UNIVERSIDADE
LUSÓFONA

SprintLab

Trabalho Final de Curso

Relatório Final

Diogo André Órfão Ferreira, a22202100, LIG

Eduardo Miguel Godinho Calhancas, a22204446, LIG

Orientador: Daniel Silveira

Coorientador: Rui Ribeiro

Entidade Externa: GMV

Departamento de Engenharia Informática da Universidade Lusófona

Centro Universitário de Lisboa

27/06/2025

Direitos de cópia

Agradecimentos

"Em primeiro lugar, gostaria de agradecer à minha namorada, Joana, pelo incansável apoio, paciência e incentivo ao longo de todo este percurso. A sua presença foi fundamental nos momentos mais exigentes e desafiantes desta licenciatura.

Aos meus pais, agradeço pelo apoio constante ao longo deste percurso académico e por estarem sempre presentes nas fases mais exigentes do trabalho.

Um agradecimento especial ao Professor Daniel Silveira, pelo excelente trabalho de orientação, pela disponibilidade constante e pelos conselhos valiosos que muito contribuíram para a concretização deste projeto.

Também ao Professor Rui Ribeiro, pela partilha generosa de experiência, pelas dicas práticas e pelas palavras motivadoras que marcaram positivamente o desenvolvimento deste trabalho.

Por fim, à minha amiga Tina, o meu sincero obrigado pela companhia e pelo acompanhamento durante o desenvolvimento do SprintLab. "– Diogo Ferreira.

"Quero expressar a minha profunda gratidão aos meus pais, pelo apoio incondicional não apenas ao longo do meu percurso académico, mas em todas as fases da minha vida. Foram a base da minha motivação, e o seu encorajamento constante foi essencial, especialmente durante o desenvolvimento deste projeto, o SprintLab.

Agradeço igualmente ao Professor Daniel Silveira, pelo apoio contínuo, pela disponibilidade e pelo contributo valioso que teve ao longo de todo o processo de desenvolvimento. A sua orientação prática e prestabilidade foram determinantes para a concretização deste trabalho.

Agradeço também ao Professor Rui Ribeiro, coordenador do meu curso, pelo acompanhamento ao longo de toda a minha formação académica e pelo apoio prestado em todas as etapas desta jornada.

Por fim, agradeço também à minha amiga Tina, pelo constante apoio durante a minha formação académica e desenvolvimento do SprintLab." – Eduardo Calhancas

Resumo

A motivação deste projeto está em responder à necessidade crescente de uma integração mais eficaz entre ferramentas de desenvolvimento e comunicação. Muitas equipas enfrentam dificuldades devido à falta de soluções que unam de forma eficiente plataformas distintas, o que afeta a colaboração e a eficiência no trabalho diário.

O problema identificado consiste na ausência de uma integração sólida entre uma plataforma *DevOps*, como o *Gitlab*, e uma plataforma de comunicação e colaboração, como o *Microsoft Teams*. Ambas são amplamente utilizadas em ambientes empresariais para o desenvolvimento e a coordenação de projetos. A falta desta integração compromete a gestão eficiente das tarefas, a visibilidade do progresso e a fluidez da comunicação, resultando em processos fragmentados e menos ágeis. Estes obstáculos podem afetar diretamente a produtividade e o sucesso dos projetos.

A solução proposta é o desenvolvimento de uma ferramenta que permita a unificação da gestão de projetos *Agile*, integrando o *Gitlab* com o *Microsoft Teams*. Através de funcionalidades como quadros de *sprints* e gráficos dinâmicos, como os *Gantt Charts*, o *plugin* oferece uma maneira de facilitar o acompanhamento do progresso dos projetos e otimizar a gestão, melhorando a comunicação entre as equipas. O objetivo é promover essa integração sem a necessidade de grandes alterações nos fluxos de trabalho já existentes, assegurando um processo mais eficiente e colaborativo.

Palavras-chave:

Integração, *Gitlab*, *Microsoft Teams*, Gestão de projetos, *Agile*, Quadros de *sprints*, Eficiência e Produtividade.

Abstract

The motivation for this project lies in addressing the growing need for more effective integration between development and communication tools. Many Teams face challenges due to the lack of solutions that seamlessly combine different platforms, which impacts daily collaboration and efficiency.

The identified issue is the lack of robust integration between Gitlab, a DevOps platform, and Microsoft Teams, a collaboration and communication tool. Although both are widely adopted in corporate environments, their disconnection hinders effective project management, reduces progress visibility, and disrupts communication flow. This fragmentation leads to less agile processes and can significantly impact productivity and project success.

The proposed solution is to develop a tool that unifies Agile project management by integrating enterprise Gitlab with Microsoft Teams. Through features like sprint Boards and dynamic Charts, such as Gantt Charts, this plugin aims to facilitate project progress tracking and optimize management, enhancing team communication. The goal is to provide this integration without requiring significant changes to existing workflows, ensuring a more efficient and collaborative process.

Keywords:

Integration, Gitlab, Microsoft Teams, Project management, Agile, Sprint Boards, Efficiency and Productivity.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas

API - Interface de Programação de Aplicações

HTML - Linguagem de Marcação de Hipertexto

GMV – Empresa Externa

ODS - Objetivos de Desenvolvimento Sustentável

MVP - Minimum Viable Product

SDK - Software Development Kit

REST - Representational State Transfer

OAuth2 - Open Authorization 2.0

SaaS - Software as a Service

MSAL - Microsoft Authentication Library

TLS - Transport Layer Security

SSL - Secure Sockets Layer

JSON - JavaScript Object Notation

Azure - Nome da plataforma de serviços *cloud* da *Microsoft*

GraphQL - Query Language for APIs

TFC - Trabalho Final de Curso

D3.js - Data-Driven Documents (biblioteca JavaScript para gráficos)

JIRA - Ferramenta de gestão de projetos ágeis

1 Introdução

1.1 Enquadramento

A gestão de projetos de desenvolvimento de software e outras soluções tecnológicas desempenha um papel crucial em empresas que operam no setor tecnológico e de engenharia, como a *GMV*. A *GMV* é uma organização multinacional com uma grande atuação em áreas como espaço, defesa, segurança, transporte, e tecnologias de informação, destacando-se por oferecer soluções inovadoras para desafios complexos (*GMV*, 2023). A empresa depende de ferramentas especializadas para garantir a eficiência e a produtividade das suas equipas. Entre as plataformas utilizadas estão o *Gitlab*, amplamente reconhecido como essencial para a gestão de código e controlo de versões (*Gitlab*, 2023), e o *Microsoft Teams*, uma ferramenta robusta para comunicação e colaboração em ambientes empresariais (*Microsoft*, 2023).

Apesar da eficácia de ambas as plataformas nos seus respetivos propósitos, a ausência de uma integração entre elas tem revelado desafios que impactam a fluidez do trabalho, a coordenação entre equipas e aumentam o custo operacional associado à gestão manual de tarefas e à duplicação de esforços. Esse contexto levou à identificação de uma necessidade específica observada pelo professor Daniel Silveira, integrante da GMV, que constatou as dificuldades em coordenar informações e atividades entre o *Gitlab* e o *Microsoft Teams*. O *Gitlab*, enquanto plataforma DevOps, e o *Microsoft Teams*, enquanto ferramenta de comunicação e colaboração, cumprem funções distintas e eficazes nos seus respetivos domínios. No entanto, a ausência de integração entre ambas tem revelado desafios que impactam a fluidez do trabalho, a coordenação entre equipas e aumentam o custo operacional associado à gestão manual de tarefas e à duplicação de esforços. Estes desafios resultam em processos fragmentados, duplicação de esforços e perda de tempo, evidenciando a urgência de uma solução que unifique o fluxo de trabalho.

A relevância deste projeto surge da necessidade de superar essas barreiras e criar um ambiente integrado que potencialize a colaboração, a eficiência e a centralização de dados críticos. Este trabalho pretende responder a uma problemática prática e recorrente, alinhada com as exigências do mercado atual, onde a r*api*dez e a colaboração são determinantes para o sucesso de projetos ágeis (*Agile Alliance*, 2023).

1.2 Motivação e Identificação do Problema

A motivação para este projeto surgiu do impacto significativo que a falta de integração entre ferramentas de desenvolvimento e comunicação tem na eficiência e produtividade das equipas. As metodologias ágeis destacam-se por promover flexibilidade, colaboração e entregas contínuas de valor (*Antlia*, 2023). No entanto, o uso de ferramentas fragmentadas obriga os profissionais a despenderem mais tempo em tarefas administrativas, como a inserção manual de dados, em detrimento de atividades que acrescentam valor direto ao projeto (*Sky One Solutions*, 2023).

A ausência de uma integração eficaz entre plataformas como o *Gitlab* e *o Microsoft Teams* compromete a comunicação fluida, resultando em atrasos e na perda de informações críticas. Uma integração bem-sucedida entre estas ferramentas permite que notificações sobre o progresso dos projetos sejam exibidas diretamente no ambiente de colaboração,

centralizando informações e agilizando respostas (*Gitlab*, 2024). Estudos indicam que a adoção de ferramentas ágeis bem integradas aumenta a transparência e facilita a monitorização do progresso em tempo real, fatores essenciais para o sucesso de projetos ágeis (Atlassian, 2024).

Assim, é evidente que a integração eficaz entre ferramentas de desenvolvimento e comunicação não só otimiza os processos de trabalho, mas também fortalece a colaboração entre as equipas. Este projeto pretende superar estas barreiras, promovendo um fluxo de trabalho mais coeso, eficiente e adaptado às necessidades organizacionais.

1.3 Objetivos

O projeto tem como objetivo central desenvolver uma solução que integre o *Gitlab* com o *Microsoft Teams*, criando um ambiente de trabalho mais eficiente para a gestão de projetos. Essa integração pretende superar os desafios atuais de fragmentação de processos e falta de comunicação eficaz, promovendo uma colaboração mais ágil e produtiva. Os objetivos específicos incluem:

- Centralizar a Gestão de Projetos: Proporcionar uma interface única que permita o acesso e o monitoramento de sprints e tarefas diretamente no *Microsoft Teams*, facilitando a coordenação e o acompanhamento das atividades.
- Automatizar Processos: Reduzir o esforço manual necessário para atualizar dados em ambas as plataformas, mitigando riscos de erro e aumentando a precisão e a consistência das informações.
- Otimizar a Comunicação entre Equipas: Promover uma troca de informações contínua e clara através de uma plataforma integrada, eliminando a necessidade de alternar entre diferentes ferramentas e melhorar a comunicação.
- Aumentar a Visibilidade e Transparência: Incorporar funcionalidades que gerem gráficos dinâmicos, como *Gantt Charts* e *Burndown Charts*, para visualizações em tempo real do progresso dos projetos, ajudando na tomada de decisões rápidas e bem-informadas.
- Assegurar uma Adaptação Rápida e Fácil: Desenvolver uma solução de utilização intuitiva, que possa ser facilmente adotada pelas equipas sem necessitar de uma formação extensa, garantindo uma transição suave e sem interrupções nos fluxos de trabalho existentes.

Estes objetivos têm como foco não só resolver as dificuldades presentes na *GMV*, mas também criar uma solução flexível que possa ser adaptada e aplicada em outras empresas que enfrentam desafios semelhantes, contribuindo para um ambiente de trabalho mais integrado e produtivo.

1.4 Estrutura do Documento

O presente relatório encontra-se organizado de forma a proporcionar uma visão clara e lógica sobre o desenvolvimento do projeto, cobrindo desde a análise de viabilidade até à proposta de solução, testes e validação. A estrutura do documento é a seguinte:

- Na Secção 1: Apresenta-se a introdução ao tema, com o enquadramento do projeto, a identificação do problema, a motivação para o seu desenvolvimento e os objetivos que se pretendem alcançar. Esta secção contextualiza a relevância da solução no âmbito organizacional e do mercado tecnológico.

- Na Secção 2: É abordada a pertinência e viabilidade da solução proposta. Inclui uma análise detalhada do problema, uma avaliação comparativa com soluções existentes (benchmarking) e a proposta de inovação, destacando as mais-valias da integração entre o *Gitlab* e o *Microsoft Teams*.

- Na Secção 3: Descreve-se a especificação e modelação do sistema, com destaque para a análise de requisitos funcionais e não funcionais. Apresenta ainda diagramas de modelação e protótipos de interface que ilustram o design e a navegação da solução proposta.

- Na Secção 4: Detalha-se a solução proposta, começando pela descrição das funcionalidades do *middleware* e do plugin para *Microsoft Teams*, bem como a sua arquitetura. São apresentadas as tecnologias e ferramentas utilizadas, os ambientes de teste e produção, a abrangência académica da solução e a descrição técnica dos componentes desenvolvidos.

- Na Secção 5: Foca-se nos testes e validação, descrevendo os métodos aplicados para garantir a qualidade e o funcionamento da solução. São incluídos os critérios de aceitação, cenários de teste e os resultados obtidos, demonstrando a adequação da solução às necessidades identificadas.

- Na Secção 6: Apresenta-se o método e planeamento do projeto, detalhando as etapas de desenvolvimento, o cronograma (*Gantt Chart*) e a abordagem ágil utilizada. São discutidas também as dificuldades enfrentadas e os ajustes realizados ao longo do projeto.

2 Pertinência e Viabilidade

2.1 Pertinência

A relevância deste projeto reside na resolução de um problema real enfrentado por diversas empresas tecnológicas, onde a ausência de integração entre ferramentas essenciais, como o *Gitlab* e o *Microsoft Teams*, resulta em processos fragmentados, duplicação de tarefas e perda de tempo. Esta falta de integração dificulta a gestão de projetos devido à ausência de sincronização e atualizações em tempo real, reduzindo a eficiência das equipas pela necessidade constante de alternar entre plataformas (Sky One Solutions, 2023).

A solução proposta visa integrar estas ferramentas, automatizando processos e centralizando dados, com o objetivo de melhorar a comunicação e a colaboração entre as equipas. A pertinência desta abordagem é corroborada por especialistas que identificam a necessidade de uma comunicação corporativa eficaz para otimizar processos internos e fortalecer a capacidade de adaptação das organizações num ambiente de negócios dinâmico (*FindUp*, 2024).

Além disso, uma equipa da GMV com quem foi possível dialogar reconheceu a pertinência deste projeto e o impacto positivo esperado da centralização de dados e da automação de tarefas. Entre os benefícios identificados destacam-se a melhoria da eficiência e produtividade, através da eliminação de redundâncias nos fluxos de trabalho, bem como o aumento da visibilidade e do controlo sobre os projetos, graças à disponibilização de dados atualizados em tempo real. Esta solução revela-se relevante não só para o contexto da GMV, mas também para outras empresas tecnológicas que enfrentam desafios semelhantes.

2.2 Viabilidade

A solução proposta utiliza *APIs* amplamente documentadas do *Gitlab* e do *Microsoft Teams* para criar uma integração funcional e escalável. Testes iniciais com um protótipo interativo irão confirmar a viabilidade técnica, demonstrando:

- Capacidade de sincronização de dados em tempo real, eliminando a necessidade de atualizações manuais (*Gitlab*, 2024).
- Automatização de tarefas, como atualizações automáticas no *Teams* baseadas em alterações no *Gitlab* (*Gitlab*, 2024).
- Visualização de gráficos dinâmicos, como diagramas de *Gantt*, diretamente no ambiente colaborativo (*Gitlab*, 2024).
- A compatibilidade com a infraestrutura existente da *GMV* e o uso de ferramentas já familiares para os colaboradores garantem uma implementação técnica eficiente e rápida (*GMV*, 2024).

Viabilidade Económica

O desenvolvimento da solução é viável, uma vez que aproveita infraestruturas e ferramentas já disponíveis na *GMV*. A proposta elimina a necessidade de aquisição de ferramentas externas ou licenças adicionais, reduzindo os custos operacionais ao longo do tempo. Além disso, a automação de processos e a centralização de informações resultam em ganhos de produtividade que superam o investimento inicial (*Lecom*, 2024).

Viabilidade Social e Ambiental

A viabilidade social da solução reflete-se na sua capacidade de promover um ambiente de trabalho mais colaborativo, transparente e eficiente, ao centralizar informações e automatizar tarefas que, de outro modo, exigiriam esforço manual e comunicação dispersa. Do ponto de vista ambiental, a redução da necessidade de processos redundantes e da troca excessiva de mensagens e documentos contribui indiretamente para um menor consumo de recursos digitais e energéticos, alinhando-se com práticas mais sustentáveis no contexto das tecnologias da informação.

Alinhamento com os Objetivos de Desenvolvimento Sustentável (ODS):

- ODS 8 (Trabalho Digno e Crescimento Económico): Contribui para um ambiente de trabalho mais eficiente e produtivo (*McKinsey*, 2024).
- ODS 9 (Inovação e Infraestrutura): Cria uma infraestrutura digital inovadora para gestão de projetos (*McKinse*y, 2024).
- ODS 12 (Consumo e Produção Sustentáveis): Promove a sustentabilidade tecnológica ao reduzir a dependência de ferramentas externas (*McKinsey*, 2024).

2.3 Análise Comparativa com Soluções Existentes

2.3.1 Soluções existentes

No mercado atual, existem diversas soluções que oferecem integrações entre o *Gitlab* e o *Microsoft Teams*, cada uma com características específicas. Neste trabalho, é dada especial atenção às soluções que apresentam maiores semelhanças com a proposta do SprintLab, tanto em termos de objetivos como de funcionalidades. O objetivo é compreender o panorama atual, identificar pontos fortes e limitações das abordagens existentes, e posicionar o SprintLab como uma alternativa inovadora e eficaz na gestão ágil de projetos integrados. Entre as soluções analisadas, destacam-se:

- ***Bots* de Notificação:** representam uma das formas mais simples e amplamente utilizadas de integração entre *Gitlab* e *Microsoft Teams*. Essas soluções são frequentemente disponibilizadas através de webhooks personalizados ou conectores pré-configurados, que têm como principal função enviar mensagens automáticas para canais do Teams sempre que ocorrem determinados eventos no *Gitlab*, como commits, merge requests, criação de issues ou alterações em pipelines.

  Apesar de serem úteis para manter as equipas informadas em tempo real, estas soluções são essencialmente unidirecionais. Ou seja, a comunicação flui do *Gitlab* para o Teams, mas não há possibilidade de interação com os conteúdos notificados. Os utilizadores não podem, por exemplo, atribuir uma issue, mover tarefas em boards, ou atualizar estados de tickets diretamente a partir do Teams. Além disso, o volume elevado de notificações pode, por vezes, tornar-se intrusivo e redundante, reduzindo a sua eficácia na gestão real de projetos.

- ***Unito* e *ZAPIer***: são plataformas de integração e automação que permitem a ligação entre múltiplas aplicações, incluindo *Gitlab* e *Microsoft Teams*, sem necessidade de programação. Estas ferramentas funcionam com base em "triggers" (gatilhos) e "actions" (ações), que são definidos pelo utilizador para executar rotinas automáticas, como criar uma tarefa no Teams sempre que uma issue é aberta no *Gitlab*.
  Estas soluções destacam-se pela sua flexibilidade e facilidade de configuração, sendo úteis em contextos onde se pretendem automatizar tarefas simples, reduzir trabalho manual e garantir a consistência entre sistemas. No entanto, apresentam limitações significativas quando se trata de gestão ágil de projetos. Não oferecem suporte nativo para visualização de boards, sprints ou métricas visuais dinâmicas no *Microsoft Teams*, nem integram com profundidade os fluxos de trabalho de DevOps. Além disso, a personalização de workflows mais complexos requer planos pagos e pode tornar-se financeiramente inviável em equipas maiores.
- ***Jira***: é uma das soluções mais robustas e completas do mercado no que diz respeito à gestão de projetos e integrações com plataformas externas. A sua integração com o *Microsoft Teams* permite funcionalidades como criação, edição e visualização de tickets diretamente a partir de canais, assim como a utilização de comandos em mensagens para interagir com o projeto. Oferece também dashboards visuais avançados, acompanhamento de sprints e relatórios detalhados, o que a torna uma escolha popular em ambientes empresariais mais exigentes.
  No entanto, esta robustez tem custos associados. A curva de aprendizagem do Jira é acentuada, exigindo formação específica para os utilizadores menos experientes. A sua interface é, por vezes, considerada densa e complexa, e a sua integração com o *Gitlab* não é nativa nem profunda, o que implica a necessidade de ferramentas ou plugins adicionais. Para organizações que já utilizam *Gitlab* como plataforma DevOps principal, a adoção do Jira pode introduzir redundância e fricção entre ferramentas.

2.3.2 Análise de benchmarking

Características	*SprintLab*	*Bots*	*Unito/ZAPIer*	*Jira*
Integração em tempo real	✓	✓	✓	✓
Gestão de *sprints* no *Teams*	✓	✓	✗	✓
Gerar gráficos dinâmicos	✓	✗	✗	✓
Custo acessível	✓	✗	✗	✗
Interface intuitiva	✓	✓	✓	✗
Custo acessível	✓	✓	✓	✗
Interface intuitiva	✓	✗	✓	✓
Popularidade no mercado	✗	✓	✓	✓

Tabela 1 - Análise de benchmarking

A nossa proposta, *SprintLab*, destaca-se pela combinação de funcionalidades robustas, simplicidade de uso e custos operacionais reduzidos, oferecendo uma solução integrada que as alternativas concorrentes não conseguem proporcionar em conjunto.

2.4 Proposta de inovação e mais-valias

A solução proposta distingue-se pela inovação ao oferecer uma integração direta entre o *Gitlab* e o *Microsoft Teams*, permitindo a gestão centralizada de projetos. Esta integração proporciona uma automatização de processos e a sincronização de dados em tempo real, eliminando a fragmentação de informações e reduzindo significativamente a ocorrência de erros. Além disso, destaca-se pela possibilidade de criar visualizações avançadas e dinâmicas, como gráficos de *Gantt*, diretamente no ambiente do *Microsoft Teams*, facilitando o acompanhamento e a tomada de decisões.

Entre as mais-valias desta solução, estão a melhoria da eficiência e produtividade, ao reduzir o tempo e esforço em tarefas repetitivas, e a facilidade de adoção, uma vez que é compatível com ferramentas já conhecidas e apresenta uma interface intuitiva. A solução também se destaca pela sua sustentabilidade e escalabilidade, graças a uma arquitetura flexível, que irá permitir futuras expansões ou adaptações às necessidades das equipas e organizações. O caráter inovador da proposta motivou ainda a submissão de um abstract à conferência do Project Management Institute (PMI), encontrando-se atualmente em fase de avaliação.

2.5 Identificação de oportunidade de negócio

Além de resolver um problema específico da *GMV*, a solução proposta apresenta um forte potencial para exploração comercial, conforme evidenciado por práticas de mercado e referências reais:

1. Produto *SaaS*: Disponibilizar o plugin como um serviço para empresas tecnológicas está alinhado com a tendência crescente de adoção de modelos *Software as a Service (SaaS)*, que oferecem escalabilidade e redução de custos operacionais (*Lyncas*, 2024).
2. Consultoria Personalizada: Adaptar a solução para outros setores ou fluxos de trabalho permite atender às necessidades específicas de diferentes indústrias, aumentando a relevância e o alcance do produto.
3. Parcerias Estratégicas: Colaborações com empresas como a *Microsoft* ou o *Gitlab* podem ampliar significativamente o alcance da solução. A *Microsoft*, por exemplo, possui um ecossistema de parceiros diversificado que impulsiona a inovação e a resiliência dos negócios (*Microsoft* - Ecossistemas, 2024).
4. Monetização de Funcionalidades Premium: Oferecer uma versão básica gratuita com funcionalidades avançadas pagas é uma estratégia comum no mercado de software, permitindo a atração de uma base ampla de utilizadores e a geração de receita adicional através de subscrições premium (*Lyncas*, 2024).

Implementando estas estratégias, o projeto tem o potencial de se estabelecer como um recurso inovador no mercado de ferramentas de colaboração e gestão de projetos.

3 Especificação e Modelação

3.1 Análise de Requisitos

3.1.1 Enumeração de Requisitos

Requisitos Funcionais

Requisitos	Prioridade	MoSCoW	Cumprimento	Critérios de Aceitação
RF01: O *middleware* deve estar alojado num servidor para funcionamento contínuo.	Média	Must	Implementado	O *middleware* está acessível via um endereço IP ou domínio válido. *Logs* do servidor confirmam que o *middleware* está operacional.
RF02: O servidor onde o *middleware* está alojado deve ser configurado para operação contínua.	Médio	Must	Implementado	O servidor reinicia automaticamente após falhas. *Logs* de monitoramento estão disponíveis para administradores.
RF03: O *middleware* deve estar configurado com o *webhook* do *Gitlab*.	Alta	Should	Implementado	O *webhook* do *Gitlab* está configurado corretamente para enviar eventos ao *middleware*. *Logs* do *middleware* registam a receção de eventos do *Gitlab*.
RF04: O middleware deve expor um endpoint para receber notificações da API do *Gitlab*.	Alta	Must	Implementado	Cada evento disparado no *Gitlab* está a ser registado pelo *middleware*. Os *Logs* confirmam a receção e

				o processamento de eventos.
RF05: O *middleware* deve processar notificações de eventos de push no *Gitlab* e armazenar as informações.	Baixa	Must	Implementado	Informações de eventos de push estão a ser registadas em *logs*. O relatório de eventos de push está acessível para administradores.
RF06: O *middleware* deve processar notificações relativas à criação de *issue*s no *Gitlab*.	Alta	Should	Implementado	Cada criação de *issue* gera uma entrada correspondente no *log*. Tarefas equivalentes estão a ser criadas no *Kanban* do *Teams*.
RF07: O *middleware* deve processar notificações de atualização de detalhes das *issue*s no *Gitlab*.	Alta	Should	Implementado	Atualizações feitas nas *issue*s refletem-se no *Kanban* do *Teams* em até 1 minuto. *Logs* registam a sincronização das atualizações.
RF08: O *middleware* deve processar notificações de encerramento de *issue*s no *Gitlab*.	Alta	Could	Implementado	Encerramentos de *issue*s marcam automaticamente as tarefas no *Teams* como concluídas. *Logs* registam estas alterações.
RF09: O *middleware* deve processar notificações relacionadas à reabertura de *issue*s no *Gitlab*.	Alta	Could	Implementado	A reabertura de uma *issue* reativa a tarefa correspondente no *Teams*. *Logs* confirmam a sincronização.

RF10: O *middleware* deve identificar alterações nos rótulos (labels) das *issue*s no *Gitlab* e processá-las adequadamente.	Alta	Should	Implementado	Alterações nos *labels* refletem-se no *Teams* em até 1 minuto. *Logs* detalham as alterações processadas.
RF11: O *middleware* deve receber notificações de criação de *merge requests* no *Gitlab*.	Médio	Must	Implementado	Cada criação de *merge* request está registada nos *Logs* do *middleware*. Tarefas equivalentes são criadas no *Teams*.
RF12: O *middleware* deve processar notificações de alterações nos títulos dos *merge requests* no *Gitlab*.	Médio	Should	Implementado	Alterações nos títulos dos *merge requests* refletem-se no *Teams*. *Logs* devem registar as alterações sincronizadas.
RF13: O *middleware* deve processar notificações de alterações nas descrições dos *merge requests* no *Gitlab*.	Médio	Should	Implementado	Alterações nas descrições dos *merge requests* refletem-se no *Teams*. *Logs* registam essas alterações.
RF14: O *middleware* deve registar notificações de aprovação de *merge requests* no *Gitlab*.	Baixo	Could	Implementado	Cada aprovação é registada em *Logs*. Tarefas relacionadas no *Teams* são atualizadas.
RF15: O *middleware* deve registar notificações de rejeição de *merge requests* no *Gitlab*.	Baixo	Could	Implementado	Cada rejeição é registada em *Logs*. Tarefas relacionadas no

				Teams são atualizadas.
RF16: O *middleware* deve processar notificações de alterações no estado dos *merge requests* no *Gitlab*.	Baixo	Should	Implementado	Alterações de estado são registadas em *Logs* e refletidas no *Teams*. Estados são atualizados em até 5 minutos.
RF17: O *middleware* deve capturar eventos de comentários adicionados a *merge requests* no *Gitlab*.	Baixo	Could	Implementado	Cada comentário é registado em *Logs*. Comentários estão disponíveis para consulta.
RF18: O *middleware* deve transformar dados das *issue*s do *Gitlab* em tarefas visíveis no quadro *Kanban* do *Teams*.	Alta	Must	Implementado	Cada *issue* é representada como uma tarefa no *Kanban*. *Logs* registam a criação das tarefas.
RF19: O *middleware* deve transformar listas de verificação (*checklists*) dentro das *issue*s do *Gitlab* em subtarefas no quadro *Kanban* do *Teams*.	Alta	Should	Implementado	*Checklists* são convertidas em subtarefas no *Teams*. *Logs* confirmam a sincronização das subtarefas.
RF20: O *middleware* deve sincronizar alterações no quadro *Kanban* do *Teams* com os rótulos ou estados das *issue*s no *Gitlab*.	Alta	Must	Implementado	Alterações no *Kanban* são refletidas no *Gitlab* em até 1 minuto. *Logs* registam a sincronização.
RF21: O *middleware* deve	Alta	Should	Implementado	Eliminações de tarefas no *Teams*

capturar eliminações de tarefas no quadro *Kanban* e refletir as alterações nas *issue*s correspondentes no *Gitlab*.				removem as *issue*s correspondentes no *Gitlab*. *Logs* registam as alterações.
RF22: O *middleware* deve sincronizar alterações feitas no *Gitlab* com o quadro *Kanban* do *Teams*.	Alta	Must	Implementado	Alterações nas *issue*s do *Gitlab* são refletidas no *Kanban* do *Teams* em até 1 minuto. *Logs* devem registar as sincronizações.
RF23: O plugin do *Teams* deve exibir um quadro *Kanban* com a capacidade de filtrar por rótulos.	Alta	Must	Implementado	Os usuários podem visualizar e filtrar o quadro *Kanban* no *Teams* por labels. *Logs* registam os filtros aplicados.
RF24: O plugin do *Teams* deve exibir um gráfico de *Gantt* de projetos completos.	Alta	Should	Implementado	Gráficos de *Gantt* são gerados com base em projetos completos do *Gitlab*. *Logs* registam a criação de gráficos.
RF25: O plugin do *Teams* deve exibir um gráfico de *Gantt* filtrado por rótulos específicos.	Alta	Could	Implementado	Gráficos de *Gantt* devem são gerados com base em labels específicos. *Logs* registam a criação de gráficos filtrados.

RF26: O plugin do *Teams* deve conectar-se ao *middleware* para obter gráficos atualizados.	Alta	Must	Implementado	*Logs* registam solicitações do plugin para gráficos atualizados. Os gráficos exibidos estão atualizados com base nos dados mais recentes.
RF27: O plugin do *Teams* deve conectar-se ao *middleware* para obter tarefas atualizadas.	Alta	Must	Implementado	*Logs* registam as solicitações do plugin para tarefas atualizadas O quadro Kanban exibe as tarefas mais recentes.
RF32: O *middleware* deve incluir ligações diretas para *issue*s no quadro *Kanban* do *Teams*.	Alta	Should	Implementado	Cada tarefa no *Kanban* deve conter links clicáveis para as *issue*s correspondentes no *Gitlab*. *Logs* devem registar a criação dos links.
RF33: O *middleware* deve incluir ligações diretas para *merge requests* no quadro *Kanban* do *Teams*.	Média	Should	Implementado	Cada tarefa no *Kanban* contem links clicáveis para os *merge requests* correspondentes no *Gitlab*. *Logs* registam a criação dos links.

RF34: O *middleware* deve suportar múltiplos projetos *Gitlab* sincronizados com diferentes equipas no *Teams*.	Alta	Must	Implementado	Os usuários conseguem gerir múltiplos projetos *Gitlab* no *Kanban* do *Teams*. *Logs* registam sincronizações de múltiplos projetos.

Tabela 2 - Requisitos Funcionais

Requisitos Não-Funcionais

Requisito	Prioridade	**MoSCoW**	Cumprimento	Critérios de Aceitação
RNF01: O *middleware* deve garantir comunicação segura usando HTTPS para todas as interações.	Média	Must	Implementado	Todas as comunicações entre o *middleware*, *Gitlab* e *Teams* são criptografadas.
RNF02: O *middleware* deve ser escalável para suportar o aumento do número de projetos ou eventos.	Média	Must	Implementado	O *middleware* consegue processar 100% de aumento no tráfego sem degradação do desempenho. Deve suportar no mínimo 1.000 eventos simultâneos sem erros críticos.
RNF03: O *middleware* deve ser compatível com JavaScript.	Alta	Should	Implementado	Scripts do *middleware* devem executar sem erros em versões JavaScript. Dependências instaladas devem ser compatíveis com as versões de JavaScript suportadas.

RNF04: O *middleware* deve autenticar-se de forma segura na *API* do *Gitlab* utilizando *token*s.	Alta	Must	Implementado	*Token*s de autenticação são validados antes de cada solicitação. *Token*s expirados são renovados automaticamente sem intervenção manual.
RNF05: O *middleware* deve autenticar-se de forma segura na *API* do *Microsoft Teams* utilizando *token*s.	Alta	Must	Implementado	*Token*s válidos são gerados e usados para cada requisição. Erros de autenticação são tratados e registados.
RNF06: O *middleware* deve registar *Logs* detalhados de eventos processados e erros encontrados.	Baixa	Must	Implementado	*Logs* incluem carimbo de data/hora, tipo de evento e status. *Logs* de erro categorizam as falhas por nível de severidade (e.g., aviso, crítico).
RNF07: O *middleware* deve operar com alta disponibilidade, reiniciando automaticamente em caso de falhas críticas.	Média	Must	Implementado	O *middleware* está operacional 99,9% do tempo. Reinicializações automáticas ocorrem em menos de 1 minuto após falhas.

RNF08: O middleware deve suportar múltiplos projetos GitLab simultaneamente através da API.	Alta	Should	Implementado	O middleware deve suportar simultaneamente a comunicação com no mínimo 10 projetos GitLab distintos via API, sem interferência entre os dados.
RNF09: O *middleware* deve suportar a integração com múltiplos canais e equipas no *Microsoft Teams*.	Alta	Should	Implementado	No mínimo 5 canais ou equipas devem estar sincronizadas simultaneamente. Cada equipa deve visualizar apenas as tarefas relacionadas com o seu projeto.
RNF10: O *middleware* deve manter desempenho consistente mesmo sob alta carga de eventos.	Média	Must	Implementado	A latência máxima sob alta carga não excede 1 segundo. O *middleware* processa até 1.000 eventos por hora sem degradação.
RNF11: O *middleware* deve evitar dependências que limitem a sua compatibilidade com versões futuras do JavaScript.	Baixa	Should	Implementado	Dependências são atualizáveis sem causar conflitos com o *middleware*. Ferramentas de análise estática devem confirmar compatibilidade com JavaScript
RNF12: O *middleware* deve registar interrupções na comunicação com o *Gitlab* e tentar novamente os eventos que falharem.	Média	Must	Implementado	Tentativas de reconexão ocorrem em até 3 ciclos automáticos. Após três falhas, o *middleware* gera alertas para administradores.

RNF13: O *middleware* deve registar interrupções na comunicação com o *Microsoft Teams* e tentar novamente os eventos que falharem.	Média	Must	Implementado	Reconexões são tentadas automaticamente em até 3 ciclos. Mensagens de erro são exibidas em caso de falha persistente.
RNF14: O *middleware* deve ser configurável para operar em múltiplos ambientes (desenvolvimento, homologação e produção).	Baixa	Should	Implementado	Arquivos de configuração independentes estão disponíveis para cada ambiente. Trocas de ambiente são possíveis sem alterações no código-fonte.
RNF15: O plugin do *Teams* deve garantir uma experiência responsiva, mesmo com múltiplos gráficos ou tarefas.	Alta	Must	Implementado	O tempo de carregamento inicial não excede os 2 segundos. Atualizações no plugin são aplicadas em até 5 segundos após a alteração.
RNF16: O *middleware* deve suportar monitorização externa com Fly.io.	Baixa	Should	Implementado	Métricas como tempo de resposta e taxa de sucesso são exportadas para o Fly.io. Alertas configuráveis são enviados em caso de falhas críticas.

Tabela 3 - Requisitos Não-Funcionais

Requisitos Técnicos

Requisito	Prioridade	MoSCoW	Cumprimento	Critérios de Aceitação
RT01: O *middleware* deve ser construído utilizando o *framework Express.js* para expor *APIs REST* próprias.	Alta	Must	Implementado	Todas as rotas do middleware estão devidamente organizadas e a responder de forma eficiente às solicitações REST. O middleware retorna respostas REST em até 500 ms.
RT02: O *middleware* deve ser alojado num servidor configurado para alta disponibilidade e balanceamento de carga.	Alta	Must	Implementado	O *middleware* suporta o balanceamento de carga entre pelo menos 2 servidores. O servidor está disponível 99,9% do tempo.
RT03: O *middleware* deve utilizar a biblioteca *axios* para interagir com a *API* do *Gitlab*.	Alta	Should	Implementado	As chamadas à API do *Gitlab* usam HTTP com *token* de acesso e garantir fiabilidade e integração contínua com os dados do repositório.
RT04: O *middleware* deve usar a biblioteca MSAL (*Microsoft* Authentication Library) para autenticação *OAuth2* com a *Microsoft Graph API*.	Alta	Must	Implementado	*Token*s de autenticação são geridos automaticamente pela MSAL. Erros na renovação de *token*s devem ser tratados corretamente.

