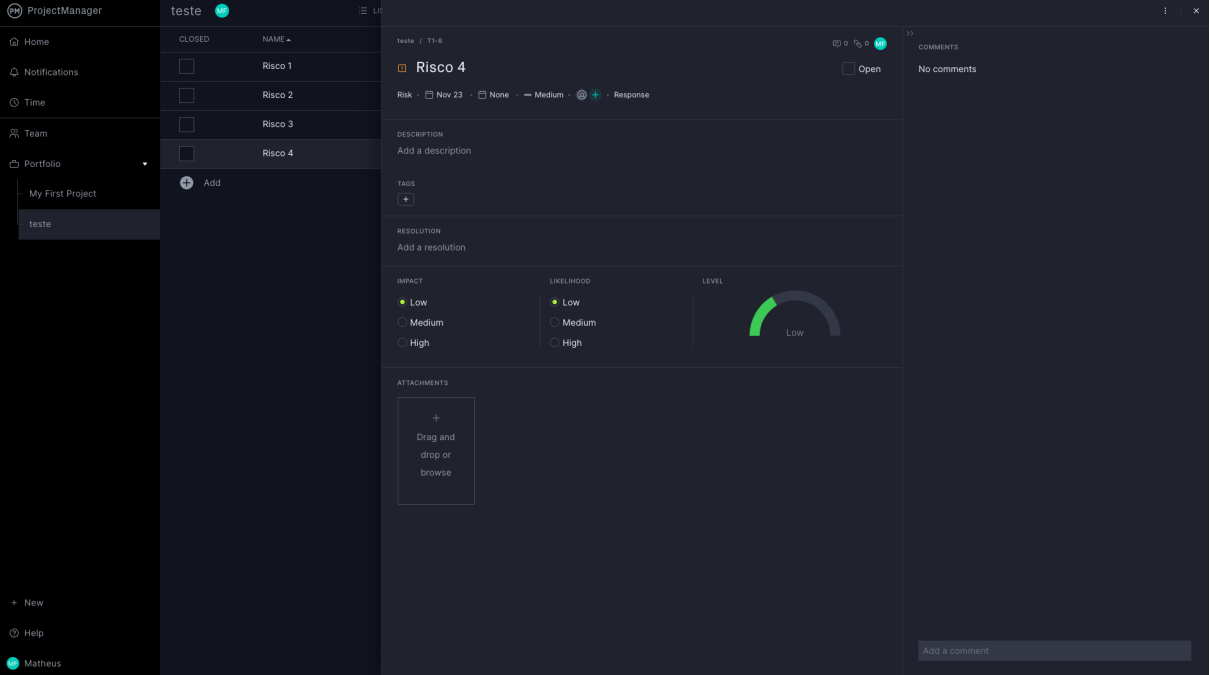
A gestão de riscos no ProjectManager é estruturada por meio de um *RAID log* (*Risks, Assumptions, Issues, Decisions/Dependencies*), que consolida riscos, premissas, problemas e decisões em um único repositório. A ferramenta permite cadastrar riscos, associar responsáveis, descrever impactos, registrar probabilidade, prioridade e acompanhar o status de tratamento. O nível de severidade é calculada a partir de impacto e probabilidade, auxiliando na priorização.

Apesar desse suporte, a abordagem permanece relativamente simples: não há, na interface principal, recursos nativos mais avançados de análise qualitativa ou quantitativa (como matrizes de probabilidade x impacto configuráveis, simulações ou análises estatísticas). A modelagem detalhada dos riscos e a profundidade das análises dependem em grande medida da configuração feita pelo usuário dentro do RAID log.

Gestão de Cronograma

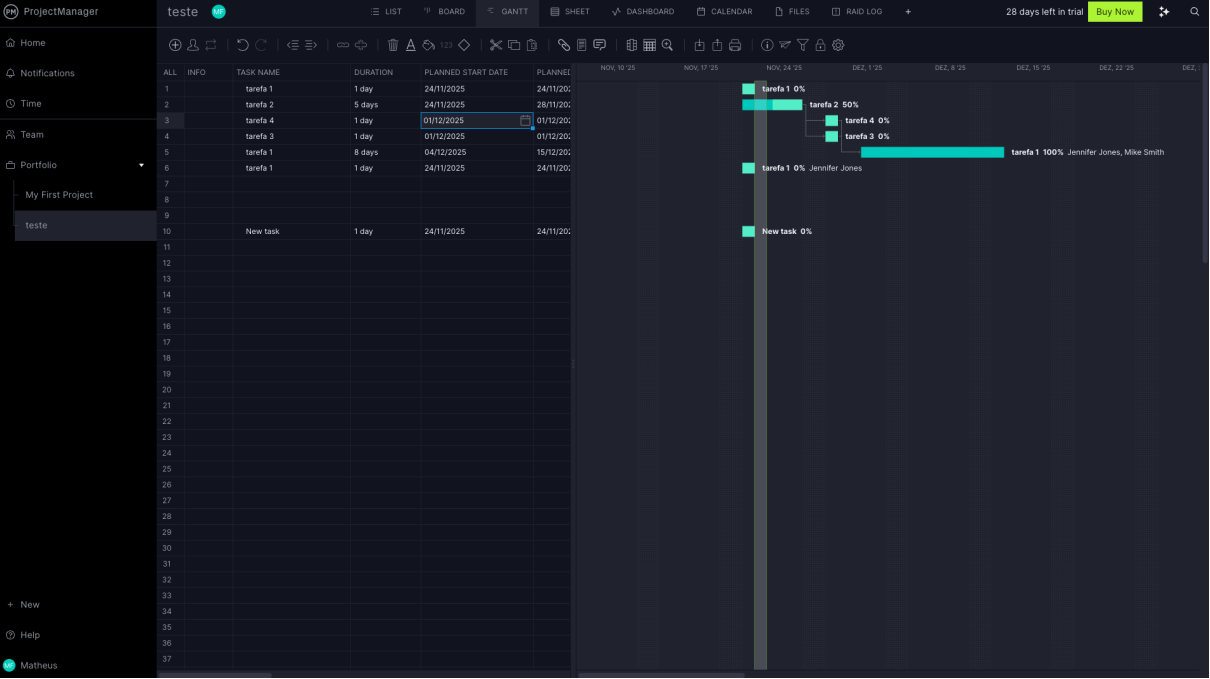
No planejamento do cronograma, o ProjectManager oferece um gráfico de Gantt com representação de tarefas, marcos, dependências e possibilidade de identificação do caminho crítico, além de funcionalidades de ajuste de linha de base. A ferramenta também disponibiliza um quadro Kanban geral do projeto para acompanhamento do fluxo de

Figura 12 – Exemplo de RAID log para gestão de riscos no ProjectManager



trabalho, bem como uma visão de calendário que exibe as tarefas conforme suas datas de início e término. No gráfico de Gantt, é possível aplicar um filtro que exibe apenas as atividades que compõem o caminho crítico, facilitando sua visualização direta no cronograma.

Figura 13 – Gestão de cronograma com gráfico de Gantt no ProjectManager



Não há, contudo, um módulo específico para gerenciamento de sprints como em

ferramentas voltadas exclusivamente a métodos ágeis; o acompanhamento de trabalho iterativo precisa ser modelado com colunas personalizadas no Kanban ou divisões em tarefas e marcos.

Gestão de Custos

O ProjectManager incorpora funcionalidades de controle de custos integradas às tarefas e recursos. É possível cadastrar custos por tarefa, bem como definir taxas horárias por projeto e por membro da equipe. Ao associar um membro a uma tarefa e informar as horas planejadas, o sistema calcula automaticamente o custo planejado daquela atividade, permitindo consolidar informações em níveis superiores de projeto e portfólio. Também há suporte para acompanhamento de custos planejados e reais, visão de orçamento por projeto e relatórios de acompanhamento financeiro integrados às visualizações de cronograma e recursos.

Observações e Limitações

A interface do ProjectManager centraliza recursos de cronograma, tarefas, custos e relatórios em visualizações gráficas e painéis de acompanhamento, o que favorece a organização das informações do projeto. Em termos de riscos, contudo, a solução permanece focada no RAID log, sem oferecer recursos nativos mais sofisticados de análise qualitativa/quantitativa ou integração profunda com métricas de desempenho do projeto. Na gestão de cronograma, apesar da robustez do Gantt e da possibilidade de identificar o caminho crítico, a ausência de um módulo dedicado a sprints limita o uso em cenários que adotam Scrum de forma mais estruturada. Já na gestão de custos, embora os recursos de custo por tarefa, taxa horária e consolidação por projeto sejam úteis e relativamente completos, a ferramenta não se posiciona como um sistema financeiro ou de contabilidade, sendo mais adequada ao acompanhamento de orçamento e custos do ponto de vista do gerenciamento de projetos.

3.3.5 Jira

Descrição Geral

O Jira é uma plataforma de gerenciamento de projetos utilizada em equipes de desenvolvimento de software e em ambientes ágeis. A ferramenta organiza o trabalho em projetos e *tickets* (épicas, histórias, tarefas, subtarefas, etc.). O Jira oferece uma variedade de modelos (*templates*) pré-configurados para diferentes tipos de projeto, como Scrum, Kanban, rastreamento de bugs, gestão de projetos de negócios e gestão de serviços. Cada template define, de forma inicial, o fluxo de trabalho (*workflow*), os tipos de *issue*, os campos padrão e as visualizações principais (quadros, backlog, cronograma, relatórios),

reduzindo o esforço de configuração inicial. A partir desses modelos, o usuário pode adaptar colunas, estados, tipos de tarefa, campos personalizados e regras de automação, criando variantes específicas para o contexto da organização. Assim, os templates funcionam como pontos de partida reutilizáveis que aceleram a criação de novos projetos e favorecem a padronização das práticas de gerenciamento.

Licenciamento

O Jira é oferecido no modelo de assinatura por usuário. O plano *Free* possui limite reduzido de usuários e recursos básicos para equipes pequenas, enquanto os planos *Standard*, *Premium* e *Enterprise* expandem o número de usuários, armazenamentos, automações, recursos de auditoria e funcionalidades avançadas de planejamento (como *Advanced Roadmaps*). No plano *Standard* e *Premium*, existe a possibilidade de 14 dias de teste gratuito.

Funcionalidades Principais

- Organização do trabalho por meio de épicos, histórias de usuário, tarefas, subtarefas e outros tipos de tickets personalizáveis.
- Backlog para criação, priorização e refinamento dos itens de trabalho.
- Planejamento e acompanhamento de sprints em projetos que utilizam Scrum.
- Quadros Scrum e Kanban com colunas e fluxos de trabalho configuráveis.
- Visualizações em lista, quadro, calendário e linha do tempo para acompanhamento das atividades.
- Roadmaps para o planejamento de versões, iniciativas e entregas ao longo do tempo.
- Estimativa de esforço por meio de pontos de história ou estimativas baseadas em tempo.
- Registro de horas trabalhadas por meio de apontamentos de trabalho (*worklogs*).
- Dashboards configuráveis para a consolidação de filtros, indicadores, gráficos e informações de diferentes projetos.
- Possibilidade de modelar registros de riscos por meio de tipos de tickets e campos personalizados.
- Expansão das funcionalidades por meio de aplicativos disponíveis no *marketplace*.

Gestão de Riscos

A gestão de riscos no Jira é realizada principalmente por meio de configuração e customização. A abordagem mais comum consiste na criação de um tipo de *ticket* específico para riscos, com campos dedicados para probabilidade, impacto, categoria, consequências e estratégia de resposta. Esses riscos podem ser organizados em um quadro próprio (por exemplo, Kanban) ou agrupados em *filters* e painéis que funcionam como um *risk register* básico.

Figura 14 – Card de riscos configurado no Jira

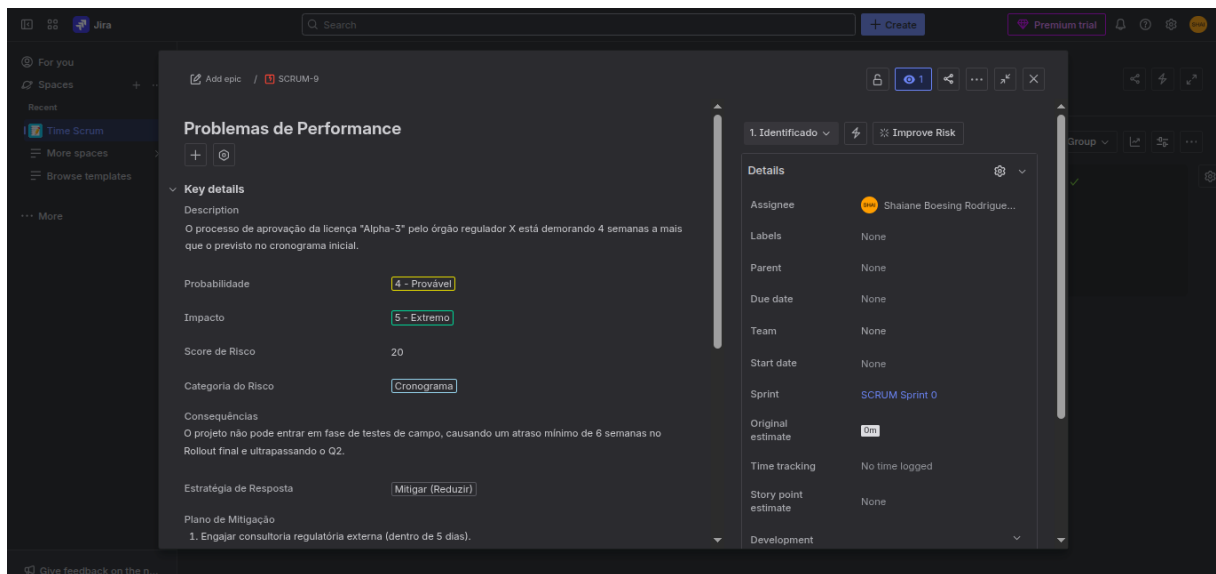
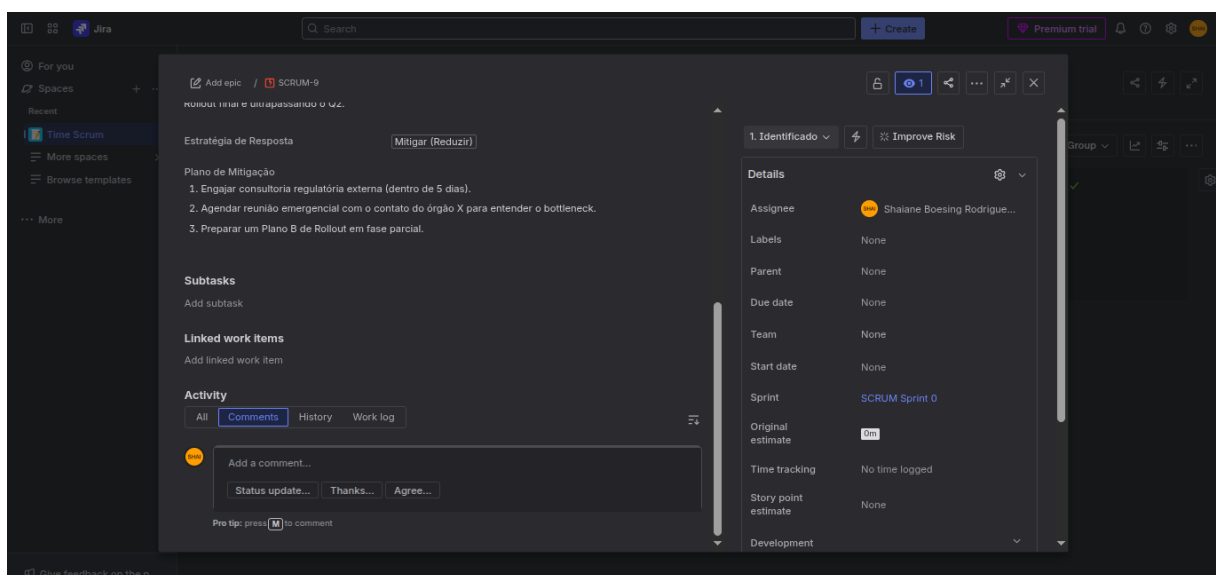


Figura 15 – Card de riscos configurado no Jira

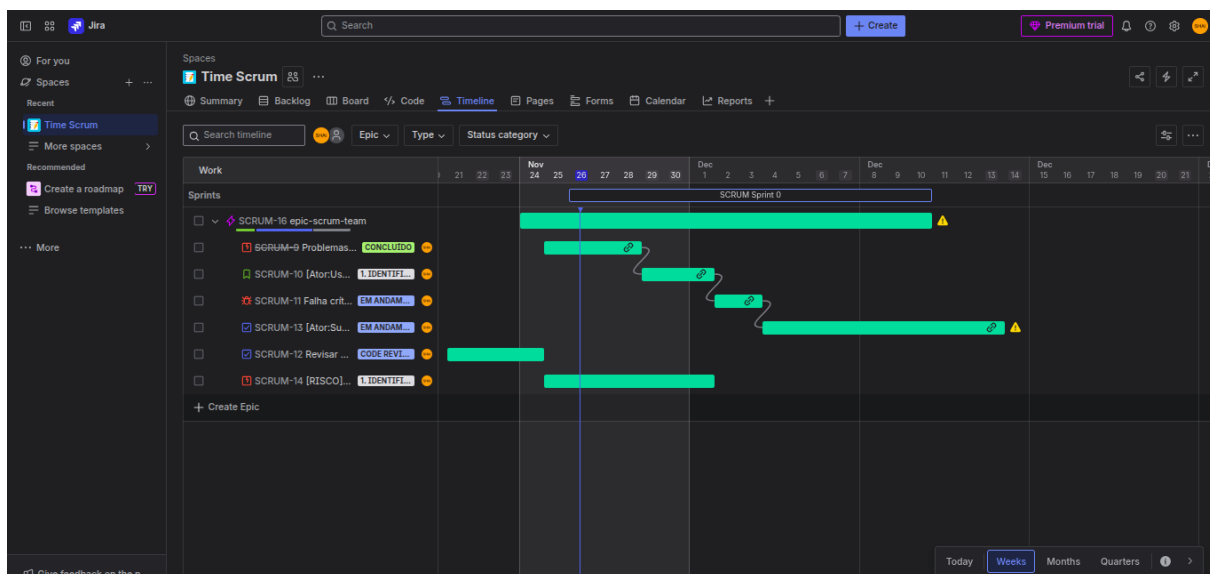


Essa abordagem atende à identificação, categorização e monitoramento contínuo de riscos, mas funcionalidades mais avançadas — como matrizes de probabilidade e impacto, análise quantitativa ou relatórios específicos de risco — normalmente dependem de aplicativos adicionais do *marketplace*. Desse modo, o suporte do Jira à gestão de riscos é flexível, porém não estruturado de forma nativa.

Gestão de Cronograma

O Jira oferece suporte significativo à gestão de cronograma por meio de *backlogs*, *sprints*, visualizações em calendário, *timeline* e *roadmaps*. As tarefas podem ser distribuídas ao longo do tempo, associadas a versões (*releases*) e vinculadas entre si por dependências básicas. O recurso *Advanced Roadmaps*, disponível em planos superiores, amplia essas capacidades ao permitir consolidar o planejamento de múltiplos projetos, simular cenários, visualizar capacidades de equipes e gerenciar dependências entre iniciativas em nível de programa.

Figura 16 – Planejamento de cronograma com *roadmap* e *timeline* no Jira



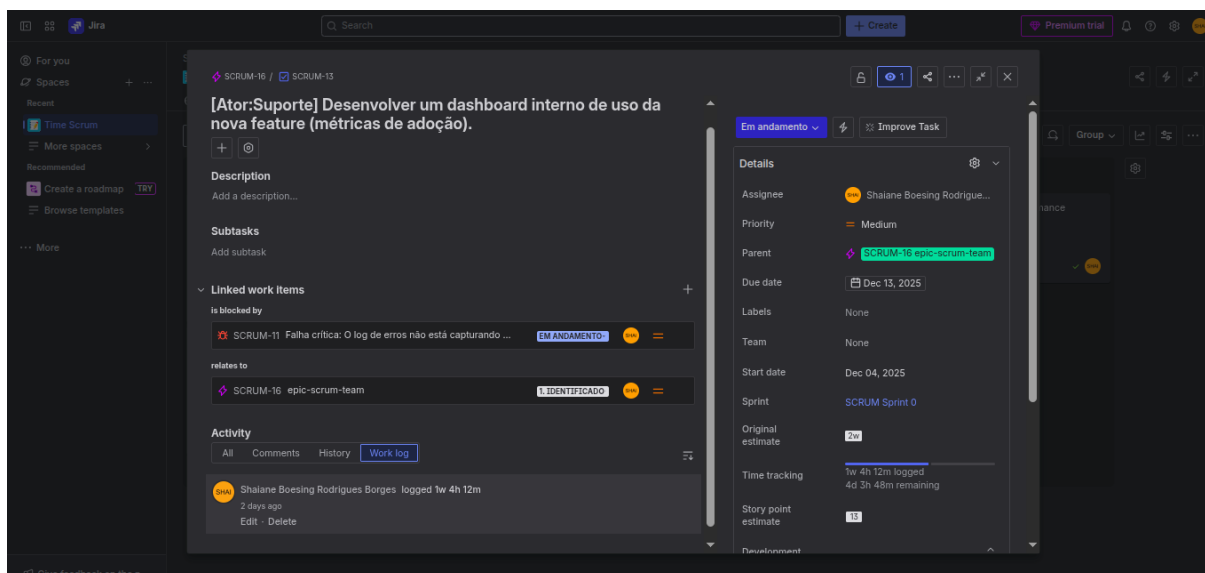
Apesar disso, o suporte a gráficos de Gantt tradicionais e a representações mais prescritivas de cronograma de engenharia é limitado no núcleo da ferramenta, sendo frequentemente complementado por aplicativos de terceiros especializados em Gantt e portfólio.

Gestão de Custos

A gestão de custos no Jira é indireta e depende de integrações ou customizações. A ferramenta permite registrar estimativas de esforço, horas trabalhadas (*worklogs*) e associar tarefas a versões e componentes, o que abre espaço para cálculos de custo planejado e

custo realizado a partir de tarifas horárias ou centros de custo externos. No entanto, não há um módulo financeiro nativo com orçamento, estrutura de contas, controle de despesas e métricas consolidadas de valor agregado (*Earned Value*).

Figura 17 – Exemplo de estimativas de esforço e registro de horas trabalhadas no Jira



Organizações que necessitam de um controle mais formal de custos costumam recorrer a aplicativos do *marketplace* ou a sistemas financeiros externos que consomem os dados de tempo e esforço registrados no Jira.

Observações e Limitações

O Jira é uma solução amplamente adotada por equipes de desenvolvimento e oferece recursos para planejamento iterativo, organização e rastreamento do trabalho, além de apoiar a colaboração entre os integrantes da equipe (Atlassian, 2026b,a). Entretanto, seu suporte nativo à gestão estruturada de riscos e custos é limitado, exigindo customizações extensas e, muitas vezes, aplicativos complementares para alcançar o nível de formalização esperado em ambientes de PMO. Na gestão de cronograma, a ferramenta oferece recursos robustos para cenários ágeis, mas ainda depende de extensões para representar cronogramas tradicionais baseados em Gantt. Essa combinação de flexibilidade elevada com pouca estruturação padrão reforça, no contexto deste trabalho, a motivação para investigar e propor funcionalidades mais específicas de riscos, cronograma e custos em ferramentas como o Taiga.

3.3.6 Artia

Descrição Geral

O Artia é uma plataforma nacional de gestão de projetos, tarefas e equipes, oferecida como serviço em nuvem. A ferramenta centraliza portfólio de projetos, cronogramas, controle financeiro, apontamento de horas e colaboração em um único ambiente, combinando gráficos de Gantt, quadros Kanban, dashboards analíticos e relatórios gerenciais. A solução é orientada a escritórios de projetos e organizações que precisam acompanhar prazos, esforço, custos e rentabilidade em múltiplos projetos, oferecendo mais de trinta ferramentas integradas para planejamento, execução e controle.

Licenciamento

O Artia adota um modelo de assinatura por usuário, com planos em camadas. Há um plano *Free*, gratuito e limitado, que dá acesso a funcionalidades fundamentais de organização de projetos. Os planos pagos (*Slim*, *Business* e *Enterprise*) ampliam o número de usuários, a quantidade de projetos e os recursos disponíveis. A partir do plano *Slim*, são oferecidos Gantt, Kanban, modelos de projeto, calendário, módulo financeiro, apontamento de horas e dashboards. Planos superiores adicionam Gantt avançado, EAP/WBS, alocação de equipe, relatórios de situação, recursos de *analytics*, linha de base de cronograma e integração com ferramentas de Business Intelligence. Todos os planos pagos contam com período de teste gratuito.

Funcionalidades Principais

- Organização do trabalho por meio de projetos, tarefas, subtarefas, responsáveis e prazos.
- Visualização e acompanhamento das atividades por meio de listas, calendários e quadros Kanban.
- Estruturação do escopo do projeto por meio de uma Estrutura Analítica do Projeto (EAP).
- Planejamento e acompanhamento do cronograma por meio de gráficos de Gantt.
- Registro e acompanhamento de riscos do projeto.
- Controle de orçamentos, custos e horas trabalhadas.
- Dashboards e relatórios para acompanhamento do andamento e do desempenho dos projetos.

Gestão de Riscos

O Artia oferece um módulo próprio para gestão de riscos, no qual é possível identificar, registrar, analisar e monitorar riscos de forma contínua. Os riscos são cadastrados com informações como descrição, impacto, probabilidade, categoria, responsável e plano de resposta, podendo receber classificações e prioridades de tratamento. Eles podem ser relacionados a projetos específicos ou ao portfólio. Essa abordagem fornece um registro estruturado de riscos, permitindo documentar eventos de risco, responsáveis, planos de ação e informações relacionadas ao acompanhamento dos riscos do projeto. No entanto, não disponibiliza recursos como matriz probabilidade $\times$ impacto ou mecanismos mais avançados de análise e monitoramento visual dos riscos.

Figura 18 – Exemplo 1 de registro de riscos configurado no Artia

Grupo de Trabalho / Projeto TCC

1. Universidade em greve + Pessoas LT

Ver planos

Informações

Título
1. Universidade em greve

Tipo
Ameaça

Descrição

Categorias

Tags

Análise Qualitativa

Probabilidade Média (5)	Impacto Alto (4)	Exposição ao risco 20,00
-----------------------------------	----------------------------	------------------------------------

Análise Quantitativa

Probabilidade
50,00%

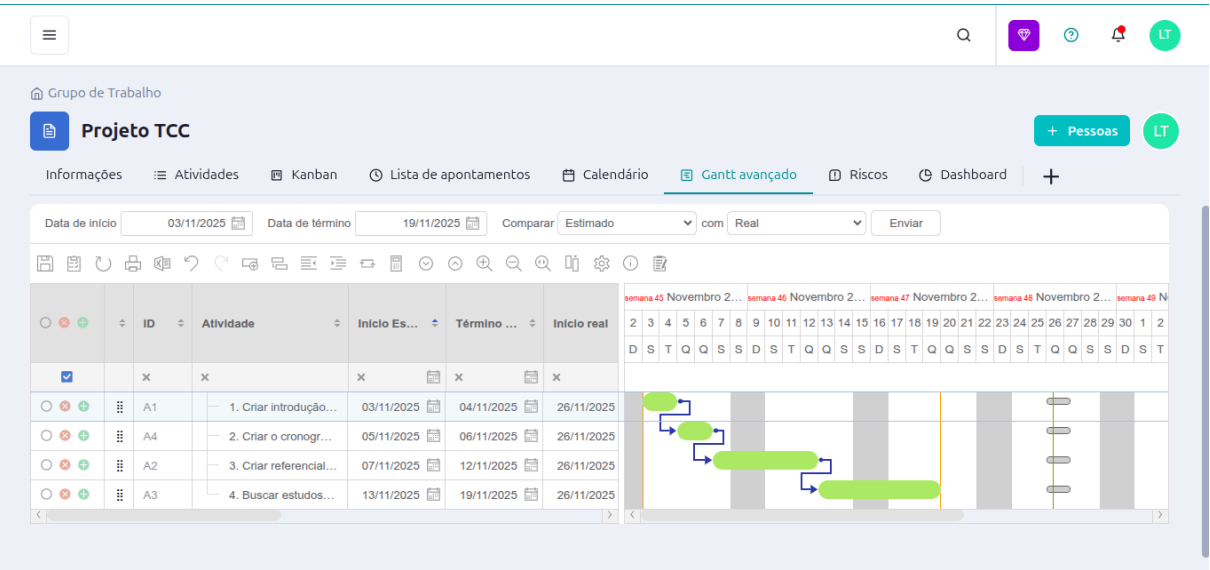
Gestão de Cronograma

Na gestão de cronograma, o Artia utiliza o gráfico de Gantt como principal recurso de planejamento e acompanhamento. Cada projeto é estruturado em uma EAP com tarefas, marcos, datas de início e término, dependências e responsáveis. O cronograma permite visualizar o caminho das atividades ao longo do tempo, ajustar prazos de forma interativa e acompanhar o avanço físico do projeto. Em planos mais avançados, é possível utilizar linha de base de cronograma, alocação de recursos e visões consolidadas de portfólio, além de empregar recursos de inteligência artificial para gerar cronogramas automaticamente a partir de dados fornecidos pelo usuário.

Figura 19 – Exemplo 2 de registro de riscos configurado no Artia



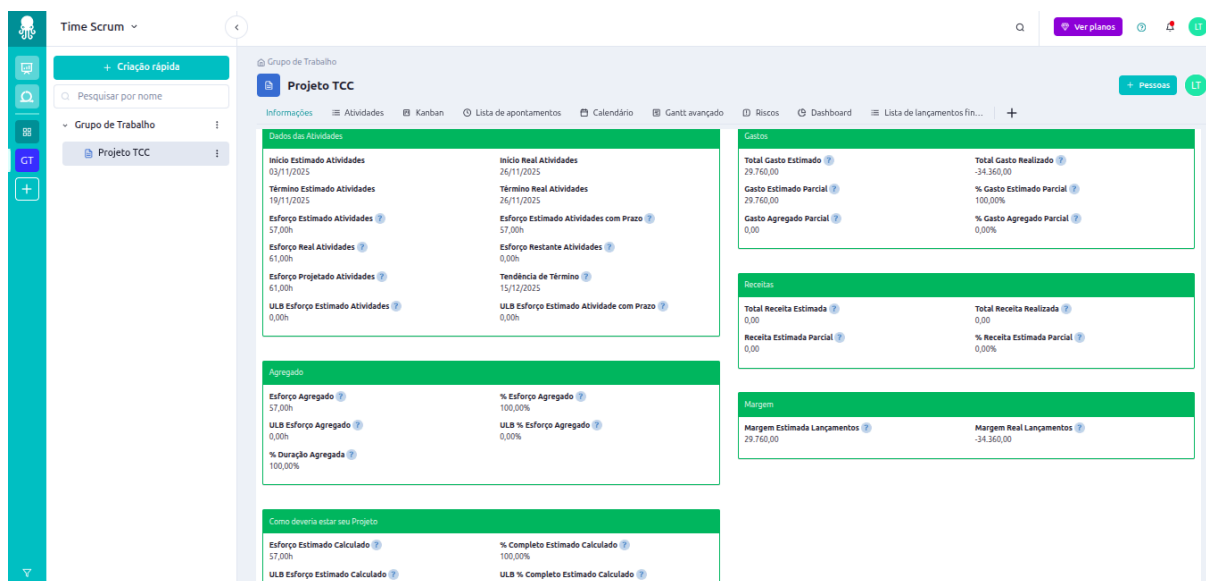
Figura 20 – Planejamento e acompanhamento de cronograma no Artia



Gestão de Custos

O Artia inclui funcionalidades de controle financeiro integradas aos projetos. É possível definir orçamentos, registrar custos planejados e realizados, acompanhar horas trabalhadas e horas faturáveis, bem como comparar valores planejados com valores efetivos para analisar rentabilidade e variações de custo. Os dados financeiros podem ser consolidados em relatórios, permitindo ao gestor visualizar indicadores de desempenho econômico por projeto ou cliente. Embora ofereça um controle de custos mais estruturado que ferramentas voltadas apenas para métodos ágeis, o Artia não se posiciona como sistema contábil completo.

Figura 21 – Dashboard de indicadores de um projeto no Artia



Observações e Limitações

O Artia se apresenta como uma solução completa de gestão de projetos, com integração explícita entre cronograma, riscos, custos, portfólio e colaboração, o que o aproxima das necessidades típicas de um PMO. Em contrapartida, essa abrangência implica maior complexidade de configuração e curva de aprendizado mais elevada em comparação com ferramentas mais leves, especialmente para equipes que atuam apenas com quadros Kanban ou listas simples de tarefas. Além disso, parte das funcionalidades mais avançadas — como EAP, Gantt avançado, linha de base, *analytics* e integrações de alto nível — está restrita aos planos superiores, o que pode limitar o uso completo dos recursos em organizações que dependem do plano gratuito ou de entradas de menor custo. No contexto deste trabalho, o Artia ilustra uma abordagem em que riscos, cronograma e custos são tratados de forma integrada e especializada, servindo como referência de maturidade em contraste com ferramentas mais focadas em métodos ágeis básicas, como o Taiga.

3.3.7 Smartsheet

Descrição Geral

O Smartsheet é uma plataforma de gerenciamento de projetos em nuvem, com interface tabular semelhante a uma planilha, que podem ser visualizados como planilha, quadro, tabela, gantt, cartão (*card*), linha do tempo e, até mesmo, calendário combinando recursos de cronograma e colaboração em um único ambiente, mantendo a flexibilidade

de customização de planilhas eletrônicas. Essa flexibilidade permite alternar entre uma visualização analítica ou operacional, conforme a necessidade da equipe.

Licenciamento

O Smartsheet é comercializado no modelo de assinatura. Há uma opção gratuita com número limitado de folhas e colaboradores, voltada a uso individual básico, e planos pagos (*Pro*, *Business* e *Enterprise*) que expandem a quantidade de usuários, recursos de automação, dashboards, integrações e governança. Os planos pagos incluem período de avaliação gratuita por tempo limitado, mas não há uma versão gratuita permanente com todos os recursos.

Funcionalidades Principais

- Gestão de cronograma
- Automação de fluxos de trabalho
- Dashboards
- Formulários de coleta de dados
- Integrações externas
- Controle de permissões e colaboração
- Gestão de arquivos e anexos
- Visualizações múltiplas (Planilha, Gantt, Calendário, Quadro e Timeline)

Gestão de Riscos

A gestão de riscos no Smartsheet não é suportada por um módulo dedicado, sendo construída inteiramente a partir de planilhas configuráveis, colunas personalizadas e templates de risco. Um registro de riscos típico é estruturado como uma planilha com colunas para descrição, causa, consequência, probabilidade, impacto, categoria, score calculado por fórmula, plano de resposta, responsável e status. Esses dados podem ser filtrados, destacados com formatação condicional e agrupados em relatórios e painéis para apoiar a identificação e o monitoramento. No entanto, de forma nativa, não é possível criar matrizes gráficas de probabilidade e impacto ou modelos de análise quantitativa avançada.

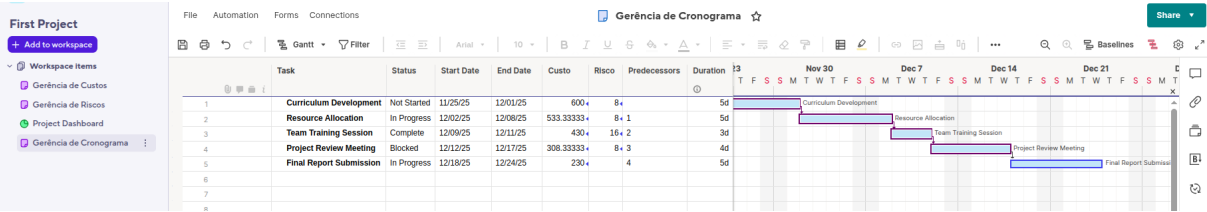
Figura 22 – Exemplo de registro de riscos configurado no Smartsheet

	REF ID	CATEG	DESCRIPTION	PROBABILITY	IMPACT	SCORE	LIKELY TIMING OF RISK	RISK TRIGGER	MITIGATION / RESPONSE PLAN	STATUS OF MITIGATION / RESPONSE
1	10001	Materi als	If there are changes in architectural plans that require additional or different resources, then there will be delays.	1	8	8	The risk is present throughout the process of finalizing architectural plans.	There are changes in the architectural plans.	A designated team member will continually monitor any proposed changes in architectural plans and ensure communication between architectural and construction teams about proposed changes.	No current issues
2	10002	Contra cts	If contractual issues among contractors, subcontractors, or vendors arise, then there will be delays or cost overruns.	2	4	8	The risk is present before most contract work commences. However, the risk will also be present when additional contracts are added in the early phases of construction.	The contracts manager reports any potential contract issues.	The contracts manager will continually monitor contracts, alert the team when there are issues, and ensure that the team addresses and resolves contract issues before they cause delays.	Working on one unresolved issue with the electrician's contract; meeting set for May 2nd
3	10003	Permit s / Regul ations	If obtaining appropriate construction permits takes longer than planned, then there will be delays.	2	8	16	The risk is present throughout the construction phase of the project, including when securing required permits at the beginning of construction.	Construction permits do not arrive within the required timeframe.	The construction manager, or another designated team member, will monitor the project schedule and deadlines and ensure that all permits can accommodate the project's construction schedule.	All deadlines currently being monitored
4	10004	Constr uction Site	If unexpected hazardous waste is found on the property, it will require mitigation or cleanup.	1	8	8	The risk is present throughout the project, especially toward the beginning when conducting site preparation work.	On-site managers report an issue, especially during site preparation.	The construction manager will assign a team member to monitor and explore any possible issues with hazardous waste.	No issues found to this point
5										
6										
7										

Gestão de Cronograma

Na gestão de cronograma, o Smartsheet oferece recursos prescritivos de planejamento baseados em planilhas estruturadas, com colunas de datas, duração, predecessoras, marcos e dependências. As visualizações em Gantt, calendário, quadro e timeline permitem analisar o cronograma sob diferentes perspectivas, enquanto o caminho crítico pode ser destacado diretamente na visualização em Gantt. A estrutura hierárquica de linhas possibilita representar a decomposição do trabalho, e o cálculo automático de início e término facilita o sequenciamento das atividades. Essas visualizações coexistem sobre a mesma fonte de dados, garantindo consistência e comunicação clara do plano ao longo do projeto.

Figura 23 – Visualização de cronograma em gráfico de Gantt no Smartsheet



Gestão de Custos

A gestão de custos no Smartsheet é baseada em colunas personalizadas e fórmulas, permitindo estruturar orçamento, custo planejado e custo real por tarefa ou projeto, bem

como registrar despesas e calcular variações de forma simples. A plataforma possibilita combinar horas registradas, taxas horárias e percentuais concluídos para gerar estimativas e acompanhar desvios, com apoio de dashboards financeiros compostos por gráficos e indicadores derivados de relatórios agregados. Entretanto, o Smartsheet não oferece um módulo financeiro dedicado: não há criação formal de linhas de base de custo, previsões automáticas, controle avançado de lançamentos ou integrações contábeis robustas, o que restringe seu uso a cenários de menor complexidade e análises essencialmente operacionais.

Figura 24 – Dashboard de custos e orçamento configurado no Smartsheet

Column1	Column3	Column2	Column4	Column5	Column6	Column7	Column8
DATA	REUNIDOS POR	DESCRIÇÃO DO ITEM/TAREFA	MELHOR CASO	MAIS PROVÁVEL	PIOR CASO	MÉDIA PONDERADA	INFORMAÇÕES ADICIONAIS/NOTAS
01/01/19	Nome/Sobrenome	Nome e descrição do produto	500.00	R\$600.00	R\$700.00	600	Fornecedor, representante etc.
01/02/19	Nome/Sobrenome	Título e descrição da tarefa	400.00	R\$500.00	R\$800.00	533.33333	Horas, taxa etc.
01/03/19	Nome/Sobrenome	Nome e descrição do serviço	300.00	R\$400.00	R\$680.00	430	Horas, taxa etc.
01/04/19	Nome/Sobrenome	Nome e descrição do item	200.00	R\$300.00	R\$450.00	308.33333	Detalhes adicionais do item etc.
01/05/19	Nome/Sobrenome	Nome e descrição do produto	100.00	R\$200.00	R\$480.00	230	Fornecedor, representante etc.

Observações e Limitações

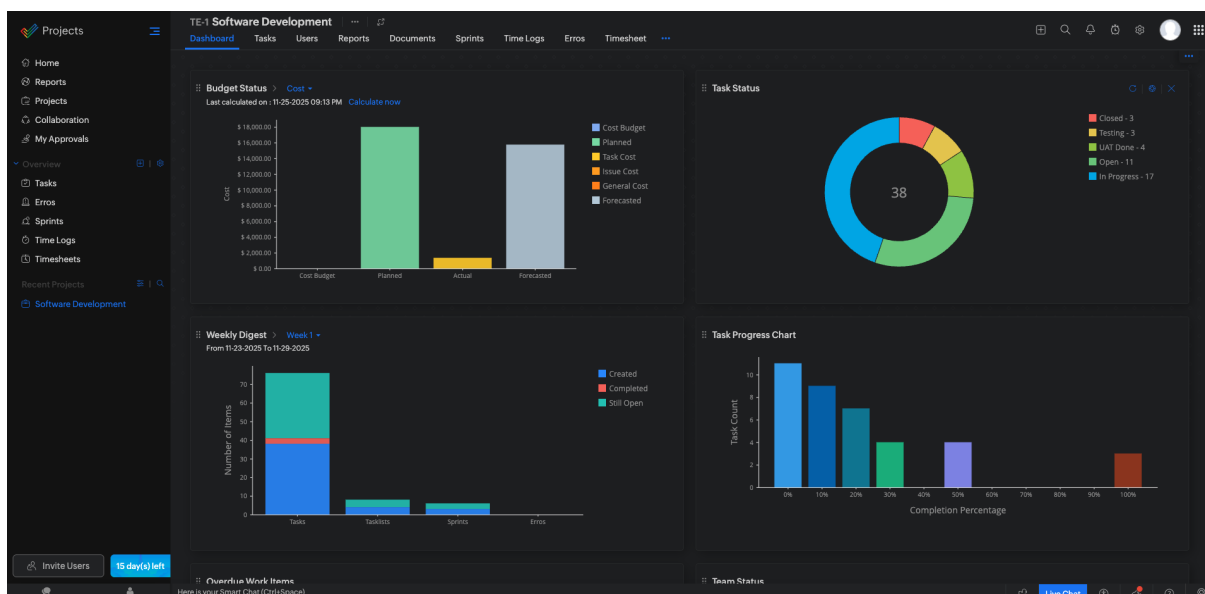
O Smartsheet combina a familiaridade de uma planilha com recursos de Gantt, automação e colaboração, o que o torna atrativo para equipes que desejam estruturar cronogramas, riscos e custos sem abandonar o modelo tabular. Em comparação com ferramentas mais ágeis, oferece recursos mais robustos de cronograma; por outro lado, a gestão de riscos e de custos baseia-se fortemente em modelagem via colunas e fórmulas, sem um módulo nativamente especializado.

3.3.8 Zoho Projects

Descrição Geral

O Zoho Projects é uma plataforma de gerenciamento de projetos baseada em nuvem, integrada ao ecossistema Zoho. A ferramenta oferece suporte ao planejamento, acompanhamento e colaboração por meio de módulos de tarefas, marcos, cronograma, acompanhamento de tempo e relatórios. Sua abordagem é voltada principalmente para projetos prescritivos, porém com possibilidade de integração a ferramentas ágeis (como o Zoho Sprints), permitindo cenários híbridos de gestão.

Figura 25 – Visão geral de um projeto no Zoho Projects



Licenciamento

O Zoho Projects adota modelo de licenciamento *freemium*⁴. Há um plano gratuito com número limitado de usuários, projetos e recursos, voltado a equipes pequenas e cenários simples. Os planos pagos (Premium e Enterprise) são comercializados por usuário/mês e liberam funcionalidades adicionais, como maior capacidade de projetos, campos personalizados avançados, automações, integrações ampliadas e recursos de orçamento e acompanhamento de custos. O acesso é realizado via assinatura, com opção de período de avaliação gratuito.

Funcionalidades Principais

- Cadastro e acompanhamento de tarefas, marcos e listas de tarefas, com atribuição de responsáveis, datas e percentuais de conclusão.
- Visualizações múltiplas do projeto, incluindo listas, gráfico de Gantt, quadro Kanban, planilha e calendário.
- Registro de horas trabalhadas por meio de *timesheets*, com posterior uso em relatórios e faturamento.
- Módulo de acompanhamento de problemas (*issues/bugs*), integrado às tarefas do projeto.

⁴ Modelo de licenciamento em que a ferramenta oferece uma versão gratuita com recursos limitados, enquanto funcionalidades avançadas e maior capacidade de uso são disponibilizadas apenas em planos pagos.

- Personalização por campos, layouts, perfis, fluxos de trabalho e integrações com outros produtos Zoho e aplicações de terceiros.

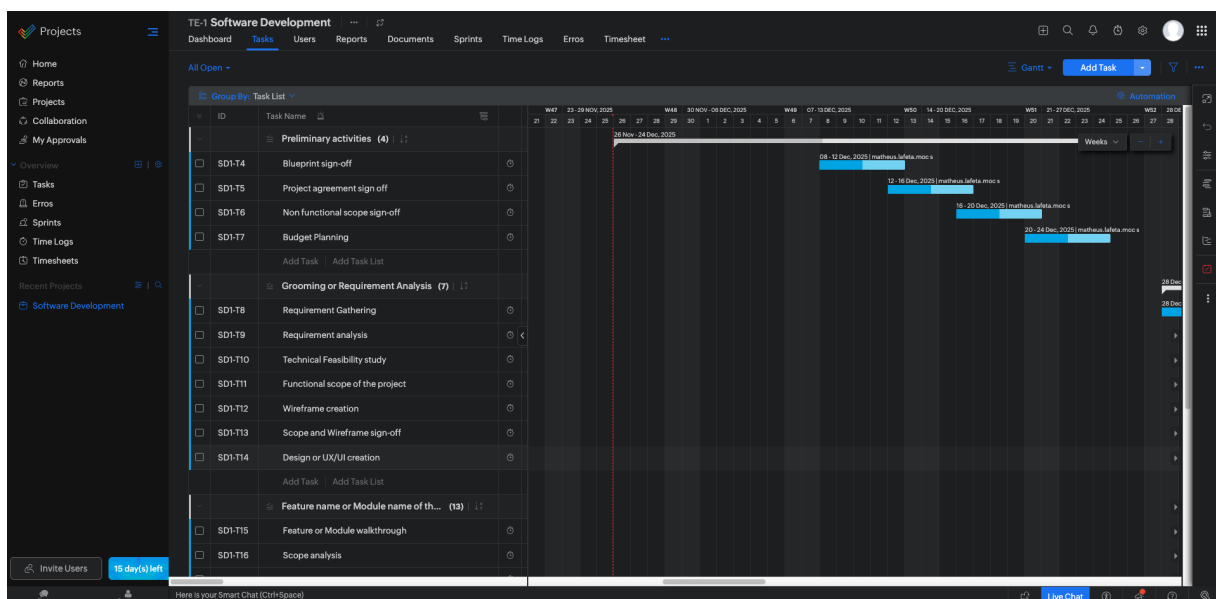
Gestão de Riscos

O Zoho Projects, na configuração padrão, não dispõe de um módulo nativo dedicado exclusivamente à gestão de riscos. A identificação e o acompanhamento de riscos podem ser realizados de forma indireta, modelando cada risco como tarefa ou *issue*, complementada por campos personalizados (por exemplo, probabilidade, impacto, prioridade e plano de resposta). Adicionalmente, é possível estender a gestão de riscos por meio de integrações e aplicações do marketplace Zoho, que oferecem funcionalidades específicas para registro e monitoramento de riscos.

Gestão de Cronograma

No planejamento do cronograma, o Zoho Projects oferece um gráfico de Gantt com representação de tarefas, marcos e dependências, além de recursos de linha de base e identificação do caminho crítico. A ferramenta disponibiliza ainda um quadro Kanban geral para acompanhamento do fluxo de trabalho e uma visão de calendário que organiza as tarefas de acordo com suas datas de início e término, permitindo monitorar a completude de tarefas e projetos. Para equipes que adotam práticas ágeis, é possível integrar o Zoho Projects ao Zoho Sprints, passando a visualizar sprints e itens de backlog a partir do próprio ambiente do Projects em um modelo híbrido.

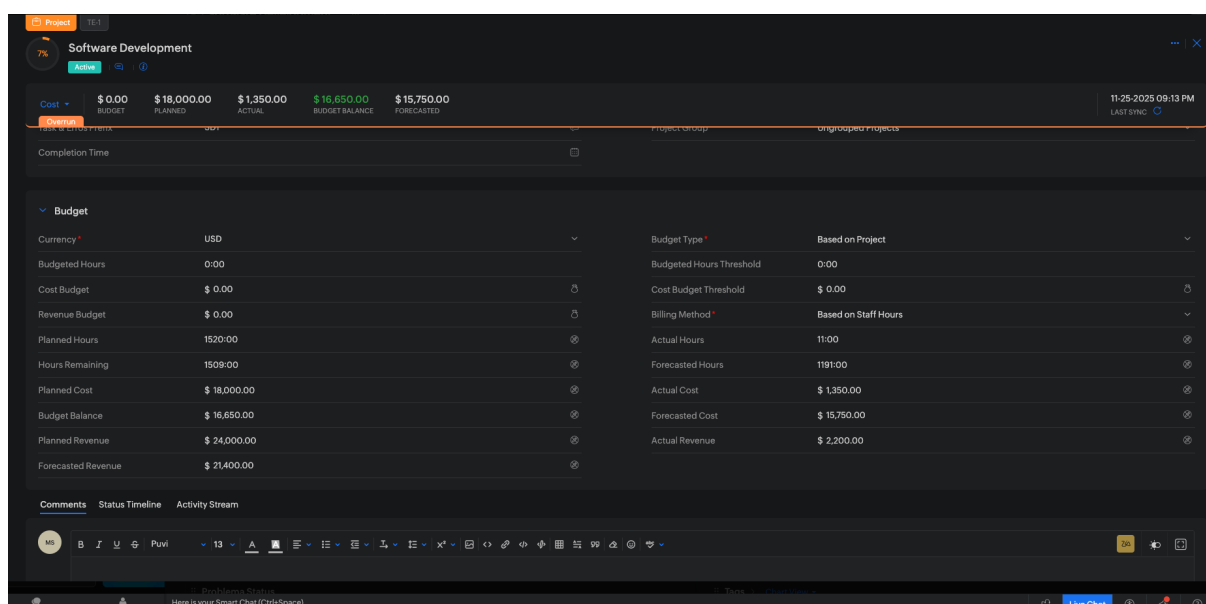
Figura 26 – Gestão de cronograma com gráfico de Gantt no Zoho Projects



Gestão de Custos

O Zoho Projects possui recursos nativos de orçamento e controle de custos, permitindo definir um orçamento por projeto e acompanhar custos planejados e realizados. É possível configurar métodos de faturamento baseados em horas de projeto, horas por membro da equipe (*staff hour*) ou custo fixo, bem como associar taxas horárias a usuários e tarefas. A partir de horas planejadas e registradas nas *timesheets*, o sistema calcula custos planejados e reais, e disponibiliza visões consolidadas por projeto para análise de aderência ao orçamento e identificação de possíveis estouros de custo. A plataforma também se integra a ferramentas de faturamento da própria Zoho para emissão de faturas com base nos dados de horas e custos registrados.

Figura 27 – Dashboard de orçamento e custos no Zoho Projects



Observações e Limitações

O Zoho Projects apresenta um conjunto amplo de funcionalidades e opções de configuração, porém essa riqueza de recursos torna a interface visualmente carregada e, em alguns momentos, pouco intuitiva. Durante os testes realizados neste trabalho, observou-se que a navegação entre tarefas, cronogramas, relatórios e configurações exige um tempo de adaptação, em especial devido à quantidade de menus, abas e ícones disponíveis simultaneamente na tela. Além disso, algumas funcionalidades relevantes — como a configuração de *budget* do projeto — encontram-se em seções menos evidentes da interface, o que dificulta sua descoberta por novos usuários. Em contrapartida, a gestão de riscos depende de modelagem manual via tarefas ou *issues* e de customizações adicionais, não havendo um módulo de riscos com suporte estruturado às etapas do gerenciamento de riscos do

PMBOK. Na gestão de cronograma e de custos, os recursos são mais robustos, porém algumas funcionalidades, como orçamentos avançados, métodos de faturamento e personalizações de relatórios, estão condicionadas aos planos pagos. Para cenários fortemente orientados a Scrum, épicos e backlog detalhado, a adoção conjunta de Zoho Projects e Zoho Sprints oferece maior cobertura, embora o módulo de orçamento do Zoho Sprints seja mais simples do que soluções financeiras especializadas, sendo focado em acompanhar custos e orçamento no contexto das iterações ágeis.

3.4 DISCUSSÃO

Com base na análise individual das ferramentas apresentada na Seção 3.3, esta seção discute comparativamente o suporte à gestão de cronograma, custos e riscos nos diferentes softwares avaliados, bem como posiciona o Taiga em relação a essas soluções. O objetivo é evidenciar limitações e oportunidades de aprimoramento que fundamentam as funcionalidades propostas neste trabalho para o Taiga, especialmente nos domínios de caminho crítico, gestão financeira integrada e gerenciamento de riscos.

O quadro 6 sintetiza, em uma escala de 0 a 2, o nível de suporte de cada ferramenta às quatro classes de funcionalidades consideradas: F1 (gestão de cronograma), F2 (gestão financeira), F3 (gestão de riscos) e F4 (possibilidade de hospedagem *on-premise* com todas as funcionalidades disponíveis de forma gratuita). Nessa escala, 0 indica ausência de suporte, 1 indica suporte parcial ou limitado, e 2 indica suporte considerado abrangente para o contexto deste estudo.

Observa-se que ferramentas voltadas a contextos de *Project Portfolio Management* (PPM), como OpenProject, ProjectManager, Artia e Zoho Projects, tendem a apresentar níveis mais elevados em F1 e F2. Esses sistemas oferecem gráficos de Gantt completos, com dependências, caminho crítico, linha de base e, em alguns casos, visualizações em nível de portfólio, além de módulos de orçamento, custos planejados e custos realizados, muitas vezes integrados a apontamento de horas e indicadores financeiros. Contudo, essas soluções permanecem predominantemente comerciais: mesmo quando disponibilizam opção *on-premise*, como é o caso de algumas edições do OpenProject, o acesso às funcionalidades mais avançadas depende de licenças pagas, o que explica a classificação F4 igual a 0 ou 1 nesses casos.

Ferramentas com forte orientação ágil e colaborativa, como Monday, Smartsheet e Jira, apresentam um perfil intermediário. Em F1, Monday e Smartsheet alcançam nível 2 por combinarem Gantt, dependências e visualizações alternativas (calendário, *timeline*, quadros), ainda que a representação de sprints e épicos dependa, em parte, de modelagem pelo usuário. O Jira, por sua vez, é robusto na gestão de backlog, sprints, épicos e tarefas, mas carece de suporte nativo a Gantt tradicional e caminho crítico, motivo pelo qual foi classificado como nível 1 em F1. Em F2 e F3, essas ferramentas tendem a se apoiar em colunas numéricas, fórmulas, templates e integrações com outros sistemas, o que justifica

Quadro 6 – Matriz de funcionalidades das ferramentas analisadas

Ferramenta	F1	F2	F3	F4	T
Taiga	1	0	0	2	3
Monday	2	1	1	0	4
OpenProject	2	2	1	1	6
ProjectManager	2	2	1	0	5
Jira	1	1	1	0	3
Artia	2	1	1	0	4
Smartsheet	2	1	1	0	4
Zoho Projects	2	2	1	0	5
Taiga futuro	2	2	2	2	8

Legenda de features: F1 – gestão de cronograma (gráfico de Gantt, caminho crítico, folga das atividades, sprints/épicas/histórias/tarefas); F2 – gestão financeira (custo por tarefa, hora trabalhada por integrante, valor de receita do projeto e visão consolidada de finanças); F3 – gestão de riscos (registro de riscos, análise qualitativa, matriz probabilidade $\times$ impacto, acompanhamento e monitoramento de riscos); F4 – possibilidade de implantação *on-premise* com acesso a todas as funcionalidades de forma gratuita (software livre, sem *paywall* funcional).

Escala de maturidade: 0 – não há suporte à classe de funcionalidades; 1 – suporte parcial ou limitado; 2 – suporte abrangente.

T: total obtido pela soma das pontuações F1, F2, F3 e F4.

Fonte: Elaborado pelos autores (2025).

a classificação como suporte parcial: é possível montar controles financeiros e registros de riscos, mas geralmente sem um módulo dedicado e sem cobertura completa das práticas de orçamento e gerenciamento de riscos preconizadas em guias como o PMBOK. Além disso, todas essas soluções são centradas em modelos de assinatura em nuvem, sem opção de implantação *on-premise* gratuita com acesso a todas as funcionalidades ($F4 = 0$).

O Taiga, foco deste trabalho, aparece em uma posição mais restrita na matriz quando se consideram apenas F1, F2 e F3. Em F1, obtém nível 1 por oferecer boa cobertura do ponto de vista ágil (backlog, épicas, histórias, tarefas e sprints), mas sem recursos de abordagens prescritivas de cronograma, como gráfico de Gantt, caminho crítico e visualização de folga das atividades. Em F2 e F3, o Taiga foi classificado com nível 0, uma vez que não dispõe de módulos nativos de gestão de custos ou de riscos, nem de mecanismos estruturados para armazenamento de orçamento, controle de custos planejados e realizados, ou registro de riscos com probabilidade, impacto e estratégias de resposta. Por outro lado, em F4, o Taiga atinge nível 2: por ser uma ferramenta open source, permite implantação *on-premise* com acesso a todas as funcionalidades sem custo de licença, característica que o diferencia das demais soluções analisadas.

Essa comparação evidencia um padrão: quanto mais completa é a cobertura de cronograma, custos e riscos (F1, F2 e F3), mais as ferramentas tendem a se posicionar como soluções PPM robustas, geralmente comerciais, com maior complexidade de configuração e curva de aprendizado, além de dependência de licenças para acesso pleno às funcionalidades.

Por outro lado, ferramentas mais leves e centradas em métodos ágeis, como o Taiga, oferecem ferramentas para gerência de backlog e sprints, mas sacrificam funcionalidades estruturadas de planejamento, orçamento e riscos. Surge, assim, uma lacuna entre soluções puramente ágeis (simples, mas limitadas em F2 e F3) e soluções PPM completas (robustas, porém mais pesadas, proprietárias e, em geral, sem a combinação de liberdade de uso e implantação *on-premise* oferecida por projetos open source).

No contexto deste trabalho, as funcionalidades propostas para o Taiga buscam justamente reduzir essa lacuna. No eixo de gestão de cronograma (F1), pretende-se introduzir um gráfico de Gantt integrado ao backlog, com cálculo automático de caminho crítico e visualização da folga de cada atividade, preservando ao mesmo tempo o modelo de sprints, épicos, histórias e tarefas já consolidado na ferramenta. Em gestão financeira (F2), são planejados mecanismos para cadastro de custo por tarefa, parametrização de horas trabalhadas por integrante com taxas associadas, definição de valor de receita do projeto e uma página de visão geral financeira que consolide orçamento, custos planejados, custos realizados e margens. Por fim, em gestão de riscos (F3), propõe-se a criação de um registro de riscos associado ao projeto, com campos de probabilidade e impacto, visualização em matriz probabilidade $\times$ impacto, definição de responsáveis e estratégias de resposta aos riscos, bem como recursos para acompanhamento histórico e monitoramento contínuo da evolução dos riscos ao longo do projeto.

Ao elevar o suporte do Taiga nessas três dimensões (F1, F2 e F3), mantendo simultaneamente sua característica de software livre e de implantação *on-premise* com todas as funcionalidades disponíveis sem custo de licença, almeja-se aproximar a ferramenta do nível de maturidade observado em soluções PPM mais completas, sem perder as características que a tornam atrativa em contextos ágeis: simplicidade de uso, foco em backlog e sprints, e natureza open source. Dessa forma, o módulo proposto neste TCC busca transformar o Taiga em uma alternativa singular entre as ferramentas estudadas: após a implementação, apenas o sistema proposto passa a reunir, simultaneamente, suporte abrangente a cronograma, custos e riscos (F1, F2 e F3 em nível 2) e a possibilidade de ser hospedado *on-premise* com todas as suas funcionalidades de forma gratuita (F4 em nível 2), representado na tabela por *Taiga futuro*, oferecendo uma combinação de capacidades que não foi observada de maneira integrada nas demais soluções analisadas.

4 PROPOSTA DE SOLUÇÃO

Este capítulo apresenta a proposta de ampliação das funcionalidades do Taiga.io, com foco na incorporação de recursos de gestão de cronograma, gestão financeira e gestão de riscos. A solução proposta é organizada em épicos e histórias de usuário, de modo a alinhar o desenvolvimento às práticas de métodos ágeis já utilizadas pela própria ferramenta. Em seguida, são detalhados os requisitos funcionais por meio da descrição das histórias e de seus respectivos testes de aceitação.

4.1 SELEÇÃO DA FERRAMENTA

A escolha do Taiga.io como foco deste trabalho decorre, primeiramente, da análise comparativa realizada no Capítulo 3, que evidenciou seu destaque no critério de abertura e aderência ao modelo open source. No quadro 6 de funcionalidades, o Taiga é a única ferramenta que atinge o nível máximo em F4, por permitir implantação on-premise com acesso a todas as funcionalidades sem paywall funcional, o que o torna a opção mais alinhada à filosofia de software livre entre as soluções analisadas. Essa característica favorece tanto a transparência do código e a possibilidade de auditoria quanto a replicação do ambiente em contextos acadêmicos e organizacionais sem custos de licença.

Outro aspecto determinante para a seleção do Taiga.io é o conjunto de tecnologias que compõem sua arquitetura: back-end em Python/Django, banco de dados PostgreSQL, front-end em AngularJS e orquestração por contêineres Docker. Essa proximidade tecnológica reduz a curva de aprendizado para entendimento e extensão do código-fonte, aumenta a viabilidade de manutenção das funcionalidades propostas e reforça o potencial do trabalho como experiência de integração prática dos conteúdos estudados ao longo do curso.

Por fim, durante a avaliação inicial das ferramentas, o Taiga.io mostrou-se adequado ao uso pedagógico por concentrar seu fluxo de trabalho em elementos centrais dos métodos ágeis, como backlog, épicos, histórias e sprints, sem exigir configurações avançadas para sua utilização básica. Essa organização favorece a compreensão do fluxo de trabalho ágil por estudantes e iniciantes, ao mesmo tempo em que fornece base consistente para a introdução de conceitos mais avançados de gestão de cronograma, custos e riscos. Além disso, por se tratar de uma ferramenta open source com possibilidade de implantação local, o Taiga.io permite que esses conceitos sejam explorados em ambientes acadêmicos sem dependência de soluções proprietárias ou recursos pagos. Dessa forma, o Taiga.io se apresenta como um candidato equilibrado entre aderência à filosofia open source, afinidade tecnológica e potencial de uso didático em disciplinas de gerência de projetos.

