UNIVERSIDADE DE RIO VERDE

ENGENHARIA DE SOFTWARE
GERÊNCIA DE CONFIGURAÇÃO

# Notix – MindBoard

Rian Guedes Rodrigues
Júlio Cezar Rodrigues Correia
Augusto Wobeto Carvalho
Eduardo Machado Martins

Rio Verde – GO
19 de junho de 2025

UNIVERSIDADE DE RIO VERDE

# ENGENHARIA DE SOFTWARE
# GERÊNCIA DE CONFIGURAÇÃO

Rian Guedes Rodrigues
Júlio Cezar Rodrigues Correia
Augusto Wobeto Carvalho
Eduardo Machado Martins

Trabalho final apresentado como exigência parcial para aprovação na disciplina de Gerência de Configuração, do curso de Engenharia de Software, da Universidade de Rio Verde – Campus de Rio Verde – GO.

**Orientador:** Wayrone Klaiton Luiz Silva

Rio Verde – GO
19 de junho de 2025

# Sumário

# 1 Introdução

## 1.1 Objetivo do Plano

Este plano estabelece diretrizes, ferramentas e responsabilidades para o gerenciamento de configuração dos artefatos do projeto **Notix**, assegurando integridade, rastreabilidade e controle de versões durante o desenvolvimento.

## 1.2 Escopo do Projeto

O **Notix** é uma aplicação web leve baseada no conceito de "MindBoard Lite", que permite aos usuários criar, visualizar e apagar notas rápidas em um quadro. As notas são armazenadas localmente no navegador, oferecendo uma ferramenta ágil e intuitiva para organização pessoal.

## 1.3 Justificativa da GCS

A aplicação da Gerência de Configuração de Software (GCS) é fundamental em um projeto colaborativo. Ela garante controle sobre as alterações no sistema, evita conflitos de código, organiza o fluxo de trabalho e permite acompanhar o progresso por meio do versionamento e documentação.

# 2 Estrutura Organizacional e Papéis

## 2.1 Composição da Equipe

Tabela 1: Composição da Equipe e Papéis

| **Nome** | **GitHub** | **Papel no Projeto** | **Papel na GCS** |
|---|---|---|---|
| Rian Guedes Rodrigues | `riangrodrigues` | Dev. Full-Stack | Gerente de Configuração |
| Augusto Wobeto Carvalho | `Otsugya` | Dev. Frontend | Controlador de Mudanças |
| Júlio Cezar R. Correia | `muddyorc` | Dev. Backend | Revisor de Código |
| Eduardo Machado Martins | `M4chado` | Dev. Frontend | Revisor de Documentação |

## 2.2 Responsabilidades Específicas

- **Gerente de Configuração:** estrutura o repositório, realiza merges na branch `main`, cria tags e releases.
- **Controlador de Mudanças:** gerencia issues, garante que mudanças estejam documentadas e facilita comunicação.
- **Revisores:** auditam código, verificam documentação e validam critérios de aceitação.

# 3 Ferramentas e Ambientes

Tabela 2: Ferramentas Utilizadas no Projeto

| **Ferramenta** | **Finalidade** | **Responsável Principal** |
|---|---|---|
| Git | Controle de versão local e ramificação | Todos |
| GitHub | Repositório remoto, colaboração, Pull Requests | Todos |
| GitHub Issues | Rastreamento de tarefas e mudanças | Controlador de Mudanças |
| VS Code | Editor de código | Todos |

# 4 Estrutura de Diretórios e Arquivos

- **Raiz**: .gitignore, README.md, CHANGELOG.md
- /**src**: index.html, style.css, script.js
- /**docs**: Plano de Gerência de Configuração (este documento)

# 5 Itens de Configuração

## 5.1 Itens Controlados

Tabela 3: Itens de Configuração Controlados

| **Tipo** | **Nome / Descrição** | **Localização** | **Status** | **Responsável** |
|---|---|---|---|---|
| Código-Fonte | `index.html`, `style.css`, `script.js` | `/src/` | Concluído | Todos |
| Documentação | `README.md`, `CHANGELOG.md` | Raiz | Concluído | Revisor de Docs |
| Plano de GCS | Plano de Gerência de Configuração | `/docs/` | Concluído | Todos |

# 6 Controle de Versão

## 6.1 Estratégia de Branching

Adotamos o modelo **GitHub Flow**. As ramificações seguem as seguintes convenções:

- `main`: contém o código estável e pronto para produção.
- `feature/<nome>`: destinada ao desenvolvimento de novas funcionalidades.
- `fix/<nome>`: utilizada para correções de erros.
- `refactor/<nome>`: empregada para melhorias na estrutura do código.

Esse modelo permite integração contínua e maior controle sobre o ciclo de desenvolvimento.