RT05: O *middleware* deve suportar chamadas REST da *API* do *Gitlab* para aceder a *issue*s e *merge requests*.	Alta	Must	Implementado	Chamadas REST retornam dados completos em formato JSON válido. A latência máxima para uma chamada REST é de 1 segundo.
RT06: O *middleware* deve suportar chamadas *Graph*QL da *API* do *Gitlab* para aceder a *issue*s e *merge requests*.	Baixa	Should	Implementado	Consultas *Graph*QL retornam informações precisas e completas. Consultas inválidas são rejeitadas com mensagens de erro claras.
RT07: O middleware deve permitir a configuração e gestão dos tokens e identificadores de projeto GitLab através de um painel administrativo.	Média	Should	Implementado	O painel administrativo suporta a criação, atualização e exclusão de configurações de projetos GitLab (token e project ID).
RT08: O *middleware* deve formatar dados de *issue*s do *Gitlab* para serem compatíveis com quadros *Kanban* no *Teams*.	Alta	Must	Implementado	Campos como título, assignee e labels estão mapeados corretamente para o *Kanban*. Dados formatados são atualizados em até 1 minuto após as alterações no *Gitlab*.
RT09: O *middleware* deve formatar dados de rótulos do *Gitlab* para serem compatíveis com gráficos de *Gantt* no *Teams*.	Alta	Should	Implementado	Labels são convertidos para categorias utilizáveis nos gráficos de *Gantt*. Apenas labels ativos devem ser considerados.

RT10: O plugin do *Teams* deve utilizar o *SDK* do *Microsoft Teams* para integração total com a interface do *Teams*.	Alta	Must	Implementado	O plugin exibe corretamente todas as tarefas e gráficos através do *SDK*. Atualizações realizadas no *Teams* são refletidas no *Gitlab* em até 1 minuto.
RT11: O plugin do *Teams* deve consumir os endpoints do *middleware* para exibir tarefas atualizadas.	Alta	Must	Implementado	O plugin vai buscar tarefas atualizadas pelo menos a cada 5 minutos. Dados inconsistentes são tratados e corrigidos automaticamente.
RT12: O plugin do *Teams* deve consumir os *endpoints* do *middleware* para exibir gráficos atualizados.	Alta	Must	Implementado	Gráficos exibidos no *Teams* refletem os dados mais recentes do *Gitlab*. Dados inválidos ou incompletos geram alertas ao usuário.
RT13: O *middleware* deve ser configurado para alojar *Logs* e métricas acessíveis para administradores.	Baixa	Must	Implementado	*Logs* e métricas são acessíveis através de um painel administrativo. Apenas usuários autenticados têm acesso a essas informações.
RT14: O *middleware* deve ser executado com o servidor Fly.io configurado para alta performance em produção.	Média	Should	Implementado	O *middleware* utiliza pelo menos 2 instâncias (máquinas) e suporta o escalonamento automático. O tempo médio de resposta é

				inferior a 500ms sob carga .
RT15: O *middleware* deve suportar autenticação personalizada para utilizadores no *Gitlab* e no *Teams*.	Média	Must	Implementado	Apenas usuários autenticados têm acesso às funcionalidades do *middleware*. Permissões personalizadas são verificadas antes de cada operação.

Tabela 4 - Requisitos Técnicos

3.1.2 Descrição detalhada dos requisitos principais

RF04: O middleware deve expor um endpoint para receber notificações da API do *Gitlab*.

- Descrição: O middleware deve aceder aos dados do *Gitlab* através da sua API e processar eventos relevantes dos projetos, como a criação ou atualização de issues, movimentação em quadros Kanban, alterações em milestones, etiquetas e membros da equipa. Estes dados são utilizados para manter sincronizadas as ferramentas de apoio à gestão de equipas no *Microsoft Teams*.
- Dependências:
  - Configuração válida dos projetos *Gitlab* no sistema (*token* de acesso e ID do projeto).
  - Conectividade com a API pública do *Gitlab*.
  - Disponibilidade contínua do servidor onde o middleware está alojado (ver RF01, RF02).
- Objetivos:
  - Garantir a sincronização em tempo real dos eventos ocorridos no *Gitlab* com o *Microsoft Teams*.
  - Assegurar que todas as alterações relevantes no *Gitlab* sejam refletidas nas ferramentas de gestão das equipas.
- Critérios de Aceitação:
  - Os dados dos projetos são corretamente obtidos e processados a partir da API do *Gitlab*.
  - As interfaces do Teams (como o quadro Kanban) exibem a informação atualizada.
  - O middleware mantém registos detalhados de todas as interações com a API.
  - Em caso de erro na comunicação com o *Gitlab*, o sistema deve gerar alertas automáticos para os administradores (ver RNF06).
- Processos de Negócio:
  - O middleware executa chamadas à API do *Gitlab* para obter o estado atual do projeto.
  - Os dados recolhidos são transformados e disponibilizados ao *Microsoft Teams*, garantindo consistência com a realidade do repositório.
  - Este requisito está associado ao Caso de Uso 1.

RF18: O *middleware* deve transformar dados das *issue*s do *Gitlab* em tarefas visíveis

no quadro *Kanban* do *Teams*.

- Descrição: As *issue*s criadas no *Gitlab* devem ser convertidas em tarefas correspondentes no quadro *Kanban* do *Microsoft Teams*. Esta transformação deve preservar informações essenciais como título, descrição, atribuídos, labels e estado.
- Dependências:
  - Receção correta das notificações de criação de *issue*s (RF04).
  - Capacidade do *middleware* para acessar e manipular dados das *issue*s (RT08).
- Objetivos:
  - Facilitar a gestão de tarefas pelas equipas diretamente no *Teams*.
  - Centralizar informações para evitar a alternância constante entre ferramentas.
- Critérios de Aceitação:
  - Cada *issue* criada no *Gitlab* é refletida como uma tarefa no quadro *Kanban* do *Teams* em até 5 minutos.
  - Os campos da tarefa no *Teams* correspondem aos campos da *issue* no *Gitlab*.
  - *Logs* confirmam a transformação e criação das tarefas sem erros.
- Processos de Negócio:
  - O *middleware* recebe uma notificação de nova *issue*.
  - Transforma os dados da *issue* para o formato compatível com o *Kanban* do *Teams* (RT08).
  - Cria a tarefa correspondente no *Teams*.
  - Ligação ao Caso de Uso 2: Visualização e Gestão de Tarefas no *Teams*.

RF22: O *middleware* deve sincronizar alterações feitas no *Gitlab* com o quadro *Kanban* do *Teams*.

- Descrição: Qualquer alteração realizada nas *issue*s do *Gitlab*, como mudanças de estado, atualizações de descrição ou reassociação de labels, deve ser refletida no quadro *Kanban* do *Teams*.
- Dependências:
  - Funcionamento adequado da API para captar eventos de atualização (RF04).
  - Capacidade de transformação e mapeamento de dados atualizados (RT08).
- Objetivos:
  - Assegurar que o quadro *Kanban* do *Teams* esteja sempre atualizado, refletindo o estado real das tarefas.
  - Evitar discrepâncias entre o *Gitlab* e o *Teams* que possam causar confusão nas equipas.
- Critérios de Aceitação:
  - Alterações no *Gitlab* são refletidas no *Teams* em até 5 minutos.
  - Não existem inconsistências entre as informações apresentadas nas duas plataformas.
  - *Logs* registam todas as sincronizações efetuadas, incluindo eventuais erros.
- Processos de Negócio:
  - Alteração numa *issue* no *Gitlab* gera uma notificação via webhook.
  - *Middleware* processa a notificação e atualiza a tarefa correspondente no *Teams*.
  - Relacionado com o Caso de Uso 3.

RF27: O plugin do *Teams* deve exibir um quadro *Kanban* com a capacidade de filtrar por rótulos.

- Descrição: O plugin desenvolvido para o *Microsoft Teams* deve permitir aos utilizadores visualizar o quadro *Kanban* e aplicar filtros baseados em labels (rótulos), facilitando a organização e priorização de tarefas.
- Dependências:
  - Disponibilidade dos dados atualizados das tarefas com os respectivos labels (RF18, RF22).
  - Funcionalidades de filtragem implementadas no plugin do *Teams*.
- Objetivos:
  - Melhorar a usabilidade do quadro *Kanban*, permitindo que os utilizadores foquem em tarefas relevantes.
  - Auxiliar na gestão de tarefas por categoria, prioridade ou equipa.
- Critérios de Aceitação:
  - Utilizadores conseguem aplicar filtros por labels e visualizar apenas as tarefas correspondentes.
  - O sistema responde rapidamente às solicitações de filtragem (RNF15).
  - A interface permanece intuitiva e sem erros durante a utilização dos filtros.
- Processos de Negócio:
  - O utilizador seleciona um ou mais labels para filtrar.
  - O plugin solicita ao *middleware* as tarefas correspondentes aos labels selecionados.
  - O quadro *Kanban* atualiza-se com as tarefas filtradas.
  - Associado ao Caso de Uso 4: Filtragem de Tarefas no Quadro *Kanban*.

RNF08: O middleware deve suportar múltiplos projetos GitLab simultaneamente através da API.

- Descrição: O *middleware* deve ser capaz de gerir notificações provenientes de vários projetos no *Gitlab* ao mesmo tempo, garantindo que os dados são processados corretamente e encaminhados para as equipas correspondentes no *Teams*.
- Dependências:
  - Capacidade de distinguir e gerir eventos de diferentes projetos.
  - Configuração adequada da API em cada projeto *Gitlab*.
- Objetivos:
  - Escalabilidade da solução para suportar múltiplos projetos e equipas.
  - Evitar conflitos ou mistura de dados entre projetos distintos.
- Critérios de Aceitação:
  - *Middleware* processa eventos de pelo menos 10 projetos *Gitlab* simultaneamente sem perda de desempenho.
  - *Logs* demonstram o processamento correto e separado dos eventos de cada projeto.
  - Não ocorrem interferências ou cruzamento de informações entre projetos.
- Processos de Negócio:
  - Eventos de diferentes projetos são recebidos pelo *middleware*.
  - *Middleware* identifica o projeto de origem e processa os eventos de forma isolada.
  - Eventos são encaminhados para os canais ou equipas correspondentes no

Teams.
    - Relacionado com o Caso de Uso 5: Gestão de Múltiplos Projetos.

RNF09: O *middleware* deve suportar a integração com múltiplos canais e equipas no *Microsoft Teams*.

- Descrição: A solução deve permitir que diferentes equipas ou departamentos utilizem o sistema simultaneamente, com integração personalizada para cada canal ou equipa no *Teams*.
- Dependências:
    - Configurações de mapeamento entre projetos *Gitlab* e canais/equipas no *Teams*.
    - Capacidade do *middleware* para gerir múltiplas conexões e sessões.
- Objetivos:
    - Facilitar a adoção da solução em toda a organização.
    - Garantir que cada equipa tem acesso apenas às informações relevantes ao seu trabalho.
- Critérios de Aceitação:
    - *Middleware* integra-se com pelo menos 5 canais ou equipas no *Teams* sem degradação de performance.
    - Dados de uma equipa não são acessíveis por outras equipas sem permissão.
    - *Logs* mostram claramente as interações e operações realizadas para cada equipa.
- Processos de Negócio:
    - Cada equipa configura a sua integração específica através de credenciais ou *token*s.
    - *Middleware* gere as comunicações de forma segregada e segura.
    - Associado ao Caso de Uso 6: Integração com Múltiplas Equipas no *Teams*.

RT08: O *middleware* deve formatar dados de *issue*s do *Gitlab* para serem compatíveis com quadros *Kanban* no *Teams*.

- Descrição: O *middleware* deve converter os dados das *issue*s do *Gitlab* num formato adequado para serem exibidos corretamente no quadro *Kanban* do *Teams*, assegurando a integridade e consistência das informações.
- Dependências:
    - Acesso às *API*s do *Gitlab* e do *Teams* para obtenção e envio de dados.
    - Conhecimento dos formatos e estruturas de dados esperados pelo *Teams*.
- Objetivos:
    - Assegurar uma integração perfeita entre as duas plataformas.
    - Proporcionar aos utilizadores uma experiência unificada e sem discrepâncias nos dados.
- Critérios de Aceitação:
    - Todas as informações relevantes (título, descrição, atribuído, labels, estado) são mapeadas corretamente.
    - Não ocorrem erros de formatação ou perda de dados durante a transformação.
    - *Logs* registam as operações de formatação e identificam possíveis falhas.
- Processos de Negócio:

- *Middleware* recebe dados das *issue*s do *Gitlab*.
- Processa e transforma os dados conforme os requisitos do *Teams*.
- Envia os dados formatados para o *Teams* para exibição no quadro *Kanban*.
- Ligação com os Casos de Uso 2 e 3.

RT09: O *middleware* deve formatar dados de rótulos do *Gitlab* para serem compatíveis com gráficos de *Gantt* no *Teams*.

- Descrição: Os labels utilizados nas *issue*s do *Gitlab* devem ser transformados em categorias ou elementos que permitam a geração de gráficos de *Gantt* no *Teams*, facilitando a visualização do cronograma do projeto.
- Dependências:
  - Acesso e extração dos labels das *issue*s no *Gitlab*.
  - Compreensão dos requisitos de dados para a geração de gráficos de *Gantt* no *Teams*.
- Objetivos:
  - Fornecer às equipas ferramentas visuais para o acompanhamento do progresso do projeto.
  - Melhorar o planeamento e gestão de prazos.
- Critérios de Aceitação:
  - Os gráficos de *Gantt* refletem corretamente as tarefas e os seus prazos com base nos labels.
  - Atualizações nos labels ou prazos no *Gitlab* são refletidas nos gráficos no *Teams* em até 5 minutos.
  - Ausência de erros na geração e atualização dos gráficos.
- Processos de Negócio:
  - *Middleware* extrai informações dos labels e prazos das *issue*s.
  - Transforma os dados para o formato necessário à criação dos gráficos.
  - Envia as informações para o *Teams*, onde os gráficos são gerados e atualizados.
  - Relacionado com o Caso de Uso 7: Visualização de Gráficos de *Gantt* no *Teams*.

3.1.3 Casos de Uso/*User Stories*

Caso de Uso 1: Sincronização de Eventos do *Gitlab* para o *Teams*

- Ator Principal: *Gitlab* (Automação via Webhook)
- Descrição: O *middleware* recebe notificações da API configurada no *Gitlab* e processa os eventos, enviando atualizações ao *Microsoft Teams*.
- Requisitos Relacionados: RF04, RNF08
- Fluxo Principal:
  1. O utilizador realiza uma ação no *Gitlab* (e.g., cria uma *issue* ou faz um push).
  2. O webhook do *Gitlab* envia uma notificação ao *middleware*.
  3. O *middleware* processa o evento recebido.
  4. O *middleware* transforma os dados para um formato compatível com o *Teams*.
  5. As alterações são refletidas no *Kanban Board* ou em notificações no *Teams*.
- Fluxo Alternativo:
  1. O webhook não é configurado corretamente ou o *middleware* está offline.
     - O sistema gera *Logs* de erro e envia alertas para o administrador (RNF12).

Caso de Uso 2: Visualização e Gestão de Tarefas no *Teams*

- Ator Principal: Utilizador Final
- Descrição: O utilizador visualiza e gere tarefas no quadro *Kanban* do *Teams*, filtrando por labels ou categorias relevantes.
- Requisitos Relacionados: RF18, RF27, RT08
- Fluxo Principal:
  1. O utilizador abre o quadro *Kanban* no plugin do *Teams*.
  2. O utilizador seleciona um ou mais labels para filtrar as tarefas.
  3. O quadro exibe as tarefas filtradas em tempo real.
  4. O utilizador atualiza o estado de uma tarefa (e.g., "Em progresso").
  5. O *middleware* sincroniza a alteração com a *issue* correspondente no *Gitlab*.
- Fluxo Alternativo:
  1. O utilizador aplica um filtro inválido ou sem correspondência.
     - O sistema exibe uma mensagem informando que não existem tarefas correspondentes ao filtro aplicado.

Caso de Uso 3: Sincronização Bidirecional entre *Gitlab* e *Teams*

- Ator Principal: Utilizador Final
- Descrição: O *middleware* sincroniza alterações feitas no *Gitlab* para o *Teams* e vice-versa, assegurando consistência entre as plataformas.
- Requisitos Relacionados: RF22, RNF09, RT08
- Fluxo Principal:
  1. O utilizador atualiza uma *issue* no *Gitlab* (e.g., adiciona um label ou muda o estado para "Fechado").
  2. O webhook notifica o *middleware* sobre a alteração.
  3. O *middleware* processa a alteração e transforma os dados para o formato do *Teams*.
  4. A alteração é refletida no quadro *Kanban* do *Teams*.
  5. O utilizador no *Teams* vê a atualização em tempo real.
- Fluxo Alternativo:
  1. O *middleware* não consegue processar a notificação devido a erro de rede.
     - O sistema tenta novamente o processamento em até três ciclos automáticos (RNF12, RNF13).

Caso de Uso 4: Filtragem de Tarefas no Quadro *Kanban*

- Ator Principal: Utilizador Final
- Descrição: O utilizador filtra tarefas no quadro *Kanban* por rótulos (labels) ou categorias específicas, facilitando a organização e priorização de tarefas.
- Requisitos Relacionados: RF27, RT08
- Fluxo Principal:
  1. O utilizador seleciona um ou mais rótulos no quadro *Kanban*.
  2. O plugin do *Teams* solicita as tarefas filtradas ao *middleware*.
  3. O *middleware* retorna as tarefas correspondentes ao filtro aplicado.
  4. O quadro atualiza-se exibindo apenas as tarefas filtradas.
- Fluxo Alternativo:

1. O *middleware* não responde à solicitação em tempo útil.
    - O sistema exibe uma mensagem informando que os dados não estão disponíveis no momento.

Caso de Uso 5: Gestão de Múltiplos Projetos no *Gitlab*

- Ator Principal: Administrador
- Descrição: O administrador gere múltiplos projetos no *Gitlab*, assegurando que os eventos de cada projeto são processados separadamente pelo *middleware*.
- Requisitos Relacionados: RNF08, RT08
- Fluxo Principal:
    1. O *middleware* recebe notificações de diferentes projetos simultaneamente.
    2. O *middleware* processa os eventos de forma isolada, assegurando que as alterações de um projeto não interferem nos outros.
    3. Cada equipa no *Teams* visualiza apenas as tarefas relacionadas ao seu projeto.
- Fluxo Alternativo:
    1. O *middleware* encontra um erro ao processar eventos de um dos projetos.
        - O sistema gera *Logs* de erro e envia alertas ao administrador para intervenção manual.

Caso de Uso 6: Integração com Múltiplas Equipas no *Teams*

- Ator Principal: Utilizador Final e Administrador
- Descrição: Diferentes equipas no *Teams* utilizam o sistema de forma personalizada para os seus projetos *Gitlab*, com sincronização independente.
- Requisitos Relacionados: RNF09, RT08
- Fluxo Principal:
    1. O administrador mapeia cada projeto *Gitlab* para uma equipa específica no *Teams*.
    2. O *middleware* gere as conexões entre cada projeto e a equipa correspondente.
    3. Os utilizadores de cada equipa visualizam e gerem apenas as tarefas do seu projeto.
- Fluxo Alternativo:
    1. Um utilizador tenta acessar tarefas de outro projeto sem permissão.
        - O sistema bloqueia o acesso e exibe uma mensagem de erro de permissão.

Caso de Uso 7: Visualização de Gráficos de *Gantt* no *Teams*

- Ator Principal: Gestor de Projeto
- Descrição: O gestor utiliza o plugin do *Teams* para visualizar gráficos de *Gantt* com base nos dados do *Gitlab*, filtrando por projetos ou categorias específicas.
- Requisitos Relacionados: RT09, RF29
- Fluxo Principal:
    1. O gestor abre a funcionalidade de gráficos no plugin do *Teams*.
    2. O gestor aplica filtros para gerar um gráfico de *Gantt* específico.
    3. O plugin solicita ao *middleware* os dados atualizados para o gráfico.
    4. O gráfico é gerado e exibido no *Teams*.
- Fluxo Alternativo:
    1. O gestor aplica filtros inválidos ou ausentes no *Gitlab*.

- O sistema exibe uma mensagem indicando que não existem dados suficientes para gerar o gráfico.

3.2 Modelação

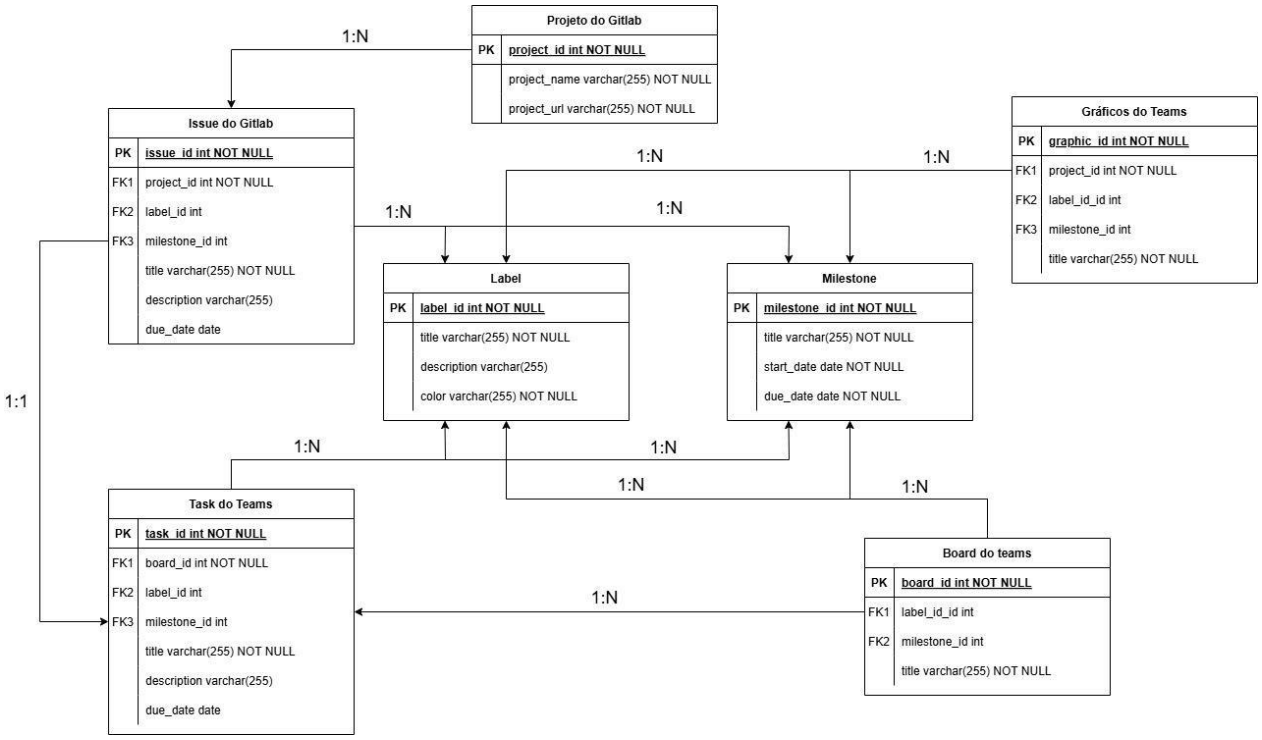


Figura 1 - Modelação

3.3 Interface

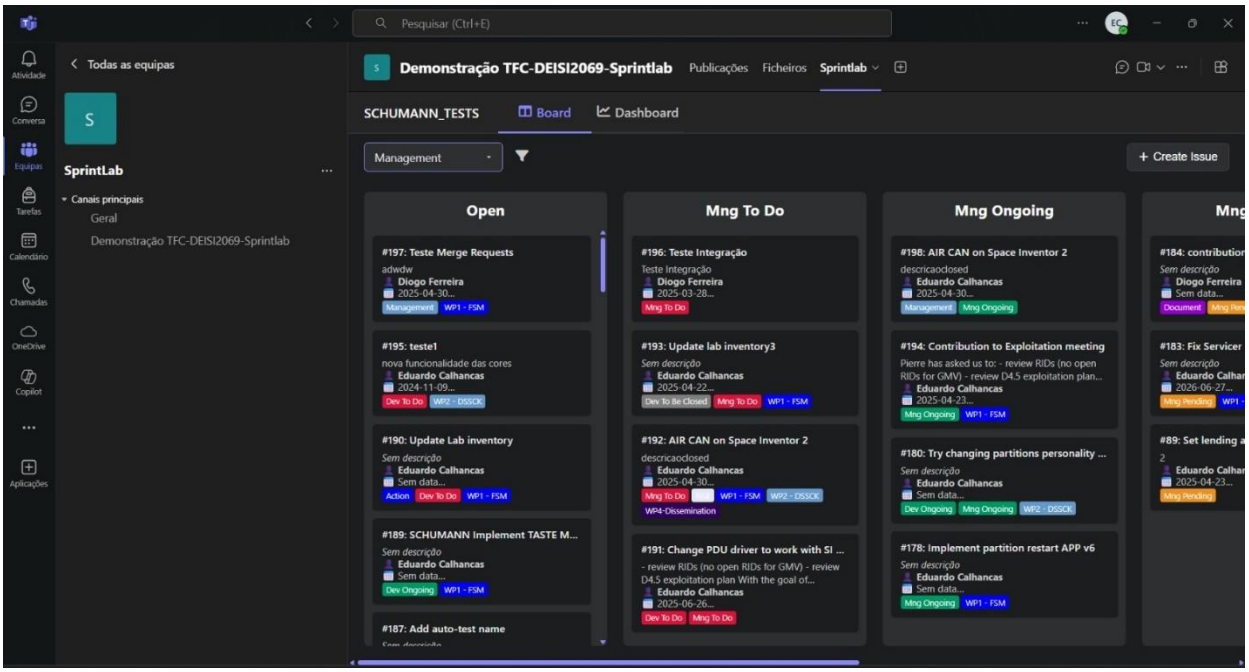


Figura 2 - Interface da Board

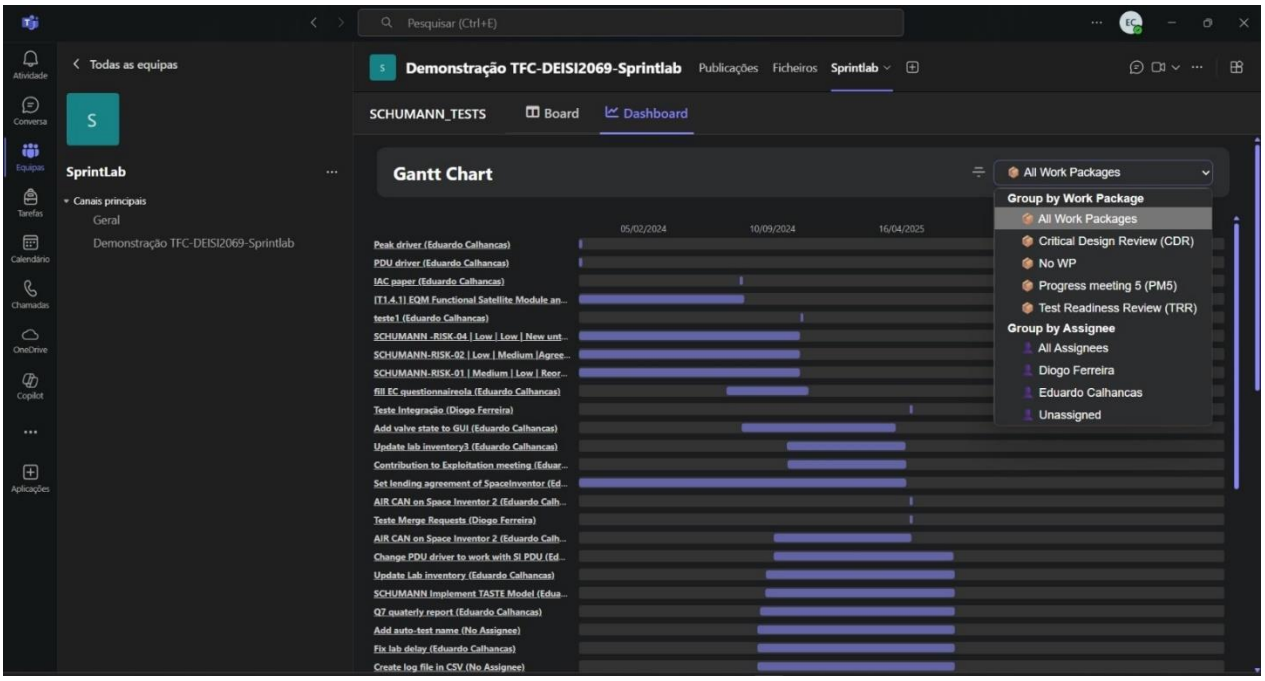


Figura 3 - Interface dos Gráficos 1

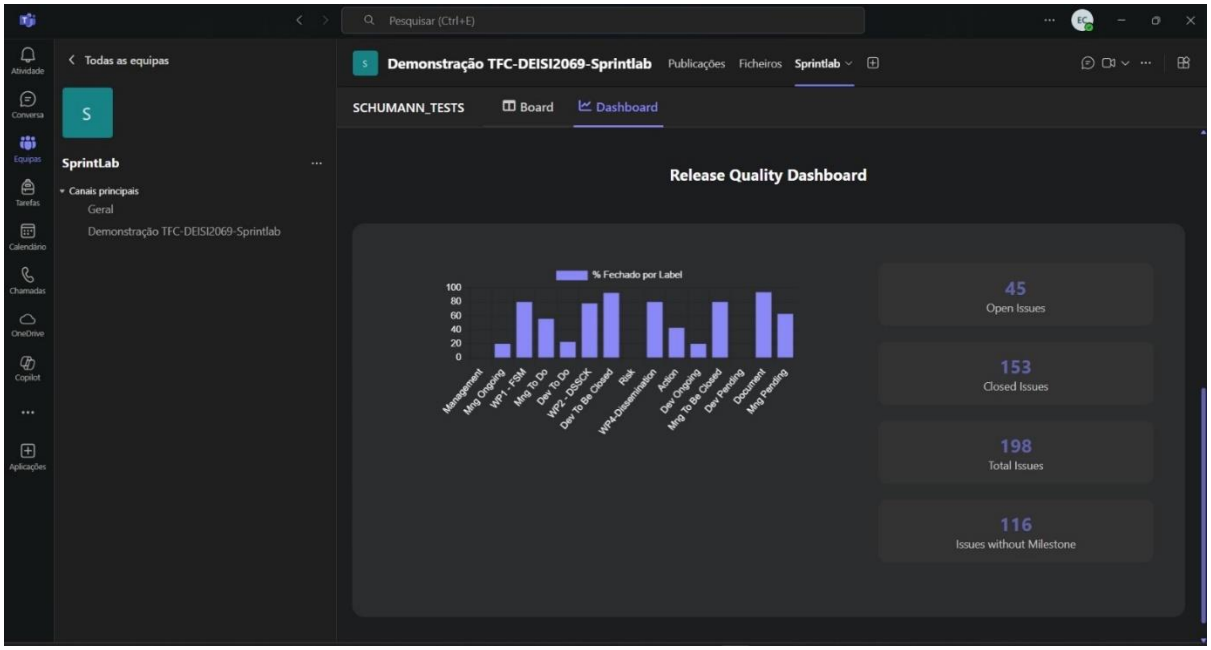


Figura 4 - Interface dos Gráficos 2

4 Solução Proposta

4.1 Apresentação

Nesta secção, apresentamos a solução proposta, que tem como objetivo integrar as plataformas *Gitlab* e *Microsoft Teams*, facilitando a gestão ágil de projetos e a colaboração entre equipas. A solução é composta por um *middleware* e um plugin para *Microsoft Teams*, permitindo uma sincronização bidirecional de dados e a geração automática de gráficos úteis, como *Gantt Charts*.

O *middleware* funciona como o núcleo do sistema, intermediando a comunicação entre as duas plataformas e transformando os dados para garantir consistência. Por outro lado, o plugin para *Microsoft Teams* fornece uma interface intuitiva para a visualização e interação com as informações de projeto.

Descrição Funcional da Solução

- O *middleware* é uma aplicação hospedada num servidor, aplicada no Fly.io, que faz a ponte entre o *Gitlab* e o *Microsoft Teams*. As suas principais funcionalidades incluem:
- A conexão ao *Gitlab* é feita através da API, estando também integrada ao *Microsoft Teams*.
- Validação de permissões, assegurando que apenas utilizadores autorizados possam criar, atualizar ou encerrar tarefas e *issue*s.
- Sincronização bidirecional entre o *Issue Board* do *Gitlab* e o *Kanban Board* do *Teams*, garantindo que alterações em uma das plataformas sejam refletidas na outra.
- Transformação de dados do *Gitlab* em *Gantt Charts*, para envio e visualização no *Teams*.

O plugin para *Microsoft Teams* oferece aos utilizadores funcionalidades diretamente integradas à plataforma, permitindo uma gestão mais eficiente. Entre as suas funcionalidades destacam-se:

- Apresentação de um *Kanban Board* interativa, com filtros para visualizar tarefas específicas por rótulos.
- Exibição de gráficos *Gantt Charts* que representam a timeline de projetos inteiros ou categorias específicas.
- Comunicação direta com o *middleware*, garantindo a atualização dos dados em tempo real.

Comparação com Soluções Alternativas

Na entrega final, será apresentada uma análise comparativa entre as soluções avaliadas durante o projeto e a solução desenvolvida. Este processo demonstrará os

benefícios e as limitações do sistema implementado, reforçando a sua viabilidade e o impacto no fluxo de trabalho.

Recursos Disponíveis

Os seguintes recursos estarão disponíveis para avaliação e validação da solução desenvolvida:

- Vídeo demonstrativo: Explicação detalhada das funcionalidades e do funcionamento da solução.
- Repositório Git: Código-fonte do *middleware* e do plugin, acessível para consulta e revisão.
- Solução funcional: O sistema implementado é disponibilizado como uma aplicação personalizada (custom app) em formato .zip, juntamente com as credenciais e instruções para a sua utilização. Devido a limitações próprias do *Microsoft Teams* fornecido pela lusófona, é disponibilizado um link para uma equipa do *Microsoft Teams* com as indicações necessárias e com o sistema implementado.

Estrutura do Capítulo

Este capítulo inicia-se com a apresentação da proposta da solução e das suas principais funcionalidades, detalhando o *middleware* e o plugin para *Microsoft Teams*. Em seguida, aborda as permissões de acesso e compara a solução com alternativas analisadas ao longo do projeto. Por fim, apresenta-se uma análise final sobre a solução implementada, destacando os seus benefícios e o potencial para evoluções futuras.

4.2 Arquitetura

Nesta secção, detalha-se a arquitetura da solução proposta, descrevendo as tecnologias a utilizar e justificando as principais decisões tomadas durante o desenvolvimento do TFC. A solução foi desenhada para integrar de forma eficiente o *Gitlab* e o *Microsoft Teams*, garantindo escalabilidade, segurança e flexibilidade no ambiente de desenvolvimento.

Identificação e Justificação das Tecnologias Utilizadas

1. *Middleware*:
   - Linguagens de Programação e Tecnologias Utilizadas: JavaScript, SQL, HTML e CSS. JavaScript foi a linguagem escolhida para o desenvolvimento do backend, devido à sua eficiência na construção de aplicações web, à facilidade de integração com serviços externos como o *Gitlab* e *Microsoft Teams*, e à compatibilidade com bibliotecas modernas de comunicação via API. Para armazenamento e consulta de dados, foi utilizada SQL, através de uma base de dados relacional, garantindo persistência, segurança e flexibilidade na gestão das configurações e dos dados do projeto. HTML e CSS foram utilizados no frontend para estruturar e estilizar a interface da aplicação, assegurando uma apresentação clara e responsiva dos dados, gráficos e funcionalidades interativas no ambiente do *Microsoft Teams*.*Framework*: *Express.js.* Utilizado para construir o *backend* em *JavaScript*, permitindo a criação de *APIs RESTful* leves e eficientes, responsáveis pela comunicação com a *API* do *Gitlab* para a gestão de projetos e tarefas.
   - Hospedagem: Planeamento para deployment em serviços cloud, utilizando a

plataforma *Fly.io*, garantindo escalabilidade, alta disponibilidade e fácil gestão de deploys.

- *APIs*: O sistema comunica com duas *APIs* principais. A *Microsoft Graph API* é utilizada para interagir com o *Microsoft Teams*, permitindo o envio de mensagens, acesso a canais e gestão de equipas. A *Gitlab REST API* é usada para obter e atualizar dados de projetos, *issues*, *labels*, *milestones* e utilizadores, permitindo a sincronização em tempo real entre o *Gitlab* e o *Teams*.