4.2 LEVANTAMENTO DE REQUISITOS

O levantamento de requisitos foi conduzido a partir dos conceitos e boas práticas apresentados na fundamentação teórica (Capítulo 2) e foi reforçado pela análise comparativa das ferramentas correlatas descrita no Capítulo 3. Com base nas necessidades identificadas no Taiga.io em relação à gestão de cronograma, custos e riscos, foram definidos três épicos principais, cada um agrupando um conjunto de histórias de usuário que representam funcionalidades desejadas sob a perspectiva de usuários típicos da ferramenta (gerente de projetos, membro da equipe e *stakeholders*).

Os épicos definidos são:

- **E1 – Gestão de Cronograma:** engloba funcionalidades relacionadas à visualização e ao controle do cronograma do projeto, incluindo gráfico de Gantt, caminho crítico, folga das atividades e integração com sprints, épicos, histórias e tarefas;
- **E2 – Gestão Financeira:** abrange funcionalidades voltadas ao cadastro e acompanhamento de custos e receitas do projeto, com consolidação em uma visão financeira integrada;
- **E3 – Gestão de Riscos:** contempla funcionalidades voltadas ao registro, análise e monitoramento de riscos do projeto, permitindo documentar riscos, avaliar sua criticidade por meio de matrizes de probabilidade $\times$ impacto e acompanhar sua evolução ao longo do ciclo de vida do projeto.

Nas subseções a seguir são apresentadas as principais histórias de usuário levantadas para cada épico, em nível de granularidade adequado para orientar o detalhamento de requisitos na Seção 4.3.

4.2.1 Épico E1 – Gestão de Cronograma

O épico de gestão de cronograma tem como objetivo permitir que o Taiga ofereça uma visão estruturada do planejamento temporal do projeto, sem perder a aderência às práticas ágeis já existentes na ferramenta. As histórias de usuário associadas a esse épico são:

- **CRON-01** – Visualizar o cronograma do projeto em um gráfico de Gantt, a partir das tarefas e histórias já cadastradas no Taiga;
- **CRON-02** – Identificar visualmente o caminho crítico do projeto no gráfico de Gantt;
- **CRON-03** – Visualizar, para cada atividade, o quanto ela pode atrasar sem impactar a data de término do projeto (folga total);

4.2.1.1 Integração entre o planejamento ágil e a gestão de cronograma

A incorporação do gráfico de Gantt ao Taiga não pretende substituir o backlog, as sprints ou o planejamento iterativo, mas acrescentar uma visão complementar do cronograma. Os épicos, as histórias de usuário e as tarefas continuam sendo criados e priorizados segundo o fluxo ágil da ferramenta, enquanto o Gantt organiza esses mesmos itens em uma escala temporal, evidenciando suas datas, durações e relações de precedência. Dessa forma, a solução combina o planejamento estruturado do Gantt com a flexibilidade dos métodos ágeis, adotando uma abordagem híbrida (Jahr, 2014; Reiff; Schlegel, 2022).

No planejamento ágil, o trabalho é detalhado ao longo do projeto. A visão do produto orienta as entregas, as histórias de usuário descrevem as funcionalidades e as tarefas representam o trabalho necessário para implementá-las. Na solução proposta, esses mesmos elementos do Taiga são utilizados para compor o gráfico de Gantt, evitando a criação de um cronograma separado do backlog. Assim, conforme o backlog é refinado, as datas e durações podem ser atualizadas, mantendo o diagrama alinhado ao planejamento mais recente (PMI, 2017; Jahr, 2014).

A relação hierárquica entre épicos, histórias de usuário e tarefas não determina, por si só, a ordem temporal de execução. O fato de uma tarefa pertencer a uma história indica uma relação de decomposição do trabalho, mas não significa necessariamente que ela deva ocorrer antes ou depois das demais tarefas. As dependências surgem quando a execução ou a conclusão de um item condiciona outro, como nos casos em que uma funcionalidade necessita do resultado de outra, uma atividade de interface depende da implementação de uma API, um teste depende da conclusão do componente testado ou uma entrega depende de um elemento externo ao projeto. Estudos sobre desenvolvimento ágil em larga escala mostram que dependências entre histórias e equipes permanecem relevantes mesmo em ciclos iterativos e podem afetar releases, coordenação e prazos (Biesialska; Franch; Muntés-Mulero, 2021; Stray; Moe; Aasheim, 2019).

Na solução proposta, as dependências não são inferidas automaticamente a partir da hierarquia dos itens ou de sua associação a uma sprint. Elas são identificadas pela equipe durante o refinamento do backlog e o planejamento das entregas ou das iterações, sendo registradas no gráfico de Gantt por meio da ligação explícita entre uma atividade predecessora e uma atividade sucessora. Essas relações são utilizadas para representar a sequência lógica de execução e fornecer a base para o cálculo do caminho crítico e da folga total. Como o planejamento ágil está sujeito a revisões, as dependências e as datas podem ser atualizadas quando ocorrerem mudanças no backlog, nas estimativas ou na estratégia de implementação.

4.2.2 Épico E2 – Gestão Financeira

O épico de gestão financeira busca prover ao Taiga a capacidade de representar custos e receitas do projeto, permitindo que o gerente acompanhe orçamento, custos planejados e custos realizados em um único painel. As histórias levantadas são:

- **CUST-01** – Cadastrar o custo planejado de uma tarefa;
- **CUST-02** – Registrar as horas trabalhadas por integrante em uma tarefa, associando uma taxa horária para cálculo de custo realizado;
- **CUST-03** – Definir o valor de receita contratada do projeto (orçamento ou valor vendido);
- **CUST-04** – Visualizar uma página de resumo financeiro do projeto, consolidando custos planejados, custos realizados, receita e margem;

4.2.3 Épico E3 – Gestão de Riscos

O épico de gestão de riscos tem como objetivo permitir que o Taiga registre, avalie e acompanhe riscos de forma estruturada, alinhando-se às boas práticas de gerenciamento de projetos. As principais histórias de usuário são:

- **RISK-01** – Cadastrar, listar, editar e excluir riscos, permitindo o gerenciamento completo (CRUD) de um registro de riscos associado ao projeto. A modelagem contempla a captura de informações como causa, consequência, classificação de probabilidade e impacto, responsável atribuído, estratégia de mitigação e plano de resposta;
- **RISK-02** – Visualizar os riscos do projeto em uma matriz de probabilidade $\times$ impacto, distribuindo e agrupando automaticamente os riscos de acordo com seus níveis de probabilidade e gravidade, de forma a apoiar a priorização e a tomada de decisão;
- **RISK-03** – Acompanhar a evolução dos riscos ao longo do tempo por meio de um dashboard analítico, contendo um gráfico que apresenta a variação do risco do projeto, alimentado por snapshots históricos gerados automaticamente ou manualmente a cada alteração relevante ou início de nova Sprint;

4.3 ANÁLISE DE REQUISITOS

Esta seção detalha alguma das histórias de usuário para cada épico (o restante da análise das histórias estão localizadas no apêndice A deste documento), descrevendo-as no formato “Eu, como <tipo de usuário>...” e definindo testes de aceitação correspondentes. O objetivo é tornar explícitos os critérios que permitirão verificar se a implementação atende

às necessidades identificadas no levantamento de requisitos. Para cada épico, é analisada uma história de usuário representativa, enquanto as demais histórias são detalhadas no Apêndice A.

4.3.1 Histórias do Épico E1 – Gestão de Cronograma

História CRON-01 – Visualizar cronograma em gráfico de Gantt

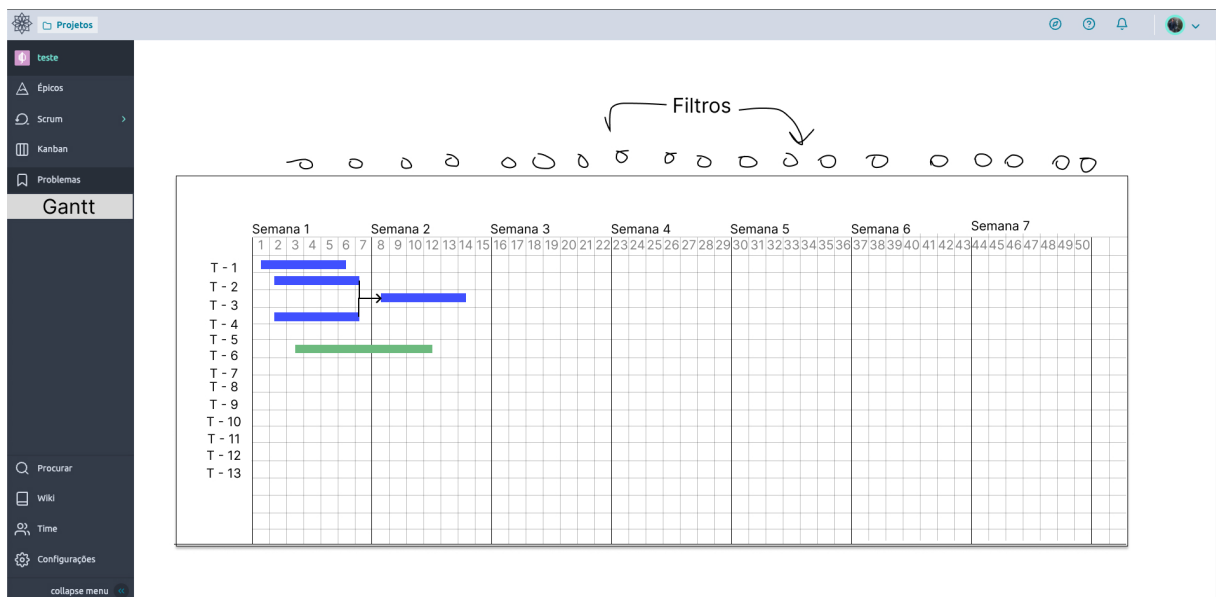
Descrição da história:

Eu, como gerente de projetos, desejo visualizar o cronograma do projeto em um gráfico de Gantt baseado nas tarefas e histórias já cadastradas no Taiga, para compreender de forma clara a organização no tempo das atividades e seus relacionamentos.

Teste de aceitação:

- Dado que existam tarefas e histórias com datas de início e término definidas, quando o gerente de projetos acessar a nova aba de cronograma do projeto, então deve ser exibido um gráfico de Gantt contendo barras para cada atividade, listadas à esquerda em uma hierarquia demonstrando suas relações, posicionadas conforme suas datas de início e fim.

Figura 28 – Protótipo Gráfico de Gantt



4.3.2 Histórias do Épico E2 – Gestão Financeira

História CUST-02 – Registrar horas trabalhadas e calcular custo realizado

Descrição da história:

Eu, como integrante da equipe de projeto, desejo registrar as horas que trabalhei em uma tarefa, para que o sistema calcule automaticamente o custo realizado com base na taxa horária configurada.

Testes de aceitação:

- Dado que exista uma tarefa atribuída a um integrante com uma taxa horária previamente configurada, quando esse integrante registrar um apontamento de horas para a tarefa e salvar o registro, então o sistema deve armazenar a quantidade de horas trabalhadas e calcular o custo realizado correspondente (horas trabalhadas multiplicadas pela taxa horária).

- Dado que existam múltiplos registros de horas para a mesma tarefa, quando o gerente de projetos acessar o detalhamento da tarefa, então o sistema deve exibir o total de horas trabalhadas e o custo realizado acumulado, calculado a partir da soma de todos os registros de horas associados.

4.3.3 Histórias do Épico E3 – Gestão de Riscos

História RISK-02 – Visualizar matriz probabilidade $\times$ impacto

Descrição da história:

Eu, como gerente de projetos, desejo visualizar os riscos do projeto em uma matriz probabilidade $\times$ impacto, para identificar rapidamente quais riscos são mais críticos e priorizar ações de resposta.

Teste de aceitação:

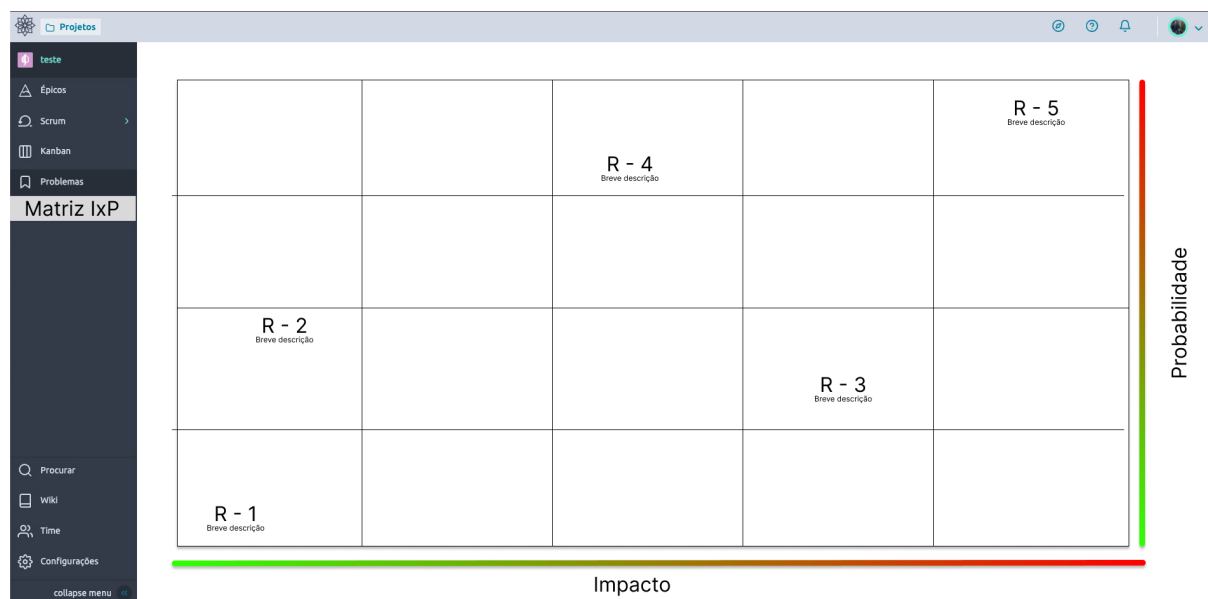
- Dado que existam riscos cadastrados com probabilidade e impacto definidos de acordo com escalas pré-configuradas, quando o gerente acessar a visualização de matriz, então cada risco deve ser representado em uma célula correspondente à combinação de probabilidade e impacto, com identificação clara do risco.

- Dado que o gerente altere a probabilidade ou o impacto de um risco no registro de riscos, quando a matriz for recarregada, então a posição do risco na matriz deve ser atualizada para refletir os novos valores.

4.4 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada a proposta de solução para ampliação das funcionalidades do Taiga.io, a partir de uma seleção fundamentada da ferramenta e de um levantamento estruturado de requisitos. Inicialmente, justificou-se a escolha do Taiga.io com base em sua aderência ao modelo open source, na possibilidade de implantação sem restrições funcionais e na afinidade tecnológica com as competências dos autores. Em seguida, foram definidos três épicos principais, gestão de cronograma, gestão financeira e gestão de riscos, cada um desdobrado em histórias de usuário que expressam as funcionalidades desejadas sob a perspectiva de gerentes de projeto, membros de equipe e demais interessados. Para algumas dessas histórias, foram descritos testes de aceitação e protótipos de interface, estabelecendo critérios objetivos para avaliação da futura implementação.

Figura 29 – Prototipo Matriz Pxl



A partir desse escopo funcional, os próximos capítulos passam a tratar da materialização da solução proposta. Em um primeiro momento, o trabalho se dedica ao estudo detalhado do Taiga.io, abordando sua arquitetura, o código-fonte e o modelo de dados, de modo a identificar pontos de extensão e integrações necessárias para os novos módulos. Na sequência, é descrito o desenvolvimento incremental das funcionalidades de cronograma, finanças e riscos, bem como a forma como são incorporadas à ferramenta existente. Posteriormente, são apresentados os procedimentos de verificação e teste adotados, baseados nos critérios de aceitação definidos neste capítulo, avaliando se as implementações atendem aos requisitos e se se integram adequadamente aos fluxos de trabalho já oferecidos pelo Taiga.io.

5 DESENVOLVIMENTO DAS NOVAS FUNCIONALIDADES DO TAIGA

Este capítulo descreve o desenvolvimento das novas funcionalidades propostas para o Taiga, contemplando as etapas de estudo da arquitetura da ferramenta, preparação do ambiente de desenvolvimento, definição dos pontos de extensão e implementação incremental dos módulos de gestão de cronograma, gestão financeira e gestão de riscos. O desenvolvimento foi conduzido a partir dos requisitos apresentados no capítulo anterior, buscando integrar os novos recursos aos fluxos já existentes da aplicação, de modo a preservar sua lógica de uso baseada em projetos, épicos, histórias, tarefas e sprints. Ao longo do processo, foram realizadas adaptações no back-end, no front-end e no modelo de dados, com o objetivo de incorporar as funcionalidades planejadas de forma compatível com a estrutura original do Taiga.

Para organizar a apresentação do desenvolvimento, este capítulo está dividido em cinco etapas principais. Inicialmente, descreve-se a arquitetura do Taiga, destacando a organização geral do sistema, seus principais componentes e a relação entre back-end, front-end, banco de dados e serviços de apoio pertinentes ao desenvolvimento deste trabalho. Em seguida, apresenta-se a configuração do ambiente de desenvolvimento, incluindo as tecnologias utilizadas para execução, modificação e apoio à implementação da aplicação e posteriormente como iniciar a aplicação em uma nova instalação.

Na sequência, descreve-se como as funcionalidades foram implementadas no sistema, relacionando as decisões de desenvolvimento aos requisitos definidos neste documento. Depois, são apresentados os testes de aceitação realizados, acompanhados de evidências que demonstram o funcionamento das funcionalidades implementadas. Por fim, são identificados os arquivos originais do Taiga que foram alterados durante o desenvolvimento, destacando os pontos que demandam maior atenção em futuras atualizações do código-fonte da ferramenta.

O código-fonte resultante das adaptações e funcionalidades desenvolvidas neste trabalho está disponível no repositório <https://codigos.ufsc.br/gqs/taiga-gqs>.

5.1 AMBIENTE DE DESENVOLVIMENTO E FERRAMENTAS UTILIZADAS

O desenvolvimento das funcionalidades propostas foi realizado em ambiente local com sistema operacional macOS e Ubuntu 24.04 utilizando o Visual Studio Code como editor principal de código, além do Google Chrome para execução, inspeção e teste da interface da aplicação. O ambiente de execução do Taiga foi apoiado por Docker e Docker Compose, responsáveis por coordenar os serviços necessários ao funcionamento da aplicação, como back-end, front-end, banco de dados, mensageria, processamento assíncrono e gateway HTTP. Essa combinação permitiu editar os repositórios localmente, executar os serviços de forma containerizada e verificar o comportamento das funcionalidades diretamente no navegador.

5.2 ARQUITETURA DO SISTEMA

O Taiga consiste em uma plataforma open source para gerenciamento ágil de projetos composta por um conjunto de serviços que atuam de forma integrada. Sua arquitetura é organizada de maneira modular, separando as responsabilidades de interface com o usuário, regras de negócio, persistência de dados e execução de serviços auxiliares. Essa abordagem favorece a manutenção da aplicação e possibilita a evolução independente de seus componentes. Os principais componentes da arquitetura são apresentados no Quadro 7.

Quadro 7 – Serviços que compõem a arquitetura da plataforma Taiga

Serviço	Responsabilidade
taiga-back	Disponibilização da API REST, implementação das regras de negócio, mapeamento dos modelos de dados e gerenciamento das migrações do banco de dados.
taiga-front	Interface principal da aplicação, responsável pela interação dos usuários com a plataforma.
taiga-front-next	Implementação de componentes modernos de interface integrados à aplicação principal.
taiga-events	Gerenciamento das notificações e atualizações em tempo real entre os usuários da plataforma.
taiga-async	Execução de tarefas assíncronas, incluindo envio de notificações, exportação de dados e processamento de integrações.
taiga-protected	Controle de acesso a arquivos protegidos, anexos e recursos de mídia da plataforma.
taiga-gateway	Roteamento centralizado das requisições e comunicação entre os serviços internos da aplicação.
taiga-events-rabbitmq	Serviço de mensageria utilizado pelas notificações em tempo real.
taiga-async-rabbitmq	Serviço de mensageria utilizado pelo processamento de tarefas assíncronas.
taiga-db	Armazenamento e persistência das informações da aplicação.

Fonte: Elaborado pelos autores (2026).

De forma geral, o fluxo de funcionamento da aplicação inicia-se na interface web. As requisições são encaminhadas ao gateway da aplicação, responsável por direcionar o tráfego aos serviços adequados. O frontend consome a API REST disponibilizada pelo backend, que executa as regras de negócio e realiza a persistência das informações no banco de dados. Paralelamente, serviços auxiliares são responsáveis por atividades complementares, como processamento assíncrono de tarefas e comunicação entre componentes.

A Figura 30 apresenta uma visão simplificada da arquitetura da plataforma Taiga, destacando os principais componentes envolvidos no fluxo de execução da aplicação. A interação dos usuários ocorre por meio do navegador web, que acessa a interface da

plataforma através do frontend. Esse frontend é composto por dois projetos distintos: o *taiga-front* e o *taiga-front-next*.

O *taiga-front* corresponde à aplicação principal do sistema e é desenvolvido utilizando AngularJS 1.5, além de tecnologias complementares como CoffeeScript e Jade para implementação dos controladores e templates da interface. Já o *taiga-front-next* é desenvolvido utilizando Angular, TypeScript e SCSS. Esse projeto surge como parte de uma iniciativa de modernização tecnológica da plataforma, cujo objetivo consiste em migrar gradualmente a interface legada para versões mais recentes do framework Angular. Entretanto, essa migração não é concluída integralmente, resultando em uma arquitetura híbrida na qual apenas determinados módulos e componentes passam a ser implementados no *taiga-front-next*, enquanto a maior parte da aplicação permanece no *taiga-front*. Dessa forma, ambos os projetos coexistem e operam de maneira integrada. Na prática, componentes desenvolvidos no *taiga-front-next* são incorporados ao *taiga-front*, permitindo a utilização de tecnologias mais recentes sem a necessidade de reescrever integralmente a aplicação legada.

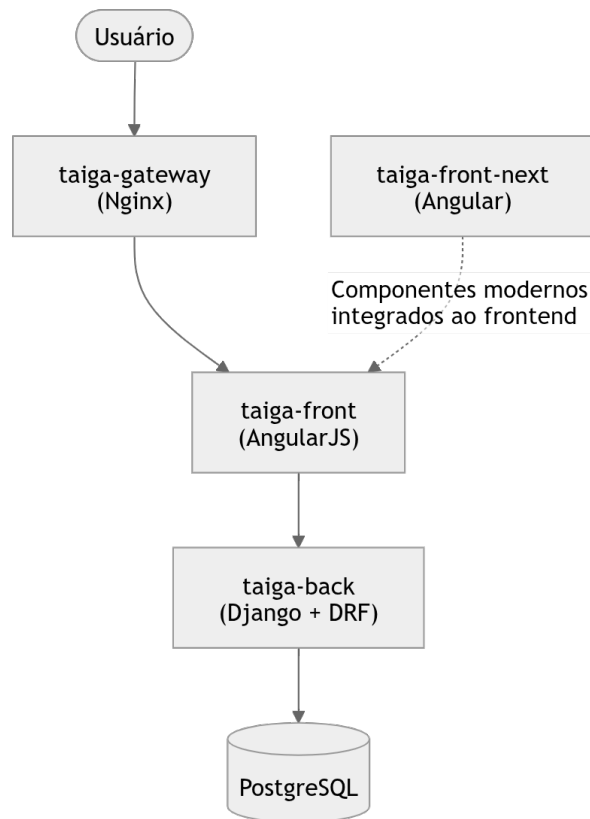
As requisições realizadas pela interface são encaminhadas ao *taiga-back*, componente responsável pelo processamento das regras de negócio da aplicação. O backend é desenvolvido em Python utilizando o framework Django em conjunto com o Django REST Framework (DRF), disponibilizando uma API REST consumida pelo frontend. Além das operações de negócio, esse componente também é responsável pela validação de dados, controle de permissões, persistência das informações e integração com os demais serviços da plataforma.

Os dados da aplicação são armazenados no banco de dados PostgreSQL. Complementarmente, o Taiga dispõe de serviços auxiliares responsáveis por funcionalidades específicas. Entre eles destacam-se os serviços de comunicação em tempo real, implementados por meio de WebSockets, os mecanismos de processamento assíncrono de tarefas, executados através do Celery, e os serviços de controle de acesso a arquivos protegidos e recursos de mídia.

De forma geral, o fluxo de comunicação da aplicação inicia-se na interação do usuário com a interface. As ações executadas no frontend resultam em requisições à API disponibilizada pelo backend, que processa as regras de negócio correspondentes e realiza consultas ou alterações no banco de dados. Quando necessário, serviços auxiliares são acionados para execução de tarefas específicas, como notificações em tempo real ou processamento assíncrono de operações. Essa organização em componentes especializados favorece a manutenção da plataforma e permite a incorporação de novas funcionalidades sem impactar significativamente os demais módulos do sistema.

Embora a plataforma seja composta por diversos serviços, o desenvolvimento descrito neste trabalho concentra-se principalmente nos repositórios *taiga-back*, *taiga-front* e *taiga-front-next*. O primeiro é responsável pelas regras de negócio, disponibilização da

Figura 30 – Visão geral da arquitetura do Taiga



Fonte: Elaborado pelos autores (2026).

API REST e persistência dos dados. O segundo corresponde à interface principal utilizada pelos usuários, enquanto o terceiro abriga componentes modernos de interface integrados ao frontend existente.

Esses três repositórios constituem os principais pontos de modificação para incorporação das funcionalidades propostas neste trabalho. As alterações realizadas em cada um deles para suportar os módulos de gestão de cronograma, gestão financeira e gestão de riscos são detalhadas nas seções subsequentes.

5.3 CONFIGURAÇÃO DO AMBIENTE DE DESENVOLVIMENTO

Inicialmente, ao consultar a documentação do Taiga, verificou-se que a instalação recomendada previa a configuração do ambiente em uma máquina com Ubuntu 20.04, com os repositórios da aplicação organizados no diretório `/home` do sistema operacional. Essa recomendação representou uma limitação prática, pois um dos ambientes de desenvolvimento utilizados neste trabalho executava o sistema operacional macOS. Dessa forma, tornou-se necessário buscar alternativas que permitissem executar, modificar e testar o Taiga sem depender rigidamente da configuração indicada na documentação consultada.

A primeira alternativa considerada consistiu em inicializar separadamente os servi-

ços necessários à aplicação por meio da containerização dos repositórios do Taiga envolvidos no desenvolvimento deste trabalho. Embora essa abordagem fosse tecnicamente viável, ela apresentou diversas dificuldades operacionais, pois exigia a configuração individual do back-end, do front-end, do banco de dados, dos serviços de mensageria e dos demais componentes necessários à execução da aplicação. Durante o estudo da estrutura dos repositórios do Taiga, identificou-se uma alternativa mais adequada para o contexto do trabalho.

Conforme mencionado no capítulo anterior, o Taiga possui um repositório chamado **taiga-docker**, que facilita a inicialização da aplicação ao reduzir a necessidade de configurações diretamente no sistema hospedeiro. Esse repositório utiliza Docker Compose para coordenar os serviços necessários ao funcionamento do sistema, favorecendo a portabilidade do ambiente por meio da containerização. Com isso, tornou-se possível executar componentes como PostgreSQL, RabbitMQ, back-end, front-end, processamento assíncrono, serviço de eventos e gateway HTTP de forma integrada.

Entretanto, a configuração padrão do **taiga-docker** utiliza imagens previamente construídas e disponibilizadas pela organização do Taiga. Essa característica é adequada para a execução da aplicação, mas não atende integralmente às necessidades de um ambiente de desenvolvimento, pois alterações realizadas nos repositórios locais, especialmente em **taiga-back** e **taiga-front**, não seriam refletidas diretamente nos serviços executados a partir das imagens oficiais. Assim, foi necessária uma adaptação da configuração original.

O Docker Compose permite combinar múltiplos arquivos de configuração, de modo que um arquivo complementar possa sobrescrever ou estender definições presentes no arquivo principal. A partir desse recurso, adotou-se o repositório **taiga-docker** como base para inicialização do sistema, mas foram alterados os serviços diretamente relacionados ao desenvolvimento deste trabalho. Em vez de utilizar apenas imagens prontas, os serviços **taiga-back**, **taiga-async** e **taiga-front** passaram a utilizar os arquivos locais dos respectivos repositórios.

No caso do back-end, os serviços **taiga-back** e **taiga-async** foram configurados para serem construídos a partir do repositório local **taiga-back**, utilizando o **Dockerfile** já existente no próprio projeto. Além disso, o código-fonte local foi montado como volume dentro dos contêineres, permitindo que as alterações realizadas no repositório fossem utilizadas durante a execução da aplicação.

```
taiga-back:
  build:
    context: ../taiga-back
    dockerfile: docker/Dockerfile
  entrypoint: ["/bin/bash", "/taiga-back/docker/entrypoint.sh"]
  environment:
    DEBUG: "True"
```

```
command: ["--log-level=debug"]
volumes:
  - ../taiga-back:/taiga-back
  - taiga-static-data:/taiga-back/static
  - taiga-media-data:/taiga-back/media
```

Código 5.1 – Configuração local do serviço de back-end no Docker Compose

Outra particularidade do desenvolvimento no back-end esteve relacionada às alterações no modelo de dados. Como o Taiga utiliza Django, mudanças em modelos persistidos exigem a geração de arquivos de migração por meio do comando `python manage.py makemigrations`. Após a criação desses arquivos, as alterações precisam ser aplicadas ao banco de dados com `python manage.py migrate`, comando que também é executado durante a inicialização do serviço `taiga-back` no ambiente containerizado. Esse fluxo não foi introduzido por este trabalho, mas corresponde ao mecanismo padrão do Django já utilizado pelo repositório oficial `taiga-back`. Da mesma forma, o arquivo `manage.py` é parte do código original do Taiga e não foi modificado para a configuração do ambiente de desenvolvimento.

Para o front-end, foi adotada uma estratégia semelhante. Em vez da imagem oficial `taigaio/taiga-front`, utilizou-se uma imagem Node.js, com o diretório local `taiga-front` montado no contêiner. Nessa configuração, o próprio contêiner instala as dependências com `npm ci`, prepara o arquivo de configuração da interface e executa o servidor de desenvolvimento com `npm start`. Essa adaptação permitiu trabalhar com as tecnologias originais do front-end do Taiga, como AngularJS, CoffeeScript, Jade/Pug, SCSS, Node.js, npm e Gulp.

```
taiga-front:
  image: node:16-bullseye
  working_dir: /taiga-front
  environment:
    CHOKIDAR_USEPOLLING: "true"
    CHOKIDAR_INTERVAL: "1000"
  command:
    - /bin/bash
    - -lc
    - >
      if [ ! -d node_modules ] || [ ! -f node_modules/.package-lock.json ] || !
      cmp -s package-lock.json node_modules/.package-lock.json; then
        npm ci --legacy-peer-deps &&
        cp package-lock.json node_modules/.package-lock.json;
      fi;
    mkdir -p dist;
    ln -sf /taiga-front/conf/conf.example.json /taiga-front/dist/conf.json;
```

```
npm start
volumes:
  - ../taiga-front:/taiga-front
  - taiga-front-node-modules:/taiga-front/node_modules
  - ./conf.front.dev.json:/taiga-front/conf/conf.example.json:ro
```

Código 5.2 – Configuração local do serviço de front-end no Docker Compose

Outra adaptação necessária esteve relacionada ao gateway da aplicação. Na configuração original, o front-end é servido por uma imagem previamente construída. No ambiente de desenvolvimento configurado neste trabalho, o gateway Nginx foi ajustado para encaminhar as requisições da interface ao servidor de desenvolvimento do front-end, enquanto as chamadas de API continuaram sendo direcionadas ao serviço `taiga-back`. Dessa forma, a aplicação permaneceu acessível por uma única URL local, simplificando os testes de integração durante o desenvolvimento.

```
location / {
    proxy_pass http://taiga-front:9001;
}

location /api/ {
    proxy_pass http://taiga-back:8000;
}

location /events {
    proxy_pass http://taiga-events:8888/events;
}
```

Código 5.3 – Trecho da configuração local do gateway Nginx

Para reduzir o esforço de inicialização do ambiente e evitar a repetição manual de comandos, foi criado um script auxiliar denominado `start-taiga-local.sh`. Esse script centralizou a preparação e a execução do ambiente local, realizando verificações de dependências, sincronização de submódulos, criação de arquivos de configuração quando necessário, carregamento de variáveis de ambiente e inicialização dos serviços na ordem adequada.

Uma particularidade do back-end consistiu na necessidade do arquivo `taiga-back/settings/config.py`. Para evitar falhas na primeira execução, o script verifica se esse arquivo existe e, caso necessário, cria uma cópia a partir da configuração fornecida pelo próprio repositório.

```
if [ ! -f "${ROOT_DIR}/taiga-back/settings/config.py" ]; then
    echo "[1/7] Criando taiga-back/settings/config.py a partir de docker/config.py..."
```

```
cp "${ROOT_DIR}/taiga-back/docker/config.py" "${ROOT_DIR}/taiga-back/settings
/config.py"
fi
```

Código 5.4 – Criação automática do arquivo de configuração do back-end

Além disso, o script também tratou variáveis de ambiente exigidas pelo Docker Compose. Algumas dessas variáveis apresentavam nomes diferentes entre versões ou configurações do Taiga, por isso foram definidos valores padrão e mapeamentos de compatibilidade. Essa medida tornou a inicialização menos sensível a diferenças entre arquivos `.env` e reduziu erros durante a preparação do ambiente.

```
export SECRET_KEY="${SECRET_KEY:-${TAIGA_SECRET_KEY:-insecure-secret-key-change
-in-production}}"
export RABBITMQ_USER="${RABBITMQ_USER:-${TAIGA_ASYNC_RABBITMQ_USER:-${
TAIGA_EVENTS_RABBITMQ_USER:-taiga}}}"
export RABBITMQ_PASS="${RABBITMQ_PASS:-${TAIGA_ASYNC_RABBITMQ_PASS:-${
TAIGA_EVENTS_RABBITMQ_PASS:-taiga}}}"
export RABBITMQ_VHOST="${RABBITMQ_VHOST:-${TAIGA_ASYNC_RABBITMQ_VHOST:-${
TAIGA_EVENTS_RABBITMQ_VHOST:-taiga}}}"
```

Código 5.5 – Mapeamento de variáveis de ambiente utilizadas pelo Docker Compose

No caso do front-end, uma particularidade relevante foi a persistência das dependências instaladas pelo npm. Como a instalação completa das dependências pode ser demorada, utilizou-se um volume Docker específico para o diretório `node_modules`. O comando de inicialização verifica se as dependências já existem e se correspondem ao arquivo `package-lock.json`; caso contrário, executa novamente `npm ci`. Essa solução evitou reinstalações desnecessárias e, ao mesmo tempo, manteve o ambiente coerente quando ocorreram alterações nas dependências.

Além do front-end legado, algumas partes da interface do Taiga dependem do repositório `taiga-front-next`. Conforme apresentado anteriormente, esse repositório contém componentes desenvolvidos em Angular mais recente, que são compilados como *web components* e integrados ao front-end legado por meio do arquivo `elements.js`. Dessa forma, alterações realizadas em componentes do `taiga-front-next` não são refletidas diretamente apenas com a execução do `taiga-front`; é necessário compilar esse repositório e copiar o artefato gerado para o repositório `taiga-front`.

Para automatizar esse processo, foi utilizado o script `compile-front-end-next.sh`. Esse script verifica a existência dos diretórios necessários, seleciona a versão de Node.js definida pelo arquivo `.nvmrc` do `taiga-front-next`, executa os comandos de compilação do *web component* e copia o arquivo final `elements.js` para o diretório do `taiga-front`. Essa etapa é necessária especialmente quando são realizadas alterações em elementos de navegação ou componentes compartilhados provenientes do front-end mais recente.

```
NEXT_ELEMENTS="${NEXT_DIR}/dist/elements/elements.js"
FRONT_ELEMENTS="${FRONT_DIR}/elements.js"
NVMRC_PATH="${NEXT_DIR}/.nvmrc"

REQUIRED_NODE_VERSION="$(tr -d '[:space:]' < "${NVMRC_PATH}")"

echo "[1/4] Selecionando Node ${REQUIRED_NODE_VERSION} via nvm..."
load_nvm
nvm use "${REQUIRED_NODE_VERSION}"

echo "[2/4] Compilando webcomponent em taiga-front-next..."
(
  cd "${NEXT_DIR}"
  npm run build:elements
  npm run pack:elements
)

echo "[3/4] Copiando elements.js para taiga-front..."
cp "${NEXT_ELEMENTS}" "${FRONT_ELEMENTS}"
```

Código 5.6 – Compilação do front-end moderno e cópia do artefato para o front-end legado

Por fim, o script passou a interromper os contêineres em execução, reconstruir as imagens locais necessárias e inicializar os serviços principais do Taiga. Após a verificação dos serviços base, o gateway é iniciado e a aplicação fica disponível em <http://localhost:9000>.

```
compose_base down --remove-orphans

compose_base build taiga-back taiga-async

compose_base up -d \
  taiga-db \
  taiga-async-rabbitmq \
  taiga-events-rabbitmq \
  taiga-events \
  taiga-protected \
  taiga-back \
  taiga-async \
  taiga-front

compose_base up -d taiga-gateway
```

Código 5.7 – Inicialização dos serviços locais do Taiga

A configuração final do ambiente de desenvolvimento utilizou Docker e Docker Compose como base para orquestração dos serviços. O banco de dados utilizado foi o PostgreSQL, enquanto o RabbitMQ foi empregado como serviço de mensageria para os componentes assíncronos e de eventos do Taiga. O back-end foi executado em Python com Django, Django REST Framework, Gunicorn e Celery. Já o front-end foi executado em Node.js, utilizando AngularJS, CoffeeScript, Jade/Pug, SCSS, npm e Gulp para construção e execução da interface.

Com essa configuração, o ambiente de desenvolvimento combinou a infraestrutura fornecida pelo `taiga-docker` com os repositórios locais modificáveis do projeto. Essa solução permitiu desenvolver e testar as novas funcionalidades mantendo os principais serviços do Taiga containerizados, reduzindo diferenças entre sistemas operacionais e tornando o processo de inicialização mais reproduzível.

Em síntese, o fluxo de desenvolvimento adotado baseou-se no uso do script `start-taiga-local.sh` para inicialização e reinicialização da aplicação local, especialmente em situações nas quais alterações no código não eram detectadas automaticamente pelos serviços em execução. As modificações no back-end e no front-end legado eram realizadas diretamente nos repositórios locais `taiga-back` e `taiga-front`, montados nos respectivos contêineres. Já as alterações em componentes do `taiga-front-next` exigiam uma etapa adicional de compilação por meio do script `compile-front-end-next.sh`, que gerava o artefato `elements.js` e o copiava para o front-end legado.

5.4 EXECUÇÃO DA APLICAÇÃO EM UMA NOVA INSTALAÇÃO

Além da configuração do ambiente descrita na seção anterior, é importante registrar o procedimento necessário para colocar a aplicação em execução a partir de uma instalação limpa, de modo que o sistema possa ser reproduzido em outra máquina sem depender do histórico de configuração já existente no ambiente dos autores. Esta seção apresenta os pré-requisitos de software e o passo a passo dos comandos que devem ser executados para clonar o repositório, construir as imagens locais, subir os serviços containerizados e disponibilizar a aplicação no navegador. O fluxo descrito é o mesmo automatizado pelos scripts auxiliares apresentados na seção de configuração do ambiente de desenvolvimento, de forma que a maior parte das etapas é executada de maneira centralizada, reduzindo a chance de erros manuais durante a preparação.

5.4.1 Pré-requisitos

A execução do sistema depende de um conjunto reduzido de ferramentas instaladas previamente na máquina hospedeira, uma vez que os componentes da aplicação (back-end,

front-end, banco de dados, mensageria e serviços auxiliares) são executados de forma containerizada. Os pré-requisitos são os seguintes:

- Sistema operacional **Linux** (Ubuntu 24.04) ou **macOS**, com interpretador de comandos compatível com **bash** para execução dos scripts auxiliares.
- **Docker Engine** e **Docker Compose** (plugin `docker compose`, versão 2), responsáveis por construir e orquestrar os serviços da aplicação.
- Recursos mínimos de hardware suficientes para executar os contêineres simultaneamente, além da porta 9000 disponível na máquina, utilizada pelo gateway para expor a aplicação.
- Opcionalmente, **nvm** e **Node.js** na versão definida pelo arquivo `.nvmrc` do `taiga-front-next`, necessários apenas quando houver alterações em componentes do front-end moderno que exijam recompilação do artefato `elements.js`.

5.4.2 Passo a passo de execução

Com os pré-requisitos atendidos, a aplicação é colocada em execução por meio da sequência de comandos descrita a seguir. O primeiro passo consiste em clonar o repositório principal já com os submódulos, garantindo que os três repositórios da aplicação sejam obtidos em conjunto.

```
git clone https://codigos.ufsc.br/gqs/taiga-gqs.git
cd taiga-gqs
```

Código 5.8 – Clonagem do repositório com submódulos

Em seguida, executa-se o script `start-taiga-local.sh`, localizado na raiz do repositório, responsável por automatizar toda a preparação e a inicialização do ambiente. Esse script cria o arquivo de configuração `taiga-back/settings/config.py` a partir do modelo do repositório quando necessário, carrega as variáveis de ambiente, interrompe contêineres remanescentes, constrói as imagens locais de back-end e processamento assíncrono e sobe os serviços na ordem adequada.

```
./start-taiga-local.sh
```

Código 5.9 – Inicialização do ambiente local

Ao final da execução, o script aguarda a disponibilidade dos serviços base, inicia o gateway e informa que a aplicação está acessível no endereço `http://localhost:9000`. Nesse momento, o sistema já está em funcionamento, porém ainda sem usuários cadastrados em uma instalação totalmente nova. Para criar o primeiro usuário administrador, utiliza-se o utilitário de gerenciamento do back-end por meio do script `taiga-manage.sh`,

disponível no diretório `taiga-docker`, que executa comandos do `manage.py` do Django dentro de um contêiner dedicado.

```
cd taiga-docker
./taiga-manage.sh createsuperuser
```

Código 5.10 – Criação do usuário administrador inicial

Após informar nome de usuário, e-mail e senha solicitados pelo comando, o usuário administrador é criado no banco de dados, permitindo o acesso à aplicação pelo navegador no endereço `http://localhost:9000`. A partir desse ponto, é possível autenticar-se, criar projetos e utilizar as funcionalidades de cronograma, gestão financeira e gestão de riscos desenvolvidas neste trabalho.

Por fim, quando o desenvolvimento envolver alterações em componentes do `taiga-front-next`, é necessário recompilar o webcomponent e copiar o artefato gerado para o front-end legado, etapa realizada pelo script `compile-front-end-next.sh`. Essa recompilação não é exigida em uma execução padrão da aplicação, sendo necessária apenas para refletir mudanças feitas no front-end-next.

```
cd taiga-front-next
npm install
cd ..
./compile-front-end-next.sh
```

Código 5.11 – Recompilação opcional do front-end moderno

Por fim, vale registrar um comportamento esperado na primeira execução em uma instalação nova. Caso o navegador apresente um erro 502 ao acessar `http://localhost:9000`, recomenda-se verificar os registros (*logs*) do contêiner `taiga-front`. Se a última mensagem exibida for `npm WARN deprecated messageformat@1.0.2: Package renamed as '@messageformat/core', see messageformat.github.io for more details. 'messageformat' will eventually provide a polyfill for Intl.MessageFormat, once it's been defined by Unicode & ECMA.`, isso pode indicar que o serviço de front-end ainda está instalando as dependências necessárias para sua execução. Nesse caso, basta aguardar a conclusão do processo: o front-end é considerado pronto para acesso quando os *logs* apresentam uma mensagem no formato `Finished 'default' after ...s`, momento em que a aplicação passa a responder normalmente no endereço informado.

5.5 IMPLEMENTAÇÃO DAS FUNCIONALIDADES PROPOSTAS

Nesta seção, são apresentados os resultados da implementação das funcionalidades propostas a partir dos requisitos descritos no Capítulo 4, Seção 4.2. Para cada conjunto de requisitos, organizados nos épicos de gestão de cronograma, gestão financeira e gestão

de riscos, descreve-se a solução implementada e suas principais funcionalidades. Também são apresentados os recursos adicionados ao sistema e sua contribuição para ampliar o suporte da ferramenta a práticas de gerenciamento de projetos.

5.5.1 Visão geral das alterações no projeto do sistema

Além da criação das telas descritas nas subseções anteriores, o desenvolvimento das funcionalidades propostas exigiu alterações estruturais no projeto do sistema, tanto em nível de código quanto de modelo de dados. Esta subseção apresenta uma visão consolidada dessas alterações, descrevendo os novos módulos de software, as classes e regras de negócio introduzidas e as novas tabelas criadas no banco de dados.

As três funcionalidades foram implementadas como aplicativos Django independentes, adicionados à configuração do *back-end* (`INSTALLED_APPS`) e seguindo o mesmo padrão de organização já utilizado pelos módulos nativos do Taiga. Foram criados os aplicativos `taiga.projects.schedule` (gestão de cronograma), `taiga.projects.finances` (gestão financeira) e `taiga.projects.risks` (gestão de riscos). Essa escolha de projeto buscou isolar o novo código do núcleo original da ferramenta, reduzindo a superfície de conflito em futuras atualizações e mantendo a coesão de cada domínio.

Cada aplicativo foi estruturado em camadas, refletindo a arquitetura do Taiga descrita na Seção 5.2. As principais classes e responsabilidades introduzidas são:

- **Modelos (`models.py`):** classes que mapeiam as entidades de domínio para as tabelas do banco de dados por meio do ORM do Django, incluindo restrições de integridade e índices;
- **Serializadores (`serializers.py`) e validadores (`validators.py`):** responsáveis por converter os modelos na representação JSON consumida pelo *front-end* e por validar os dados recebidos nas requisições;
- **Visões (`api.py`):** os *viewsets* do Django REST Framework que expõem os *endpoints* da API REST de cada módulo, registrados no roteador central (`routers.py`);
- **Permissões (`permissions.py`):** classes que controlam o acesso às operações de cada recurso, integradas ao sistema de permissões granulares do Taiga;
- **Serviços (`services.py`):** concentram as regras de negócio mais complexas, que não devem residir nas visões nem nos modelos;
- **Sinais e configuração (`signals.py`, `apps.py`):** conectam os novos módulos ao núcleo da aplicação de forma desacoplada, reagindo a eventos das entidades existentes (épicas, histórias e tarefas) sem alterar diretamente o seu código.

5.5.1.1 Modelo de dados

A incorporação das novas funcionalidades exigiu a criação de novas tabelas no banco de dados PostgreSQL, geradas e versionadas por meio do mecanismo de migrações do Django. Cada módulo introduziu um conjunto próprio de tabelas, todas carregando a referência ao projeto (`project_id`) para fins de escopo, indexação e autorização. O Quadro 8 sintetiza as tabelas criadas e suas respectivas responsabilidades.

Quadro 8 – Novas tabelas introduzidas no banco de dados

Módulo	Tabela	Responsabilidade
Cronograma (Figura 31)	<code>schedule_schedule</code>	Datas e cor de agendamento de cada atividade.
	<code>schedule_scheduledependency</code>	Dependências (predecessora/sucessora) entre atividades.
	<code>schedule_scheduleitemorder</code>	Ordem/posição das atividades no Gantt e na lista.
Financeiro (Figura 32)	<code>finances_activitycost</code>	Agregados financeiros por atividade (horas e custos, planejado x realizado).
	<code>finances_activitytimeentry</code>	Lançamentos de tempo (manual ou timer) e custos calculados.
	<code>finances_memberhourlyrate</code>	Custo/hora por membro, com vigência.
	<code>finances_projectincome</code>	Receitas do projeto (pontuais ou recorrentes).
	<code>finances_activitycostcategory</code>	Catálogo de categorias de custo do projeto.
	<code>finances_activitycostcategoryallocation</code>	Alocação do custo planejado por categoria.
Riscos (Figura 33)	<code>risks_risk</code>	Registro de riscos do projeto.
	<code>risks_riskevaluation</code>	Histórico de avaliações de probabilidade e impacto.

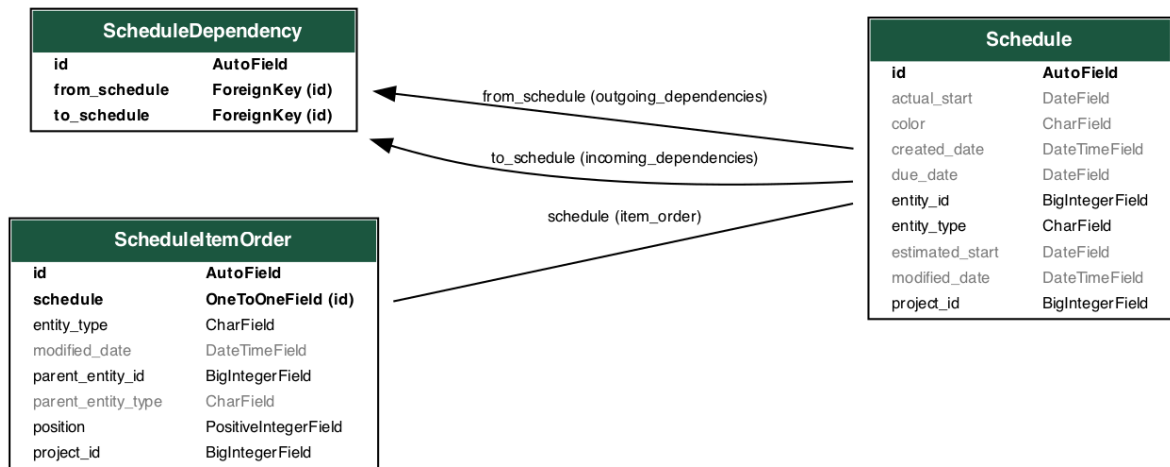
Fonte: Elaborado pelos autores (2026)

Nas figuras a seguir são apresentados os modelos de dados de cada módulo, gerados a partir dos modelos do Django. Para facilitar a leitura, os diagramas destacam apenas as tabelas próprias de cada funcionalidade e os relacionamentos entre elas; as entidades do núcleo do Taiga referenciadas por chave estrangeira (projeto, épico, história de usuário, tarefa, *membership* e *sprint*) são citadas no texto, mas omitidas dos diagramas.

5.5.2 Épico E1 - Gestão de Cronograma

O épico de gestão de cronograma reúne as funcionalidades implementadas para representar, acompanhar e analisar o planejamento temporal do projeto dentro do Taiga. As implementações desse épico buscaram integrar elementos prescritivos de gerenciamento de cronograma, como gráfico de Gantt, dependências entre atividades, caminho crítico

Figura 31 – Modelo de dados do módulo de gestão de cronograma



Fonte: Elaborado pelos autores (2026)

e folga, aos elementos já existentes na ferramenta, como épicos, histórias de usuário e tarefas.

Para concretizar os objetivos desse épico, foram criadas duas páginas principais no sistema: `/schedulelist` e `/ganttt`. A página `/schedulelist` foi concebida como uma lista de atividades do projeto, com foco na visualização e edição das datas relevantes para o planejamento, incluindo data de início prevista, data de início real e data de término. Essa página também incorporou filtros seguindo o padrão de outras listas existentes no Taiga, permitindo organizar e localizar atividades de forma consistente com os demais módulos da ferramenta.

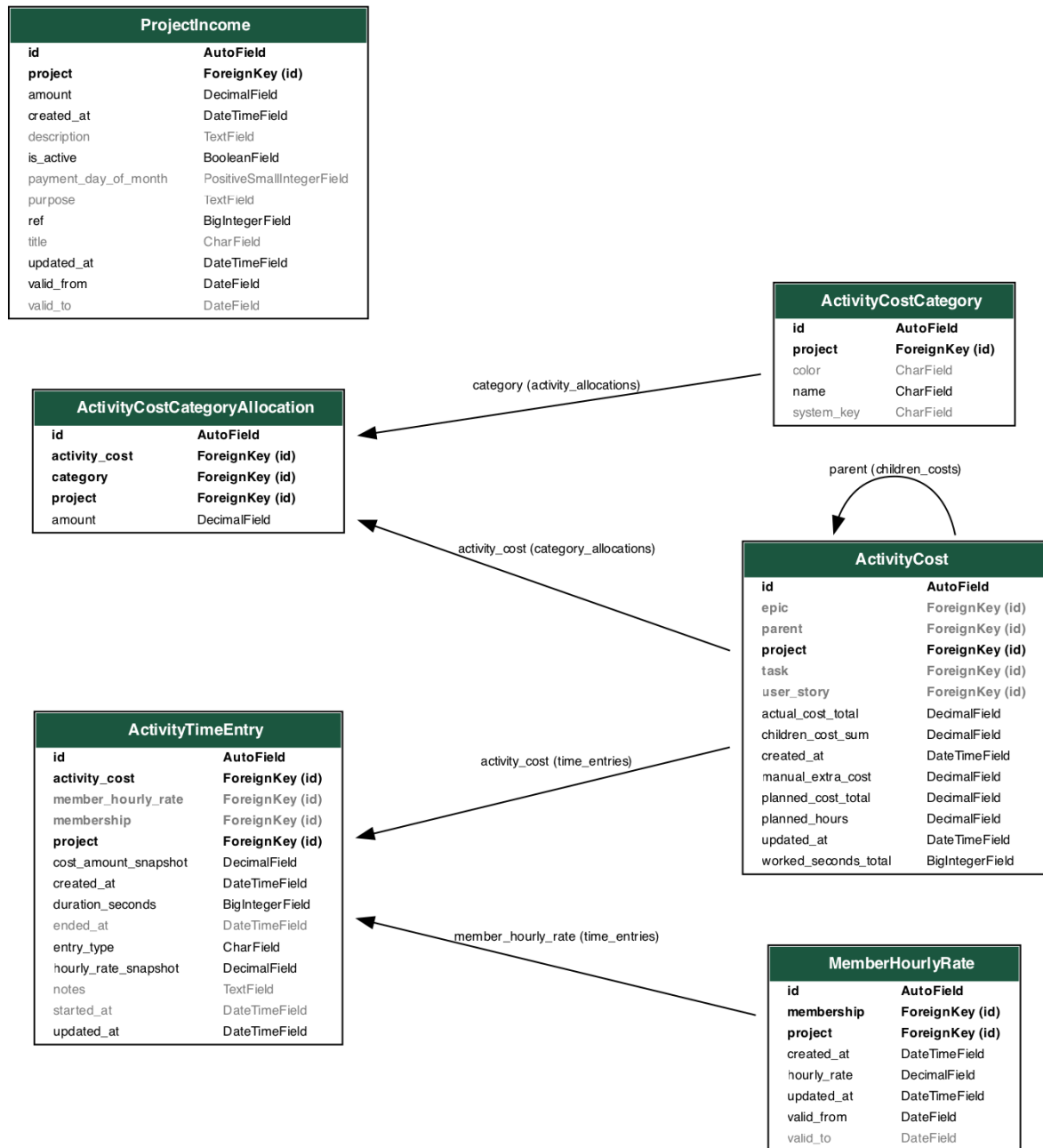
A página `/ganttt`, por sua vez, constitui a principal interface visual do épico de gestão de cronograma. Nela, as informações preenchidas na página `/schedulelist` são representadas graficamente em um diagrama de Gantt, permitindo a visualização temporal das atividades do projeto. Além da representação gráfica, essa página permite criar e alterar datas das atividades, definir ligações de dependência entre atividades, visualizar o caminho crítico, consultar a folga total de cada atividade e utilizar recursos de organização do diagrama.

CRON-01 – Visualização do cronograma em gráfico de Gantt

A história CRON-01 foi atendida por meio da criação da página `/ganttt`, responsável por apresentar o cronograma do projeto em formato de diagrama de Gantt. Conforme ilustrado na Figura 34, essa interface permite visualizar, em uma única tela, a estrutura hierárquica das atividades do projeto e sua representação temporal. Dessa forma, histórias de usuário, tarefas e épicos cadastrados no Taiga passam a ser exibidos de maneira integrada ao planejamento do cronograma.

A tela foi desenvolvida com diferentes recursos de navegação, visualização e edição.

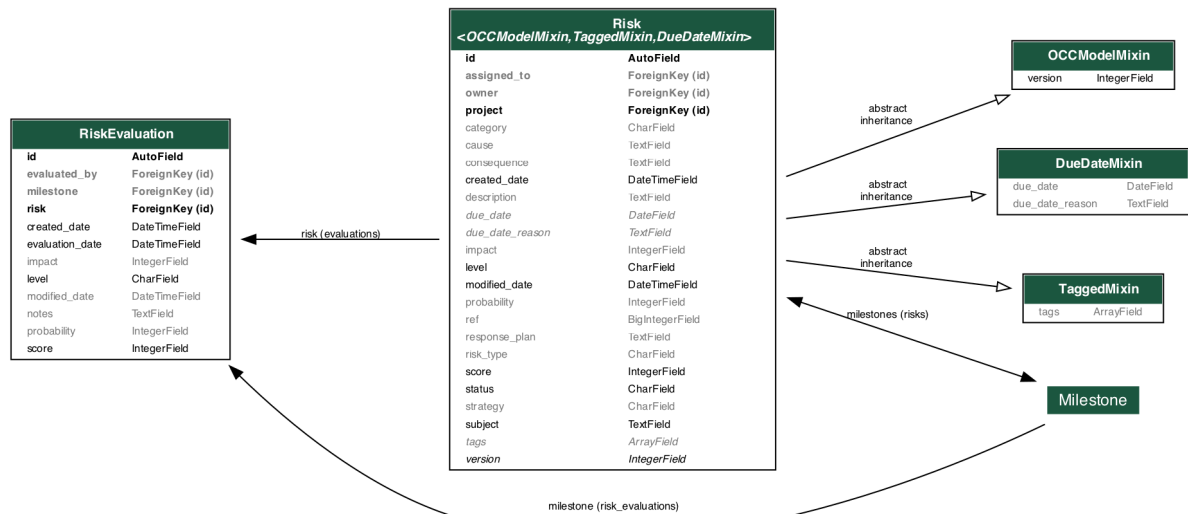
Figura 32 – Modelo de dados do módulo de gestão financeira



Fonte: Elaborado pelos autores (2026)

Na parte superior da interface, há um menu de botões com ferramentas de manipulação do diagrama, destacado na Figura 37. Abaixo desse menu, a tela é dividida em duas áreas principais. À esquerda, apresenta-se a lista de atividades organizada em estrutura de árvore, na qual os épicos são representados por ícone de montanha, as histórias de usuário por ícone de pergaminho e as tarefas por ícone de martelo. Essa lista contém colunas com informações relevantes de cada item, incluindo título da atividade, responsável, data de início, data de término e percentual de completude. Entre as duas áreas, há uma barra divisória ajustável, permitindo alterar a proporção ocupada pela lista e pelo diagrama.

Figura 33 – Modelo de dados do módulo de gestão de riscos



Fonte: Elaborado pelos autores (2026)

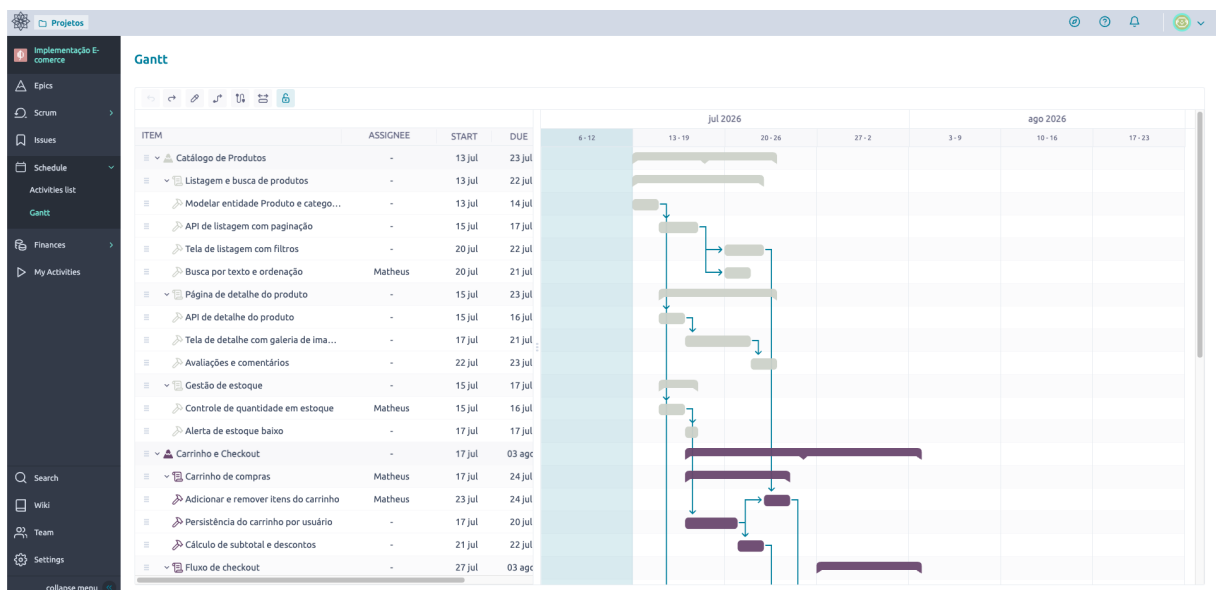


Figura 34 – Visualização da página do diagrama de Gantt

À direita, encontra-se o diagrama de Gantt propriamente dito, no qual as atividades são representadas por barras posicionadas de acordo com suas datas de início e término. A escala horizontal do diagrama pode ser ajustada para visualização por dias, semanas ou meses, e o período atual é destacado visualmente para facilitar a orientação temporal. As barras também possuem representações visuais distintas conforme o tipo de atividade, conforme apresentado na Figura 36, permitindo diferenciar épicos, histórias de usuário e tarefas durante a leitura do cronograma.

No diagrama de Gantt, as relações de dependência entre as atividades são representadas por linhas de conexão. Quando uma atividade atua como predecessora, a ligação

parte da extremidade direita de sua barra; quando atua como sucessora, a ligação chega à extremidade esquerda. Essas conexões evidenciam a ordem lógica de execução das atividades e constituem uma das bases para o cálculo do caminho crítico. Quando um item hierárquico está recolhido, as dependências de suas atividades filhas permanecem visíveis na barra do item pai, sendo posicionadas horizontalmente de acordo com a data correspondente ao término da atividade filha. Dessa forma, mesmo sem expandir a estrutura, é possível identificar em quais pontos do período representado pelo item pai existem dependências associadas às atividades subordinadas. Esse comportamento é exemplificado na Figura 36, na qual a atividade “Catálogo de Produtos”, embora esteja recolhida, apresenta três conexões vinculadas às suas atividades filhas em diferentes posições ao longo da barra.