## 6.2 Commits e Pull Requests

Todos os commits seguem o padrão **Conventional Commits**. Cada **Pull Request (PR)** deve:

- Referenciar uma `Issue` associada à alteração.
- Ser revisado e aprovado por pelo menos um membro da equipe antes do merge na `main`.

Esse processo garante maior qualidade no código e promove a colaboração.

# 7 Controle de Mudança

O controle de todas as alterações, sejam novas funcionalidades, melhorias ou correções de bugs, foi gerenciado através do sistema de **GitHub Issues**, seguindo um processo bem definido:

1. **Criação da Issue:** Para toda mudança necessária, uma nova Issue era aberta, descrevendo o objetivo e os critérios de aceitação.
2. **Discussão e Atribuição:** A equipe discutia a Issue para garantir o entendimento comum, e um membro era atribuído para a implementação.
3. **Desenvolvimento em Branch:** O desenvolvedor criava um branch específico para a Issue, garantindo o isolamento do trabalho.
4. **Vinculação e Resolução:** O Pull Request aberto para resolver a tarefa sempre referenciava a Issue correspondente. Após o merge do PR, a Issue era fechada automaticamente, garantindo a rastreabilidade completa.

# 8 Auditoria e Status de Configuração

## 8.1 Auditorias

- **Código**: todo código é revisado durante o processo de Pull Request.
- **Documentos**: a documentação é validada semanalmente.
- **Estrutura**: a estrutura do repositório é limpa antes de cada release.

## 8.2 Relatório de Status

O status do projeto foi acompanhado em tempo real através do quadro Kanban no GitHub Projects, que oferece uma visão clara das tarefas em "Backlog", "In progress", "In review"e "Done".

# 9 Linhas de Base (Baselines)

## 9.1 Marcos e Releases

As linhas de base são registradas por meio de tags no repositório GitHub e documentadas conforme a tabela a seguir:

Tabela 4: Marcos e Releases

| **Marco** | **Tag** | **Descrição** | **Data** |
|---|---|---|---|
| Estrutura inicial | v0.1.0 | Setup do repositório e docs iniciais | 15/06/2025 |
| Criação de notas | v0.2.0 | Implementação da criação de notas | 15/06/2025 |
| Exclusão de notas | v0.3.0 | Implementação da exclusão de notas | 15/06/2025 |
| Persistência de dados | v0.4.0 | Adição da persistência com `localStorage` | 17/06/2025 |
| Refatoração da UI/UX | v0.5.0 | Melhorias de UI/UX, cores e edição | 18/06/2025 |
| Versão final | v1.0.0 | Melhorias de usabilidade e robustez | 19/06/2025 |

## 9.2 Critérios para Nova Baseline

Uma nova linha de base é criada quando:

- Todos os Pull Requests referentes ao marco foram fechados.
- A documentação foi revisada e atualizada (`CHANGELOG.md`).
- A tag correspondente foi criada no GitHub e uma Release foi publicada.

# 10 Rastreabilidade

## 10.1 Como conectar itens?

A rastreabilidade é assegurada pela vinculação entre Issues, branches, Pull Requests e commits. Exemplo do nosso projeto:

- **Issue #4**: "[FEAT] Implementar funcionalidade de deleção de notas".
- **Branch**: `feature/deletar-nota` criado para a Issue #4.
- **PR #5**: Resolve a Issue #4.
- **Commit**: `feat: Implementa funcionalidade de deleção de notas` faz parte do PR #5.

## 10.2 Matriz de Rastreamento

Tabela 5: Matriz de Rastreabilidade do Projeto Notix

| **Requisito** | **Issue** | **Branch** | **PR** | **Status** |
|---|---|---|---|---|
| Estrutura inicial do projeto | - | `feature/initial-structure` | #1 | Concluído |
| Implementar criação de notas | #2 | `feature/criar-nota` | #3 | Concluído |
| Implementar deleção de notas | #4 | `feature/deletar-nota` | #5 | Concluído |
| Adicionar persistência de dados | #6 | `feature/salvar-notas` | #8 | Concluído |
| Refatorar UI/UX e código | #7 | `refactor/melhorar-ux` | #9 | Concluído |
| Melhorar usabilidade/acessibilidade | #10 | `feature/quality-of-life...` | #11 | Concluído |
| Corrigir bug de contraste da nota | #13 | `fix/contraste-nota` | #12 | Concluído |

# 11 Evidências

A seguir, são apresentadas algumas evidências visuais do processo de Gerência de Configuração aplicado no projeto.