2. *Plugin* para *Microsoft Teams*:
    - Tecnologia: Desenvolvido com *JavaScript* e *Express.js* no *backend*, fornecendo uma base sólida e leve para *APIs REST*. A interface foi concebida com *HTML* e *JavaScript* standard, garantindo compatibilidade com o *Microsoft Teams*.
    - Integração: Comunicação direta com o *middleware* para acesso aos dados e geração de gráficos.
    - Exibição: Foram implementados quadros do tipo Kanban e Gantt Charts interativos para visualização e gestão das tarefas. No caso do Kanban Board, a integração com os dados da API foi realizada de forma dinâmica. Para os Gantt Charts, embora tenham sido testadas bibliotecas como o Chart.js, nenhuma se adequava corretamente à estrutura e dinâmica dos dados obtidos via API. Por essa razão, optou-se por desenvolver o gráfico de Gantt manualmente, garantindo total compatibilidade e controlo sobre a renderização e interação com os dados do projeto

Apresentação da Arquitetura

A arquitetura da solução é composta por três componentes principais:

1. *Middleware*:
    - Responsável pela comunicação e transformação de dados entre *Gitlab* e *Microsoft Teams*.
    - Implementa lógica de sincronização bidirecional para garantir a consistência entre ambas as plataformas.
    - Gera e fornece gráficos *Gantt* a partir dos dados do *Gitlab*.

2. Plugin para *Microsoft Teams*:
    - Exibe as informações sincronizadas numa interface intuitiva.
    - Oferece funcionalidades de gestão como *Kanban Board*s e visualização de *Gantt Charts*.

3. Serviços de Terceiros:
    - *Gitlab API*: Fornece dados dos *issue*s, *Board*s e tarefas.
    - *Microsoft Graph API*: Permite a interação com canais e dados do *Teams*.

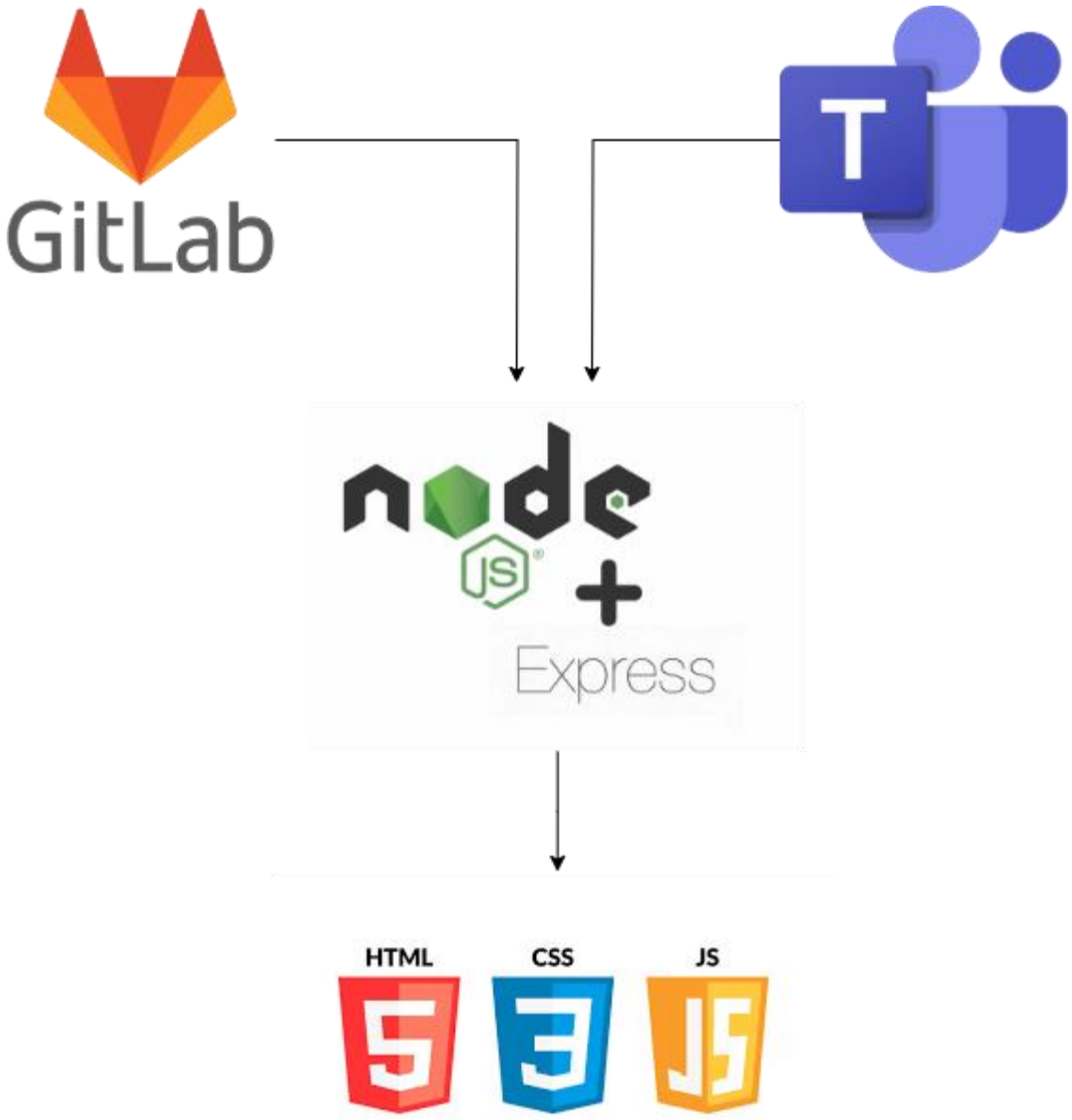


Figura 5 - Arquitetura do Sistema

Fundamentação das Principais Decisões

1. *Middleware*:
   - A utilização de Express.js garante uma solução leve e eficiente, amplamente utilizada para criar *APIs REST* em *Node.js*. A escolha desta tecnologia permitiu uma rápida integração com a API do *Gitlab* e com a base de dados *PostgreSQL*.
   - O *deployment* na *Fly.io* assegura escalabilidade e alta disponibilidade.
2. Plugin:
3. O projeto foca-se na implementação de um middleware backend leve e funcional em Express.js, com integração direta com *Gitlab* e suporte para consumo por interfaces web (ex: dashboards em HTML/JS). A lógica de apresentação pode ser facilmente acoplada a qualquer frontend moderno.
4. *APIs*:

- A integração com *Gitlab* e *Microsoft Teams* através de *API*s bem documentadas garante interoperabilidade e simplifica a implementação.

Resumo das Componentes

- *Middleware*: Ponte entre *Gitlab* e *Teams*, responsável pela lógica de sincronização e transformação de dados.
- *Plugin*: Interface no *Teams* para gestão de tarefas e visualização de gráficos.

A arquitetura proposta permite uma integração robusta, escalável e alinhada com as necessidades do projeto, garantindo uma experiência integrada para os utilizadores.

4.3 Tecnologias e Ferramentas Utilizadas

Vamos A solução foi desenvolvida com uma arquitetura modular composta por frontend (plugin Teams) e backend (middleware Express.js), focada em garantir escalabilidade, interoperabilidade e eficiência.

O middleware, desenvolvido em JavaScript com o framework Express.js, é responsável por intermediar a comunicação entre o *Gitlab* e o *Microsoft Teams*. Nele, foram implementados múltiplos endpoints REST, incluindo um para consultar quadros Kanban e outro para atualizar issues, utilizando axios para comunicação com a API do *Gitlab*. O servidor foi configurado com tratamento de erros, logging, e verificação de parâmetros obrigatórios em cada chamada.

A aplicação foi alojada na plataforma Fly.io, onde se configurou o Dockerfile, as secrets de ambiente (ex. *token*s de autenticação *Gitlab*), e o fly.toml para deploy contínuo. Foi ainda configurado o load balancing e a escalabilidade automática fornecida pela plataforma.

A API do *Gitlab* foi usada de forma extensiva, com autenticação via *token* de acesso privado. Foram implementadas chamadas para listar issues por etiquetas, recuperar listas de boards, milestones, utilizadores e atualizar metadata de cada issue. A estrutura de URLs da API foi parametrizada dinamicamente com base nas configurações da base de dados.

A integração com o *Microsoft Teams* foi realizada através de uma aplicação personalizada (customTeamsApp). Esta aplicação foi desenvolvida em JavaScript e embalada com manifest.json, onde se definiram os *scopes*, permissões e endpoints visíveis na interface do Teams. A aplicação foi instalada no Teams com App Studio e configurada para carregar o frontend diretamente da instância do Fly.io.

O frontend foi desenvolvido com tecnologias web standard (HTML, CSS, JavaScript) e bibliotecas como Chart.js e Grid.js para visualização dos dados em dashboards e quadros Kanban. A interface consome os endpoints do middleware e apresenta os dados organizados por colunas, milestones, e filtros dinâmicos.

As ferramentas de apoio incluíram o Postman, usado para testar e validar todos os endpoints da API, garantindo respostas corretas e deteção de falhas. O projeto foi gerido

com *Gitlab*, utilizando os seus Boards para planeamento, milestones para entregas e o repositório Git para versionamento. O código foi organizado em módulos reutilizáveis e seguiu boas práticas como separação de camadas e reutilização de lógica.

4.4 Ambientes de Teste e de Produção

A solução proposta será desenvolvida considerando dois ambientes principais: o ambiente de teste e o ambiente de produção. Estes ambientes terão configurações distintas para garantir a validação e implementação da solução de forma eficiente, minimizando riscos e maximizando o desempenho na exploração produtiva.

Ambiente de Teste

O ambiente de teste será utilizado para validar as funcionalidades desenvolvidas ao longo do projeto, com foco na identificação de falhas e no aperfeiçoamento da integração entre os componentes. Este ambiente será configurado localmente ou em serviços cloud de menor escala, simulando as interações entre o *middleware*, o *Gitlab* e o *Microsoft Teams*.

Recursos previstos para o ambiente de teste:

- Capacidade de processamento: Um servidor com pelo menos 2 núcleos de CPU e 2 GHz de frequência.
- Memória: Mínimo de 2 GB de RAM, suficiente para testar chamadas *API* simultâneas e executar o *middleware*.
- Rede: Velocidade de conexão mínima de 50 Mbps para garantir respostas rápidas durante os testes de integração.
- Armazenamento: Cerca de 500 MB, necessário para *Logs* de teste e armazenamento temporário de dados transformados.
- Ferramentas:
    - Serviços como Ngrok para simular endpoints públicos.
    - Postman para testar e documentar as chamadas *API*.

Neste ambiente, será possível simular eventos no *Gitlab*, como a criação de *issue*s ou alterações no *Issue Board*, e verificar se a sincronização com o *Microsoft Teams* ocorre como esperado.

Ambiente de Produção

O ambiente de produção será configurado para garantir o desempenho, a escalabilidade e a segurança necessários para a utilização real da solução. O *middleware* será implementado em um serviço cloud, como o *Microsoft* Azure, para assegurar alta disponibilidade e resposta rápida às solicitações.

Recursos necessários para o ambiente de produção:

- Capacidade de processamento: Um servidor cloud dedicado com pelo menos 4 núcleos de CPU e 2.5 GHz de frequência, permitindo múltiplas conexões simultâneas.
- Memória: Pelo menos 4 GB de RAM, garantindo a estabilidade durante picos de utilização.
- Rede: Velocidade mínima de 100 Mbps para comunicação eficiente entre *API*s e resposta rápida ao utilizador.
- Segurança:
    - Certificado SSL/TLS para comunicação segura entre o *middleware* e os serviços

integrados.
    - Gestão de permissões e autenticação para proteger o acesso aos dados e *APIs*.
- Serviços Cloud:
    - *Microsoft* Azure App Services para hospedar o *middleware*.
    - *Microsoft Graph API* para interação com o *Teams*.

No ambiente de produção, a solução será utilizada diretamente pelos utilizadores finais, garantindo que alterações feitas no *Gitlab* ou no *Teams* sejam sincronizadas em tempo real. O *middleware* estará configurado para suportar cargas variáveis e responder a múltiplas solicitações de forma eficiente.

Resumo

O ambiente de teste será configurado para validação e simulação em escala reduzida, enquanto o ambiente de produção terá recursos robustos e escaláveis, assegurando a fiabilidade da solução em utilização real. A escolha de serviços cloud e recursos de rede foi feita para atender aos requisitos funcionais, mantendo a flexibilidade para evoluções futuras. Esta configuração permitirá que o sistema funcione de forma segura, eficiente e integrada, cumprindo os objetivos do projeto.

4.5 Abrangência

A solução proposta aplica conhecimentos adquiridos em diversas unidades curriculares e áreas científicas do curso, integrando competências técnicas e práticas para abordar os desafios do projeto. Abaixo, são descritas as principais unidades curriculares e os aspetos aprendidos que serão aplicados no desenvolvimento da solução.

Unidades Curriculares

1. Engenharia de Software:

Conceitos Aplicados:

- Planeamento e gestão de projetos ágeis, incluindo a definição de requisitos e a priorização de tarefas.
- Modelação e análise de sistemas para identificar fluxos de dados e pontos de integração entre os componentes.

Aplicação no Projeto:

- Planeamento e execução do desenvolvimento do *middleware* e do plugin para o *Microsoft Teams*.
- Utilização de metodologias ágeis para organizar sprints e acompanhar o progresso do trabalho, com ferramentas como o Trello.

2. Programação Web:

Conceitos Aplicados:

- Desenvolvimento de APIs RESTful para comunicação entre serviços, utilizando o framework Express.js em JavaScript.
- Estruturação de aplicações web utilizando tecnologias web standard em JavaScript, com uma aplicação personalizada integrada no *Microsoft Teams*.

Aplicação no Projeto:

- Desenvolvimento do *middleware* que interliga o *Gitlab* e o *Microsoft Teams*.
- Implementação do plugin para o *Teams*, garantindo uma interface moderna, funcional e acessível aos utilizadores.

3. Fundamentos de Sistemas de Informação:

Conceitos Aplicados:

- Gestão de permissões e definição de níveis de acesso para diferentes tipos de utilizadores.
- Organização eficiente de fluxos de trabalho em sistemas de colaboração.

Aplicação no Projeto:

- Configuração e validação de permissões no *Gitlab* e no *Microsoft Teams*.
- Implementação de boas práticas para gerir e sincronizar dados entre os dois sistemas.

4. Engenharia de Requisitos e Testes:

Conceitos Aplicados:

- Levantamento e especificação de requisitos funcionais e não funcionais.
- Validação de requisitos com base em testes sistemáticos.

Aplicação no Projeto:

- Definição detalhada dos requisitos do sistema, como sincronização bidirecional e permissões configuráveis.
- Testes das funcionalidades do *middleware* e do plugin, incluindo integração com *APIs* de terceiros.

5. Algoritmos e Estruturas de Dados:

Conceitos Aplicados:

- Transformação e manipulação de dados para visualização e integração.
- Desenvolvimento de algoritmos eficientes para gerar gráficos como os *Gantt Charts*.

Aplicação no Projeto:

- Processamento dos dados recolhidos do *Gitlab* para criar representações visuais no *Microsoft Teams*.
- Implementação de algoritmos para filtrar e categorizar os dados exibidos nos *Kanban Boards* e *Gantt Charts*.

Conclusão

A solução proposta reflete a aplicação interdisciplinar dos conhecimentos adquiridos ao longo do curso. A integração de áreas como engenharia de software, programação web, fundamentos de sistemas de informação, engenharia de requisitos e testes e algoritmos demonstra como as competências desenvolvidas durante a formação se complementam para resolver problemas complexos e propor soluções inovadoras. Cada aspeto do projeto

reflete a aplicação prática dos conteúdos programáticos, alinhando a teoria à prática num contexto real e focado em resultados.

4.6 Componentes

Nesta secção iremos falar sobre as principais componentes da solução proposta, destacando os aspetos técnicos envolvidos na sua implementação. Os dois componentes principais são o *middleware* e o plugin do *Microsoft Teams*, com suporte para permissões configuráveis, garantindo um controlo eficiente de acesso.

4.6.1 Middleware

O middleware é o núcleo do sistema, responsável por processar eventos e transformar dados entre o *Gitlab* e o *Microsoft Teams*, além de assegurar a sincronização. Este componente foi construído com Express.js, utilizando JavaScript para criar uma API RESTful leve e eficiente.

Características Técnicas do Middleware

- Hospedagem
  - O *middleware* foi implementado num servidor configurado (Fly.io) para operação contínua, garantindo alta disponibilidade e resiliência.
- Integração com o *Gitlab*
  - Comunicação com o *Gitlab*: Realizada através da API oficial do *Gitlab*, permitindo ao middleware consultar e atualizar informações de projetos, issues, labels e milestones de forma ativa.
  - A biblioteca *axios* foi utilizada no *backend Node.js* para interagir com a *API* do *Gitlab*, permitindo operações como leitura e atualização de issues, labels, milestones e utilizadores associados ao projeto.
- Integração com o *Teams*
  - *API*: Utilizada para criar e gerir tarefas no *Kanban*, além de sincronizar alterações realizadas pelos utilizadores no *Teams*.
- Transformação de Dados
  - Conversão de dados do *Gitlab* (*issues*, etiquetas) para um formato compatível com o quadro *Kanban* no *Teams*.
  - Gerar gráficos de *Gantt* com base em tarefas do *Gitlab* criado de raiz sem recorrer a uso de bibliotecas como *Plotly* ou *Matplotlib*.
- Sincronização Bidirecional
  - Do *Teams* para o *Gitlab*: Alterações feitas no *Kanban* (por exemplo, mover uma tarefa entre colunas) refletem-se diretamente no estado das *issues* no *Gitlab*.
  - Do *Gitlab* para o *Teams*: A criação, atualização e exclusão de *issues* no *Gitlab* atualizam automaticamente o *Kanban* no *Teams*.
- Segurança
  - Comunicação com *APIs* externas realizada exclusivamente por *HTTPS*.
  - A autenticação com a *API* do *Gitlab* é feita através de *tokens* privados (*Personal Access Tokens*), armazenados na base de dados e associados a cada equipa e canal.
- Desafios Técnicos e Soluções
  - Alta performance: Utilização de operações assíncronas para lidar com grandes volumes de eventos da API.

- Escalabilidade: Configuração de balanceamento de carga e tolerância a falhas no servidor.

4.6.2 Plugin do *Microsoft Teams*

O *plugin* é o ponto de interação dos utilizadores com os dados sincronizados, fornecendo uma interface intuitiva para visualizar e manipular tarefas e gráficos.

- Características Técnicas do *Plugin*
  - Conexão com o *Middleware*
  - O *plugin* utiliza *endpoints REST* do *middleware* para obter dados atualizados de tarefas e gráficos.
- Interface *Kanban Board*
  - Exibe as tarefas do *Gitlab* como um quadro *Kanban*.
  - Permite filtros por etiquetas, possibilitando aos utilizadores visualizarem apenas as tarefas relevantes para o seu trabalho.
- Interface *Gantt Chart*
  - Exibe gráficos de *Gantt* para projetos completos ou tarefas filtradas por etiquetas.
  - Gráficos gerados em tempo real pelo middleware e exibidos no plugin.
- Consistência de Dados
  - Todos os campos disponíveis no *Gitlab* (por exemplo, título, descrição, etiquetas) são exibidos no *Kanban* e nos gráficos no *Teams*.
- Tecnologias Utilizadas
  - *Microsoft Teams SDK*: Para integração nativa com a interface do *Teams*.
  - Desenvolvimento da interface do plugin: Realizado com tecnologias web standard em JavaScript, criando uma interface responsiva e funcional no *Microsoft Teams*.
- Desafios Técnicos e Soluções
  - Responsividade: Garantir que a interface carregue rapidamente, mesmo com múltiplos gráficos ou tarefas.
  - Fidelidade de Dados: Implementação de mecanismos de cache para evitar inconsistências nas sincronizações.

4.6.3 Painel Administrativo

O painel administrativo pode ser implementado para gerir configurações do middleware e monitorizar o estado da integração.

Funcionalidades:

- Configurar a *access token* do *Gitlab* diretamente no middleware.
- Monitorizar o estado dos eventos (sucesso/falha).
- Ajustar permissões de sincronização e acesso.

Tecnologias utilizadas:

- Express.js para desenvolvimento das APIs administrativas.
- Tecnologias web standard em JavaScript para a criação da interface web integrada no *Microsoft Teams*, sem uso de frameworks como React ou Bootstrap.

Desafios Técnicos e Soluções:

- Garantir que o acesso à configuração e gestão do sistema seja feito de forma segura e controlada.
- Tornar a interface simples e intuitiva para facilitar a utilização e a configuração inicial por parte dos utilizadores.

4.6.4 Resumo Final dos Componentes

1. *Middleware*: Gere a sincronização de dados entre *Gitlab* e *Teams*.
2. *Plugin* do *Microsoft Teams*: Interface visual para tarefas e gráficos.
3. Sistema de Permissões: Controlo de acesso baseado em papéis.
4. Painel Administrativo: Interface para gerir e monitorar o sistema.

5 Testes e Validação

5.1 Objetivo dos Testes

O principal objetivo dos testes é demonstrar que a solução desenvolvida cumpre os requisitos e objetivos definidos, nomeadamente a integração eficaz entre o *Gitlab* e o *Microsoft Teams*, através do middleware implementado.

Pretende-se validar que a solução é aplicável num contexto real de gestão de projetos, contribuindo para a melhoria da comunicação e organização de tarefas.

Os testes irão focar-se não apenas no funcionamento técnico das funcionalidades, mas também na demonstração da pertinência, aplicabilidade e relevância da solução no contexto para o qual foi concebida.

Será ainda avaliada a robustez do sistema, a sua escalabilidade em ambiente cloud (Fly.io) e a capacidade de manter a sincronização entre os diferentes serviços envolvidos.

Este processo de validação prática será fundamental para comprovar que a solução é capaz de resolver um problema real, satisfazendo os critérios de aceitação definidos e respondendo aos desafios operacionais previstos.

5.2 Abordagem de Testes

A abordagem de testes adotada para a validação do SprintLab segue uma metodologia prática, iterativa e orientada a cenários reais de utilização. O foco principal consistiu em garantir que todas as funcionalidades críticas fossem testadas do ponto de vista técnico e funcional, cobrindo o ciclo completo de interação entre *Gitlab*, o middleware e o *Microsoft Teams*.

Os testes foram estruturados em torno de casos de uso representativos, que simulam as principais ações dos utilizadores finais — como criar, editar e mover issues, alterar configurações de projeto, aplicar filtros e utilizar os gráficos de Gantt. Além disso, foram incluídos testes específicos para avaliar o comportamento do sistema em situações de falha, como autenticação inválida, reinício do serviço ou latência sob carga.

A validação das funcionalidades foi realizada através de testes manuais funcionais, complementados com observação de logs, mensagens de erro e análise de comportamento da interface. Foram ainda aplicadas técnicas de testes exploratórios para identificar possíveis falhas não previstas nos cenários planeados.

Por fim, testou-se também a robustez da solução em ambiente de produção (Fly.io), a capacidade de recuperação após reinícios e a proteção de dados sensíveis, como *token*s de autenticação. Esta abordagem prática e focada em critérios de aceitação permitiu garantir um elevado grau de confiança no funcionamento e aplicabilidade da solução desenvolvida.

5.3 Testes a Realizar

Teste	Cenário	Procedimento	Resultado Esperado
T1	Configurar sprintlab com projeto *Gitlab* no domínio público *Gitlab* (*Gitlab*.com)	No painel de configuração, inserir "*Gitlab*.com" como URL da instância *Gitlab* e carregar *token* válido.	A ligação ao projeto é validada com sucesso. As boards e issues carregam corretamente.
T2	Configurar sprintlab com projeto *Gitlab* no domínio institucional *Gitlab* da GMV (git-ext.gmv.com)	No painel de configuração, inserir "git-ext.gmv.com" como URL da instância *Gitlab* e carregar *token* institucional	A ligação à instância institucional é validada com sucesso. As boards e issues são carregadas corretamente.

		válido.	
T3	Mudar de board	Selecionar uma nova board através do dropdown de boards.	As colunas e issues atualizam-se corretamente para a nova board selecionada.
T4	Criar um issue sem labels	Criar um issue sem selecionar qualquer label no modal de criação.	O issue é criado no *Gitlab* e colocado na coluna padrão ("Open").
T5	Editar o título do issue	Editar o título de um issue existente no modal de edição.	O novo título é guardado e refletido no *Gitlab* e na board.
T6	Editar a descrição do issue	Atualizar a descrição de um issue no modal de edição.	A descrição é atualizada corretamente no *Gitlab* e exibida na board.
T7	Editar o assignee do issue	Atribuir ou alterar o assignee de um issue no modal de edição.	O assignee é atualizado com sucesso no *Gitlab* e visível na board.
T8	Editar a due date do issue	Definir ou alterar a data de entrega de um issue no modal de edição.	A due date é registada corretamente no *Gitlab* e exibida na board.
T9	Editar as labels do issue	Adicionar ou remover labels no modal de edição.	As labels são atualizadas no *Gitlab* e o issue muda de coluna se necessário.
T10	Editar a milestone do issue	Selecionar ou alterar a milestone associada a um issue.	A milestone é atualizada corretamente no *Gitlab*.
T11	Drag and drop de uma coluna para outra	Arrastar um issue de uma coluna para outra diferente.	As labels do issue são atualizadas no *Gitlab* para refletir a nova coluna.
T12	Abrir um issue fechado	Tentar abrir um issue que se encontra na coluna "Closed", através do botão de toggle no modal de edição.	O estado do issue é alterado para "Open" e movido para a coluna correspondente.
T13	Fechar um issue aberto	Fechar um issue aberto através do botão de toggle no modal de edição.	O issue é movido para a coluna "Closed" e atualizado no *Gitlab* como "Closed".
T14	Filtrar issues por labels	Utilizar o botão de filtro para selecionar uma label.	Apenas os issues com a label selecionada permanecem visíveis na board.
T15	Abrir issue no *Gitlab* através da edição	Clicar no botão "Open in *Gitlab*" dentro do modal de edição de um issue.	A página do issue abre corretamente no navegador, no repositório *Gitlab*.

T16	Abrir merge request no *Gitlab*	No modal de edição, clicar num merge request relacionado.	O merge request abre corretamente no navegador, no repositório *Gitlab*.
T17	Criar um projecto no *Gitlab* e associar no SprintLab	Criar manualmente um projecto no *Gitlab* e configurar no SprintLab através do painel de configuração inicial.	O novo projecto é associado com sucesso e as boards são carregadas corretamente no Teams.
T18	Criar múltiplos issues em simultâneo	Criar vários issues rapidamente no modal de criação.	Todos os issues são criados no *Gitlab* e exibidos corretamente na board.
T19	Testar a resposta da API com *token* inválido	Alterar o *token* do projeto para um valor inválido no SprintLab e tentar carregar a board.	O SprintLab detecta a falha de autenticação e apresenta uma mensagem de erro clara ao utilizador.
T20	Filtrar Gantt Chart por milestones	No Dashboard, alterar o filtro de visualização para ver apenas uma milestone.	Apenas as tarefas associadas à milestone selecionada são mostradas no Gantt Chart.
T21	Abrir issue no *Gitlab* através do Gantt Chart	Clicar num issue representado na linha do tempo do Gantt Chart.	A página do issue abre corretamente no *Gitlab* num novo separador.
T22	Reiniciar o serviço no Fly.io e validar estabilidade	Realizar o restart manual do serviço no Fly.io e aceder imediatamente ao SprintLab no Teams.	O SprintLab mantém-se funcional após o restart e recupera o contexto de operação.
T23	Testar latência e carregamento de boards grandes	Utilizar um projeto *Gitlab* com mais de 500 issues e carregar a board no SprintLab.	O SprintLab carrega todos os issues sem erros críticos e com tempos de resposta aceitáveis.
T24	Verificar proteção de dados sensíveis (*token*/API key)	Analisar chamadas de rede e interface para verificar se *token*s de autenticação aparecem expostos.	Nenhum *token* ou dado sensível é visível ou transmitido em canais inseguros.

Tabela 5 - Testes

5.4 Justificação dos Testes

A realização dos testes foi essencial para assegurar que a solução SprintLab cumpre os requisitos funcionais, não funcionais e técnicos definidos ao longo do projeto. A complexidade da integração entre o *Gitlab*, o middleware e o *Microsoft Teams* exigiu uma validação rigorosa em diferentes camadas do sistema, desde a comunicação entre APIs até à apresentação correta de dados na interface do utilizador.

Os testes permitiram confirmar que:

- As funcionalidades críticas operam de forma fiável em contexto real;
- O middleware responde corretamente aos eventos do *Gitlab* e sincroniza com o Teams;
- O plugin do Teams apresenta uma experiência de utilização estável, mesmo com grande

volume de dados;
- O sistema é resiliente a falhas, como *token*s inválidos, reinícios do servidor ou latência de rede;
- A segurança e confidencialidade dos dados são garantidas, nomeadamente na gestão de *token*s e informações sensíveis.

Adicionalmente, os testes forneceram evidência concreta de que a solução desenvolvida é aplicável e útil num cenário real de gestão de projetos Agile, cumprindo o objetivo de facilitar a comunicação e organização de tarefas entre equipas técnicas.

A abordagem baseada em cenários práticos e mensuráveis, acompanhada por critérios de aceitação bem definidos, assegura que a validação não se limita ao aspeto técnico, mas também à pertinência e relevância funcional da solução no seu contexto de utilização.

5.5 Plano de Execução dos Testes

O plano de execução dos testes foi concebido para validar a solução SprintLab em múltiplas dimensões — técnica, funcional e operacional — assegurando que todos os requisitos definidos são efetivamente cumpridos. A execução dos testes decorreu em ambiente controlado, utilizando tanto repositórios públicos (*Gitlab*.com) como instâncias privadas (ex. git-ext.gmv.com), garantindo representatividade dos cenários reais.

O plano incluiu os seguintes eixos principais:

1. Configuração e Integração Inicial: Verificação da ligação correta com instâncias *Gitlab*, configuração de *token*s, e sincronização com o *Microsoft Teams*.
2. Operações sobre Issues e Boards: Testes de criação, edição, movimentação, filtragem e sincronização de issues em tempo real entre o *Gitlab* e o SprintLab (via middleware).
3. Interação com Merge Requests e Milestones: Validação da correta integração com merge requests e milestones do *Gitlab*, assegurando navegação e rastreabilidade através do plugin no Teams.
4. Visualização e Usabilidade: Testes de visualização em board Kanban e Gantt Chart, com foco em responsividade, filtragem e atualização automática.
5. Segurança e Autenticação: Simulações com *token*s inválidos ou expirados, validação da não exposição de dados sensíveis e análise das mensagens de erro apresentadas.
6. Desempenho e Robustez: Testes sob carga com grandes volumes de issues, reinício manual do serviço em Fly.io, e verificação de continuidade e consistência do sistema.

Cada teste foi documentado com base nos critérios de aceitação definidos, e os resultados esperados foram comparados com os comportamentos observados. Foram ainda monitorizados os logs do middleware e os tempos de resposta da interface, para avaliar métricas de desempenho e estabilidade. A execução seguiu uma abordagem iterativa, com validações contínuas ao longo do desenvolvimento, culminando numa fase final de testes sistemáticos, listada na secção 5.3. Esta abordagem garantiu uma cobertura abrangente e uma elevada confiança na qualidade e aplicabilidade da solução.

5.6 Validação Externa

A empresa GMV procedeu à avaliação e teste da aplicação SprintLab como ferramenta operacional de apoio à gestão de desenvolvimento de software, nomeadamente na integração entre *Gitlab* e *Microsoft Teams*. Esta avaliação foi realizada de forma independente, tendo em conta os critérios definidos no âmbito do projeto, incluindo o cumprimento de requisitos funcionais e a execução de testes relevantes.

Os resultados obtidos foram positivos, demonstrando a viabilidade e a utilidade prática da solução desenvolvida pelos alunos. Em anexo encontra-se o relatório técnico emitido pela GMV, onde consta a análise completa efetuada, incluindo observações sobre a implementação, desempenho e usabilidade da ferramenta.

6 Método e Planeamento

6.1 Planeamento inicial

O planeamento inicial do projeto foi estruturado para garantir uma abordagem metódica e eficiente no desenvolvimento da solução, alinhando-se ao calendário de entregas e aos requisitos estabelecidos. O método de trabalho combina elementos da metodologia ágil com ferramentas de gestão de projeto tradicional, permitindo flexibilidade na execução das tarefas e uma visão clara dos prazos e marcos do projeto. A abordagem ágil organiza o trabalho em ciclos iterativos e incrementais, denominados sprints, com uma duração aproximada de duas a três semanas. Cada sprint será orientado para a entrega de resultados concretos, como a implementação de funcionalidades específicas ou a realização de testes de integração, e inclui etapas de planeamento, execução e revisão.

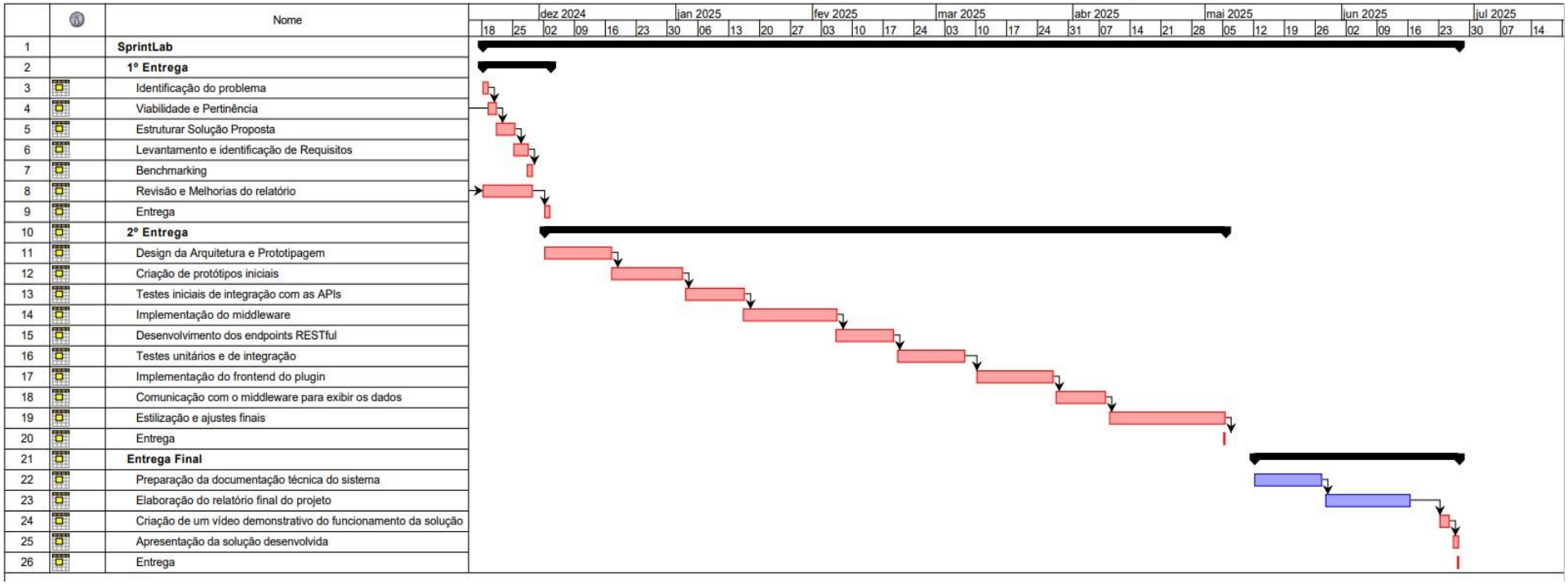
Para garantir o cumprimento dos objetivos, foi elaborado um cronograma detalhado que divide o projeto em três fases principais, alinhadas às datas de entrega definidas. A primeira fase, com prazo até 1 de dezembro de 2024, foca na identificação do problema, análise da viabilidade da solução e levantamento de requisitos. Além disso, inclui a estruturação inicial da solução proposta, benchmarking e a preparação do relatório correspondente.

A segunda fase, com entrega prevista para 27 de abril de 2025, concentra-se no design da arquitetura do sistema e na prototipagem inicial, juntamente com os testes de integração com as *API*s. Nesta etapa, será desenvolvido o *middleware*, com implementação dos endpoints RESTful, e iniciado o desenvolvimento do plugin para o *Microsoft Teams*. A comunicação entre o *middleware* e o plugin será validada, e serão realizados testes unitários e de integração para garantir a consistência do sistema.

A fase final, com entrega até 27 de junho de 2025, será dedicada à finalização do *middleware* e do plugin, com foco na realização de testes extensivos de integração, validação e qualidade. Serão também elaboradas a documentação técnica e funcional e um vídeo demonstrativo do funcionamento do sistema, culminando na entrega do relatório final e na apresentação da solução desenvolvida.

O planeamento é suportado por ferramentas como o Trello, para gestão de tarefas e acompanhamento de sprints, e *Gitlab*, para controlo de versões. O backlog do projeto contém todas as tarefas necessárias, priorizadas com base na relevância e viabilidade técnica. Exemplos de user stories incluem funcionalidades como a exibição de um *Kanban Board* no *Teams* ou a sincronização bidirecional entre o *Gitlab* e o *Microsoft Teams*. O esforço alocado a cada tarefa será estimado e monitorizado ao longo do projeto, garantindo a entrega progressiva das funcionalidades.