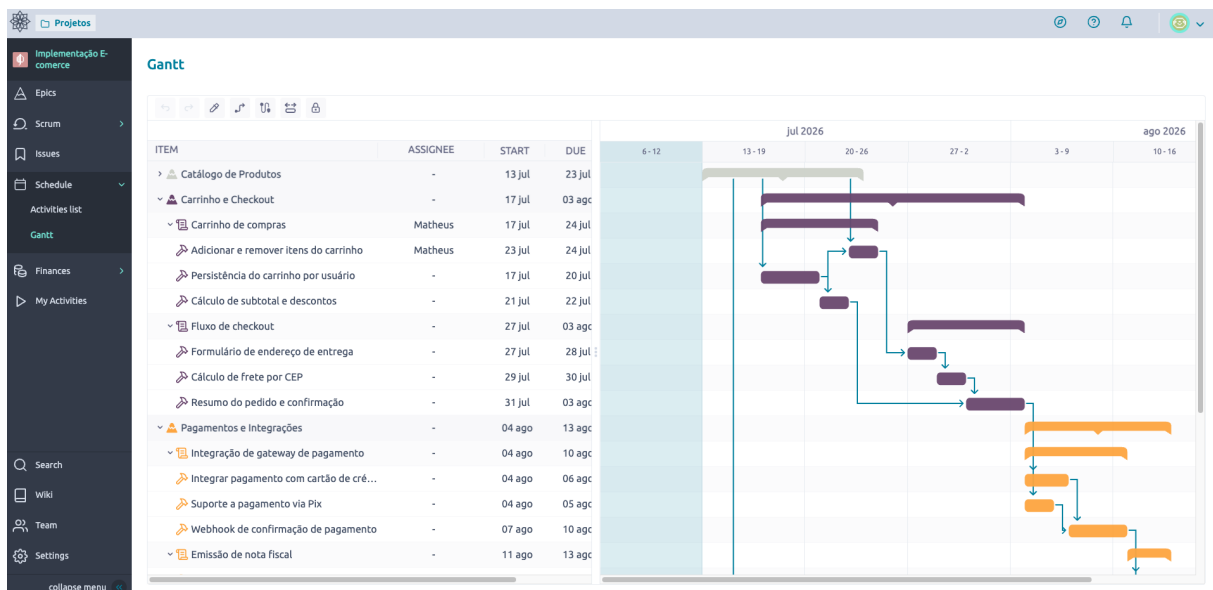


Figura 35 – Visualização da página do diagrama de Gantt com atividade “Catálogo de Produtos” recolhida

O menu de ferramentas, apresentado na Figura 37, concentra as principais ações disponíveis na página:

- **Desfazer e refazer:** permitem reverter ou reaplicar a última alteração realizada no diagrama.
- **Coloração:** permite selecionar uma atividade na lista e definir a cor de sua barra correspondente no gráfico de Gantt.
- **Vínculo entre atividades:** permite criar dependências entre atividades a partir da seleção de duas barras no diagrama. Essa funcionalidade é utilizada como base para a representação de relacionamentos no cronograma e para o cálculo do caminho crítico.
- **Caminho crítico:** ativa a visualização das atividades que compõem o caminho crítico do projeto.



Figura 36 – Representação das barras no diagrama de Gantt: acima, barra de época; ao centro, barra de história de usuário; abaixo, barra de tarefa.

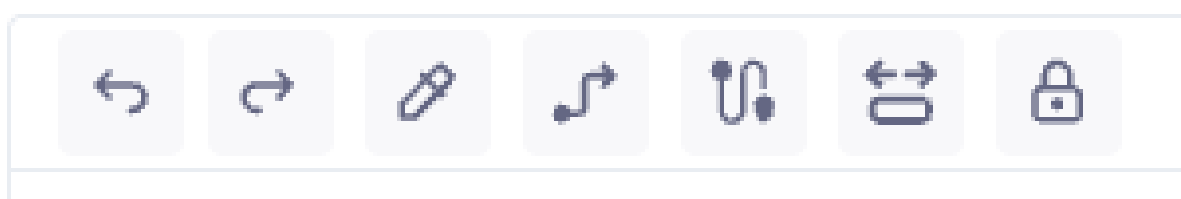


Figura 37 – Menu de opções e edição da página Gantt

- **Ajuste de escala:** permite alternar a escala horizontal do diagrama entre dias, semanas e meses.
- **Modo de edição:** controlado pelo botão de cadeado, permite habilitar ou bloquear alterações diretas no diagrama. Quando desbloqueado, torna-se possível arrastar barras para alterar datas, redimensionar suas extremidades para ajustar início ou término e reorganizar verticalmente as atividades por meio da lista lateral.

CRON-02 e CRON-03 – Caminho crítico e folga total

As histórias CRON-02 e CRON-03 foram atendidas por meio da implementação da visualização do caminho crítico e da exibição da folga total das atividades no gráfico de Gantt. Ao acionar o botão de caminho crítico, mencionado na seção anterior, o sistema destaca as atividades e ligações pertencentes ao caminho crítico, mantendo-as com maior evidência visual. As demais atividades e dependências permanecem visíveis, porém com

opacidade reduzida, conforme apresentado na Figura 38. Essa abordagem permite identificar rapidamente quais atividades possuem maior impacto sobre a duração total do projeto sem remover o restante do contexto do cronograma.

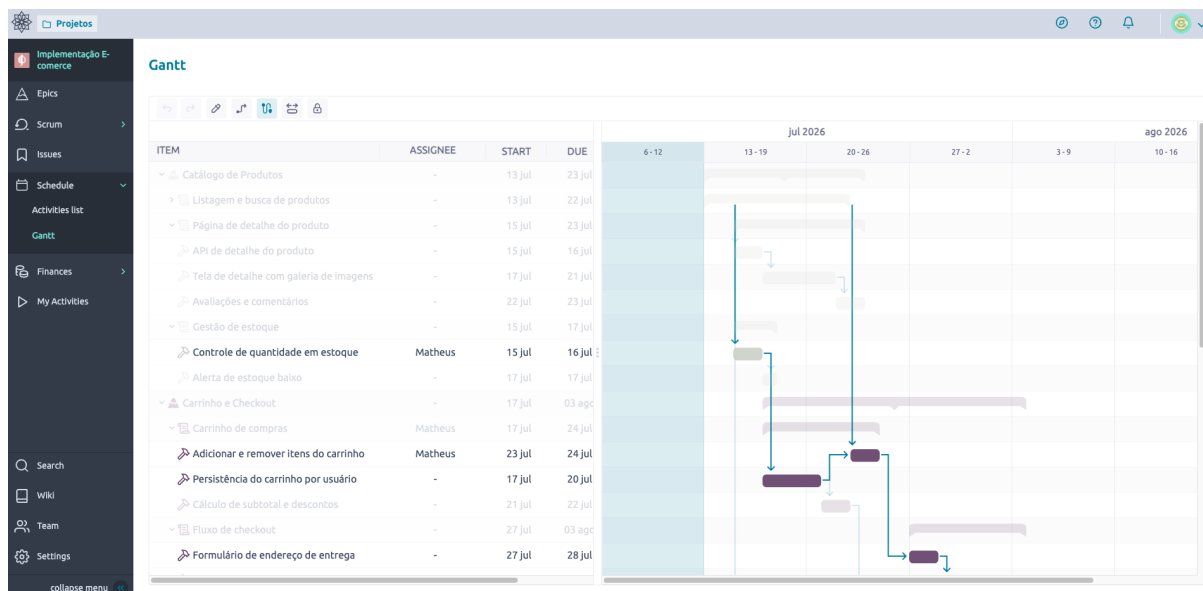


Figura 38 – Demonstração de caminho crítico no diagrama de Gantt

O cálculo do caminho crítico é realizado no front-end, a partir das informações de datas e dependências já carregadas na página do Gantt. A implementação considera as datas de início e término das atividades para calcular suas durações, constrói um grafo direcionado com base nas dependências cadastradas e utiliza esse grafo para calcular os caminhos relevantes do cronograma.

Para a implementação desse requisito, foram consideradas algumas particularidades do uso do Gantt em projetos reais. Primeiramente, o caminho crítico não foi definido apenas como o conjunto de atividades com folga igual a zero, mas como o conjunto de atividades com a menor folga calculada no cronograma. Essa decisão permite lidar com situações em que existem intervalos entre atividades, datas já estabelecidas ou períodos de espera que não aparecem necessariamente como uma dependência explícita no diagrama.

Além disso, o algoritmo considera como candidatos ao caminho crítico apenas os caminhos que alcançam a atividade, ou conjunto de atividades, com a última data de término do projeto. Dessa forma, uma sequência de atividades que não contribui para a conclusão final do projeto não é classificada como caminho crítico, ainda que apresente folga reduzida em relação ao seu próprio encadeamento. Também são consideradas apenas atividades sem filhos, pois atividades hierárquicas, como épicas ou histórias que agrupam tarefas, representam decomposições do trabalho. Nesses casos, as atividades de menor nível são utilizadas no cálculo por representarem unidades mais específicas do cronograma.

A folga total é calculada para as atividades consideradas no grafo do caminho crítico. Para atividades que não alcançam a última atividade do projeto, o sistema utiliza

como referência a data final do projeto para preencher uma folga baseada na diferença entre a data de término da atividade e a data de término do projeto. Com isso, mesmo atividades que não pertencem aos caminhos candidatos podem apresentar uma indicação de folga ao usuário.

No modo de visualização do caminho crítico, ao posicionar o cursor sobre uma atividade, o sistema exibe um painel informativo com a folga total calculada, conforme apresentado na Figura 39. Essa informação permite avaliar quanto uma atividade pode atrasar sem impactar a data final considerada para o projeto, contribuindo para a análise do cronograma e para a priorização de ações de acompanhamento.

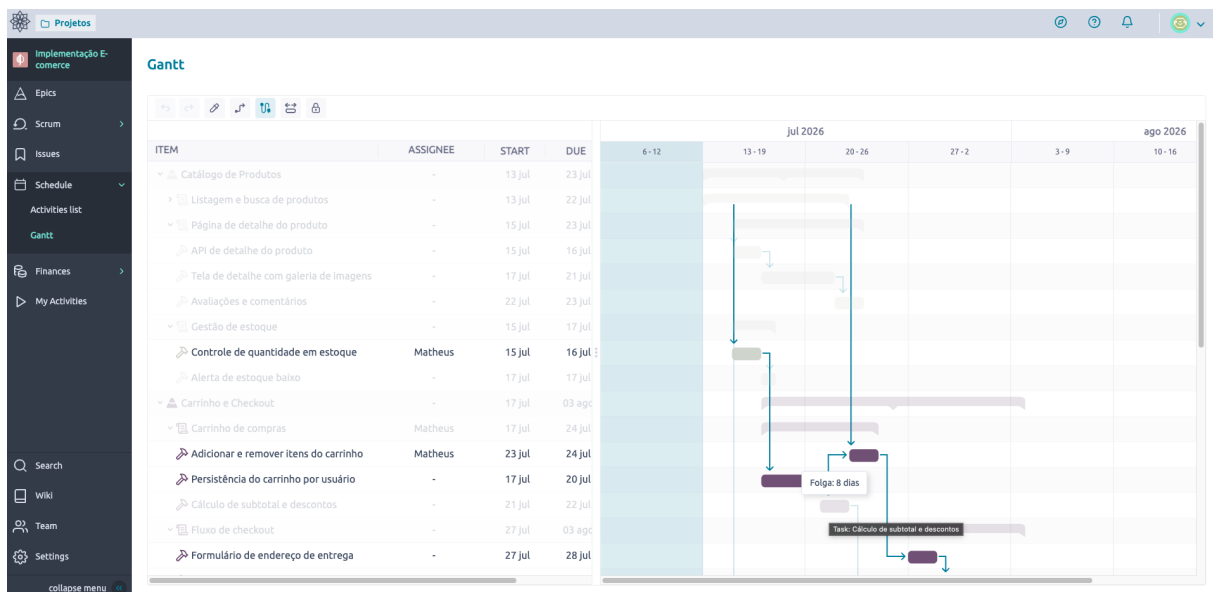


Figura 39 – Representação visual da folga total de uma atividade

5.5.3 Épico E2 – Gestão Financeira

O épico de gestão financeira reúne as funcionalidades implementadas para registrar, acompanhar e consolidar informações de custos e receitas no Taiga. As implementações desse épico buscaram complementar os elementos já existentes da ferramenta, como épicos, histórias de usuário, tarefas e membros do projeto, com informações financeiras associadas ao planejamento e à execução das atividades.

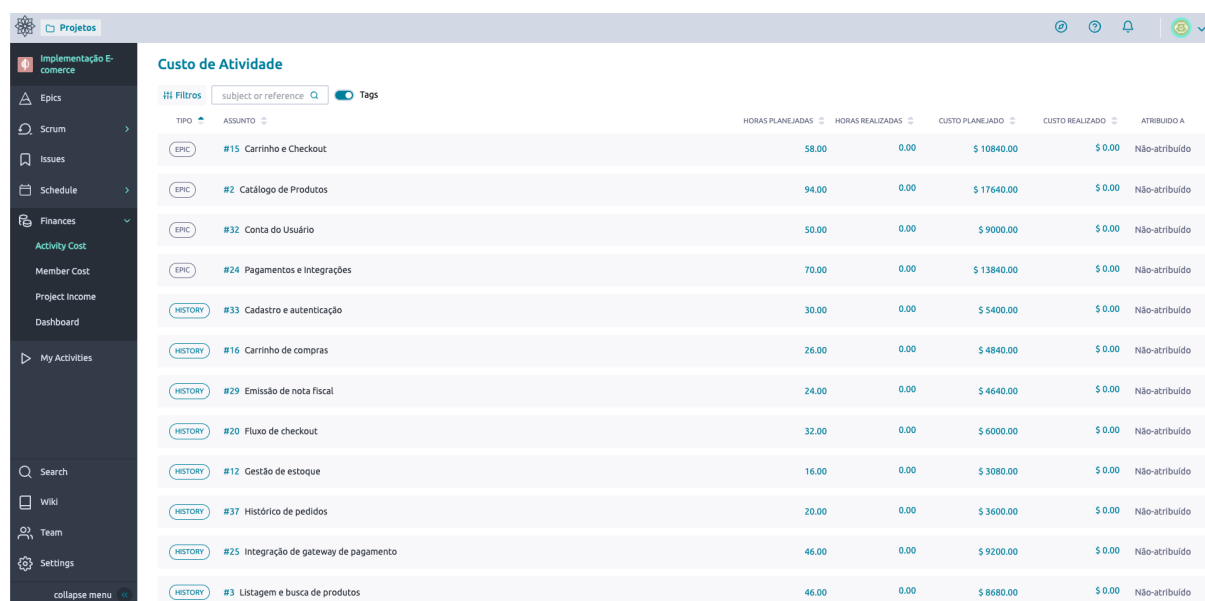
Para concretizar os objetivos desse épico, foram criadas páginas voltadas ao acompanhamento financeiro do projeto. A página `/activity-cost` apresenta os custos associados às atividades do projeto, incluindo horas planejadas, horas realizadas, custo planejado e custo realizado. A página `/member-cost` permite configurar o valor/hora dos integrantes do projeto, informação utilizada no cálculo dos custos realizados. A página `/my-activities` permite que integrantes registrem o tempo trabalhado nas atividades por meio de um temporizador. Além disso, a página `/project-income` permite cadastrar receitas do projeto,

enquanto a página `/project-dashboard` consolida indicadores financeiros em uma visão geral.

CUST-01 – Custo planejado da tarefa

A história CUST-01 foi atendida por meio da criação da estrutura de custos por atividade. No sistema implementado, épicos, histórias de usuário e tarefas passam a possuir informações financeiras associadas, permitindo registrar horas planejadas, custo planejado próprio, custo planejado de atividades subordinadas e custo planejado total. Dessa forma, o planejamento financeiro passa a acompanhar a hierarquia já utilizada pelo Taiga para organizar o trabalho do projeto.

Na interface, essa funcionalidade é apresentada inicialmente na página `/activity-cost`, conforme ilustrado na Figura 40. Nessa tela, as atividades do projeto são listadas com colunas específicas para horas planejadas, horas realizadas, custo planejado, custo realizado e responsável. A página também mantém recursos de busca, filtros, ordenação e exibição de tags, seguindo o padrão de outras listas existentes no Taiga.



The screenshot shows the 'Custo de Atividade' (Activity Cost) page in the Taiga project management tool. The page has a sidebar on the left with navigation options like 'Epics', 'Scrum', 'Issues', 'Schedule', 'Finances', 'Activity Cost', 'Member Cost', 'Project Income', 'Dashboard', and 'My Activities'. The main content area displays a table of activities with the following columns: TIPO, ASSUNTO, HORAS PLANEJADAS, HORAS REALIZADAS, CUSTO PLANEJADO, CUSTO REALIZADO, and ATRIBUÍDO A. The table lists 12 activities, including 'Carrinho e Checkout', 'Catálogo de Produtos', 'Conta do Usuário', 'Pagamentos e Integrações', 'Cadastro e autenticação', 'Carrinho de compras', 'Emissão de nota fiscal', 'Fluxo de checkout', 'Gestão de estoque', 'Histórico de pedidos', 'Integração de gateway de pagamento', and 'Listagem e busca de produtos'. Each activity row shows its type (EPIC or HISTORY), ID, title, and the corresponding financial values for hours and costs, along with the assigned user (all listed as 'Não-atribuído').

TIPO	ASSUNTO	HORAS PLANEJADAS	HORAS REALIZADAS	CUSTO PLANEJADO	CUSTO REALIZADO	ATRIBUÍDO A
EPIC	#15 Carrinho e Checkout	58.00	0.00	\$ 10840.00	\$ 0.00	Não-atribuído
EPIC	#2 Catálogo de Produtos	94.00	0.00	\$ 17640.00	\$ 0.00	Não-atribuído
EPIC	#32 Conta do Usuário	50.00	0.00	\$ 9000.00	\$ 0.00	Não-atribuído
EPIC	#24 Pagamentos e Integrações	70.00	0.00	\$ 13840.00	\$ 0.00	Não-atribuído
HISTORY	#33 Cadastro e autenticação	30.00	0.00	\$ 5400.00	\$ 0.00	Não-atribuído
HISTORY	#16 Carrinho de compras	26.00	0.00	\$ 4840.00	\$ 0.00	Não-atribuído
HISTORY	#29 Emissão de nota fiscal	24.00	0.00	\$ 4640.00	\$ 0.00	Não-atribuído
HISTORY	#20 Fluxo de checkout	32.00	0.00	\$ 6000.00	\$ 0.00	Não-atribuído
HISTORY	#12 Gestão de estoque	16.00	0.00	\$ 3080.00	\$ 0.00	Não-atribuído
HISTORY	#37 Histórico de pedidos	20.00	0.00	\$ 3600.00	\$ 0.00	Não-atribuído
HISTORY	#25 Integração de gateway de pagamento	46.00	0.00	\$ 9200.00	\$ 0.00	Não-atribuído
HISTORY	#3 Listagem e busca de produtos	46.00	0.00	\$ 8680.00	\$ 0.00	Não-atribuído

Figura 40 – Página com a lista de atividades com atributos financeiros

Na listagem de custos por atividade, é possível editar diretamente os valores de horas planejadas e custo planejado, desde que o usuário possua as permissões financeiras necessárias. As horas planejadas representam a estimativa de esforço prevista para a atividade, enquanto o custo planejado representa o valor financeiro esperado para sua execução. Esse valor pode considerar tanto despesas relacionadas à mão de obra quanto outros custos previstos, como aquisição de equipamentos, licenças de software, materiais ou serviços. Quando uma atividade possui atividades subordinadas, o sistema considera os

valores planejados dos filhos na composição dos totais, permitindo que histórias e épicos consolidem os custos das tarefas relacionadas.

Embora a página também apresente horas realizadas e custo realizado, esses valores não são editados diretamente nesse contexto. Eles são calculados a partir dos registros reais de trabalho feitos pelos integrantes do projeto, conforme detalhado na história CUST-02. Assim, a listagem diferencia os campos de planejamento, que podem ser ajustados manualmente, dos campos de execução, que são derivados dos apontamentos de tempo e dos valores/hora configurados para os integrantes. Na implementação realizada, o custo realizado contempla exclusivamente despesas associadas à mão de obra, não sendo possível registrar diretamente custos realizados referentes a equipamentos, licenças de software, materiais ou serviços externos.

Além da listagem, foi criada uma página de detalhamento financeiro da atividade, apresentada na Figura 41. Essa página permite consultar informações financeiras específicas de uma atividade, incluindo a distribuição do custo planejado por categorias e o histórico de entradas de trabalho registradas. A presença do histórico nessa tela permite relacionar os valores planejados com os registros de execução que compõem as horas realizadas e o custo real da atividade, os quais são aprofundados na história CUST-02.

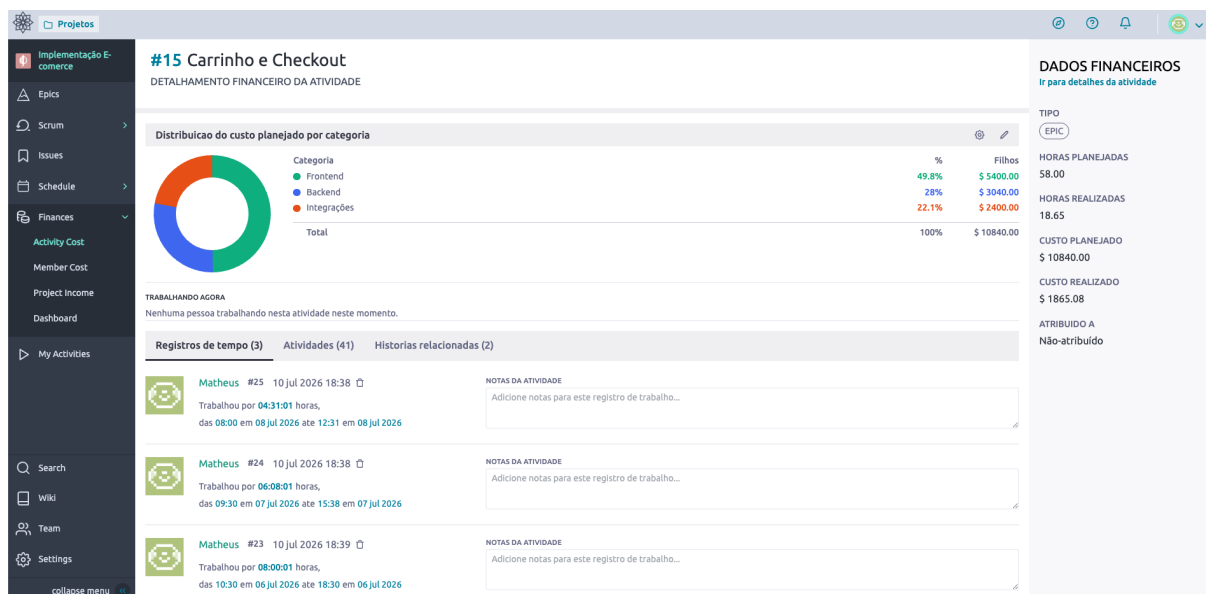


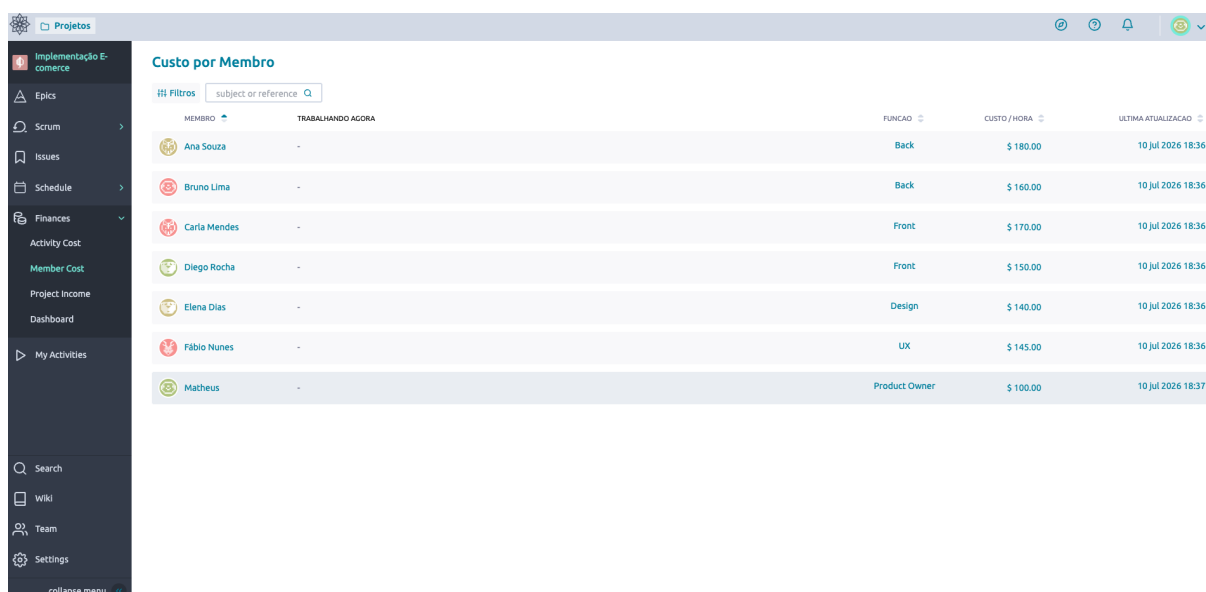
Figura 41 – Página com o detalhamento financeiro de uma atividade

Na página de detalhe, também é possível distribuir o custo planejado por categorias. Essa distribuição permite classificar o custo de uma atividade em grupos definidos no projeto, como desenvolvimento, documentação, gestão ou outras categorias relevantes. Quando uma atividade possui filhos, o sistema também apresenta a composição entre custo próprio e custo das atividades subordinadas, permitindo compreender como o custo planejado total é formado.

CUST-02 – Horas trabalhadas e custo realizado

A história CUST-02 foi atendida por meio da criação de funcionalidades para registrar o tempo trabalhado pelos integrantes do projeto e calcular automaticamente o custo realizado com base no valor/hora configurado para cada membro. Essa implementação complementa a estrutura de custos por atividade descrita na história CUST-01, pois os registros de trabalho alimentam os valores de horas realizadas e custo realizado exibidos na listagem e no detalhamento financeiro da atividade, conforme apresentado na Figura 41.

Para que o custo realizado possa ser calculado, foi criada a página `/member-cost`, apresentada na Figura 42, responsável por configurar o valor/hora dos integrantes do projeto. Nessa tela, os membros são apresentados em uma lista com informações como nome, papel no projeto, atividade em execução no momento, valor/hora configurado e data da última atualização. O valor/hora de cada integrante pode ser editado diretamente na listagem, desde que o usuário possua as permissões necessárias. Esse valor é utilizado para calcular o custo realizado a partir dos registros de horas trabalhadas. No momento em que um registro de trabalho é criado, o valor/hora vigente do integrante é armazenado junto ao próprio registro, preservando o valor utilizado no cálculo. Dessa forma, alterações posteriores no valor/hora não recalculam retroativamente os custos já registrados, sendo aplicadas somente aos novos registros de trabalho realizados após a alteração.

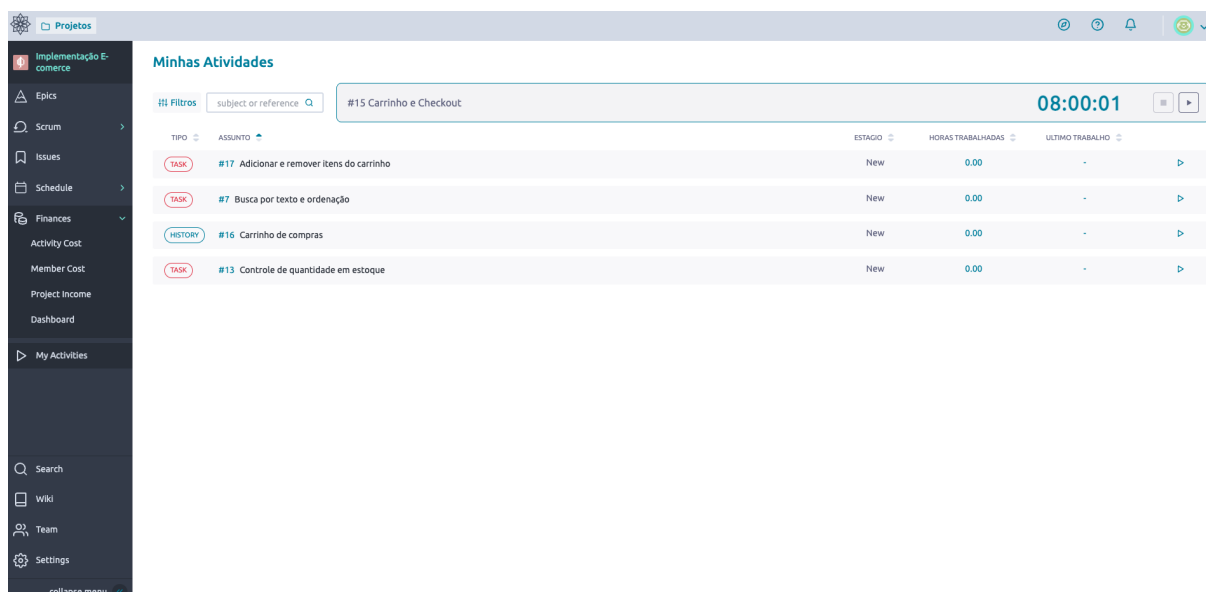


MEMBRO	TRABALHANDO AGORA	FUNCAO	CUSTO / HORA	ULTIMA ATUALIZACAO
Ana Souza	-	Back	\$ 180.00	10 jul 2026 18:36
Bruno Lima	-	Back	\$ 160.00	10 jul 2026 18:36
Carla Mendes	-	Front	\$ 170.00	10 jul 2026 18:36
Diego Rocha	-	Front	\$ 150.00	10 jul 2026 18:36
Elena Dias	-	Design	\$ 140.00	10 jul 2026 18:36
Fábio Nunes	-	LUX	\$ 145.00	10 jul 2026 18:36
Matheus	-	Product Owner	\$ 100.00	10 jul 2026 18:37

Figura 42 – Página com detalhamento valor/hora de membro do projeto

O registro das horas trabalhadas ocorre na página `/my-activities`, conforme a Figura 43. Nessa tela, o usuário visualiza as atividades associadas a ele e pode iniciar ou interromper um temporizador de trabalho. Ao iniciar uma atividade, o sistema registra o início do apontamento associado ao integrante e à atividade correspondente. Ao encerrar o temporizador, o sistema calcula a duração trabalhada e registra o custo realizado com base no valor/hora vigente no momento do apontamento. Conforme discutido na história

CUST-01, o custo realizado contemplado nesta implementação corresponde exclusivamente aos custos de mão de obra derivados dos registros de tempo.



TIPO	ASSUNTO	ESTAGIO	HORAS TRABALHADAS	ULTIMO TRABALHO
TASK	#17 Adicionar e remover itens do carrinho	New	0.00	-
TASK	#7 Busca por texto e ordenação	New	0.00	-
HISTORY	#16 Carrinho de compras	New	0.00	-
TASK	#13 Controle de quantidade em estoque	New	0.00	-

Figura 43 – Página com as atividades designadas a um membro

Essa abordagem permite que as horas realizadas e o custo realizado sejam derivados de entradas reais de trabalho, e não de edição manual direta. Com isso, os valores apresentados em `/activity-cost` e no detalhamento financeiro da atividade refletem os apontamentos registrados pelos integrantes do projeto, contribuindo para maior rastreabilidade entre trabalho executado, horas consumidas e custo financeiro realizado.

CUST-03 – Receita contratada do projeto

A história CUST-03 foi atendida por meio da criação da área de receitas do projeto, acessível pela página `/project-income`, conforme ilustrado na Figura 44. Em vez de limitar o projeto a um único valor de receita, a implementação permite cadastrar múltiplas receitas, cada uma com título, descrição, valor, finalidade, data inicial de validade, data final opcional, status ativo/inativo e, quando aplicável, dia de pagamento mensal.

Essa abordagem permite representar tanto receitas pontuais quanto receitas recorrentes. Uma receita sem dia de pagamento mensal pode representar um valor recebido em uma data específica, enquanto uma receita com dia de pagamento pode representar entradas recorrentes ao longo do período de validade informado. A página de listagem permite visualizar e ordenar as receitas cadastradas, e a página de detalhe permite consultar e editar os dados de cada receita conforme as permissões do usuário.

A página de listagem permite visualizar e ordenar as receitas cadastradas, e a página de detalhe, apresentada na Figura 45, permite consultar e editar os dados de cada receita conforme as permissões do usuário.

TÍTULO DA RECEITA	VALOR	DIA DO PAGAMENTO	VALIDO DE	VALIDO ATE	ULTIMA ATUALIZACAO
#4 Contrato de manutenção mensal	\$ 3500.00	10	17 ago 2026	-	10 jul 2026 18:51
#3 Entrega final (30%)	\$ 18000.00	pagamento único	14 ago 2026	-	10 jul 2026 18:51
#2 Parcela intermediária (30%)	\$ 18000.00	pagamento único	31 jul 2026	-	10 jul 2026 18:51
#1 Sinal do contrato (40%)	\$ 24000.00	pagamento único	13 jul 2026	-	10 jul 2026 18:51

Figura 44 – Página com as receitas do projeto

RECEITA
#4 Contrato de manutenção mensal

DESCRICAO
Receita recorrente mensal de manutenção, vencimento todo dia 10.

FINALIDADE
Sustentação, correções e pequenas melhorias pós-lançamento.

DADOS DA RECEITA
VALOR
\$ 3500.00
DIA DO PAGAMENTO
10
VALIDO DE
17 ago 2026
VALIDO ATE
-
ULTIMA ATUALIZACAO
10 jul 2026 18:51

DEFINIR COMO INATIVA

Figura 45 – Página com o detalhamento de uma receita

As receitas ativas são utilizadas posteriormente na consolidação dos indicadores financeiros exibidos no dashboard, permitindo comparar entradas previstas ou recebidas com os custos realizados no projeto.

CUST-04 – Resumo financeiro do projeto

A história CUST-04 foi atendida por meio da criação da página `/project-dashboard`, responsável por consolidar as informações financeiras do projeto em uma visão de resumo. Por concentrar indicadores e diferentes gráficos em uma única página, sua visualização completa exige rolagem vertical; por isso, a interface é apresentada em duas partes nas Figuras 46 e 47. Essa página reúne dados provenientes dos custos planejados das atividades, dos apontamentos de horas trabalhadas, das taxas horárias dos membros e das receitas cadastradas no projeto.

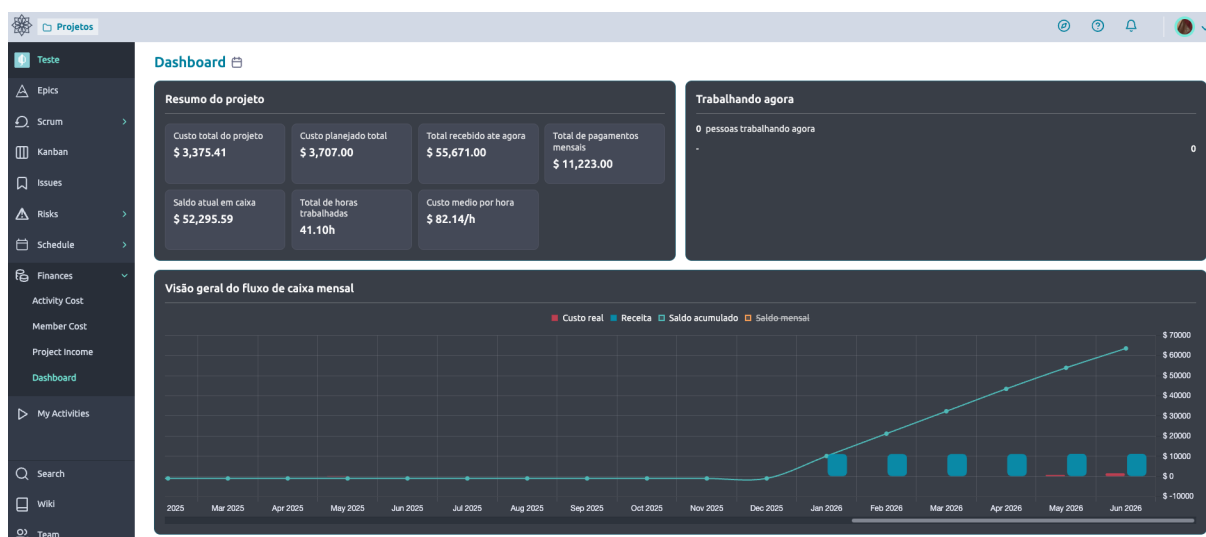


Figura 46 – Página com um resumo financeiro do projeto, primeira parte

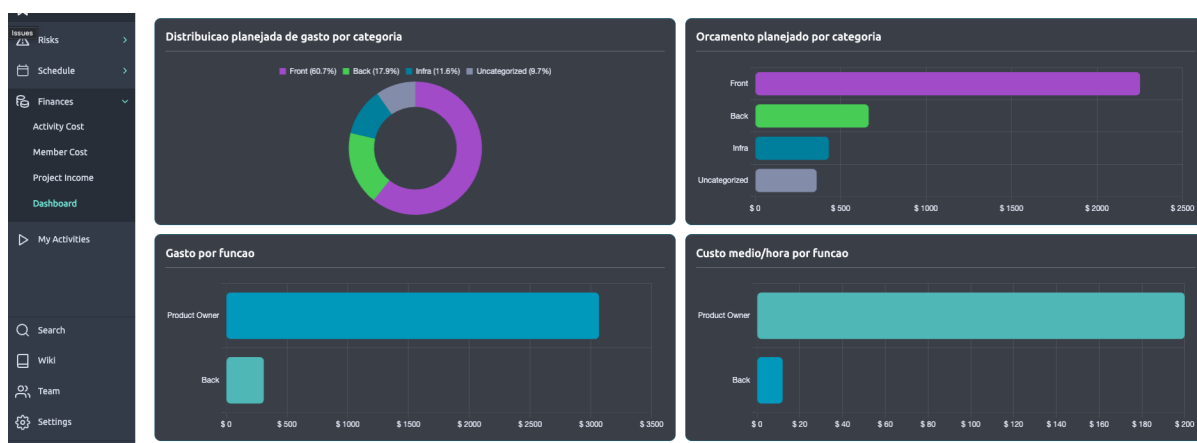


Figura 47 – Página com um resumo financeiro do projeto, segunda parte

O resumo financeiro apresenta indicadores como custo total realizado, custo planejado total, receita recebida até o momento, total mensal de pagamentos previstos, saldo financeiro atual, total de horas trabalhadas e custo médio por hora. Além dos indicadores numéricos, a página também apresenta gráficos para apoiar a análise financeira, incluindo custo real em comparação com receitas ao longo do tempo, custos por categoria, custos por papel dos membros e custo médio por hora por papel.

Dessa forma, a página de resumo financeiro permite acompanhar a situação econômica do projeto sem depender de cálculos externos, reunindo em uma única visualização os principais dados necessários para comparar planejamento, execução e receitas.

5.5.4 Épico E3 – Gestão de Riscos

O épico de gestão e evolução de riscos consolida as funcionalidades implementadas para cadastrar, analisar e monitorar ameaças e oportunidades ao longo da execução dos projetos no Taiga. As implementações deste épico introduziram uma arquitetura completa de gerenciamento de incertezas, permitindo que os riscos sejam não apenas registrados, mas também classificados quantitativamente e acompanhados graficamente ao longo do seu ciclo de vida.

Para apoiar a análise dos riscos, o sistema adota escalas padronizadas de probabilidade e impacto. Ambas utilizam cinco níveis de classificação, permitindo avaliar de forma consistente a chance de ocorrência de um evento e seu potencial efeito sobre os objetivos do projeto. A escala de probabilidade utilizada é apresentada no Quadro 9.

Quadro 9 – Escala de probabilidade utilizada no módulo de riscos

Nível	Probabilidade estimada
1 – Muito Baixa	Até 10%
2 – Baixa	11% a 30%
3 – Média	31% a 50%
4 – Alta	51% a 70%
5 – Muito Alta	Acima de 70%

Fonte: Elaborado pelos autores (2026).

A criticidade de cada risco é calculada pela multiplicação entre os valores de probabilidade e impacto ($P \times I$), gerando uma pontuação que varia de 1 a 25. Essa pontuação é utilizada pelo sistema para classificar automaticamente os riscos em diferentes níveis de severidade, conforme apresentado no Quadro 10.

Quadro 10 – Classificação da criticidade dos riscos

Pontuação ($P \times I$)	Nível de criticidade
1 a 4	Baixo
5 a 9	Médio
10 a 14	Alto
15 a 25	Crítico

Fonte: Elaborado pelos autores (2026).

Além dessas classificações, o sistema permite categorizar os riscos em diferentes áreas de origem, incluindo riscos técnicos, gerenciais, comerciais e externos. Também é possível classificá-los como ameaças ou oportunidades, definir estratégias de resposta apropriadas e acompanhar sua evolução por meio de estados que representam as etapas do processo de gerenciamento de riscos.

RISK-01 – Gerenciar registro de riscos

A história RISK-01 foi implementada por meio da criação da tela principal de gerenciamento de riscos, acessível pela rota `/risks`, e da respectiva tela de detalhes de cada registro. A página de listagem reúne todos os riscos cadastrados no projeto em uma única visualização, permitindo consultar rapidamente suas principais informações e acompanhar o estado atual de cada item.

A interface conta com uma tabela que apresenta colunas relacionadas à probabilidade, impacto, assunto, status, pontuação (*score*), nível de criticidade, categoria, estratégia de resposta, data da última modificação e responsável atribuído. Os registros podem ser ordenados por diferentes critérios por meio dos cabeçalhos das colunas, permitindo reorganizar a listagem de acordo com a informação desejada. Além disso, a tela disponibiliza recursos de busca textual e filtros que auxiliam na localização de riscos específicos e na segmentação das informações exibidas.

Conforme apresentado na Figura 48, a tela de listagem centraliza as principais informações dos riscos cadastrados no projeto. Além dos recursos de busca e filtragem, a interface permite ordenar os registros por diferentes critérios e realizar alterações rápidas em campos como status e responsável diretamente na tabela.

Riscos

Filtros subject or reference Tags + NOVO RISCO

PROBABILIDADE	IMPACTO	ASSUNTO	STATUS	PONTUAÇÃO	NÍVEL	CATEGORIA	ESTRATÉGIA	MODIFIED	ATRIBUÍDO A
5	5	#50 Falta de espaço em disco no servidor	Identificado	25	CRÍTICO	Gerencial	-	11 jun 2026	
4	5	#58 Débito técnico excessivo	Analisando	20	CRÍTICO	Comercial	-	04 jun 2026	
4	3	#52 Lentidão extrema na rede interna	Monitorando	12	ALTO	Técnico	-	28 mar 2026	
2	5	#57 Atraso na entrega da feature X	Analisando	10	ALTO	Externo	-	24 mai 2026	
3	3	#55 Vazamento acidental de credenciais	Analisando	9	MÉDIO	Comercial	Mitigar	11 jun 2026	
3	3	#53 Vulnerabilidade crítica em biblioteca	Analisando	9	MÉDIO	Comercial	-	19 mai 2026	
2	4	#51 Falha na rotina de backup	Analisando	8	MÉDIO	Comercial	-	07 jun 2026	
3	2	#62 Massa de dados insuficiente para testes	Monitorando	6	MÉDIO	Gerencial	-	03 jun 2026	
1	5	#66 Alta rotatividade na equipe	Analisando	5	MÉDIO	Comercial	-	30 mai 2026	
5	1	#65 Falta de definição clara nos requisitos	Analisando	5	MÉDIO	Comercial	-	25 mai 2026	
1	5	#49 Pico de tráfego derrubando load balancer	Analisando	5	MÉDIO	Comercial	-	02 jun 2026	
1	4	#63 Mudança repentina de escopo	Analisando	4	BAIXO	Comercial	-	02 jun 2026	

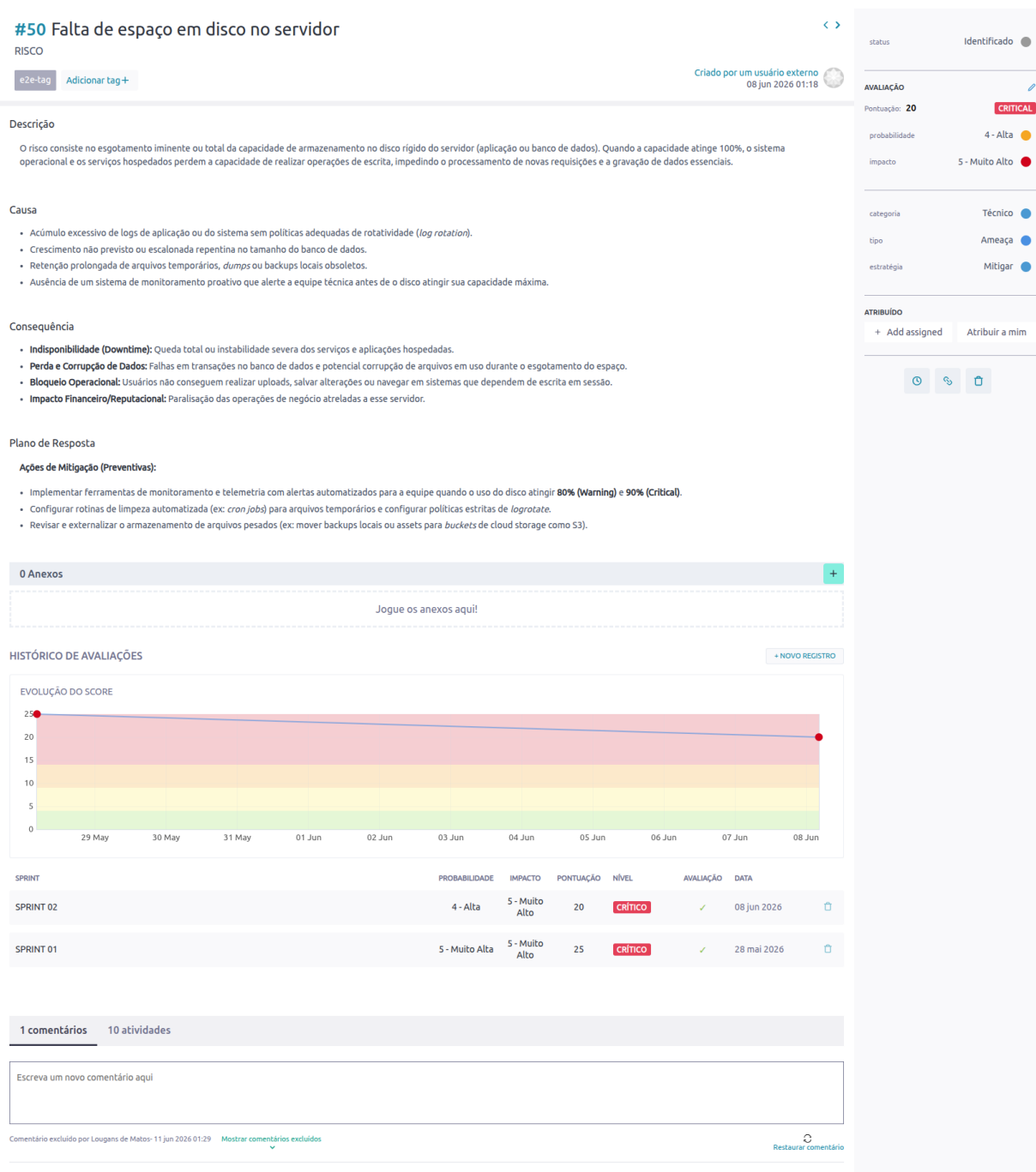
Figura 48 – Tela de listagem de riscos

Com o objetivo de agilizar atividades rotineiras, determinadas ações podem ser realizadas diretamente na listagem, sem a necessidade de acessar a tela de detalhes. Entre elas estão a alteração do status do risco, a definição ou troca do responsável e a criação de novos registros. Esses recursos reduzem a quantidade de navegações necessárias durante o acompanhamento do projeto e tornam o gerenciamento dos riscos mais eficiente.

O gerenciamento detalhado dos riscos é realizado na tela de detalhes do registro. Nela, é possível visualizar e editar informações como título, descrição, probabilidade, impacto, causa, consequência, responsável, estratégia de mitigação e plano de resposta. A interface também disponibiliza recursos para anexar documentos e complementar as análises por meio de campos de texto mais extensos e comentários, permitindo registrar informações relevantes ao longo do ciclo de vida do risco.

A Figura 49 apresenta a tela de detalhes de um risco. Nessa interface estão reunidas as informações utilizadas durante sua análise e tratamento, incluindo descrição, causas, consequências, plano de resposta, anexos, comentários e histórico de alterações. A mesma tela também reúne os mecanismos de acompanhamento da evolução do risco ao longo do projeto.

À medida que os valores de probabilidade e impacto são alterados, o sistema recalcula automaticamente a pontuação e atualiza a classificação do risco. A tela de detalhes também permite relacionar riscos às sprints do projeto. Com isso, um mesmo risco pode ser acompanhado em diferentes ciclos de trabalho, enquanto uma sprint pode conter vários riscos associados. Essa funcionalidade auxilia o planejamento das ações de mitigação e facilita o acompanhamento dos riscos ao longo das iterações do projeto, permitindo identificar em quais períodos cada atividade de tratamento será conduzida.



RISK-02 – Visualizar matriz probabilidade $\times$ impacto

A história RISK-02 foi implementada por meio da página `/risks/matrix`, responsável por fornecer uma visão consolidada da distribuição dos riscos do projeto de acordo com seus níveis de probabilidade e impacto. Conforme ilustrado na Figura 50, a interface é dividida em duas áreas principais. À esquerda encontra-se a matriz de riscos propriamente dita, enquanto à direita é apresentada uma listagem com os riscos cadastrados e suas respectivas pontuações.

Figura 50 – Matriz de Probabilidade $\times$ Impacto

A matriz é organizada em uma grade de cinco níveis de probabilidade por cinco níveis de impacto, resultando em vinte e cinco combinações possíveis. Cada célula exibe o valor da pontuação correspondente à combinação dos dois fatores e utiliza uma escala de cores para representar o nível de criticidade. Riscos de baixa criticidade são representados em verde, níveis intermediários utilizam tons de amarelo e laranja, enquanto riscos críticos são destacados em vermelho. Além disso, cada célula apresenta a quantidade de riscos atualmente posicionados naquela região da matriz, permitindo identificar rapidamente áreas de maior concentração de ameaças.

Observa-se que a matriz permite identificar visualmente as regiões de maior criticidade do projeto, facilitando a priorização dos riscos que demandam tratamento imediato.

Complementando a análise visual, a página disponibiliza uma legenda de criticidade, filtros e um campo de busca para localização rápida dos registros. A listagem lateral apresenta informações resumidas dos riscos, incluindo seu identificador, título e pontuação calculada a partir da combinação entre probabilidade e impacto. Dessa forma, o gerente de projetos consegue analisar simultaneamente a distribuição geral dos riscos e os registros específicos que compõem cada faixa de criticidade.

RISK-03 – Acompanhar a evolução dos riscos do projeto

Para atender à história RISK-03, foram projetadas ferramentas nativas para o acompanhamento histórico dos riscos, combinando a visualização de gráficos com o gerenciamento de avaliações históricas (*evaluations*). Essa funcionalidade se divide entre o acompanhamento individual, disponível na própria tela de detalhes do risco, e o acompanhamento consolidado de múltiplos riscos do projeto, localizado na página `/risks/evolution`.

A Figura 51 apresenta a interface principal de evolução dos riscos. A tela combina um gráfico temporal com uma lista de seleção dos riscos disponíveis, permitindo comparar simultaneamente a evolução de múltiplos registros do projeto.

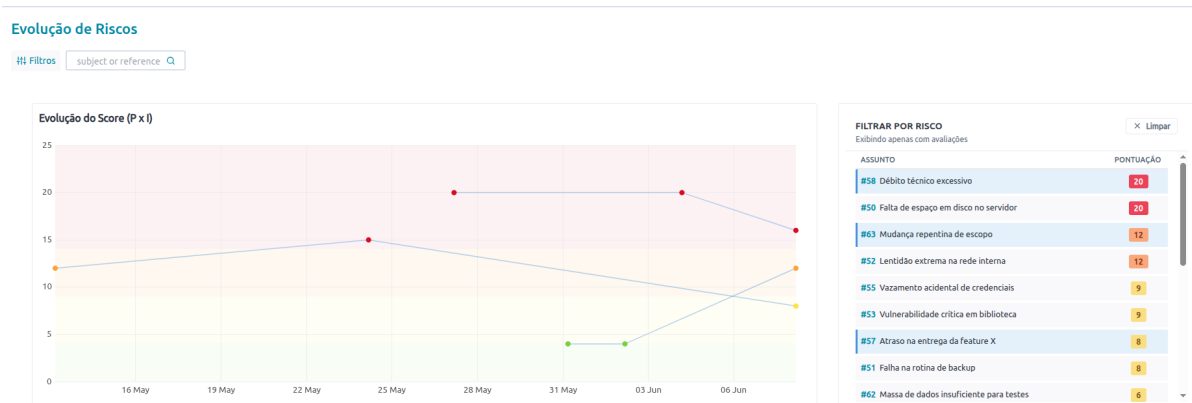


Figura 51 – Painel de evolução dos riscos do projeto

O acompanhamento histórico é baseado em avaliações registradas ao longo do tempo. O sistema gerencia essas avaliações de duas formas: automática e manual. Uma avaliação é criada automaticamente sempre que ocorre uma alteração relevante na criticidade do risco, como mudanças de probabilidade, impacto ou associação a uma sprint. Complementarmente, o gerente do projeto pode adicionar avaliações manuais por meio de uma tabela específica disponível na interface.

Além da visualização consolidada do projeto, o sistema também disponibiliza uma visão individual da evolução de cada risco. Conforme apresentado anteriormente na Figura 49, a parte inferior da tela de detalhes contém um gráfico específico para o risco selecionado, acompanhado de uma tabela histórica contendo as avaliações registradas. Essa abordagem permite analisar tanto o comportamento global dos riscos do projeto quanto a evolução detalhada de cada registro individualmente.

As avaliações podem ser criadas e editadas diretamente na tabela histórica, sem necessidade de abrir formulários adicionais. Para facilitar esse processo, a interface disponibiliza componentes específicos para seleção de datas, classificação de probabilidade e impacto e associação com sprints. Caso uma avaliação seja criada sem os campos obrigatórios de probabilidade e impacto preenchidos, o sistema impede sua conclusão até que as informações necessárias sejam fornecidas.

As informações registradas são utilizadas para compor os gráficos de evolução. O intervalo de datas exibido no eixo horizontal é ajustado automaticamente de acordo com o período efetivamente coberto pelas avaliações dos riscos selecionados, evitando a exibição de períodos sem registros relevantes.

A análise visual da interface `/risks/evolution` também permite filtros mais específicos. O usuário pode selecionar um ou mais riscos para exibição simultânea no gráfico, possibilitando comparar diferentes comportamentos ao longo do tempo. Sempre que a seleção é alterada, o gráfico é recalculado automaticamente para refletir apenas os riscos escolhidos.

Por fim, ao posicionar o cursor sobre os pontos do gráfico, o sistema exibe informações complementares por meio de *tooltips*, incluindo data da avaliação, pontuação calculada, classificação de criticidade e eventual sprint associada ao registro histórico. Dessa forma, o gerente de projetos consegue acompanhar a evolução dos riscos, avaliar a efetividade das ações de mitigação e identificar tendências de aumento ou redução da criticidade ao longo da execução do projeto.

5.6 TESTES

Nesta seção, são apresentados os testes de aceitação utilizados para verificar se as funcionalidades implementadas atendem aos requisitos definidos no Capítulo 4. Para cada épico, são descritos os cenários de teste, as ações realizadas no sistema e as evidências de que os comportamentos esperados foram contemplados. Dessa forma, busca-se demonstrar a aderência das implementações aos critérios estabelecidos para cada requisito.

5.6.1 Épico E1 – Gestão de Cronograma

Para testar as funcionalidades do épico de gestão de cronograma, foi criado um cenário de teste comum às histórias CRON-01, CRON-02 e CRON-03. Esse cenário contempla dois épicos, três histórias de usuário e seis tarefas, com datas de início e término definidas e dependências entre atividades. A utilização do mesmo conjunto de dados permite verificar, sobre um único cronograma, a visualização em gráfico de Gantt, a identificação do caminho crítico e a exibição da folga total.

O teste é executado conforme os seguintes passos:

1. Criar o primeiro épico, denominado *Épico 1 – Planejamento*, com data de início em 02/06/2026 e data de término em 07/06/2026.
2. Criar, dentro do primeiro épico, a história de usuário *História 1 – Levantamento inicial*, com data de início em 03/06/2026 e data de término em 04/06/2026.
3. Criar, dentro da primeira história, a tarefa *Tarefa 1 – Coletar requisitos*, com data de início em 03/06/2026 e data de término em 04/06/2026.
4. Criar, ainda no primeiro épico, a história de usuário *História 2 – Preparação da solução*, com data de início em 05/06/2026 e data de término em 07/06/2026.
5. Criar, dentro da segunda história, a tarefa *Tarefa 2 – Desenhar protótipo*, com data de início em 05/06/2026 e data de término em 07/06/2026.
6. Criar, dentro da segunda história, a tarefa *Tarefa 3 – Revisar alternativa técnica*, com data de início em 05/06/2026 e data de término em 05/06/2026.
7. Criar o segundo épico, denominado *Épico 2 – Implementação*, com data de início em 08/06/2026 e data de término em 15/06/2026.
8. Criar, dentro do segundo épico, a história de usuário *História 3 – Desenvolvimento do módulo*, com data de início em 08/06/2026 e data de término em 15/06/2026.
9. Criar, dentro da terceira história, a tarefa *Tarefa 4 – Implementar back-end*, com data de início em 08/06/2026 e data de término em 10/06/2026.

10. Criar, dentro da terceira história, a tarefa *Tarefa 5 – Implementar front-end*, com data de início em 11/06/2026 e data de término em 13/06/2026.
11. Criar, dentro da terceira história, a tarefa *Tarefa 6 – Testar integração*, com data de início em 14/06/2026 e data de término em 15/06/2026. Essa tarefa representa a última atividade do cronograma e define a data final do projeto neste cenário.
12. Criar as dependências entre as tarefas na seguinte ordem: *Tarefa 1* para *Tarefa 2*, *Tarefa 1* para *Tarefa 3*, *Tarefa 2* para *Tarefa 4*, *Tarefa 3* para *Tarefa 4*, *Tarefa 4* para *Tarefa 5* e *Tarefa 5* para *Tarefa 6*.

CRON-01 – Visualização do cronograma em gráfico de Gantt

A partir do cenário de teste descrito, acessa-se a página `/gantt` do projeto para verificar a representação visual do cronograma. Conforme apresentado na Figura 52, o sistema exibe os dois épicos, as três histórias de usuário e as seis tarefas na estrutura hierárquica da lista lateral, além de representar cada atividade por meio de barras posicionadas de acordo com suas datas de início e término. Também é possível observar as ligações entre as tarefas, evidenciando que as dependências cadastradas são apresentadas no diagrama.

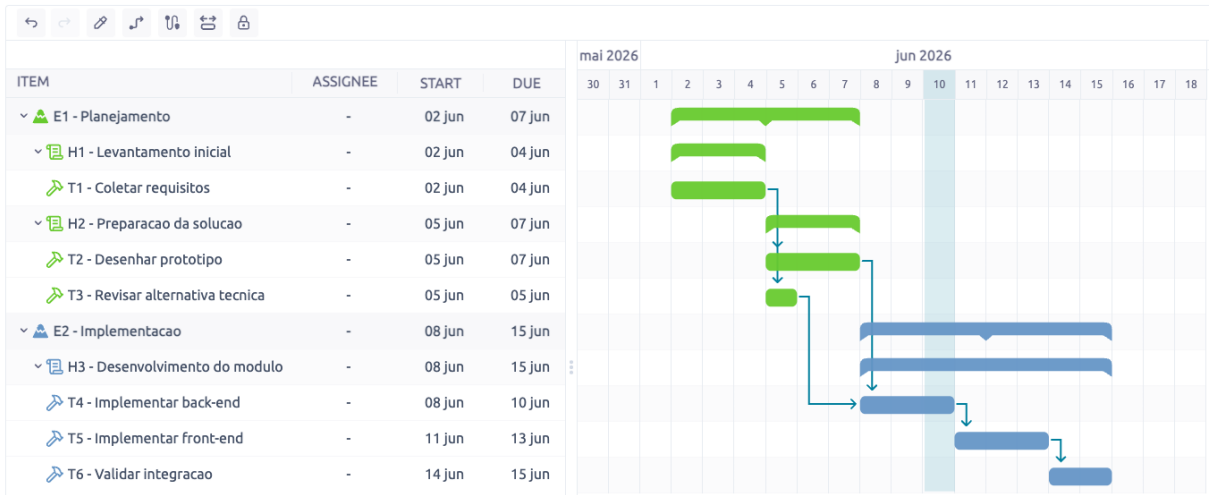


Figura 52 – Resultado da visualização do cronograma no gráfico de Gantt

CRON-02 – Caminho crítico no gráfico de Gantt

Com base no cenário de teste definido no início do épico, acessa-se a página `/gantt` e aciona-se a ferramenta de caminho crítico. O resultado esperado é a diferenciação visual das atividades que compõem o caminho crítico do projeto, isto é, a sequência de atividades cuja postergação impacta diretamente a data final do cronograma.

Conforme apresentado na Figura 53, o sistema destaca o caminho formado pelas tarefas *T1 – Coletar requisitos*, *T2 – Desenhar protótipo*, *T4 – Implementar back-end*, *T5 – Implementar front-end* e *T6 – Testar integração*. As demais atividades permanecem visíveis, porém com opacidade reduzida, permitindo identificar que não pertencem ao caminho crítico calculado para o cenário. Dessa forma, o teste evidencia que a funcionalidade atende ao requisito ao permitir a identificação visual do caminho crítico diretamente no gráfico de Gantt.

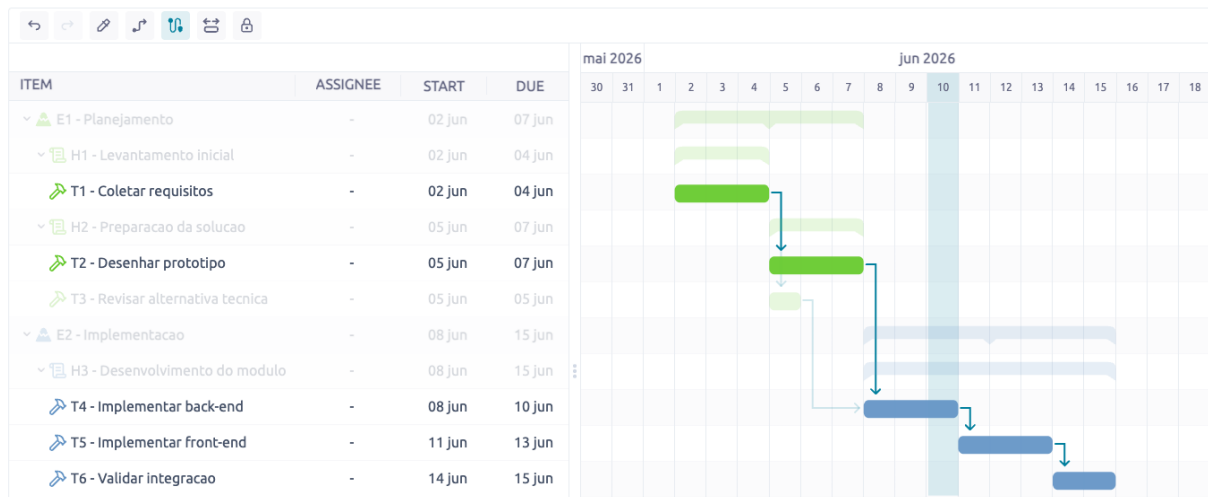


Figura 53 – Resultado da visualização do caminho crítico no gráfico de Gantt

CRON-03 – Folga total das atividades

A avaliação da folga total utiliza o mesmo cenário de teste empregado na visualização do gráfico de Gantt e do caminho crítico. Após acionar a ferramenta de caminho crítico, posiciona-se o cursor sobre as atividades do diagrama para verificar a exibição da folga calculada pelo sistema.