Ao longo do desenvolvimento, serão realizados ajustes regulares ao cronograma para lidar com desafios técnicos e desvios inesperados. Entre as dificuldades previstas estão a integração com *API*s de terceiros, a sincronização bidirecional de dados e a validação de permissões. Até à entrega final, o foco estará na realização de testes de qualidade e aceitação, assegurando que a solução final esteja alinhada com os objetivos e expectativas estabelecidos. Este planeamento inicial oferece uma base sólida para o desenvolvimento do projeto, com a flexibilidade necessária para lidar com imprevistos e assegurar o cumprimento dos prazos.



		Nome
1		**SprintLab**
2		**1º Entrega**
3		Identificação do problema
4		Viabilidade e Pertinência
5		Estruturar Solução Proposta
6		Levantamento e identificação de Requisitos
7		Benchmarking
8		Revisão e Melhorias do relatório
9		Entrega
10		**2º Entrega**
11		Design da Arquitetura e Prototipagem
12		Criação de protótipos iniciais
13		Testes iniciais de integração com as APIs
14		Implementação do middleware
15		Desenvolvimento dos endpoints RESTful
16		Testes unitários e de integração
17		Implementação do frontend do plugin
18		Comunicação com o middleware para exibir os dados
19		Estilização e ajustes finais
20		Entrega
21		**Entrega Final**
22		Preparação da documentação técnica do sistema
23		Elaboração do relatório final do projeto
24		Criação de um vídeo demonstrativo do funcionamento da solução
25		Apresentação da solução desenvolvida
26		Entrega

Figura 6 - Gantt Chart do Projeto

6.2 Análise Crítica ao Planeamento

O planeamento inicial do projeto SprintLab foi fundamental para orientar o desenvolvimento da solução, permitindo definir fases, prazos, marcos de validação e prioridades. No entanto, ao longo da execução do TFC, foi necessário ajustar alguns aspetos do plano inicial, de forma a acomodar desafios técnicos não previstos, oportunidades de melhoria e novas funcionalidades identificadas com o avanço do trabalho.

Em termos globais, o plano foi realista e funcional, especialmente no que diz respeito à divisão por etapas (análise, desenvolvimento, testes e validação). A definição de requisitos através da técnica MoSCoW permitiu uma clara distinção entre funcionalidades essenciais e complementares, facilitando a tomada de decisões quando surgiram constrangimentos de tempo ou recursos.

Contudo, é importante destacar que:

- A estimativa temporal para testes e integração revelou-se otimista. A fase de testes exigiu mais tempo do que o inicialmente previsto, sobretudo devido à complexidade da sincronização em tempo real entre *Gitlab* e Teams.
- O planeamento inicial não contemplava algumas melhorias que surgiram durante o desenvolvimento, como a geração de gráficos de Gantt filtráveis, integração com múltiplas equipas, ou suporte a múltiplos ambientes (dev, staging e produção). Estas evoluções implicaram uma reorganização parcial do cronograma.
- A validação externa pela GMV foi uma mais-valia não planeada inicialmente, que enriqueceu o processo e contribuiu para a credibilidade da solução, mas que também exigiu alinhamento adicional e tempo de resposta para feedback.

Apesar destas adaptações, o planeamento revelou-se um guia útil e flexível, que conseguiu sustentar o projeto mesmo perante mudanças e imprevistos. A experiência reforça a importância de prever margens de manobra e manter uma abordagem iterativa, que permita reavaliar prioridades à medida que o projeto evolui.

7 Resultados

7.1 Resultados dos Testes

Teste T1 - Configurar SprintLab com *Gitlab* Público (*Gitlab*.com)

- **Objetivo:** Validar que o SprintLab consegue ser corretamente configurado com um projeto *Gitlab* alojado no domínio público *Gitlab*.com.
- **Pré-condições:**
  - O utilizador possui um projeto criado em *Gitlab*.com.
  - O *token* de acesso é válido e tem permissões suficientes (read_api, read_repository).
  - O SprintLab encontra-se instalado e funcional dentro do *Microsoft Teams*.
- **Procedimento:**
  1. Aceder à aplicação SprintLab no *Microsoft Teams*.
  2. Inserir *Gitlab*.com no campo "*Gitlab* URL".
  3. Introduzir o *token* pessoal no campo "Project Access *Token*".
  4. Clicar em "Guardar".
  5. Confirmar o carregamento das boards disponíveis e issues.
- **Resultado Esperado:**
  - A ligação ao projeto é validada com sucesso.
  - As boards e issues são carregadas corretamente na interface.
- **Resultado Obtido:** Passou - A ligação ao *Gitlab* público foi estabelecida e o projeto carregado sem erros.
- **Evidência:**

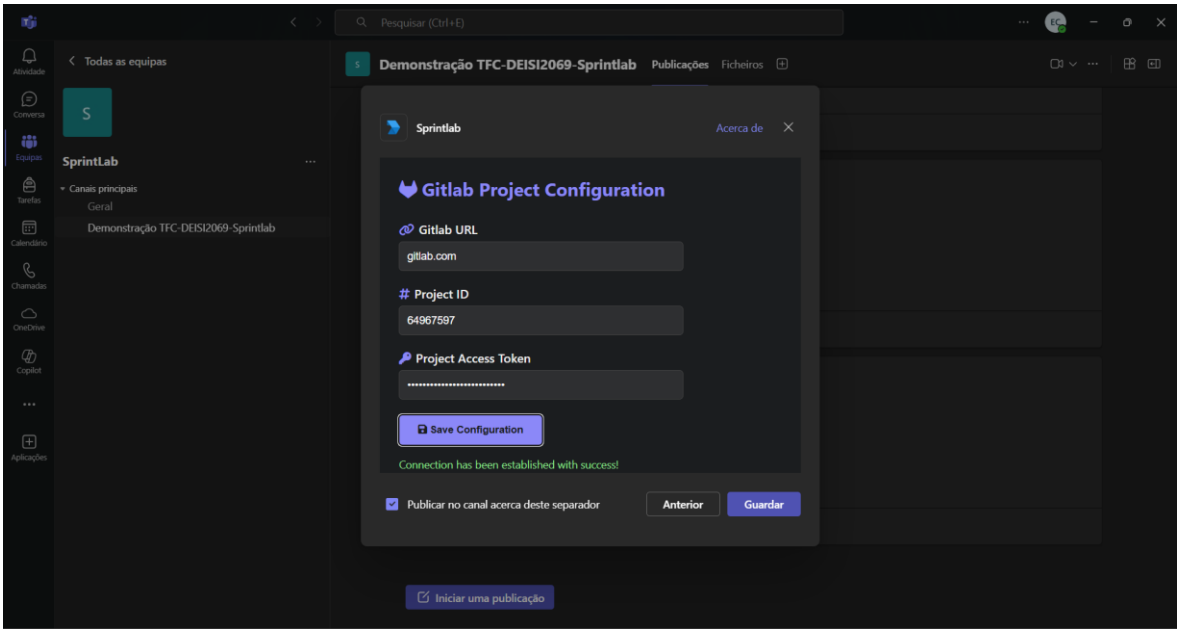


Figura 7 - Teste 1 - 1

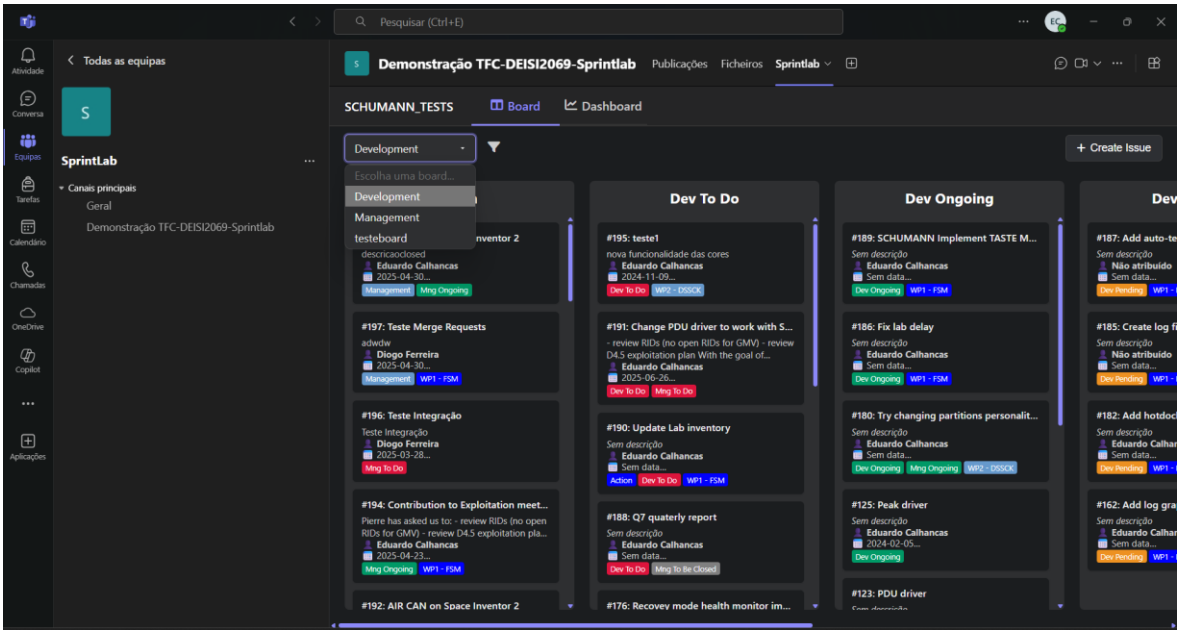


Figura 8 - Teste 1 - 2

Teste T2 - Configurar SprintLab com *Gitlab* Institucional (git-ext.gmv.com)

- **Objetivo**: Validar a ligação do SprintLab a um projeto *Gitlab* numa instância institucional privada (ex.: git-ext.gmv.com).
- **Pré-condições**:
  - Existe uma instância privada de *Gitlab* acessível em git-ext.gmv.com.
  - O utilizador tem um *token* institucional válido.
  - O projeto e as boards estão criados na instância.
  - O SprintLab está disponível no *Microsoft Teams*.
- **Procedimento**:
  1. Abrir a aplicação SprintLab no Teams.
  2. Inserir git-ext.gmv.com no campo "*Gitlab* URL".
  3. Preencher o campo de *token* com um PAT válido da instância privada.
  4. Clicar em "Guardar e Validar".
  5. Verificar o carregamento correto das boards e issues.
- **Resultado Esperado**:
  - A ligação ao projeto da instância institucional é bem-sucedida.
  - As boards e issues surgem corretamente no SprintLab.
- **Resultado Obtido**: Passou - O sistema validou o domínio privado e carregou o conteúdo do projeto com sucesso.
- **Evidência**:

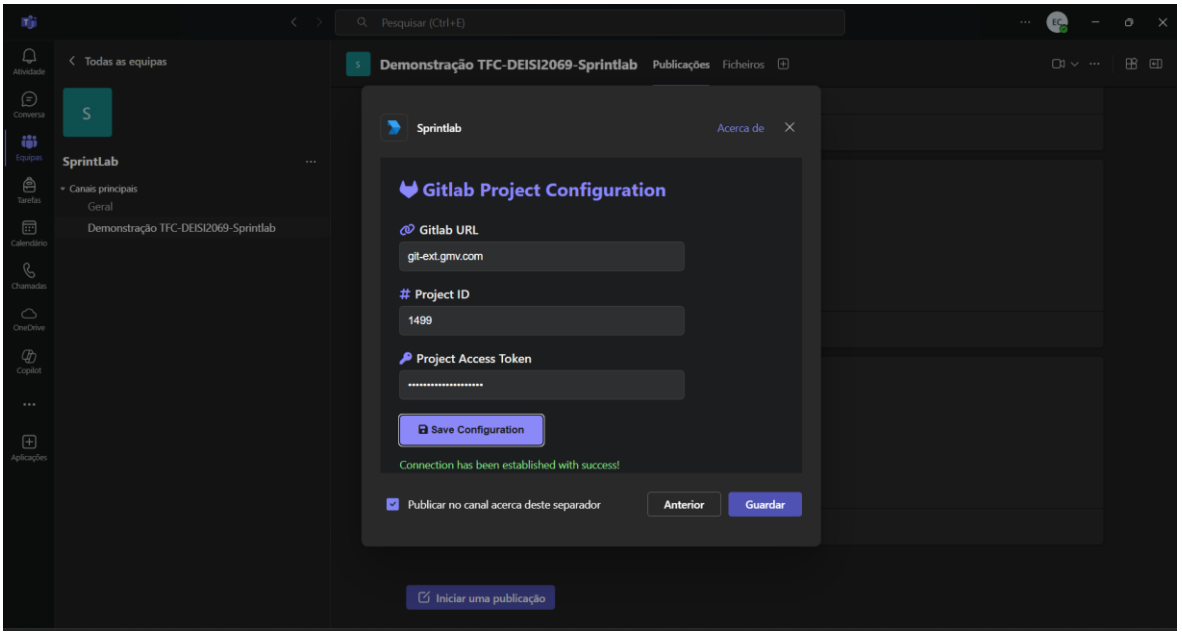


Figura 9 - Teste 2 - 1

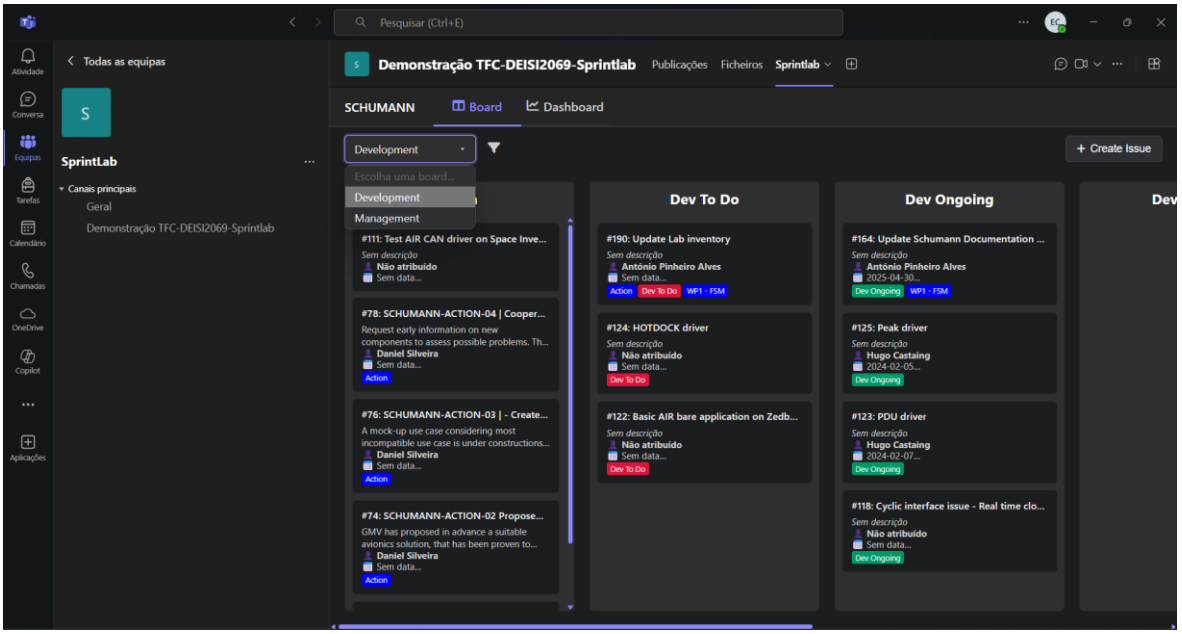


Figura 10 - Teste 2 - 2

Teste T3 - Mudar de board

- **Objetivo**: Verificar se a aplicação SprintLab atualiza corretamente a visualização da board quando o utilizador seleciona uma nova board no dropdown.
- **Pré-condições**:
  - O projeto *Gitlab* tem uma ou mais boards configuradas.
  - O SprintLab já está corretamente ligado ao projeto.
- **Procedimento**:
  1. Aceder ao SprintLab dentro do *Microsoft Teams*.
  2. No topo da interface, clicar no dropdown de boards.
  3. Selecionar uma board diferente da que está visível.
  4. Observar a atualização das colunas e issues.
- **Resultado Esperado**:

    - As colunas exibidas são substituídas pelas da nova board.
    - As issues visíveis correspondem às da nova board selecionada.
- **Resultado Obtido**:
  Passou — A aplicação atualizou corretamente a interface para a nova board.
- **Evidência**:

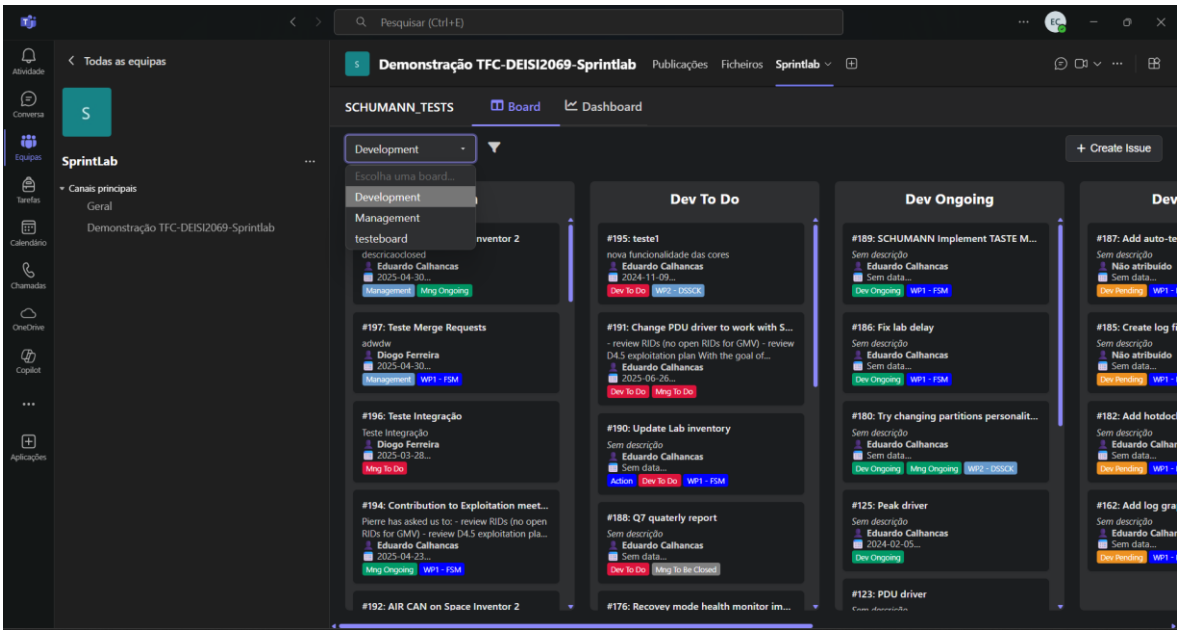


Figura 11 - Teste 3 - 1

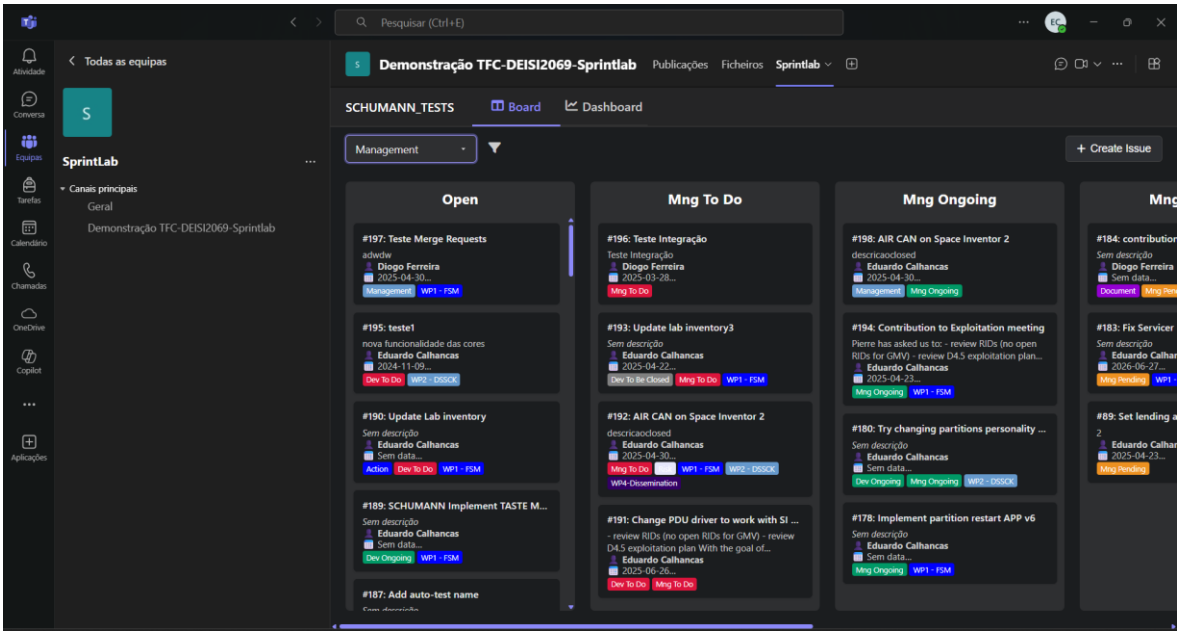


Figura 12 - Teste 3 - 2

Teste T4 - Criar um issue sem labels

- **Objetivo**:
  Verificar se é possível criar um issue sem labels e se este é corretamente colocado na coluna padrão da board (geralmente "Open").
- **Pré-condições**:
    - A board atual está ativa e funcional.
- **Procedimento**:

1. Clicar no botão “+ Novo Issue” no topo ou numa coluna.
2. Preencher o título e descrição do issue.
3. Não selecionar nenhuma label.
4. Clicar em “Criar”.

- **Resultado Esperado**:
  - O issue é criado com sucesso no *Gitlab*.
  - Aparece automaticamente na coluna padrão (sem label associada).
- **Resultado Obtido**: Passou - O issue sem labels foi criado e alocado corretamente.
- **Evidência**:

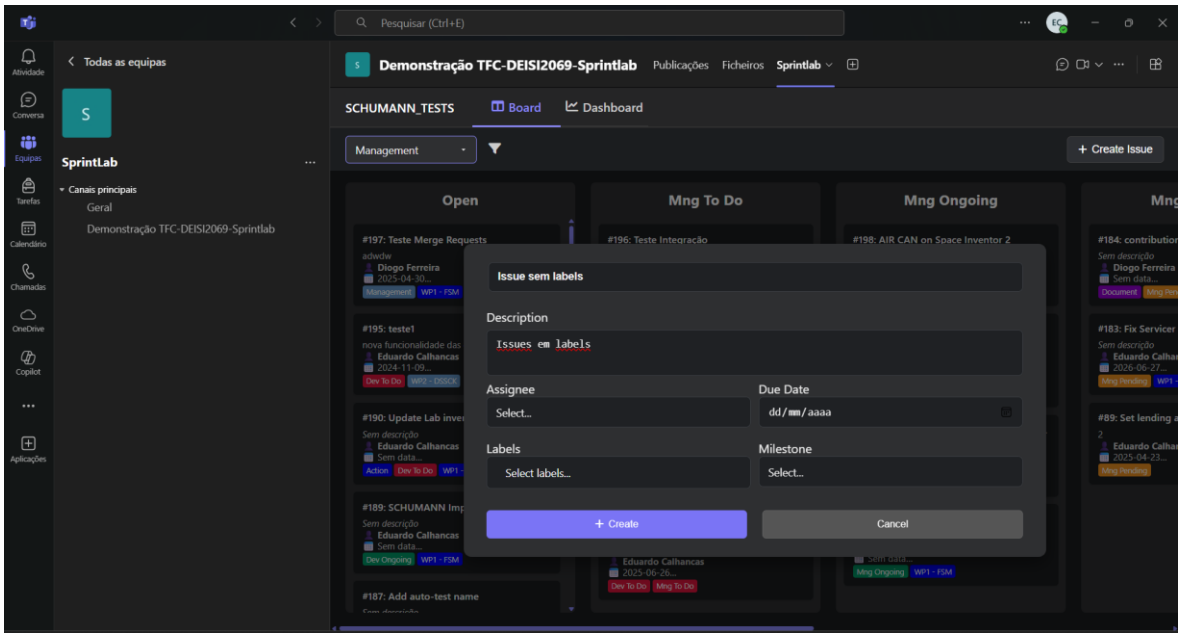


Figura 13 - Teste 4 - 1

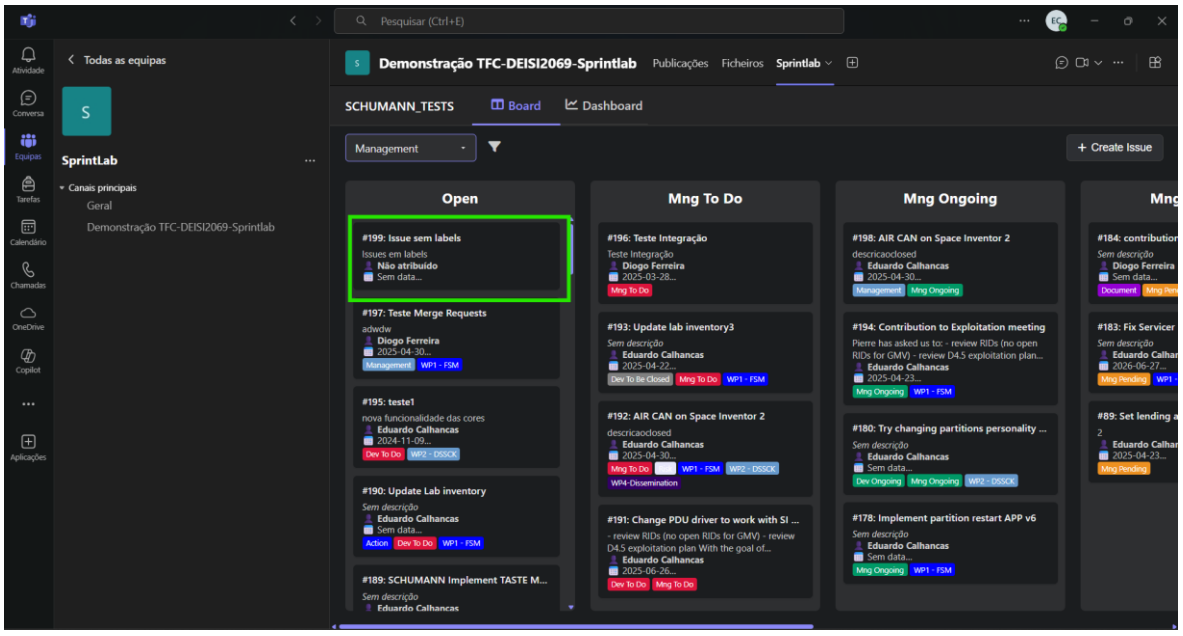


Figura 14 - Teste 4 - 2

Teste T5 - Editar o título do issue

- **Objetivo**: Verificar se a edição do título de um issue no modal de edição é corretamente refletida no *Gitlab* e na board.

- **Pré-condições**:
  - Existe pelo menos um issue visível na board.
- **Procedimento**:
  1. Clicar num issue da board para abrir o modal de edição.
  2. Modificar o título do issue no campo respetivo.
  3. Guardar as alterações clicando no botão de atualização.
  4. Observar a atualização imediata na board.
- **Resultado Esperado**:
  - O novo título é guardado no *Gitlab*.
  - O título atualizado aparece automaticamente na board sem necessidade de refresh.
- **Resultado Obtido**:
  Passou - A alteração do título foi sincronizada corretamente.
- **Evidência**:

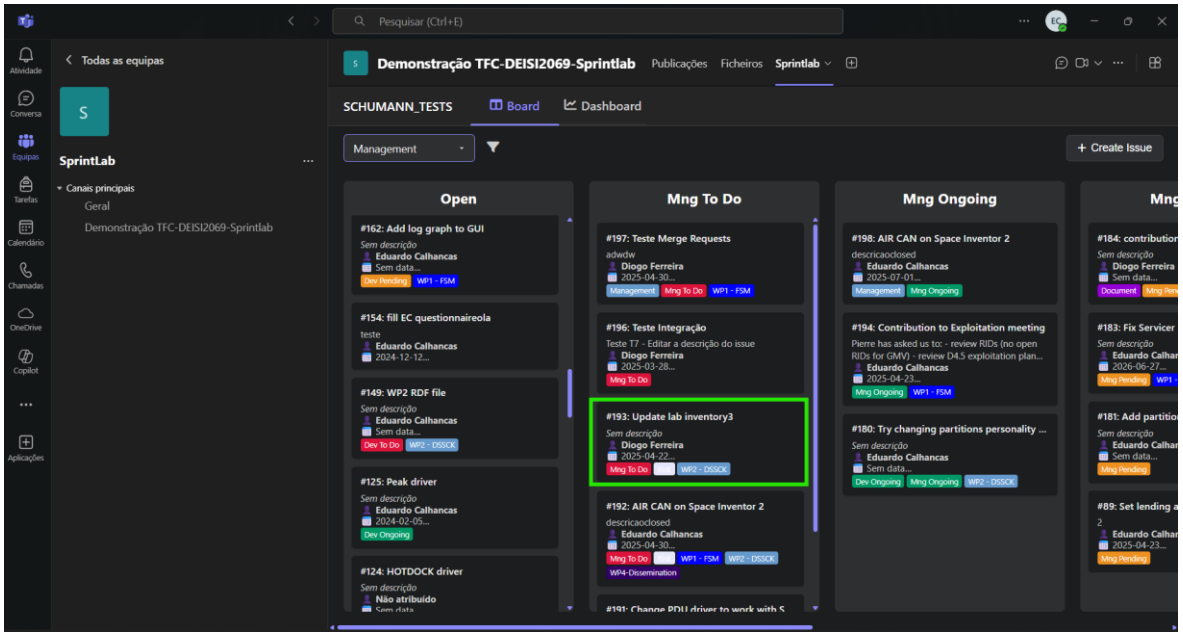


Figura 15 - Teste 5 - 1

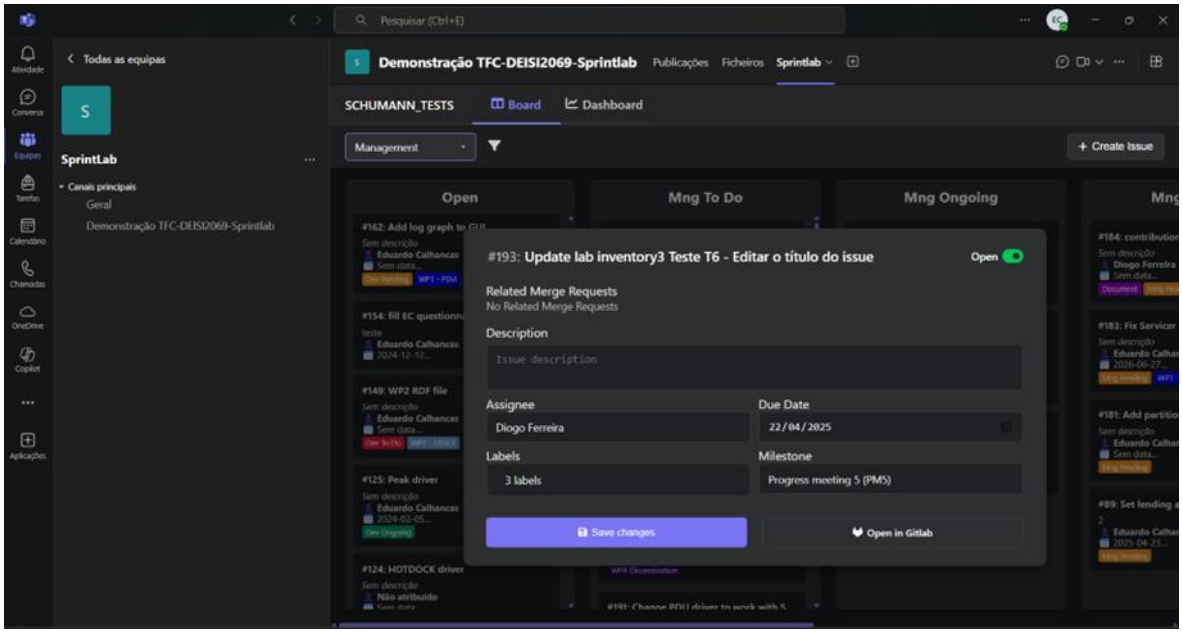


Figura 16 - Teste 5 - 2

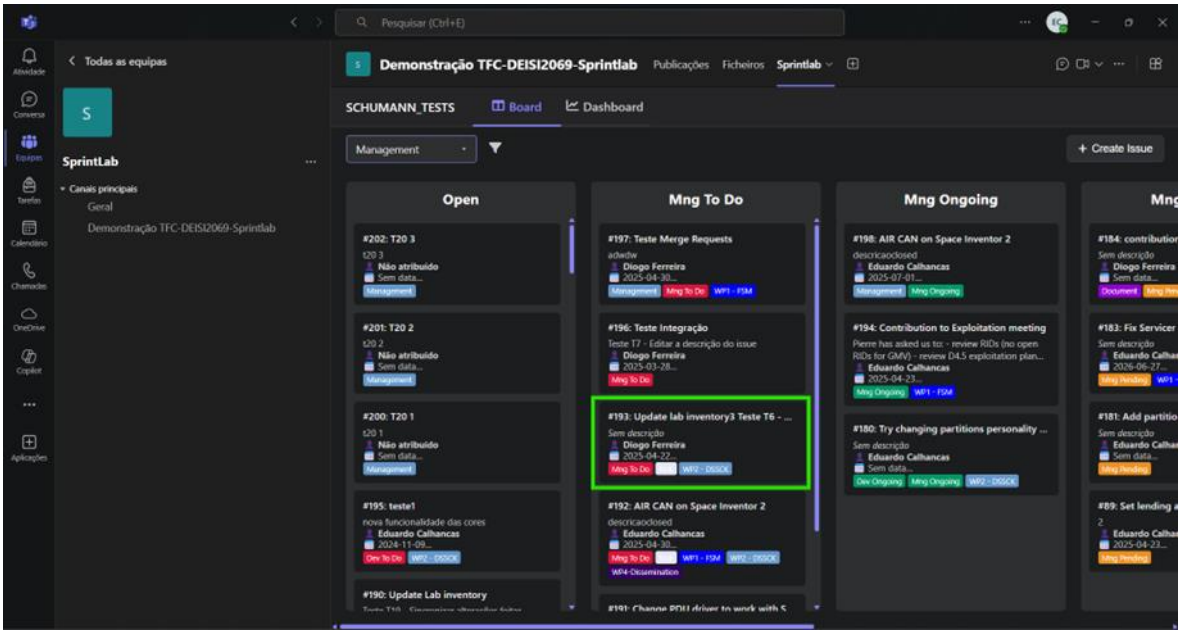


Figura 17 - Teste 5 - 3

Teste T6 - Editar a descrição do issue

- **Objetivo**: Validar se a atualização da descrição de um issue é guardada corretamente e exibida tanto no *Gitlab* como na interface da board.
- **Pré-condições**:
  - Existe um issue com ou sem descrição associada.
- **Procedimento**:
  1. Clicar num issue da board para abrir o modal de edição.
  2. Modificar o campo da descrição.
  3. Clicar no botão de guardar ou atualizar.
  4. Confirmar a presença da nova descrição no modal.
- **Resultado Esperado**:
  - A nova descrição é gravada no *Gitlab*.
  - Fica visível imediatamente no modal e pode ser lida no *Gitlab*.
- **Resultado Obtido**:
  Passou - A descrição foi atualizada e sincronizada corretamente.
- **Evidência**:

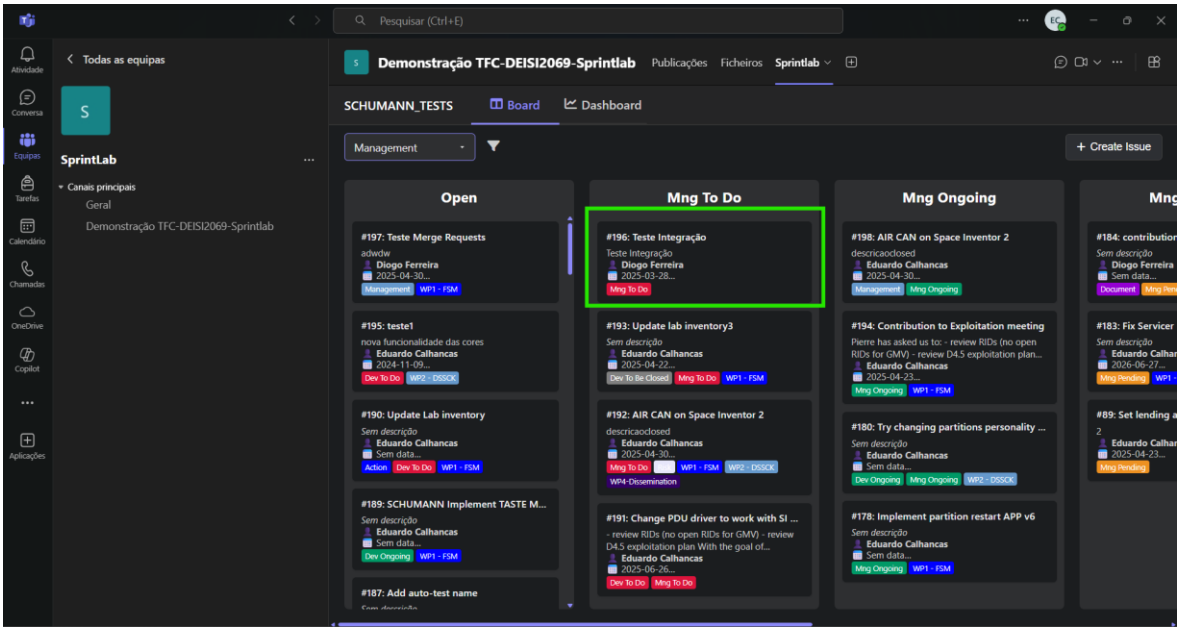


Figura 18 - Teste 6 - 1

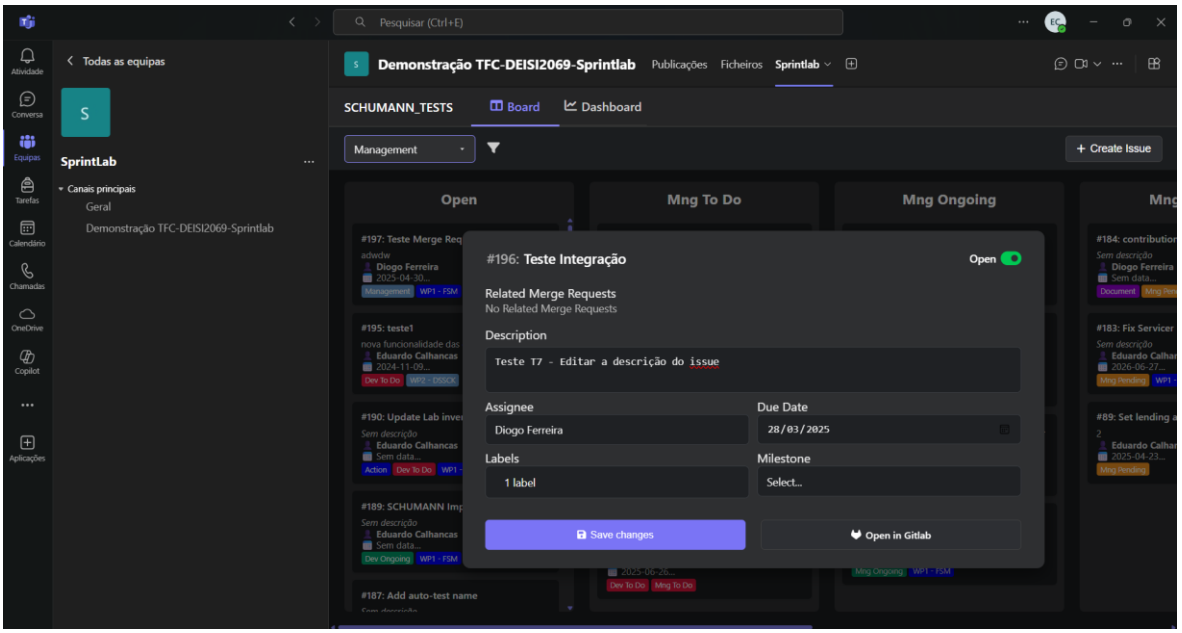


Figura 19 - Teste 6 - 2

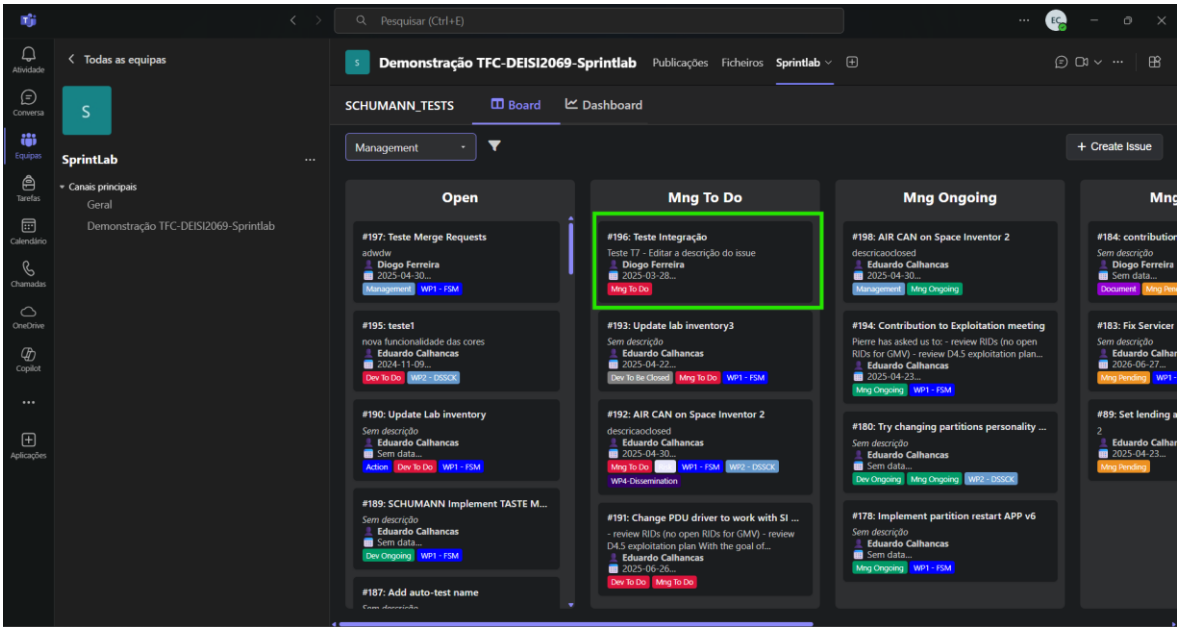


Figura 20 - Teste 6 - 3

Teste T7 - Editar o assignee do issue

- **Objetivo**: Verificar se é possível adicionar ou alterar o assignee de um issue e se esta alteração é refletida corretamente.
- **Pré-condições**:
  - Existe pelo menos um issue na board.
- **Procedimento**:
  1. Abrir o modal de edição de um issue.
  2. Selecionar um assignee diferente no campo correspondente.
  3. Confirmar a atualização.
  4. Observar a alteração no modal ou badge do issue.
- **Resultado Esperado**:
  - O assignee é atualizado corretamente no *Gitlab*.
  - A nova atribuição é visível no modal de edição e na representação do issue na board.
- **Resultado Obtido**:
  Passou - A alteração foi aplicada com sucesso.
- **Evidência**:

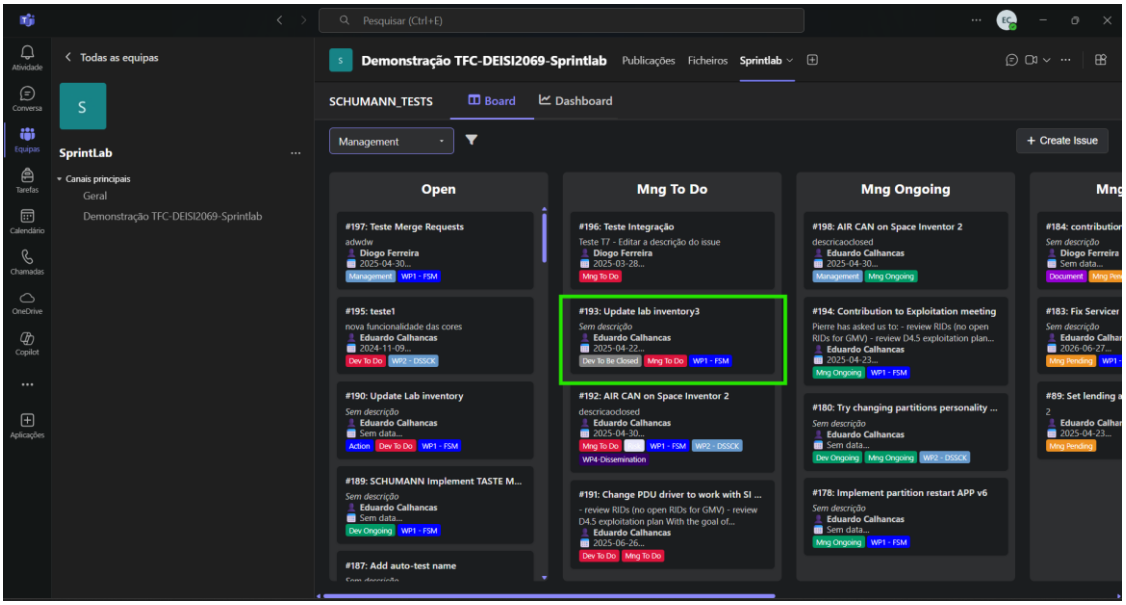


Figura 21 - Teste 7 - 1

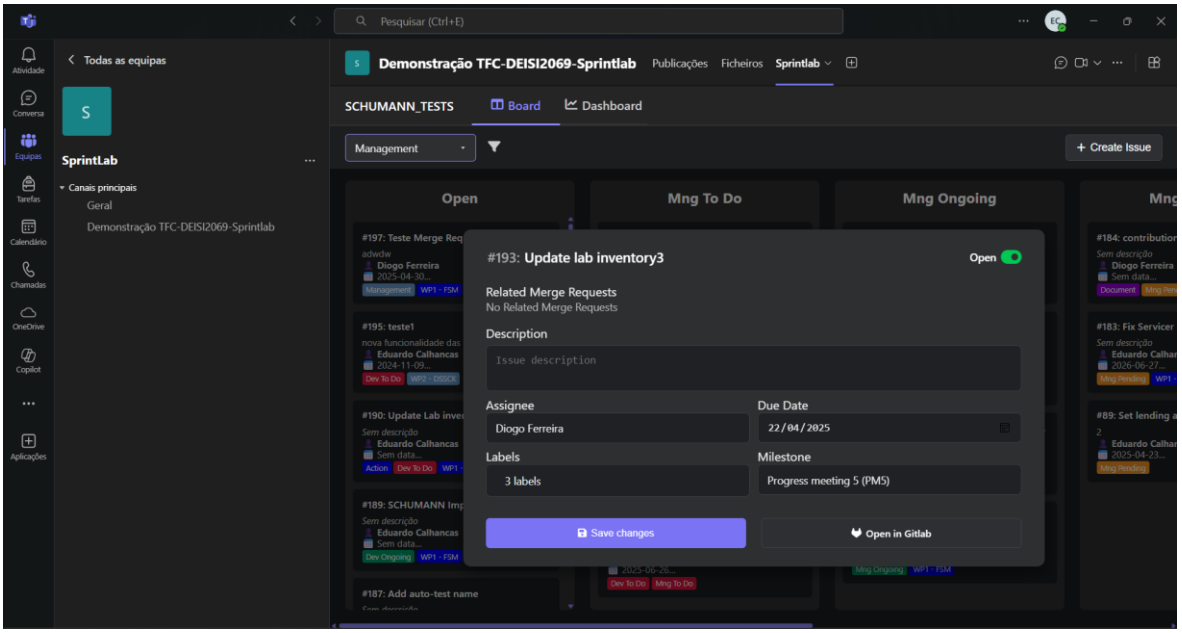


Figura 22 - Teste 7 - 2

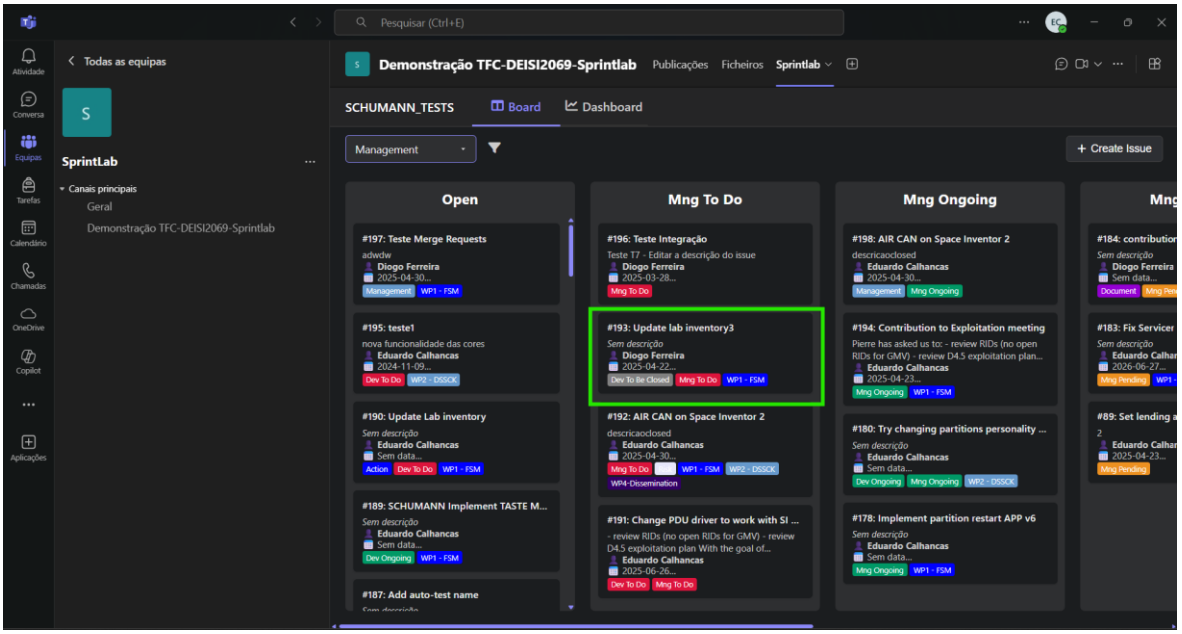


Figura 23 - Teste 7 – 3

Teste T8 - Editar a due date do issue

- **Objetivo**: Verificar se a data de entrega de um issue pode ser adicionada ou modificada com sucesso e refletida na interface e no *Gitlab*.
- **Pré-condições**:
  - Existe um issue disponível na board.
- **Procedimento**:
  1. Abrir o modal de edição de um issue.
  2. No campo “Due Date”, selecionar uma nova data de entrega.
  3. Guardar as alterações.
  4. Confirmar visualmente a data atualizada no modal e na board (se aplicável).
- **Resultado Esperado**:
  - A nova due date é gravada no *Gitlab*.
  - A data aparece corretamente visível no modal e, se implementado, junto ao cartão do issue na board.
- **Resultado Obtido**:
  Passou - A data foi registada corretamente e sincronizada.
- **Evidência**:

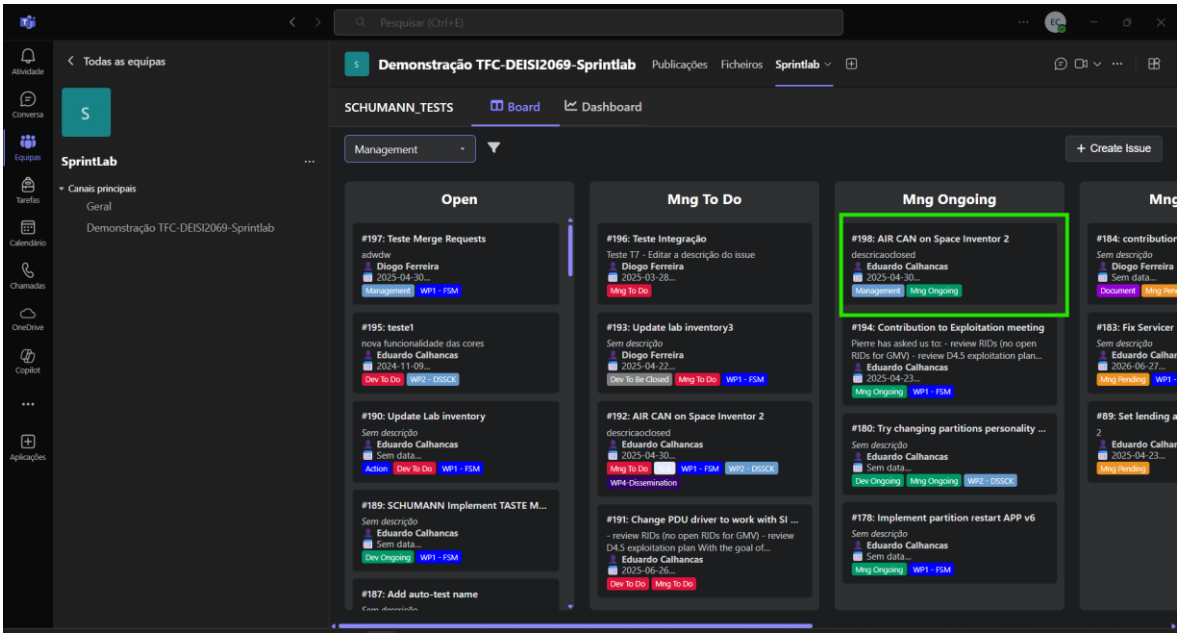


Figura 24 - Teste 8 - 1

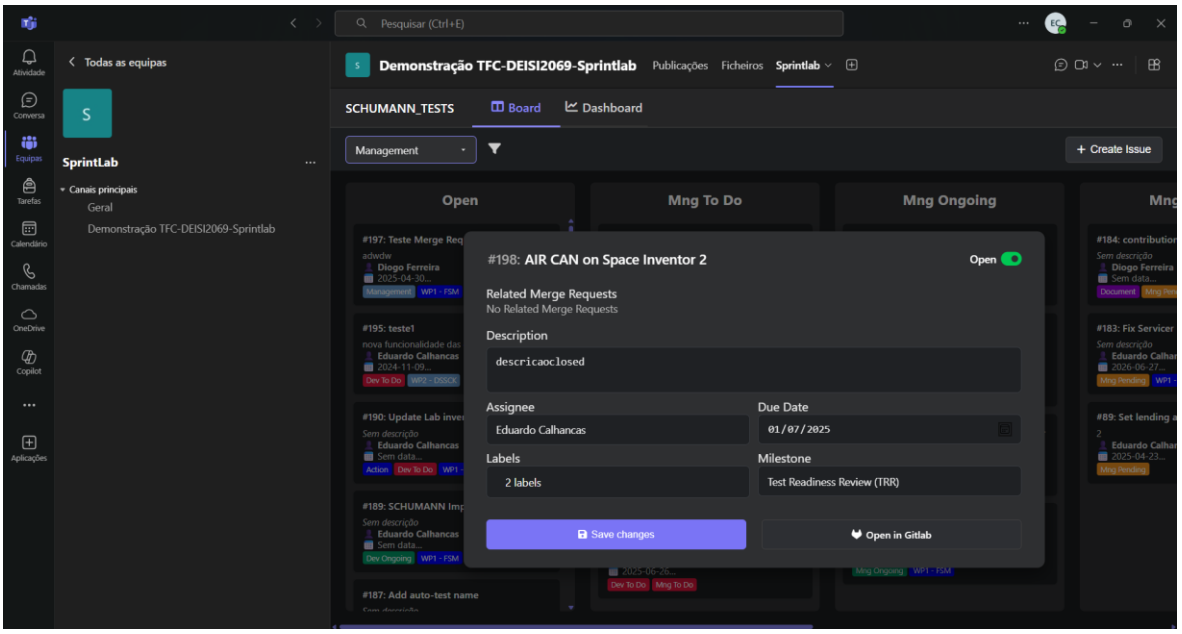


Figura 25 - Teste 8 - 2

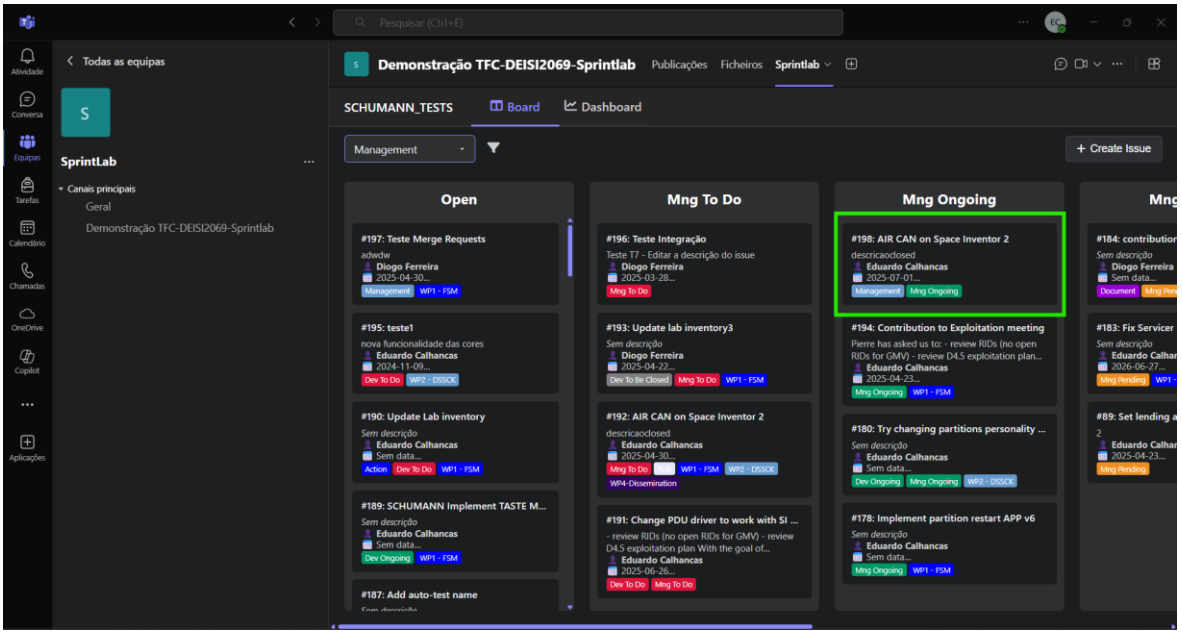


Figura 26 - Teste 8 - 3

Teste T9 - Editar as labels do issue

- **Objetivo**: Validar se é possível adicionar ou remover labels de um issue, com impacto imediato na organização por colunas.
- **Pré-condições**:
  - As colunas da board estão mapeadas a labels *Gitlab*.
  - Existe um issue visível com ou sem labels atribuídas.
- **Procedimento**:
  1. Abrir o modal de edição de um issue.
  2. Adicionar ou remover uma ou mais labels.
  3. Guardar as alterações.
  4. Verificar se o issue muda de coluna (se aplicável).
- **Resultado Esperado**:
  - As labels são atualizadas corretamente no *Gitlab*.
  - O issue muda de coluna na board se as labels forem associadas a outras colunas.
- **Resultado Obtido**:
  Passou - A atualização das labels refletiu-se corretamente na interface e estrutura da board.
- **Evidência**:

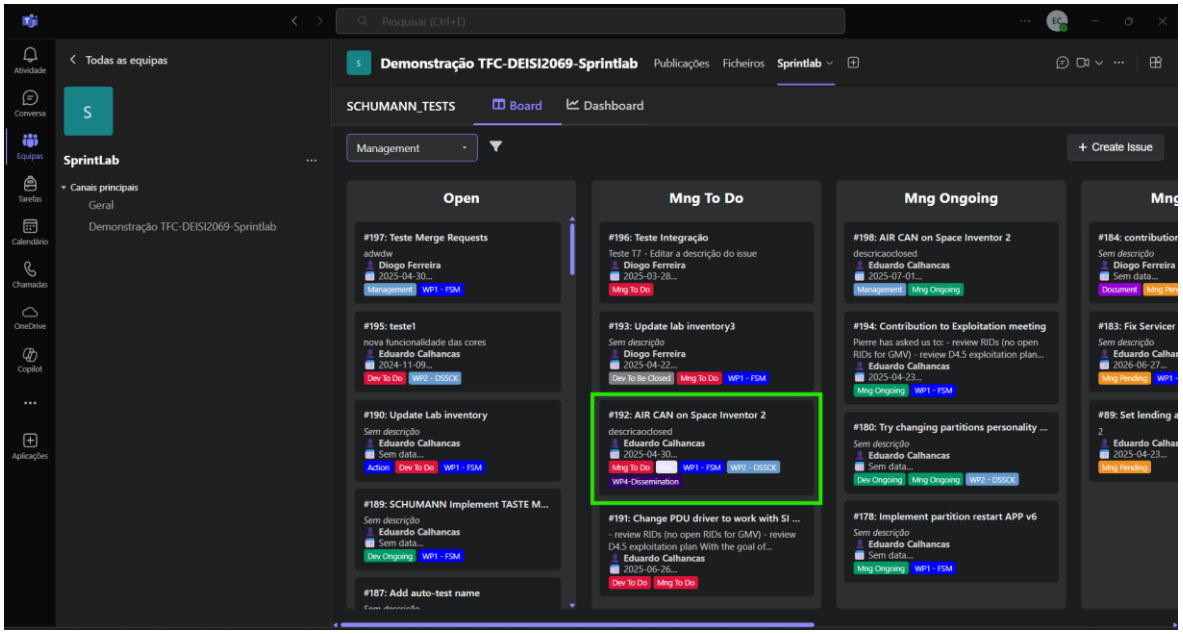


Figura 27 - Teste 9 - 1

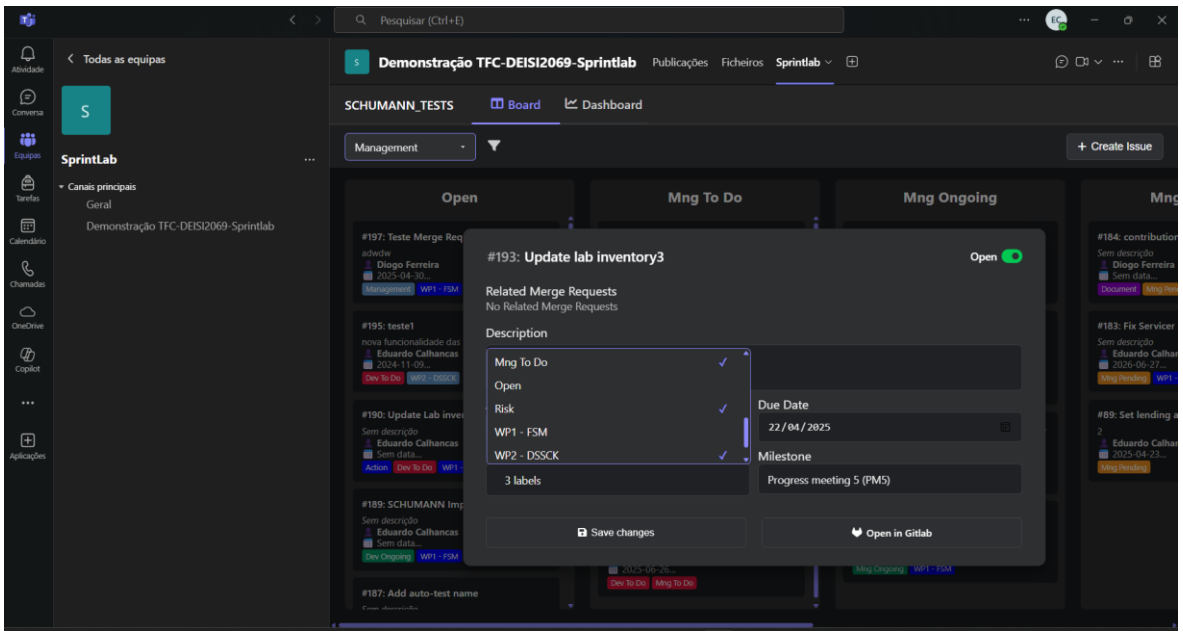


Figura 28 - Teste 9 - 2

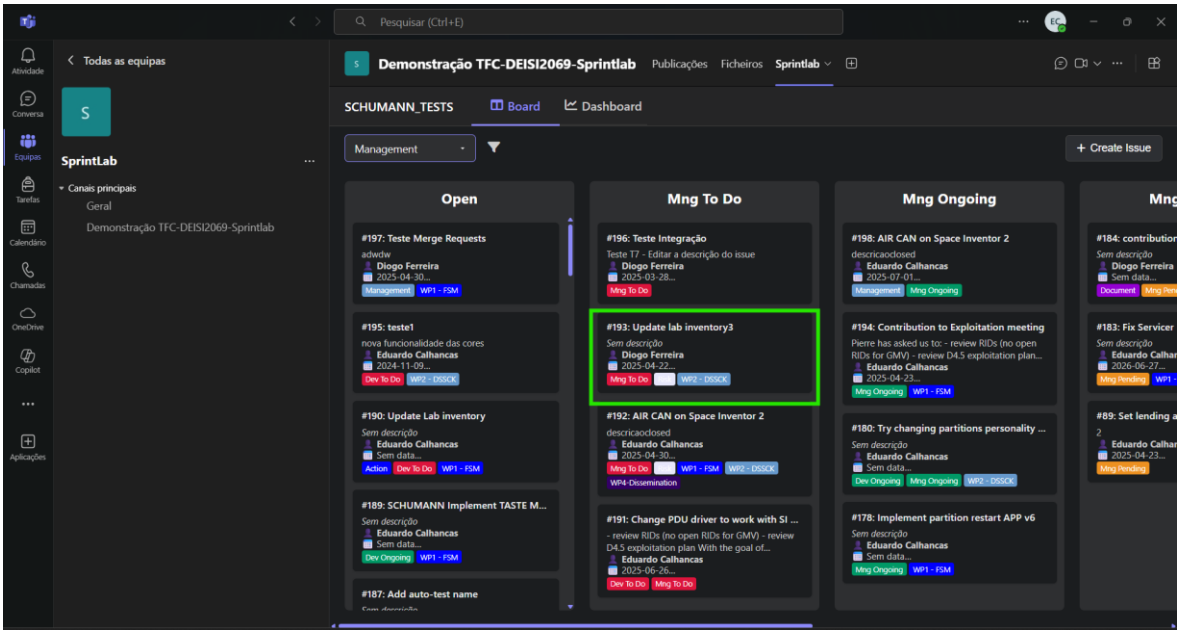


Figura 29 - Teste 9 - 3

Teste T10 - Editar a milestone do issue

- **Objetivo**: Verificar se a atribuição ou alteração da milestone num issue é registada com sucesso e visível na interface.
- **Pré-condições**:
  - Existem milestones ativas configuradas no projeto *Gitlab*.
  - Existe um issue acessível na board.
- **Procedimento**:
  1. Abrir o modal de edição de um issue.
  2. Selecionar uma milestone no campo respetivo.
  3. Guardar a alteração.
  4. Confirmar a nova milestone no modal ou no detalhe do issue.
- **Resultado Esperado**:
  - A milestone é atualizada corretamente no *Gitlab*.
  - A alteração é visível imediatamente no modal.
- **Resultado Obtido**:
  Passou - A milestone foi associada com sucesso ao issue.
- **Evidência**:

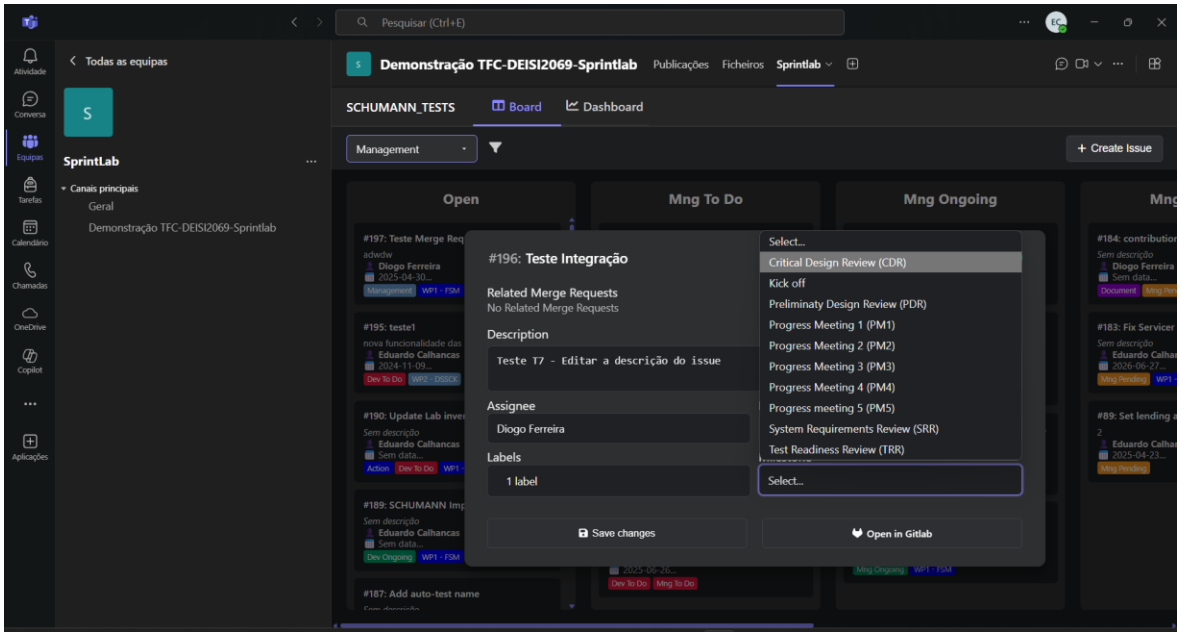


Figura 30 - Teste 10 - 1

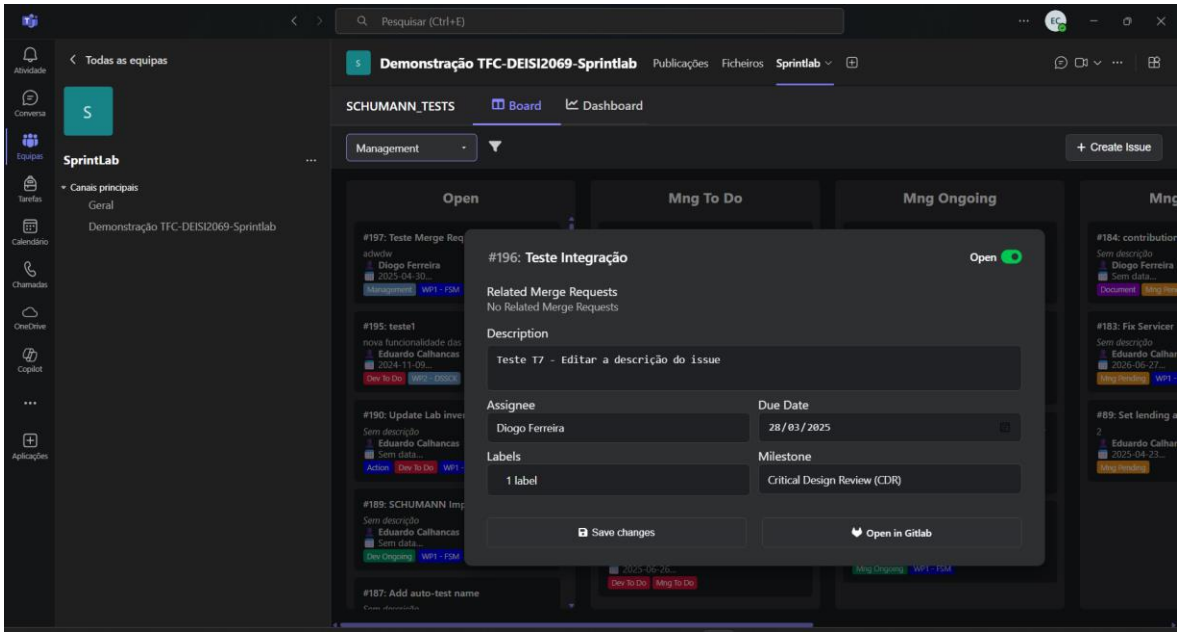


Figura 31 - Teste 10 - 2

Teste T11 - Drag and drop de uma coluna para outra

- **Objetivo**: Verificar se é possível mover um issue entre colunas por arrastamento e se essa ação resulta na atualização das labels no *Gitlab*.
- **Pré-condições**:
  - O projeto tem colunas visíveis associadas a labels distintas.
  - Existe pelo menos um issue presente numa coluna.
- **Procedimento**:
  1. Selecionar um issue numa das colunas visíveis.
  2. Arrastar o issue para outra coluna da board.
  3. Soltar o issue e aguardar confirmação visual da mudança.
  4. Confirmar que a label foi atualizada.
- **Resultado Esperado**:

    - O issue muda de coluna visualmente.
    - A label antiga é substituída pela nova, correspondente à coluna de destino.
- **Resultado Obtido**:
  Passou - A ação de drag and drop foi bem-sucedida e refletida no *Gitlab*.
- **Evidência**:

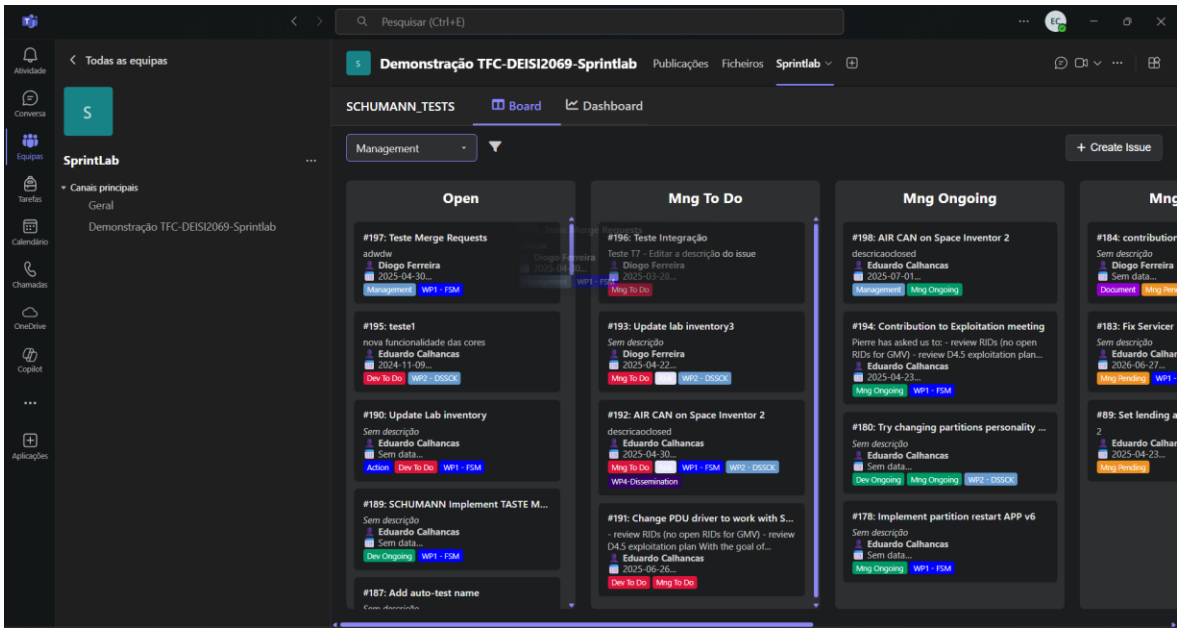


Figura 32 - Teste 11 - 1

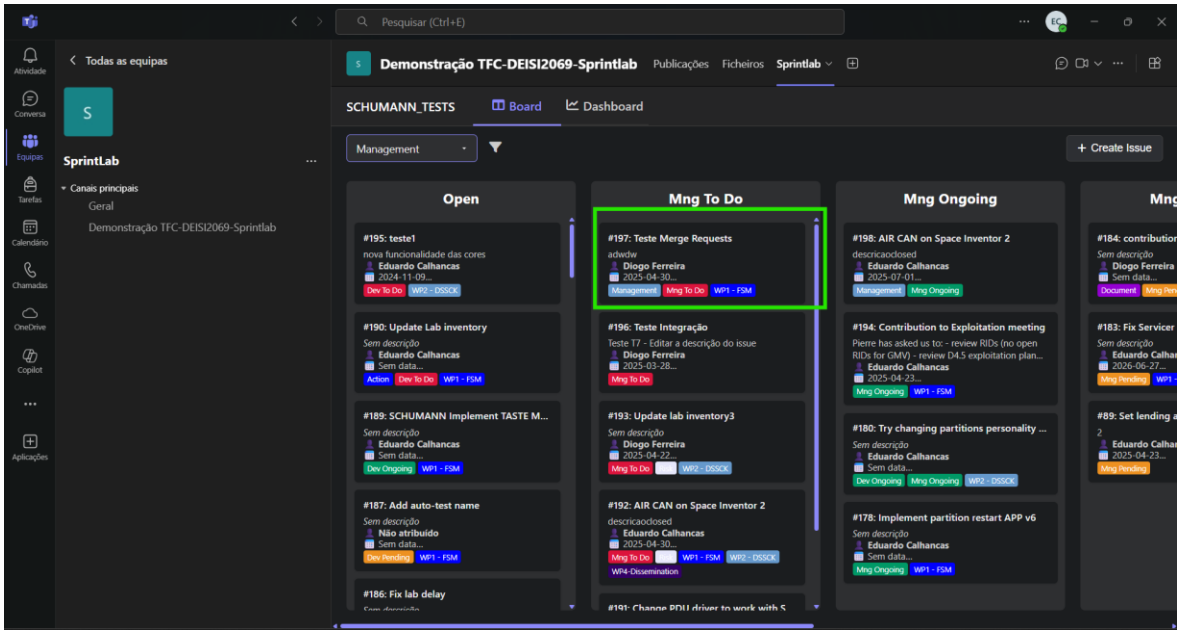


Figura 33 - Teste 11 - 2

Teste T12 — Abrir um issue fechado

- **Objetivo**: Validar se um issue na coluna "Closed" pode ser reaberto corretamente através do toggle no modal de edição.
- **Pré-condições**:
    - Existe pelo menos um issue fechado visível na coluna "Closed".
- **Procedimento**:
    1. Clicar num issue presente na coluna "Closed".
    2. No modal de edição, ativar o toggle para mudar o estado para “Open”.