Como apresentado na Figura 54, o sistema exibe a folga da atividade por meio de uma mensagem sobreposta ao gráfico. No cenário construído, a tarefa *T3 – Revisar alternativa técnica* não pertence ao caminho crítico, pois sua conclusão ocorre em 05/06/2026 e a próxima atividade dependente, *T4 – Implementar back-end*, tem início em 08/06/2026. Assim, considerando as datas cadastradas, espera-se uma folga total de 2 dias para essa tarefa. O resultado do teste demonstra que o sistema calcula e apresenta essa informação ao usuário, permitindo identificar quanto uma atividade pode atrasar sem comprometer o término do cronograma.

5.6.2 Épico E2 – Gestão Financeira

Para testar as funcionalidades do épico de gestão financeira, utiliza-se o mesmo projeto e o mesmo conjunto de atividades definido nos testes de cronograma. O cenário,



Figura 54 – Resultado da visualização da folga total de uma atividade

portanto, permanece composto por dois épicos, três histórias de usuário e seis tarefas, permitindo avaliar as funcionalidades financeiras sobre uma estrutura hierárquica já cadastrada no Taiga.

CUST-01 – Custo planejado da tarefa

A avaliação do requisito CUST-01 é realizada a partir da página `/activity-cost`, na qual são cadastradas as horas planejadas e os custos planejados das atividades previamente criadas. Para o cenário de teste, os valores são inseridos diretamente na listagem de custos por atividade, permitindo verificar se o sistema registra o planejamento financeiro associado às tarefas e apresenta os valores consolidados conforme a hierarquia de épicos, histórias e tarefas.

O teste é executado conforme os seguintes passos:

1. Acessar o projeto *Simulação*.
2. Acessar a página `/activity-cost`.
3. Localizar as tarefas criadas no cenário de teste do cronograma.
4. Preencher, para cada tarefa, os valores de horas planejadas e custo planejado.
5. Verificar se os valores cadastrados permanecem associados às tarefas após a atualização da página.
6. Verificar se histórias de usuário e épicos apresentam a consolidação dos custos planejados das atividades subordinadas.
7. Acessar o detalhamento financeiro de uma atividade.
8. Distribuir o custo planejado da atividade entre categorias financeiras do projeto.
9. Verificar se a distribuição por categoria é exibida no detalhamento financeiro da atividade.

A Tabela 1 apresenta os valores utilizados para o cadastro de horas planejadas e custos planejados no cenário de teste.

Como resultado esperado, a página `/activity-cost` deve apresentar as tarefas com seus respectivos valores de horas planejadas e custo planejado, conforme ilustrado na Figura 55. Além disso, as histórias de usuário e os épicos devem refletir a consolidação

Tabela 1 – Valores planejados cadastrados para teste do requisito CUST-01

Atividade	Horas planejadas	Custo planejado
T1 – Coletar requisitos	8	R\$ 800,00
T2 – Desenhar protótipo	12	R\$ 1.200,00
T3 – Revisar alternativa técnica	4	R\$ 400,00
T4 – Implementar back-end	16	R\$ 1.600,00
T5 – Implementar front-end	16	R\$ 1.600,00
T6 – Testar integração	8	R\$ 800,00

dos valores cadastrados em suas atividades subordinadas, permitindo acompanhar o planejamento financeiro na mesma estrutura hierárquica utilizada para organizar o trabalho do projeto.

Custo de Atividade

Filtros
Tags

TIPO	ASSUNTO	HORAS PLANEJADAS	HORAS REALIZADAS	CUSTO PLANEJADO	CUSTO REALIZADO	ATRIBUÍDO A
EPIC	#1 E1 - Planejamento	32.00	0.00	\$ 2400.00	\$ 0.00	Não-atribuído
EPIC	#2 E2 - Implementacao	40.00	0.00	\$ 4000.00	\$ 0.00	Não-atribuído
HISTORY	#3 H1 - Levantamento inicial	8.00	0.00	\$ 800.00	\$ 0.00	Não-atribuído
HISTORY	#4 H2 - Preparacao da solucao	24.00	0.00	\$ 1600.00	\$ 0.00	Não-atribuído
HISTORY	#5 H3 - Desenvolvimento do modulo	40.00	0.00	\$ 4000.00	\$ 0.00	Não-atribuído
TASK	#6 T1 - Coletar requisitos	8.00	0.00	\$ 800.00	\$ 0.00	Não-atribuído
TASK	#7 T2 - Desenhar prototipo	12.00	0.00	\$ 1200.00	\$ 0.00	Não-atribuído
TASK	#8 T3 - Revisar alternativa tecnica	4.00	0.00	\$ 400.00	\$ 0.00	Não-atribuído
TASK	#9 T4 - Implementar back-end	16.00	0.00	\$ 1600.00	\$ 0.00	Não-atribuído
TASK	#10 T5 - Implementar front-end	16.00	0.00	\$ 1600.00	\$ 0.00	Não-atribuído
TASK	#11 T6 - Validar integracao	8.00	0.00	\$ 800.00	\$ 0.00	Não-atribuído

Figura 55 – Cadastro de horas planejadas e custo planejado na listagem de custos por atividade

No detalhamento financeiro da atividade, o custo planejado também pode ser distribuído entre categorias financeiras do projeto. Conforme apresentado na Figura 56, essa funcionalidade permite verificar não apenas o valor planejado total da atividade, mas também sua composição por categoria, apoiando uma análise mais detalhada do planejamento financeiro.

Para testar as funcionalidades do épico de gestão financeira, utiliza-se o mesmo projeto e o mesmo conjunto de atividades definido nos testes de cronograma. O cenário, portanto, permanece composto por dois épicos, três histórias de usuário e seis tarefas, permitindo avaliar as funcionalidades financeiras sobre uma estrutura hierárquica já cadastrada no Taiga. Para teste da distribuição do custo planejado, as atividades são associadas a categorias financeiras do projeto, conforme apresentado na Tabela 2.

Tabela 2 – Categorias financeiras utilizadas no cenário de teste

Atividade	Categoria financeira
T1 – Coletar requisitos	Gestão
T2 – Desenhar protótipo	Desenvolvimento
T3 – Revisar alternativa técnica	Gestão
T4 – Implementar back-end	Desenvolvimento
T5 – Implementar front-end	Desenvolvimento
T6 – Testar integração	Garantia de qualidade

#1 Épico 1 – Planejamento

DETALHAMENTO FINANCEIRO DA ATIVIDADE

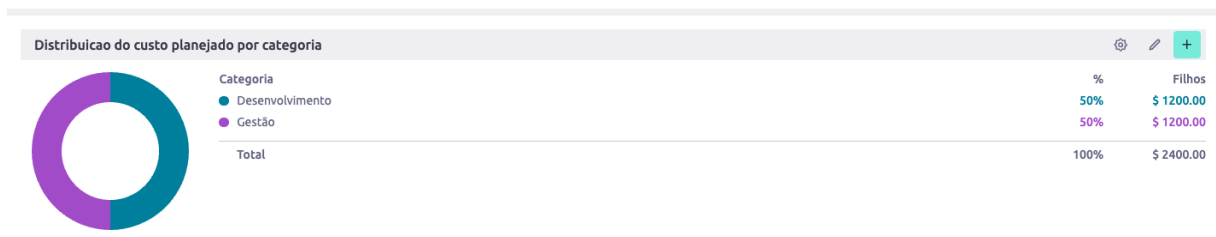


Figura 56 – Distribuição do custo planejado por categorias no detalhamento financeiro do épico 1

CUST-02 – Horas trabalhadas e custo realizado

O teste do requisito CUST-02 tem como objetivo verificar o registro de horas trabalhadas e o cálculo automático do custo realizado das atividades. Para isso, utiliza-se o mesmo conjunto de tarefas do cenário de teste, configurando inicialmente o valor/hora dos integrantes do projeto e, em seguida, registrando entradas reais de trabalho associadas às tarefas.

O teste é executado conforme os seguintes passos:

1. Acessar o projeto *Simulação*.
2. Acessar a página `/member-cost`.
3. Configurar o valor/hora dos integrantes do projeto, conforme apresentado na Figura 57.
4. Inserir entradas de tempo simuladas no banco de dados para os integrantes do projeto.
5. Acessar a página `/activity-cost`.
6. Verificar se as horas realizadas são atualizadas a partir das entradas registradas.
7. Verificar se o custo realizado é calculado automaticamente a partir das horas trabalhadas e do valor/hora configurado para cada integrante.

8. Acessar o detalhamento financeiro de uma atividade e verificar se as entradas de trabalho aparecem no histórico da atividade.

Tabela 3 – Valores/hora configurados para os integrantes do projeto

Integrante	Papel no projeto	Valor/hora
user1	Project Owner	R\$ 120,00
user2	Front	R\$ 80,00
user3	Back	R\$ 100,00

Custo por Membro

Filtros subject or reference

MEMBRO	TRABALHANDO AGORA	FUNCAO	CUSTO / HORA	ULTIMA ATUALIZACAO
user 1	-	Product Owner	\$ 120.00	10 jun 2026 17:18
user 2	-	Front	\$ 80.00	10 jun 2026 17:18
user3	-	Back	\$ 100.00	10 jun 2026 17:18

Figura 57 – Configuração do valor/hora dos integrantes do projeto

O registro de uma entrada de tempo pelo usuário é realizado pela página `/my-activities`, apresentada na Figura 58. Nessa página, o integrante do projeto inicia o temporizador associado a uma atividade, criando uma entrada de tempo em aberto. Enquanto o temporizador permanece ativo, a entrada possui apenas o instante de início. Ao interromper o temporizador, o sistema registra o instante de fim e completa a entrada de trabalho, permitindo calcular a duração do período trabalhado e o custo realizado a partir do valor/hora configurado para o integrante.

Minhas Atividades

Filtros subject or reference

#8 T3 - Revisar alternativa tecnica 01:30:00

TIPO	ASSUNTO	ESTAGIO	HORAS TRABALHADAS	ULTIMO TRABALHO
TASK	#6 T1 - Coletar requisitos	New	6.00	02 jun 2026 16:00
TASK	#8 T3 - Revisar alternativa tecnica	New	3.00	05 jun 2026 12:00

Figura 58 – Página de listas de atividades de um integrante do projeto

Para a execução do teste, além da configuração dos valores/hora na interface do sistema, foram inseridas manualmente no banco de dados entradas de tempo simuladas para os integrantes do projeto. Essa inserção direta não representa o fluxo usual de uso da funcionalidade, mas é utilizada apenas como mecanismo de preparação do cenário de teste, permitindo representar registros históricos de trabalho já concluídos para diferentes membros e atividades. As entradas utilizadas são apresentadas na Tabela 4.

Como resultado esperado, a página `/activity-cost` deve apresentar as horas realizadas e o custo realizado das atividades, sem permitir a edição direta desses valores,

Tabela 4 – Entradas de tempo simuladas utilizadas para teste do requisito CUST-02

Atividade	Integrante	Início	Fim	Custo realizado
T1	user1	02/06/2026 09:00	02/06/2026 12:00	R\$ 360,00
T1	user1	02/06/2026 13:00	02/06/2026 16:00	R\$ 360,00
T2	user2	05/06/2026 08:00	05/06/2026 13:00	R\$ 400,00
T2	user2	05/06/2026 14:00	05/06/2026 19:00	R\$ 400,00
T3	user1	05/06/2026 09:00	05/06/2026 10:30	R\$ 180,00
T3	user1	05/06/2026 10:30	05/06/2026 12:00	R\$ 180,00
T4	user3	08/06/2026 09:00	08/06/2026 16:00	R\$ 700,00
T4	user3	09/06/2026 09:00	09/06/2026 16:00	R\$ 700,00

pois eles são derivados dos apontamentos de trabalho e da taxa horária configurada para cada membro. Esse resultado é apresentado na Figura 59.

Custo de Atividade

Filtros Tags

TIPO	ASSUNTO	HORAS PLANEJADAS	HORAS REALIZADAS	CUSTO PLANEJADO	CUSTO REALIZADO	ATRIBUIDO A
EPIC	#1 E1 - Planejamento	32.00	19.00	\$ 2400.00	\$ 1880.00	Não-atribuído
EPIC	#2 E2 - Implementacao	40.00	14.00	\$ 4000.00	\$ 1400.00	Não-atribuído
HISTORY	#3 H1 - Levantamento inicial	8.00	6.00	\$ 800.00	\$ 720.00	Não-atribuído
HISTORY	#4 H2 - Preparacao da solucao	24.00	13.00	\$ 1600.00	\$ 1160.00	Não-atribuído
HISTORY	#5 H3 - Desenvolvimento do modulo	40.00	14.00	\$ 4000.00	\$ 1400.00	Não-atribuído
TASK	#6 T1 - Coletar requisitos	8.00	6.00	\$ 800.00	\$ 720.00	Não-atribuído
TASK	#7 T2 - Desenhar prototipo	12.00	10.00	\$ 1200.00	\$ 800.00	Não-atribuído
TASK	#8 T3 - Revisar alternativa tecnica	4.00	3.00	\$ 400.00	\$ 360.00	Não-atribuído
TASK	#9 T4 - Implementar back-end	16.00	14.00	\$ 1600.00	\$ 1400.00	Não-atribuído
TASK	#10 T5 - Implementar front-end	16.00	0.00	\$ 1600.00	\$ 0.00	Não-atribuído
TASK	#11 T6 - Validar integracao	8.00	0.00	\$ 800.00	\$ 0.00	Não-atribuído

Figura 59 – Horas realizadas e custo realizado calculados na listagem de custos por atividade

No detalhamento financeiro da atividade, apresentado na Figura 60, também é possível verificar as entradas de trabalho que compõem as horas realizadas e o custo realizado. Dessa forma, o teste evidencia que o sistema calcula automaticamente os valores executados e mantém a rastreabilidade das entradas utilizadas na composição do custo real.

CUST-03 – Receita contratada do projeto

O teste do requisito CUST-03 tem como objetivo verificar o cadastro de receitas contratadas do projeto. Para isso, utiliza-se a página `/project-income`, na qual são registradas receitas associadas ao projeto *Simulação*. O cenário de teste contempla três



Figura 60 – Entradas de trabalho no detalhamento financeiro da atividade de T2

entradas de receita: duas receitas de pagamento único e uma receita recorrente mensal, configurada para ocorrer durante dois meses, com início em abril.

O teste é executado conforme os seguintes passos:

1. Acessar o projeto *Simulação*.
2. Acessar a página `/project-income`.
3. Cadastrar duas receitas de pagamento único.
4. Cadastrar uma receita com pagamento mensal por dois meses, iniciando em abril.
5. Verificar se as três receitas são apresentadas na listagem de receitas do projeto.
6. Acessar o detalhamento de uma receita cadastrada.
7. Verificar se os dados da receita são apresentados corretamente na página de detalhamento.

A Tabela 5 apresenta as receitas utilizadas no cenário de teste.

Tabela 5 – Receitas cadastradas para teste do requisito CUST-03

Receita	Tipo	Valor	Dia de pagamento	Período de validade
Contrato inicial	Pagamento único	R\$ 8.000,00	–	01/04/2026
Entrega intermediária	Pagamento único	R\$ 4.000,00	–	15/05/2026
Suporte mensal	Pagamento mensal	R\$ 2.000,00	10	01/04/2026 a 31/05/2026

Como resultado esperado, a página `/project-income` deve apresentar as três receitas cadastradas, permitindo identificar o título, o valor, o período de validade, o status e, quando aplicável, o dia de pagamento mensal. Esse resultado é apresentado na Figura 61.

Além da listagem, o detalhamento de uma receita permite consultar seus dados específicos e realizar alterações conforme as permissões do usuário. Conforme apresentado na Figura 62, a tela de detalhe exibe os campos da receita selecionada, incluindo título, descrição, valor, finalidade, período de validade, status e dia de pagamento mensal quando houver recorrência.

TITULO DA RECEITA	VALOR	DIA DO PAGAMENTO	VALIDO DE	VALIDO ATE	ULTIMA ATUALIZACAO
#1 Contrato inicial	\$ 8000.00	pagamento único	01 abr 2026	-	10 jun 2026 17:41
#2 Entrega intermediaria	\$ 4000.00	pagamento único	15 mai 2026	-	10 jun 2026 17:41
#3 Suporte mensal	\$ 2000.00	10	01 abr 2026	31 mai 2026	10 jun 2026 17:41

Figura 61 – Listagem de receitas cadastradas no projeto

#2 Entrega intermediaria

RECEITA

DESCRICAO

Receita simulada de pagamento unico para uma entrega intermediaria do projeto.

FINALIDADE

Pagamento unico associado a uma entrega intermediaria.

DADOS DA RECEITA ATIVA
VALOR
\$ 4000.00
DIA DO PAGAMENTO
pagamento único
VALIDO DE
15 mai 2026
VALIDO ATE
-
ULTIMA ATUALIZACAO
10 jun 2026 17:41
[DEFINIR COMO INATIVA](#)

Figura 62 – Detalhamento de uma receita cadastrada no projeto

CUST-04 – Resumo financeiro do projeto

O teste da história CUST-04 é realizada a partir da página `/project-dashboard`, responsável por consolidar os dados financeiros cadastrados nos requisitos anteriores. Como a tela apresenta diferentes grupos de indicadores e gráficos, o resultado do teste é registrado em duas figuras: a Figura 63 apresenta os indicadores financeiros consolidados, enquanto a Figura 64 apresenta os gráficos exibidos após a rolagem da página.

No cenário de teste, os valores planejados cadastrados no requisito CUST-01 totalizam 64 horas planejadas e R\$ 6.400,00 de custo planejado. A distribuição por categoria resulta em R\$ 1.200,00 para gestão, R\$ 4.400,00 para desenvolvimento e R\$ 800,00 para teste. A partir das entradas de trabalho registradas no requisito CUST-02, considerando apenas as atividades T1, T2, T3 e T4, o projeto passa a apresentar 33 horas realizadas e R\$ 3.280,00 de custo realizado, resultando em custo médio aproximado de R\$ 99,39 por hora.

As receitas cadastradas no requisito CUST-03 totalizam R\$ 16.000,00 recebidos no período considerado. Esse valor é composto por duas receitas pontuais, de R\$ 8.000,00 e R\$ 4.000,00, e por duas ocorrências da receita mensal de R\$ 2.000,00, referentes aos meses de abril e maio. Dessa forma, o saldo financeiro esperado no dashboard é de R\$ 12.720,00, obtido pela diferença entre a receita recebida e o custo realizado.

Com isso, o teste demonstra que a página de resumo financeiro consolida, em uma única visualização, os custos planejados, os custos realizados, as horas trabalhadas, as receitas cadastradas e o saldo financeiro do projeto. Também é possível observar que

Tabela 6 – Resumo dos valores esperados no dashboard financeiro

Indicador	Valor esperado
Custo planejado total	R\$ 6.400,00
Horas realizadas	33 h
Custo realizado total	R\$ 3.280,00
Custo médio por hora	R\$ 99,39
Receita recebida	R\$ 16.000,00
Saldo financeiro	R\$ 12.720,00

atividades sem entradas reais de trabalho, como T5 e T6 neste cenário, permanecem sem contribuição para o custo realizado, ainda que possuam valores planejados associados.

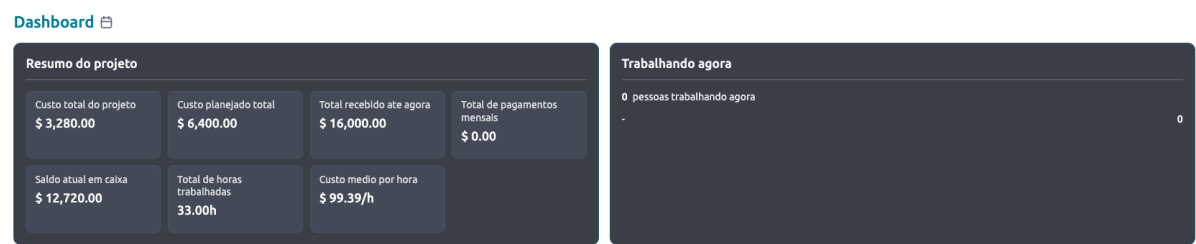


Figura 63 – Indicadores financeiros consolidados no dashboard do projeto



Figura 64 – Gráficos financeiros consolidados no dashboard do projeto

5.6.3 Épico E3 – Gestão de Riscos

Para testar as funcionalidades implementadas no módulo de gestão de riscos, foi elaborado um cenário de teste contendo dez riscos distribuídos em diferentes combinações de probabilidade e impacto. O conjunto de dados foi planejado de forma a contemplar riscos classificados nos níveis Baixo, Médio, Alto e Crítico, permitindo verificar o correto funcionamento dos mecanismos de cadastro, classificação, filtragem, busca, acompanhamento e atualização dos registros.

Os riscos utilizados no cenário representam situações comuns em projetos de desenvolvimento de software, incluindo problemas relacionados a requisitos, integração entre sistemas, segurança, desempenho, escopo e gestão da equipe. A distribuição dos registros em diferentes níveis de criticidade também possibilita testar as funcionalidades de análise visual implementadas nas histórias subsequentes do épico.

Os dados utilizados durante os testes são apresentados na Tabela 7, servindo como base para os testes das histórias RISK-01, RISK-02 e RISK-03.

Tabela 7 – Riscos utilizados no cenário de teste

Assunto	Probabilidade	Impacto	Score	Nível
Requisitos Mal Definidos	5	5	25	Crítico
Falha na Integração com APIs de Terceiros	3	4	12	Alto
Escopo Ampliado pelo Cliente	3	3	9	Médio
Falhas de Segurança	2	4	8	Médio
Atraso na Homologação pelo Cliente	2	3	6	Médio
Problemas de Performance	3	2	6	Médio
Adoção de IA Generativa para Desenvolvimento	4	1	4	Baixo
Turnover de Desenvolvedor-Chave	1	2	2	Baixo
Ambiente de Desenvolvimento Inconsistente	1	1	1	Baixo
Dívida Técnica Acumulada	4	4	0	Baixo

Fonte: Elaborado pelo autor

RISK-01 – Gerenciar registro de riscos

O teste da história RISK-01 tem como objetivo verificar as funcionalidades de cadastro, consulta, edição, busca, filtragem, ordenação e atribuição de responsáveis aos riscos cadastrados no projeto.

O teste é executado conforme os seguintes passos:

1. Acessar o projeto utilizado para teste do módulo de riscos.
2. Navegar até a página de gerenciamento de riscos (`/risks`).
3. Cadastrar os riscos definidos no cenário de teste.
4. Informar para cada risco seus respectivos valores de probabilidade, impacto, categoria, estratégia de resposta e responsável.
5. Verificar se os riscos cadastrados são exibidos corretamente na listagem principal.
6. Executar ordenações utilizando diferentes colunas da tabela.
7. Realizar pesquisas por meio do campo de busca textual.
8. Aplicar filtros utilizando os critérios disponibilizados pela interface.
9. Alterar o status de riscos diretamente pela listagem.
10. Atribuir responsáveis diretamente pela listagem.
11. Acessar a tela de detalhes de um risco.
12. Complementar as informações do risco com descrição, causa, consequência e plano de resposta.
13. Definir ou alterar a data limite do risco, registrando a justificativa associada ao prazo estabelecido.
14. Associar o risco a uma ou mais Sprints do projeto.
15. Alterar os valores de probabilidade e impacto e verificar se a pontuação e o nível de criticidade são recalculados automaticamente pelo sistema.

Inicialmente é realizado o cadastro dos riscos definidos para o cenário de teste. Conforme pode ser observado na Figura 65, o formulário de cadastro permite registrar informações descritivas do risco, como assunto, descrição, causa, consequência e plano de resposta. Além disso, a interface disponibiliza campos para definição da probabilidade, impacto, categoria, tipo, estratégia de resposta e responsável. A partir dos valores de probabilidade e impacto informados, o sistema calcula automaticamente a pontuação e o nível de criticidade do risco.

Além dos campos textuais utilizados para descrever o risco, a interface também permite definir informações relacionadas à sua classificação e tratamento. Entre elas estão a probabilidade de ocorrência, o impacto esperado, a categoria do risco, a estratégia de

NEW RISK

Subject
Ambiente de Desenvolvimento Inconsistente

Description
Description

Cause
Enter the root cause of the risk...

Consequence
Enter the potential consequences...

Response Plan
Define the planned actions...

Attributes:

- status: Identified
- probability: 1 - Very Low
- impact: 1 - Very Low
- Score: 1 - Low
- category: Technical
- type: Threat
- strategy: Avoid

CREATE

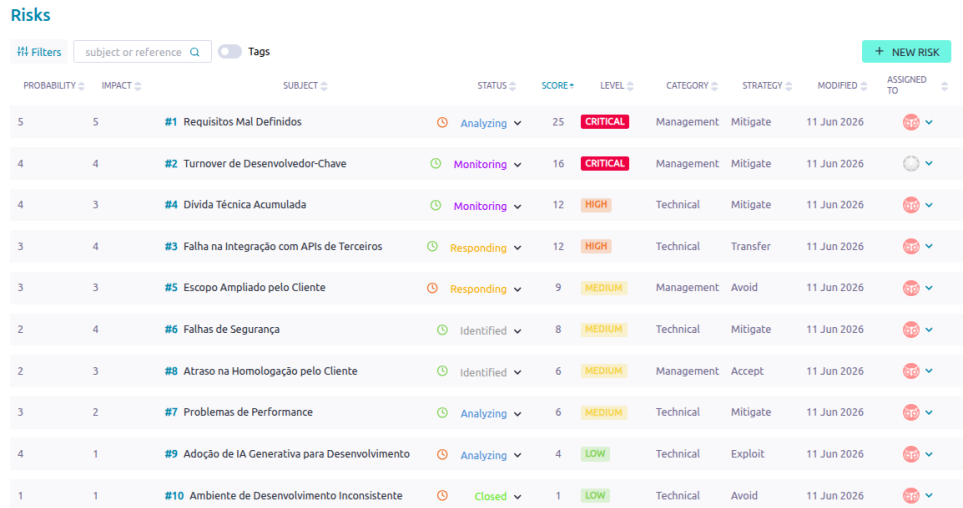
Figura 65 – Tela de cadastro de um novo risco

resposta adotada, o responsável pelo acompanhamento e a data prevista para monitoramento ou mitigação. Essas informações são utilizadas posteriormente pelos mecanismos de classificação e acompanhamento implementados no módulo.

Após o cadastro dos registros, a página principal de riscos passou a exibir todos os itens cadastrados em formato de lista. Conforme apresentado na Figura 66, a interface reúne em uma única visualização as principais informações utilizadas no acompanhamento dos riscos.

A lista apresenta informações relacionadas à probabilidade, impacto, assunto, status, pontuação, nível de criticidade, categoria, estratégia de resposta, data da última atualização e responsável atribuído. Os cabeçalhos das colunas podem ser utilizados para reordenar os registros conforme o critério desejado, facilitando a identificação dos riscos mais relevantes para o projeto. No exemplo, a lista em questão foi ordenada pela pontuação (score).

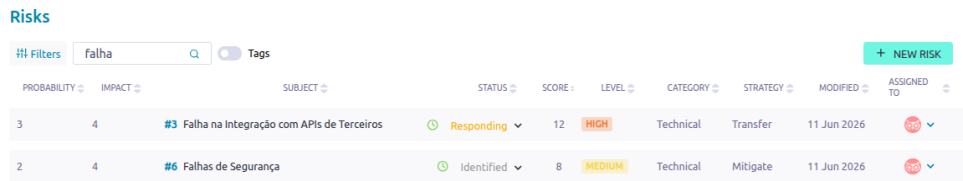
Para auxiliar a localização de registros específicos, a interface disponibiliza um mecanismo de busca textual. A Figura 67 apresenta o resultado de uma pesquisa realizada utilizando o termo “falha”, retornando apenas os riscos compatíveis com o critério



PROBABILITY	IMPACT	SUBJECT	STATUS	SCORE	LEVEL	CATEGORY	STRATEGY	MODIFIED	ASSIGNED TO
5	5	#1 Requisitos Mal Definidos	Analyzing	25	CRITICAL	Management	Mitigate	11 Jun 2026	
4	4	#2 Turnover de Desenvolvedor-Chave	Monitoring	16	CRITICAL	Management	Mitigate	11 Jun 2026	
4	3	#4 Dívida Técnica Acumulada	Monitoring	12	HIGH	Technical	Mitigate	11 Jun 2026	
3	4	#3 Falha na Integração com APIs de Terceiros	Responding	12	HIGH	Technical	Transfer	11 Jun 2026	
3	3	#5 Escopo Ampliado pelo Cliente	Responding	9	MEDIUM	Management	Avoid	11 Jun 2026	
2	4	#6 Falhas de Segurança	Identified	8	MEDIUM	Technical	Mitigate	11 Jun 2026	
2	3	#8 Atraso na Homologação pelo Cliente	Identified	6	MEDIUM	Management	Accept	11 Jun 2026	
3	2	#7 Problemas de Performance	Analyzing	6	MEDIUM	Technical	Mitigate	11 Jun 2026	
4	1	#9 Adoção de IA Generativa para Desenvolvimento	Analyzing	4	LOW	Technical	Exploit	11 Jun 2026	
1	1	#10 Ambiente de Desenvolvimento Inconsistente	Closed	1	LOW	Technical	Avoid	11 Jun 2026	

Figura 66 – Listagem principal dos riscos cadastrados

informado.



PROBABILITY	IMPACT	SUBJECT	STATUS	SCORE	LEVEL	CATEGORY	STRATEGY	MODIFIED	ASSIGNED TO
3	4	#3 Falha na Integração com APIs de Terceiros	Responding	12	HIGH	Technical	Transfer	11 Jun 2026	
2	4	#6 Falhas de Segurança	Identified	8	MEDIUM	Technical	Mitigate	11 Jun 2026	

Figura 67 – Busca textual de riscos

Além da busca textual, o sistema disponibiliza filtros avançados que permitem restringir os resultados apresentados na listagem. Esses filtros podem ser utilizados para segmentar os riscos por categoria, nível de criticidade, status, responsável, autor, tipo ou etiquetas associadas. A Figura 68 apresenta um exemplo de filtragem por categoria.

Com o objetivo de reduzir a quantidade de navegações necessárias durante o gerenciamento dos riscos, determinadas ações podem ser executadas diretamente na listagem principal. Entre elas está a alteração do status do risco, realizada por meio de um seletor disponível na própria tabela como mostra a Figura 69

Da mesma forma, o responsável pelo acompanhamento do risco pode ser definido sem a necessidade de acessar sua tela de detalhes. Ao selecionar o campo correspondente, o sistema apresenta uma janela para pesquisa e atribuição dos usuários cadastrados no projeto, conforme apresentado na Figura 70.

Embora diversas operações possam ser realizadas diretamente na listagem, o gerenciamento completo dos riscos é realizado por meio da tela de detalhes do registro. Essa interface centraliza todas as informações relacionadas ao risco e constitui o principal ponto de acompanhamento durante seu ciclo de vida.

Conforme ilustrado na Figura 71, a área central da tela reúne informações descritti-

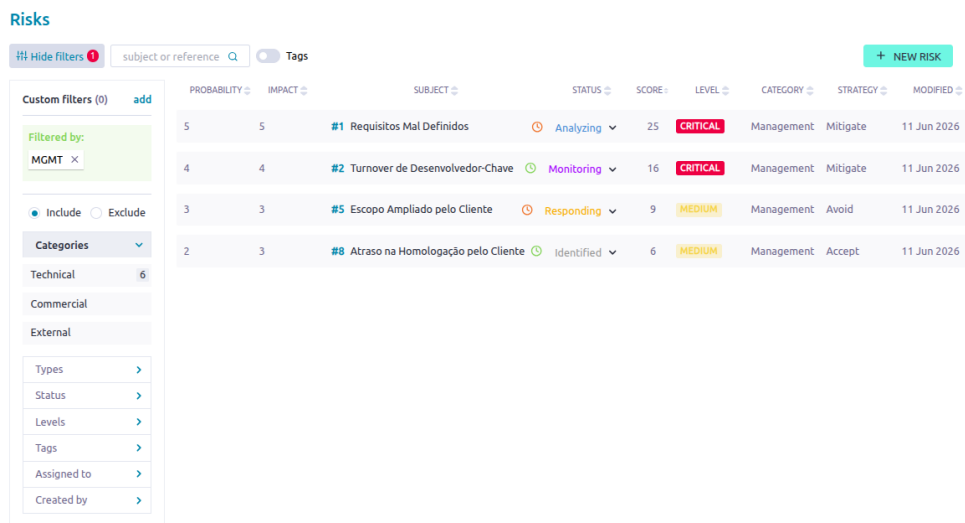


Figura 68 – Filtragem de riscos por categoria

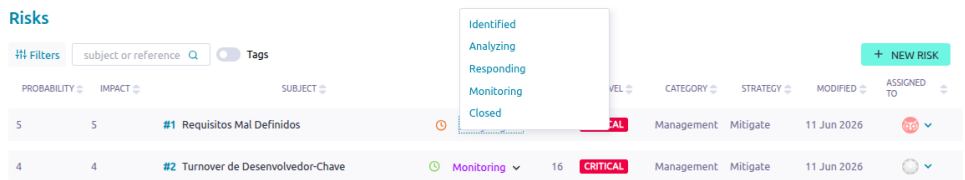


Figura 69 – Alteração de status diretamente pela listagem

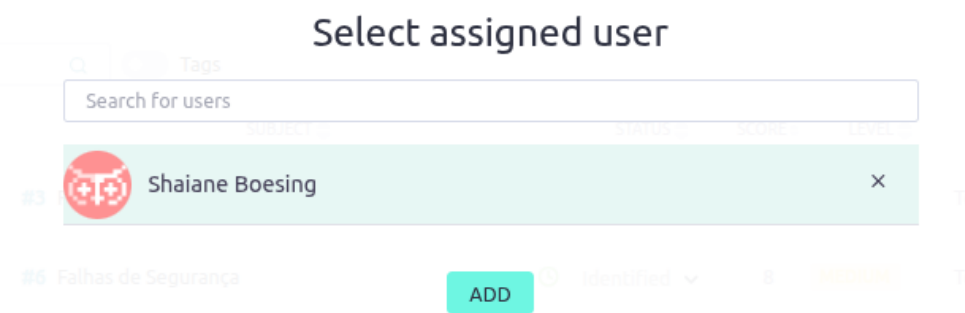
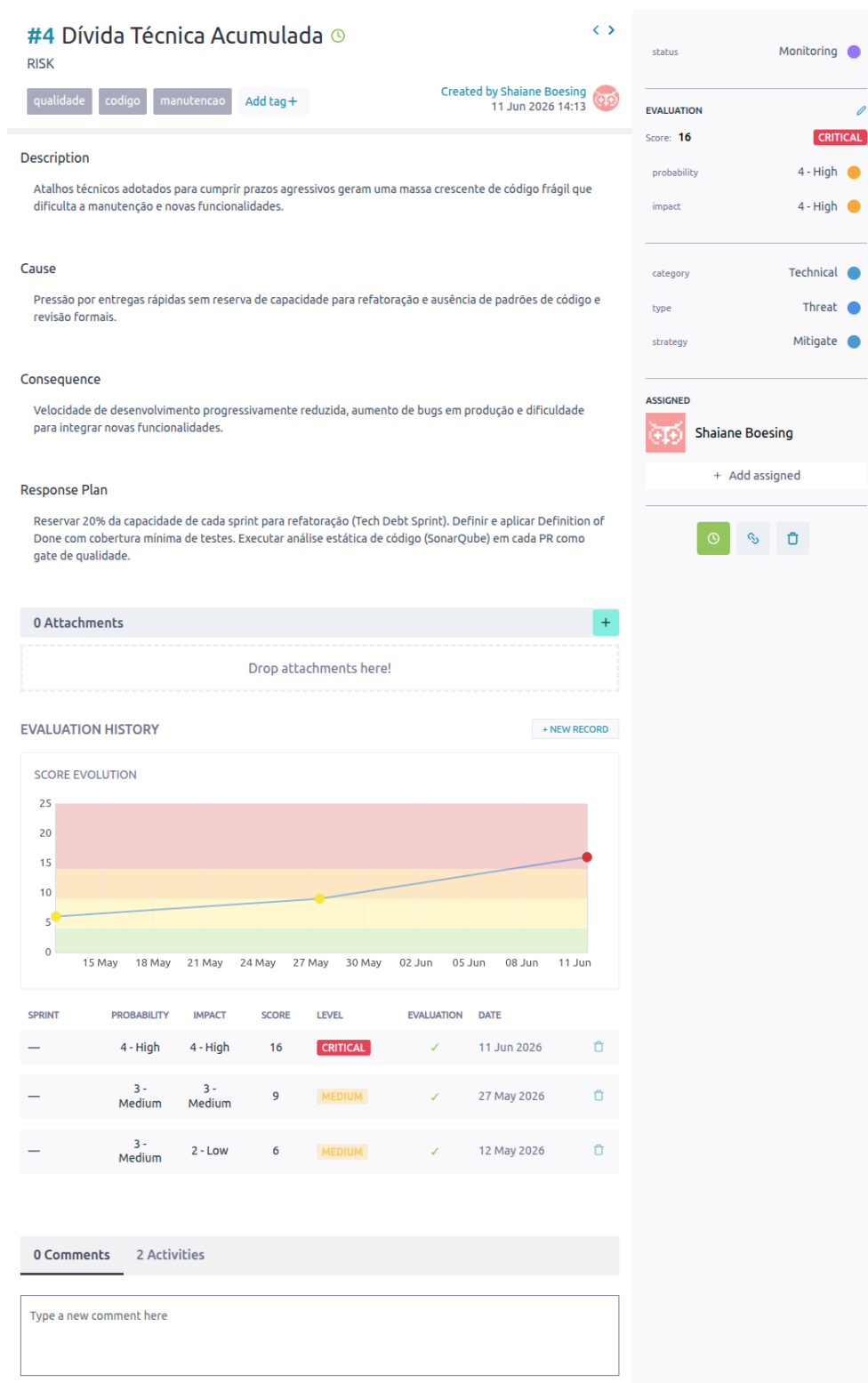


Figura 70 – Atribuição de responsável

vas do risco, incluindo descrição, causa, consequência e plano de resposta. Esses campos permitem documentar a origem do problema, seus possíveis impactos e as ações planejadas para tratamento ou mitigação.

Na lateral direita são apresentadas as informações de classificação do risco, incluindo probabilidade, impacto, pontuação calculada, nível de criticidade, categoria, tipo e estratégia de resposta. Sempre que os valores de probabilidade ou impacto são alterados, o sistema recalcula automaticamente a pontuação e atualiza a classificação correspondente.

A tela também disponibiliza recursos para anexação de documentos, registro de



comentários, acompanhamento das atividades realizadas e evolução do risco com o passar do tempo e das sprints do projeto. Dessa forma, torna-se possível registrar tanto as informações técnicas do risco quanto o histórico de ações executadas durante seu acompanhamento. Os componentes relacionados à evolução do risco e ao acompanhamento gráfico das avaliações são apresentados em maior detalhe no teste da história RISK-03.

Além do cadastro e edição das informações principais do risco, a tela de detalhes disponibiliza recursos complementares para apoiar seu acompanhamento ao longo do projeto. Um desses recursos é a definição de uma data limite para tratamento ou monitoramento do risco. Ao acionar essa funcionalidade, o sistema apresenta um formulário específico para seleção da data desejada e para o registro da justificativa associada ao prazo estabelecido.

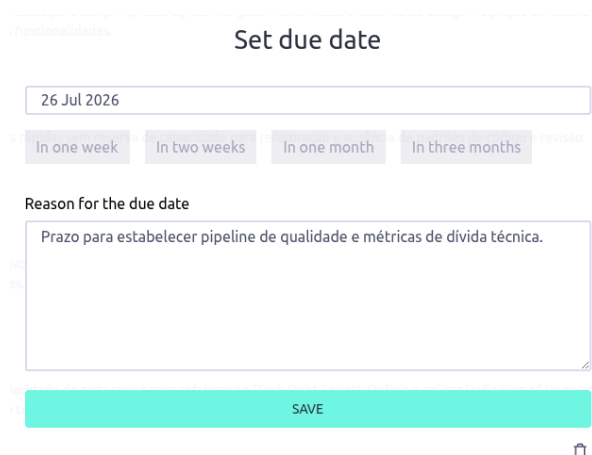


Figura 72 – Definição da data de validade e justificativa do risco

Conforme ilustrado na Figura 72, o recurso permite definir o prazo de validade para o tratamento do risco, registrando também o motivo da data escolhida. Essa informação auxilia o acompanhamento das ações de mitigação e facilita a gestão das responsabilidades relacionadas ao risco.

Outro recurso disponível consiste na associação de riscos às Sprints do projeto. Essa funcionalidade permite vincular um risco às sprints em que serão executadas atividades de monitoramento, mitigação ou resposta.

A Figura 73 apresenta a interface utilizada para essa associação. Por meio dela, o usuário pode selecionar uma ou mais Sprints disponíveis, permitindo acompanhar a evolução do risco em diferentes ciclos de trabalho do projeto.

Durante os testes foi verificado que alterações nos níveis de probabilidade e impacto refletem imediatamente na pontuação calculada e na classificação do risco, mantendo consistência com as regras de criticidade definidas para o módulo de gestão de riscos.

Os resultados obtidos demonstraram o correto funcionamento das operações de cadastro, consulta, busca, filtragem, ordenação, atribuição de responsáveis e manutenção dos registros de risco. Dessa forma, conclui-se que as funcionalidades implementadas atendem aos requisitos definidos para a história RISK-01.

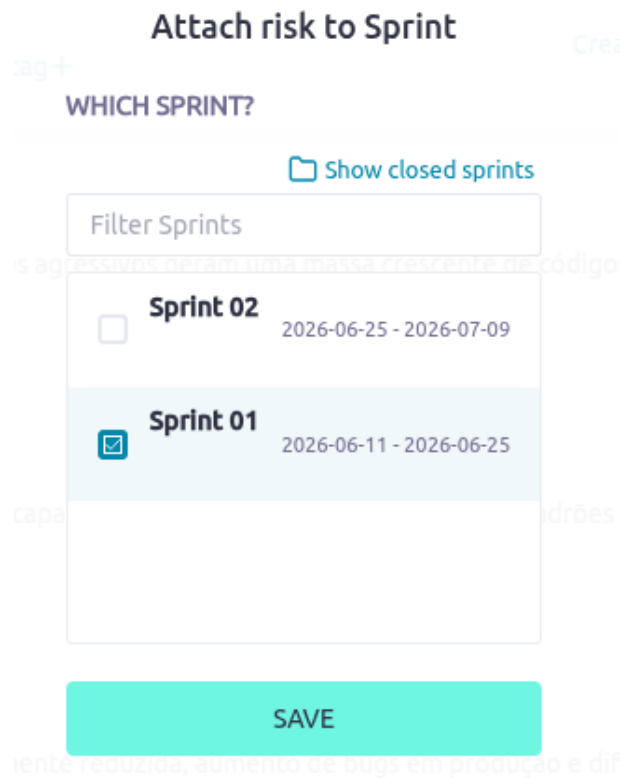


Figura 73 – Associação das sprints ao risco

RISK-02 – Visualizar matriz probabilidade $\times$ impacto

O objetivo deste teste consiste em avaliar os mecanismos de análise visual dos riscos implementados na matriz de probabilidade $\times$ impacto. Para isso, são verificados o posicionamento automático dos riscos na matriz, a classificação por criticidade, a aplicação de filtros e buscas, bem como a interação entre a matriz e a listagem de riscos exibida na mesma página.

Para execução dos testes é utilizado o conjunto de riscos previamente cadastrado durante a avaliação da história RISK-01. Esses registros possuem diferentes combinações de probabilidade e impacto, permitindo testar o comportamento da matriz em diferentes níveis de criticidade.

O teste é executado conforme os seguintes passos:

1. Acessar o projeto de teste.
2. Acessar a página `/risks/matrix`.
3. Verificar se todos os riscos cadastrados são distribuídos automaticamente na matriz conforme seus valores de probabilidade e impacto.
4. Verificar se as células apresentam cores compatíveis com o nível de criticidade calculado.

5. Verificar se a listagem lateral apresenta todos os riscos cadastrados e suas respectivas pontuações.
6. Selecionar uma célula da matriz contendo riscos.
7. Verificar se a listagem lateral passa a exibir apenas os riscos pertencentes à célula selecionada.
8. Limpar o filtro da célula selecionada.
9. Utilizar o campo de busca para localizar riscos específicos.
10. Aplicar filtros avançados por tipo de risco.
11. Verificar se a matriz e a listagem lateral são atualizadas de acordo com os filtros aplicados.

Ao acessar a página de matriz de riscos, o sistema apresenta uma visão consolidada da distribuição dos riscos do projeto. Conforme ilustrado na Figura 74, os riscos são posicionados automaticamente em uma grade composta por cinco níveis de probabilidade e cinco níveis de impacto. Cada célula representa uma combinação possível entre esses fatores e exibe a quantidade de riscos alocados naquela posição.

Matriz de Riscos

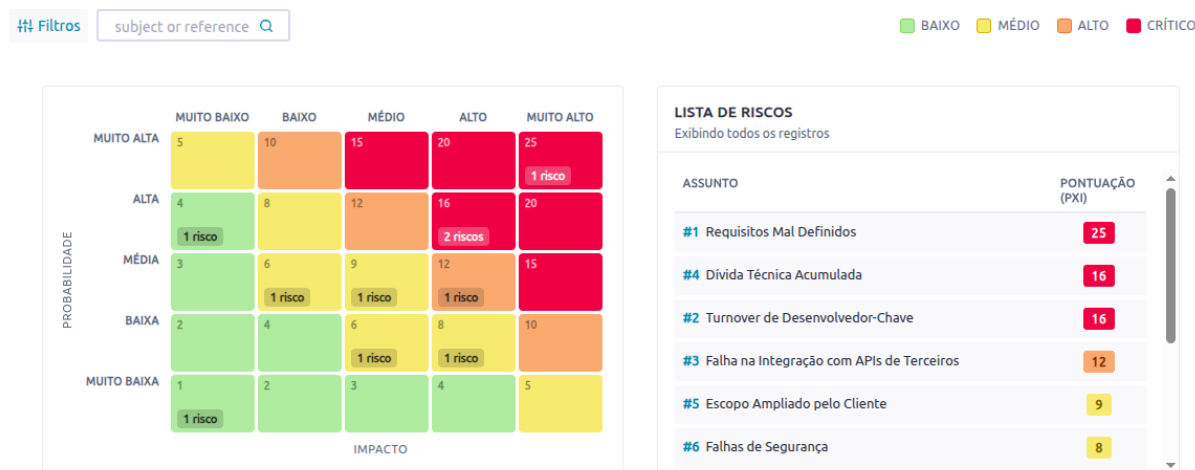


Figura 74 – Visão geral da matriz de riscos

A criticidade dos riscos é calculada automaticamente por meio da multiplicação entre probabilidade e impacto ($P \times I$). O resultado é utilizado para determinar a cor da célula correspondente, permitindo identificar visualmente riscos de baixa, média, alta e crítica severidade. Além disso, a interface apresenta uma legenda com os níveis de criticidade e uma listagem lateral contendo os riscos cadastrados e suas respectivas pontuações.

Durante os testes também foi avaliado o mecanismo de filtragem por célula. Ao selecionar uma posição específica da matriz, o sistema aplica automaticamente um filtro sobre a listagem lateral. A Figura 75 apresenta a seleção da célula correspondente à combinação de probabilidade muito alta e impacto muito alto, resultando em uma pontuação de 25. Nesse cenário, apenas o risco associado àquela posição permanece visível na listagem lateral, permitindo ao gerente analisar individualmente os registros pertencentes a uma determinada faixa de criticidade.

Risk Matrix

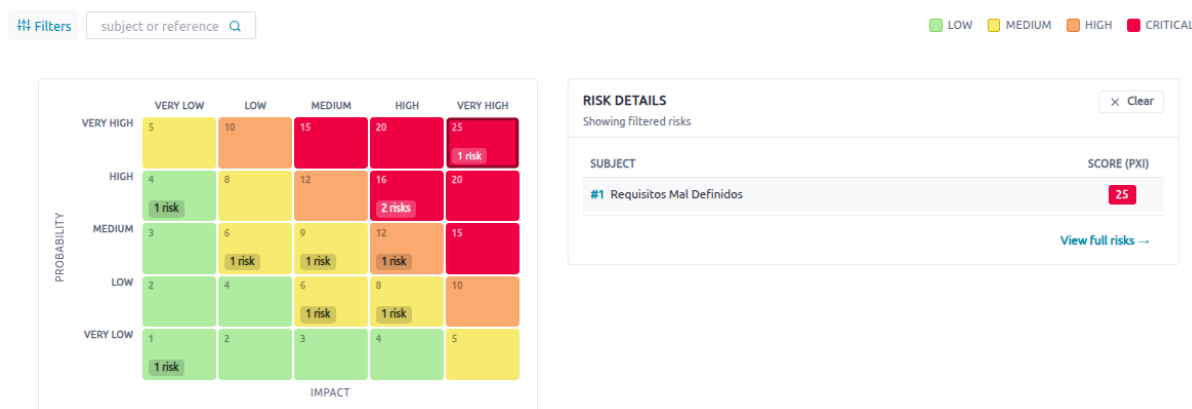


Figura 75 – Filtragem automática da listagem após seleção de uma célula da matriz

Foi verificado que a remoção do filtro restaura imediatamente a visualização completa da matriz e da listagem de riscos, sem necessidade de recarregar a página.

Outro recurso avaliado foi a busca textual. Conforme ilustrado na Figura 76, ao informar parte do título de um risco no campo de pesquisa, a interface atualiza dinamicamente os resultados exibidos. Esse comportamento facilita a localização de riscos específicos em projetos que possuam uma quantidade elevada de registros.

Além da busca textual, a funcionalidade disponibiliza filtros avançados para segmentação dos registros. Conforme apresentado na Figura 77, a interface permite filtrar riscos por categoria, tipo, status, criticidade, responsável e outros atributos disponíveis no módulo.

Durante o teste foi aplicado um filtro para exibir apenas riscos classificados como oportunidades. Ainda demonstrado na Figura 77, a matriz foi recalculada automaticamente, mantendo apenas os registros compatíveis com o critério selecionado. Da mesma forma, a listagem lateral passou a apresentar exclusivamente os riscos pertencentes ao conjunto filtrado.

Os resultados observados durante os testes demonstraram que a funcionalidade atende aos requisitos definidos para a história RISK-02. O sistema posiciona corretamente

Risk Matrix



Figura 76 – Busca textual aplicada sobre os riscos da matriz

Risk Matrix



Figura 77 – Painel de filtros avançados da matriz de riscos

os riscos na matriz, calcula automaticamente sua criticidade, permite análises visuais por meio da distribuição espacial dos registros e fornece mecanismos de busca, filtragem e navegação que auxiliam na priorização e no acompanhamento dos riscos do projeto.

RISK-03 – Acompanhar a evolução dos riscos do projeto

O objetivo deste teste consiste em avaliar os mecanismos de acompanhamento histórico dos riscos implementados no módulo de evolução de riscos. A funcionalidade permite monitorar a variação da criticidade dos riscos ao longo do tempo, registrar avalia-

ções periódicas e analisar tendências de aumento ou redução da exposição do projeto a ameaças e oportunidades.

Para execução dos testes é utilizado o conjunto de riscos previamente cadastrado durante a avaliação da história RISK-01, complementado por múltiplas avaliações realizadas em datas distintas. Dessa forma, foi possível verificar o comportamento dos gráficos evolutivos, filtros, mecanismos de busca e recursos de gerenciamento das avaliações.

O teste é executado conforme os seguintes passos:

1. Acessar o projeto de teste.
2. Acessar a página `/risks/evolution`.
3. Verificar se o gráfico apresenta a evolução histórica dos riscos avaliados.
4. Confirmar se os níveis de criticidade são representados visualmente por meio das faixas de cores do gráfico.
5. Verificar se a listagem lateral apresenta apenas riscos que possuem avaliações registradas.
6. Utilizar o campo de busca para localizar riscos específicos.
7. Aplicar filtros avançados por categoria.
8. Selecionar um risco na listagem lateral.
9. Verificar se o gráfico passa a exibir exclusivamente a evolução do risco selecionado além da lista com todas as avaliações realizadas neste risco.
10. Acessar o histórico detalhado de avaliações do risco.
11. Associar uma avaliação a uma sprint do projeto.
12. Criar uma nova avaliação.
13. Confirmar uma avaliação pendente.
14. Excluir uma avaliação existente.
15. Verificar se as alterações são refletidas automaticamente no gráfico evolutivo.

Ao acessar a página de evolução de riscos, o sistema apresenta uma visão consolidada do comportamento histórico dos riscos avaliados. Conforme ilustrado na Figura 78, cada linha representa a evolução temporal do score de um risco, calculado pela multiplicação entre probabilidade e impacto ($P \times I$).

Risk Evolution

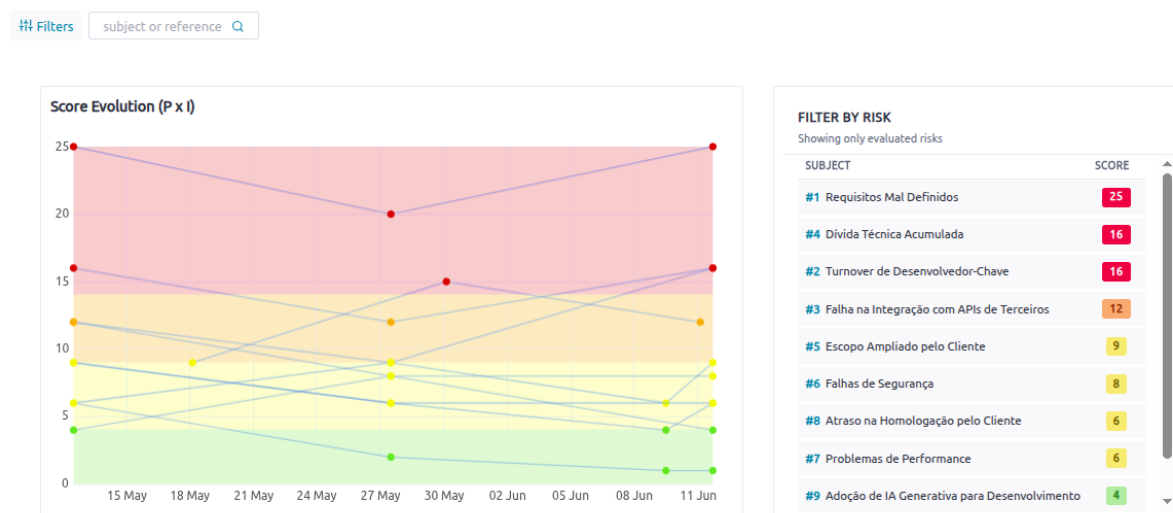


Figura 78 – Visão geral da evolução dos riscos

As faixas coloridas do gráfico representam os diferentes níveis de criticidade definidos pelo módulo de riscos, permitindo identificar visualmente períodos em que determinados riscos se encontravam classificados como baixos, médios, altos ou críticos.

Além da visualização consolidada, a funcionalidade disponibiliza mecanismos de filtragem avançada. Durante os testes foi aplicado um filtro para exibir apenas riscos pertencentes à categoria gerencial. Conforme demonstrado na Figura 79, tanto o gráfico quanto a listagem lateral foram atualizados automaticamente para refletir somente os registros compatíveis com o filtro selecionado.

Risk Evolution

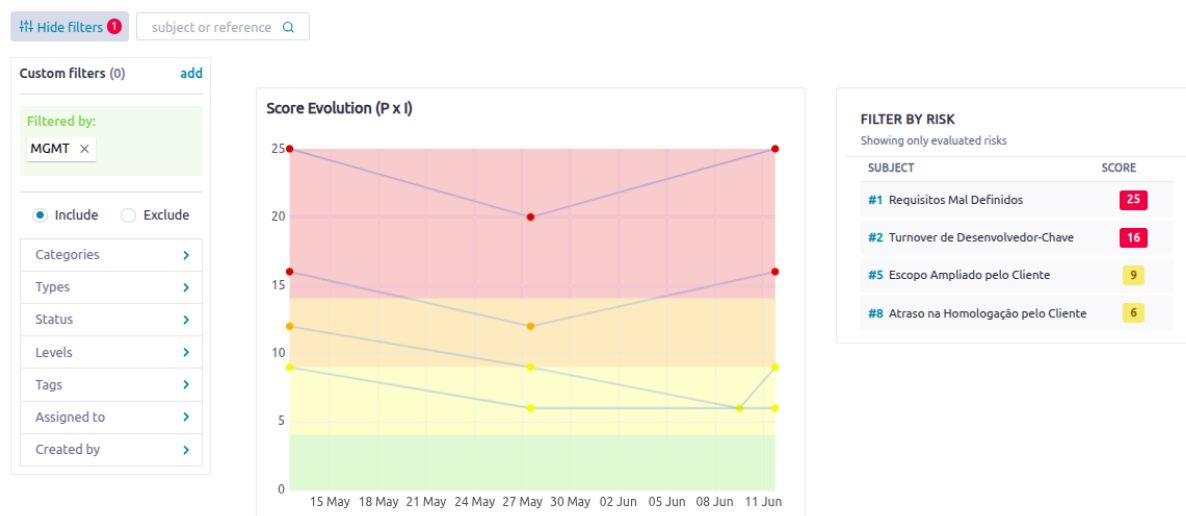
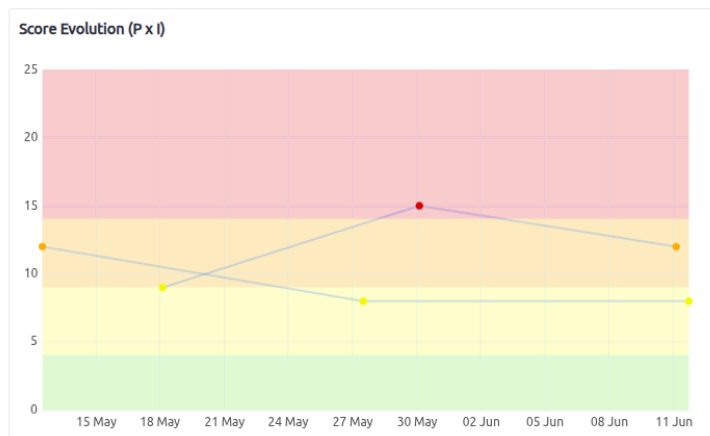


Figura 79 – Aplicação de filtros avançados na evolução dos riscos

Também foi testado o mecanismo de busca textual. Ao informar parte do título de um risco, a interface restringe os resultados exibidos no gráfico e na listagem lateral, facilitando a localização de registros específicos em projetos que possuam grande quantidade de riscos cadastrados.

Risk Evolution

Filters Falha



FILTER BY RISK	
Showing only evaluated risks	
SUBJECT	SCORE
#3 Falha na Integração com APIs de Terceiros	12
#6 Falhas de Segurança	8

Figura 80 – Busca textual aplicada na evolução dos riscos

Outra funcionalidade testada foi a análise individual de riscos. Ao selecionar um item da listagem lateral, o sistema passa a exibir exclusivamente a curva evolutiva daquele registro. Conforme apresentado na Figura 81, essa abordagem facilita a análise detalhada do comportamento de um risco específico ao longo do projeto.

Ao acessar os detalhes do risco selecionado, o sistema apresenta o histórico completo de avaliações realizadas. Conforme ilustrado na Figura 81, cada registro contém informações sobre sprint associada, probabilidade, impacto, score calculado, criticidade resultante e data da avaliação.

Durante a avaliação também foi testada a associação de avaliações às sprints do projeto. Conforme demonstrado na Figura 82, o sistema permite selecionar a sprint relacionada à avaliação, possibilitando rastrear em qual iteração determinada alteração de criticidade ocorreu.

A criação de novas avaliações também foi testada. O sistema disponibiliza uma linha de inserção onde o usuário informa os valores de probabilidade e impacto. A partir desses dados, a pontuação e o nível de criticidade são calculados automaticamente, garantindo consistência entre os registros históricos.

Além do cadastro, foi testado o processo de confirmação das avaliações registradas. Avaliações pendentes permanecem destacadas visualmente até serem confirmadas pelo responsável. Após a confirmação, o sistema passa a considerá-las oficialmente no histórico do risco.

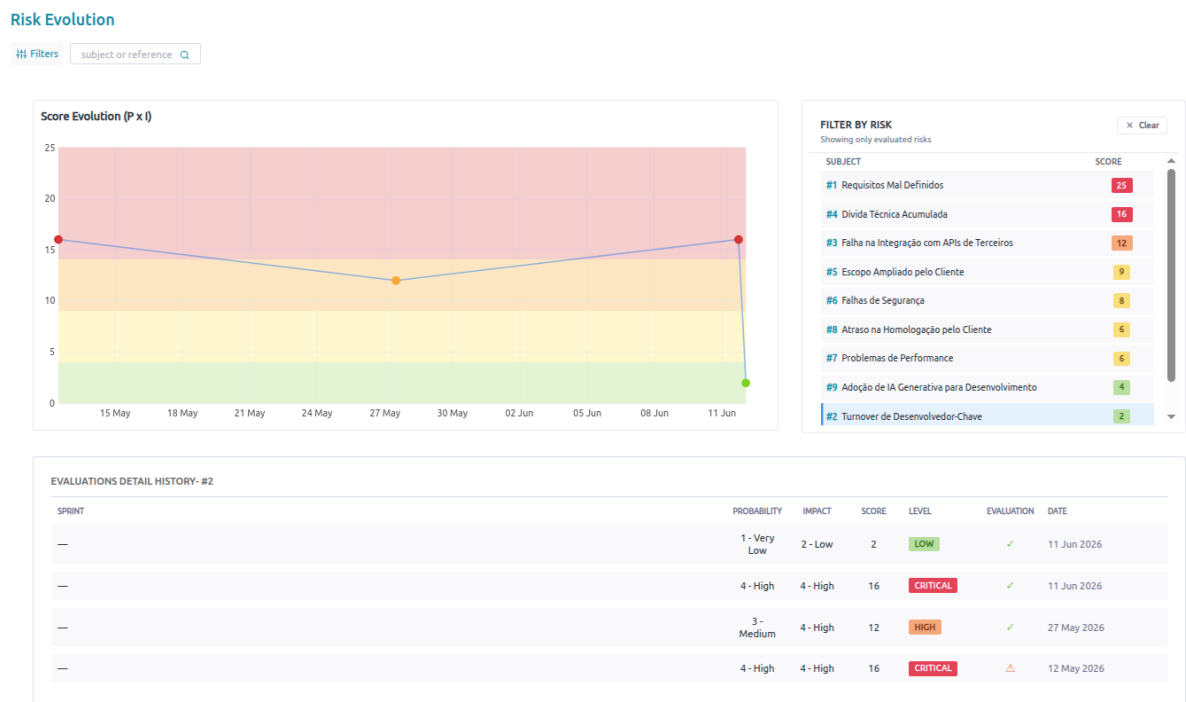


Figura 81 – Visualização individual da evolução de um risco

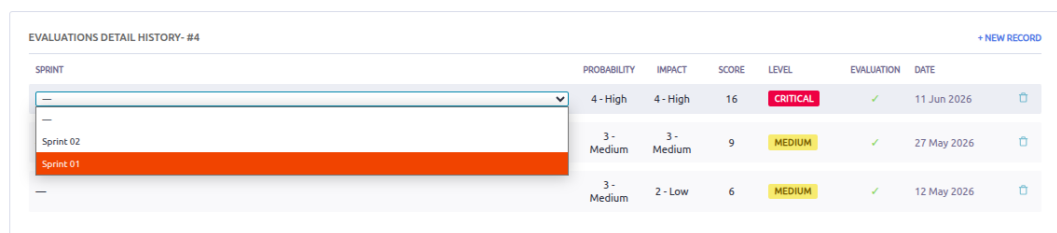


Figura 82 – Associação de avaliação a uma sprint

Por fim, foi testado o mecanismo de remoção de avaliações. Antes da exclusão, o sistema solicita confirmação explícita do usuário, evitando remoções acidentais de informações históricas importantes para a análise do risco.