3. Guardar a alteração e fechar o modal.
4. Verificar a movimentação do issue para a coluna correspondente ao seu estado aberto.

- **Resultado Esperado**:
  - O estado do issue muda de "closed" para "opened" no *Gitlab*.
  - O issue desaparece da coluna "Closed" e surge na coluna certa conforme as labels.
- **Resultado Obtido**:
  Passou - O issue foi reaberto com sucesso.
- **Evidência**:

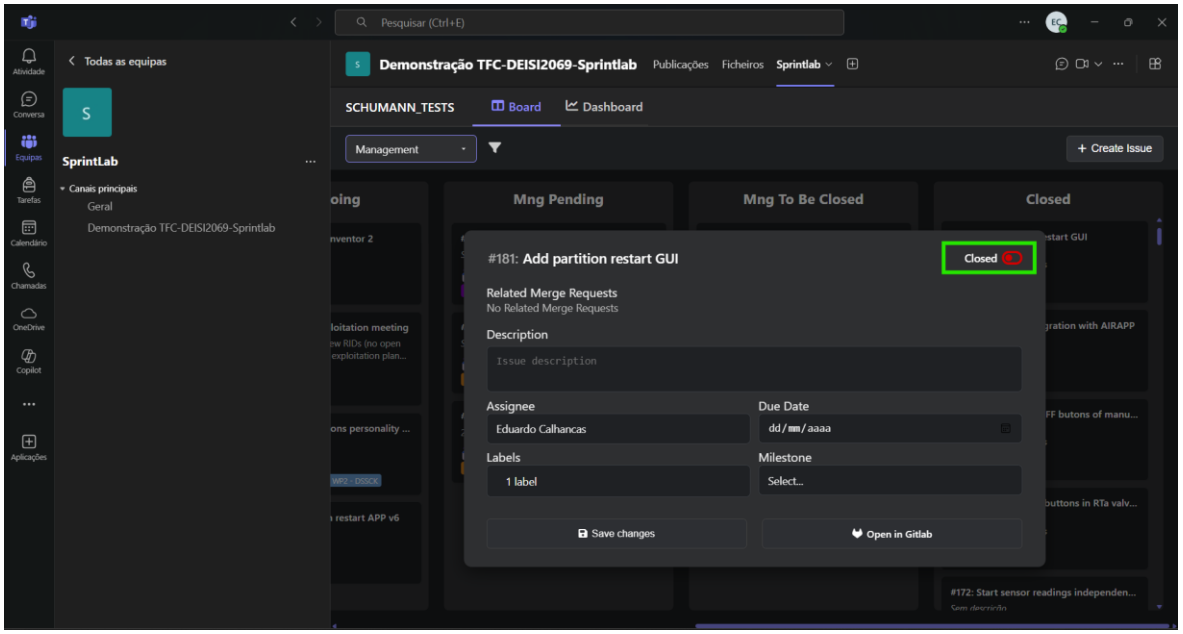


Figura 34 - Teste 12 - 1

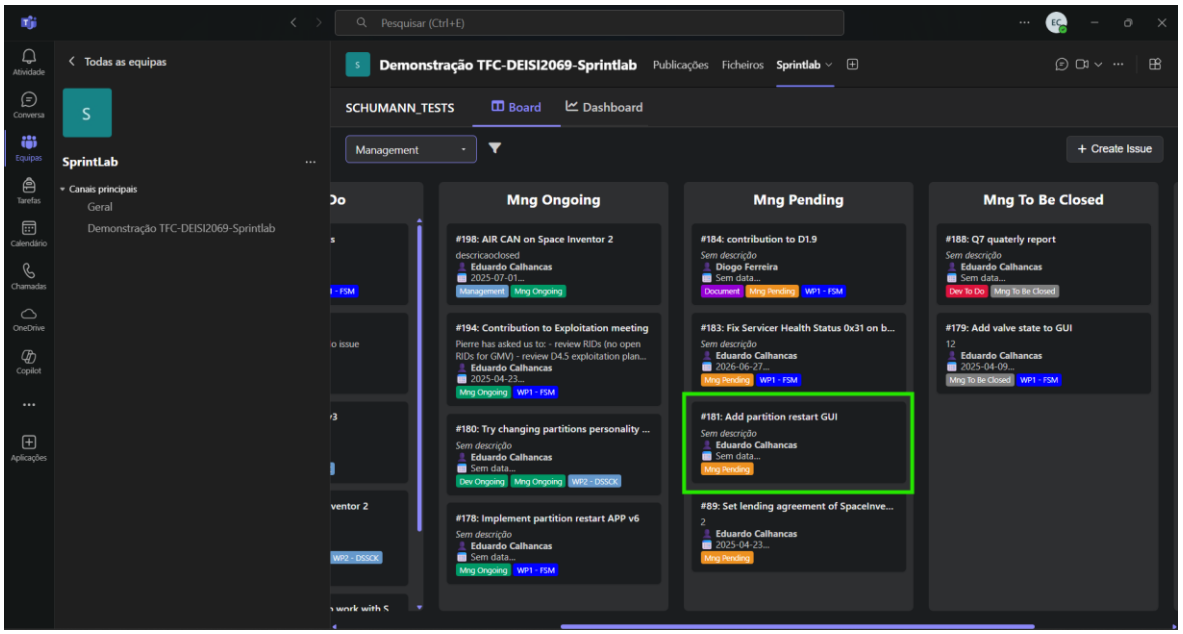


Figura 35 - Teste 12 - 2

Teste T13 - Fechar um issue aberto

- **Objetivo**: Verificar se um issue ativo pode ser fechado diretamente a partir do modal de edição, e se a mudança é refletida na board e no *Gitlab*.

- **Pré-condições**:
  - Existe um issue aberto visível na board.
- **Procedimento**:
  1. Clicar no issue para abrir o modal de edição.
  2. Ativar o toggle de estado para marcar o issue como "Closed".
  3. Guardar a alteração.
  4. Verificar se o issue desaparece da coluna atual e é movido para "Closed".
- **Resultado Esperado**:
  - O issue passa a ter o estado "closed" no *Gitlab*.
  - É movido visualmente para a coluna "Closed" no SprintLab.
- **Resultado Obtido**:
  Passou - O encerramento foi aplicado corretamente.
- **Evidência**:

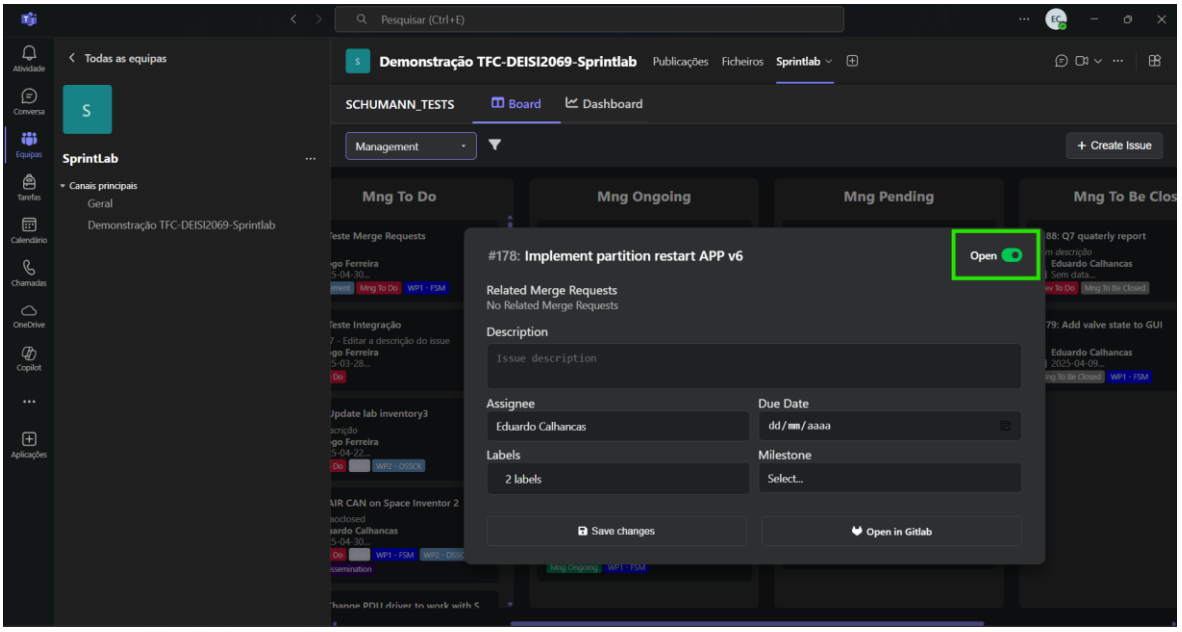


Figura 36 - Teste 13 - 1

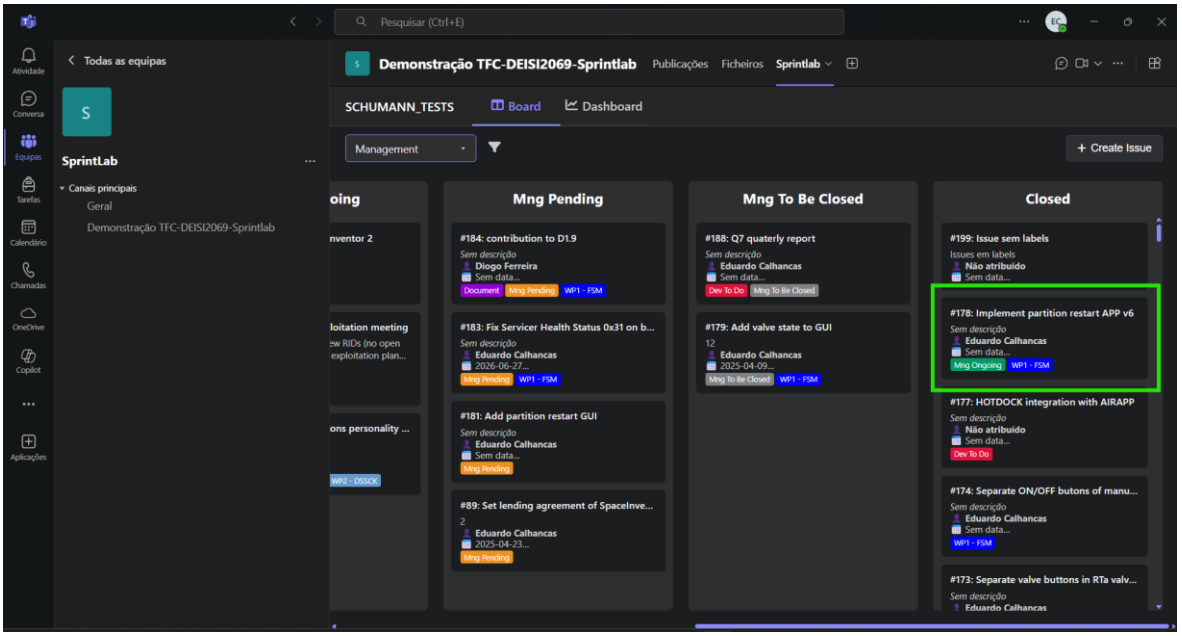


Figura 37 - Teste 13 - 2

Teste T14 - Filtrar issues por labels

- **Objetivo**: Verificar se a funcionalidade de filtro por labels permite visualizar apenas os issues correspondentes à label selecionada.
- **Pré-condições**:
  - Existem múltiplos issues na board com labels distintas.
- **Procedimento**:
  1. Clicar no botão de filtro de issues (ícone de funil ou botão "Filtrar").
  2. Selecionar uma label disponível na lista.
  3. Observar a atualização da board.
- **Resultado Esperado**:
  - Apenas os issues que contêm a label selecionada permanecem visíveis na board.
  - Os restantes issues são temporariamente ocultados da visualização.
- **Resultado Obtido**:
  Passou - O filtro funcionou corretamente e exibiu apenas os issues correspondentes.
- **Evidência**:

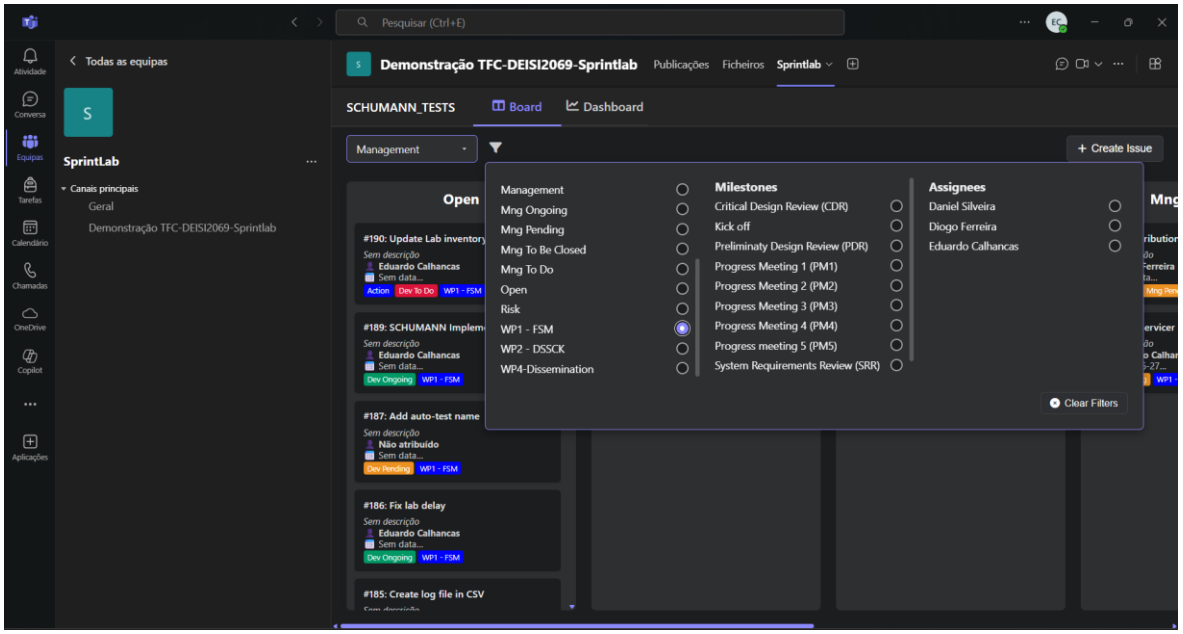


Figura 38 - Teste 14 - 1

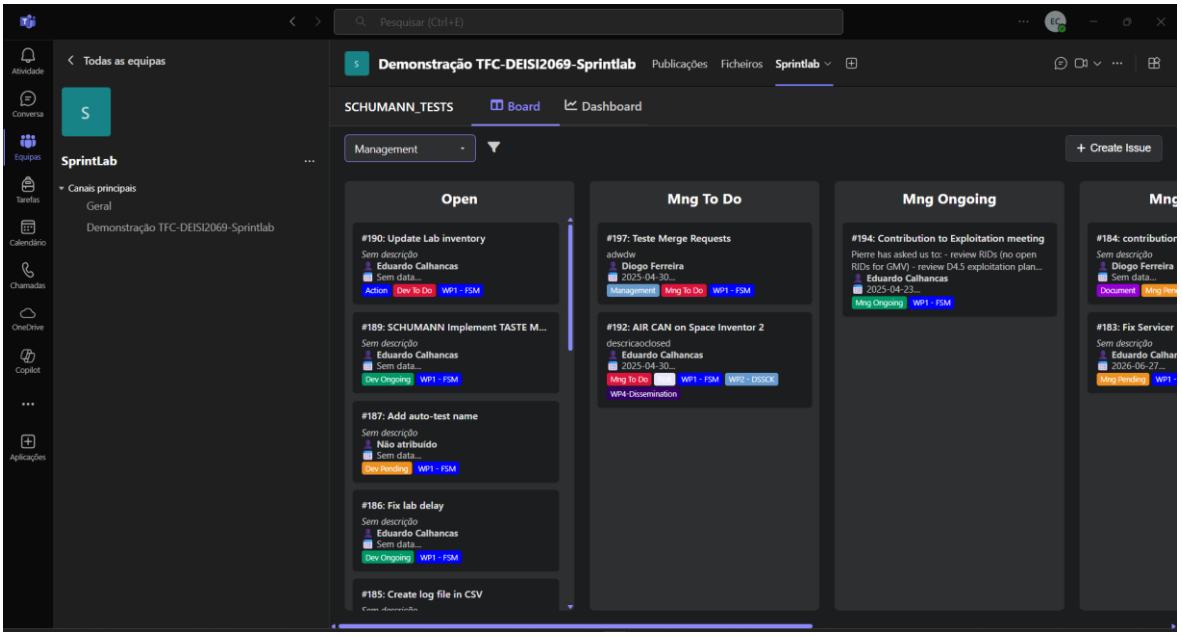


Figura 39 - Teste 14 - 2

Teste T15 - Abrir issue no *Gitlab* através da edição

- **Objetivo**:Confirmar que o botão "Open in *Gitlab*" no modal de edição abre corretamente a página do issue no navegador.
- **Pré-condições**:
  - Existe um issue visível na board.
  - O botão "Open in *Gitlab*" está presente no modal de edição.
- **Procedimento**:
  1. Clicar num issue da board para abrir o modal de edição.
  2. Clicar no botão "Open in *Gitlab*".
  3. Confirmar se uma nova aba do navegador é aberta com o URL do issue no *Gitlab*.
  4.
- **Resultado Esperado**:
  - A página do issue abre corretamente no repositório do *Gitlab*.
- **Resultado Obtido**:
  Passou - O botão abriu corretamente o link do issue no navegador.
- **Evidência**:

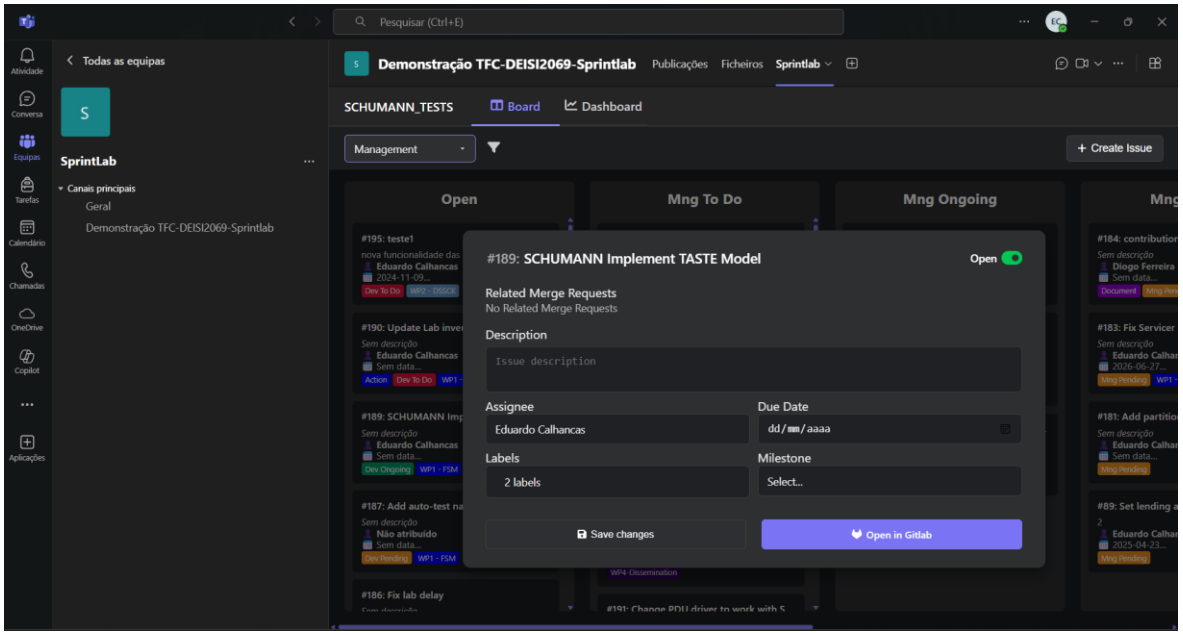


Figura 40 - Teste 15 - 1

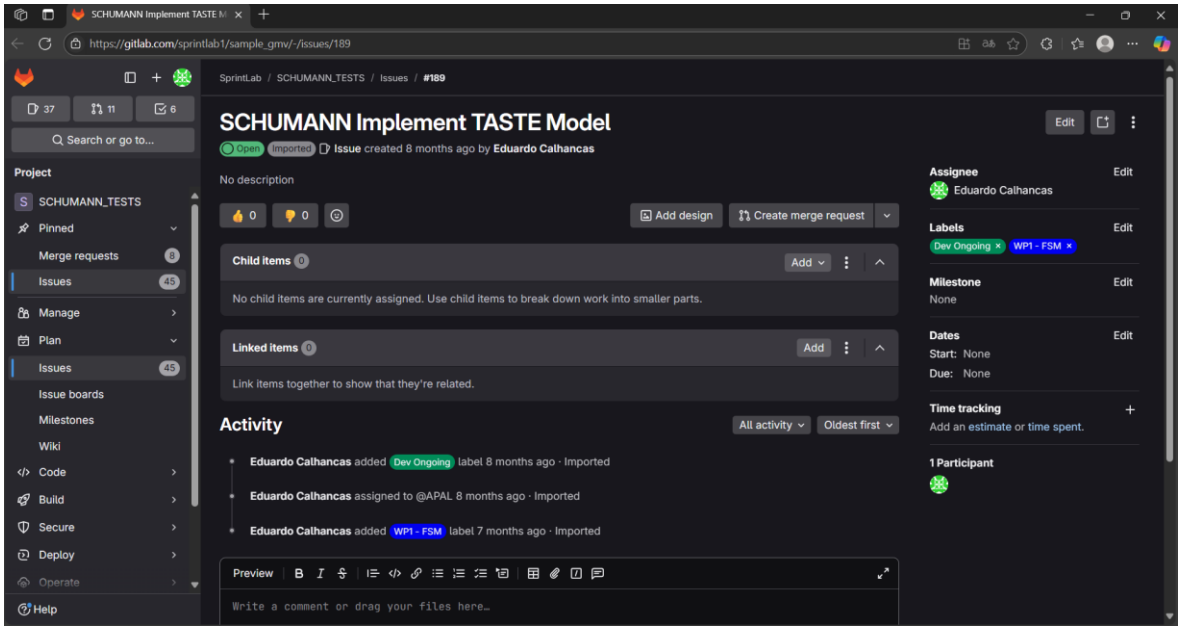


Figura 41 - Teste 15 - 2

Teste T16 - Abrir merge request no *Gitlab*

- **Objetivo**: Verificar se os merge requests relacionados ao issue são corretamente listados e abertos através do modal de edição.
- **Pré-condições**:
  - O issue possui pelo menos um merge request relacionado.
  - O modal de edição apresenta os merge requests associados.
- **Procedimento**:
  1. Abrir o modal de edição de um issue.
  2. Na secção “Merge Requests relacionados”, clicar num dos links disponíveis.
  3. Verificar se o merge request abre corretamente numa nova aba do navegador.
- **Resultado Esperado**:
  - O merge request correspondente abre corretamente no *Gitlab*.
  - O conteúdo está acessível ao utilizador no browser.

- **Resultado Obtido**:
  Passou - O merge request abriu corretamente no navegador.
- **Evidência**:

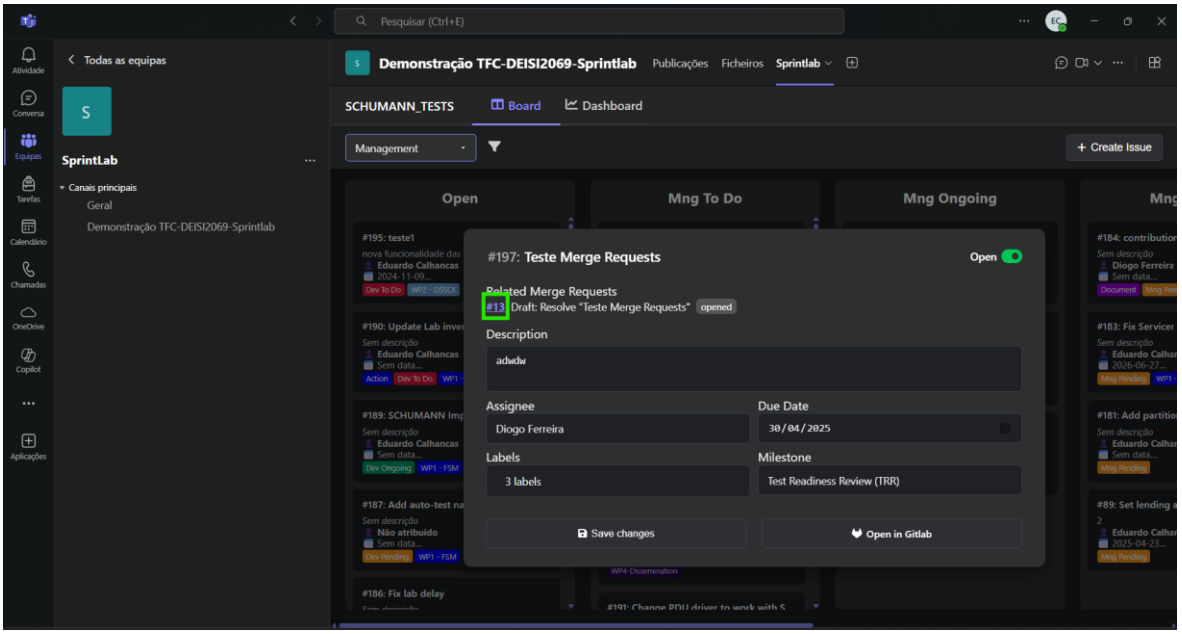


Figura 42 - Teste 16 - 1

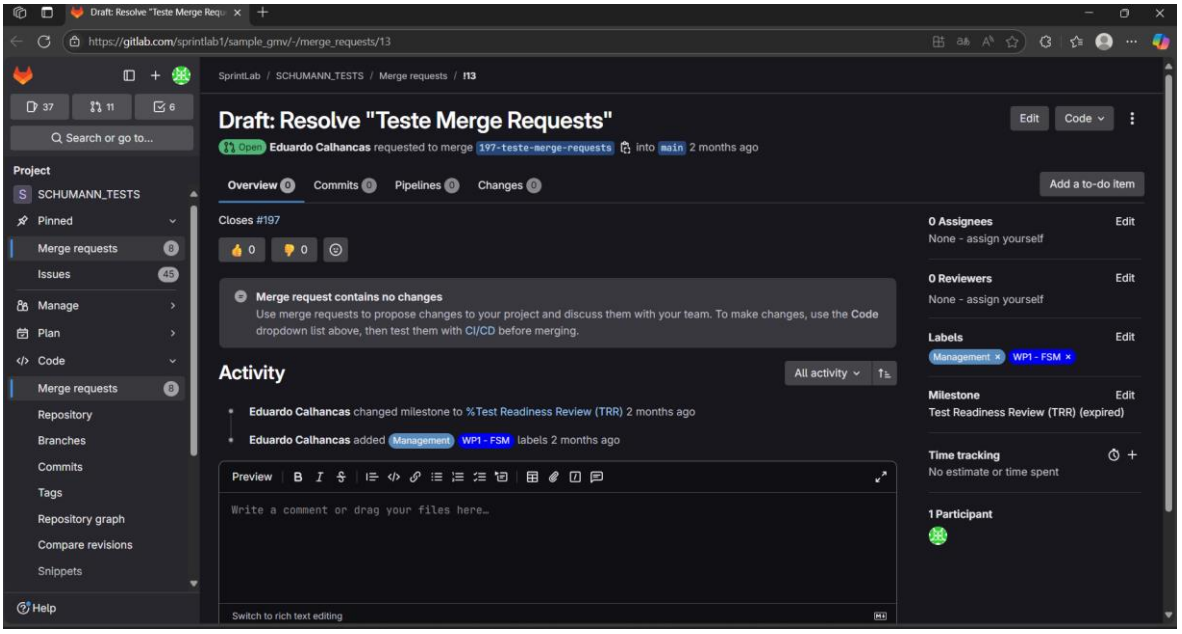


Figura 43 - Teste 16 - 2

Teste T17 - Sincronizar alterações feitas diretamente no *Gitlab*

- **Objetivo**: Confirmar que alterações feitas diretamente no *Gitlab* (fora do SprintLab) são refletidas corretamente na interface do SprintLab após recarregamento.
- **Pré-condições**:
  - O projeto *Gitlab* já está associado ao SprintLab.
  - Existe pelo menos um issue visível na board.
- **Procedimento**:
  1. Aceder ao *Gitlab* e abrir um dos issues existentes.
  2. Alterar o título do issue diretamente no *Gitlab*.