Os resultados observados demonstraram que a funcionalidade atende aos requisitos definidos para a história RISK-03. O sistema registra adequadamente o histórico das avaliações, mantém a rastreabilidade das alterações realizadas ao longo do projeto e disponibiliza recursos gráficos que auxiliam na análise da evolução dos riscos e na tomada de decisões relacionadas ao seu tratamento.

5.7 MODIFICAÇÕES NO CÓDIGO ORIGINAL E MANUTENÇÃO FUTURA

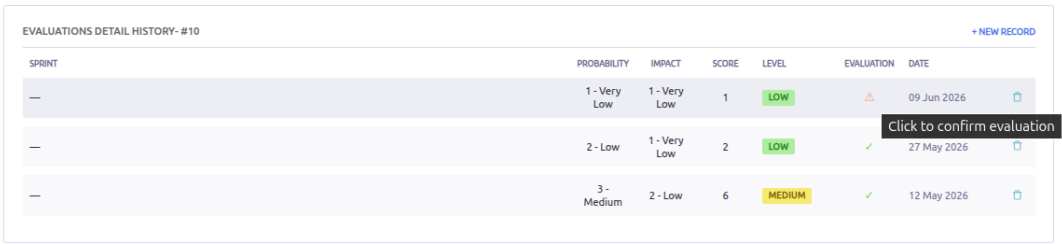
A estratégia adotada no desenvolvimento dos novos módulos buscou, sempre que possível, isolar as funcionalidades acrescentadas do código original do Taiga, de

EVALUATIONS DETAIL HISTORY- #2

+ NEW RECORD

SPRINT	PROBABILITY	IMPACT	SCORE	LEVEL	EVALUATION	DATE
—	+ Select	Select	-	LOW	⚠	11 Jun 2026
—	4 - High	4 - High	16	CRITICAL	✓	11 Jun 2026
—	3 - Medium	4 - High	12	HIGH	✓	27 May 2026
—	4 - High	4 - High	16	CRITICAL	⚠	12 May 2026

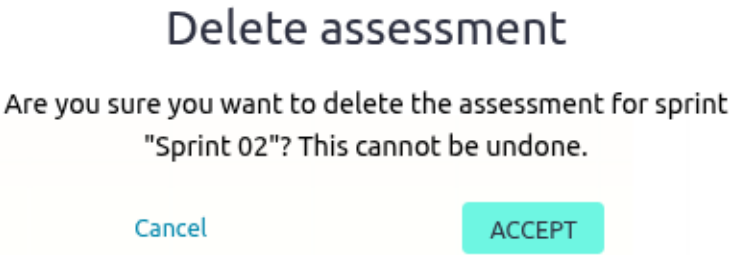
Figura 83 – Inclusão de uma nova avaliação



The screenshot shows a table titled 'EVALUATIONS DETAIL HISTORY- #10' with a '+ NEW RECORD' link in the top right. The table has columns: SPRINT, PROBABILITY, IMPACT, SCORE, LEVEL, EVALUATION, and DATE. There are three rows of data: (1 - Very Low, 1 - Very Low, 1, LOW, warning triangle, 09 Jun 2026), (2 - Low, 1 - Very Low, 2, LOW, green checkmark, 27 May 2026), and (3 - Medium, 2 - Low, 6, MEDIUM, green checkmark, 12 May 2026). Each row has a trash icon in the DATE column. A tooltip 'Click to confirm evaluation' is visible over the green checkmark in the second row.

SPRINT	PROBABILITY	IMPACT	SCORE	LEVEL	EVALUATION	DATE
—	1 - Very Low	1 - Very Low	1	LOW	⚠	09 Jun 2026
—	2 - Low	1 - Very Low	2	LOW	✓	27 May 2026
—	3 - Medium	2 - Low	6	MEDIUM	✓	12 May 2026

Figura 84 – Confirmação de avaliação pendente



The dialog has a title 'Delete assessment' in bold. Below it is the text 'Are you sure you want to delete the assessment for sprint "Sprint 02"? This cannot be undone.' At the bottom are two buttons: 'Cancel' in blue and 'ACCEPT' in a red button.

Delete assessment

Are you sure you want to delete the assessment for sprint "Sprint 02"? This cannot be undone.

Cancel ACCEPT

Figura 85 – Confirmação para exclusão de avaliação

modo a reduzir a superfície de conflito em futuras atualizações do repositório oficial. Cada épico foi implementado como uma unidade independente: no back-end, por meio de aplicativos Django próprios (`taiga.projects.schedule`, para o cronograma, `taiga.projects.finances`, para o módulo financeiro e `taiga.projects.risks`, para o módulo de riscos); no front-end, por meio de módulos e telas dedicados, sem reaproveitar diretamente os arquivos das funcionalidades nativas. A integração com o núcleo foi feita, preferencialmente, por mecanismos que não exigem editar arquivos originais, como sinais (*signals*), gatilhos de banco de dados (*triggers*) e *endpoints* dedicados.

Ainda assim, um conjunto residual de arquivos originais precisou ser alterado, e é justamente esse conjunto que constitui os pontos de atenção em manutenções futuras. Cabe observar que vários arquivos originais (registro de aplicativos, permissões, modelo de projeto, navegação e arquivos de tradução) são alterados por ambos os épicos, criando assim uma interseção de arquivos alterados.

Os arquivos originais do Taiga modificados em cada épico são relacionados a seguir, separados por camada (back-end e front-end). A justificativa individual de cada alteração, acompanhada do trecho de código correspondente e dos pontos que demandam atenção durante a integração de novas versões, encontra-se no Apêndice B.

- **Épico CRON (Cronograma)**

- **Back-end**

- * settings/common.py
 - * taiga/permissions/choices.py
 - * taiga/projects/admin.py
 - * taiga/projects/epics/api.py
 - * taiga/projects/epics/apps.py
 - * taiga/projects/epics/services.py
 - * taiga/projects/history/freeze_impl.py
 - * taiga/projects/models.py
 - * taiga/projects/serializers.py
 - * taiga/routers.py

- **Front-end**

- * app/coffee/app.coffee
 - * app/coffee/modules/admin/roles.coffee
 - * app/coffee/modules/base.coffee
 - * app/coffee/modules/resources.coffee
 - * app/locales/taiga/locale-en.json
 - * app/locales/taiga/locale-pt-br.json
 - * app/partials/admin/admin-project-modules.jade
 - * app/svg/sprite.svg
 - * elements.js
 - * gulpfile.js

- **Front-end-Next**

- * src/app/api/projects/projects.model.mock.ts
 - * src/app/api/projects/projects.model.ts
 - * src/app/api/roles/roles.model.ts
 - * src/app/commons/components/svg-sprite
/svg-sprite.component.html
 - * src/app/commons/project-navigation
/project-navigation.component.css

- * src/app/commons/project-navigation
/project-navigation.component.html
- * src/app/commons/project-navigation
/project-navigation.component.spec.ts
- * src/app/commons/project-navigation
/project-navigation.component.ts
- * src/assets/i18n/en.json

- **Épico CUST (Gestão Financeira)**

- **Back-end**

- * settings/common.py
- * taiga/locale/pt_BR/LC_MESSAGES/django.po
- * taiga/permissions/choices.py
- * taiga/projects/admin.py
- * taiga/projects/fixtures/initial_project_templates.json
- * taiga/projects/history/api.py
- * taiga/projects/history/freeze_impl.py
- * taiga/projects/history/permissions.py
- * taiga/projects/history/services.py
- * taiga/projects/models.py
- * taiga/projects/serializers.py
- * taiga/routers.py
- * taiga/timeline/service.py
- * taiga/timeline/timeline_implementations.py

- **Front-end**

- * app/coffee/app.coffee
- * app/coffee/modules/admin/roles.coffee
- * app/coffee/modules/base.coffee
- * app/coffee/modules/resources.coffee
- * app/locales/taiga/locale-en.json
- * app/locales/taiga/locale-pt-br.json
- * app/partials/admin/admin-project-modules.jade
- * app/styles/modules/admin/admin-functionalities.scss
- * app/styles/modules/admin/admin-roles.scss
- * elements.js

- * gulpfile.js
- * package-lock.json
- * package.json

– **Front-end-Next**

- * src/app/api/project-templates/project-templates.model.mock.ts
- * src/app/api/project-templates/project-templates.model.ts
- * src/app/api/projects/projects.model.mock.ts
- * src/app/api/projects/projects.model.ts
- * src/app/api/roles/roles.model.ts
- * src/app/commons/project-navigation
/project-navigation.component.css
- * src/app/commons/project-navigation
/project-navigation.component.html
- * src/app/commons/project-navigation
/project-navigation.component.spec.ts
- * src/app/commons/project-navigation
/project-navigation.component.ts
- * src/assets/i18n/en.json

- **Épico RISK (Gestão de Riscos)**

– **Back-end**

- * settings/common.py
- * taiga/permissions/choices.py
- * taiga/base/filters.py
- * taiga/projects/attachments/api.py
- * taiga/projects/attachments/permissions.py
- * taiga/projects/fixtures/initial_project_templates.json
- * taiga/projects/history/api.py
- * taiga/projects/history/freeze_impl.py
- * taiga/projects/history/permissions.py
- * taiga/projects/history/services.py
- * taiga/projects/models.py
- * taiga/projects/references/models.py
- * taiga/projects/serializers.py
- * taiga/routers.py

– Front-end

- * app/coffee/app.coffee
- * app/coffee/modules/admin/roles.coffee
- * app/coffee/modules/base.coffee
- * app/coffee/modules/resources.coffee
- * app/coffee/modules/common/components.coffee
- * app/coffee/modules/common/sections.coffee
- * app/coffee/modules/taskboard/main.coffee
- * app/locales/taiga/locale-en.json
- * app/locales/taiga/locale-pt-br.json
- * app/modules/components/project-menu/project-menu.controller.coffee
- * app/modules/components/wysiwyg/item-wysiwyg.directive.coffee
- * app/modules/history/history/history-diff.jade
- * app/modules/services/attachments.service.coffee
- * app/partials/admin/admin-project-modules.jade
- * app/styles/dependencies/mixins/lightbox.scss
- * app/styles/extras/dependencies.scss
- * app/styles/modules/common/lightbox.scss
- * app/styles/modules/common/ticket-data.scss

– Front-end-Next

- * src/app/api/projects/projects.model.ts
- * src/app/api/roles/roles.model.ts
- * src/app/commons/components/svg-sprite/svg-sprite.component.html
- * src/app/commons/legacy/legacy-loader.component.html
- * src/app/commons/legacy/legacy.module.ts
- * src/app/commons/project-navigation/project-navigation.component.html
- * src/app/commons/project-navigation/project-navigation.component.ts
- * src/app/webcomponent.module.ts

6 CONCLUSÃO

6.1 CONSIDERAÇÕES FINAIS

Este Trabalho de Conclusão de Curso apresenta a evolução de uma ferramenta open source de gerenciamento ágil de projetos, tendo o Taiga.io como objeto de estudo e aprimoramento. A pesquisa parte da constatação de que ferramentas ágeis oferecem bom suporte a práticas como backlog, histórias de usuário e sprints, mas nem sempre contemplam, de forma integrada, práticas prescritivas de gerenciamento de projetos, como gestão de riscos, análise de cronograma e controle orçamentário. Nesse contexto, o trabalho amplia o Taiga.io com funcionalidades voltadas à gestão de cronograma, gestão financeira e gestão de riscos, aproximando a ferramenta de práticas consolidadas de gerenciamento sem descaracterizar sua natureza ágil e open source.

A análise de ferramentas de gerenciamento de projetos permitiu compreender como diferentes soluções tratam riscos, cronograma e orçamento. A comparação entre Taiga.io, OpenProject, Monday, ProjectManager, Jira, Artia, Smartsheet e Zoho Projects evidenciou diferentes níveis de suporte a essas dimensões, mas também mostrou que nem sempre elas aparecem de forma integrada, gratuita e compatível com implantação própria. Esse estudo permitiu identificar as lacunas funcionais existentes, orientar a definição dos módulos desenvolvidos e fundamentar a escolha da ferramenta utilizada como objeto de evolução.

A partir dessa análise, o Taiga.io foi selecionado por reunir características adequadas ao escopo do trabalho, como natureza open source, aderência a métodos ágeis e possibilidade de implantação on-premise. Sua arquitetura também se mostrou apropriada para extensão, permitindo incorporar funcionalidades tradicionalmente associadas a abordagens mais estruturadas de gerenciamento de projetos.

No eixo de cronograma, foi desenvolvido um módulo que permite estruturar atividades, representar dependências e visualizar o planejamento temporal por meio de um gráfico de Gantt. A implementação também contempla a identificação do caminho crítico e o cálculo da folga total das atividades, permitindo compreender quais tarefas impactam diretamente a duração mínima do projeto e quais possuem margem de atraso, atendendo ao objetivo de estruturar atividades, representar dependências e calcular a duração mínima do projeto por meio da identificação do caminho crítico.

No eixo financeiro, foi implementado um módulo de gestão orçamentária que permite cadastrar custos planejados, configurar valores por hora dos integrantes, registrar horas trabalhadas, calcular custos realizados, cadastrar receitas e consolidar indicadores financeiros em um dashboard. A solução possibilita acompanhar custos, receitas, saldo financeiro e distribuição de custos por categorias, reduzindo a necessidade de controles externos ao Taiga.io.

No eixo de gestão de riscos, foi implementado um módulo para registrar, analisar e acompanhar riscos do projeto. A solução contempla informações como probabilidade,

impacto, causa, consequência, responsável, estratégia de mitigação e plano de resposta, além de visualizações de apoio, como a matriz de probabilidade e impacto e a evolução histórica dos riscos. Com isso, o Taiga.io passa a oferecer suporte à análise qualitativa e ao acompanhamento de riscos ao longo do projeto.

Outro aspecto considerado no desenvolvimento foi a tentativa de manter a implementação o mais desacoplada possível do núcleo original do Taiga.io, a fim de reduzir conflitos em futuras atualizações da ferramenta. Embora parte das funcionalidades tenha sido organizada em módulos próprios e telas dedicadas, a integração completa com os fluxos nativos do sistema exigiu alterações em arquivos centrais do back-end, do front-end e do modelo de dados. Dessa forma, esse objetivo não pôde ser plenamente atendido. Para amenizar essa limitação, foi elaborada uma documentação detalhada no Apêndice B, registrando os arquivos modificados, as justificativas de cada alteração e os pontos de atenção para futuras integrações com novas versões do Taiga.io.

Dessa forma, considera-se que o objetivo geral deste trabalho foi alcançado, ao evoluir o Taiga.io com funcionalidades que ampliam seu suporte à gestão de projetos. A solução integra práticas de gestão de riscos, cronograma e orçamento ao ambiente ágil da ferramenta, mantendo sua proposta open source e ampliando sua aplicabilidade em contextos acadêmicos e profissionais.

Além dos resultados obtidos em termos funcionais, o trabalho também reforça o potencial do Taiga.io como ferramenta de apoio ao ensino de gerenciamento de projetos. A incorporação de conceitos relacionados a cronograma, custos e riscos em uma plataforma livre e aderente a métodos ágeis possibilita que esses temas sejam explorados de forma integrada em atividades acadêmicas, reduzindo a necessidade de utilização de múltiplas ferramentas para abordar diferentes aspectos da gerência de projetos. Dessa forma, a solução amplia as possibilidades de utilização da plataforma tanto em contextos profissionais quanto em disciplinas voltadas ao ensino de gerenciamento de projetos e engenharia de software.

6.2 TRABALHOS FUTUROS

Como continuidade deste trabalho, propõe-se a realização de testes em ambiente acadêmico, especialmente em disciplinas de gerência de projetos ou engenharia de software. Essa avaliação permitiria observar o uso das funcionalidades em cenários reais ou simulados de planejamento, acompanhamento e controle de projetos, fornecendo evidências sobre sua utilidade pedagógica.

Outra possibilidade de evolução consiste em aprimorar a integração entre os módulos desenvolvidos. A relação entre riscos, cronograma e custos pode ser explorada de forma mais aprofundada, permitindo associar riscos a atividades críticas, estimar impactos financeiros decorrentes de eventos de risco e visualizar como alterações no cronograma influenciam custos planejados e realizados. No contexto da gestão de riscos, também podem

ser incorporadas funcionalidades complementares abordadas na fundamentação teórica deste trabalho, como mecanismos de apoio à análise SWOT e suporte à aplicação da técnica Delphi, ampliando as possibilidades de identificação, avaliação e acompanhamento dos riscos do projeto.

Além disso, trabalhos futuros podem investigar melhorias de usabilidade nas novas interfaces. A aplicação de avaliações com usuários permitiria identificar dificuldades de navegação, compreensão dos gráficos e clareza das informações apresentadas, contribuindo para o refinamento das telas de Gantt, dashboard financeiro, matriz de riscos e evolução dos riscos.

Por fim, recomenda-se a continuidade da manutenção técnica das funcionalidades implementadas, com revisão dos arquivos modificados, criação de documentação para desenvolvedores, ampliação da cobertura de testes automatizados e adaptação das alterações a futuras versões do Taiga.io.

REFERÊNCIAS

ATLASSIAN. **About Us**. 2026. Disponível em: <https://www.atlassian.com/company>. Acesso em: 10 jul. 2026.

ATLASSIAN. **Jira: Project Management for the AI Era**. 2026. Disponível em: <https://www.atlassian.com/software/jira>. Acesso em: 10 jul. 2026.

BARBOSA, Christina et al. **Gerenciamento de custos em projetos**. 6. ed. Rio de Janeiro: FGV Management, 2019.

BARCAUI, André. **Gerenciamento do tempo em projetos**. Rio de Janeiro: FGV, 2012.

BIESIALSKA, Katarzyna; FRANCH, Xavier; MUNTÉS-MULERO, Victor. Mining Dependencies in Large-Scale Agile Software Development Projects: A Quantitative Industry Study. In: PROCEEDINGS of the 25th International Conference on Evaluation and Assessment in Software Engineering. [S.l.]: Association for Computing Machinery, 2021. p. 20–29. DOI: 10.1145/3463274.3463323.

BLOSS, R. et al. **Critical Path Management of the Mining Value Chain Using a Probabilistic Approach**. Australia, 2025.

CHESNEAU, Benoit. 2024. Disponível em: <https://docs.gunicorn.org/en/latest/index.html>.

COFFEESCRIPT.ORG. **Documentação CoffeScript**. Disponível em: <https://coffeescript.org/>.

DINGSØYR, Torgeir; PETIT, Michael. **Managing Layers of Risk: Uncertainty in Large Development Programs Combining Agile Software Development and Traditional Project Management**. Noruega: De Gruyter, 2021.

DOCKER INC. **O que é Docker**. 2025. Disponível em: <https://docs.docker.com/get-started/docker-overview/>.

EBY, Phillip J. 2010. Disponível em: <https://peps.python.org/pep-3333/>. Acesso em: 25 set. 2025.

FOUNDATION, Django Software. **Django Documentation**. 2025. Disponível em: <https://docs.djangoproject.com/en/stable/>. Acesso em: 25 set. 2025.

JAHR, Michael. A Hybrid Approach to Quantitative Software Project Scheduling within Agile Frameworks. **Project Management Journal**, v. 45, n. 3, p. 35–45, 2014. DOI: 10.1002/pmj.21411.

MDN WEB DOCS. **Document Object Model (DOM) — Overview and Guides**. 2025. Disponível em:

https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model. Acesso em: 10 set. 2025.

MDN WEB DOCS. **HTML: HyperText Markup Language**. 2025. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTML>. Acesso em: 10 set. 2025.

MDN WEB DOCS. **HTTP: Hypertext Transfer Protocol**. 2025. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTTP>. Acesso em: 10 set. 2025.

MDN WEB DOCS. **Introduction to client-side frameworks**. 2025. Disponível em: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Frameworks_libraries/Introduction. Acesso em: 25 set. 2025.

MDN WEB DOCS. **Introduction to the server side**. 2025. Disponível em: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/First_steps/Introduction. Acesso em: 10 set. 2025.

MDN WEB DOCS. **Introduction to web APIs**. 2025. Disponível em: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Client-side_APIs/Introduction. Acesso em: 10 set. 2025.

MDN WEB DOCS. **JavaScript — Guide and Overview**. 2025. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Acesso em: 10 set. 2025.

MDN WEB DOCS. **Learn web development**. Front-end skill set overview. 2025. Disponível em: https://developer.mozilla.org/en-US/docs/Learn_web_development. Acesso em: 10 set. 2025.

MICROSOFT. **Azure - O que são bancos de dados?** 2025. Disponível em: <https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-are-databases>.

NPMJS. **Sobre o NPM**. Disponível em: <https://docs.npmjs.com/about-npm>.

OPENJS FOUNDATION. **Sobre o Node.js**. Disponível em: <https://nodejs.org/pt/about>.

OPENPROJECT. **Página do OpenProject**. Disponível em: <https://www.openproject.org/pt/>.

PROJECT MANAGEMENT INSTITUTE. **Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK®)**. 6. ed. Newtown Square, PA: [s.n.], 2017.

PUGJS.ORG. **Documentação PugJS**. Disponível em:
<https://pugjs.org/api/getting-started.html>.

PYTHON SOFTWARE FOUNDATION. **What is Python? Executive Summary**. 2025. Disponível em: <https://www.python.org/doc/essays/blurbs/>. Acesso em: 10 set. 2025.

REIFF, Janine; SCHLEGEL, Dennis. Hybrid Project Management – A Systematic Literature Review. **International Journal of Information Systems and Project Management**, v. 10, n. 2, p. 45–63, 2022. DOI: 10.12821/ijispm100203.

SALLES JR., Carlos Alberto Corrêa. **Gerenciamento de riscos em projetos**. Rio de Janeiro: FGV, 2010.

SANITY. **Front-end — Glossary**. Ago. 2024. Disponível em:
<https://www.sanity.io/glossary/front-end>. Acesso em: 10 set. 2025.

SOLEM, Ask. **Documentação Celery**. 2023. Disponível em:
<https://docs.celeryq.dev/en/v5.5.3/index.html>.

STRAY, Viktoria; MOE, Nils Brede; AASHEIM, Andreas. Dependency Management in Large-Scale Agile: A Case Study of DevOps Teams. In: PROCEEDINGS of the 52nd Hawaii International Conference on System Sciences. [S.l.: s.n.], 2019. p. 7007–7016. DOI: 10.24251/HICSS.2019.840.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. **Documentação Postgres**. 2025. Disponível em: <https://www.postgresql.org/docs/current/>.

VIERLBOECK, M. et al. Budgeting for Agile Product Development. In: PROCEEDINGS of the 22nd International Conference on Engineering Design (ICED19). Delft: Cambridge University Press, 2019.

W3C. **CSS — Cascading Style Sheets (Overview)**. 2025. Disponível em:
<https://www.w3.org/Style/CSS/Overview.en.html>. Acesso em: 10 set. 2025.

APÊNDICE A – ANÁLISE RESTANTE DAS HISTÓRIAS DE USUÁRIO

Histórias do Épico E1 – Gestão de Cronograma

História CRON-02 – Identificar caminho crítico no gráfico de Gantt

Descrição da história:

Eu, como gerente de projetos, desejo identificar visualmente o caminho crítico do projeto no gráfico de Gantt, para compreender quais atividades não possuem folga e que, se atrasarem, impactarão diretamente a data de término do projeto.

Testes de aceitação:

- Dado que o cronograma do projeto possua atividades com dependências definidas, quando o gerente acessar o gráfico de Gantt e habilitar a opção de exibir o caminho crítico, então o sistema deve destacar, de forma visualmente diferenciada (por exemplo, por cor ou estilo de linha), todas as atividades que compõem o caminho crítico.
- Dado que o gerente altere a duração ou as datas de início/término de uma atividade relevante para o caminho crítico, quando o gráfico de Gantt for recalculado, então o conjunto de atividades destacadas como caminho crítico deve ser atualizado automaticamente, refletindo o novo cenário.

História CRON-03 – Visualizar atraso permitido das atividades

Descrição da história:

Eu, como gerente de projetos, desejo visualizar quanto uma atividade pode atrasar sem impactar a data de término do projeto, para avaliar o impacto de possíveis atrasos e priorizar ações de mitigação.

Testes de aceitação:

- Dado que o cronograma do projeto possua dependências entre atividades, quando o gerente visualizar o gráfico de Gantt, então cada atividade deve exibir, em um campo ou indicador, o valor de folga calculado com base nas datas mais cedo e mais tarde de início e término.
- Dado que o gerente altere a duração ou as datas de uma atividade, quando o sistema recalcular o cronograma, então os valores de folga das atividades relacionadas devem ser atualizados automaticamente, refletindo o novo cenário.

Histórias do Épico E2 – Gestão Financeira

História CUST-01 – Cadastrar custo planejado de uma tarefa

Descrição da história:

Eu, como gerente de projetos, desejo cadastrar o custo planejado de cada tarefa, para que o sistema consolide o orçamento total do projeto e permita a comparação entre

valores planejados e realizados.

Testes de aceitação:

- Dado que o gerente esteja visualizando o detalhamento de uma tarefa, quando ele informar um valor de custo planejado e salvar a tarefa, então o valor deve ser armazenado e exibido novamente ao reabrir o detalhamento dessa tarefa.
- Dado que existam múltiplas tarefas com custo planejado definido, quando o gerente acessar a página de resumo financeiro do projeto, então o sistema deve apresentar o somatório dos custos planejados das tarefas como orçamento planejado do projeto.

História CUST-03 – Definir receita contratada do projeto

Descrição da história:

Eu, como gerente de projetos, desejo definir o valor de receita contratada do projeto (orçamento ou valor vendido), para que o sistema utilize esse valor como referência na análise de margem e na comparação com os custos planejados e realizados.

Testes de aceitação:

- Dado que o gerente esteja na página de configurações ou detalhes do projeto, quando ele informar um valor de receita contratada e salvar as alterações, então o sistema deve armazenar esse valor e apresentá-lo novamente ao reabrir os detalhes do projeto.
- Dado que exista um valor de receita contratada definido, quando o gerente acessar a página de resumo financeiro do projeto, então o sistema deve exibir esse valor como receita de referência nas métricas financeiras (por exemplo, na margem do projeto).

História CUST-04 – Visualizar resumo financeiro do projeto

Descrição da história:

Eu, como gerente de projetos, desejo visualizar em uma única página o resumo financeiro do projeto, incluindo custos planejados, custos realizados, receita e margem, para acompanhar a saúde financeira do projeto de forma rápida.

Testes de aceitação:

- Dado que existam tarefas com custo planejado e horas registradas pelos integrantes, quando o gerente acessar a página de resumo financeiro, então o sistema deve exibir, no mínimo, os valores de custo planejado total, custo realizado total, receita cadastrada e margem (receita menos custo realizado).
- Dado que novos registros de horas ou ajustes de custo planejado sejam efetuados, quando a página de resumo financeiro for atualizada, então os valores exibidos devem refletir os dados mais recentes, sem necessidade de cálculos manuais externos.

Histórias do Épico E3 – Gestão de Riscos

História RISK-01 – Gerenciar registro de riscos

Descrição da história:

Eu, como gerente de projetos, desejo cadastrar, consultar, editar e remover riscos em um registro de riscos associado ao projeto, informando probabilidade, impacto, causa, consequência, responsável, estratégia de mitigação e plano de resposta, para garantir que os principais riscos estejam documentados e possam ser monitorados ao longo do projeto.

Testes de aceitação: - Dado que o gerente esteja na seção de riscos do projeto, quando ele criar um novo registro de risco informando, no mínimo, o título do risco, então o sistema deve salvar o registro e exibi-lo na lista de riscos do projeto.

- Dado que exista um risco previamente cadastrado, quando o gerente alterar suas informações e salvar as alterações, então o sistema deve atualizar os dados exibidos na listagem e nos detalhes do risco.

- Dado que exista um risco cadastrado, quando o gerente solicitar sua exclusão e confirmar a operação, então o risco deve ser removido da lista de riscos do projeto. Dado que existam múltiplos riscos cadastrados, quando o gerente ordenar a lista por qualquer um dos campos, então o sistema deve reorganizar a listagem conforme o critério selecionado.

História RISK-03 – Acompanhar a evolução dos riscos do projeto

Descrição da história:

Eu, como gerente de projetos, desejo acompanhar a evolução dos riscos ao longo do tempo, tanto individualmente quanto de forma consolidada no projeto, para identificar tendências, avaliar a efetividade das ações de mitigação e apoiar a tomada de decisão baseada no histórico dos riscos.

Testes de aceitação:

- Dado que existam avaliações históricas de um risco, quando o gerente acessar os detalhes desse risco, então o sistema deve apresentar um gráfico contendo sua evolução ao longo do tempo.

- Dado que existam avaliações históricas dos riscos do projeto, quando o gerente acessar o menu de evolução de riscos, então o sistema deve apresentar um gráfico consolidado contendo a evolução dos riscos ao longo do tempo.

- Dado que o gerente selecione um ou mais riscos específicos no painel de evolução, quando o gráfico for atualizado, então o sistema deve exibir apenas as séries correspondentes aos riscos selecionados.

- Dado que um risco tenha sua probabilidade ou impacto alterado, quando uma nova avaliação histórica for registrada, então os gráficos de evolução devem refletir a nova situação do projeto.

APÊNDICE B – DETALHAMENTO DAS MODIFICAÇÕES NO CÓDIGO ORIGINAL

Este apêndice detalha, arquivo a arquivo, as modificações realizadas em arquivos originais do Taiga durante o desenvolvimento dos módulos de cronograma (épico CRON) e de gestão financeira (épico CUST), relacionados em síntese na Seção 5.7. As entradas estão organizadas por épico e, dentro de cada épico, por camada (back-end e front-end). Para cada arquivo, indica-se a localização da alteração (o escopo global do arquivo, uma classe ou uma função), o trecho de código correspondente, a justificativa da mudança e os pontos que demandam atenção na integração de versões futuras do repositório oficial.

B.1 ÉPICO E1 - GESTÃO DE CRONOGRAMA

B.1.1 Back-end

`settings/common.py`

O arquivo `settings/common.py` reúne as configurações globais do back-end do Taiga, entre elas a lista `INSTALLED_APPS`, que registra os aplicativos Django reconhecidos pelo projeto. A única modificação realizada nesse arquivo foi a inclusão do novo aplicativo `taiga.projects.schedule` nessa lista, conforme o Código B.1.

```
INSTALLED_APPS = [  
    # ...  
    "taiga.projects.epics",  
    "taiga.projects.userstories",  
    "taiga.projects.tasks",  
    "taiga.projects.schedule",  
    "taiga.projects.issues",  
    "taiga.projects.wiki",  
    # ...  
]
```

Código B.1 – Registro do aplicativo de cronograma em `INSTALLED_APPS`

A alteração situa-se no escopo global do arquivo, no interior da lista `INSTALLED_APPS`, e consiste em uma única linha acrescentada. Trata-se de uma modificação inevitável no código original, pois o módulo de cronograma foi implementado como um aplicativo Django independente (`taiga.projects.schedule`); para que o framework reconheça seus modelos, execute suas migrações e conecte os sinais registrados em seu `AppConfig`, o aplicativo precisa constar em `INSTALLED_APPS`, que é um arquivo mantido pelo repositório original.

Em atualizações futuras, a `INSTALLED_APPS` é um ponto provável de modificação por parte do projeto oficial, uma vez que o acréscimo ou a remoção de aplicativos altera

essa lista. Por isso, ao integrar novas versões do Taiga, deve-se verificar se a linha referente ao aplicativo `taiga.projects.schedule` permanece presente, já que sua ausência desativa por completo o módulo de cronograma.

`taiga/permissions/choices.py`

O arquivo `taiga/permissions/choices.py` centraliza a definição de todas as permissões do Taiga, organizadas em listas que associam o codinome (*codename*) de cada permissão a um rótulo legível. Duas dessas listas foram alteradas para acomodar o módulo de cronograma: `ANON_PERMISSIONS`, que define as permissões disponíveis para usuários anônimos em projetos públicos, e `MEMBERS_PERMISSIONS`, que define as permissões atribuíveis aos papéis (*roles*) dos membros de um projeto. As inclusões aparecem no Código B.2.

```
1 ANON_PERMISSIONS = [  
2     # ...  
3     ('view_issues', _('View issues')),  
4     ('view_schedule', _('View schedule')),  
5     ('view_gantt', _('View gantt')),  
6     ('view_wiki_pages', _('View wiki pages')),  
7     # ...  
8 ]  
9  
10 MEMBERS_PERMISSIONS = [  
11     # ...  
12     ('delete_issue', _('Delete issue')),  
13     # Schedule / Gantt permissions  
14     ('view_schedule', _('View schedule')),  
15     ('view_gantt', _('View gantt')),  
16     ('modify_schedule_links', _('Modify schedule links')),  
17     ('modify_schedule_dates', _('Modify schedule expected start dates')),  
18     ('modify_schedule_color', _('Modify schedule color')),  
19     ('modify_gantt_list_order', _('Modify gantt list order')),  
20     # Wiki page permissions  
21     ('view_wiki_pages', _('View wiki pages')),  
22     # ...  
23 ]
```

Código B.2 – Permissões de cronograma e gantt acrescentadas em `ANON_PERMISSIONS` e `MEMBERS_PERMISSIONS`

As alterações situam-se no escopo global do arquivo, no interior dessas duas listas. Em `ANON_PERMISSIONS` foram acrescentadas as permissões de visualização `view_schedule`

e `view_gantt`. Em `MEMBERS_PERMISSIONS` foi acrescentado um bloco com seis permissões: as duas de visualização (`view_schedule` e `view_gantt`) e quatro de modificação granular (`modify_schedule_links`, `modify_schedule_dates`, `modify_schedule_color` e `modify_gantt_list_order`).

A modificação é inevitável no código original porque o sistema de permissões do Taiga é centralizado: o back-end autoriza o acesso comparando o codinome verificado em suas APIs com os codinomes declarados nessas listas, e a interface de administração de papéis monta seus controles a partir delas. Assim, para que as permissões do cronograma e do gantt possam ser concedidas aos papéis e reconhecidas em projetos públicos, elas precisam estar declaradas neste arquivo, que é mantido pelo repositório original.

Em atualizações futuras, `ANON_PERMISSIONS` e `MEMBERS_PERMISSIONS` são pontos prováveis de modificação por parte do projeto oficial, pois cada nova funcionalidade do Taiga tende a acrescentar suas próprias permissões a essas listas, o que as torna focos recorrentes de conflito de integração. Ao incorporar novas versões, deve-se verificar se as entradas de cronograma e gantt permanecem presentes nas duas listas, já que sua ausência impede a atribuição das permissões e, por consequência, o acesso ao módulo. Além disso, os codinomes aqui declarados precisam permanecer idênticos aos verificados nas APIs do back-end e na interface, sob risco de o controle de acesso deixar de funcionar de forma silenciosa.

`taiga/projects/admin.py`

O arquivo `taiga/projects/admin.py` configura a apresentação dos modelos do aplicativo `projects` na interface administrativa do Django (*Django Admin*). A classe `ProjectAdmin` define, por meio do atributo `fieldsets`, como os campos do modelo `Project` são agrupados e exibidos no formulário de edição. A modificação acrescentou o campo `is_schedule_activated` ao grupo *Modules*, que reúne as *flags* de ativação dos módulos do projeto, conforme o Código B.3.

```
class ProjectAdmin(admin.ModelAdmin):
    # ...
    fieldsets = (
        # ...
        (_("Modules"), {
            "classes": ("collapse",),
            "fields": (("is_backlog_activated", "total_milestones", "
total_story_points"),
                    "is_kanban_activated",
                    "is_schedule_activated",
                    "is_issues_activated",
                    "is_wiki_activated",
```

```

        ("videoconferences", "videoconferences_extra_data")),
    },
    # ...
)

```

Código B.3 – Inclusão de `is_schedule_activated` no grupo *Modules* de `ProjectAdmin`

A alteração situa-se no interior da classe `ProjectAdmin`, no atributo `fieldsets`, especificamente na lista de campos do grupo *Modules*, entre `is_kanban_activated` e `is_issues_activated`, e consiste em uma única linha acrescentada. O campo `is_schedule_activated` foi adicionado ao modelo `Project` (detalhado no bloco de `taiga/projects/models.py`) para permitir ativar ou desativar o módulo de cronograma por projeto, seguindo a convenção das demais *flags* de módulo do Taiga. Para que esse campo possa ser visualizado e alternado pela interface administrativa do Django, ele precisa constar na declaração de `fieldsets` de `ProjectAdmin`, já que campos não listados ali não aparecem no formulário. Como o *Django Admin* do projeto é mantido pelo repositório original, a inclusão nesse arquivo é necessária para expor o controle de forma consistente com os módulos nativos.

Em atualizações futuras, o grupo *Modules* do atributo `fieldsets` é um ponto provável de modificação por parte do projeto oficial, pois a inclusão ou a remoção de módulos altera essa estrutura. Ao integrar novas versões, deve-se verificar se a entrada `is_schedule_activated` permanece no grupo; sua ausência não desativa o módulo, uma vez que a *flag* continua sendo manipulada pela interface do projeto, mas remove o controle da interface administrativa do Django. O nome do campo declarado aqui também precisa corresponder exatamente ao definido no modelo `Project`, sob risco de erro no carregamento do *Django Admin*.

`taiga/projects/epics/api.py`

O arquivo `taiga/projects/epics/api.py` concentra as APIs do aplicativo de épicos. A classe `EpicRelatedUserStoryViewSet` gerencia as histórias de usuário relacionadas a um épico, e seu método `bulk_create` é responsável por criar várias dessas histórias relacionadas de uma só vez. A modificação envolveu a chamada de criação em lote em um bloqueio de exclusão (*advisory lock*), conforme o Código B.4.

```

class EpicRelatedUserStoryViewSet(NestedViewSetMixin, HistoryResourceMixin,
    ...):
    # ...
    def bulk_create(self, request, **kwargs):
        # ...
        if project.blocked_code is not None:
            raise exc.Blocked(_("Blocked element"))

```

```
with advisory_lock("epic-related-user-stories-creation-{}".format(epic.
id)):
    related_userstories = services.create_related_userstories_in_bulk(
        data["bulk_userstories"],
        epic,
        project=project,
        owner=request.user
    )

    for related_userstory in related_userstories:
        self.persist_history_snapshot(obj=related_userstory)
    # ...
```

Código B.4 – Bloqueio de exclusão na criação em lote de histórias relacionadas ao épico

A alteração situa-se no interior do método `bulk_create` da classe `EpicRelatedUserStoryView`. A única linha acrescentada é a abertura do bloco `with advisory_lock(...)`; a chamada já existente a `services.create_related_userstories_in_bulk` foi apenas deslocada para dentro do escopo desse bloqueio. Vale observar que tanto a importação de `advisory_lock` quanto a própria chave de bloqueio já constavam no arquivo original, empregadas pelo caminho de criação individual de história relacionada, de modo que nenhuma importação nova foi necessária.

A modificação é necessária porque a criação de histórias relacionadas a um épico passa a repercutir no módulo de cronograma: por meio de um sinal, essa operação dispara a sincronização da ordenação dos itens de cronograma correspondentes (a ordem usada na lista e no `gantt`), que persiste posições protegidas por uma restrição de unicidade na tabela de ordenação (`schedule_scheduleitemorder`). Sem serialização, duas criações em lote concorrentes para o mesmo épico poderiam disputar a escrita e colidir nessa restrição, deixando a ordenação inconsistente. O *advisory lock* serializa essas operações por épico e reaproveita exatamente a mesma chave de bloqueio já utilizada pelo caminho de criação individual, mantendo os dois fluxos coerentes entre si.

Em atualizações futuras, o método `bulk_create` é mantido pelo repositório original, de modo que qualquer refatoração nele (mudança de assinatura, do corpo ou da indentação) tende a conflitar com o bloco `with` inserido. Ao integrar novas versões, deve-se verificar se o bloqueio continua envolvendo a chamada a `create_related_userstories_in_bulk`. A remoção do bloco não interrompe a execução do código, mas reintroduz a condição de corrida na ordenação do cronograma para as histórias relacionadas criadas em lote.

`taiga/projects/epics/apps.py`

O arquivo `taiga/projects/epics/apps.py` define a configuração do aplicativo de épicos por meio da classe `EpicsAppConfig`, responsável por sua inicialização (entre outras

tarefas, a conexão dos sinais no método `ready`). A modificação acrescentou a essa classe o atributo `watched_types`, conforme o Código B.5.

```
class EpicsAppConfig(AppConfig):  
    name = "taiga.projects.epics"  
    verbose_name = "Epics"  
    watched_types = ["epics.epic"]  
  
    def ready(self):  
        connect_all_epics_signals()
```

Código B.5 – Registro do tipo `epics.epic` no sistema de eventos em tempo real

A alteração situa-se no corpo da classe `EpicsAppConfig` e consiste em uma única linha acrescentada, o atributo de classe `watched_types = ["epics.epic"]`. Esse atributo integra o tipo de conteúdo `epics.epic` ao sistema de eventos em tempo real do Taiga: durante a inicialização, o aplicativo `taiga.events` reúne os `watched_types` declarados por cada configuração de aplicativo em um único conjunto (`events_watched_types`), e os tratadores de sinal consultam esse conjunto para decidir se uma mudança em determinado tipo deve emitir um evento ao vivo para os clientes conectados. No código original, o aplicativo de épicos não declarava esse atributo, ao contrário de histórias de usuário, tarefas e issues, de modo que alterações em épicos não geravam eventos em tempo real.

A modificação é necessária porque as telas de cronograma e de gantt dependem da sincronização em tempo real das alterações entre seções e sessões abertas. Ao registrar `epics.epic` como tipo observado, as modificações em épicos passam a emitir eventos por meio do `taiga.events`, mantendo essas telas consistentes sem necessidade de recarga manual da página.

Em atualizações futuras, a classe `EpicsAppConfig` é mantida pelo repositório original. Caso uma versão nova do Taiga passe a declarar `watched_types` no aplicativo de épicos, ou altere a classe de outra forma, pode haver conflito ou duplicação. Ao integrar novas versões, deve-se verificar se o atributo permanece presente (ou se um equivalente do projeto oficial já cobre `epics.epic`). Sua ausência não interrompe a execução do código, mas desabilita a emissão de eventos em tempo real para alterações em épicos, degradando a sincronização ao vivo do cronograma e do gantt.

`taiga/projects/epics/services.py`

O arquivo `taiga/projects/epics/services.py` concentra a lógica de negócio dos épicos, incluindo a criação em lote e a reordenação em lote de histórias de usuário relacionadas. A modificação fez com que essas duas operações emitissem um sinal genérico ao concluir a respectiva escrita em banco, conforme o Código B.6.

```
from . import models
```

```

from . import signals as epic_signals

# ...

def create_related_userstories_in_bulk(bulk_data, epic, **additional_fields):
    # ...
    db.save_in_bulk(related_userstories)

    epic_signals.related_userstories_bulk_changed.send(
        sender=models.RelatedUserStory,
        epic_id=epic.id,
        project_id=epic.project_id,
        userstory_ids=[userstory.id for userstory in userstories],
    )
    project.update_role_points(user_stories=userstories)
    # ...

def update_epic_related_userstories_order_in_bulk(bulk_data: list, epic: object
):
    # ...
    db.update_attr_in_bulk_for_ids(rus_orders, "order", models.RelatedUserStory
    )

    affected_userstory_ids = {
        item["us_id"] for item in bulk_data if item["us_id"] in rus_conversion
    }
    epic_signals.related_userstories_bulk_changed.send(
        sender=models.RelatedUserStory,
        epic_id=epic.id,
        project_id=epic.project_id,
        userstory_ids=list(affected_userstory_ids),
    )

    return rus_orders

```

Código B.6 – Emissão do sinal de mudança em lote de histórias relacionadas

As alterações compreendem três acréscimos: a importação do módulo de sinais dos épicos (no escopo do módulo) e, no interior das funções `create_related_userstories_in_bulk` e `update_epic_related_userstories_order_in_bulk`, a emissão do sinal `related_userstories_b` após a operação em lote correspondente, carregando `epic_id`, `project_id` e os identificadores das histórias afetadas (`userstory_ids`).

A modificação é necessária porque as operações em lote sobre histórias rela-

cionadas gravam diretamente no banco em bloco. A reordenação, em particular, usa SQL direto (`db.update_attr_in_bulk_for_ids`), que não dispara os sinais nativos de modelo do Django. Dessa forma, o módulo de cronograma não dispõe de um gancho nativo para reagir a essas mudanças e ressincronizar a ordenação dos itens de cronograma. Para suprir isso, os serviços emitem explicitamente um sinal de domínio (`related_userstories_bulk_changed`), ao qual o módulo de cronograma se inscreve. O sinal é nomeado em termos do domínio de épicos, e não do cronograma; com isso, o núcleo dos épicos não importa nem referencia o módulo de cronograma, ficando a dependência invertida, já que é o aplicativo de cronograma que depende do sinal dos épicos.

Em atualizações futuras, as funções `create_related_userstories_in_bulk` e `update_epic_related_userstories_order_in_bulk` são mantidas pelo repositório original, de modo que alterações em seus corpos tendem a conflitar com as emissões inseridas; a linha de importação, por ficar no topo do módulo, também está em área comumente editada pelo projeto oficial. Ao integrar novas versões, deve-se verificar se as chamadas a `.send()` permanecem após as respectivas operações em lote e se o conteúdo enviado (`epic_id`, `project_id` e `userstory_ids`) continua compatível com o que o tratador do cronograma espera. A remoção das emissões não interrompe a execução do código, mas quebra a ressincronização da ordenação do cronograma após a criação ou a reordenação em lote.

`taiga/projects/history/freeze_impl.py`

O arquivo `taiga/projects/history/freeze_impl.py` implementa os *freezers* do sistema de histórico do Taiga. Um *freezer* produz um retrato (*snapshot*) dos campos relevantes de uma entidade, usado pelo módulo de histórico para registrar o estado dos objetos a cada alteração e calcular as diferenças entre versões. A função `project_freezer` monta o retrato de um projeto, listando os campos a serem capturados. A modificação acrescentou o campo `is_schedule_activated` a essa lista, conforme o Código B.7.

```
def project_freezer(project) -> dict:
    fields = ("name",
              # ...
              "is_epics_activated",
              "is_backlog_activated",
              "is_kanban_activated",
              "is_schedule_activated",
              "is_wiki_activated",
              "is_issues_activated")
    return _generic_extract(project, fields)
```

Código B.7 – Inclusão de `is_schedule_activated` no *snapshot* de histórico do projeto

A alteração situa-se no interior da função `project_freezer`, na tupla `fields`, entre os demais campos de ativação de módulos (`is_*_activated`), e consiste em uma única linha acrescentada. O campo `is_schedule_activated` foi adicionado ao modelo `Project` (detalhado no bloco de `taiga/projects/models.py`) para ativar ou desativar o módulo de cronograma por projeto. Para que as mudanças nessa *flag* (ativação ou desativação do módulo) sejam registradas no histórico do projeto e exibidas como diferença, da mesma forma que as demais *flags* de módulo, o campo precisa constar entre os capturados pelo *freezer*; sem isso, alternar o módulo não seria registrado no histórico.

Em atualizações futuras, a tupla `fields` de `project_freezer` é mantida pelo repositório original, e o acréscimo ou a remoção de campos do projeto a modifica, o que a torna um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se a entrada `is_schedule_activated` permanece na lista. Sua ausência não interrompe a execução do código, mas omite a *flag* dos *snapshots* de histórico do projeto, de modo que mudanças na ativação do módulo deixariam de aparecer no histórico de atividades.

`taiga/projects/models.py`

O arquivo `taiga/projects/models.py` define os modelos de dados centrais do aplicativo de projetos, entre eles `Project` e `ProjectTemplate`. A modificação acrescentou o campo `is_schedule_activated` aos dois modelos e o propagou nos dois métodos de `ProjectTemplate` que sincronizam as *flags* entre um projeto e um *template*, conforme o Código B.8.

```
class Project(ProjectDefaults, TaggedMixin, TagsColorsMixin, models.Model):
    # ...
    is_kanban_activated = models.BooleanField(default=False, null=False, blank=True,
                                              verbose_name=_("active kanban panel"))
    is_schedule_activated = models.BooleanField(default=True, null=False, blank=True,
                                              verbose_name=_("active schedule and gantt panels"))
    is_wiki_activated = models.BooleanField(default=True, null=False, blank=True,
                                           verbose_name=_("active wiki panel"))
    )
    # ...

class ProjectTemplate(TaggedMixin, TagsColorsMixin, models.Model):
    # ...
```

```

    is_kanban_activated = models.BooleanField(default=False, null=False, blank=
True,
                                           verbose_name=_("active kanban
panel"))
    is_schedule_activated = models.BooleanField(default=True, null=False, blank
=True,
                                           verbose_name=_("active schedule
and gantt panels"))
    is_wiki_activated = models.BooleanField(default=True, null=False, blank=
True,
                                           verbose_name=_("active wiki panel")
)
# ...

def load_data_from_project(self, project):
    # ...
    self.is_kanban_activated = project.is_kanban_activated
    self.is_schedule_activated = project.is_schedule_activated
    self.is_wiki_activated = project.is_wiki_activated
    # ...

def apply_to_project(self, project):
    # ...
    project.is_kanban_activated = self.is_kanban_activated
    project.is_schedule_activated = self.is_schedule_activated
    project.is_wiki_activated = self.is_wiki_activated
    # ...

```

Código B.8 – Campo `is_schedule_activated` em `Project` e `ProjectTemplate`

As alterações compreendem quatro acréscimos: a declaração do campo `is_schedule_activated` no corpo da classe `Project` e no da classe `ProjectTemplate`, e a propagação desse campo nos métodos `load_data_from_project` (que copia as *flags* do projeto para o *template*) e `apply_to_project` (que aplica as *flags* do *template* a um projeto). O campo é um `BooleanField` com `default=True`, valor que preserva os projetos já existentes mantendo o módulo ativo.

A modificação é a origem da *flag* de ativação do módulo de cronograma por projeto e segue a convenção das demais *flags* de módulo do Taiga (`is_backlog_activated`, `is_kanban_activated`, entre outras). A inclusão em `ProjectTemplate` e nos dois métodos de cópia garante que o estado do módulo seja preservado tanto ao criar um projeto a partir de um *template* quanto ao gerar um *template* a partir de um projeto, de forma coerente com o comportamento dos demais módulos. A mudança de esquema correspondente

é materializada por uma migração nova do aplicativo `projects`, que cria as colunas nas duas tabelas.

Em atualizações futuras, as classes `Project` e `ProjectTemplate` e os métodos `load_data_from_project` e `apply_to_project` são mantidos pelo repositório original; como a evolução dos módulos do Taiga toca exatamente esses pontos, eles são prováveis focos de conflito. Ao integrar novas versões, deve-se verificar se `is_schedule_activated` permanece declarado nos dois modelos e copiado nos dois métodos. Sua ausência no modelo desfaz o controle de ativação do módulo por projeto e exige a remoção da migração correspondente; além disso, qualquer divergência entre o campo e a migração faz o Django acusar mudanças pendentes em `makemigrations`.

`taiga/projects/serializers.py`

O arquivo `taiga/projects/serializers.py` define os serializadores que convertem os modelos de projeto na representação JSON devolvida pela API REST. Três serializadores foram alterados: `ProjectSerializer` (representação completa do projeto), `ProjectLightSerializer` (representação reduzida) e `ProjectTemplateSerializer` (*template* de projeto). Cada um recebeu a declaração do campo `is_schedule_activated`, conforme o Código B.9.

```
class ProjectSerializer(serializers.LightSerializer):
    # ...
    is_kanban_activated = Field()
    is_schedule_activated = Field()
    is_wiki_activated = Field()
    # ...

class ProjectLightSerializer(serializers.LightSerializer):
    # ...
    is_kanban_activated = Field()
    is_schedule_activated = Field()
    is_wiki_activated = Field()
    # ...

class ProjectTemplateSerializer(serializers.LightSerializer):
    # ...
    is_kanban_activated = Field()
    is_schedule_activated = Field()
    is_wiki_activated = Field()
    # ...
```

Código B.9 – Exposição de `is_schedule_activated` nos serializadores de projeto

As alterações situam-se no corpo das três classes de serializador, entre os demais campos de ativação de módulos (`is_*_activated`), e consistem em uma única linha acrescentada em cada uma. O campo `is_schedule_activated` foi adicionado ao modelo `Project` (detalhado no bloco de `taiga/projects/models.py`), mas só chega ao cliente se for declarado nos serializadores, pois o serializador-base utilizado (`LightSerializer`) expõe apenas os campos explicitamente listados. A inclusão nas representações completa e reduzida garante que a *flag* esteja disponível independentemente de qual representação o cliente receba, e a inclusão no serializador de *template* mantém a coerência da API correspondente. No *front-end*, esse valor é lido para habilitar ou desabilitar as seções de cronograma e de gantt.

Em atualizações futuras, os três serializadores são mantidos pelo repositório original, e a inclusão ou a remoção de *flags* de módulo edita exatamente essas listas de campos, o que as torna prováveis focos de conflito. Ao integrar novas versões, deve-se verificar se `is_schedule_activated = Field()` permanece nas três classes. Sua ausência não interrompe a execução do código, mas oculta a *flag* da API, de modo que o *front-end* deixaria de receber o estado de ativação e não conseguiria habilitar ou desabilitar o módulo corretamente.

`taiga/routers.py`

O arquivo `taiga/routers.py` é o registro central das rotas da API REST do Taiga: ele importa os *viewsets* e os registra no *router* global, definindo as URLs expostas sob `/api/v1/`. A modificação importou dois *viewsets* do aplicativo de cronograma e os registrou em suas respectivas rotas, conforme o Código B.10.

```
# ...
from taiga.projects.wiki.api import WikiWatchersViewSet
from taiga.projects.schedule.api import ScheduleDependencyViewSet
from taiga.projects.schedule.api import ScheduleItemViewSet

# ...

router.register(r"wiki-links", WikiLinkViewSet,
               base_name="wiki-links")
router.register(r"schedule-dependencies", ScheduleDependencyViewSet,
               base_name="schedule-dependencies")
router.register(r"schedule-items", ScheduleItemViewSet,
               base_name="schedule-items")
```

Código B.10 – Registro das rotas da API do módulo de cronograma

As alterações compreendem dois acréscimos, ambos no escopo do módulo: a importação dos *viewsets* `ScheduleDependencyViewSet` e `ScheduleItemViewSet`, junto às demais importações de *viewsets*, e o registro dessas classes no *router* global sob as rotas `schedule-dependencies` e `schedule-items`, junto aos demais registros.

A modificação é inevitável porque os *endpoints* do módulo de cronograma só ficam acessíveis pela API se forem registrados neste *router* central. Como o módulo é um aplicativo independente, seus *viewsets* precisam ser importados e registrados aqui. A rota `schedule-dependencies` atende às dependências entre atividades e à aplicação de datas em lote, enquanto a rota `schedule-items` atende ao *endpoint* dedicado de atualização de datas (`update_dates`).

Em atualizações futuras, o `taiga/routers.py` é mantido pelo repositório original, e a inclusão ou a remoção de módulos da API edita tanto o bloco de importações quanto o de registros, o que torna esses dois trechos prováveis focos de conflito. Ao integrar novas versões, deve-se verificar se ambas as importações e ambos os `router.register` permanecem presentes. Sua ausência não impede a compilação do módulo, mas torna os *endpoints* de cronograma inacessíveis (erro 404), desabilitando as dependências, as datas em lote e a escrita dedicada de datas, o que inviabiliza as funcionalidades de cronograma e gantt que dependem desses *endpoints*.

B.1.2 Front-end

`app/coffee/app.coffee`

O arquivo `app/coffee/app.coffee` é o *bootstrap* da aplicação *front-end* em AngularJS. Ele configura as rotas da aplicação (por meio de `$routeProvider`) e lista os módulos Angular que a compõem. A modificação registrou as rotas das telas de cronograma e de gantt e acrescentou os módulos correspondentes à lista de módulos da aplicação, conforme o Código B.11.

```
configure = ($routeProvider, ...) ->
  # ...
  $routeProvider.when("/project/:pslug/gantt",
    {
      templateUrl: "gantt/gantt.html",
      loader: true,
      section: "gantt"
    }
  )
  # ...
```

```

# Schedule list
$routeProvider.when("/project/:pslug/schedulelist",
  {
    templateUrl: "schedule/schedule.html",
    loader: true,
    section: "schedule"
  }
)

# Backwards compatibility for old schedule URL
$routeProvider.when("/project/:pslug/schedule",
  {
    redirectTo: "/project/:pslug/schedulelist"
  }
)

# ...

modules = [
  # ...
  "taigaTeam",
  "taigaGantt",
  "taigaSchedule",
  "taigaWiki",
  # ...
]

```

Código B.11 – Rotas e módulos de cronograma e gantt no *bootstrap* da aplicação

As alterações compreendem três acréscimos: a rota `/project/:pslug/gantt`, que aponta para o *template* do gantt; a rota `/project/:pslug/schedulelist`, que corresponde à lista de atividades, acompanhada de um redirecionamento de compatibilidade da URL antiga `/project/:pslug/schedule` para a nova; e os módulos `taigaGantt` e `taigaSchedule` na lista de módulos da aplicação.

`app/coffee/modules/admin/roles.coffee`

O arquivo `app/coffee/modules/admin/roles.coffee` implementa a tela de administração de papéis (`/project/:project/admin/roles`), que lista as permissões por categoria e permite concedê-las a cada papel. A modificação tem três frentes: incluir as permissões de cronograma nas listas de permissões padrão; acrescentar a categoria de permissões de cronograma e gantt exibida na tela; e ajustar o tratamento das permissões em projetos públicos. O Código B.12 mostra as inclusões nas listas e o ajuste de projetos públicos.

```

publicPermissions = ['view_epics', 'view_milestones', 'view_us', 'view_tasks',
  'view_issues', 'view_schedule', 'view_gantt', 'view_wiki_pages', '
  view_wiki_links']

# ...

NewRoleDirective = ($tgrepo, $confirm) ->
  DEFAULT_PERMISSIONS = ["view_project", "view_milestones", "view_us", "
  view_tasks", "view_issues", "view_schedule", "view_gantt"]
  # ...

# ...

  generateCategoriesFromRole = (role) ->
    getPermissionsForDisplay = () ->
      permissions = role.permissions or []

      if !$scope.project.is_private
        permissions = _.uniq(permissions.concat($scope.project.
public_permissions or []))

      return permissions

    setActivePermissions = (permissions) ->
      activePermissions = getPermissionsForDisplay()

      return _.map(permissions, (x) -> _.extend({}, x, {active: x["
key"] in activePermissions}))

```

Código B.12 – Permissões padrão de cronograma e tratamento de projetos públicos em `roles.coffee`

As permissões `view_schedule` e `view_gantt` foram acrescentadas à lista `publicPermissions` (permissões consideradas públicas) e à lista `DEFAULT_PERMISSIONS` (permissões padrão de papéis novos). No tratamento de projetos públicos houve um ajuste: o código original, no controlador, ao carregar os papéis de um projeto público, copiava as permissões públicas para o próprio *array* de permissões do papel, uma mutação que podia acabar persistida. Esse bloco foi removido e sua intenção reimplementada apenas para exibição, na diretiva: a função `getPermissionsForDisplay` monta a união das permissões do papel com as permissões públicas do projeto (`$scope.project.public_permissions`) quando o projeto não é privado, e `setActivePermissions` passou a marcar como ativas

as permissões dessa união, sem alterar o papel salvo.

O Código B.13 mostra a categoria de permissões de cronograma e gantt acrescentada à montagem das categorias exibidas na tela.

```
# ...

    scheduleGanttPermissions = [
        { key: "view_schedule", name: "COMMON.PERMISSIONS_CATEGORIES.
        SCHEDULE_GANTT.VIEW_SCHEDULE"}
        { key: "view_gantt", name: "COMMON.PERMISSIONS_CATEGORIES.
        SCHEDULE_GANTT.VIEW_GANTT"}
        { key: "modify_schedule_links", name: "COMMON.
        PERMISSIONS_CATEGORIES.SCHEDULE_GANTT.MODIFY_LINKS"}
        { key: "modify_schedule_dates", name: "COMMON.
        PERMISSIONS_CATEGORIES.SCHEDULE_GANTT.MODIFY_EXPECTED_START_DATES"}
        { key: "modify_schedule_color", name: "COMMON.
        PERMISSIONS_CATEGORIES.SCHEDULE_GANTT.MODIFY_COLOR"}
        { key: "modify_gantt_list_order", name: "COMMON.
        PERMISSIONS_CATEGORIES.SCHEDULE_GANTT.MODIFY_GANTT_LIST_ORDER"}
    ]

    categories.push({
        name: "COMMON.PERMISSIONS_CATEGORIES.SCHEDULE_GANTT.NAME",
        permissions: setActivePermissions(scheduleGanttPermissions)
    })
# ...
```

Código B.13 – Categoria de permissões de cronograma e gantt na tela de papéis

A lista `scheduleGanttPermissions` define as seis permissões do módulo (duas de visualização e quatro de modificação granular), cada uma com sua chave de tradução, e é empilhada em `categories` para aparecer como uma categoria própria na tela de papéis.

Em atualizações futuras, o `roles.coffee` é mantido pelo repositório original; as listas `publicPermissions` e `DEFAULT_PERMISSIONS`, a função `generateCategoriesFromRole` e o trecho do controlador que tratava projetos públicos são áreas que o projeto oficial edita ao evoluir permissões e módulos, sendo prováveis focos de conflito; em especial, a remoção do bloco do controlador conflitará caso o *upstream* altere aquele trecho. Ao integrar novas versões, deve-se verificar se `view_schedule` e `view_gantt` permanecem nas duas listas, se a categoria `scheduleGanttPermissions` continua sendo empilhada e se o tratamento de projetos públicos por `getPermissionsForDisplay` não foi desfeito por uma reintrodução do bloco antigo no controlador, o que reativaria a mutação indevida das permissões do papel. O bloco também depende das chaves `COMMON.PERMISSIONS_CATEGORIES.SCHEDULE_GANTT.*` (arquivos de *locale*), que precisam existir.

`app/coffee/modules/base.coffee`

O arquivo `app/coffee/modules/base.coffee` define, entre outras coisas, o objeto `urls`, que mapeia chaves de navegação para padrões de URL. Esse mapa é consumido pelo serviço de navegação (`tgNavUrls`, usado pela diretiva `tg-nav`) para resolver os *links* ao longo da aplicação. A modificação acrescentou duas entradas, referentes às telas de cronograma e de gantt, conforme o Código B.14.

```
urls = {  
  # ...  
  "project-kanban": "/project/:project/kanban"  
  "project-gantt": "/project/:project/gantt"  
  "project-issues": "/project/:project/issues"  
  "project-epics": "/project/:project/epics"  
  "project-schedule": "/project/:project/schedulelist"  
  "project-search": "/project/:project/search"  
  # ...  
}
```

Código B.14 – Rotas de navegação de cronograma e gantt no mapa `urls`

As alterações situam-se no interior do objeto `urls`, entre as demais entradas `project-*`, e consistem em duas linhas acrescentadas: a chave `project-gantt`, que aponta para `/project/:project/gantt`, e a chave `project-schedule`, que aponta para `/project/:project/schedulelist` (a rota renomeada da lista de atividades).

A modificação é necessária porque o sistema de navegação resolve os *links* por chave (por exemplo, `tg-nav="project-schedule:project=..."`); para que o menu, os redirecionamentos e os *links* para as telas de cronograma e de gantt sejam resolvidos, essas chaves precisam estar declaradas neste mapa.

Em atualizações futuras, o mapa `urls` de `base.coffee` é mantido pelo repositório original, e a inclusão ou a remoção de seções navegáveis o edita, o que o torna um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se as entradas `project-gantt` e `project-schedule` permanecem presentes. Sua ausência quebra a resolução dos *links* para essas telas, pois a diretiva `tg-nav` não conseguiria resolver a chave, afetando o menu e qualquer referência de navegação a essas seções.

`app/coffee/modules/resources.coffee`

O arquivo `app/coffee/modules/resources.coffee` define o objeto `urls` que mapeia chaves de recurso para os caminhos dos *endpoints* da API. Esse mapa é consumido pela camada de recursos e repositório do *front-end* (por exemplo, `@repo.queryMany("schedule-items", ...)`) para realizar as requisições HTTP ao

back-end. A modificação acrescentou quatro entradas referentes aos *endpoints* do módulo de cronograma, conforme o Código B.15.

```
urls = {
    # ...
    # Tasks
    "tasks": "/tasks"
    "schedule-dependencies": "/schedule-dependencies"
    "schedule-items": "/schedule-items"
    "schedule-items-update": "/schedule-items/update_item"
    "schedule-bulk-date-apply": "/schedule-dependencies/bulk_apply_dates"
    "bulk-create-tasks": "/tasks/bulk_create"
    # ...
}
```

Código B.15 – *Endpoints* da API do módulo de cronograma no mapa de recursos

As alterações situam-se no interior do objeto `urls`, próximas às entradas de tarefas, e consistem em quatro linhas acrescentadas: `schedule-dependencies` (dependências entre atividades), `schedule-items` (recurso de itens do cronograma), `schedule-items-update` (atualização de datas do item) e `schedule-bulk-date-apply` (aplicação de datas em lote). Cada chave aponta para uma rota registrada no *router* do *back-end* (`taiga/routers.py`).

A modificação é necessária porque a camada de recursos do *front-end* resolve os caminhos da API por chave; para que o módulo de cronograma consulte e persista dados pela API, essas chaves precisam estar declaradas neste mapa.

Em atualizações futuras, o mapa `urls` de `resources.coffee` é mantido pelo repositório original, e a inclusão ou a remoção de recursos o edita, o que o torna um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se as quatro entradas de cronograma permanecem presentes. Sua ausência quebra as chamadas do *front-end* à API do módulo, pois a chave de recurso não resolveria para um caminho, inviabilizando o carregamento e a persistência de dados do cronograma.

`app/locales/taiga/locale-en.json` e `app/locales/taiga/locale-pt-br.json`

Os arquivos `app/locales/taiga/locale-en.json` e `app/locales/taiga/locale-pt-br.json` são os dicionários de tradução do *front-end* (inglês e português do Brasil, respectivamente). O módulo de cronograma acrescentou o mesmo conjunto de chaves a ambos, diferindo apenas nos valores traduzidos. Os Códigos B.16 e B.17 mostram um trecho representativo dos arquivos em inglês e em português do Brasil, respectivamente; todo o conteúdo apresentado é acréscimo.

```
"PERMISSIONS_CATEGORIES": {
  "SCHEDULE_GANTT": {
    "NAME": "Schedule",
```

```

    "VIEW_SCHEDULE": "View activities list",
    "VIEW_GANTT": "View gantt",
    "MODIFY_LINKS": "Modify links",
    "MODIFY_EXPECTED_START_DATES": "Modify expected start dates",
    "MODIFY_COLOR": "Modify color",
    "MODIFY_GANTT_LIST_ORDER": "Modify gantt list order"
  }
}
"MODULES": {
  "SCHEDULE_GANTT": "Schedule",
  "SCHEDULE_GANTT_DESCRIPTION": "Plan and visualize activities, dates and
dependencies in Schedule and Gantt views."
}
"GANTT": {
  "ZOOM_OPTIONS": "Zoom options",
  "LINK_MODE": "Connect activities",
  "ZOOM": { "DAILY": "Daily", "WEEKLY": "Weekly", "MONTHLY": "Monthly" },
  "CRITICAL_PATH": { "SLACK_LABEL": "Slack", "SLACK_DAY_SINGULAR": "day", "
SLACK_DAY_PLURAL": "days" },
  "RESIZE_POPUP": { "START": "Start", "END": "End", "DATE_FORMAT": "DD MMMM
YYYY" }
}
"SCHEDULE": {
  "FILTERS": { "ESTIMATED_START_RANGE": "Estimated start", "DUE_DATE_RANGE":
"Due date", ... },
  "TABLE": { "COLUMNS": { "TYPE": "Type", "SUBJECT": "Subject", "DUE_DATE": "
Due Date", ... } }
}

```

Código B.16 – Trecho das chaves de tradução do módulo de cronograma (arquivo em inglês)

```

"PERMISSIONS_CATEGORIES": {
  "SCHEDULE_GANTT": {
    "NAME": "Schedule",
    "VIEW_SCHEDULE": "Ver lista de atividades",
    "VIEW_GANTT": "Ver gantt",
    "MODIFY_LINKS": "Modificar links",
    "MODIFY_EXPECTED_START_DATES": "Modificar datas esperadas de início",
    "MODIFY_COLOR": "Modificar cor",
    "MODIFY_GANTT_LIST_ORDER": "Modificar ordem da lista do gantt"
  }
}

```

```

"MODULES": {
  "SCHEDULE_GANTT": "Schedule",
  "SCHEDULE_GANTT_DESCRIPTION": "Planeje e visualize atividades, datas e
  dependências nas visões de Schedule e Gantt."
}
"GANTT": {
  "ZOOM_OPTIONS": "Opções de zoom",
  "LINK_MODE": "Ligar atividades",
  "ZOOM": { "DAILY": "Diário", "WEEKLY": "Semanal", "MONTHLY": "Mensal" },
  "CRITICAL_PATH": { "SLACK_LABEL": "Folga", "SLACK_DAY_SINGULAR": "dia", "
  SLACK_DAY_PLURAL": "dias" },
  "RESIZE_POPUP": { "START": "Começo", "END": "Fim", "DATE_FORMAT": "DD MMMM
  YYYY" }
}
"SCHEDULE": {
  "FILTERS": { "ESTIMATED_START_RANGE": "Início estimado", "DUE_DATE_RANGE":
  "Data de vencimento", ... },
  "TABLE": { "COLUMNS": { "TYPE": "Type", "SUBJECT": "Subject", "DUE_DATE": "
  Due Date", ... } }
}

```

Código B.17 – Trecho das chaves de tradução do módulo de cronograma (arquivo em português do Brasil)

As chaves acrescentadas distribuem-se em cinco grupos, cada um consumido por uma parte da interface: o grupo `PERMISSIONS_CATEGORIES.SCHEDULE_GANTT` fornece os rótulos da categoria de permissões do módulo, exibidos na tela de papéis (`roles.coffee`); o grupo `MODULES.SCHEDULE_GANTT` (nome e descrição) alimenta o *card* de ativação na administração de módulos (`admin-project-modules.jade`); as chaves de menu (`SCHEDULE`, `ACTIVITIES_LIST` e `GANTT`) rotulam o menu lateral; o grupo `GANTT` reúne os textos da interface do gantt (opções de *zoom*, modo de ligação de atividades, caminho crítico e folga, e o *popup* de redimensionamento); e o grupo de primeiro nível `SCHEDULE` reúne os textos da lista de atividades (filtros e colunas da tabela). Os dois arquivos recebem exatamente a mesma estrutura de chaves, diferindo apenas nos valores traduzidos.

`app/partials/admin/admin-project-modules.jade`

O arquivo `app/partials/admin/admin-project-modules.jade` é o *template* da página de administração de módulos do projeto (`/project/:project/admin/project-profile/modu`) na qual cada módulo é apresentado como um *card* com um interruptor de ativação. A modificação acrescentou o *card* do módulo de cronograma e gantt, conforme o Código B.18.

```

div.wrapper(...)
    // ...
    .module.module-kanban(...)
        // ...
        .module.module-schedule-gantt(ng-class="{true:'active', false:''}[project.
is_schedule_activated]")
            .module-icon
                tg-svg(svg-icon="icon-module-schedule")
            .module-name(translate="ADMIN.MODULES.SCHEDULE_GANTT")
            .module-desc(translate="ADMIN.MODULES.SCHEDULE_GANTT_DESCRIPTION")
            .module-activation.module-direct-active
                div.check
                    input.activate-input(
                        id="functionality-schedule-gantt"
                        name="functionality-schedule-gantt"
                        type="checkbox"
                        ng-checked="project.is_schedule_activated"
                        ng-model="project.is_schedule_activated"
                    )
                div
                    span.check-text.check-yes(translate="COMMON.YES")
                    span.check-text.check-no(translate="COMMON.NO")
            .module.module-issues(...)
        // ...

```

Código B.18 – *Card* de ativação do módulo de cronograma e gantt na administração de módulos

A alteração consiste em um bloco acrescentado no interior do contêiner de módulos (`div.wrapper`), entre o *card* do kanban e o de issues. O novo *card* (`.module.module-schedule-gantt`) exibe o ícone, o nome e a descrição do módulo (por meio das chaves de tradução `ADMIN.MODULES.SCHEDULE_GANTT` e `ADMIN.MODULES.SCHEDULE_GANTT_DESCRIPTION`) e uma caixa de seleção vinculada a `project.is_schedule_activated` (`ng-model`), que alterna a *flag* por projeto pelo mesmo fluxo de salvamento dos demais módulos.

A modificação adiciona o controle, na interface padrão de administração do projeto, que ativa ou desativa o módulo de cronograma e gantt. Sem ele, a *flag* `is_schedule_activated` só seria editável pela administração do Django ou pela API, e não pela tela usual de administração do projeto. O acréscimo do *card* mantém o módulo consistente com os demais apresentados nessa página.

Em atualizações futuras, este *template* lista os *cards* de módulos e é mantido pelo repositório original; a inclusão ou a remoção de módulos edita exatamente este

arquivo, o que o torna um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se o *card* de cronograma e gantt permanece presente. Sua ausência não quebra a página, mas remove o interruptor da interface de administração do projeto, de modo que o módulo deixaria de poder ser ativado ou desativado por ali. O bloco também depende das chaves de tradução `ADMIN.MODULES.SCHEDULE_GANTT*` (arquivos de *locale*) e do ícone `icon-module-schedule` (definido no *sprite* de SVG), que precisam permanecer consistentes.

app/svg/sprite.svg

O arquivo `app/svg/sprite.svg` é o *sprite* de SVG da aplicação, um único arquivo que agrega todos os ícones como elementos `symbol`, referenciados ao longo da interface por `tg-svg(svg-icon="...")` (que resolve para um `use` apontando ao id do símbolo). A modificação acrescentou um símbolo, `icon-module-schedule`, que é o ícone do módulo de cronograma, conforme o Código B.19.

```
<svg ...>
  <!-- ... -->
  <symbol id="icon-module-schedule" stroke="none" viewBox="0 0 24 24" >
    <title>schedule</title>
    <path fill-rule="evenodd" clip-rule="evenodd" d="M6.75 2C7.16421 2 ...
Z" />
  </symbol>
  <!-- ... -->
</svg>
```

Código B.19 – Símbolo SVG do ícone do módulo de cronograma

A alteração consiste em um único elemento `symbol` acrescentado, com `id="icon-module-schedule"`, contendo o título e o traçado (`path`) do ícone, inserido entre os demais símbolos no escopo raiz do SVG.

A modificação é necessária porque o *card* do módulo na administração (`admin-project-modules.jade`), entre outros pontos da interface, referencia esse ícone pelo id (`tg-svg(svg-icon="icon-module-schedule")`). Para que o ícone seja renderizado, o símbolo correspondente precisa existir no *sprite*. Como o *sprite* é um arquivo único e compartilhado, mantido pelo repositório original, o símbolo teve de ser acrescentado aqui.

Em atualizações futuras, o `sprite.svg` é mantido pelo repositório original, e a inclusão ou a remoção de ícones o edita; por ser um arquivo único que concentra todos os símbolos, é um ponto provável de conflito, e as mesclagens nele tendem a ser ruidosas. Ao integrar novas versões, deve-se verificar se o símbolo `icon-module-schedule` permanece presente e se seu `id` continua idêntico ao referenciado nos *templates*. Sua ausência não

quebra a página, mas faz o ícone deixar de ser renderizado (ícone vazio ou quebrado) onde for referenciado.

`elements.js`

O arquivo `elements.js` não é código-fonte editado manualmente, e sim um artefato de *build*: trata-se do *bundle* compilado dos *webcomponents* do `taiga-front-next` (entre eles o `tg-project-navigation`, que renderiza o menu lateral), gerado pela compilação e copiado para dentro do `taiga-front` para que a aplicação legada possa carregar os componentes modernos (via `tg-legacy-loader`). É um arquivo minificado, com cerca de 1,7 MB em uma única linha lógica, como ilustra o Código B.20.

```
!function(e){function r(r){for(var n,l,f=r[0],i=r[1],p=r[2],c=0,s=[]; ... }
```

Código B.20 – Início do *bundle* minificado `elements.js` (amostra ilustrativa)

A diferença observada nesse arquivo na branch do cronograma é a regeneração do *bundle* refletindo as mudanças feitas no código-fonte do `taiga-front-next` (em especial a integração da navegação de cronograma e gantt no menu lateral). Como o conteúdo é minificado e ocupa uma única linha, o sistema de controle de versão registra a alteração como a substituição dessa linha inteira; não há, portanto, trecho legível a destacar.

A modificação é consequência inevitável da integração de menu no `taiga-front-next`: sempre que aquele projeto é alterado, o *bundle* precisa ser recompilado e copiado para `taiga-front/elements.js`, o que é feito pelo *script* `compile-front-end-next.sh` na raiz do monorepo.

Em atualizações futuras, alguns cuidados específicos se aplicam a este arquivo. Ele nunca deve ser editado à mão; deve ser regenerado executando o `compile-front-end-next.sh` após qualquer mudança no `taiga-front-next`. Conflitos de mesclagem neste arquivo não devem ser resolvidos manualmente, pois é inviável mesclar um *bundle* minificado de linha única; o procedimento correto é recompilar o `taiga-front-next` e deixar o *script* sobrescrever o `elements.js`. Por fim, o arquivo precisa permanecer sincronizado com o código-fonte do `taiga-front-next`: um `elements.js` desatualizado renderiza uma versão antiga do menu e dos *webcomponents*, mesmo que o código-fonte esteja correto.

`gulpfile.js`

O arquivo `gulpfile.js` é a configuração de *build* (Gulp) do `taiga-front`. O *array* `paths.coffee_order` define a lista ordenada de *globs* de arquivos CoffeeScript a serem concatenados e compilados no *bundle* da aplicação; a ordem importa porque as definições de módulos e dependências do AngularJS precisam ser carregadas de forma consistente. A modificação acrescentou o *glob* dos arquivos do módulo de cronograma, conforme o Código B.21.

```
paths.coffee_order = [  
  // ...  
  paths.app + "coffee/modules/tasks/*.coffee",  
  paths.app + "coffee/modules/team/*.coffee",  
  paths.app + "coffee/modules/schedule/*.coffee",  
  paths.app + "coffee/modules/wiki/*.coffee",  
  paths.app + "coffee/modules/admin/*.coffee",  
  // ...  
]
```

Código B.21 – Inclusão do módulo de cronograma na ordem de compilação CoffeeScript

A alteração situa-se no interior do *array* `paths.coffee_order`, entre os demais *globs* de módulos, e consiste em uma única linha acrescentada, referente aos arquivos em `coffee/modules/schedule/`.

A modificação é necessária porque os arquivos CoffeeScript do módulo de cronograma precisam ser incluídos no *build*, com ordem definida em relação aos demais módulos. Sem essa entrada, os controladores e diretivas do módulo não seriam compilados no *bundle* da aplicação, e a tela da lista de atividades (`schedulelist`) deixaria de carregar por falta do módulo registrado.

Em atualizações futuras, o `gulpfile.js` e a lista `paths.coffee_order` são mantidos pelo repositório original, e a inclusão ou a remoção de módulos edita essa lista, o que a torna um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se o *glob* `coffee/modules/schedule/*.coffee` permanece presente. Sua ausência exclui o módulo de cronograma do *build*, de modo que a tela da lista de atividades falharia ao carregar, por o módulo e seus controladores não estarem registrados.

B.1.3 Front-end-Next

`src/app/api/projects/projects.model.mock.ts`

O arquivo `projects.model.mock.ts` define a *factory* de objetos de projeto usada nos testes (`ProjectMockFactory`). A modificação acrescentou o campo `isScheduleActivated` a essa *factory*, conforme o Código B.22.

```
export const ProjectMockFactory = Factory.Sync.makeFactory<Project>({  
  // ...  
  isIssuesActivated: faker.random.boolean(),  
  isKanbanActivated: faker.random.boolean(),  
  isScheduleActivated: true,  
  isLookingForPeople: faker.random.boolean(),  
  // ...  
});
```

Código B.22 – Campo `isScheduleActivated` na *factory* de *mock* de projeto

A alteração consiste em uma única linha acrescentada. Como o tipo `Project` passou a declarar o campo `isScheduleActivated` (ver o bloco de `projects.model.ts`), a *factory* de *mock* precisa fornecer um valor para ele, sob pena de o TypeScript acusar campo ausente na construção do objeto nos testes. O valor fixo `true` reflete o comportamento de módulo ativo por padrão.

Em atualizações futuras, este arquivo é mantido pelo repositório original e acompanha a evolução do tipo `Project`, sendo um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se o campo permanece na *factory*. Sua ausência não afeta a aplicação em produção, mas pode quebrar a compilação dos testes se o campo for obrigatório no tipo.

`src/app/api/projects/projects.model.ts`

O arquivo `projects.model.ts` define o tipo TypeScript do projeto na aplicação moderna. A modificação acrescentou o campo `isScheduleActivated` tanto à interface `Project` quanto à projeção `ProjectListEntry`, conforme o Código B.23.

```
export type ProjectListEntry = Pick<Project,
  // ...
  'isIssuesActivated' |
  'isKanbanActivated' |
  'isScheduleActivated' |
  'isLookingForPeople' |
  // ...
>;

export interface Project {
  // ...
  isIssuesActivated: boolean;
  isKanbanActivated: boolean;
  isScheduleActivated: boolean;
  isLookingForPeople: boolean;
  // ...
}
```

Código B.23 – Campo `isScheduleActivated` no tipo `Project`

A alteração compreende dois acréscimos: a declaração do campo `isScheduleActivated` (do tipo `boolean`) na interface `Project` e a inclusão da mesma chave na projeção `ProjectListEntry`. Esse campo é a contraparte tipada da *flag* `is_schedule_activated`

exposta pela API (em *camelCase*); o componente de navegação lê `project.isScheduleActivated` para condicionar a visibilidade do menu de cronograma e gantt. Sem a declaração do campo, o TypeScript não compilaria as referências a ele.

Em atualizações futuras, este tipo é mantido pelo repositório original, e a inclusão ou a remoção de *flags* de módulo o edita, o que o torna um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se `isScheduleActivated` permanece declarado na interface e na projeção. Sua ausência quebra a compilação do componente de navegação, que referencia o campo.

`src/app/api/roles/roles.model.ts`

O arquivo `roles.model.ts` define o enum `Permissions`, que associa chaves em *camelCase* aos codinomes de permissão do *back-end*. A modificação acrescentou as seis permissões de cronograma e gantt, conforme o Código B.24.

```
export enum Permissions {  
  // ...  
  deleteWikiPage = 'delete_wiki_page',  
  // Schedule & Gantt  
  viewSchedule = 'view_schedule',  
  viewGantt = 'view_gantt',  
  modifyScheduleLinks = 'modify_schedule_links',  
  modifyScheduleDates = 'modify_schedule_dates',  
  modifyScheduleColor = 'modify_schedule_color',  
  modifyGanttListOrder = 'modify_gantt_list_order',  
  // Project  
  viewProject = 'view_project',  
  // ...  
}
```

Código B.24 – Permissões de cronograma e gantt no enum `Permissions`

A alteração situa-se no interior do enum `Permissions` e consiste em seis entradas acrescentadas (`viewSchedule`, `viewGantt`, `modifyScheduleLinks`, `modifyScheduleDates`, `modifyScheduleColor` e `modifyGanttListOrder`). Cada entrada mapeia para o codinome correspondente do *back-end* (`view_schedule`, `view_gantt` e demais). O componente de navegação usa esses valores (por exemplo, `Permissions.viewSchedule`) para condicionar a visibilidade dos itens de menu.

Em atualizações futuras, o enum é mantido pelo repositório original, e a inclusão ou a remoção de permissões o edita, o que o torna um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se as seis entradas permanecem e se seus valores continuam idênticos aos codinomes declarados no *back-end* (`taiga/permissions/choices.py`). Sua ausência quebra a compilação e a checagem de permissão no componente de navegação.

`src/app/commons/components/svg-sprite/svg-sprite.component.html`

O arquivo `svg-sprite.component.html` é o *sprite* de SVG da aplicação moderna, que agrega os ícones como elementos `symbol`. A modificação acrescentou o símbolo `calendar`, usado pelo item de menu de cronograma, conforme o Código B.25.

```
<svg ...>
  <!-- ... -->
  <symbol id="calendar" viewBox="0 0 24 24">
    <path fill-rule="evenodd" clip-rule="evenodd" d="M6.75 2C7.16421 2 ... Z"
  />
</symbol>
  <!-- ... -->
</svg>
```

Código B.25 – Símbolo `calendar` no *sprite* do `taiga-front-next`

A alteração consiste em um único elemento `symbol` acrescentado, com `id="calendar"`, contendo o traçado do ícone. O *template* do menu referencia esse símbolo por `<use href="#calendar"/>`; para que o ícone seja renderizado no menu moderno, o símbolo precisa existir neste *sprite*. Vale notar que este *sprite* é distinto do *sprite* do `taiga-front`, no qual o ícone equivalente recebeu o `id icon-module-schedule`.

Em atualizações futuras, este arquivo é mantido pelo repositório original; por concentrar todos os ícones, é um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se o símbolo `calendar` permanece e se seu `id` continua idêntico ao referenciado no *template*. Sua ausência não quebra a página, mas faz o ícone do item de menu deixar de ser renderizado.

`src/app/commons/project-navigation/project-navigation.component.css`

O arquivo `project-navigation.component.css` contém os estilos do componente de menu lateral. A modificação estendeu ao *dropdown* de cronograma os mesmos estilos já aplicados ao *dropdown* de `scrum`, conforme o Código B.26.

```
:host(.collapsed) {
  width: 48px;

  & .scrum .child-menu,
  & .schedule .child-menu {
    display: none;
  }
}

/* ... */
```

```
.scrum,
.schedule {
  &.active {
    & .agile-chevron {
      transform: rotate(90deg);
    }
    /* ... */
  }
}
```

Código B.26 – Estilos do *dropdown* de cronograma no menu lateral

A alteração compreende dois acréscimos de seletor: a inclusão de `& .schedule` `.child-menu` na regra que oculta o submenu quando o menu está recolhido, e a inclusão de `.schedule` junto a `.scrum` na regra de estado ativo (rotação do *chevron* e exibição do submenu). Como o item de cronograma é um *dropdown* análogo ao de *scrum*, ele reaproveita os mesmos estilos.

Em atualizações futuras, este arquivo é mantido pelo repositório original; caso o projeto oficial reestruture esses seletores, pode haver conflito. Ao integrar novas versões, deve-se verificar se o seletor `.schedule` permanece nas duas regras. Sua ausência não quebra a navegação, mas o *dropdown* de cronograma perde o comportamento visual de recolhimento e de estado ativo.

`src/app/commons/project-navigation/project-navigation.component.html`

O arquivo `project-navigation.component.html` é o *template* do menu lateral. A modificação acrescentou o item de menu de cronograma e *gant*t, no formato de *dropdown* com dois subitens, conforme o Código B.27.

```
<ul class="menu-main main-menu">
  <!-- ... -->
  <li
    *ngIf="isMenuScheduleGroupEnabled"
    class="menu-option schedule"
    [class.active]="scheduleVisible"
    (mouseenter)="popupSchedule($event)"
    (mouseleave)="out()">
    <button (click)="toggleSchedule()" type="button"
      title="{{'PROJECT.SECTION.SCHEDULE' | translate}}">
      <svg><use href="#calendar" /></svg>
      <span class="menu-option-text" data-animation="text">
        {{'PROJECT.SECTION.SCHEDULE' | translate}}
```

```

    </span>
    <svg class="agile-chevron" data-animation="text"><use href="#chevron-
left" /></svg>
  </button>
  <ul class="child-menu">
    <li *ngIf="isMenuScheduleEnabled" class="menu-option">
      <a [routerLink]="[baseHref, 'project', project.slug, 'schedulelist']"
        routerLinkActive="active">
        <span class="menu-option-text" data-animation="text">
          {{'PROJECT.SECTION.ACTIVITIES_LIST' | translate}}
        </span>
      </a>
    </li>
    <li *ngIf="isMenuGanttEnabled" class="menu-option">
      <a [routerLink]="[baseHref, 'project', project.slug, 'gantt']"
        routerLinkActive="active">
        <span class="menu-option-text" data-animation="text">
          {{'PROJECT.SECTION.GANTT' | translate}}
        </span>
      </a>
    </li>
  </ul>
</li>
<!-- ... -->
</ul>

```

Código B.27 – Item de menu de cronograma e gantt no *template* de navegação

A alteração acrescentou um bloco `<li>` com a classe `menu-option` `schedule`, exibido quando `isMenuScheduleGroupEnabled` é verdadeiro. O botão alterna o *dropdown* (`toggleSchedule`) e mostra o ícone *calendar*; o submenu (`child-menu`) traz os subitens *Activities list* (`schedulelist`) e *Gantt*, cada um condicionado à sua permissão (`isMenuScheduleEnabled` e `isMenuGanttEnabled`). Além desse bloco, dois condicionais do diálogo de *popup* (exibido quando o menu está recolhido) foram ajustados para tratar o tipo `schedule` da mesma forma que o tipo `scrum`: o *link* direto passou a excluir `schedule`, e o título sem *link* passou a incluí-lo.

Em atualizações futuras, este *template* é mantido pelo repositório original, e a inclusão ou a remoção de itens de menu o edita, o que o torna um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se o bloco do item de cronograma e os ajustes nos condicionais do diálogo permanecem. Sua ausência remove o acesso ao cronograma e ao gantt pelo menu lateral.

`src/app/commons/project-navigation/project-navigation.component.spec.ts`

O arquivo `project-navigation.component.spec.ts` é a suíte de testes do componente de navegação. A modificação acrescentou a importação do enum `Permissions` e três blocos de teste cobrindo a visibilidade do menu de cronograma e gantt, conforme o Código B.28.

```
import { Permissions } from '@app/api/roles/roles.model';

describe('ProjectNavigationComponent', () => {
  // ...

  describe('schedule menu', () => {
    it('enabled when user has view_schedule permission', () => {
      const project = {
        ...ProjectMockFactory.build(),
        myPermissions: [Permissions.viewSchedule],
      };

      spectator = createHost('<tg-project-navigation [project]="project"></tg-project-navigation>', {
        hostProps: { project },
      });

      expect(spectator.component.isMenuScheduleEnabled).toBeTruthy();
    });
    // ...
  });
  // ...
});
```

Código B.28 – Testes de visibilidade do menu de cronograma e gantt

A modificação acrescentou três blocos `describe` (*schedule menu*, *schedule group menu* e *gantt menu*), além da importação de `Permissions`. Os testes verificam os *getters* de visibilidade (`isMenuScheduleEnabled`, `isMenuGanttEnabled` e `isMenuScheduleGroupEnabled`) sob diferentes combinações de permissão e de ativação do módulo, incluindo o caso de módulo desativado. O Código mostra o primeiro teste como representante; os demais seguem a mesma estrutura.

Em atualizações futuras, esta suíte espelha o componente e é mantida pelo repositório original, sendo sensível a mudanças na estrutura do menu. Ao integrar novas versões, deve-se verificar se os blocos de teste de cronograma permanecem. Por se tratar de arquivo de teste, sua ausência não afeta a aplicação em produção, mas remove a cobertura de regressão da visibilidade do menu.

src/app/commons/project-navigation/project-navigation.component.ts

O arquivo `project-navigation.component.ts` é o controlador do componente de menu lateral. A modificação acrescentou o estado e a lógica de visibilidade do *dropdown* de cronograma e gantt, conforme o Código B.29.

```
export class ProjectNavigationComponent implements OnChanges, OnInit {  
  // ...  
  public scrumVisible = false;  
  public scheduleVisible = false;  
  // ...  
  
  get isMenuScheduleEnabled() {  
    return this.project.isScheduleActivated !== false &&  
      this.project.myPermissions.includes(Permissions.viewSchedule);  
  }  
  get isMenuGanttEnabled() {  
    return this.project.isScheduleActivated !== false &&  
      this.project.myPermissions.includes(Permissions.viewGantt);  
  }  
  get isMenuScheduleGroupEnabled() {  
    return this.isMenuScheduleEnabled || this.isMenuGanttEnabled;  
  }  
  // ...  
  
  public toggleSchedule() {  
    if (this.collapsed) {  
      this.router.navigate(['/project', this.project.slug, 'schedulelist']);  
    } else {  
      this.scheduleVisible = !this.scheduleVisible;  
      localStorage.setItem('projectnav-schedule', String(this.scheduleVisible))  
    }  
  }  
  // ...  
}
```

Código B.29 – *Getters* de visibilidade e alternância do *dropdown* de cronograma

Os *getters* `isMenuScheduleEnabled` e `isMenuGanttEnabled` calculam a visibilidade a partir do objeto `project` (`isScheduleActivated` diferente de `false` combinado com a permissão correspondente), e `isMenuScheduleGroupEnabled` agrega os dois para decidir se o *dropdown* aparece. O método `toggleSchedule` navega direto para a lista de atividades quando o menu está recolhido ou, caso contrário, alterna a visibilidade

do submenu, persistindo o estado em `localStorage`. Além do mostrado, a modificação inclui o campo `scheduleVisible`, a leitura desse estado no `ngOnInit` quando a seção é `schedule` ou `gantt`, o método `popupSchedule` (abre o diálogo quando recolhido), o *getter* `scheduleMenuChildren` (monta os subitens do diálogo) e o reset de `scheduleVisible` ao recolher o menu.

Em atualizações futuras, este componente é mantido pelo repositório original, e a inclusão ou a remoção de itens de menu o edita, o que o torna um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se os *getters*, o estado e os métodos de cronograma permanecem. Sua ausência remove a lógica de visibilidade do menu de cronograma e `gantt`, ou quebra a compilação caso o *template* continue referenciando os *getters* removidos.

`src/assets/i18n/en.json`

O arquivo `src/assets/i18n/en.json` é o dicionário de tradução em inglês do `taiga-front-next`. A modificação acrescentou os rótulos de menu de cronograma e `gantt`, conforme o Código B.30.

```
"SECTION": {  
  "BACKLOG": "Backlog",  
  "KANBAN": "Kanban",  
  "ISSUES": "Issues",  
  "SCHEDULE": "Schedule",  
  "ACTIVITIES_LIST": "Activities list",  
  "GANTT": "Gantt",  
  "WIKI": "Wiki",  
  "TEAM": "Team"  
}
```

Código B.30 – Rótulos de menu de cronograma e `gantt` no *i18n* do `taiga-front-next`

A alteração acrescentou três chaves ao grupo de seção do menu (`SCHEDULE`, `ACTIVITIES_LIST` e `GANTT`), referenciadas pelo *template* de navegação. Vale notar que, na aplicação integrada, a tradução em tempo de execução do *webcomponent* provém das tabelas de *locale* do `taiga-front`; este `en.json` atende ao `taiga-front-next` em execução isolada e ao desenvolvimento.

Em atualizações futuras, o arquivo é mantido pelo repositório original e é editado por praticamente toda nova funcionalidade, sendo foco frequente de conflito. Ao integrar novas versões, deve-se verificar se as três chaves permanecem e se o JSON continua válido. A ausência de uma chave faz a interface exibir o identificador bruto no lugar do texto.

B.2 ÉPICO E2 - GESTÃO FINANCEIRA

B.2.1 Back-end

settings/common.py

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.1: a inclusão de um novo aplicativo Django na lista `INSTALLED_APPS`. No caso do módulo financeiro, o aplicativo registrado é `taiga.projects.finances`, conforme o Código B.31. A justificativa e os pontos de atenção são os mesmos já apresentados naquele bloco.

```
INSTALLED_APPS = [  
    # ...  
    "taiga.projects.wiki",  
    "taiga.projects.contact",  
    "taiga.projects.settings",  
    "taiga.projects.finances",  
    "taiga.searches",  
    # ...  
]
```

Código B.31 – Registro do aplicativo financeiro em `INSTALLED_APPS`

taiga/locale/pt_BR/LC_MESSAGES/django.po

O arquivo `taiga/locale/pt_BR/LC_MESSAGES/django.po` é o catálogo de tradução do *back-end* para português do Brasil. A modificação acrescentou as traduções de mensagens de erro do módulo financeiro que chegam ao usuário, conforme o Código B.32.

```
#: taiga/projects/finances/api.py:226  
msgid ""  
"Planned cost cannot be lower than children total plus categorized own "  
"costs. Minimum allowed is %(minimum)s. ..."  
msgstr ""  
"O custo planejado não pode ser menor que o total dos filhos mais os custos "  
"próprios categorizados. O mínimo permitido é %(minimum)s. ..."  
  
#: taiga/projects/finances/api.py:900  
msgid "The Uncategorized category name cannot be changed."  
msgstr "O nome da categoria Uncategorized não pode ser alterado."
```

Código B.32 – Traduções de mensagens de erro do módulo financeiro

As entradas acrescentadas associam as mensagens originais (`msgid`, em inglês, emitidas pelos testes em `finances/api.py`) às suas traduções (`msgstr`). Sem elas, essas mensagens seriam exibidas em inglês para o usuário em português.

Em atualizações futuras, este catálogo é mantido pelo repositório original e é recompilado com `compilemessages`; é um ponto provável de conflito quando o projeto oficial revisa traduções. Ao integrar novas versões, deve-se verificar se as entradas de finanças permanecem e recompilar o catálogo. Sua ausência não interrompe a execução, mas faz a mensagem aparecer em inglês.

`taiga/permissions/choices.py`

A alteração neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.1: a declaração de novas permissões. No módulo financeiro, foram acrescentadas as permissões de finanças à lista `MEMBERS_PERMISSIONS`, conforme o Código B.33. Diferentemente do cronograma, nenhuma permissão financeira foi adicionada a `ANON_PERMISSIONS`, pois os dados financeiros não são expostos a usuários anônimos. A justificativa e os pontos de atenção são os mesmos apresentados naquele bloco.

```
MEMBERS_PERMISSIONS = [
    # ...
    ('delete_wiki_link', _('Delete wiki link')),
    # Finances permissions
    ('view_finances_activity_cost', _('View activity costs')),
    ('view_finances_planned_hours', _('View planned hours in activity costs')),
    ('view_finances_planned_cost', _('View planned cost in activity costs')),
    ('modify_finances_planned_cost', _('Modify planned cost in activity costs'))
),
    ('view_finances_project_income', _('View project incomes')),
    ('add_finances_project_income', _('Add project incomes')),
    ('view_finances_member_cost', _('View member hourly cost')),
    ('view_finances_dashboard', _('View finance dashboard')),
    ('view_finances_own_time_entries', _('View own activity time entries')),
    # ... (demais permissões de finanças, 23 no total)
]
```

Código B.33 – Permissões de finanças acrescentadas em `MEMBERS_PERMISSIONS`

`taiga/projects/admin.py`

A alteração neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.1: a inclusão de uma *flag* de ativação de módulo no grupo *Modules* de `ProjectAdmin`. No módulo financeiro, o campo é `is_finances_activated`,

conforme o Código B.34. A justificativa e os pontos de atenção são os mesmos daquele bloco.

```
class ProjectAdmin(admin.ModelAdmin):
    # ...
    fieldsets = (
        # ...
        (_("Modules"), {
            "fields": (("is_backlog_activated", "total_milestones", "
total_story_points"),
                    "is_kanban_activated",
                    "is_issues_activated",
                    "is_finances_activated",
                    "is_wiki_activated",
                    ("videoconferences", "videoconferences_extra_data")),
        }),
        # ...
    )
```

Código B.34 – Inclusão de `is_finances_activated` no grupo *Modules*

`taiga/projects/fixtures/initial_project_templates.json`

O arquivo `initial_project_templates.json` define os *templates* de projeto usados na criação de novos projetos, incluindo as permissões padrão de cada papel (*role*). A modificação acrescentou três permissões de finanças ao conjunto de permissões dos papéis, conforme o Código B.35.

```
{
  "permissions": [
    ...
    "comment_issue",
    "comment_wiki_page",
    "view_finances_my_activities",
    "view_finances_own_time_entries",
    "modify_finances_own_time_entry_notes"
  ]
}
```

Código B.35 – Permissões de finanças pré-concedidas nos *templates* de papel

A alteração acrescenta as permissões `view_finances_my_activities`, `view_finances_own_time_entries` e `modify_finances_own_time_entry_notes` ao *array* de permissões dos papéis, repetindo-se em cada papel dos *templates*. O objetivo

é que projetos novos já venham com essas permissões habilitadas por padrão. Os papéis de projetos já existentes recebem as mesmas permissões por meio de migrações de dados específicas.

Em atualizações futuras, este arquivo é mantido pelo repositório original, e a evolução de permissões padrão o edita, o que o torna um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se as permissões de finanças permanecem nos papéis e se o JSON continua válido. Sua ausência faz com que projetos novos não venham com essas permissões pré-habilitadas.

taiga/projects/history/api.py

O arquivo `taiga/projects/history/api.py` expõe os *viewsets* de histórico de mudanças das entidades do Taiga. A modificação acrescentou os *viewsets* de histórico das entidades financeiras (receita de projeto e custo de atividade), conforme o Código B.36.

```
class FinancesHistoryModuleMixin:
    def get_object(self):
        obj = super().get_object()
        if not getattr(obj.project, "is_finances_activated", False):
            raise exc.PermissionDenied(
                _("The finances module is disabled for this project.")
            )
        return obj

class ProjectIncomeHistory(FinancesHistoryModuleMixin, HistoryViewSet):
    content_type = "finances.projectincome"
    permission_classes = (permissions.ProjectIncomeHistoryPermission,)

class ActivityCostHistory(FinancesHistoryModuleMixin, HistoryViewSet):
    content_type = "finances.activitycost"
    permission_classes = (permissions.ActivityCostHistoryPermission,)

    def get_history_queryset(self, obj):
        return services.get_history_queryset_by_model_instance(
            obj, types=(HistoryType.create, HistoryType.change, HistoryType.
delete),
        )
```

Código B.36 – *Viewsets* de histórico das entidades financeiras

As classes `ProjectIncomeHistory` e `ActivityCostHistory` são *viewsets* de leitura que expõem o histórico de mudanças das duas entidades financeiras, seguindo o padrão dos *viewsets* nativos (épico, história, tarefa, issue e wiki) deste arquivo. O *mixin* `FinancesHistoryModuleMixin` nega acesso quando o módulo financeiro está desativado no projeto, e `ActivityCostHistory` sobrescreve a consulta para incluir os eventos de criação, alteração e remoção. A modificação também introduziu, na classe base `HistoryViewSet`, o método `get_history_queryset` (extraído do `retrieve`) para permitir essa sobrescrita.

Em atualizações futuras, este arquivo é mantido pelo repositório original e abriga os *viewsets* de histórico nativos, sendo um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se as classes financeiras e o *hook* `get_history_queryset` permanecem. Sua ausência torna os *endpoints* de histórico financeiro inacessíveis.

`taiga/projects/history/freeze_impl.py`

A modificação neste arquivo tem duas partes. A primeira é da mesma natureza da descrita para o cronograma na Seção B.1.1: a inclusão do campo de ativação do módulo (`is_finances_activated`) na tupla capturada por `project_freezer`. A segunda é nova: a definição de dois *freezers* para as entidades financeiras, conforme o Código B.37.

```
def project_freezer(project) -> dict:
    fields = (# ...
              "is_issues_activated",
              "is_finances_activated")
    return _generic_extract(project, fields)

# ...

def project_income_freezer(income) -> dict:
    snapshot = {
        "title": income.title,
        "amount": str(income.amount) if income.amount is not None else None,
        # ...
    }
    return snapshot

def activity_cost_freezer(activity_cost) -> dict:
    snapshot = {
        "parent_id": activity_cost.parent_id,
        "planned_hours": str(activity_cost.planned_hours) ... ,
        # ...
    }
```

```
return snapshot
```

Código B.37 – Campo de ativação e *freezers* das entidades financeiras

A inclusão de `is_finances_activated` segue o mesmo raciocínio do bloco referenciado. Os *freezers* `project_income_freezer` e `activity_cost_freezer` produzem os retratos (*snapshots*) dos campos relevantes das duas entidades, usados pelo sistema de histórico para registrar mudanças e calcular diferenças entre versões. Esses *freezers* precisam ser registrados no sistema de histórico (ver o bloco de `history/services.py`).

Em atualizações futuras, a tupla de `project_freezer` é mantida pelo repositório original (ponto provável de conflito), enquanto os dois *freezers* novos são adições de baixo risco. Ao integrar novas versões, deve-se verificar se `is_finances_activated` permanece na tupla e se os *freezers* continuam coerentes com os campos das entidades.

`taiga/projects/history/permissions.py`

O arquivo `taiga/projects/history/permissions.py` define as classes de permissão dos *viewsets* de histórico. A modificação acrescentou as classes de permissão das duas entidades financeiras, conforme o Código B.38.

```
class ProjectIncomeHistoryPermission(TaigaResourcePermission):
    retrieve_perms = HasProjectPerm('view_finances_project_income')
    # ...

class ActivityCostHistoryPermission(TaigaResourcePermission):
    retrieve_perms = HasProjectPerm('view_finances_activity_history')
    # ...
```

Código B.38 – Classes de permissão do histórico financeiro

As classes condicionam a leitura do histórico de cada entidade à permissão de visualização correspondente (`view_finances_project_income` e `view_finances_activity_history`), seguindo o padrão das classes de permissão de histórico nativas deste arquivo.

Em atualizações futuras, este arquivo é mantido pelo repositório original, sendo um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se as classes financeiras permanecem e se os codinomes de permissão continuam idênticos aos declarados em `choices.py`. Sua ausência impede a checagem de permissão dos *endpoints* de histórico financeiro.

`taiga/projects/history/services.py`

O arquivo `taiga/projects/history/services.py` registra, no sistema de histórico, qual *freezer* usar para cada tipo de conteúdo. A modificação importou e registrou os

dois *freezers* financeiros, conforme o Código B.39.

```
from .freeze_impl import wikipage_freezer
from .freeze_impl import project_income_freezer
from .freeze_impl import activity_cost_freezer

# ...

register_freeze_implementation("finances.projectincome", project_income_freezer
    )
register_freeze_implementation("finances.activitycost", activity_cost_freezer)
```

Código B.39 – Registro dos *freezers* financeiros no sistema de histórico

A alteração compreende a importação dos dois *freezers* (definidos em `freeze_impl.py`) e seu registro por tipo de conteúdo (`finances.projectincome` e `finances.activitycost`), de modo que o sistema de histórico saiba como capturar o estado dessas entidades a cada mudança.

Em atualizações futuras, este arquivo é mantido pelo repositório original, sendo um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se as importações e os registros permanecem e se as cadeias de tipo continuam coerentes com os *freezers*. Sua ausência desabilita o registro do histórico das entidades financeiras.

`taiga/projects/models.py`

A alteração neste arquivo é da mesma natureza da descrita para o cronograma na Seção B.1.1: a declaração do campo de ativação do módulo (`is_finances_activated`) nos modelos `Project` e `ProjectTemplate` e sua propagação nos dois métodos de cópia, conforme o Código B.40. A justificativa e os pontos de atenção são os mesmos daquele bloco.

```
class Project(ProjectDefaults, TaggedMixin, TagsColorsMixin, models.Model):
    # ...
    is_issues_activated = models.BooleanField(default=True, null=False, blank=True,
                                              verbose_name=_("active issues
panel"))
    is_finances_activated = models.BooleanField(default=True, null=False, blank=True,
                                              verbose_name=_("active finances
panel"))
    # ...
```

```

class ProjectTemplate(TaggedMixin, TagsColorsMixin, models.Model):
    # ...
    is_finances_activated = models.BooleanField(default=True, null=False, blank
    =True,
                                                    verbose_name=_("active finances
    panel"))
    # ...

    def load_data_from_project(self, project):
        # ...
        self.is_finances_activated = project.is_finances_activated
        # ...

    def apply_to_project(self, project):
        # ...
        project.is_finances_activated = self.is_finances_activated
        # ...

```

Código B.40 – Campo `is_finances_activated` em `Project` e `ProjectTemplate`

taiga/projects/serializers.py

A alteração neste arquivo é da mesma natureza da descrita para o cronograma na Seção B.1.1: a exposição do campo de ativação do módulo (`is_finances_activated`) nos três serializadores de projeto, conforme o Código B.41. A justificativa e os pontos de atenção são os mesmos daquele bloco.

```

class ProjectSerializer(serializers.LightSerializer):
    # ...
    is_issues_activated = Field()
    is_finances_activated = Field()
    # ...

class ProjectLightSerializer(serializers.LightSerializer):
    # ...
    is_finances_activated = Field()
    # ...

class ProjectTemplateSerializer(serializers.LightSerializer):
    # ...
    is_finances_activated = Field()
    # ...

```

Código B.41 – Exposição de `is_finances_activated` nos serializadores de projeto

`taiga/routers.py`

A alteração neste arquivo é da mesma natureza da descrita para o cronograma na Seção B.1.1: a importação e o registro de *viewsets* no *router* global. No módulo financeiro, foram registrados os *viewsets* das telas financeiras e os *viewsets* de histórico das entidades financeiras, conforme o Código B.42. A justificativa e os pontos de atenção são os mesmos daquele bloco.

```
# ...
from taiga.projects.finances.api import ActivityCostViewSet
from taiga.projects.finances.api import MemberHourlyRateViewSet
from taiga.projects.finances.api import MyActivitiesViewSet
from taiga.projects.finances.api import ProjectFinanceDashboardViewSet
from taiga.projects.finances.api import ProjectIncomeViewSet
# ...
router.register(r"activity-costs", ActivityCostViewSet, base_name="activity-
    costs")
router.register(r"member-hourly-rates", MemberHourlyRateViewSet, base_name="
    member-hourly-rates")
router.register(r"my-activities", MyActivitiesViewSet, base_name="my-activities
    ")
router.register(r"project-incomes", ProjectIncomeViewSet, base_name="project-
    incomes")
router.register(r"project-finance-dashboard", ProjectFinanceDashboardViewSet,
    base_name="project-finance-dashboard")
# ...
from taiga.projects.history.api import ProjectIncomeHistory
from taiga.projects.history.api import ActivityCostHistory
# ...
router.register(r"history/project-income", ProjectIncomeHistory, base_name="
    project-income-history")
router.register(r"history/activity-cost", ActivityCostHistory, base_name="
    activity-cost-history")
```

Código B.42 – Registro das rotas da API e do histórico do módulo financeiro

`taiga/timeline/service.py`

O arquivo `taiga/timeline/service.py` reúne as funções que extraem, de cada entidade, os dados exibidos na *timeline* do projeto. A modificação acrescentou duas funções

de extração para as entidades financeiras, conforme o Código B.43.

```
def extract_project_income_info(instance):
    return {
        "id": instance.pk,
        "ref": instance.ref,
        "title": instance.title,
    }

def extract_activity_cost_info(instance):
    # ... resolve entity_type, ref e subject (task, user story ou epic)
    return {
        "id": instance.pk,
        "entity_type": entity_type,
        "ref": ref,
        "subject": subject,
        # ... agregados financeiros
    }
```

Código B.43 – Extratores de *timeline* das entidades financeiras

As funções `extract_project_income_info` e `extract_activity_cost_info` montam o conteúdo da *timeline* para receita de projeto e custo de atividade, respectivamente. São consumidas pelas implementações de *timeline* (ver o bloco de `timeline_implementations.py`).

Em atualizações futuras, este arquivo é mantido pelo repositório original e abriga os extratores nativos, sendo um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se os dois extratores permanecem e se continuam coerentes com os campos das entidades. Sua ausência quebra a geração das entradas de *timeline* financeiras.

`taiga/timeline/timeline_implementations.py`

O arquivo `taiga/timeline/timeline_implementations.py` registra, por tipo de conteúdo e evento, como montar cada entrada de *timeline*. A modificação acrescentou as implementações de *timeline* das duas entidades financeiras, conforme o Código B.44.

```
@register_timeline_implementation("finances.projectincome", "create")
@register_timeline_implementation("finances.projectincome", "change")
@register_timeline_implementation("finances.projectincome", "delete")
def project_income_timeline(instance, extra_data={}):
    result = {
        "projectincome": service.extract_project_income_info(instance),
        "project": service.extract_project_info(instance.project),
```

```

    }
    result.update(extra_data)
    return result

@register_timeline_implementation("finances.activitycost", "create")
# ... change, delete
def activity_cost_timeline(instance, extra_data={}):
    result = {
        "activitycost": service.extract_activity_cost_info(instance),
        "project": service.extract_project_info(instance.project),
    }
    result.update(extra_data)
    return result

```

Código B.44 – Implementações de *timeline* das entidades financeiras

As funções registram as implementações de *timeline* para os eventos de criação, alteração e remoção das duas entidades financeiras, montando a entrada a partir dos extratores de `service.py`. Esse registro é necessário em conjunto com o do histórico: ao se tirar um *snapshot* de uma entidade, o sinal que alimenta a *timeline* procura a implementação registrada para o evento; sem ela, a operação falha. Por isso, habilitar histórico para uma entidade nova exige registrar tanto o *freezer* quanto a implementação de *timeline*.

Em atualizações futuras, este arquivo é mantido pelo repositório original, sendo um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se as implementações financeiras permanecem registradas, em conjunto com os *freezers*. Sua ausência quebra a gravação de histórico das entidades financeiras.

B.2.2 Front-end

`app/coffee/app.coffee`

A alteração neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.2: o registro de rotas e de módulos AngularJS no *bootstrap* do *front-end*. No módulo financeiro, foram acrescentadas rotas para custo de atividade, custo por membro, minhas atividades, receitas de projeto e *dashboard* financeiro, além dos módulos Angular correspondentes, conforme o Código B.45. A justificativa geral e os pontos de atenção são os mesmos daquele bloco, com a diferença de que a alteração financeira envolve várias telas independentes em vez de apenas lista de cronograma e gantt.

```
configure = ($routeProvider, ...) ->
```

```
# ...

# Activity cost
$routeProvider.when("/project/:pslug/activity-cost",
    { templateUrl: "activity-cost/activity-cost.html", loader: true,
    section: "activity-cost" }
)
$routeProvider.when("/project/:pslug/activity-cost/:activityid",
    { templateUrl: "activity-cost/activity-cost-detail.html", loader: true,
    section: "activity-cost" }
)

# Member cost
$routeProvider.when("/project/:pslug/member-cost",
    { templateUrl: "member-cost/member-cost.html", loader: true, section: "
member-cost" }
)

# My activities
$routeProvider.when("/project/:pslug/my-activities",
    { templateUrl: "my-activities/my-activities.html", loader: true,
    section: "my-activities" }
)
$routeProvider.when("/project/:pslug/my-activities/:activityid",
    { templateUrl: "activity-cost/activity-cost-detail.html", loader: true,
    section: "my-activities" }
)

# Project income
$routeProvider.when("/project/:pslug/project-income",
    { templateUrl: "project-income/project-income.html", loader: true,
    section: "project-income" }
)
$routeProvider.when("/project/:pslug/project-income/:incomeref",
    { templateUrl: "project-income/project-income-detail.html", loader:
true, section: "project-income" }
)

# Project dashboard
$routeProvider.when("/project/:pslug/dashboard",
    { templateUrl: "project-dashboard/project-dashboard.html", loader: true
, section: "dashboard" }
)
```

```

modules = [
  # ...
  "taigaTeam",
  "taigaActivityCost",
  "taigaMemberCost",
  "taigaMyActivities"
  "taigaProjectIncome"
  "taigaWiki",
  # ...
]

```

Código B.45 – Rotas e módulos financeiros no *bootstrap* da aplicação

`app/coffee/modules/admin/roles.coffee`

A alteração neste arquivo é análoga à inclusão da categoria de cronograma e gantt descrita na Seção B.1.2: a tela de administração de papéis passa a exibir uma categoria própria para permissões financeiras. No caso de CUST, a alteração não adiciona permissões financeiras a `publicPermissions` nem a `DEFAULT_PERMISSIONS`; ela apenas torna configuráveis, na interface, as permissões financeiras já definidas pelo *back-end*, conforme o Código B.46.

```

# ...
financesPermissions = [
  { key: "view_finances_activity_cost", name: "COMMON.
PERMISSIONS_CATEGORIES.FINANCES.VIEW_ACTIVITY_COSTS"}
  { key: "view_finances_planned_hours", name: "COMMON.
PERMISSIONS_CATEGORIES.FINANCES.VIEW_PLANNED_HOURS"}
  { key: "view_finances_actual_hours", name: "COMMON.
PERMISSIONS_CATEGORIES.FINANCES.VIEW_ACTUAL_HOURS"}
  { key: "view_finances_planned_cost", name: "COMMON.
PERMISSIONS_CATEGORIES.FINANCES.VIEW_PLANNED_COST"}
  { key: "view_finances_actual_cost", name: "COMMON.
PERMISSIONS_CATEGORIES.FINANCES.VIEW_ACTUAL_COST"}
  { key: "modify_finances_planned_hours", name: "COMMON.
PERMISSIONS_CATEGORIES.FINANCES.MODIFY_PLANNED_HOURS"}
  { key: "modify_finances_planned_cost", name: "COMMON.
PERMISSIONS_CATEGORIES.FINANCES.MODIFY_PLANNED_COST"}
  { key: "modify_finances_activity_categorization", name: "COMMON.
.PERMISSIONS_CATEGORIES.FINANCES.MODIFY_ACTIVITY_CATEGORIZATION"}
  { key: "manage_finances_project_categories", name: "COMMON.
PERMISSIONS_CATEGORIES.FINANCES.MANAGE_PROJECT_CATEGORIES"}

```

<code>{ key: "view_finances_project_income", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.VIEW_PROJECT_INCOME"}</code>
<code>{ key: "add_finances_project_income", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.ADD_PROJECT_INCOME"}</code>
<code>{ key: "modify_finances_project_income", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.MODIFY_PROJECT_INCOME"}</code>
<code>{ key: "delete_finances_project_income", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.DELETE_PROJECT_INCOME"}</code>
<code>{ key: "view_finances_member_cost", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.VIEW_MEMBER_COST"}</code>
<code>{ key: "modify_finances_member_cost", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.MODIFY_MEMBER_COST"}</code>
<code>{ key: "view_finances_dashboard", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.VIEW_DASHBOARD"}</code>
<code>{ key: "view_finances_my_activities", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.VIEW_MY_ACTIVITIES"}</code>
<code>{ key: "view_finances_activity_history", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.VIEW_ACTIVITY_HISTORY"}</code>
<code>{ key: "view_finances_all_time_entries", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.VIEW_ALL_TIME_ENTRIES"}</code>
<code>{ key: "view_finances_own_time_entries", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.VIEW_OWN_TIME_ENTRIES"}</code>
<code>{ key: "modify_finances_all_time_entries", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.MODIFY_ALL_TIME_ENTRIES"}</code>
<code>{ key: "modify_finances_own_time_entries", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.MODIFY_OWN_TIME_ENTRIES"}</code>
<code>{ key: "modify_finances_own_time_entry_notes", name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.MODIFY_OWN_TIME_ENTRY_NOTES"}</code>
<code>]</code>
<code>categories.push({</code>
<code> name: "COMMON.PERMISIONS_CATEGORIES.FINANCES.NAME",</code>
<code> permissions: setActivePermissions(financesPermissions)</code>
<code>})</code>

Código B.46 – Categoria de permissões financeiras na tela de papéis

`app/coffee/modules/base.coffee`

A alteração neste arquivo é da mesma natureza da descrita para cronograma na Seção B.1.2: a inclusão de chaves de navegação no mapa `urls`. No módulo financeiro, foram acrescentadas oito chaves `project-*`, correspondentes às telas financeiras e aos

seus detalhes, conforme o Código B.47. A justificativa e os pontos de atenção são os mesmos daquele bloco.

```
urls = {
    # ...
    "project-issues": "/project/:project/issues"
    "project-epics": "/project/:project/epics"
    "project-activity-cost": "/project/:project/activity-cost"
    "project-activity-cost-detail": "/project/:project/activity-cost/:
activityid"
    "project-member-cost": "/project/:project/member-cost"
    "project-my-activities": "/project/:project/my-activities"
    "project-my-activities-detail": "/project/:project/my-activities/:
activityid"
    "project-income": "/project/:project/project-income"
    "project-income-detail": "/project/:project/project-income/:incomeref"
    "project-dashboard": "/project/:project/dashboard"
    "project-search": "/project/:project/search"
    # ...
}
```

Código B.47 – Rotas de navegação financeiras no mapa urls

app/coffee/modules/resources.coffee

A alteração neste arquivo é da mesma natureza da descrita para cronograma na Seção B.1.2: a inclusão de chaves no mapa de recursos da API. No módulo financeiro, foram acrescentados recursos para custos de atividade, registros de tempo, distribuição de custo planejado, categorias, custo por membro, minhas atividades, receitas, *dashboard*, cronômetro e histórico financeiro, conforme o Código B.48. A justificativa e os pontos de atenção são os mesmos daquele bloco.

```
urls = {
    # ...
    # Finances
    "activity-costs": "/activity-costs"
    "activity-cost-time-entries": "/activity-costs/%s/time_entries"
    "activity-cost-planned-cost-distribution": "/activity-costs/%s/
planned_cost_distribution"
    "activity-cost-categories": "/activity-costs/%s/categories"
    "member-hourly-rates": "/member-hourly-rates"
    "member-hourly-rates-working-now": "/member-hourly-rates/working_now"
    "my-activities": "/my-activities"
```

```

"project-incomes": "/project-incomes"
"project-finance-dashboard": "/project-finance-dashboard"
"my-activities-timer-state": "/my-activities/timer_state"
"my-activities-start-timer": "/my-activities/start_timer"
"my-activities-stop-timer": "/my-activities/stop_timer"

# ...
"history/project-income": "/history/project-income"
"history/activity-cost": "/history/activity-cost"
# ...
}

```

Código B.48 – *Endpoints* da API financeira no mapa de recursos

app/locales/taiga/locale-en.json e app/locales/taiga/locale-pt-br.json

A alteração nestes arquivos é da mesma natureza da descrita para cronograma na Seção B.1.2: inclusão de chaves de tradução em inglês e português do Brasil para a interface do novo módulo. No caso financeiro, os dicionários receberam textos para permissões, administração de módulos, menus, telas de custo de atividade, custo por membro, receitas de projeto, minhas atividades, *dashboard* financeiro e mensagens de *timeline*, conforme os Códigos B.49 e B.50. Além desses acréscimos, o arquivo em inglês remove uma duplicação da chave `COMMON.MOVE_TO_TOP`, mantendo apenas uma ocorrência do mesmo rótulo.

```

"PERMISSIONS_CATEGORIES": {
  "FINANCES": {
    "NAME": "Finances",
    "VIEW_ACTIVITY_COSTS": "View activity costs",
    "VIEW_PROJECT_INCOME": "View project incomes",
    "VIEW_DASHBOARD": "View finance dashboard",
    "VIEW_MY_ACTIVITIES": "View my activities page"
  }
}
"MODULES": {
  "FINANCES": "Finances",
  "FINANCES_DESCRIPTION": "Track planned costs, actual costs, incomes and
project financial dashboards."
}
"ACTIVITY_COST": {
  "DETAIL": { "SECTION_NAME": "Financial activity detail" },
  "TABLE": { "COLUMNS": { "PLANNED_COST": "Planned cost", "ACTUAL_COST": "
Actual cost" } }
}

```

```

}
"MEMBER_COST": { "TABLE": { "COLUMNS": { "HOURLY_RATE": "Cost / hour" } } }
"PROJECT_INCOME": { "SECTION_NAME": "Income", "ACTION_DELETE": "Delete income"
}
"MY_ACTIVITIES": { "TIMER": { "PLAY": "Play", "STOP": "Stop" } }
"PROJECT_DASHBOARD": {
  "SUMMARY": { "TOTAL_PLANNED_COST": "Total planned cost", "
CURRENT_CASH_BALANCE": "Current cash balance" }
}
"PROJECT": {
  "SECTION": { "FINANCES": "Finances", "ACTIVITY_COST": "Activity Cost", "
MEMBER_COST": "Member Cost", "MY_ACTIVITIES": "My Activities", "
PROJECT_INCOME": "Project Income", "DASHBOARD": "Dashboard" }
}
"TIMELINE": {
  "PROJECT_INCOME_CREATED": "{{username}} has created a new income {{obj_name
}} in {{project_name}}",
  "ACTIVITY_COST_TIME_ENTRY_EVENT": "{{username}} has {{
time_entry_event_action}} a time entry in {{obj_name}}"
}

```

Código B.49 – Trecho das chaves de tradução do módulo financeiro (arquivo em inglês)

```

"PERMISSIONS_CATEGORIES": {
  "FINANCES": {
    "NAME": "Finanças",
    "VIEW_ACTIVITY_COSTS": "Visualizar custos de atividades",
    "VIEW_PROJECT_INCOME": "Visualizar receitas do projeto",
    "VIEW_DASHBOARD": "Visualizar dashboard financeiro",
    "VIEW_MY_ACTIVITIES": "Visualizar página \"Minhas atividades\""
  }
}
"MODULES": {
  "FINANCES": "Finanças",
  "FINANCES_DESCRIPTION": "Acompanhe custos planejados, custos reais,
receitas e dashboards financeiros do projeto."
}
"ACTIVITY_COST": {
  "DETAIL": { "SECTION_NAME": "Detalhamento financeiro da atividade" },
  "TABLE": { "COLUMNS": { "PLANNED_COST": "Custo planejado", "ACTUAL_COST": "
Custo realizado" } }
}
"MEMBER_COST": { "TABLE": { "COLUMNS": { "HOURLY_RATE": "Custo / hora" } } }

```

```

"PROJECT_INCOME": { "SECTION_NAME": "Receita", "ACTION_DELETE": "Excluir
    receita" }
"MY_ACTIVITIES": { "TIMER": { "PLAY": "Play", "STOP": "Stop" } }
"PROJECT_DASHBOARD": {
    "SUMMARY": { "TOTAL_PLANNED_COST": "Custo planejado total", "
    CURRENT_CASH_BALANCE": "Saldo atual em caixa" }
}
"PROJECT": {
    "SECTION": { "FINANCES": "Finanças", "ACTIVITY_COST": "Custo de Atividade",
    "MEMBER_COST": "Custo por Membro", "MY_ACTIVITIES": "Minhas Atividades", "
    PROJECT_INCOME": "Receita do Projeto", "DASHBOARD": "Dashboard" }
}
"TIMELINE": {
    "PROJECT_INCOME_CREATED": "{{username}} criou uma nova receita {{obj_name}}
    em {{project_name}}",
    "ACTIVITY_COST_TIME_ENTRY_EVENT": "{{username}} {{time_entry_event_action}}
    um registro de tempo em {{obj_name}}"
}

```

Código B.50 – Trecho das chaves de tradução do módulo financeiro (arquivo em português do Brasil)

app/partials/admin/admin-project-modules.jade

A alteração neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.2: a inclusão de um *card* de ativação na página de módulos do projeto. No caso financeiro, o *card* usa a *flag* `project.is_finances_activated`, os textos `ADMIN.MODULES.FINANCES` e `ADMIN.MODULES.FINANCES_DESCRIPTION`, e um ícone SVG de moedas definido diretamente no *template*, conforme o Código B.51.

```

div.wrapper(...)
    // ...
    .module.module-finances(ng-class="{true:'active', false:''}[project.
    is_finances_activated]")
        .module-icon
            svg(...)
                path(stroke="none" d="M0 0h24v24H0z" fill="none")
                path(d="M9 14c0 1.657 2.686 3 6 3s6 -1.343 6 -3s-2.686 -3 -6 -3
                s-6 1.343 -6 3")
                path(d="M9 14v4c0 1.656 2.686 3 6 3s6 -1.344 6 -3v-4")
                path(d="M3 6c0 1.072 1.144 2.062 3 2.598s4.144 .536 6 0c1.856
                -.536 3 -1.526 3 -2.598")
                path(d="M3 6v10c0 .888 .772 1.45 2 2")

```

```

        path(d="M3 11c0 .888 .772 1.45 2 2")
    .module-name(translate="ADMIN.MODULES.FINANCES")
    .module-desc(translate="ADMIN.MODULES.FINANCES_DESCRIPTION")
    .module-activation.module-direct-active
    div.check
        input.activate-input(
            id="functionality-finances"
            name="functionality-finances"
            type="checkbox"
            ng-checked="project.is_finances_activated"
            ng-model="project.is_finances_activated"
        )
    div
        span.check-text.check-yes(translate="COMMON.YES")
        span.check-text.check-no(translate="COMMON.NO")
    .module.module-wiki(...)
    // ...