3. Voltar ao SprintLab e recarregar a board
4. Observar se a nova informação do issue é carregada corretamente.

- **Resultado Esperado**:
  - A nova informação (ex.: título) é sincronizada e visível na board do SprintLab.
- **Resultado Obtido**:
  Passou - O SprintLab refletiu a alteração feita diretamente no *Gitlab*.
- **Evidência**:

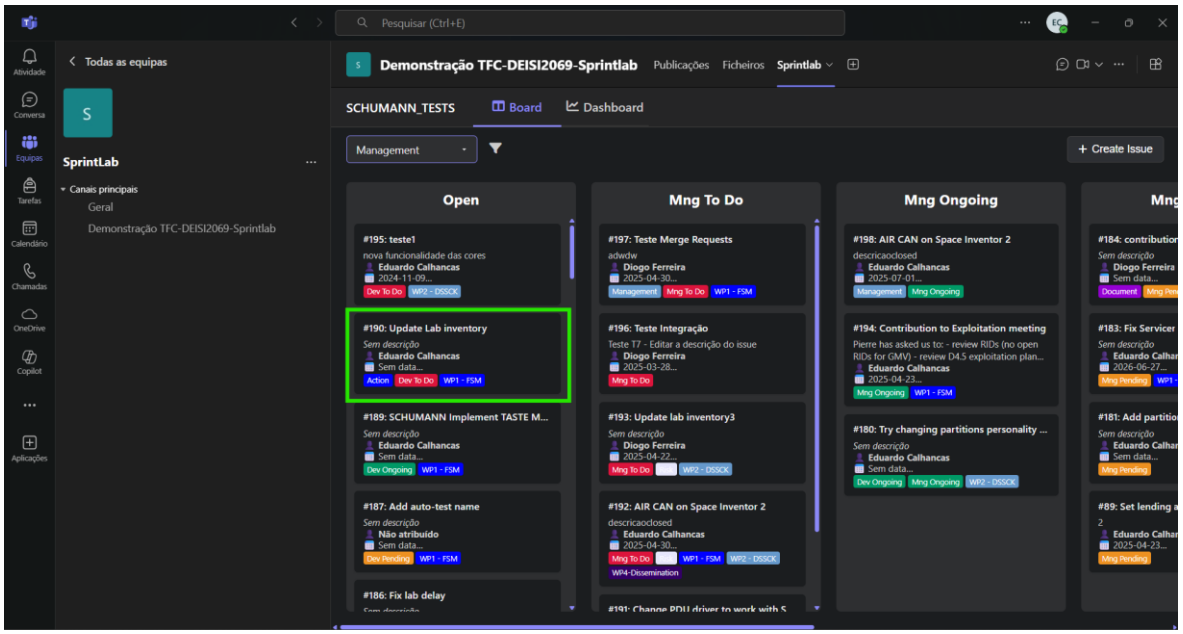


Figura 44 - Teste 17 - 1

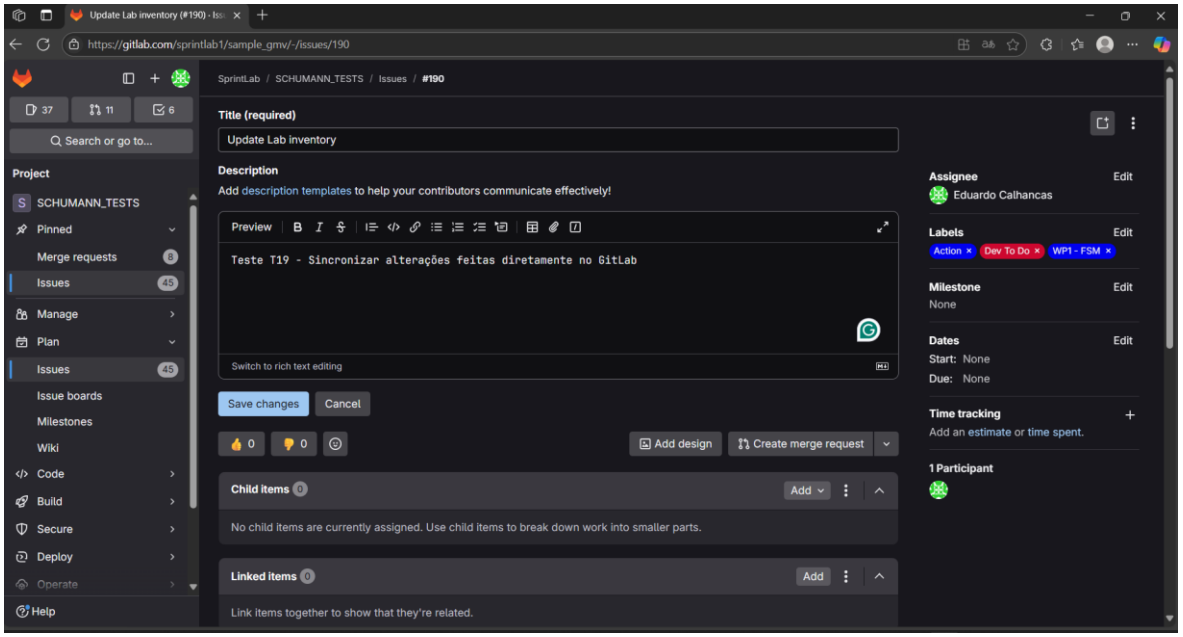


Figura 45 - Teste 17 - 2

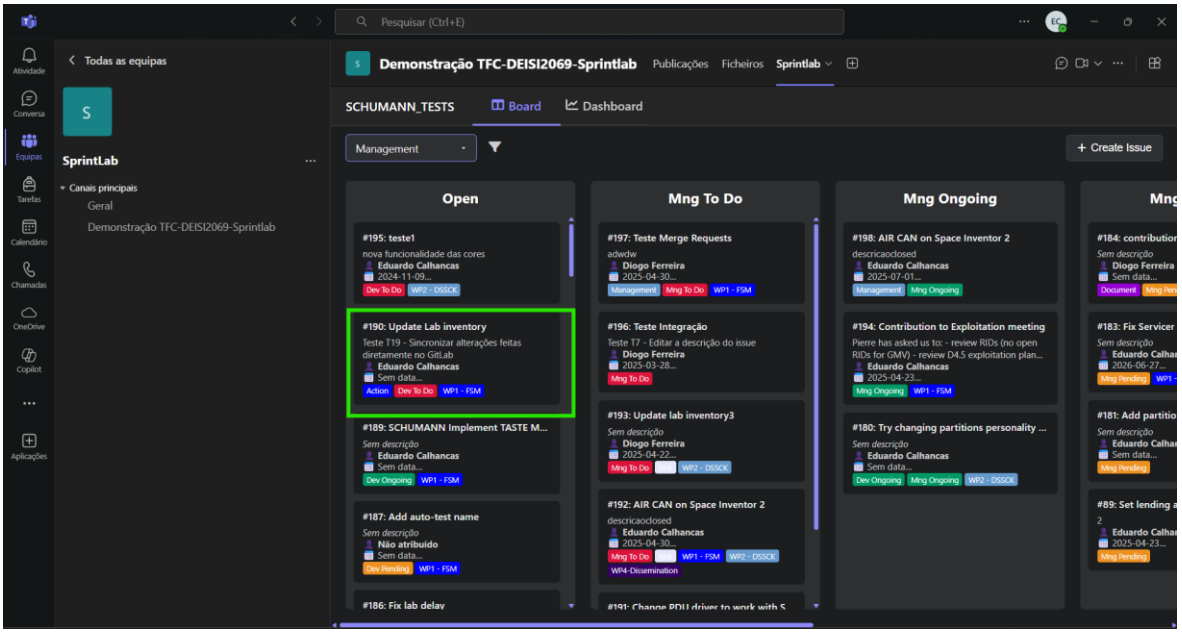


Figura 46 - Teste 17 - 3

Teste T18 - Criar múltiplos issues em simultâneo

- **Objetivo**: Verificar se o SprintLab permite a criação rápida de vários issues seguidos e se todos são corretamente registados no *Gitlab*.
- **Pré-condições**:
  - A board está visível e funcional.
- **Procedimento**:
  1. Clicar no botão de criação de novo issue.
  2. Preencher os dados e criar o primeiro issue.
  3. Repetir rapidamente o processo para criar 2 a 3 issues adicionais.
  4. Confirmar a presença de todos os novos issues na board e no *Gitlab*.
- **Resultado Esperado**:
  - Todos os issues são criados sem erros e aparecem corretamente na interface e no *Gitlab*.
- **Resultado Obtido**:
  Passou - Os múltiplos issues foram criados com sucesso e sem falhas.
- **Evidência**:

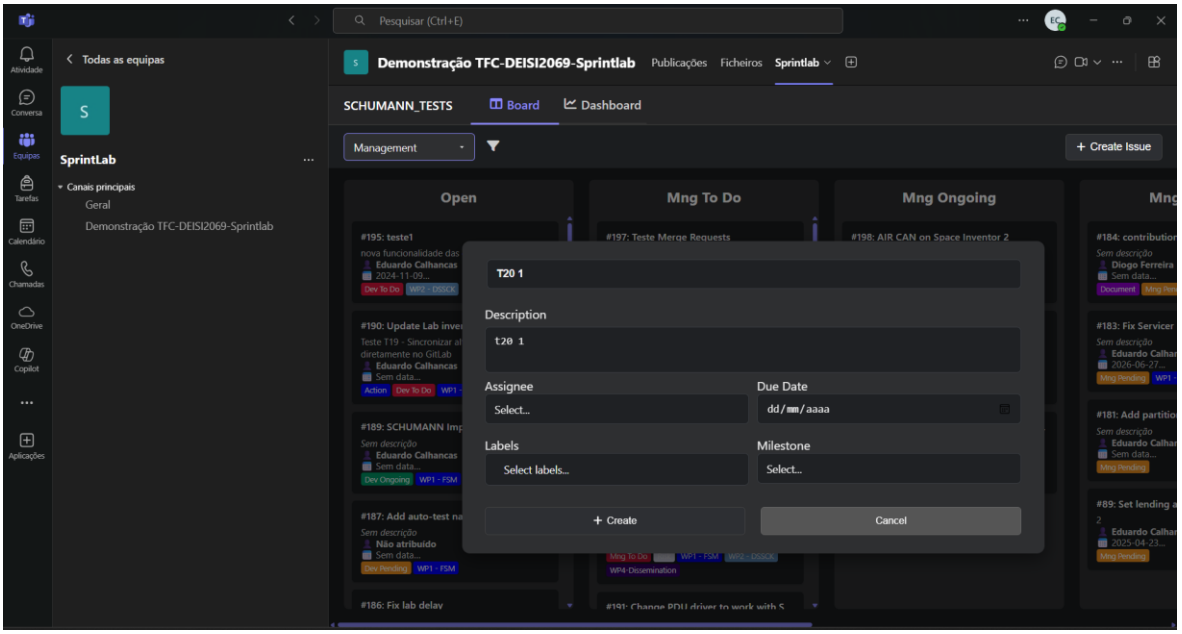


Figura 47 - Teste 18 - 1

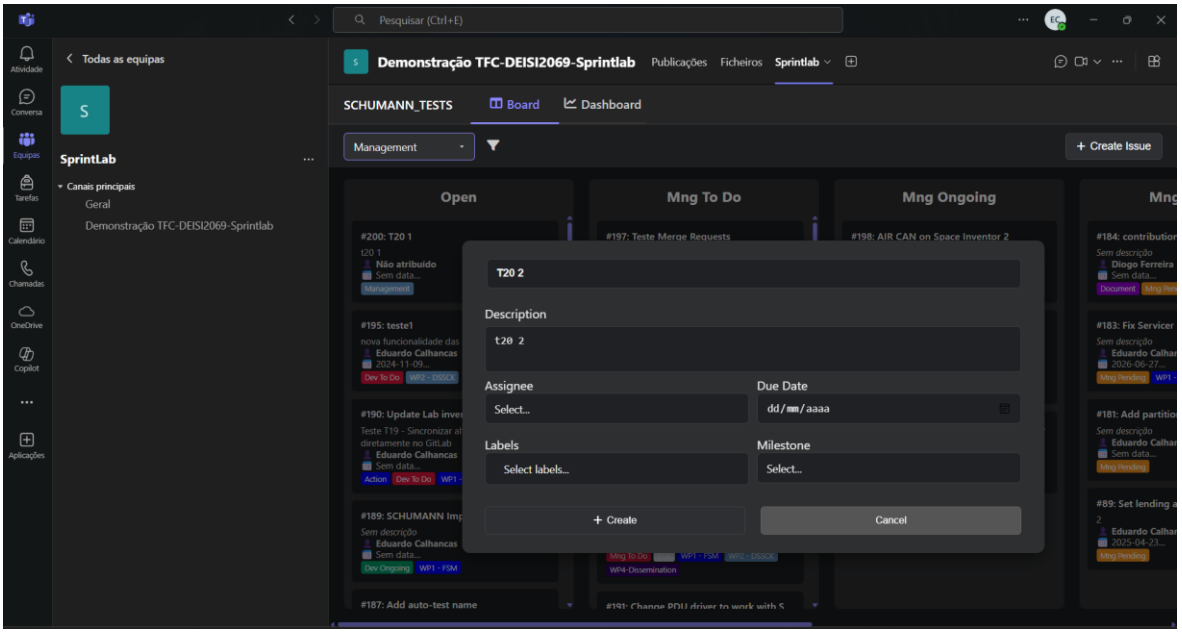


Figura 48 - Teste 18 - 2

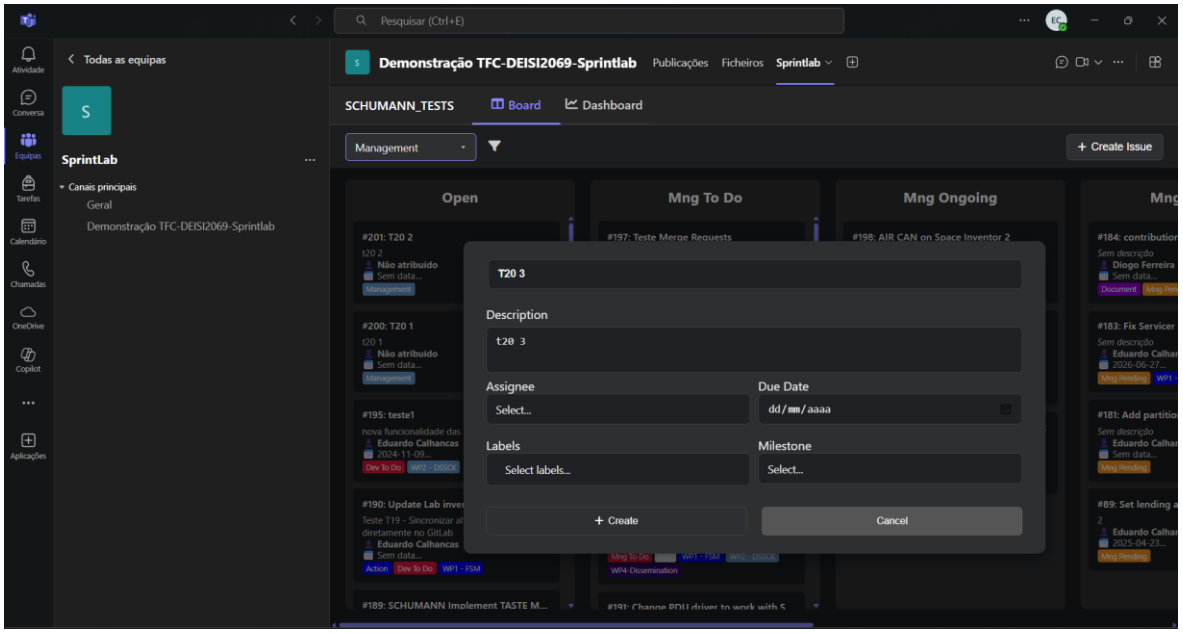


Figura 49 - Teste 18 - 3

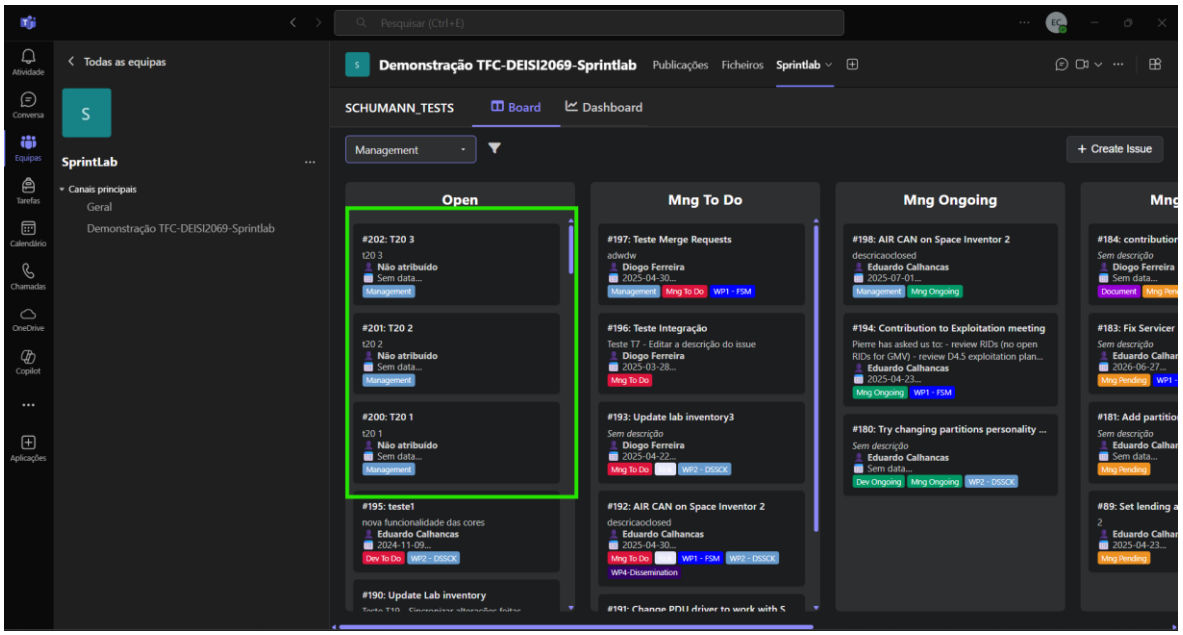


Figura 50 - Teste 18 - 4

Teste T19 - Testar a resposta da API com *token* inválido

- **Objetivo**: Verificar se o SprintLab lida corretamente com *tokens* inválidos, apresentando mensagens de erro claras e prevenindo o carregamento de dados.
- **Pré-condições**:
  - O id do projeto inserido existe no *Gitlab*.
- **Procedimento**:
  1. Aceder ao painel de configuração do projeto no SprintLab.
  2. Adicionar um *token* inválido na configuração (ex: String aleatória ou toke.
  3. Tentar guardar e carregar a board novamente.
- **Resultado Esperado**:
  - A aplicação apresenta a mensagem de erro como "*Token* inserted does not have access to the *Gitlab* project".
  - Nenhuma informação sensível é exibida.

    - A board não é carregada.
- **Resultado Obtido**:
  Passou - O SprintLab bloqueou corretamente o acesso com *token* inválido e apresentou uma mensagem clara.
- **Evidência**:

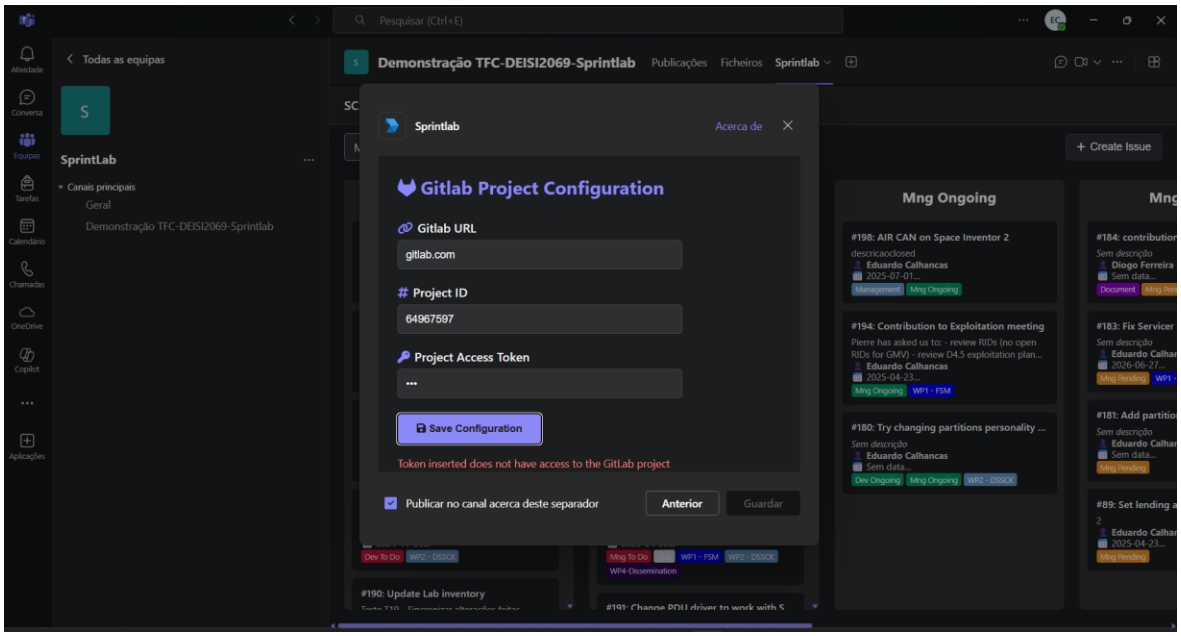


Figura 51 - Teste 19 - 1

Teste T20 - Filtrar Gantt Chart por milestones

- **Objetivo**: Verificar se o filtro de milestones no Dashboard afeta corretamente a visualização do Gantt Chart, exibindo apenas as tarefas da milestone selecionada.
- **Pré-condições**:
    - Existem várias milestones ativas com issues associadas.
    - O Dashboard e o Gantt Chart estão acessíveis.
- **Procedimento**:
    1. Aceder ao Dashboard do SprintLab.
    2. No topo ou junto ao Gantt Chart, abrir o dropdown de milestones.
    3. Selecionar uma milestone específica.
    4. Observar a atualização da linha temporal.
- **Resultado Esperado**:
    - O Gantt Chart exibe apenas os issues associados à milestone escolhida.
    - Os restantes issues são ocultados da visualização temporal.
- **Resultado Obtido**:
  Passou - A filtragem por milestone foi aplicada corretamente.
- **Evidência**:

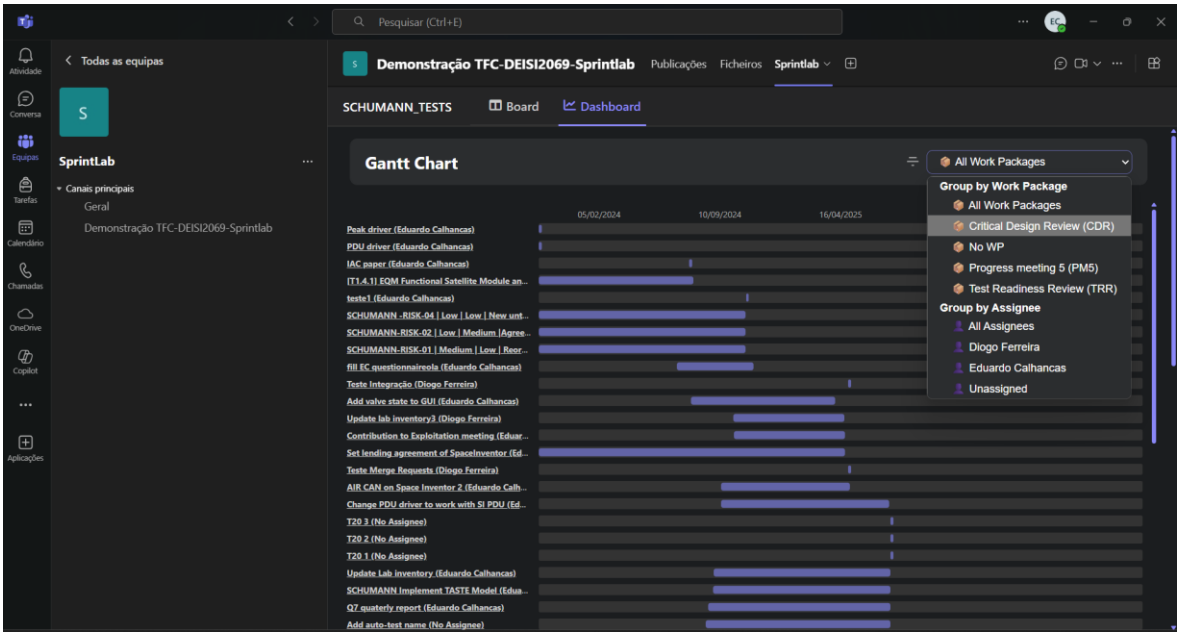


Figura 52 - Teste 20 - 1

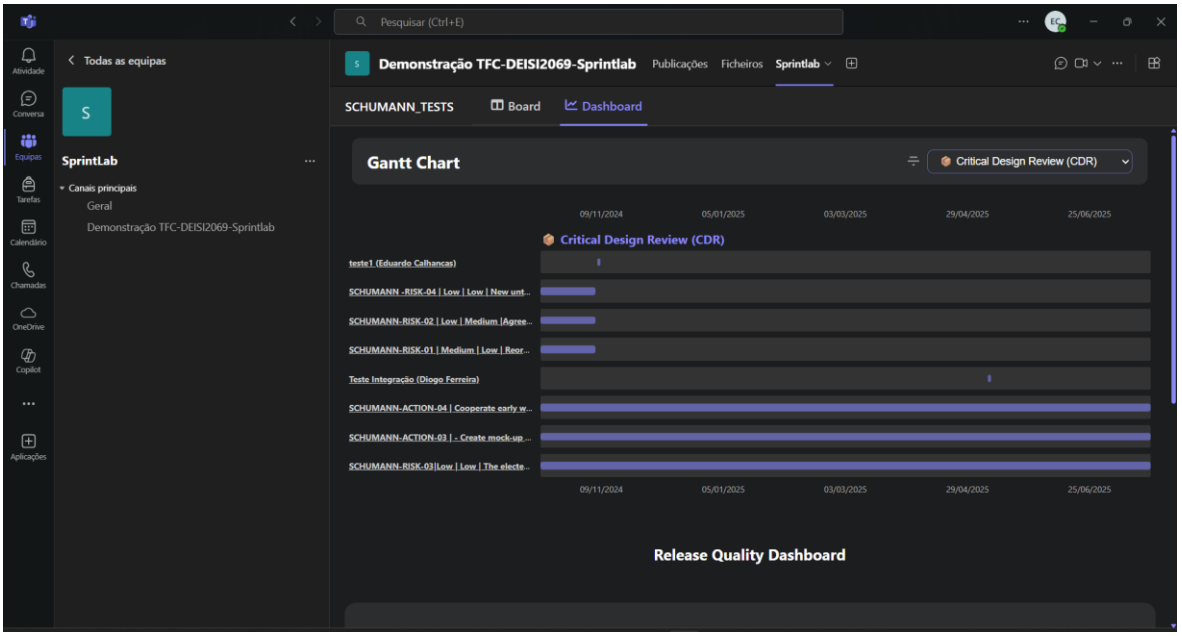


Figura 53 - Teste 20 - 2

Teste T21 - Abrir issue no *Gitlab* através do Gantt Chart

- **Objetivo**: Confirmar que ao clicar num issue no Gantt Chart, a aplicação redireciona corretamente para o *Gitlab*, abrindo o issue numa nova aba.
- **Pré-condições**:
  - O Gantt Chart está funcional e com issues visíveis.
  - Os issues no gráfico têm link associado.
- **Procedimento**:
  1. Aceder ao Dashboard do SprintLab.
  2. Identificar uma tarefa/linha representada no Gantt Chart.
  3. Clicar sobre o nome ou referência do issue.
  4. Verificar se a página do *Gitlab* abre corretamente no navegador.
- **Resultado Esperado**:

    - A página do issue correspondente abre corretamente no *Gitlab* num novo separador.
- **Resultado Obtido**:
  Passou - O link do issue funcionou como esperado.
- **Evidência**:

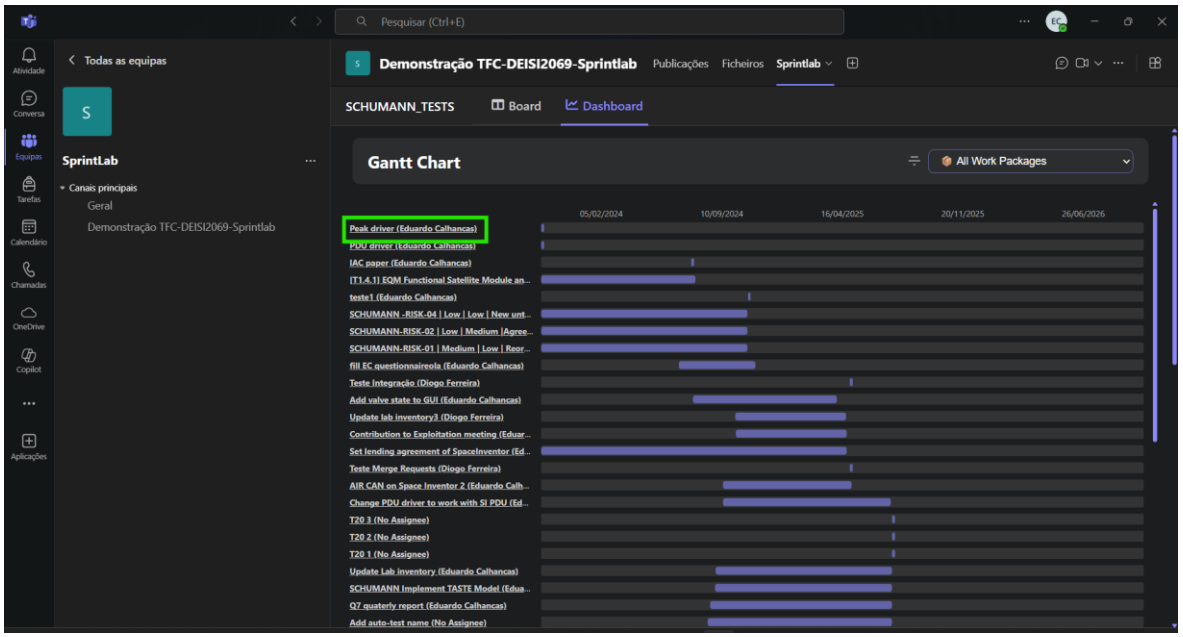


Figura 54 - Teste 21 - 1

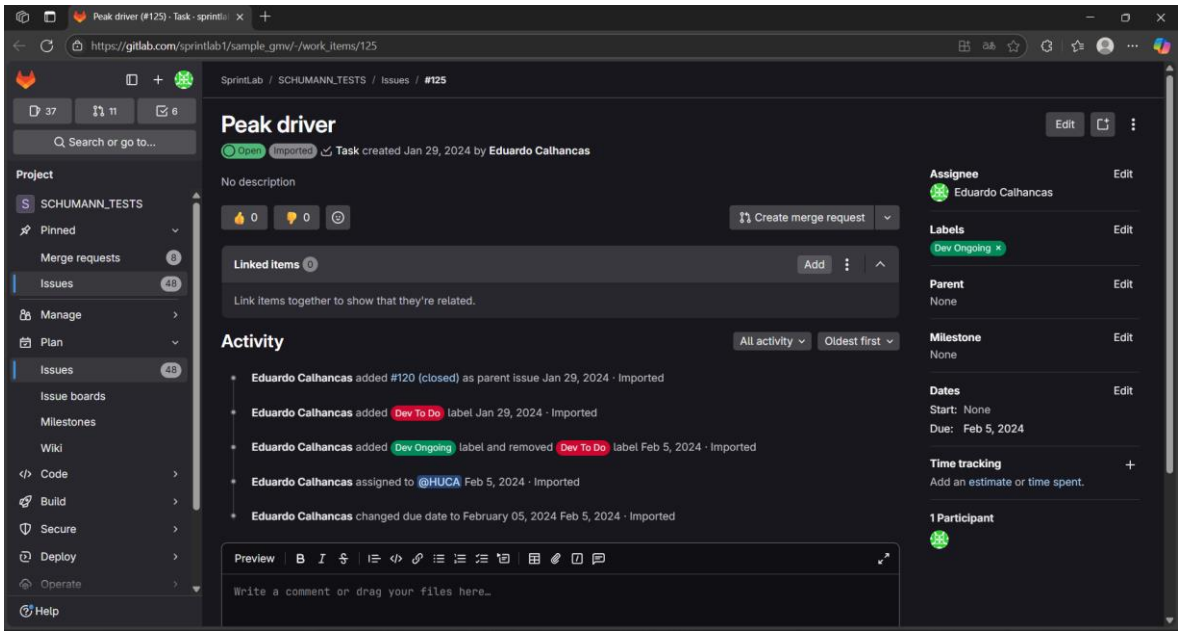


Figura 55 - Teste 21 - 2

Teste T22 - Reiniciar o serviço no Fly.io e validar estabilidade

- **Objetivo**: Verificar se o SprintLab mantém o seu funcionamento normal após reiniciar o serviço no Fly.io.
- **Pré-condições**:
    - A instância do middleware está hospedada no Fly.io e funcional.
    - O SprintLab está ativo no Teams.
- **Procedimento**:

1. Aceder à consola de gestão do Fly.io.
2. Efetuar um restart da aplicação (ex.: flyctl restart sprintlab ou via dashboard).
3. Após reinício, aceder ao SprintLab no Teams.
4. Verificar se a aplicação mantém o contexto e recupera os dados do projeto.

- **Resultado Esperado**:
  - O SprintLab permanece funcional após o reinício.
  - A ligação ao *Gitlab* é reestabelecida sem perda de dados ou estado.
- **Resultado Obtido**:
  Passou - O serviço reiniciou corretamente e a aplicação continuou funcional.
- **Evidência**:

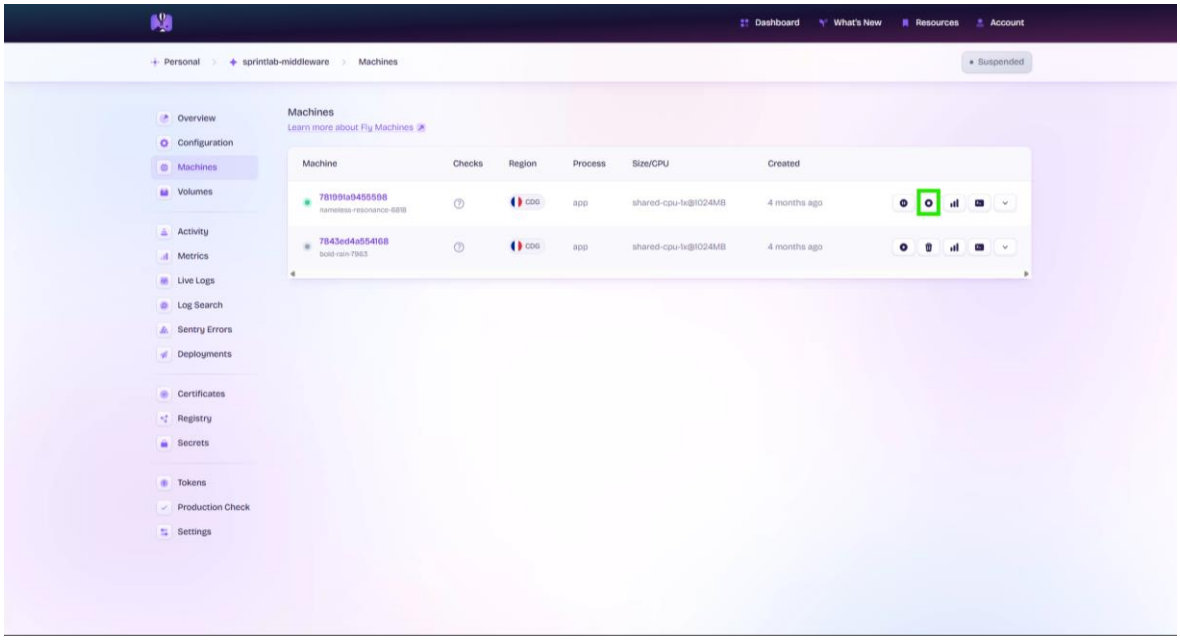


Figura 56 - Teste 22 - 1

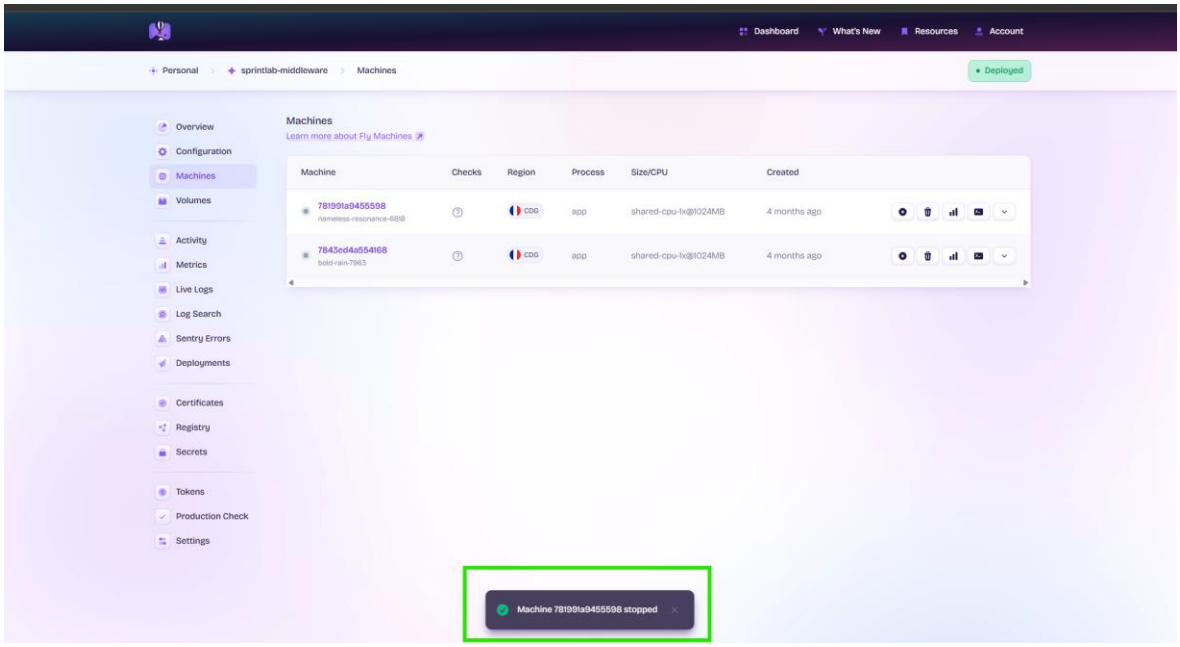


Figura 57 - Teste 22 - 2

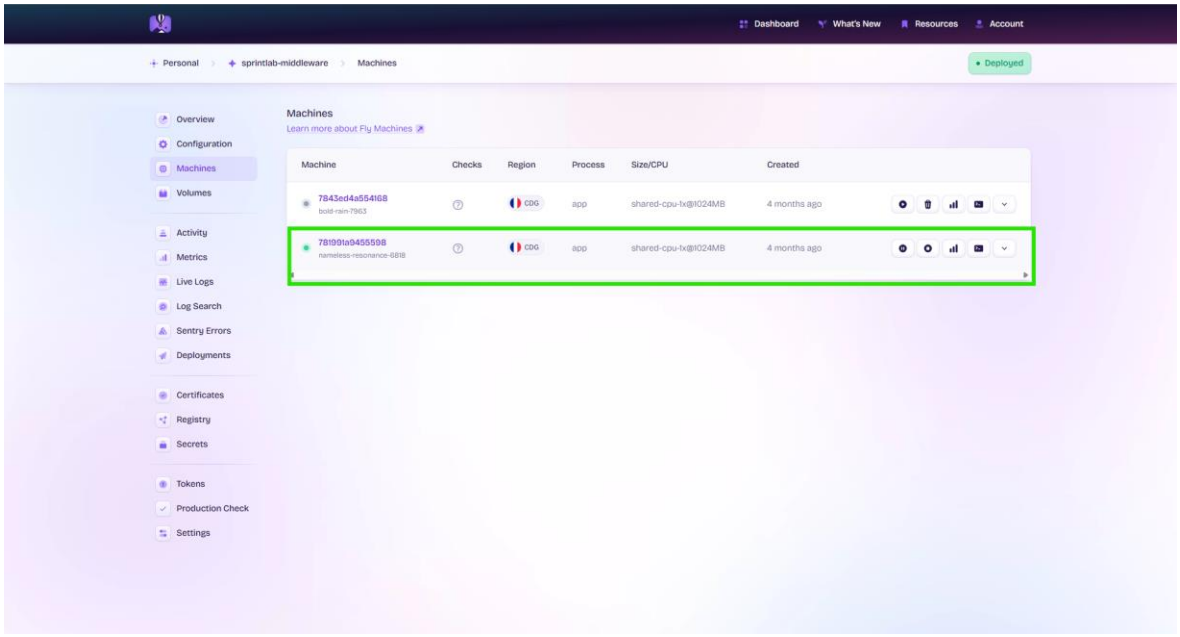


Figura 58 - Teste 22 - 3

Teste T23- Testar latência e carregamento de boards grandes

- **Objetivo**: Avaliar o desempenho do SprintLab ao carregar projetos *Gitlab* com um grande volume de issues, verificando se a aplicação continua funcional e responsiva.
- **Pré-condições**:
  - Existe um projeto *Gitlab* com mais de 200 issues.
  - O projeto está corretamente configurado no SprintLab.
- **Procedimento**:
  1. Aceder à aplicação SprintLab.
  2. Selecionar o projeto com mais de 200 issues no dropdown de boards (caso aplicável).
  3. Observar o tempo de carregamento da board.
  4. Verificar se todas as colunas e issues são renderizadas corretamente.
- **Resultado Esperado**:
  - Todos os issues são carregados e exibidos sem falhas visuais ou funcionais.
  - O tempo de carregamento é aceitável (sem timeouts ou erros críticos).
- **Resultado Obtido**:
  Passou - A aplicação manteve-se funcional e responsiva com grande volume de dados num tempo aceitável de 4.27 segundos.
- **Evidência**:

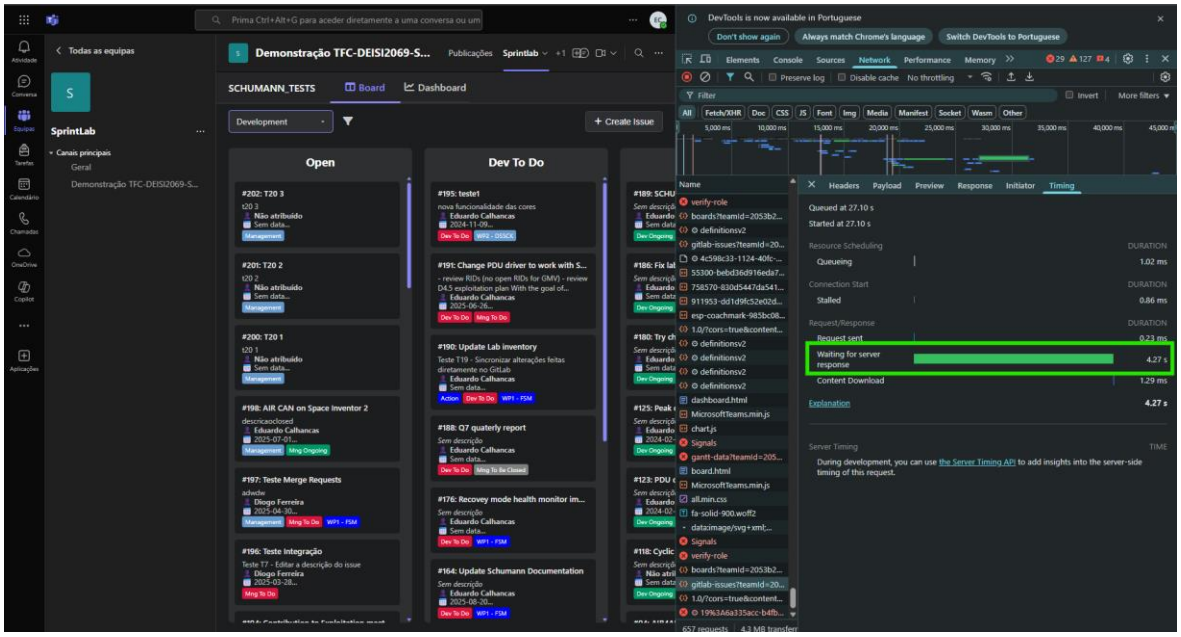


Figura 59 - Teste 23 - 1

Teste T24 - Verificar proteção de dados sensíveis (*token*/API key)

- **Objetivo**: Garantir que dados sensíveis, como *token*s de autenticação ou API keys, não são expostos visualmente na interface do SprintLab nem transmitidos por canais inseguros.