```

Código B.51 – *Card* de ativação do módulo financeiro na administração de módulos

app/styles/modules/admin/admin-functionalities.scss

O arquivo `app/styles/modules/admin/admin-functionalities.scss` define os estilos da página de funcionalidades/módulos do projeto. A modificação acrescentou uma regra específica para `.module-finances`, conforme o Código B.52.

```

.project-values {
    // ...
    .module-finances {
        &.active .module-icon .icon,
        .module-icon .icon,
        .module-icon .icon path {
            fill: none;
        }
    }
    .module-videoconference {
        // ...
    }
}

```

Código B.52 – Ajuste do preenchimento do ícone do módulo financeiro

A regra é necessária porque o *card* financeiro usa um SVG de linha (*outline*) inserido diretamente no *template*. Os estilos gerais dos módulos podem aplicar preenchimento

aos ícones em alguns estados; para esse SVG, o preenchimento deve permanecer `none`, preservando o desenho por traços e evitando que o ícone de moedas seja renderizado como forma completamente preenchida.

Em atualizações futuras, este arquivo deve ser conferido junto ao *template* do *card* financeiro. Se o ícone passar a usar o *sprite* comum ou outro componente, esta exceção pode deixar de ser necessária; se o SVG inline permanecer, a regra precisa continuar escopada a `.module-finances` para não alterar a aparência dos demais módulos.

`app/styles/modules/admin/admin-roles.scss`

O arquivo `app/styles/modules/admin/admin-roles.scss` define os estilos da tela de administração de papéis. A modificação acrescentou uma exceção para as listas abertas de permissões dentro de `.admin-roles`, conforme o Código B.53.

```
.admin-roles {  
  // ...  
  .category-config {  
    .category-items.open {  
      max-height: none;  
      overflow: visible;  
      transition: none;  
    }  
  }  
}
```

Código B.53 – Abertura sem limite de altura para categorias de permissões em papéis

A alteração decorre do tamanho da categoria financeira adicionada em `roles.coffee`. Como essa categoria reúne muitas permissões, o limite de altura e o recorte definidos pelo componente comum de categorias podem ocultar parte da lista quando a seção está aberta. O ajuste remove o limite de altura e o recorte apenas na tela de papéis, mantendo a alteração escopada ao uso real que demanda a lista longa.

Em atualizações futuras, este ponto deve ser reavaliado se o componente de permissões for redesenhado ou se a categoria financeira for dividida. O cuidado principal é manter o ajuste restrito a `.admin-roles`, para não modificar globalmente o comportamento de `.category-config` em outras telas.

`elements.js`

A alteração neste arquivo é da mesma natureza da descrita para cronograma na Seção B.1.2: `elements.js` é um *bundle* minificado gerado a partir do `taiga-front-next`, e a diferença registrada corresponde à recompilação desse artefato após as mudanças de navegação do módulo financeiro. Como o arquivo ocupa uma única linha minificada, não

há trecho legível a destacar; o Código B.54 mostra apenas uma amostra ilustrativa do início do *bundle*.

```
!function(e){function r(r){for(var n,l,f=r[0],i=r[1],p=r[2],c=0,s=[]; ... }
```

Código B.54 – Início do *bundle* minificado `elements.js` (amostra ilustrativa)

`gulpfile.js`

O arquivo `gulpfile.js` define, entre outras coisas, quais bibliotecas de terceiros são concatenadas no *bundle* de bibliotecas do `taiga-front`. A modificação acrescentou o arquivo UMD do `Chart.js` à lista `paths.libs`, conforme o Código B.55.

```
paths.libs = [
  // ...
  paths.modules + "angular-translate-loader-static-files/angular-translate-
loader-static-files.js",
  paths.modules + "angular-translate-interpolation-messageformat/angular-
translate-interpolation-messageformat.js",
  paths.modules + "moment/moment.js",
  paths.modules + "chart.js/dist/chart.umd.js",
  paths.modules + "checksley/checksley.js",
  paths.modules + "pikaday/pikaday.js",
  // ...
]
```

Código B.55 – Inclusão do `Chart.js` no *bundle* de bibliotecas

A inclusão disponibiliza `window.Chart` para o código legado que renderiza gráficos financeiros, especialmente no detalhamento de custo de atividade e no *dashboard* financeiro. Como essas telas são CoffeeScript/AngularJS e não importam módulos ES diretamente, a biblioteca precisa entrar no *bundle* global de bibliotecas.

Em atualizações futuras, este arquivo deve ser mantido coerente com `package.json` e `package-lock.json`. A dependência instalada precisa conter `dist/chart.umd.js`; se a versão do `Chart.js` ou o formato de distribuição mudar, este caminho deve ser revisto para evitar que o *build* gere um *bundle* sem a variável global `Chart`.

`package-lock.json`

O arquivo `package-lock.json` registra a resolução exata das dependências NPM do `taiga-front`. A modificação materializa a dependência `chart.js` declarada em `package.json`, incluindo sua dependência transitiva `@kurkle/color`, além de atualizar a versão raiz do *lockfile* para 6.9.1, conforme o Código B.56.

```
{
```

```
"name": "TaigaIO",
"version": "6.9.1",
"packages": {
  "": {
    "dependencies": {
      "bourbon": "git+https://github.com/thoughtbot/bourbon.git",
      "chart.js": "^4.5.1",
      "checksley": "git+https://github.com/jespino/checksley-bower.git"
    }
  },
  "node_modules/@kurkle/color": {
    "version": "0.3.4",
    "license": "MIT"
  },
  "node_modules/chart.js": {
    "version": "4.5.1",
    "dependencies": { "@kurkle/color": "^0.3.0" }
  }
}
```

Código B.56 – Resolução do Chart.js e de sua dependência transitiva no `package-lock.json`

Esse arquivo não deve ser editado manualmente: ele é resultado da resolução feita pelo NPM. Sua presença garante que outros ambientes instalem a mesma versão de Chart.js (4.5.1) e a mesma dependência transitiva, evitando diferenças de comportamento nos gráficos financeiros entre máquinas.

Em atualizações futuras, qualquer alteração em `package.json` deve ser acompanhada de uma regeneração coerente do `package-lock.json`. Conflitos neste arquivo devem ser resolvidos preservando a entrada de `chart.js`, a entrada de `@kurkle/color` e a consistência entre a versão raiz do *lockfile* e o manifesto.

`package.json`

O arquivo `package.json` é o manifesto de dependências do `taiga-front`. A modificação acrescentou `chart.js` às dependências da aplicação, conforme o Código B.57.

```
"dependencies": {
  // ...
  "awesomplete": "^1.0.0",
  "bourbon": "git+https://github.com/thoughtbot/bourbon.git",
  "chart.js": "^4.5.1",
```

```

    "checksley": "git+https://github.com/jespino/checksley-bower.git",
    "dom-autoscroll": "^2.3.4",
    // ...
  }

```

Código B.57 – Dependência Chart.js no manifesto NPM

A dependência é necessária porque os gráficos do detalhamento financeiro e do *dashboard* usam a API global do Chart.js após a biblioteca ser incluída no *bundle* pelo *gulpfile.js*. Sem essa entrada, a instalação de dependências não disponibilizaria o pacote em *node_modules*, e o caminho *node_modules/chart.js/dist/chart.umd.js* usado no *build* não existiria.

Em atualizações futuras, *package.json*, *package-lock.json* e *gulpfile.js* devem ser integrados como um conjunto. Remover ou alterar a versão de *chart.js* exige verificar se as telas financeiras continuam compatíveis com a API da biblioteca e se o arquivo UMD usado no *bundle* permanece disponível.

B.2.3 Front-end-Next

src/app/api/project-templates/project-templates.model.mock.ts

O arquivo *project-templates.model.mock.ts* define a *factory* usada nos testes para construir objetos de *template* de projeto. A modificação acrescentou o campo *isFinancesActivated* à *factory*, conforme o Código B.58.

```

export const ProjectTemplateMockFactory = Factory.Sync.makeFactory<
  ProjectTemplate>({
    // ...
    isBacklogActivated: faker.random.boolean(),
    isContactActivated: faker.random.boolean(),
    isEpicsActivated: faker.random.boolean(),
    isFinancesActivated: faker.random.boolean(),
    isIssuesActivated: faker.random.boolean(),
    isKanbanActivated: faker.random.boolean(),
    // ...
  });

```

Código B.58 – Campo *isFinancesActivated* na *factory* de *mock* de *template* de projeto

A alteração é necessária porque o tipo *ProjectTemplate* passou a declarar a *flag* de ativação do módulo financeiro. Sem esse campo, os testes que constroem *templates* por meio da *factory* poderiam gerar objetos incompletos em relação ao tipo TypeScript.

Em atualizações futuras, este arquivo deve acompanhar a interface *ProjectTemplate*. Se *isFinancesActivated* for removido do tipo ou renomeado pela API, a *factory* precisa

ser atualizada junto; caso contrário, os testes deixam de refletir o contrato real dos *templates* de projeto.

`src/app/api/project-templates/project-templates.model.ts`

O arquivo `project-templates.model.ts` define a interface TypeScript dos *templates* de projeto. A modificação acrescentou o campo `isFinancesActivated`, contraparte em *camelCase* da *flag* `is_finances_activated` exposta pela API, conforme o Código B.59.

```
export interface ProjectTemplate {  
  // ...  
  isBacklogActivated: boolean;  
  isContactActivated: boolean;  
  isEpicsActivated: boolean;  
  isFinancesActivated: boolean;  
  isIssuesActivated: boolean;  
  isKanbanActivated: boolean;  
  // ...  
}
```

Código B.59 – Campo `isFinancesActivated` no tipo `ProjectTemplate`

A declaração mantém o tipo do *taiga-front-next* alinhado ao contrato do *back-end* e permite que telas ou serviços que consomem *templates* reconheçam a ativação do módulo financeiro de forma tipada.

Em atualizações futuras, este arquivo deve ser conferido junto aos serializadores de projeto e *template* do *back-end*. A ausência do campo não afeta diretamente a navegação de um projeto já criado, mas quebra a consistência do modelo TypeScript com os dados retornados para *templates*.

`src/app/api/projects/projects.model.mock.ts`

A alteração neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.3: atualização da *factory* de projetos para refletir uma nova *flag* de módulo. No caso financeiro, foram acrescentados `isFinancesActivated` e uma permissão financeira padrão nos *mocks*, conforme o Código B.60.

```
export const ProjectMockFactory = Factory.Sync.makeFactory<Project>({  
  // ...  
  isBacklogActivated: faker.random.boolean(),  
  isContactActivated: faker.random.boolean(),  
  isEpicsActivated: faker.random.boolean(),  
  isFinancesActivated: faker.random.boolean(),  
  isIssuesActivated: faker.random.boolean(),  
})
```

```
isKanbanActivated: faker.random.boolean(),
// ...
myPermissions: [
  Permissions.viewIssues,
  Permissions.viewWikiLinks,
  Permissions.viewWikiPages,
  Permissions.viewProject,
  Permissions.viewFinancesActivityCost,
],
// ...
});
```

Código B.60 – Campo financeiro e permissão padrão na *factory* de *mock* de projeto

src/app/api/projects/projects.model.ts

A alteração neste arquivo é da mesma natureza da descrita para cronograma na Seção B.1.3: inclusão de uma *flag* de ativação de módulo no tipo de projeto. No módulo financeiro, o campo é `isFinancesActivated`, declarado tanto em `ProjectListEntry` quanto em `Project`, conforme o Código B.61.

```
export type ProjectListEntry = Pick<Project,
// ...
'id' |
'isBacklogActivated' |
'isContactActivated' |
'isEpicsActivated' |
'isFinancesActivated' |
'isFan' |
// ...
>;

export interface Project {
// ...
id: number;
isBacklogActivated: boolean;
isContactActivated: boolean;
isEpicsActivated: boolean;
isFinancesActivated: boolean;
isFan: boolean;
// ...
}
```

Código B.61 – Campo `isFinancesActivated` no tipo `Project`

`src/app/api/roles/roles.model.ts`

A alteração neste arquivo é da mesma natureza da descrita para cronograma na Seção B.1.3: inclusão de codinomes de permissão no enum `Permissions`. No módulo financeiro, foram acrescentadas cinco permissões de visualização usadas pelo menu lateral, conforme o Código B.62.

```
export enum Permissions {  
  // ...  
  removeMember = 'remove_member',  
  adminRoles = 'admin_roles',  
  // Finances  
  viewFinancesActivityCost = 'view_finances_activity_cost',  
  viewFinancesProjectIncome = 'view_finances_project_income',  
  viewFinancesMemberCost = 'view_finances_member_cost',  
  viewFinancesDashboard = 'view_finances_dashboard',  
  viewFinancesMyActivities = 'view_finances_my_activities',  
}
```

Código B.62 – Permissões financeiras no enum `Permissions`

`src/app/commons/project-navigation/project-navigation.component.css`

A alteração neste arquivo é análoga à extensão dos estilos de *dropdown* descrita para cronograma na Seção B.1.3. No módulo financeiro, o seletor `.finances` passa a compartilhar o comportamento de `.scrum`, e o SVG de linha usado no ícone financeiro recebe exceções para manter `fill: none`, conforme o Código B.63.

```
:host(.collapsed) {  
  width: 48px;  
  
  & .scrum .child-menu,  
  & .finances .child-menu {  
    display: none;  
  }  
}  
  
.menu-option {  
  & .icons-tabler-outline {  
    fill: none;  
  }  
}
```

```

}

& > a:hover > svg.icons-tabler-outline,
& > button:hover > svg.icons-tabler-outline {
  fill: none;
}
}

.scrum,
.finances {
  &.active {
    & .agile-chevron {
      transform: rotate(90deg);
    }
  }
}
}

```

Código B.63 – Estilos do *dropdown* financeiro no menu lateral

src/app/commons/project-navigation/project-navigation.component.html

A alteração neste arquivo é análoga à inclusão do *dropdown* de cronograma descrita na Seção B.1.3: o menu lateral passa a ter um grupo expansível para finanças. No caso financeiro, o *dropdown* contém custo de atividade, custo por membro, receita de projeto e *dashboard*, e My Activities entra como item separado, conforme o Código B.64.

```

<ul class="menu-main main-menu">
  <!-- ... -->
  <li
    *ngIf="isFinancesDropdownEnabled"
    class="menu-option finances"
    [class.active]="financesVisible"
    (mouseenter)="popupFinances($event)"
    (mouseleave)="out()">
    <button (click)="toggleFinances()" type="button"
      title="{{ 'PROJECT.SECTION.FINANCES' | translate }}">
      <svg class="icon icon-tabler icons-tabler-outline icon-tabler-coins
    ">...</svg>
    <span class="menu-option-text" data-animation="text">
      {{ 'PROJECT.SECTION.FINANCES' | translate }}
    </span>
    <svg class="agile-chevron" data-animation="text"><use href="#chevron-left
  " /></svg>

```

```

</button>
<ul class="child-menu">
  <li *ngIf="isMenuActivityCostEnabled" class="menu-option">
    <a [routerLink]="[baseHref, 'project', project.slug, 'activity-cost']">
      <span class="menu-option-text" data-animation="text">
        {{'PROJECT.SECTION.ACTIVITY_COST' | translate}}
      </span>
    </a>
  </li>
  <li *ngIf="isMenuMemberCostEnabled" class="menu-option">
    <a [routerLink]="[baseHref, 'project', project.slug, 'member-cost']">
      <span class="menu-option-text" data-animation="text">
        {{'PROJECT.SECTION.MEMBER_COST' | translate}}
      </span>
    </a>
  </li>
  <li *ngIf="isMenuProjectIncomeEnabled" class="menu-option">
    <a [routerLink]="[baseHref, 'project', project.slug, 'project-income
']">
      <span class="menu-option-text" data-animation="text">
        {{'PROJECT.SECTION.PROJECT_INCOME' | translate}}
      </span>
    </a>
  </li>
  <li *ngIf="isMenuDashboardEnabled" class="menu-option">
    <a [routerLink]="[baseHref, 'project', project.slug, 'dashboard']">
      <span class="menu-option-text" data-animation="text">
        {{'PROJECT.SECTION.DASHBOARD' | translate}}
      </span>
    </a>
  </li>
</ul>
</li>
<li *ngIf="isMenuMyActivitiesEnabled" class="menu-option">
  <a [routerLink]="[baseHref, 'project', project.slug, 'my-activities']">
    <svg class="icon icon-tabler icons-tabler-outline icon-tabler-player-play
">...</svg>
    <span class="menu-option-text" data-animation="text">
      {{'PROJECT.SECTION.MY_ACTIVITIES' | translate}}
    </span>
  </a>
</li>

```

```
<!-- ... -->
</ul>
```

Código B.64 – Grupo financeiro e item My Activities no *template* de navegação

src/app/commons/project-navigation/project-navigation.component.spec.ts

A alteração neste arquivo é análoga aos testes de menu de cronograma descritos na Seção B.1.3: foram acrescentados cenários de visibilidade para itens financeiros do menu. O Código B.65 mostra um cenário representativo para o item de custo de atividade.

```
import { Permissions } from '@app/api/roles/roles.model';

describe('ProjectNavigationComponent', () => {
  // ...
  describe('activity cost menu', () => {
    it('enabled when finances are active and user has view finance permission',
      () => {
        const project = {
          ...ProjectMockFactory.build(),
          isFinancesActivated: true,
          myPermissions: [Permissions.viewProject, Permissions.
viewFinancesActivityCost],
        };

        spectator = createHost('<tg-project-navigation [project]="project"></tg-
project-navigation>', {
          hostProps: { project },
        });

        expect(spectator.component.isMenuActivityCostEnabled).toBeTruthy();
      });
    });
  });
});
```

Código B.65 – Teste representativo de visibilidade do menu financeiro

src/app/commons/project-navigation/project-navigation.component.ts

A alteração neste arquivo é análoga à lógica de *dropdown* de cronograma descrita na Seção B.1.3: o componente passa a controlar estado, visibilidade, *popup* e navegação do grupo financeiro. No caso de CUST, cada item financeiro é condicionado à *flag* `isFinancesActivated` e à permissão financeira correspondente, conforme o Código B.66.

```
export class ProjectNavigationComponent implements OnChanges, OnInit {  
  // ...  
  public scrumVisible = false;  
  public financesVisible = false;  
  // ...  
  
  get isMenuActivityCostEnabled() {  
    return this.project.isFinancesActivated &&  
      this.project.myPermissions.includes(Permissions.viewFinancesActivityCost)  
  };  
}  
  get isMenuMemberCostEnabled() {  
    return this.project.isFinancesActivated &&  
      this.project.myPermissions.includes(Permissions.viewFinancesMemberCost);  
  }  
  get isMenuMyActivitiesEnabled() {  
    return this.project.isFinancesActivated &&  
      this.project.myPermissions.includes(Permissions.viewFinancesMyActivities)  
  };  
}  
  get isMenuProjectIncomeEnabled() {  
    return this.project.isFinancesActivated &&  
      this.project.myPermissions.includes(Permissions.viewFinancesProjectIncome  
    );  
  }  
  get isMenuDashboardEnabled() {  
    return this.project.isFinancesActivated &&  
      this.project.myPermissions.includes(Permissions.viewFinancesDashboard);  
  }  
  
  public toggleFinances() {  
    if (this.collapsed) {  
      const firstSlug = this.firstEnabledFinanceSlug();  
      if (firstSlug) {  
        this.router.navigate(['/project', this.project.slug, firstSlug]);  
      }  
    } else {  
      this.financesVisible = !this.financesVisible;  
      localStorage.setItem('projectnav-finances', String(this.financesVisible))  
    }  
  }  
}
```

```
}
}
```

Código B.66 – *Getters* e alternância do *dropdown* financeiro

src/assets/i18n/en.json

A alteração neste arquivo é da mesma natureza da descrita para cronograma na Seção B.1.3: inclusão de rótulos de menu no dicionário inglês do *taiga-front-next*. No módulo financeiro, foram acrescentadas seis chaves de seção, conforme o Código B.67.

```
"SECTION": {
  "BACKLOG": "Backlog",
  "KANBAN": "Kanban",
  "ISSUES": "Issues",
  "FINANCES": "Finances",
  "ACTIVITY_COST": "Activity Cost",
  "MEMBER_COST": "Member Cost",
  "MY_ACTIVITIES": "My Activities",
  "PROJECT_INCOME": "Project Income",
  "DASHBOARD": "Dashboard",
  "WIKI": "Wiki"
}
```

Código B.67 – Rótulos de menu financeiro no *i18n* do *taiga-front-next*

B.3 ÉPICO E3 - GESTÃO DE RISCOS

B.3.1 Back-end

settings/common.py

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.1: a inclusão de um novo aplicativo Django na lista `INSTALLED_APPS`. No caso do módulo de gestão de riscos, o aplicativo registrado é `taiga.projects.risks`, conforme o Código B.68. A justificativa e os pontos de atenção são os mesmos já apresentados naquele bloco.

```
INSTALLED_APPS = [
    # ...
    "taiga.projects.userstories",
    "taiga.projects.tasks",
    "taiga.projects.issues",
    "taiga.projects.risks",
```

```
"taiga.projects.wiki",
# ...
]
```

Código B.68 – Registro do aplicativo de gestão de riscos em `INSTALLED_APPS`

Em atualizações futuras, assim como para os módulos de cronograma e financeiro, deve-se verificar se a linha referente ao aplicativo `taiga.projects.risks` permanece presente na lista; sua ausência desativa por completo o módulo de gestão de riscos.

`taiga/permissions/choices.py`

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.1: a adição de permissões às listas centralizadas `ANON_PERMISSIONS` e `MEMBERS_PERMISSIONS`. No caso do módulo de gestão de riscos, foram acrescentadas a permissão de visualização `view_risks` em `ANON_PERMISSIONS` e um bloco de cinco permissões em `MEMBERS_PERMISSIONS`, conforme o Código B.69.

```
ANON_PERMISSIONS = [
    # ...
    ('view_issues', _('View issues')),
    ('view_risks', _('View risks')),
    ('view_wiki_pages', _('View wiki pages')),
    # ...
]

MEMBERS_PERMISSIONS = [
    # ...
    ('delete_issue', _('Delete issue')),
    # Risk permissions
    ('view_risks', _('View risks')),
    ('add_risk', _('Add risk')),
    ('modify_risk', _('Modify risk')),
    ('comment_risk', _('Comment risk')),
    ('delete_risk', _('Delete risk')),
    # Wiki page permissions
    ('view_wiki_pages', _('View wiki pages')),
    # ...
]
```

Código B.69 – Permissões de riscos acrescentadas em `ANON_PERMISSIONS` e `MEMBERS_PERMISSIONS`

As alterações situam-se no escopo global do arquivo, no interior dessas duas listas, e seguem o mesmo padrão adotado pelo módulo de cronograma. A justificativa completa

encontra-se na Seção B.1.1; os pontos de atenção são os mesmos, com a diferença de que, em atualizações futuras, deve-se verificar a permanência das entradas de risco em ambas as listas.

`taiga/base/filters.py`

O arquivo `taiga/base/filters.py` centraliza as classes de filtro de permissão usadas pelos *endpoints* da API REST do Taiga. Cada classe estende `PermissionBasedFilterBackend` ou `PermissionBasedAttachmentFilterBackend` e declara a permissão que o usuário precisa ter para que os objetos retornados por uma *query* sejam incluídos na resposta. A modificação acrescentou duas novas classes, referentes aos recursos principais e aos anexos do módulo de riscos, conforme o Código B.70.

```
class CanViewWikiPagesFilterBackend(PermissionBasedFilterBackend):
    permission = "view_wiki_pages"

class CanViewRisksFilterBackend(PermissionBasedFilterBackend):
    permission = "view_risks"

class CanViewWikiLinksFilterBackend(PermissionBasedFilterBackend):
    # ...

class CanViewWikiAttachmentFilterBackend(
    PermissionBasedAttachmentFilterBackend):
    permission = "view_wiki_pages"

class CanViewRiskAttachmentFilterBackend(
    PermissionBasedAttachmentFilterBackend):
    permission = "view_risks"
```

Código B.70 – Classes de filtro de permissão para riscos em `taiga/base/filters.py`

As alterações consistem em duas classes acrescentadas, inseridas imediatamente após as classes equivalentes do módulo de *wiki*, seguindo a ordem de declaração já estabelecida no arquivo. Cada classe declara apenas a permissão correspondente; toda a lógica de filtragem é herdada das classes-base.

A modificação é inevitável no código original porque as *ViewSets* do módulo de riscos precisam restringir os resultados retornados com base nas permissões do usuário requisitante. O mecanismo padrão do Taiga para isso é declarar uma classe de filtro neste arquivo e referenciá-la no atributo `filter_backends` de cada *ViewSet*; sem as

classes aqui declaradas, os *endpoints* de riscos e de anexos de riscos retornariam objetos independentemente das permissões do projeto.

Em atualizações futuras, `taiga/base/filters.py` é modificado sempre que um novo módulo introduz permissões de visualização, tornando-o um ponto provável de conflito. Ao integrar novas versões, deve-se verificar se as classes `CanViewRisksFilterBackend` e `CanViewRiskAttachmentFilterBackend` permanecem presentes; sua ausência não interrompe a execução, mas remove o controle de acesso dos *endpoints* de riscos, expondo dados a usuários sem permissão.

`taiga/projects/attachments/api.py`

O arquivo `taiga/projects/attachments/api.py` define as *ViewSet* de anexos para cada tipo de entidade do Taiga. Cada classe estende `BaseAttachmentViewSet` e declara o tipo de conteúdo e as classes de permissão e filtro específicas daquela entidade. A modificação acrescentou a *ViewSet* de anexos do módulo de riscos, conforme o Código B.71.

```
class WikiAttachmentViewSet(BaseAttachmentViewSet):
    permission_classes = (permissions.WikiAttachmentPermission,)
    filter_backends = (filters.CanViewWikiAttachmentFilterBackend,)
    content_type = "wiki.wiki"

class RiskAttachmentViewSet(BaseAttachmentViewSet):
    permission_classes = (permissions.RiskAttachmentPermission,)
    filter_backends = (filters.CanViewRiskAttachmentFilterBackend,)
    content_type = "risks.risk"
```

Código B.71 – *ViewSet* de anexos de riscos em `taiga/projects/attachments/api.py`

A alteração consiste em uma única classe acrescentada ao final do arquivo, seguindo o padrão das demais *ViewSet* de anexo. Os três atributos declarados — `permission_classes`, `filter_backends` e `content_type` — conectam a *ViewSet* às classes de permissão e filtro definidas nos arquivos correspondentes e identificam o tipo de conteúdo Django ao qual os anexos pertencem.

A modificação é necessária porque o suporte a anexos para qualquer entidade exige o registro de uma *ViewSet* neste arquivo; sem ela, o roteador não teria um *handler* para as rotas de anexos de riscos, e as requisições a esses *endpoints* retornariam erro 404. A classe `RiskAttachmentPermission` e o filtro `CanViewRiskAttachmentFilterBackend` referenciados aqui são definidos nos arquivos detalhados nas Seções B.3.1 e B.3.1, respectivamente.

Em atualizações futuras, este arquivo é modificado sempre que um novo módulo introduz suporte a anexos. Ao integrar novas versões, deve-se verificar se `RiskAttachmentViewSet` permanece presente e se os nomes das classes de permissão e filtro continuam correspondendo às definições nos arquivos de origem.

taiga/projects/attachments/permissions.py

O arquivo `taiga/projects/attachments/permissions.py` define as classes de permissão para as operações de cada tipo de anexo e contém `RawAttachmentPerm`, uma classe de despacho que resolve a permissão de leitura de um anexo a partir do tipo de conteúdo do objeto ao qual ele pertence. A modificação adicionou a classe `RiskAttachmentPermission` e um caso de despacho em `RawAttachmentPerm`, conforme o Código B.72.

```
class RiskAttachmentPermission(TaigaResourcePermission):
    retrieve_perms = HasProjectPerm('view_risks') | IsAttachmentOwnerPerm()
    create_perms = HasProjectPerm('modify_risk') | (
        CommentAttachmentPerm() & HasProjectPerm('comment_risk'))
    update_perms = HasProjectPerm('modify_risk') | IsAttachmentOwnerPerm()
    partial_update_perms = HasProjectPerm('modify_risk') |
        IsAttachmentOwnerPerm()
    bulk_update_order_perms = HasProjectPerm('modify_risk') |
        IsAttachmentOwnerPerm()
    destroy_perms = HasProjectPerm('modify_risk') | IsAttachmentOwnerPerm()
    list_perms = AllowAny()

class RawAttachmentPerm(PermissionComponent):
    def check_permissions(self, request, view, obj=None):
        # ...
        elif obj.content_type.app_label == "wiki" and \
            obj.content_type.model == "wiki":
            return WikiAttachmentPermission(...) or is_owner
        elif obj.content_type.app_label == "risks" and \
            obj.content_type.model == "risk":
            return RiskAttachmentPermission(...) or is_owner
        return False
```

Código B.72 – Classe de permissão de anexos de riscos e caso de despacho em `RawAttachmentPerm`

As alterações compreendem dois acréscimos: a classe `RiskAttachmentPermission`, que define as permissões necessárias para cada operação sobre um anexo de risco, e um caso `elif` na lógica de despacho de `RawAttachmentPerm`, que identifica anexos pertencentes a riscos pelo par `app_label/model` do tipo de conteúdo Django.

A modificação é inevitável no código original por duas razões independentes. A classe `RiskAttachmentPermission` segue o padrão centralizado deste arquivo, onde todas as permissões de anexo são declaradas; sem ela, a `ViewSet` de anexos de riscos não teria como verificar permissões. O caso em `RawAttachmentPerm` é necessário porque essa classe

percorre os tipos de conteúdo conhecidos para resolver permissões de acesso a arquivos brutos; sem a entrada para riscos, o acesso a anexos de riscos via URL direta seria sempre negado, independentemente das permissões do usuário.

Em atualizações futuras, a cadeia `elif` de `RawAttachmentPerm` e as classes de permissão neste arquivo são pontos prováveis de conflito com novos módulos. Ao integrar novas versões, deve-se verificar se `RiskAttachmentPermission` permanece definida e se o caso de despacho para `app_label == "risks"` permanece na cadeia; a ausência do caso de despacho nega silenciosamente o acesso a todos os arquivos brutos de anexos de riscos.

`taiga/projects/fixtures/initial_project_templates.json`

O arquivo `taiga/projects/fixtures/initial_project_templates.json` é o *fixture* Django que contém os modelos de projeto pré-definidos do Taiga — Scrum e Kanban — usados como ponto de partida na criação de novos projetos. Cada modelo define, entre outros dados, as permissões associadas a cada papel (*role*) padrão. A modificação acrescentou o conjunto de permissões de risco a todos os papéis de ambos os modelos, conforme o Código B.73, que exhibe um trecho representativo.

```
{
  "name": "Back Developer",
  "permissions": [
    "view_project", "view_milestones", "view_us",
    "view_tasks", "add_task", "modify_task", "delete_task",
    "view_issues", "add_issue", "modify_issue", "delete_issue",
    "view_risks", "add_risk", "modify_risk",
    "delete_risk", "comment_risk",
    "view_wiki_pages", "view_wiki_links"
  ]
}
```

Código B.73 – Permissões de risco acrescentadas aos papéis nos modelos de projeto

As alterações situam-se nos arrays de `permissions` de todos os papéis de ambos os modelos de projeto. Para cada papel que já recebia o conjunto completo de permissões de *issues* (visualizar, adicionar, modificar, excluir, comentar), o mesmo conjunto equivalente de permissões de riscos foi acrescentado, de forma que o nível de acesso ao módulo de riscos acompanha o nível de acesso ao módulo de *issues* em cada papel.

A modificação é necessária porque o *fixture* é a fonte de dados usada pelo Django para popular os modelos de projeto no banco de dados em instalações novas. Sem as permissões de risco nesse arquivo, projetos criados a partir dos modelos Scrum ou Kanban não herdariam permissões de risco nos papéis padrão, obrigando cada administrador a configurá-las manualmente após a criação do projeto.

Em atualizações futuras, este arquivo sofre modificações sempre que o projeto oficial altera os modelos de projeto ou introduz novas permissões. Ao integrar novas versões, deve-se verificar se as permissões de risco permanecem em todos os papéis relevantes. Sua ausência não impede o funcionamento do módulo, mas faz com que projetos recém-criados não disponham de permissões de risco nos papéis padrão, degradando a experiência de uso.

`taiga/projects/history/api.py`

O arquivo `taiga/projects/history/api.py` define as classes de *ViewSet* para os *endpoints* de histórico de cada entidade do Taiga. Cada classe estende `HistoryViewSet` e declara o tipo de conteúdo e a classe de permissão correspondentes. A modificação acrescentou `RiskHistory`, conforme o Código B.74.

```
class IssueHistory(HistoryViewSet):
    content_type = "issues.issue"
    permission_classes = (permissions.IssueHistoryPermission,)

class RiskHistory(HistoryViewSet):
    content_type = "risks.risk"
    permission_classes = (permissions.RiskHistoryPermission,)

class WikiHistory(HistoryViewSet):
    content_type = "wiki.wikipage"
    permission_classes = (permissions.WikiHistoryPermission,)
```

Código B.74 – ViewSet de histórico de riscos em `taiga/projects/history/api.py`

A alteração consiste em uma única classe acrescentada entre `IssueHistory` e `WikiHistory`, seguindo a ordem já estabelecida no arquivo. Os dois atributos declarados identificam o tipo de conteúdo Django ao qual o histórico pertence e a classe de permissão que controla o acesso.

A modificação é inevitável no código original porque o sistema de histórico do Taiga associa *endpoints* REST a entidades por meio dessas classes; sem `RiskHistory`, não existiria rota para consultar o histórico de um risco, e a classe de permissão referenciada na Seção B.3.1 nunca seria invocada.

Em atualizações futuras, `taiga/projects/history/api.py` é modificado sempre que um novo módulo introduz suporte a histórico. Ao integrar novas versões, deve-se verificar se `RiskHistory` permanece presente e se o `content_type` continua correspondendo ao definido no modelo Django de riscos.

taiga/projects/history/freeze_impl.py

O arquivo `taiga/projects/history/freeze_impl.py` também foi modificado nos Épicos E1 (Seção B.1.1) e E2 (Seção B.2.1), com os quais compartilha o padrão de adição de funções de congelamento para novas entidades. No Épico E3, as modificações restringiram-se às funções da entidade risco, sem alteração à função `project_freezer`: foi acrescentada a função `risk_values()`, que extrai os usuários referenciados no *diff* de um risco, e a função `risk_freezer()`, que produz o retrato completo (*snapshot*) de um risco para armazenamento no histórico, conforme o Código B.75.

```
def wikipage_values(diff):
    values = _common_users_values(diff)
    return values

def risk_values(diff):
    values = _common_users_values(diff)
    return values

# ...

def wikipage_freezer(wiki) -> dict:
    # ...
    return snapshot

def risk_freezer(risk) -> dict:
    snapshot = {
        "ref": risk.ref,
        "owner": risk.owner_id,
        "subject": risk.subject,
        "description": risk.description,
        "cause": risk.cause,
        "consequence": risk.consequence,
        "assigned_to": risk.assigned_to_id,
        "status": risk.status,
        "probability": risk.probability,
        "impact": risk.impact,
        "score": risk.score,
        "level": risk.level,
        "category": risk.category,
        "risk_type": risk.risk_type,
        "strategy": risk.strategy,
        "response_plan": risk.response_plan,
        "tags": risk.tags,
```

```

    "attachments": extract_attachments(risk),
    "due_date": str(risk.due_date) if risk.due_date else None,
}
return snapshot

```

Código B.75 – Funções `risk_values` e `risk_freezer` em `freeze_impl.py`

As alterações compreendem dois acréscimos independentes: `risk_values()`, análoga às funções `*_values()` das demais entidades, delega a extração de usuários à função compartilhada `_common_users_values`; `risk_freezer()` captura todos os campos relevantes do modelo `Risk` — incluindo campos específicos do domínio de gestão de riscos como `probability`, `impact`, `score`, `level`, `category`, `risk_type`, `strategy`, `response_plan` e `cause` — para que o sistema de histórico possa calcular diferenças entre versões e exibir o que mudou em cada revisão. Diferentemente dos Épicos E1 e E2, não foi necessário modificar `project_freezer`, pois a flag `is_risks_activated` não precisa ser rastreada no histórico do projeto nesta implementação.

Em atualizações futuras, os campos capturados por `risk_freezer()` precisam acompanhar a evolução do modelo `Risk`: a adição de novos campos ao modelo sem a inclusão correspondente no freezer faz com que esses campos sejam omitidos dos *snapshots* e, portanto, invisíveis no histórico. O registro dessas funções no sistema de histórico é feito em `history/services.py` (Seção B.3.1).

`taiga/projects/history/permissions.py`

O arquivo `taiga/projects/history/permissions.py` define as classes de permissão para os *ViewSets* de histórico. Cada classe estende `TaigaResourcePermission` e declara as permissões necessárias para recuperar, editar, excluir, restaurar e visualizar versões de comentários no histórico de uma entidade. A modificação acrescentou `RiskHistoryPermission`, conforme o Código B.76.

```

class RiskHistoryPermission(TaigaResourcePermission):
    retrieve_perms = HasProjectPerm('view_project')
    edit_comment_perms = IsCommentProjectAdmin() | IsCommentOwner()
    delete_comment_perms = IsCommentProjectAdmin() | IsCommentOwner()
    undelete_comment_perms = IsCommentProjectAdmin() | IsCommentDeleter()
    comment_versions_perms = IsCommentProjectAdmin() | IsCommentOwner()

```

Código B.76 – Classe de permissão do histórico de riscos em `taiga/projects/history/permissions.py`

A alteração situa-se no escopo global do arquivo, entre as classes equivalentes de *issues* e *wiki*, e consiste em uma única classe acrescentada. A permissão de recuperação é `view_project` — e não `view_risks` — de forma consistente com o padrão das demais entidades, uma vez que o acesso ao histórico requer apenas participação no projeto.

A modificação é necessária porque a classe `RiskHistory` definida na Seção B.3.1 referencia `RiskHistoryPermission`; sem ela, o Django levantaria um erro na inicialização do servidor ao tentar resolver a referência.

Em atualizações futuras, o padrão de permissões de histórico é mantido pelo repositório original e pode ser alterado para todas as entidades simultaneamente. Ao integrar novas versões, deve-se verificar se `RiskHistoryPermission` permanece coerente com as demais classes de permissão de histórico; em particular, o nome da permissão em `retrieve_perms` precisa corresponder a uma permissão reconhecida pelo sistema.

`taiga/projects/history/services.py`

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.1: a importação das funções de congelamento e valores de uma nova entidade e o seu registro no sistema de histórico. No módulo de gestão de riscos, as funções importadas são `risk_freezer` e `risk_values`, definidas na Seção B.3.1, conforme o Código B.77.

```
from .freeze_impl import wikipage_freezer
from .freeze_impl import risk_freezer

from .freeze_impl import wikipage_values
from .freeze_impl import risk_values

# ...

register_freeze_implementation("wiki.wikipage", wikipage_freezer)
register_freeze_implementation("risks.risk", risk_freezer)

# ...

register_values_implementation("wiki.wikipage", wikipage_values)
register_values_implementation("risks.risk", risk_values)
```

Código B.77 – Importação e registro das implementações de histórico de riscos em `services.py`

A justificativa e os pontos de atenção são os mesmos já apresentados na Seção B.1.1: sem esse registro, o sistema de histórico não saberia qual função invocar ao congelar ou calcular diferenças de um objeto do tipo `risks.risk`, e o salvamento de histórico de riscos falharia em tempo de execução.

taiga/projects/models.py

O arquivo `taiga/projects/models.py` define os modelos de dados centrais do aplicativo de projetos, entre eles `Project` e `ProjectTemplate`. A modificação acrescentou o campo `is_risks_activated` ao modelo `Project` e propagou-o no método `load_data_from_project` de `ProjectTemplate`, conforme o Código B.78.

```
class Project(ProjectDefaults, TaggedMixin, ...):
    # ...
    is_issues_activated = models.BooleanField(
        default=True, null=False, blank=True,
        verbose_name=_("active issues panel"))
    is_risks_activated = models.BooleanField(
        default=False, null=False, blank=True,
        verbose_name=_("active risks panel"))
    videoconferences = models.CharField(...)
    # ...

class ProjectTemplate(TaggedMixin, ...):
    # ...
    def load_data_from_project(self, project):
        # ...
        self.is_issues_activated = project.is_issues_activated
        self.is_risks_activated = project.is_risks_activated
        self.videoconferences = project.videoconferences
```

Código B.78 – Campo `is_risks_activated` em `Project` e propagação em `ProjectTemplate`

As alterações compreendem dois acréscimos: a declaração do campo `is_risks_activated` no corpo da classe `Project` e a propagação desse campo no método `load_data_from_project` de `ProjectTemplate`, responsável por copiar as flags de ativação de um projeto para um template. O campo é um `BooleanField` com `default=False`, valor que preserva os projetos já existentes com o módulo desativado por padrão — diferentemente de outros módulos nativos do Taiga, cujas flags equivalentes têm `default=True`, o que reflete o caráter opt-in da funcionalidade de gestão de riscos. Cabe observar que o método `apply_to_project` — que propaga flags de um template para um projeto — não foi modificado, de forma que a aplicação de um template a um projeto não força a ativação do módulo de riscos, preservando a decisão de ativação para o administrador do projeto.

A modificação é a origem da flag de ativação do módulo de riscos por projeto. Sem o campo em `Project`, não há como armazenar nem consultar o estado de ativação no banco de dados. A inclusão em `load_data_from_project` garante que, ao gerar um template a partir de um projeto, o estado do módulo de riscos seja preservado de forma coerente com as demais flags de módulo.

Em atualizações futuras, o modelo `Project` é um arquivo de alto conflito: toda nova funcionalidade que introduz configuração por projeto acrescenta campos aqui. Ao integrar novas versões, deve-se verificar se o campo `is_risks_activated` permanece em `Project` e propagado em `ProjectTemplate`; sua ausência em `Project` impede o salvamento da flag no banco de dados, enquanto sua ausência em `load_data_from_project` faz com que templates gerados a partir de projetos percam o estado de ativação do módulo.

`taiga/projects/references/models.py`

O arquivo `taiga/projects/references/models.py` gerencia o contador de referências sequenciais que o Taiga atribui a objetos de projeto (histórias de usuário, tarefas, *issues* etc.), garantindo que cada objeto receba um identificador único no formato `#N`. A modificação conectou o modelo `Risk` ao sistema de referências em três pontos, conforme o Código B.79.

```
from taiga.projects.issues.models import Issue
from taiga.projects.risks.models import Risk

# ...

def recalc_reference_counter(project):
    max_ref_us = project.user_stories.all().aggregate(...)
    max_ref_task = project.tasks.all().aggregate(...)
    max_ref_issue = project.issues.all().aggregate(...)
    max_ref_risk = project.risks.all().aggregate(max=models.Max('ref'))
    max_references = list(filter(lambda x: x is not None,
        [max_ref_us['max'], max_ref_task['max'],
        max_ref_issue['max'], max_ref_risk['max']])))
    # ...

# Issue
models.signals.pre_save.connect(
    store_previous_project, sender=Issue, dispatch_uid="refissue")
models.signals.post_save.connect(
    attach_sequence, sender=Issue, dispatch_uid="refissue")

# Risk
models.signals.pre_save.connect(
    store_previous_project, sender=Risk, dispatch_uid="refrisk")
models.signals.post_save.connect(
    attach_sequence, sender=Risk, dispatch_uid="refrisk")
```

Código B.79 – Integração do modelo `Risk` ao sistema de referências

As alterações compreendem três acréscimos: a importação do modelo `Risk`; a inclusão de `max_ref_risk` no cálculo do maior número de referência do projeto, realizado pela função `recalc_reference_counter`; e a conexão dos sinais `pre_save` e `post_save` do modelo `Risk` às funções `store_previous_project` e `attach_sequence`, que atribuem e preservam o número de referência de cada objeto.

A modificação é inevitável no código original por duas razões. Os sinais são necessários para que cada risco receba automaticamente um identificador sequencial (`#N`) no momento de sua criação — sem eles, o campo `ref` permaneceria nulo. A inclusão em `recalc_reference_counter` é necessária para que o recálculo do contador do projeto — invocado em importações e migrações de dados — considere os riscos já existentes, evitando colisões de referência com outros tipos de objeto.

Em atualizações futuras, a lista de objetos em `recalc_reference_counter` e o bloco de conexão de sinais ao final do arquivo são modificados sempre que um novo tipo de objeto é introduzido. Ao integrar novas versões, deve-se verificar se a entrada de `max_ref_risk` permanece na função e se os sinais do modelo `Risk` permanecem conectados; a ausência dos sinais impede a atribuição de referências a novos riscos, enquanto a ausência no recálculo pode gerar colisões de referência após operações de importação de dados.

`taiga/projects/serializers.py`

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.1: a declaração explícita do campo `is_risks_activated` nos três serializadores de projeto. No módulo de gestão de riscos, o campo adicionado é `is_risks_activated`, conforme o Código B.80.

```
class ProjectSerializer(serializers.LightSerializer):
    # ...
    is_issues_activated = Field()
    is_risks_activated = Field()
    videoconferences = Field()
    # ...

class ProjectLightSerializer(serializers.LightSerializer):
    # ...
    is_issues_activated = Field()
    is_risks_activated = Field()
    videoconferences = Field()
    # ...

class ProjectTemplateSerializer(serializers.LightSerializer):
    # ...
    is_issues_activated = Field()
```

```
is_risks_activated = Field()
videoconferences = Field()
# ...
```

Código B.80 – Campo `is_risks_activated` nos serializadores de projeto

A justificativa e os pontos de atenção são os mesmos apresentados na Seção B.1.1: o serializador-base `LightSerializer` expõe apenas os campos explicitamente listados; sem as três entradas, o campo `is_risks_activated` não chegaria ao cliente pela API, impedindo que o front-end determinasse se o módulo de riscos está ativo em um dado projeto.

`taiga/routers.py`

O arquivo `taiga/routers.py` é o registro central das rotas da API REST do Taiga: ele importa as *ViewSets* de cada módulo e as registra no roteador global DRF (*Django REST Framework*), definindo as URLs expostas sob `/api/v1/`. A modificação distribuiu-se em quatro blocos de acréscimos, referentes aos anexos, recursos principais, avaliações e histórico do módulo de riscos, conforme o Código B.81.

```
# Bloco 1 - Anexos
from taiga.projects.attachments.api import WikiAttachmentViewSet
from taiga.projects.attachments.api import RiskAttachmentViewSet
# ...
router.register(r"wiki/attachments", WikiAttachmentViewSet, ...)
router.register(r"risks/attachments", RiskAttachmentViewSet,
    base_name="risk-attachments")

# Bloco 2 - Recursos principais
from taiga.projects.risks.api import RiskViewSet
from taiga.projects.risks.api import RiskWatchersViewSet
from taiga.projects.risks.api import RiskEvaluationViewSet
# ...
router.register(r"risks", RiskViewSet, base_name="risks")
router.register(r"risks/(?P<resource_id>\d+)/watchers",
    RiskWatchersViewSet, base_name="risk-watchers")
router.register(r"risk-evaluations", RiskEvaluationViewSet,
    base_name="risk-evaluations")

# Bloco 3 - Histórico
from taiga.projects.history.api import IssueHistory
from taiga.projects.history.api import RiskHistory
from taiga.projects.history.api import WikiHistory
# ...
```

```
router.register(r"history/issue", IssueHistory, ...)
router.register(r"history/risk", RiskHistory, base_name="risk-history")
router.register(r"history/wiki", WikiHistory, ...)
```

Código B.81 – Importações e registros de rotas do módulo de riscos em `taiga/routers.py`

As alterações distribuem-se em três blocos independentes do arquivo, cada um correspondendo a uma dimensão da API de riscos: anexos, recursos principais (riscos, observadores e avaliações de risco) e histórico. O bloco de avaliações introduz um tipo de recurso sem equivalente nos módulos anteriores: `risk-evaluations` expõe um *endpoint* próprio para as avaliações periódicas de probabilidade e impacto associadas a cada risco. Todas as adições são de linhas novas; nenhuma linha existente foi modificada.

A modificação é inevitável no código original porque `taiga/routers.py` é o único ponto onde os *ViewSets* são associados a URLs na API; sem o registro aqui, as *views* existem mas jamais recebem requisições HTTP.

Em atualizações futuras, `taiga/routers.py` é um arquivo de alto conflito: toda nova funcionalidade do Taiga tende a registrar rotas aqui. Ao integrar novas versões, deve-se verificar se os cinco registros de riscos (`risks`, `risk-attachments`, `risk-watchers`, `risk-evaluations` e `history/risk`) permanecem presentes e se os nomes base não colidem com rotas novas introduzidas pelo repositório oficial.

B.3.2 Front-end

`app/coffee/app.coffee`

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.2: o registro de rotas e do módulo AngularJS no *bootstrap* do front-end. No módulo de gestão de riscos, foram acrescentadas quatro rotas e o módulo `taigaRisks`, conforme o Código B.82. A justificativa geral é a mesma daquele bloco; a diferença em relação ao Épico E1 está na quantidade de rotas: o módulo de riscos expõe quatro telas independentes (lista, detalhe, matriz e evolução), enquanto o de cronograma expõe duas.

```
$routeProvider.when("/project/:pslug/risks",
  { templateUrl: "risk/risks.html", section: "risks" })
$routeProvider.when("/project/:pslug/risk/:riskref",
  { templateUrl: "risk/risks-detail.html", section: "risks" })
$routeProvider.when("/project/:pslug/risks/matrix",
  { templateUrl: "risk/risk-matrix.html", section: "risks" })
$routeProvider.when("/project/:pslug/risks/evolution",
  { templateUrl: "risk/risks-evolution.html", section: "risks" })
```

```
modules = [
```

```
# ...
"taigaIssues",
"taigaRisks",
"taigaUserStories",
# ...
]
```

Código B.82 – Rotas e módulo de riscos no bootstrap da aplicação

Em atualizações futuras, assim como descrito na Seção B.1.2, deve-se verificar se as quatro rotas e o módulo `taigaRisks` permanecem presentes; a ausência das rotas impede o carregamento das telas do módulo, e a ausência do módulo remove todos os controladores e diretivas de riscos do *bundle*.

`app/coffee/modules/admin/roles.coffee`

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.2: a inclusão das permissões do novo módulo nas listas de permissões padrão e a adição de uma categoria na tela de administração de papéis. No módulo de gestão de riscos, as modificações abrangem três frentes: inclusão de `view_risks` nas listas `publicPermissions` e `DEFAULT_PERMISSIONS`, e acréscimo da categoria de permissões de riscos exibida na tela, conforme o Código B.83.

```
publicPermissions = [..., 'view_issues', 'view_wiki_pages',
  'view_risks']

DEFAULT_PERMISSIONS = ["view_project", ..., "view_issues", "view_risks"]

riskPermissions = [
  { key: "view_risks",    name: "COMMON.PERMISSIONS_CATEGORIES.RISKS.VIEW_RISKS"
  },
  { key: "add_risk",      name: "COMMON.PERMISSIONS_CATEGORIES.RISKS.ADD_RISKS"
  },
  { key: "modify_risk",   name: "COMMON.PERMISSIONS_CATEGORIES.RISKS.
MODIFY_RISKS"
  },
  { key: "comment_risk", name: "COMMON.PERMISSIONS_CATEGORIES.RISKS.
COMMENT_RISKS"
  },
  { key: "delete_risk",  name: "COMMON.PERMISSIONS_CATEGORIES.RISKS.
DELETE_RISKS"
  },
]
categories.push({ name: "COMMON.PERMISSIONS_CATEGORIES.RISKS.NAME",
  permissions: setActivePermissions(riskPermissions) })
```

Código B.83 – Permissões de riscos em roles.coffee

A justificativa e os pontos de atenção são análogos aos descritos na Seção B.1.2. Em atualizações futuras, deve-se verificar se `view_risks` permanece em ambas as listas e se a categoria de riscos permanece no array `categories`.

`app/coffee/modules/base.coffee`

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.2: o acréscimo de entradas no objeto `urls`, que mapeia chaves de navegação para padrões de URL. No módulo de gestão de riscos, foram acrescentadas três entradas, conforme o Código B.84.

```
urls = {  
  # ...  
  "project-issues": "/project/:project/issues"  
  "project-risks":   "/project/:project/risks"  
  "project-risks-detail": "/project/:project/risk/:ref"  
  "project-risks-matrix": "/project/:project/risks/matrix"  
  "project-epics": "/project/:project/epics"  
  # ...  
}
```

Código B.84 – Entradas de navegação de riscos no mapa `urls` de `base.coffee`

A justificativa e os pontos de atenção são os mesmos da Seção B.1.2. Em atualizações futuras, deve-se verificar se as três entradas permanecem presentes; sua ausência impede que a diretiva `tg-nav` resolva os links para as telas de riscos, quebrando menus e redirecionamentos.

`app/coffee/modules/resources.coffee`

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.2: o acréscimo de entradas no objeto `urls`, que mapeia chaves de recurso para os caminhos dos *endpoints* da API, e o registro do provedor de recursos do módulo. No módulo de gestão de riscos, foram acrescentadas seis entradas de *endpoint* e o provedor `$tgRisksResourcesProvider`, conforme o Código B.85.

```
urls = {  
  # ...  
  "risks":           "/risks"  
  "risks-matrix":    "/risks/matrix"  
  "risks-filters-data": "/risks/filters_data"  
  "risk-evaluations": "/risk-evaluations"  
  # ...  
  "history/issue":   "/history/issue"
```

```

    "history/risk": "/history/risk"
    "history/wiki": "/history/wiki"
    # ...
    "attachments/wiki": "/wiki/attachments"
    "attachments/risk": "/risks/attachments"
  }

  module.run([..., "$tgIssuesResourcesProvider",
    "$tgRisksResourcesProvider",
    "$tgWikiResourcesProvider", ...])

```

Código B.85 – Endpoints da API e provedor de riscos em resources.coffee

A justificativa e os pontos de atenção são análogos aos da Seção B.1.2. Em atualizações futuras, deve-se verificar se as seis entradas de *endpoint* e o provedor `$tgRisksResourcesProvider` permanecem presentes, garantindo que as chamadas à API de riscos sejam resolvidas corretamente.

`app/coffee/modules/common/components.coffee`

O arquivo `app/coffee/modules/common/components.coffee` define componentes de interface reutilizáveis da aplicação, entre eles a diretiva `DateSelectorDirective`, que encapsula o seletor de datas Pikaday. A modificação estendeu essa diretiva para suportar o vínculo bidirecional via `ngModel` e um *callback* de alteração de data por atributo, além de corrigir o comportamento de inicialização e atualização programática do campo, conforme o Código B.86.

```

DateSelectorDirective = ($rootScope, datePickerConfigService) ->
  link = ($scope, $el, $attrs, $model) ->
    userPickedDate = false

    initialize = () ->
      _.merge(datePickerConfig, {
        field: $el[0]
        onSelect: (date) ->
          return unless userPickedDate
          userPickedDate = false
          isoDate = moment(date).toISOString()
          if $model
            $scope.$apply -> $model.$setViewValue(isoDate)
          if $attrs.tgDateChange
            $scope.$apply ->
              $scope.$eval($attrs.tgDateChange, {$isoDate:
isoDate})

```

```

        onOpen: -> userPickedDate = true
    })
    $el.picker = new Pikaday(datePickerConfig)

    initialize()

    $attrs.$observe "pickerValue", (val) ->
        if val and not _.isDate(val)
            val = moment(val).format(datePickerConfigService.get().format)
        $el.val(val)
        $el.picker?.setDate(val, true)

    return {
        require: "?ngModel"
        link: link
    }

```

Código B.86 – Extensão da `DateSelectorDirective` para suporte a `ngModel` e `tgDateChange`

As alterações compreendem: a flag `userPickedDate` para distinguir seleções do usuário de atribuições programáticas; os *callbacks* `onSelect` (que propaga a data via `ngModel` e via `tgDateChange`) e `onOpen` (que ativa a flag); a chamada antecipada a `initialize()` para criar o *picker* antes do *observer*; a normalização do valor em `$attrs.$observe` para o formato do `Pikaday`; a substituição de `picker.setDate(val)` por `picker.setDate(val, true)` para suprimir o *callback* `onSelect` em atribuições programáticas; e a declaração de `require: "?ngModel"` para injetar o controlador de modelo.

A modificação é necessária porque o campo de data da avaliação de risco usa ao mesmo tempo `ngModel` (para vincular o valor ao escopo) e `tgDateChange` (para acionar ações ao selecionar uma data sem contaminar o objeto com `_modifiedAttrs`). O comportamento original da diretiva não suportava esse padrão: alterações programáticas disparavam `onSelect` e a distinção entre seleção do usuário e atribuição por código era impossível.

Em atualizações futuras, este arquivo é mantido pelo repositório original e pode ser refatorado para versões mais recentes do `Pikaday`. Ao integrar novas versões, deve-se verificar se o comportamento de `setDate(val, true)` para suprimir o *callback* e o suporte ao atributo `tgDateChange` permanecem funcionais com a nova versão da biblioteca.

`app/coffee/modules/common/sections.coffee`

O arquivo `app/coffee/modules/common/sections.coffee` define o serviço `SectionsService`, que mantém um mapa de identificadores de seção para metadados

de navegação — título, caminho de URL e nome da flag de ativação. Esse serviço é consumido pelo mecanismo de seção ativa da aplicação para determinar qual módulo está sendo exibido. A modificação acrescentou a entrada do módulo de riscos, conforme o Código B.87.

```
class SectionsService extends taiga.Service
  sections:
    4: {id: 4, title: 'KANBAN', path: 'kanban', enabled: '
is_kanban_activated'}
    5: {id: 5, title: 'ISSUES', path: 'issues', enabled: '
is_issues_activated'}
    6: {id: 6, title: 'WIKI', path: 'wiki', enabled: '
is_wiki_activated'}
    7: {id: 7, title: 'RISKS', path: 'risks', enabled: '
is_risks_activated'}
```

Código B.87 – Entrada do módulo de riscos em SectionsService

A alteração consiste em uma única linha acrescentada ao mapa `sections`, declarando o identificador numérico, o título, o caminho e a flag de ativação do módulo de riscos.

A modificação é necessária porque `SectionsService` é o único ponto onde a aplicação associa o caminho de URL `risks` à flag `is_risks_activated`; sem essa entrada, componentes que consultam o serviço para determinar se a seção atual está habilitada (como menus condicionais e *guards* de rota) não reconheceriam a seção de riscos como uma seção válida.

Em atualizações futuras, o mapa `sections` cresce sempre que um novo módulo navegável é introduzido. Ao integrar novas versões, deve-se verificar se a entrada de riscos permanece e se o identificador numérico 7 não colidiu com uma entrada nova do repositório oficial.