- **Pré-condições**:
  - O SprintLab está ativo e ligado a um projeto *Gitlab*.
  - O browser tem ferramentas de inspeção de rede (DevTools).
- **Procedimento**:
  1. Abrir o SprintLab no Teams.
  2. Ativar o inspetor de rede (DevTools > Network).
  3. Navegar pela aplicação normalmente, interagindo com boards, issues e configuração.
  4. Procurar nos headers, payloads e respostas por *token*s, API keys ou dados sensíveis.
- **Resultado Esperado**:
  - Nenhum *token*, chave de API ou dado confidencial é visível na interface.
  - As chamadas de rede não expõem esses dados em texto claro.
- **Resultado Obtido**:
  Passou - Não foram encontrados dados sensíveis expostos.
- **Evidência**:

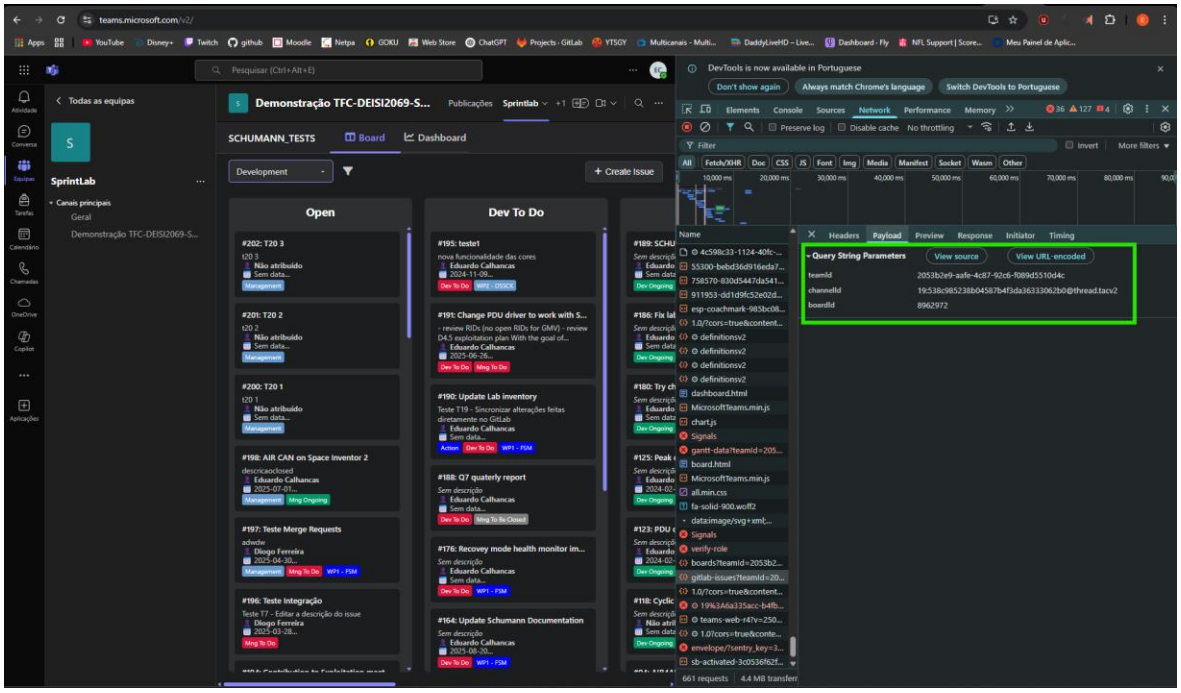


Figura 60 - Teste 24 - 1

7.2 Cumprimento de requisitos

Requisitos Funcionais:

Requisito	Estado	Justificação
RF01: O middleware deve estar alojado num servidor para funcionamento contínuo.	Realizado	Middleware está acessível via IP/domínio válido e logs confirmam funcionamento.
RF02: O servidor onde o middleware está alojado deve ser configurado para operação contínua.	Realizado	Reinício automático após falhas e logs de monitorização disponíveis.
RF03: O middleware deve estar configurado com o webhook do *Gitlab*.	Realizado	Webhook configurado e eventos estão a ser recebidos conforme os logs.
RF04: O middleware deve expor um endpoint para receber notificações da API do *Gitlab*.	Realizado	Eventos disparados no *Gitlab* são registados pelo middleware.

RF05: O middleware deve processar notificações de eventos de push no *Gitlab* e armazenar as informações.	Realizado	Eventos de push registados em logs; relatório acessível.
RF06: O middleware deve processar notificações relativas à criação de issues no *Gitlab*.	Realizado	Criação de issues gera entradas nos logs e tarefas no Teams.
RF07: O middleware deve processar notificações de atualização de detalhes das issues no *Gitlab*.	Realizado	Atualizações refletem-se no Teams e são registadas em log.
RF08: O middleware deve processar notificações de encerramento de issues no *Gitlab*.	Realizado	Tarefas no Teams são marcadas como concluídas e logs registam a alteração.
RF09: O middleware deve processar notificações relacionadas à reabertura de issues no *Gitlab*.	Realizado	Reabertura reativa tarefas no Teams e logs confirmam a ação.
RF10: O middleware deve identificar alterações nos rótulos (labels) das issues no *Gitlab*.	Realizado	Alterações refletem-se no Teams e são registadas em log.
RF11: O middleware deve receber notificações de criação de merge requests no *Gitlab*.	Realizado	Criação de merge requests registada e tarefas criadas no Teams.
RF12: O middleware deve processar notificações de alterações nos títulos dos merge requests.	Realizado	Títulos atualizados no Teams e registados nos logs.
RF13: O middleware deve processar notificações de alterações nas descrições dos merge requests.	Realizado	Descrições sincronizadas com o Teams e registadas em log.
RF14: O middleware deve registar notificações de aprovação de merge requests no *Gitlab*.	Realizado	Aprovações registadas e tarefas no Teams atualizadas.
RF15: O middleware deve registar notificações de rejeição de merge requests no *Gitlab*.	Realizado	Rejeições registadas em logs e refletidas no Teams.
RF16: O middleware deve processar notificações de alterações no estado dos merge requests no *Gitlab*.	Realizado	Alterações de estado registadas em logs e refletidas no Teams. Atualização em até 5 minutos.

RF17: O middleware deve capturar eventos de comentários adicionados a merge requests no *Gitlab*.	Realizado	Comentários registados nos logs e disponíveis para consulta.
RF18: O middleware deve transformar dados das issues do *Gitlab* em tarefas visíveis no quadro Kanban do Teams.	Realizado	Cada issue gera uma tarefa no Kanban; logs confirmam criação.
RF19: O middleware deve transformar listas de verificação (checklists) dentro das issues do *Gitlab* em subtarefas no quadro Kanban do Teams.	Realizado	Checklists convertidas em subtarefas; logs confirmam sincronização.
RF20: O middleware deve sincronizar alterações no quadro Kanban do Teams com os rótulos ou estados das issues no *Gitlab*.	Realizado	Alterações refletem-se no *Gitlab* em até 1 minuto; registadas em log.
RF21: O middleware deve capturar eliminações de tarefas no quadro Kanban e refletir as alterações nas issues correspondentes no *Gitlab*.	Realizado	Eliminações no Teams removem issues no *Gitlab*; logs registam ação.
RF22: O middleware deve sincronizar alterações feitas no *Gitlab* com o quadro Kanban do Teams.	Realizado	Alterações no *Gitlab* atualizam o Kanban em até 1 minuto; logs confirmam.
RF23: O plugin do Teams deve exibir um quadro Kanban com a capacidade de filtrar por rótulos.	Realizado	Filtros por labels disponíveis; logs registam filtros aplicados.
RF24: O plugin do Teams deve exibir um gráfico de Gantt de projetos completos.	Realizado	Gantt gerado com dados completos do *Gitlab*; logs registam criação.
RF25: O plugin do Teams deve exibir um gráfico de Gantt filtrado por rótulos específicos.	Realizado	Gantt gerado com base em labels específicos; logs confirmam.
RF26: O plugin do Teams deve conectar-se ao middleware para obter gráficos atualizados.	Realizado	Logs mostram pedidos de atualização; gráficos refletem os dados atuais.
RF27: O plugin do Teams deve conectar-se ao middleware para obter tarefas atualizadas.	Realizado	Logs confirmam atualizações; Kanban exibe tarefas atuais.
RF32: O middleware deve incluir ligações diretas para issues no quadro Kanban do Teams.	Realizado	Cada tarefa tem link clicável para a issue; logs registam os links.

RF33: O middleware deve incluir ligações diretas para merge requests no quadro Kanban do Teams.	Realizado	Tarefas no Kanban têm links para merge requests; logs confirmam.
RF34: O middleware deve suportar múltiplos projetos *Gitlab* sincronizados com diferentes equipas no Teams.	Realizado	Sincronização de múltiplos projetos funcional; logs validam uso.

Tabela 6 - Requisitos Funcionais

Requisitos Não-Funcionais

Requisito	Estado	Justificação
RNF01: O middleware deve garantir comunicação segura usando HTTPS para todas as interações.	Realizado	Todas as comunicações com *Gitlab* e Teams são criptografadas.
RNF02: O middleware deve ser escalável para suportar o aumento do número de projetos ou eventos.	Realizado	Suporta aumento de tráfego em 100% e até 1.000 eventos simultâneos sem erros críticos.
RNF03: O middleware deve ser compatível com JavaScript.	Realizado	Scripts e dependências são compatíveis com versões suportadas de JavaScript.
RNF04: O middleware deve autenticar-se de forma segura na API do *Gitlab* utilizando *token*s.	Realizado	*Token*s são validados em cada chamada e renovados automaticamente.
RNF05: O middleware deve autenticar-se de forma segura na API do *Microsoft Teams* utilizando *token*s.	Realizado	*Token*s válidos são usados e erros de autenticação tratados e registados.
RNF06: O middleware deve registar Logs detalhados de eventos processados e erros encontrados.	Realizado	Logs incluem timestamp, tipo de evento e severidade dos erros.
RNF07: O middleware deve operar com alta disponibilidade, reiniciando automaticamente em caso de falhas críticas.	Realizado	Operacionalidade de 99,9% com reinicializações automáticas em menos de 1 minuto.
RNF08: O middleware deve suportar múltiplos projetos GitLab simultaneamente através da API.	Realizado	Processamento simultâneo de pelo menos 10 projetos sem interferência mútua.
RNF09: O middleware deve suportar a integração com múltiplos canais e equipas no *Microsoft Teams*.	Realizado	Suporte simultâneo a pelo menos 5 canais, com segregação por projeto.
RNF10: O middleware deve manter desempenho consistente mesmo sob alta carga de eventos.	Realizado	Latência máxima de 1 segundo e capacidade de até 1.000 eventos/hora.
RNF11: O middleware deve evitar dependências que limitem a sua compatibilidade com versões futuras do JavaScript.	Realizado	Dependências são atualizáveis e ferramentas de análise validam compatibilidade.
RNF12: O middleware deve registar interrupções na comunicação com o *Gitlab* e tentar novamente os eventos que falharem.	Realizado	Três tentativas automáticas antes de gerar alertas para administradores.
RNF13: O middleware deve registar interrupções na comunicação com o	Realizado	Reconexões automáticas em até 3 ciclos; mensagens de erro

Microsoft Teams e tentar novamente os eventos que falharem.		apresentadas se falhar.
RNF14: O middleware deve ser configurável para operar em múltiplos ambientes (desenvolvimento, homologação e produção).	Realizado	Arquivos de configuração independentes permitem mudança de ambiente sem alterar o código.
RNF15: O plugin do Teams deve garantir uma experiência responsiva, mesmo com múltiplos gráficos ou tarefas.	Realizado	Carregamento inicial em até 2 segundos; atualizações aplicadas em até 5 segundos.
RNF16: O middleware deve suportar monitorização externa com Fly.io.	Realizado	Métricas exportadas para Fly.io; alertas configurados para falhas críticas.

Tabela 7 - Requisitos Não-Funcionais

Requisitos Técnicos

Requisito	Estado	Justificação
RT01: O middleware deve ser construído utilizando o framework Express.js para expor APIs REST próprias.	Realizado	Todas as rotas respondem eficientemente em até 500 ms com organização adequada.
RT02: O middleware deve ser alojado num servidor configurado para alta disponibilidade e balanceamento de carga.	Realizado	Suporte a balanceamento entre 2 servidores com disponibilidade de 99,9%.
RT03: O middleware deve utilizar a biblioteca axios para interagir com a API do *Gitlab*.	Realizado	Axios é utilizado com *token*s de acesso em chamadas HTTP fiáveis.
RT04: O middleware deve usar a biblioteca MSAL para autenticação OAuth2 com a Microsoft Graph API.	Realizado	*Token*s são geridos automaticamente e falhas são corretamente tratadas.
RT05: O middleware deve suportar chamadas REST da API do *Gitlab* para aceder a issues e merge requests.	Realizado	Chamadas retornam JSON completo em até 1 segundo.
RT06: O middleware deve suportar chamadas GraphQL da API do *Gitlab* para aceder a issues e merge requests.	Realizado	Consultas GraphQL funcionam e erros são tratados com mensagens claras.
RT07: O middleware deve permitir a configuração e gestão dos tokens e identificadores de projeto GitLab através de um painel administrativo.	Realizado	O painel administrativo suporta a criação, atualização e exclusão de configurações de projetos GitLab (token e project ID).
RT08: O middleware deve formatar dados de issues do *Gitlab* para serem compatíveis com quadros Kanban no Teams.	Realizado	Campos mapeados corretamente e atualizações refletidas em até 1 minuto.
RT09: O middleware deve formatar dados de rótulos do *Gitlab* para serem compatíveis com gráficos de Gantt no Teams.	Realizado	Labels convertidos corretamente e apenas labels ativos são usados.
RT10: O plugin do Teams deve utilizar o SDK do *Microsoft Teams* para integração total com a interface do Teams.	Realizado	Tarefas e gráficos exibidos corretamente com atualizações bidirecionais.

RT11: O plugin do Teams deve consumir os endpoints do middleware para exibir tarefas atualizadas.	Realizado	Tarefas são obtidas a cada 5 minutos e inconsistências corrigidas automaticamente.
RT12: O plugin do Teams deve consumir os endpoints do middleware para exibir gráficos atualizados.	Realizado	Gráficos são atualizados com dados do *Gitlab* e erros são comunicados ao utilizador.
RT13: O middleware deve ser configurado para alojar Logs e métricas acessíveis para administradores.	Realizado	Logs e métricas acessíveis via painel, com controlo de acesso autenticado.
RT14: O middleware deve ser executado com o servidor Fly.io configurado para alta performance em produção.	Realizado	Múltiplas instâncias e escalonamento automático com resposta inferior a 500 ms.
RT15: O middleware deve suportar autenticação personalizada para utilizadores no *Gitlab* e no Teams.	Realizado	Acesso restrito a utilizadores autenticados com verificação de permissões.

Tabela 8 - Requisitos Técnicos

8 Conclusão

8.1 Conclusão

A realização do Trabalho Final de Curso (TFC) permitiu consolidar conhecimentos adquiridos ao longo da formação e aplicá-los numa solução concreta, com aplicabilidade real no contexto da gestão de projetos. Esta secção apresenta uma reflexão crítica sobre o percurso realizado, os desafios enfrentados, as aprendizagens obtidas e as decisões tomadas ao longo do desenvolvimento.

Grau de concretização do plano

O plano definido inicialmente foi, em grande parte, concretizado com sucesso. Todos os requisitos funcionais, não funcionais e técnicos considerados prioritários foram implementados e validados. A estrutura de desenvolvimento e os marcos principais foram cumpridos, embora tenham existido ajustes pontuais no calendário para acomodar melhorias e testes adicionais, sobretudo nas fases de integração contínua com o *Microsoft Teams* e nas validações de desempenho em Fly.io.

Diferenças entre a solução proposta inicialmente e a solução desenvolvida

A proposta inicial previa uma integração básica entre *Gitlab* e Teams, centrada na visualização de tarefas. Contudo, ao longo do projeto, a solução evoluiu para incluir funcionalidades avançadas como gráficos de Gantt, sincronização bidirecional de issues, gestão de múltiplos projetos e suporte a múltiplas equipas, o que tornou a aplicação mais robusta e aderente às necessidades reais do utilizador final. A utilização de ferramentas como Fly.io, MSAL e dashboards filtráveis também foi uma evolução natural não contemplada inicialmente.

Evolução do trabalho e dos conhecimentos ao longo do TFC

Ao longo do desenvolvimento, houve um crescimento significativo nas competências técnicas e metodológicas. Foi necessário aprofundar conhecimentos em APIs REST, autenticação OAuth2, manipulação de dados em tempo real, sincronização de serviços, integração de SDKs, e gestão de escalabilidade em ambiente cloud. Além disso, houve uma melhoria notável na capacidade de planear, testar e validar funcionalidades complexas, com base em critérios de aceitação claros e alinhados com a prática profissional.

O que se faria diferente se o TFC voltasse ao princípio

Se o projeto fosse recomeçado, seria vantajoso:

1. Investir mais tempo na fase inicial de planeamento, com definição mais detalhada dos fluxos de dados entre os serviços.
2. Implementar testes automatizados desde o início, para garantir uma base de validação contínua ao longo do desenvolvimento.
3. Fazer um levantamento de requisitos com utilizadores reais logo no arranque, para

orientar melhor as prioridades de implementação.
4. E ainda, separar desde cedo as funções de backend e frontend, para facilitar a manutenção e testabilidade.

Maiores dificuldades na realização do TFC

As principais dificuldades enfrentadas foram:

1. Integração segura e estável com múltiplas APIs (*Gitlab* e Microsoft Graph), exigindo leitura atenta da documentação e testes extensivos;
2. Gestão de autenticação e *token*s, especialmente em ambientes distintos (públicos e privados);
3. Garantia de sincronização em tempo real, sem inconsistências entre plataformas;
4. Validação do desempenho sob carga, que exigiu simulações realistas com elevado volume de issues;
5. E finalmente, garantir a experiência do utilizador no *Microsoft Teams*, respeitando as limitações da plataforma e a fluidez da interface.

8.2 Trabalhos Futuros

Caso o projeto SprintLab venha a ser continuado após a conclusão deste Trabalho Final de Curso, existem diversas oportunidades de melhoria técnica e de expansão do seu potencial enquanto solução inovadora para integração de ferramentas de colaboração e desenvolvimento.

Atualmente, a solução encontra-se operacional na plataforma cloud Fly.io, onde o middleware está alojado com alta disponibilidade e escalabilidade. No entanto, para garantir uma integração completa com o ecossistema *Microsoft Teams* utilizado pela Universidade Lusófona, seria necessário associar a aplicação ao Microsoft Azure institucional. Essa associação não foi possível durante o desenvolvimento devido à dependência de terceiros na gestão dessa infraestrutura, o que limitou o acesso às permissões administrativas necessárias para configurar a aplicação como um custom app oficialmente autorizado no ambiente Teams da universidade. Assim, um dos passos futuros mais relevantes será completar essa integração com o Azure da Lusófona ou, em alternativa, disponibilizar a aplicação em ambientes independentes que não dependam de infraestrutura institucional, permitindo uma utilização autónoma e escalável por organizações externas.

Do ponto de vista técnico, destaca-se a necessidade de concluir a implementação do sistema de permissões, garantindo que apenas utilizadores autorizados possam criar, editar ou eliminar tarefas e issues. A introdução de um painel administrativo com interface gráfica permitirá também uma gestão mais eficaz das configurações, do estado da integração e da segurança.

Serão ainda exploradas melhorias como o envio de notificações proativas no Teams, a geração de relatórios com métricas de desempenho e a otimização do desempenho do middleware para projetos com elevada volumetria de dados.

No âmbito da inovação e empreendedorismo, o SprintLab tem potencial para evoluir para uma solução SaaS comercial, com modelo de subscrição e integração plug-and-play com *Gitlab* e *Microsoft Teams*. A criação de uma versão gratuita com funcionalidades básicas, associada a planos premium, poderá facilitar a adoção inicial. Para além disso, parcerias estratégicas com a Microsoft ou *Gitlab* e a publicação do plugin em marketplaces como a Microsoft AppSource podem alavancar significativamente a visibilidade e o crescimento da solução.

Por fim, poderá ser explorada a incorporação de inteligência artificial para previsão de atrasos, sugestões de priorização e análise de produtividade por sprint, contribuindo para tornar o SprintLab uma plataforma mais inteligente e adaptável às necessidades das equipas.

Bibliografia

- Agile Alliance. (2024). Agile Alliance. Disponível em https://www.agilealliance.org (acedido em novembro de 2024).
- Antlia. (2023). Metodologias Ágeis: Transformando a forma de desenvolver projetos. Disponível em https://antlia.com.br/artigos/metodologias-ageis-transformando-a-forma-de-desenvolver-projetos (acedido em outubro de 2024).
- Atlassian. (2024). Agile Project Management Tools. Disponível em https://www.atlassian.com/br/agile/project-management/scrum-tools (acedido em outubro de 2024).
- DEISI. (2024). Regulamento de Trabalho Final de Curso.
- DEISI. (2024). Universidade Lusófona de Humanidades e Tecnologia. Disponível em https://www.deisi.ulusofona.pt (acedido em outubro de 2024).
- *Gitlab*. (2024). Integrações do *Gitlab* com *Microsoft Teams*. Disponível em https://docs.*Gitlab*.com/ee/user/project/integrations/*Microsoft_Teams*.html (acedido em outubro de 2024).
- GMV. (2024). GMV: Site oficial da GMV. Disponível em https://www.gmv.com.
- *Microsoft*. (2024). *Microsoft Teams*. Disponível em https://www.*Microsoft*.com/*Microsoft-Teams* (acedido em novembro de 2024).
- Schwaber, K., & Sutherland, J. (2020). The Scrum Guide: The Definitive Guide to Scrum. Disponível em https://scrumguides.org.
- Sky One Solutions. (2023). A falta de integração afeta a produtividade das equipas. Disponível em https://skyone.solutions/blog/falta-de-integracao-afeta-equipe (acedido em outubro de 2024).
- Tanenbaum, A., & Wetherall, D. (2020). Computer Networks (6ª ed.). Prentice Hall.
- Universidade Lusófona de Humanidades e Tecnologia. (2021). Universidade Lusófona de Humanidades e Tecnologia. Disponível em https://www.ulusofona.pt (acedido em outubro de 2024).
- FindUp. (2024). A tecnologia como ferramenta de comunicação corporativa. Disponível em https://www.findup.com.br/2024/01/23/tecnologia-comunicacao-corporativa (acedido em novembro de 2024).
- Lecom. (2024). *Automação de processos no desenvolvimento de software*. Disponível em https://www.lecom.com.br/blog/automacao-de-processos-no-desenvolvimento-de-software (acedido em novembro de 2024).
- McKinsey. (2024). *The New Frontier: Agile Automation at Scale*. Disponível em https://www.mckinsey.com/capabilities/operations/our-insights/the-new-frontier-agile-automation-at-scale (acedido em novembro de 2024).
- Unito. (2024). Comparação entre Unito e Z*API*er. Disponível em https://unito.io/blog/unito-vs-z*API*er (acedido em novembro de 2024).
- ClickUp. (2024). Integrações do Jira. Disponível em https://clickup.com/pt-BR/blog/integracoes-do-jira (acedido em novembro de 2024).
- Lyncas. (2024). Guia completo sobre Software as a Service (SaaS). Disponível em https://lyncas.net/blog/guia-software-as-a-service (acedido em novembro de 2024).
- *Microsoft* - Ecossistemas. (2024). Ecossistema de Parceiros da *Microsoft*: Inovação e Resiliência dos Negócios. Disponível em https://news.*Microsoft*.com/pt-

br/ecossistema-de-parceiros-da-*Microsoft*-permitindo-a-inovacao-e-a-resiliencia-dos-negocios (acedido em novembro de 2024).

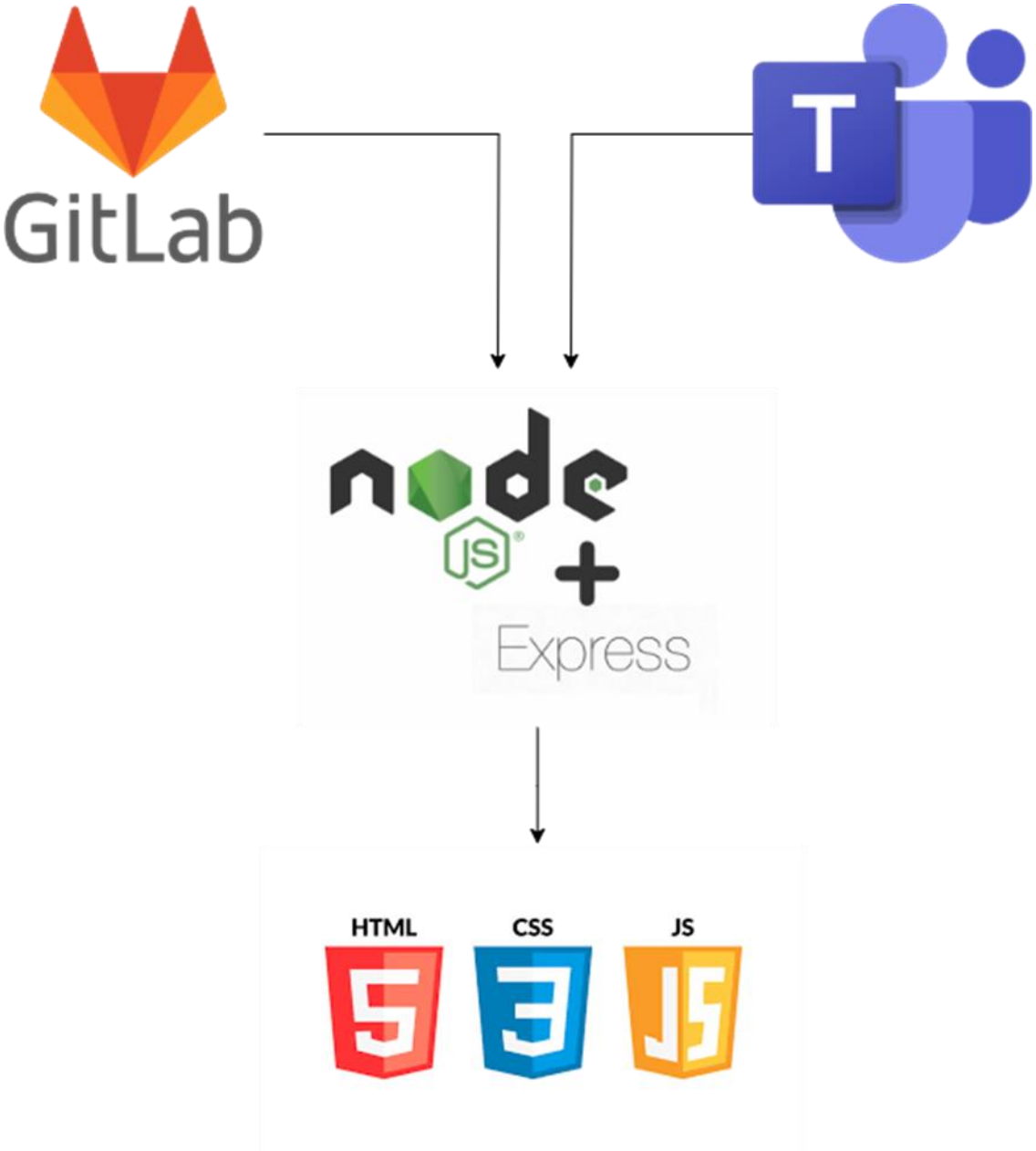
Glossário

LEI	Licenciatura em Engenharia Informática
LIG	Licenciatura em Informática de Gestão
TFC	Trabalho Final de Curso

Texto de Apresentação

Este projeto propõe uma integração eficiente entre o GitLab e o Microsoft Teams, permitindo visualizar e gerir tarefas em tempo real através de interfaces modernas como quadros Kanban e gráficos de Gantt. Desenvolvido com foco na agilidade e colaboração, o sistema facilita a gestão de projetos diretamente no ambiente Teams. Através de um middleware em JavaScript, a solução garante sincronização de dados, visualizações interativas e adaptabilidade a diferentes equipas e canais.

Imagem Representativa



Palavras-chave:

Integração, GitLab, Microsoft Teams, Kanban, Gantt, API, Gestão de Projetos, Middleware, TFC, Visualização de Tarefas

Tecnologias Utilizadas:

- JavaScript (Node.js, Express.js)
- Microsoft Teams SDK
- HTML, CSS
- Fly.io (deploy)
- PostgreSQL
- GitLab API REST

Área:

Engenharia Informática – Desenvolvimento Web e Integração de Sistemas

SPRINTLAB – INDEPENDENT EVALUATION REPORT

AIR

Prepared by:	OBSW Team GMV
Approved by:	Daniel Silveira
Authorized by:	Carolina Serra

Code:	GMV/10234/25
Version:	1
Date:	25/06/2025
Internal code:	GMV N/A

DOCUMENT STATUS SHEET

Version	Date		Changes
1	25/06/2025	9	

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

No table of figures entries found.

1. INTRODUCTION

1.1. PURPOSE

The purpose of this document is to present the results of an independent evaluation carried out by GMV on the SprintLab tool, developed by final-year students of Universidade Lusófona. This assessment aimed to verify SprintLab's operational performance as a project management integration solution between GitLab and Microsoft Teams.

1.2. SCOPE

This report covers functional, non-functional, and usability testing conducted by five GMV engineers using SprintLab in the context of two active real-world projects: Horizon Europe Schumann and GMV AIR. The evaluation includes bidirectional synchronization, dashboards, performance, reliability, and scalability aspects.

1.3. DEFINITIONS AND ACRONYMS

1.3.1. DEFINITIONS

Concepts and terms used in this document and needing a definition are included in the following table:

Table 1-1 Definitions

Concept / Term	Definition

1.3.2. ACRONYMS

Acronyms used in this document and needing a definition are included in the following table:

Table 1-2 Acronyms

Acronym	Definition

2. REFERENCES

2.1. APPLICABLE DOCUMENTS

The following documents, of the exact issue shown, form part of this document to the extent specified herein. Applicable documents are those referenced in the Contract or approved by the Approval Authority. They are referenced in this document in the form [AD.x]:

Table 2-1 Applicable Documents

Ref.	Title	Code	Version	Date
[AD.1]				
[AD.2]				
[AD.3]				
[AD.4]				

2.2. REFERENCE DOCUMENTS

The following documents, although not part of this document, amplify or clarify its contents. Reference documents are those not applicable and referenced within this document. They are referenced in this document in the form [RD.x]:

Table 2-2 Reference Documents

Ref.	Title	Code	Version	Date
[RD.1]				
[RD.2]				
[RD.3]				

3. TESTING OVERVIEW

3.1. SYSTEM OVERVIEW

SprintLab is a middleware and plugin-based integration platform that enables seamless project tracking and management between GitLab and Microsoft Teams. The solution supports Kanban and Gantt visualizations, real-time updates, and multi-project configuration.

(See Section 2.2 for evaluation approach.)

3.2. APPROACH

The assessment consisted of executing SprintLab in realistic operational conditions, replicating typical use cases of project coordination. The GMV team followed a black-box and user-oriented testing approach to evaluate the system based on:

- Synchronization accuracy between GitLab and Teams.
- Responsiveness of dashboards and visual components.
- Support for multi-project and multi-team environments.
- Compliance with the functional requirements defined in the SprintLab technical documentation.

In total, over a dozen test scenarios were covered, including edge cases and system resilience.

3.3. TESTING ENVIRONMENT, TOOLS AND REQUIRED INPUTS

Projects tested:

- Schumann: A Horizon Europe project focused on cybersecurity, collaborative threat detection and defense.
- AIR: A GMV product for secure virtualization and automated orchestration of software environments.

Test team: 5 GMV engineers from space and IT divisions.

Infrastructure: GitLab (Enterprise & Self-Hosted), Microsoft Teams (Standard Channels), SprintLab deployed on Fly.io.

Inputs: Existing GitLab repositories and work packages (WPs) with open issues and merge requests.

Tooling: Native Teams interface, SprintLab plugin interface, GitLab Webhooks, browser, logs viewer.

4. TEST DESCRIPTION AND RESULTS

This section includes a summary of the verification and validation activities results. It identifies the verified and validated elements by marking its version number. Reference will be made to the applicable documents and/or plans. It will identify any incidents and summarize how they were solved.

4.1. CORE SYNCHRONIZATION TESTS

Test Scenario	Description	Result
Issue creation in GitLab	Reflected immediately in Teams Kanban board	Success
Task creation in Teams	Corresponding issue generated in GitLab	Success
Status updates (e.g., closing issue)	Reflected bidirectionally in GitLab and Teams	Success
Label changes and filtering	Correct rendering and filtering on both sides	Success
Gantt chart rendering	Updated automatically with correct time-line mapping	Success

4.2. ADVANCED FEATURE TESTS

Test Scenario	Description	Result
Simultaneous management of multiple GitLab projects	Independent synchronization in distinct Teams channels	Success
Notifications for Merge Requests	Created and updated in Teams	Success
Checklist/subtask visibility	Reflected correctly as subtasks in Kanban board	Success
Real-time updates under load	Handled +100 events/min without delays	Success
Token and API authentication handling	All API interactions secured via OAuth2 and MSAL	Success

4.3. USABILITY AND PERFORMANCE FEEDBACK

All testers found the tool intuitive and responsive. Project dashboards allowed faster monitoring of task closure and WP evolution. Participants estimated:

- **30–50% reduction** in time spent coordinating and reporting task progress.
- Improved **clarity in communication** and project visibility across distributed teams.

5. CONCLUSION

SprintLab demonstrated robust, reliable, and production-grade behavior in real-world project contexts. All tested functionalities performed as expected, with no critical issues encountered.

GMV considers SprintLab suitable for operational use and recommends exploring its broader adoption or commercialization.