`app/coffee/modules/taskboard/main.coffee`

O arquivo `app/coffee/modules/taskboard/main.coffee` implementa o `TaskboardController`, responsável por gerenciar a página do quadro de tarefas da *sprint*. A modificação integrou a exibição e o gerenciamento de riscos associados à *sprint* ao controlador existente, distribuída em múltiplos pontos do arquivo, conforme o Código B.88.

```
# Dependência injetada
constructor: (@scope, ..., @taskboardIssuesService,
  @taskboardRisksService, @storage, ...) =>
  # ...
  taiga.defineImmutableProperty @.scope, "milestoneRisks", () =>
```

```

    return @taskboardRisksService.milestoneRisks

    @scope.issues = []
    @scope.risks = []
    @scope.$watch 'milestoneRisks', () =>
      if @scope.milestoneRisks
        levelOrder = {CRITICAL: 4, HIGH: 3, MEDIUM: 2, LOW: 1}
        @scope.risks = @scope.milestoneRisks.toJS()
          .sort (a, b) ->
            (levelOrder[b?.level] or 0) - (levelOrder[a?.level] or 0)
      else @scope.risks = []

# Carregamento
loadRisks: ->
  return @rs.risks.listInProject(
    @scope.projectId, @scope.sprintId, params).then (risks) =>
    @taskboardRisksService.init(@scope.project, @scope.usersById)
    @taskboardRisksService.set(risks)

# Assinatura WebSocket
routingKey2 = "changes.project.#{@scope.projectId}.risks"
@events.subscribe @scope, routingKey2, debounceLeading(500,
  (message) => @.loadRisks())

# Adição e avaliação de risco
addNewRisk: (type, us) ->
  @rootscope.$broadcast("genericform:new-or-existing", {
    objType: 'risk', project: @scope.project,
    sprintId: @scope.sprintId, ... })

saveRiskEvaluation: (risk) ->
  @repo.save(risk).then => @taskboardRisksService.refresh()

```

Código B.88 – Integração de riscos ao TaskboardController

As alterações compreendem: a injeção de `tgTaskboardRisks` como dependência do controlador; a definição de `milestoneRisks` como propriedade imutável do escopo; o *watcher* que mantém `$scope.risks` atualizado e ordenado por nível de criticidade; o método `loadRisks()` que busca os riscos da *sprint* via API; a assinatura WebSocket que recarrega riscos em tempo real quando o canal de mudanças é acionado; e os métodos `addNewRisk()` e `saveRiskEvaluation()` para criação e avaliação de riscos pelo quadro.

A modificação é necessária porque o quadro de *sprint* passou a exibir uma seção de riscos associados à *sprint* ativa. Assim como *issues*, os riscos precisam ser carregados

em paralelo com tarefas, reagir a eventos WebSocket em tempo real e oferecer ações de criação e atualização diretamente no quadro.

Em atualizações futuras, `main.coffee` é o arquivo de maior complexidade e maior superfície de conflito do front-end: qualquer alteração na assinatura do construtor do controlador, na lógica de carregamento ou na estrutura de eventos WebSocket pelo repositório oficial tende a conflitar com os blocos aqui inseridos. Ao integrar novas versões, deve-se revisar especialmente a lista de dependências injetadas, os métodos `loadRisks` e `loadIssues` (para manter paralelismo consistente) e a chave de roteamento WebSocket.

`app/locales/taiga/locale-en.json`

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.2: a inclusão de chaves de tradução em inglês para a interface do novo módulo. No módulo de gestão de riscos, foram acrescentadas chaves para permissões (`COMMON.PERMISSIONS_CATEGORIES.RISKS`), administração de módulos (`ADMIN.MODULES.RISKS`), seções do menu (`PROJECT.SECTION.RISKS*`), rótulos da tela de riscos (`RISKS.*`) e campos de histórico (`DIFF.*`), conforme o Código B.89.

```
"PERMISSIONS_CATEGORIES": {
  "RISKS": {
    "NAME": "Risks",
    "VIEW_RISKS": "View risks", "ADD_RISKS": "Add risks",
    "MODIFY_RISKS": "Modify risks", "DELETE_RISKS": "Delete risks"
  }
},
"MODULES": {
  "RISKS": "Risks",
  "RISKS_DESCRIPTION": "Track and mitigate the risks associated with your
project."
},
"SECTION": {
  "RISKS": "Risks",
  "RISKS_LIST": "List Risks",
  "RISKS_MATRIX": "Risk Matrix",
  "RISKS_DASHBOARD": "Risk Evolution"
},
"RISKS": {
  "ACTION_NEW_RISK": "NEW RISK",
  "LIGHTBOX": { "TITLE_NEW": "New Risk" },
  "TABLE": { "EMPTY": { "TITLE": "No risks registered." } }
  // ...
}
```

Código B.89 – Trecho das chaves de tradução do módulo de riscos (arquivo em inglês)

A justificativa e os pontos de atenção são os mesmos da Seção B.1.2. Em atualizações futuras, deve-se verificar se as chaves de riscos permanecem presentes e se os identificadores de chave não colidem com entradas novas introduzidas pelo repositório oficial.

app/locales/taiga/locale-pt-br.json

A alteração neste arquivo é da mesma natureza da descrita para `locale-en.json` na Seção B.3.2: a inclusão das mesmas chaves de tradução, desta vez com os rótulos em português do Brasil, conforme o Código B.90.

```
"PERMISSIONS_CATEGORIES": {
  "RISKS": {
    "NAME": "Riscos",
    "VIEW_RISKS": "Ver riscos", "ADD_RISKS": "Adicionar riscos",
    "MODIFY_RISKS": "Modificar riscos", "DELETE_RISKS": "Excluir riscos"
  }
},
"MODULES": {
  "RISKS": "Riscos",
  "RISKS_DESCRIPTION": "Acompanhe e mitigue os riscos associados com este projeto."
},
"SECTION": {
  "RISKS": "Riscos",
  "RISKS_LIST": "Listar Riscos",
  "RISKS_MATRIX": "Matriz de Riscos",
  "RISKS_DASHBOARD": "Evolução de Riscos"
},
"RISKS": {
  "ACTION_NEW_RISK": "NOVO RISCO",
  "LIGHTBOX": { "TITLE_NEW": "Novo Risco" },
  "TABLE": { "EMPTY": { "TITLE": "Não há riscos registrados." } }
  // ...
}
```

Código B.90 – Trecho das chaves de tradução do módulo de riscos (arquivo em português do Brasil)

A justificativa e os pontos de atenção são os mesmos da Seção B.3.2.

app/modules/components/project-menu/project-menu.controller.coffee

O arquivo `project-menu.controller.coffee` implementa `ProjectMenuController`, responsável por calcular quais itens do menu lateral devem ser exibidos para o projeto atual com base nos módulos ativos e nas permissões do usuário. A modificação acrescentou o controle de visibilidade para o item de riscos, conforme o Código B.91.

```
@.menu = Immutable.Map({
  backlog: false, kanban: false, issues: false,
  wiki: false, risks: false
})

if @.project.get("is_wiki_activated") && \
  @.project.get("my_permissions").indexOf("view_wiki_pages") != -1
  @.menu = @.menu.set("wiki", true)

if @.project.get("is_risks_activated") &&
  @.project.get("my_permissions").indexOf("view_risks") != -1
  @.menu = @.menu.set("risks", true)
```

Código B.91 – Controle de visibilidade do item de riscos no menu lateral

As alterações compreendem dois acréscimos: a declaração da chave `risks: false` no mapa inicial do menu e o bloco condicional que a define como `true` quando o projeto tem `is_risks_activated` ativo e o usuário possui a permissão `view_risks`.

A modificação é necessária porque o controlador do menu constrói dinamicamente a visibilidade de cada item; sem a chave `risks` no mapa e o bloco condicional correspondente, o item de riscos nunca apareceria no menu lateral, independentemente da configuração do projeto ou das permissões do usuário.

Em atualizações futuras, o mapa de `menu` e os blocos condicionais são modificados sempre que um novo módulo navegável é introduzido. Ao integrar novas versões, deve-se verificar se a chave `risks` permanece no mapa inicial e se o condicional permanece presente; sua ausência oculta o módulo de riscos do menu mesmo quando ativo e com permissão concedida.

app/modules/components/wysiwyg/item-wysiwyg.directive.coffee

O arquivo `item-wysiwyg.directive.coffee` define a diretiva `ItemWysiwyg`, que encapsula o editor de texto rico usado nas telas de detalhe das entidades do Taiga para edição de campos de texto longo. A modificação estendeu a diretiva de duas formas: tornou o campo-alvo configurável por atributo e adicionou o tipo `risk` ao mapa de tipos de conteúdo, conforme o Código B.92.

```
link = ($scope, $el, $attrs) ->
```

```

$scope.field = $attrs.field or "description"
# ...
$scope.saveDescription = (description, cb) ->
  transform = $modelTransform.save (item) ->
    item[$scope.field] = description
  return item

contentType =
  us: "us", issue: "issue", task: "task",
  risks: "risk", risk: "risk"

```

Código B.92 – Campo configurável e tipo risk na diretiva ItemWysiwyg

As alterações compreendem dois acréscimos independentes. A configurabilidade do campo é necessária porque a tela de detalhe de risco usa o editor para múltiplos campos textuais além de `description` — especificamente `cause`, `consequence` e `response_plan`; sem o atributo `field`, todos os campos seriam salvos sempre em `description`, sobrecrevendo os dados uns dos outros. A entrada no mapa de tipos de conteúdo é necessária porque o editor usa o tipo para resolver o caminho de *upload* de arquivos incorporados; sem ela, o *upload* de imagens em campos de texto de riscos falharia com tipo desconhecido.

Em atualizações futuras, este arquivo é mantido pelo repositório original; alterações na lógica de salvamento do `modelTransform` ou na estrutura do mapa de tipos de conteúdo podem conflitar com as modificações aqui realizadas. Ao integrar novas versões, deve-se verificar se o atributo `field` é lido antes do salvamento e se as entradas de risco permanecem no mapa de tipos.

`app/modules/history/history/history-diff.jade`

O arquivo `history-diff.jade` define o template que renderiza a visualização de diferenças entre versões de um objeto no painel de histórico. Cada tipo de campo é exibido por um bloco `.diff-wrapper` condicional que inclui um sub-template específico. A modificação acrescentou oito blocos para os campos específicos do modelo de risco, conforme o Código B.93.

```

// Blocos existentes (trecho)
.diff-wrapper(ng-if="vm.type == 'promoted_to'")
  include history-templates/history-promoted

.diff-wrapper(ng-if="vm.type == 'cause'")
  include history-templates/history-cause
.diff-wrapper(ng-if="vm.type == 'consequence'")
  include history-templates/history-consequence
.diff-wrapper(ng-if="vm.type == 'response_plan'")

```

```

    include history-templates/history-response-plan
  .diff-wrapper(ng-if="vm.type == 'probability'")
    include history-templates/history-probability
  .diff-wrapper(ng-if="vm.type == 'impact'")
    include history-templates/history-impact
  .diff-wrapper(ng-if="vm.type == 'category'")
    include history-templates/history-category
  .diff-wrapper(ng-if="vm.type == 'risk_type'")
    include history-templates/history-risk-type
  .diff-wrapper(ng-if="vm.type == 'strategy'")
    include history-templates/history-strategy

```

Código B.93 – Blocos de diff para campos do modelo de risco em history-diff.jade

As alterações consistem em oito blocos acrescentados ao final do arquivo, cada um correspondendo a um campo do modelo **Risk** que pode aparecer como diferença no histórico.

A modificação é necessária porque o mecanismo de renderização de histórico do Taiga seleciona o sub-template a exibir pelo nome do campo (`vm.type`); campos sem um bloco correspondente neste template seriam silenciosamente ignorados no painel de histórico, tornando invisíveis as alterações nesses campos. Os campos adicionados — `cause`, `consequence`, `response_plan`, `probability`, `impact`, `category`, `risk_type` e `strategy` — são todos capturados pelo `risk_freezer` definido na Seção B.3.1.

Em atualizações futuras, este arquivo é modificado sempre que um novo campo rastreável é introduzido em qualquer entidade. Ao integrar novas versões, deve-se verificar se os oito blocos permanecem presentes; a ausência de qualquer um deles não interrompe a execução, mas omite as alterações do campo correspondente no histórico de riscos.

`app/modules/services/attachments.service.coffee`

O arquivo `attachments.service.coffee` define `AttachmentsService`, que resolve o identificador de *endpoint* de *upload* a partir do tipo de entidade passado pela diretiva de anexos. A modificação acrescentou as entradas para o tipo risco, conforme o Código B.94.

```

class AttachmentsService
  # ...
  types:
    epic: "epic", userstory: "us", task: "task",
    wiki: "wiki", wikipedia: "wiki"
    risks: "risk"
    risk: "risk"

```

Código B.94 – Mapeamento do tipo risco em AttachmentsService

As duas entradas — `risks` e `risk` — cobrem as variações do nome do tipo passado ao serviço por diferentes contextos da aplicação.

A modificação é necessária porque `AttachmentsService` usa o mapa para resolver o caminho do *endpoint* de *upload*; sem as entradas, o serviço não encontraria o tipo e o *upload* de arquivos em riscos falharia silenciosamente, sem erro explícito para o usuário.

Em atualizações futuras, o mapa de tipos é modificado sempre que um novo módulo introduz suporte a anexos. Ao integrar novas versões, deve-se verificar se as entradas de risco permanecem e se a chave de *endpoint* "`risk`" continua correspondendo à entrada `attachments/risk` definida em `resources.coffee` (Seção B.3.2).

`app/partials/admin/admin-project-modules.jade`

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.2: a inclusão de um *card* de ativação na página de administração de módulos do projeto. No módulo de gestão de riscos, o *card* usa a flag `project.is_risks_activated` e os textos `ADMIN.MODULES.RISKS` e `ADMIN.MODULES.RISKS_DESCRIPTION`, reutilizando o ícone `icon-module-issues` por ausência de ícone dedicado no *sprite* do front-end original, conforme o Código B.95.

```
.module.module-risks(ng-class="{true:'active',false:''}[project.
  is_risks_activated] ")
  .module-icon
    tg-svg(svg-icon="icon-module-issues")
  .module-name(translate="ADMIN.MODULES.RISKS")
  .module-desc(translate="ADMIN.MODULES.RISKS_DESCRIPTION")
  .module-activation.module-direct-active
    div.check
      input.activate-input(id="functionality-risks"
        type="checkbox"
        ng-checked="project.is_risks_activated"
        ng-model="project.is_risks_activated")
      span.check-text.check-yes(translate="COMMON.YES")
      span.check-text.check-no(translate="COMMON.NO")
```

Código B.95 – Card de ativação do módulo de riscos na administração de módulos

A justificativa e os pontos de atenção são os mesmos da Seção B.1.2. Em atualizações futuras, deve-se verificar se o *card* permanece presente e se a flag `is_risks_activated` continua correspondendo ao campo definido no modelo `Project` (Seção B.3.1).

`app/styles/dependencies/mixins/lightbox.scss`

O arquivo `lightbox.scss` em `dependencies/mixins/` define o *mixin* SCSS de posicionamento e exibição do *overlay* genérico de *lightbox*. A modificação ajustou o alinha-

mento vertical e adicionou suporte a rolagem, conforme o Código B.96.

```
&.open {  
  align-items: flex-start;  
  display: flex;  
  justify-content: center;  
  opacity: 1;  
  overflow-x: hidden;  
  overflow-y: auto;  
  padding: 3rem 1rem;  
  transition: opacity .3s ease;  
}
```

Código B.96 – Ajuste de alinhamento e rolagem no mixin de lightbox

As alterações compreendem: a mudança de `align-items` de `center` para `flex-start`, que ancora o conteúdo ao topo do *overlay* em vez de centralizá-lo verticalmente; e a adição de `overflow-y: auto` e `padding: 3rem 1rem`, que habilitam a rolagem vertical e adicionam espaçamento interno.

A modificação é necessária porque o formulário de criação e edição de risco contém mais campos do que os formulários para os quais o *lightbox* original foi projetado. Com o alinhamento central original, formulários altos ficavam parcialmente fora da tela em monitores de baixa resolução, sem possibilidade de rolar; a alteração garante que todo o conteúdo seja acessível independentemente do tamanho da janela.

Em atualizações futuras, este *mixin* afeta todos os *lightboxes* da aplicação, não apenas os de riscos. Ao integrar novas versões, deve-se verificar se as alterações de alinhamento e rolagem não comprometem a aparência de *lightboxes* de outros módulos cujo conteúdo se beneficiava do centramento vertical original.

app/styles/extras/dependencies.scss

O arquivo `app/styles/extras/dependencies.scss` importa todos os arquivos de *mixins* SCSS reutilizáveis da aplicação. A modificação acrescentou a importação do arquivo `ticket-data`, conforme o Código B.97.

```
@import '../dependencies/mixins/radio-group';  
@import '../dependencies/mixins/sidebar';  
@import '../dependencies/mixins/ticket-data';  
@import '../dependencies/mixins/slide';
```

Código B.97 – Importação do mixin ticket-data em dependencies.scss

A alteração consiste em uma única linha acrescentada. O *mixin* `category-choices`, usado pelo componente de seleção de nível e categoria de risco, foi extraído de `ticket-data.scss` para um arquivo de *mixin* dedicado (ver Seção B.3.2); a importação

aqui garante que o *mixin* esteja disponível em qualquer ponto da folha de estilos que o referencie.

Em atualizações futuras, a ordem e o conjunto de importações neste arquivo são alterados sempre que *mixins* são adicionados ou reorganizados. Ao integrar novas versões, deve-se verificar se a importação de `ticket-data` permanece presente; sua ausência causaria erro de compilação SCSS nos componentes que usam `@include category-choices()`.

`app/styles/modules/common/lightbox.scss`

O arquivo `app/styles/modules/common/lightbox.scss` define os estilos específicos do componente de criação e edição genérico (`.lightbox-create-edit`). A modificação ajustou as dimensões do formulário, estruturou os campos da coluna principal e adicionou a classe de estilo do campo de pontuação e nível de risco, conforme o Código B.98.

```
.lightbox-create-edit {
  $width: 760px;
  $sidebar-width: 220px;
  form {
    background: $white;
    flex-basis: $width;
    padding: 2rem;
  }
  .form-wrapper {
    .main {
      flex: 1;
      fieldset { margin-bottom: $spacing; width: 100%; }
      label { display: block; margin-bottom: .25rem; }
      input, textarea, .tags-block { display: block; width: 100%; }
      textarea.description { min-height: 10rem; }
    }
    .sidebar {
      background: $grey-02;
      flex-basis: $sidebar-width;
      padding: $spacing;
    }
  }
}

.risk-score-level {
  @include category-choices;
  cursor: default;
  margin-bottom: .5rem;
}
```

Código B.98 – Ajustes de layout e classe `.risk-score-level` no `lightbox.scss`

As alterações compreendem: o aumento da largura do formulário (700px → 760px) e da barra lateral (`calc(200px + 2rem)` → 220px) para acomodar os campos adicionais de risco; a estruturação explícita dos campos da coluna principal (`.main`) com largura total, *labels* visíveis e *textarea* redimensionável; o fundo cinza na barra lateral; e a classe `.risk-score-level`, que aplica o *mixin* `category-choices` com cursor padrão ao campo que exibe a pontuação e o nível calculados do risco.

A modificação é necessária porque o formulário genérico de criação e edição passou a ser reutilizado para riscos, que possuem mais campos e uma barra lateral com elementos de seleção por categoria; sem os ajustes de dimensionamento e estrutura, os campos transbordavam ou ficavam sem *labels* visíveis.

Em atualizações futuras, este arquivo afeta o *lightbox* genérico de todos os módulos. Ao integrar novas versões, deve-se verificar se as dimensões ajustadas não comprometem a aparência de *lightboxes* de outros módulos e se a classe `.risk-score-level` permanece presente para os componentes que a referenciam.

`app/styles/modules/common/ticket-data.scss`

O arquivo `ticket-data.scss` define estilos para elementos da barra lateral de detalhes de entidades (*ticket data*). A modificação realizou duas alterações distintas: extraiu o *mixin* `category-choices` do arquivo para um arquivo de *mixin* dedicado e adicionou o seletor `.assign-risk-button` à regra existente de `.assign-issue-button`, conforme o Código B.99.

```
// Bloco removido e movido para dependencies/mixins/ticket-data.scss:
// @mixin category-choices() { ... }

.ticket-assign-button {
  &.button-set {
    &.assign-issue-button,
    &.assign-risk-button {
      &.button-set {
        background: $color-link-tertiary;
        color: $color-white;
      }
    }
  }
}
```

Código B.99 – Extração do *mixin* e adição de `.assign-risk-button` em `ticket-data.scss`

A remoção do *mixin* `category-choices` deste arquivo e sua realocação para `dependencies/mixins/ticket-data.scss` foram necessárias porque o *mixin* passou a ser referenciado pelo componente de *lightbox* de riscos (Seção B.3.2), que está em um contexto SCSS diferente do módulo `ticket-data`; mover o *mixin* para a pasta de dependências — e importá-lo em `dependencies.scss` (Seção B.3.2) — garante sua disponibilidade em qualquer ponto da folha de estilos. A adição de `.assign-risk-button` ao seletor existente reutiliza o estilo do botão de atribuição de *issue* para o botão equivalente de riscos, sem duplicação de regras CSS.

Em atualizações futuras, o seletor de botões e a estrutura deste arquivo são mantidos pelo repositório original. Ao integrar novas versões, deve-se verificar se a entrada `.assign-risk-button` permanece no seletor e se o *mixin* `category-choices` não foi re-introduzido neste arquivo pelo repositório oficial (o que causaria duplicação com a versão em `dependencies/mixins/`).

B.3.3 Front-end-Next

`src/app/api/projects/projects.model.ts`

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.3: a inclusão de uma flag de ativação de módulo no tipo de projeto. No módulo de gestão de riscos, o campo adicionado é `isRisksActivated`, declarado na interface `Project`, conforme o Código B.100.

```
export interface Project {  
  // ...  
  isWatcher: boolean;  
  isWikiActivated: boolean;  
  isRisksActivated: boolean;  
  issueCustomAttributes: CustomAttribute[];  
  // ...  
}
```

Código B.100 – Campo `isRisksActivated` na interface `Project`

A justificativa é a mesma da Seção B.1.3: o campo TypeScript é a contraparte tipada da flag `is_risks_activated` do back-end e é lido pelo componente de navegação para condicionar a exibição do item de riscos no menu. Em atualizações futuras, deve-se verificar se o campo permanece na interface `Project` e se o nome em *camelCase* continua correspondendo ao campo *snake_case* do back-end após a transformação automática da API.

src/app/api/roles/roles.model.ts

O arquivo `roles.model.ts` define o *enum* `Permissions`, que centraliza as constantes de permissão usadas em toda a aplicação moderna. A modificação acrescentou as permissões de risco ao *enum*, conforme o Código B.101.

```
export enum Permissions {  
  // ...  
  deleteWikiPage = 'delete_wiki_page',  
  // Risk  
  viewRisks = 'view_risks',  
  addRisk = 'add_risk',  
  modifyRisk = 'modify_risk',  
  deleteRisk = 'delete_risk',  
  // Project  
  viewProject = 'view_project',  
  // ...  
}
```

Código B.101 – Permissões de risco no *enum* `Permissions`

As alterações consistem em quatro entradas acrescentadas ao *enum*, cobrindo as permissões de visualização, adição, modificação e exclusão de riscos. A permissão `comment_risk`, presente no back-end, não possui entrada no *enum* porque a aplicação moderna não implementa a funcionalidade de comentários diretamente nesta camada.

A modificação é necessária porque o componente de navegação (Seção B.3.3) e os testes de especificação referenciam as permissões de risco por meio das constantes deste *enum*; o uso de *strings* literais em vez de constantes tipadas quebraria a verificação de tipos e dificultaria refatorações futuras.

Em atualizações futuras, os valores de *string* do *enum* (`'view_risks'`, `'add_risk'` etc.) precisam corresponder exatamente aos codinomes declarados no back-end em `taiga/permissions/choices.py` (Seção B.3.1); uma divergência causaria falhas silenciosas na verificação de permissões.

src/app/commons/components/svg-sprite/svg-sprite.component.html

A alteração realizada neste arquivo é da mesma natureza da descrita para o módulo de cronograma na Seção B.1.3: a adição de um símbolo SVG ao *sprite* da aplicação moderna. No módulo de gestão de riscos, o símbolo acrescentado é `warning`, usado como ícone do item de riscos no menu lateral, conforme o Código B.102.

```
<svg ...>  
  <!-- ... -->  
  <symbol id="warning" viewBox="0 0 24 24">
```

```
<path fill-rule="evenodd" clip-rule="evenodd"
      d="M12 2C11.5 2 11 2.3 ... Z" />
</symbol>
</svg>
```

Código B.102 – Símbolo warning no sprite SVG do taiga-front-next

A justificativa e os pontos de atenção são os mesmos da Seção B.1.3: o template do menu referencia o símbolo por `<use href="#warning"/>`; sem o símbolo aqui declarado, o ícone do item de riscos no menu não seria renderizado. Vale observar que este *sprite* é distinto do *sprite* do *taiga-front*, no qual o ícone equivalente está identificado como *icon-module-issues*.

`src/app/commons/legacy/legacy-loader.component.html`

O arquivo `legacy-loader.component.html` é o template do componente Angular responsável por carregar componentes da aplicação moderna a partir de chamadas originárias do shell AngularJS legado. Ele utiliza um `ngSwitch` para instanciar o componente correto com base no identificador de tipo recebido. A modificação acrescentou o caso para o componente de matriz de riscos, conforme o Código B.103.

```
<ng-container *ngSwitchCase="'tg-project-navigation'">
  <tg-project-navigation ...></tg-project-navigation>
</ng-container>
<ng-container *ngSwitchCase="'tg-risks-matrix'">
  <tg-risks-matrix
    [projectId]="params?.projectId"
    [projectSlug]="params?.projectSlug">
  </tg-risks-matrix>
</ng-container>
```

Código B.103 – Caso tg-risks-matrix no loader legado

A alteração consiste em um único bloco `ng-container` acrescentado, que instancia o componente `tg-risks-matrix` com os parâmetros de projeto recebidos do shell legado.

A modificação é necessária porque a matriz de riscos foi implementada como um componente Angular moderno embutido na rota AngularJS `/project/:pslug/risks/matrix`; o shell legado delega a renderização ao *loader* pelo identificador de tipo do componente, e sem o caso correspondente o componente não seria instanciado, deixando a página da matriz em branco.

Em atualizações futuras, este arquivo é modificado sempre que um novo componente Angular é exposto via o mecanismo legado. Ao integrar novas versões, deve-se verificar se o caso `tg-risks-matrix` permanece e se os atributos `projectId` e `projectSlug` continuam correspondendo às propriedades de entrada (`@Input`) do componente.

`src/app/commons/legacy/legacy.module.ts`

O arquivo `legacy.module.ts` declara o módulo Angular que encapsula o sistema de carregamento legado, registrando os módulos de funcionalidade cujos componentes podem ser instanciados pelo *loader*. A modificação importou e registrou `RisksMatrixModule`, conforme o Código B.104.

```
import { ProjectNavigationModule } from '../project-navigation/...';
import { RisksMatrixModule } from '../risks-matrix/risks-matrix.module';
// ...

@NgModule({
  imports: [
    CommonModule,
    ProjectNavigationModule,
    RisksMatrixModule,
  ],
  // ...
})
export class LegacyModule { }
```

Código B.104 – Importação de `RisksMatrixModule` em `legacy.module.ts`

As alterações compreendem a importação do módulo e sua inclusão no array `imports` do decorador `@NgModule`.

A modificação é inevitável no código original porque o Angular requer que módulos sejam explicitamente importados para que seus componentes estejam disponíveis; sem a importação de `RisksMatrixModule` aqui, o caso `tg-risks-matrix` no *loader* (Seção B.3.3) falharia com erro de componente desconhecido.

Em atualizações futuras, o array `imports` de `LegacyModule` é modificado sempre que um novo componente Angular é exposto via o sistema legado. Ao integrar novas versões, deve-se verificar se `RisksMatrixModule` permanece importado; sua ausência tornaria a página da matriz de riscos inacessível.

`src/app/commons/project-navigation/project-navigation.component.html`

A alteração realizada neste arquivo é da mesma natureza das descritas para o módulo de cronograma (Seção B.1.3) e para o módulo de gestão financeira (Seção B.2.3): a adição de um item de menu com sub-navegação ao template do componente de navegação lateral. No módulo de gestão de riscos, o item acrescentado é um *dropdown* com três sub-itens — lista de riscos, matriz e evolução — além do ajuste na lógica do diálogo de *popup* para incluir o tipo `risks`, conforme o Código B.105.

```
<li *ngIf="isMenuRisksEnabled" class="menu-option scrum"
```

```

    [class.active]="risksVisible" ...>
    <button (click)="toggleRisks()" type="button">
      <svg><use href="#warning" /></svg>
      <span ...>{{'PROJECT.SECTION.RISKS' | translate}}</span>
    </button>
    <ul class="child-menu">
      <li class="menu-option">
        <a [routerLink]="[baseHref,'project',project.slug,'risks']"
          routerLinkActive="active">
          {{'PROJECT.SECTION.RISKS_LIST' | translate}}
        </a>
      </li>
      <li class="menu-option">
        <a [routerLink]="[...,'risks','matrix']" routerLinkActive="active">
          {{'PROJECT.SECTION.RISKS_MATRIX' | translate}}
        </a>
      </li>
      <li class="menu-option">
        <a [routerLink]="[...,'risks','evolution']" routerLinkActive="
active">
          {{'PROJECT.SECTION.RISKS_DASHBOARD' | translate}}
        </a>
      </li>
    </ul>
  </li>

<!-- Ajuste no diálogo de popup -->
<a *ngIf="dialog.slug && dialog.type !== 'scrum'
  && dialog.type !== 'risks'" ...>

```

Código B.105 – Item de menu de riscos no template de navegação

A justificativa e os pontos de atenção são análogos aos das Seções B.1.3 e B.2.3. Em atualizações futuras, qualquer refatoração do template de navegação pelo repositório oficial pode conflitar com o bloco do item de riscos e com o ajuste na condição do diálogo.

src/app/commons/project-navigation/project-navigation.component.ts

A alteração realizada neste arquivo é da mesma natureza das descritas para o módulo de cronograma (Seção B.1.3) e para o módulo de gestão financeira (Seção B.2.3): a adição da lógica de controle do *dropdown* e de visibilidade do novo item de menu ao componente de navegação. No módulo de gestão de riscos, foram acrescentados a propriedade `risksVisible`, o *getter* `isMenuRisksEnabled`, os métodos `toggleRisks()`

e `popupRisks()`, e a inicialização do estado persistido em `localStorage`, conforme o Código B.106.

```
export class ProjectNavigationComponent ... {
    public scrumVisible = false;
    public risksVisible = false;
    // ...

    ngOnChanges() {
        // ...
        if (this.section === 'backlog') {
            this.scrumVisible = (localStorage.getItem('...') === 'true');
        }
        if (this.section === 'risks') {
            this.risksVisible =
                (localStorage.getItem('projectnav-risks') === 'true');
        }
    }

    get isMenuRisksEnabled() {
        return this.project.isRisksActivated &&
            this.project.myPermissions.includes(Permissions.viewRisks);
    }

    public toggleRisks() {
        if (this.collapsed) {
            this.router.navigate(['/project', this.project.slug, 'risks']);
        } else {
            this.risksVisible = !this.risksVisible;
            localStorage.setItem('projectnav-risks', String(this.risksVisible))
        }
    }

    public popupRisks(event: MouseEvent) { ... }

    public toggleCollapse() {
        this.collapsed = !this.collapsed;
        if (this.collapsed) {
            this.scrumVisible = false;
            this.risksVisible = false;
        }
    }
}
```

```

    }
}

```

Código B.106 – Lógica de dropdown de riscos no componente de navegação

As alterações espelham o padrão já estabelecido pelos módulos anteriores: `isMenuRisksEnabled` condiciona a visibilidade do item à flag `isRisksActivated` e à permissão `viewRisks`; `toggleRisks()` navega diretamente quando o menu está recolhido ou alterna o *dropdown* quando expandido, persistindo o estado em `localStorage`; `popupRisks()` inicializa o diálogo de *popup* com os três sub-itens quando o menu está recolhido; e `risksVisible` é zerado ao recolher o menu.

A justificativa e os pontos de atenção são análogos aos das Seções B.1.3 e B.2.3. Em atualizações futuras, qualquer refatoração do componente de navegação pelo repositório oficial que altere o padrão de *dropdown* ou a estrutura de persistência em `localStorage` pode conflitar com os métodos aqui adicionados.

`src/app/webcomponent.module.ts`

O arquivo `webcomponent.module.ts` declara o módulo Angular de nível superior usado quando a aplicação é distribuída como um conjunto de *web components* reutilizáveis. A modificação importou e registrou `RisksMatrixModule`, conforme o Código B.107.

```

import { ProjectNavigationModule } from '../commons/project-navigation/...';
import { RisksMatrixModule } from '../commons/risks-matrix/risks-matrix.module';
// ...

@NgModule({
  imports: [
    // ...
    ProjectNavigationModule,
    RisksMatrixModule,
    TranslateModule.forRoot(),
    // ...
  ],
})
export class WebcomponentModule { }

```

Código B.107 – Importação de `RisksMatrixModule` em `webcomponent.module.ts`

A alteração é análoga à realizada em `legacy.module.ts` (Seção B.3.3), porém no contexto do módulo de *web components*: sem a importação de `RisksMatrixModule` aqui, o componente `tg-risks-matrix` não estaria disponível quando a aplicação for distribuída como *web component*, mantendo consistência entre os dois pontos de entrada da aplicação.

Em atualizações futuras, o array `imports` deste módulo acompanha o de `LegacyModule`: toda vez que um novo módulo é adicionado ao sistema legado, deve-se verificar se ele também precisa ser registrado aqui. Ao integrar novas versões, deve-se verificar se `RisksMatrixModule` permanece importado em ambos os módulos.

APÊNDICE C – DISPONIBILIZAÇÃO DO CÓDIGO-FONTE

Este apêndice indica o local de acesso ao código-fonte resultante do desenvolvimento realizado neste trabalho.

Todo o código produzido, abrangendo os repositórios `taiga-back`, `taiga-front` e `taiga-front-next`, bem como os arquivos de orquestração e os *scripts* de execução do ambiente local descritos no Capítulo 5, está disponível publicamente no repositório do Grupo de Qualidade de Software (GQS) da Universidade Federal de Santa Catarina, no seguinte endereço:

`https://codigos.ufsc.br/gqs/taiga-gqs`

Conforme apresentado na Seção 5 (página 74), esse repositório reúne os três submódulos da aplicação em um único ponto de acesso, permitindo clonar, construir e reproduzir o ambiente de desenvolvimento por meio dos passos detalhados naquele capítulo.

APÊNDICE D – ARTIGO CIENTÍFICO EM FORMATO SBC

Este apêndice reúne a versão resumida deste Trabalho de Conclusão de Curso, elaborada em formato de artigo científico conforme o modelo da Sociedade Brasileira de Computação (SBC). O artigo apresenta uma síntese da motivação, da fundamentação teórica, da análise comparativa das ferramentas estudadas, da proposta de solução e dos resultados obtidos com a evolução da ferramenta de código aberto Taiga.io. Nas páginas seguintes, o artigo é reproduzido integralmente, preservando a formatação original exigida pelo modelo da SBC.

Evolução de uma ferramenta open source de gerenciamento ágil de projetos: integrando riscos, cronograma e finanças

Matheus E. L. Feliciano¹, Shaiane B. R. Borges¹, Jean Carlo R. Hauck¹

¹ Departamento de Informática e Estatística (INE)

Centro Tecnológico (CTC)

Universidade Federal de Santa Catarina (UFSC)

Campus Universitário Reitor João David Ferreira Lima

Caixa Postal 476 – CEP 88040-900 – Florianópolis – SC – Brasil

matheus.lafeta.moc@gmail.com, shaianeboesingrb@gmail.com, jean.hauck@ufsc.br

Abstract. *Project management tools have become essential for planning, executing, and monitoring projects, especially with the growth of agile methods. However, few open-source platforms offer integrated support that simultaneously covers agile practices and the more detailed processes of prescriptive management, such as risk analysis, cost control, and precise schedule management. This paper presents the evolution of the open-source tool Taiga.io through the implementation of three modules: schedule management with critical path identification, budget management, and risk management. The solution extended the Django back-end and the Angular front-end while preserving the tool's original modular architecture. As a result, Taiga.io now brings together, in a single free environment, a Gantt chart with critical path and activity slack, a consolidated financial dashboard, and a probability $\times$ impact matrix with historical risk tracking, showing that it is feasible to bring prescriptive management practices closer to an agile environment.*

Resumo. *Ferramentas de gestão de projetos têm se mostrado essenciais no apoio ao planejamento, à execução e ao monitoramento de projetos, especialmente com o crescimento do uso de métodos ágeis. No entanto, poucas plataformas open source oferecem suporte integrado que contemple, ao mesmo tempo, práticas ágeis e processos mais detalhados do gerenciamento prescritivo, como análise de riscos, controle de custos e gerenciamento preciso de cronogramas. Este artigo apresenta a evolução da ferramenta open source Taiga.io por meio da implementação de três módulos: gestão de cronograma com identificação do caminho crítico, gestão orçamentária e gestão de riscos. A solução estendeu o back-end em Django e o front-end em Angular, preservando a arquitetura modular original da ferramenta. Como resultado, o Taiga.io passa a reunir, em um único ambiente livre, um gráfico de Gantt com caminho crítico e folga das atividades, um painel financeiro consolidado e uma matriz de probabilidade $\times$ impacto com acompanhamento histórico dos riscos, evidenciando a viabilidade de aproximar práticas de gerenciamento prescritivo de um ambiente baseado em métodos ágeis.*

1. Introdução

As práticas de gerenciamento de projetos evoluíram seguindo caminhos distintos, o que fez com que os métodos ágeis e as abordagens de gerenciamento prescritivo nem sempre

conversassem de forma efetiva. Enquanto os modelos de ciclo de vida iterativos priorizam adaptação contínua, ciclos curtos e flexibilidade operacional, o gerenciamento prescritivo baseia-se em previsibilidade, planejamento detalhado e análise estruturada, como sistematizado no Guia PMBOK [Project Management Institute 2017]. Essa separação cria dificuldades práticas quando se busca integrar, em um mesmo ambiente, tanto a agilidade necessária para lidar com mudanças constantes quanto o rigor analítico indispensável a estimativas, controles e tomada de decisão.

De um lado, os métodos ágeis impulsionam colaboração contínua e adaptação rápida, mas frequentemente carecem de instrumentos para análises aprofundadas. Em sentido oposto, as práticas consolidadas de gerenciamento prescritivo oferecem rigor, mas não acompanham com a mesma agilidade as iterações rápidas típicas de equipes que atuam em contextos dinâmicos. Essa diferença reflete-se diretamente nas ferramentas: soluções orientadas à agilidade priorizam quadros visuais e ciclos iterativos, enquanto soluções prescritivas priorizam modelos de planejamento estruturado, raramente oferecendo suporte integrado aos dois modelos.

Essa assimetria revela uma lacuna amplamente discutida na literatura. Vierlboeck et al. [Vierlboeck et al. 2019] apontam a ausência de modelos consolidados que integrem práticas orçamentárias ao desenvolvimento ágil, evidenciando limitações na dimensão financeira do planejamento iterativo. Dingsøyr e Petit [Dingsøyr and Petit 2021] destacam a carência de abordagens maduras para gestão de riscos em contextos que combinam desenvolvimento ágil e gerenciamento prescritivo, reforçando que incertezas estruturais permanecem pouco tratadas em modelos iterativos. Bloss et al. [Bloss et al. 2025], por sua vez, evidenciam a importância de análises de caminho crítico apoiadas em modelos probabilísticos para aumentar a confiabilidade do planejamento, um elemento que os métodos ágeis não incorporam de forma nativa.

Atualmente há diversas ferramentas de gerenciamento ágil de projetos disponíveis. Dentre elas, o Taiga.io, open source e orientado a métodos ágeis, apresenta-se como um caso representativo dessa limitação: apesar de apoiar bem ciclos iterativos, não oferece recursos nativos para uma gerência mais abrangente, carecendo de funcionalidades de gestão de riscos, caminho crítico e orçamento. Expandir essas capacidades dentro de um ambiente ágil permite integrar práticas de gerenciamento prescritivo em uma única ferramenta e, por se tratar de uma solução livre e passível de implantação local, amplia também seu uso em contextos acadêmicos.

Diante desse cenário, o objetivo deste trabalho é evoluir a ferramenta open source Taiga.io, incorporando três módulos integrados ao seu ambiente ágil: análise de caminho crítico no cronograma, gestão orçamentária e acompanhamento contínuo de riscos. Este artigo constitui uma versão resumida do Trabalho de Conclusão de Curso correspondente e está organizado da seguinte forma: a Seção 2 apresenta a fundamentação teórica; a Seção 3 discute os trabalhos relacionados e a seleção da ferramenta; a Seção 4 descreve a proposta e a arquitetura; a Seção 5 detalha a implementação e os resultados; e a Seção 6 apresenta as conclusões.

2. Fundamentação Teórica

O embasamento deste trabalho apoia-se em três áreas de conhecimento do gerenciamento prescritivo de projetos preconizadas pelo Guia PM-

BOK [Project Management Institute 2017]: cronograma, custos e riscos, além da arquitetura de aplicações Web.

2.1. Gestão de Cronograma

A gestão de cronograma envolve não apenas a estimativa da duração de cada tarefa, mas também a ordem em que elas ocorrem e suas dependências. O Guia PMBOK organiza esse processo em etapas que incluem a definição e o sequenciamento das atividades, a estimativa de duração e a elaboração do cronograma [Project Management Institute 2017]. Barcaui complementa essa visão com um enfoque prático, ressaltando que a determinação da programação combina ciência e arte: a ciência identifica o caminho crítico e as folgas, enquanto a arte garante o cumprimento do cronograma mesmo diante de incertezas [Barcaui 2012].

O método do caminho crítico é a técnica usada para estimar a duração mínima do projeto. Ele calcula as datas de início e término mais cedo e mais tarde de cada atividade e identifica o caminho mais longo, que determina a menor duração possível do projeto; nesse caminho, a folga total costuma ser igual a zero [Project Management Institute 2017]. A folga total corresponde ao tempo que uma atividade pode atrasar sem impactar a conclusão do projeto; quando essa folga é nula, a atividade é considerada crítica, pois qualquer atraso afeta diretamente o prazo final. Uma forma usual de representar visualmente o cronograma é o diagrama de Gantt, no qual atividades são dispostas em barras horizontais sobre uma escala temporal, permitindo visualizar dependências, marcos e atividades críticas, além de comparações entre o planejado e o executado [Barcaui 2012].

2.2. Gestão de Custos

O gerenciamento de custos reúne processos que buscam estimar, orçar e controlar os gastos do projeto, a fim de garantir sua conclusão dentro do orçamento aprovado [Project Management Institute 2017]. Estimar custos é realizar uma previsão quantitativa dos recursos financeiros necessários para executar as atividades; determinar o orçamento é consolidar essas estimativas em uma linha de base de custos, que passa a servir de referência para o acompanhamento financeiro. O desempenho do projeto é medido pela comparação entre os valores planejados e os efetivamente realizados [Barbosa et al. 2019].

2.3. Gestão de Riscos

O PMBOK define risco como um evento ou condição incerta que, ao se concretizar, pode gerar impactos negativos (ameaças) ou positivos (oportunidades). Um risco possui três componentes: o evento, a probabilidade de ocorrer e o impacto sobre os objetivos do projeto [Project Management Institute 2017, Salles Jr. 2010]. O processo de gerenciamento contempla a identificação, a análise e o monitoramento dos riscos. Na análise, é comum estruturar os riscos por categorias e priorizá-los por meio de uma matriz de probabilidade $\times$ impacto, na qual a pontuação resulta da combinação desses dois fatores. O monitoramento é contínuo ao longo do ciclo de vida do projeto, acompanhando a evolução dos riscos já registrados e a identificação de novos [Project Management Institute 2017].

2.4. Arquitetura e Tecnologias Web

Para suportar essas práticas em software, a arquitetura de aplicações Web separa responsabilidades. No back-end, é comum a linguagem Python integrada ao framework Django, que expõe uma API REST apoiada em um banco de dados relacional como o PostgreSQL. No front-end, frameworks como Angular consomem essa API e renderizam interfaces dinâmicas de página única. A execução por meio de contêineres Docker favorece o desenvolvimento desacoplado e a reprodução consistente do ambiente.

3. Trabalhos Relacionados e Seleção da Ferramenta

Para fundamentar o desenvolvimento, realizou-se uma análise comparativa de oito ferramentas de gerenciamento de projetos: Taiga, Monday, OpenProject, ProjectManager, Jira, Artia, Smartsheet e Zoho Projects. A avaliação considerou quatro eixos, atribuindo a cada um uma pontuação de 0 a 2, em que 0 indica ausência de suporte, 1 indica suporte parcial ou limitado e 2 indica suporte abrangente para o contexto deste estudo. Os eixos são: F1 – gestão de cronograma (gráfico de Gantt, caminho crítico, folga das atividades e integração com sprints, épicas, histórias e tarefas); F2 – gestão financeira (custo por tarefa, hora trabalhada por integrante, receita e visão consolidada); F3 – gestão de riscos (registro, análise qualitativa, matriz probabilidade $\times$ impacto e monitoramento); e F4 – possibilidade de implantação on-premise com todas as funcionalidades disponíveis de forma gratuita, sem paywall funcional. A Tabela 1 sintetiza os resultados.

Tabela 1. Matriz de funcionalidades das ferramentas analisadas.

Ferramenta	F1	F2	F3	F4	Total
Taiga	1	0	0	2	3
Monday	2	1	1	0	4
OpenProject	2	2	1	1	6
ProjectManager	2	2	1	0	5
Jira	1	1	1	0	3
Artia	2	1	1	0	4
Smartsheet	2	1	1	0	4
Zoho Projects	2	2	1	0	5
Taiga (proposto)	2	2	2	2	8

A análise evidencia um padrão. Ferramentas voltadas ao *Project Portfolio Management* (PPM), como OpenProject, ProjectManager, Artia e Zoho Projects, tendem a apresentar níveis mais elevados em F1 e F2, oferecendo gráficos de Gantt com dependências, caminho crítico e linha de base, além de módulos de orçamento e custos. Contudo, permanecem predominantemente comerciais: mesmo quando disponibilizam opção on-premise, o acesso aos recursos mais avançados depende de licenças pagas, o que explica notas baixas em F4. Ferramentas com forte orientação ágil e colaborativa, como Monday, Smartsheet e Jira, apresentam perfil intermediário; nos eixos financeiro e de riscos, apoiam-se em colunas numéricas, fórmulas e integrações, sem módulos dedicados que cubram integralmente as práticas de orçamento e gerenciamento de riscos.

O Taiga, foco deste trabalho, ocupa posição restrita quando se consideram apenas F1, F2 e F3: obtém nível 1 em F1, por sua boa cobertura ágil sem recursos clássicos de

cronograma, e nível 0 em F2 e F3, por não dispor de módulos nativos de custos ou de riscos. Por outro lado, é a única ferramenta a atingir nível 2 em F4, permitindo implantação on-premise com acesso a todas as funcionalidades sem custo de licença. Além dessa aderência à filosofia open source, o Taiga.io foi selecionado por sua proximidade tecnológica com o back-end em Python/Django, banco PostgreSQL, front-end em Angular e orquestração por Docker e por concentrar o fluxo de trabalho em elementos centrais dos métodos ágeis, o que favorece seu uso didático. Elevar seu suporte nos três eixos, sem descaracterizar sua natureza livre, aproxima-o do nível de maturidade das soluções comerciais e o torna uma alternativa singular entre as ferramentas analisadas, como indica a linha *Taiga (proposto)* da Tabela 1.

4. Proposta de Solução e Arquitetura

A proposta organiza-se em três épicos, alinhados às práticas de métodos ágeis já adotadas pela ferramenta: E1 (Gestão de Cronograma), E2 (Gestão de Custos) e E3 (Gestão de Riscos). Cada épico agrupa histórias de usuário que representam funcionalidades desejadas sob a perspectiva de gerentes, membros da equipe e demais partes interessadas.

Do ponto de vista arquitetural, o Taiga é organizado de forma modular, separando interface, regras de negócio, persistência e serviços auxiliares (Figura 1). A interação ocorre pelo navegador, que acessa o front-end, composto por dois projetos que coexistem: o `taiga-front`, aplicação principal legada em AngularJS, e o `taiga-front-next`, em Angular e TypeScript, resultado de uma iniciativa de modernização parcial. As requisições são processadas pelo `taiga-back`, back-end em Python com Django e Django REST Framework, que expõe a API REST, valida dados, controla permissões e persiste as informações no PostgreSQL. Serviços auxiliares tratam de comunicação em tempo real e processamento assíncrono.

A solução preservou essa arquitetura. As três funcionalidades foram implementadas como aplicativos Django independentes (`taiga.projects.schedule`, `taiga.projects.finances` e `taiga.projects.risks`), adicionados à configuração do back-end e seguindo o padrão dos módulos nativos. Cada aplicativo foi estruturado em camadas (modelos, serializadores, visões, permissões, serviços e sinais), isolando o novo código do núcleo original e reduzindo a superfície de conflito em futuras atualizações. A conexão com o restante da ferramenta ocorre de forma desacoplada por meio de *signals* do Django, que permitem reagir a eventos das entidades existentes (épicos, histórias e tarefas) sem alterar diretamente seu código. No modelo de dados, novas tabelas foram criadas e versionadas por migrações, todas vinculadas à entidade `project`, entre elas `schedule_scheduledependency` (dependências entre atividades), `finances_activitycost` (agregados financeiros por atividade) e `risks_riskevaluation` (histórico de avaliações de risco). No front-end, os novos componentes foram desenvolvidos em Angular moderno, compilados como *web components* e integrados à aplicação legada, viabilizando o uso de tecnologias mais recentes sem reescrever integralmente o sistema.

5. Implementação e Resultados

5.1. Gestão de Cronograma

O épico de cronograma traduz os épicos, histórias e tarefas do backlog ágil para uma visão de planejamento temporal em duas páginas principais. A página de listagem concentra

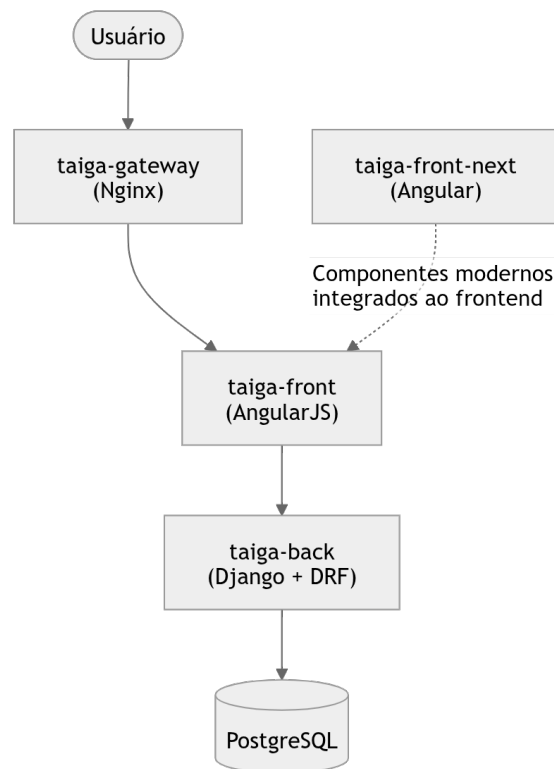


Figura 1. Visão geral da arquitetura da plataforma Taiga.

a edição das datas de início prevista, início real e término, enquanto a página de Gantt representa graficamente essas informações. No diagrama, as atividades são exibidas em estrutura de árvore à esquerda e como barras à direita, com escala ajustável por dias, semanas ou meses; as dependências são representadas por linhas de conexão que evidenciam a ordem lógica de execução e servem de base para o cálculo do caminho crítico (Figura 2).

O cálculo do caminho crítico é realizado no front-end, a partir das datas e dependências já carregadas na página. A implementação estima as durações, constrói um grafo direcionado com base nas dependências e identifica os caminhos relevantes do cronograma. Duas decisões de projeto adaptam o método a projetos reais: o caminho crítico é definido como o conjunto de atividades com a *menor folga* calculada, e não estritamente com folga igual a zero, o que lida melhor com intervalos, datas fixas e períodos de espera; e apenas os caminhos que alcançam a atividade de última data de término do projeto são considerados candidatos, de modo que sequências que não contribuem para a conclusão final não são classificadas como críticas. Ao acionar a visualização, o sistema realça as atividades e ligações críticas e reduz a opacidade das demais, permitindo identificar rapidamente os gargalos. Para cada atividade, ao posicionar o cursor, exibe-se a folga total calculada, indicando quanto ela pode atrasar sem afetar a data final considerada.

5.2. Gestão de Custos

O épico financeiro complementa os elementos existentes com informações de custos e receitas. Épicos, histórias e tarefas passam a registrar horas planejadas e custo planejado, com consolidação hierárquica: quando uma atividade possui subordinadas, seus valores compõem os totais do item pai. O custo realizado, por sua vez, é derivado de registros

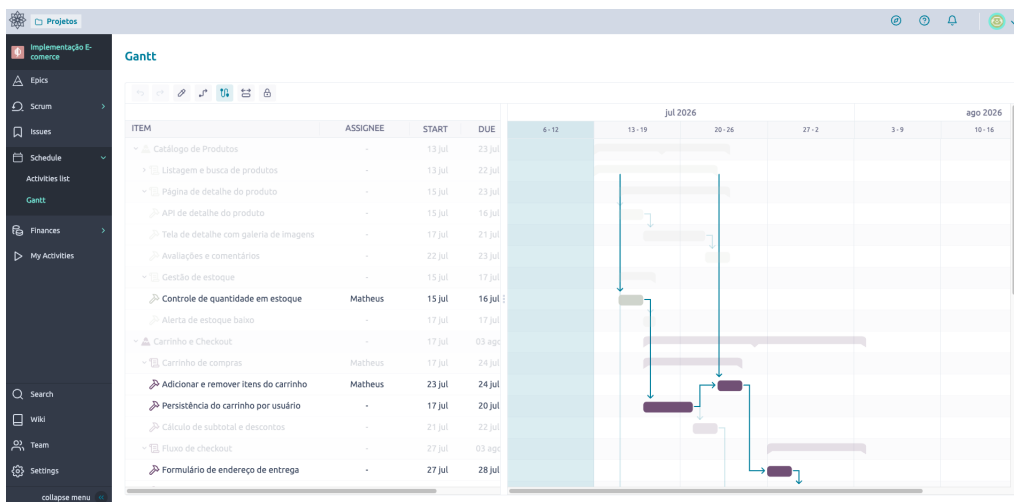


Figura 2. Caminho crítico destacado no diagrama de Gantt implementado.

reais de trabalho: cada membro tem um valor/hora configurado e, ao utilizar um temporizador em suas atividades, gera um lançamento de tempo cujo custo é calculado com base no valor/hora vigente no momento do apontamento, valor que é armazenado junto ao registro para preservar o histórico, sem recálculo retroativo. O módulo permite ainda cadastrar múltiplas receitas do projeto, com valor, finalidade e vigência.

Esses dados alimentam um dashboard financeiro (Figura 3), que consolida indicadores como custo planejado total, custo realizado, receita recebida, saldo financeiro atual, total de horas trabalhadas e custo médio por hora. A página apresenta também gráficos de apoio, custo real em comparação com receitas ao longo do tempo, custos por categoria e custos por papel dos membros, permitindo acompanhar a situação econômica do projeto sem depender de planilhas externas.

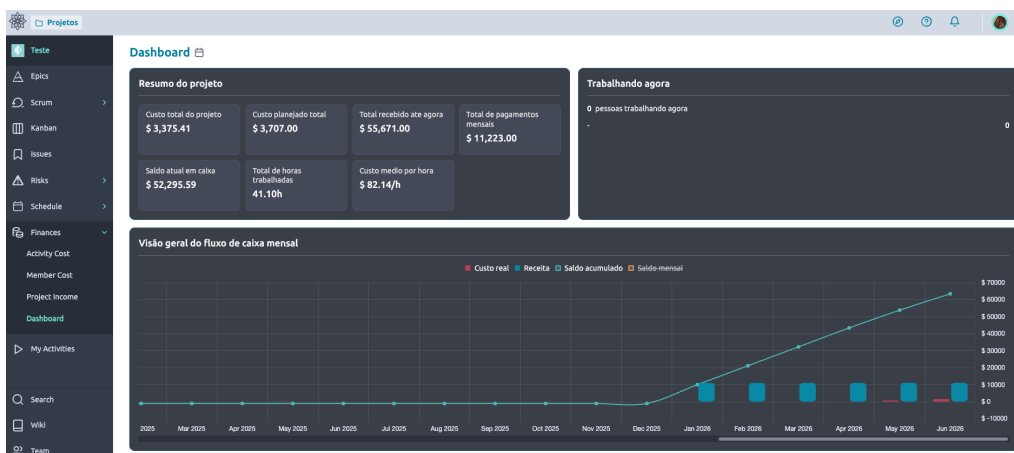


Figura 3. Indicadores e gráficos consolidados no dashboard financeiro do projeto.

5.3. Gestão de Riscos

O épico de riscos evolui de um simples cadastro para um sistema de avaliação quantitativa. Cada risco é registrado com informações como assunto, causa, consequência, categoria,

tipo (ameaça ou oportunidade), estratégia de resposta, responsável e plano de resposta. A criticidade é calculada pela multiplicação entre probabilidade e impacto, ambos em escala de 1 a 5, gerando uma pontuação de 1 a 25 que o sistema classifica automaticamente em quatro níveis: baixo (1 a 4), médio (5 a 9), alto (10 a 14) e crítico (15 a 25). A listagem principal oferece busca textual, filtros avançados, ordenação por colunas e ações diretas, como alteração de status e atribuição de responsável.

Esses dados são dispostos graficamente em uma matriz de probabilidade $\times$ impacto (Figura 4), que posiciona os riscos por criticidade, com cores por faixa de severidade, e integra-se à listagem para apoiar a priorização. Além da análise instantânea, implementou-se o acompanhamento histórico dos riscos por meio de avaliações (*evaluations*) registradas ao longo do tempo. Uma avaliação é criada automaticamente sempre que ocorre uma alteração relevante na criticidade, mudança de probabilidade, de impacto ou de associação a uma sprint, podendo também ser adicionada manualmente. Essas avaliações compõem gráficos temporais que revelam tendências de aumento ou redução da criticidade, disponíveis tanto na visão individual do risco quanto em um painel consolidado que compara múltiplos riscos simultaneamente.

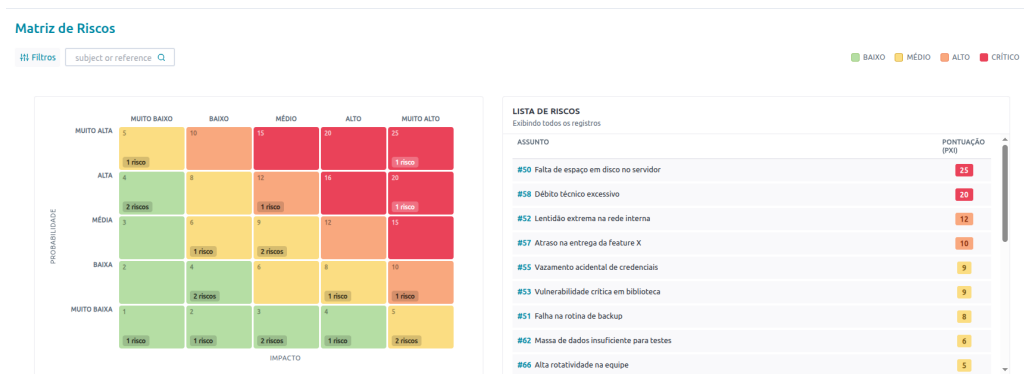


Figura 4. Matriz de probabilidade $\times$ impacto do módulo de gestão de riscos.

5.4. Síntese dos Resultados

Os três módulos foram implementados e integrados ao ambiente ágil do Taiga.io, preservando sua natureza livre e a possibilidade de implantação local. A Tabela 2 resume a principal funcionalidade entregue por cada um.

Tabela 2. Síntese dos módulos implementados e suas principais funcionalidades.

Módulo	Principal funcionalidade
Cronograma	Gráfico de Gantt com caminho crítico e folga das atividades
Financeiro	Dashboard financeiro (custos planejados/realizados e receitas)
Riscos	Matriz probabilidade $\times$ impacto e evolução histórica

Com essas adições, a ferramenta passa a reunir, em um único ambiente open source, suporte a cronograma, custos e riscos, reduzindo a lacuna entre as abordagens ágil e prescritiva identificada na análise comparativa. Os requisitos foram verificados por

meio de testes funcionais documentados a partir dos critérios de aceitação das histórias de usuário, confirmando o correto funcionamento das operações de cadastro, consulta, cálculo automático de pontuações e classificações, e a coerência das visualizações de Gantt, dashboard e matriz de riscos com os dados registrados.

6. Conclusão

Este trabalho apresentou a evolução da ferramenta open source Taiga.io por meio da implementação de três módulos, gestão de cronograma com caminho crítico, gestão orçamentária e gestão de riscos, atendendo ao objetivo de aproximar práticas de gerenciamento prescritivo de um ambiente baseado em métodos ágeis. No eixo de cronograma, a ferramenta passou a estruturar atividades, representar dependências e identificar o caminho crítico e a folga das atividades em um gráfico de Gantt. No eixo financeiro, tornou-se possível registrar custos planejados e realizados, taxas horárias por membro e receitas, consolidando tudo em um painel financeiro. No eixo de riscos, a ferramenta passou a oferecer registro, análise por matriz probabilidade $\times$ impacto e acompanhamento da evolução dos riscos.

A solução foi construída preservando a arquitetura modular original do Taiga.io. Ao isolar a implementação em aplicativos Django independentes conectados por *signals*, buscou-se manter a sustentabilidade do código diante de futuras atualizações da base, ainda que a integração completa com os fluxos nativos tenha exigido alterações pontuais em arquivos centrais, documentadas para orientar manutenções futuras. Com a inclusão de análise de caminho crítico, dashboard financeiro e matriz de riscos, a nova versão adquire características tipicamente associadas a soluções comerciais, mantendo-se livre e irrestrita. Isso amplia sua aplicabilidade tanto em organizações, como alternativa acessível de implantação própria, quanto em contextos acadêmicos, ao permitir explorar cronograma, custos e riscos de forma integrada em uma única plataforma aderente a métodos ágeis.

Como trabalhos futuros, propõe-se a avaliação da ferramenta em ambiente acadêmico, especialmente em disciplinas de gerência de projetos e engenharia de software, com estudos de usabilidade das novas interfaces. Do ponto de vista técnico, vislumbra-se o aprofundamento da integração entre os módulos, por exemplo, associar riscos a atividades críticas do cronograma e estimar impactos financeiros decorrentes de eventos de risco, e a incorporação de recursos complementares de identificação e análise de riscos, como apoio às técnicas SWOT e Delphi, discutidas na fundamentação teórica.

Referências

- Barbosa, C., Nascimento, C. A. D. d., Abdollahyan, F., and Pontes, R. M. (2019). *Gerenciamento de custos em projetos*. FGV Management, Rio de Janeiro, 6 edition.
- Barcaui, A. (2012). *Gerenciamento do tempo em projetos*. FGV, Rio de Janeiro.
- Bloss, R., Van Der Merwe, G., Ward, R., and Du Plessis, A. (2025). *Critical Path Management of the Mining Value Chain Using a Probabilistic Approach*. AusIMM, Australia.
- Dingsøyr, T. and Petit, M. (2021). *Managing Layers of Risk: Uncertainty in Large Development Programs Combining Agile Software Development and Traditional Project Management*. De Gruyter, Noruega.

- Project Management Institute (2017). *Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK)*. PMI, Newtown Square, PA, 6 edition.
- Salles Jr., C. A. C. (2010). *Gerenciamento de riscos em projetos*. FGV, Rio de Janeiro.
- Vierlboeck, M., Gövert, K., Trauer, J., and Lindemann, U. (2019). Budgeting for agile product development. In *Proceedings of the 22nd International Conference on Engineering Design (ICED19)*, Delft. Cambridge University Press.