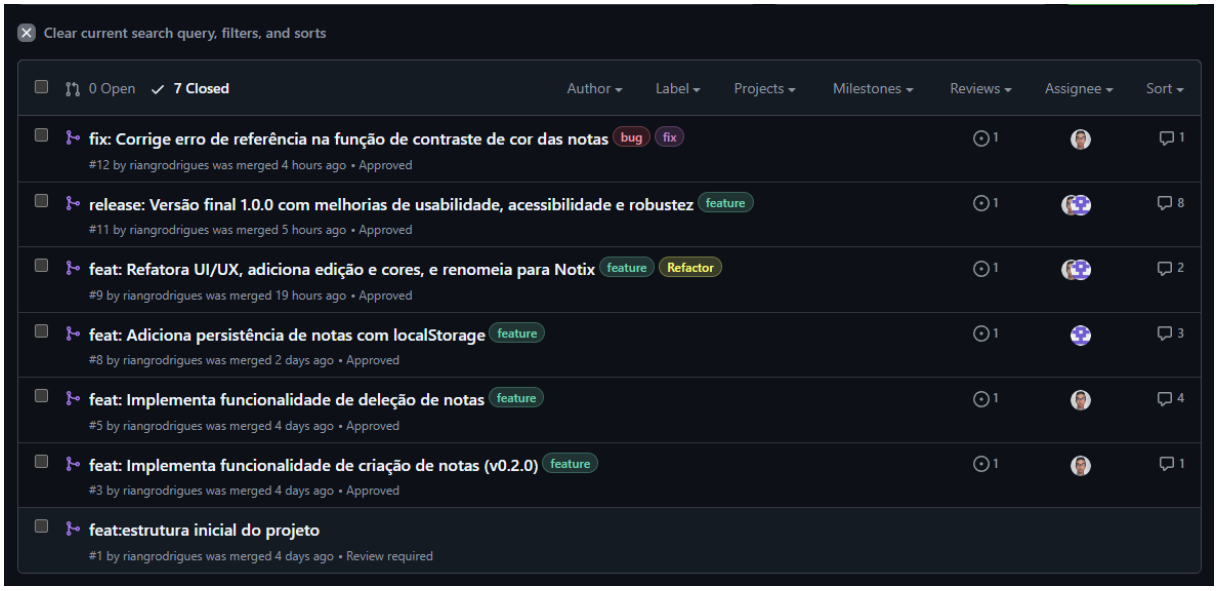


Figura 1: Visão geral das branches utilizadas durante o desenvolvimento.

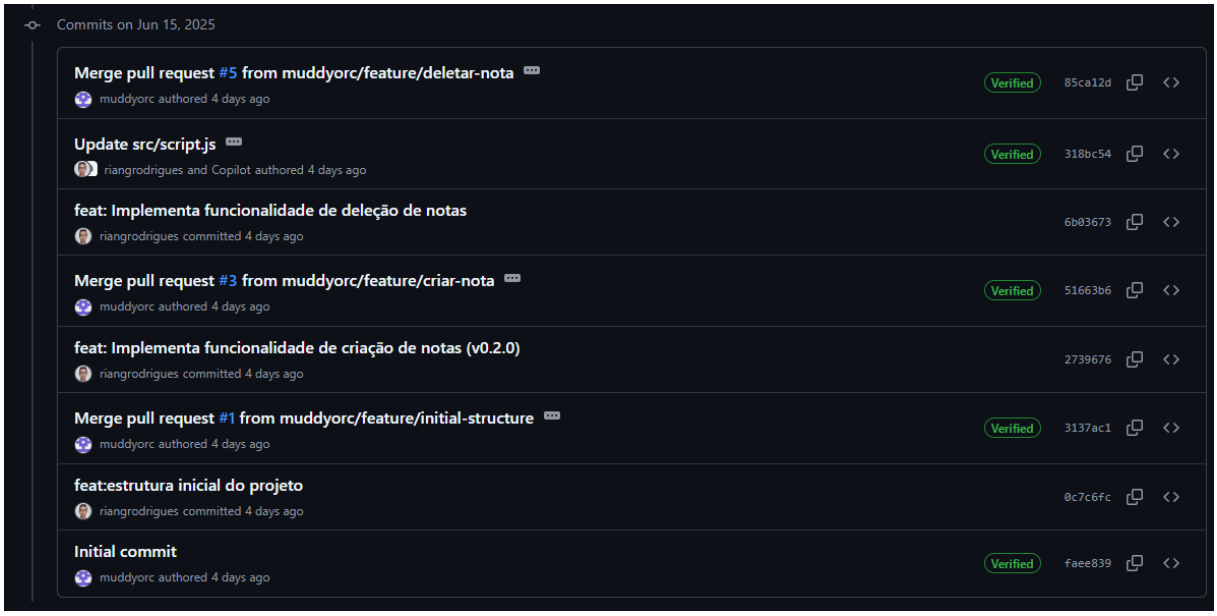


Figura 2: Histórico de commits demonstrando o fluxo de merges e a padronização das mensagens.

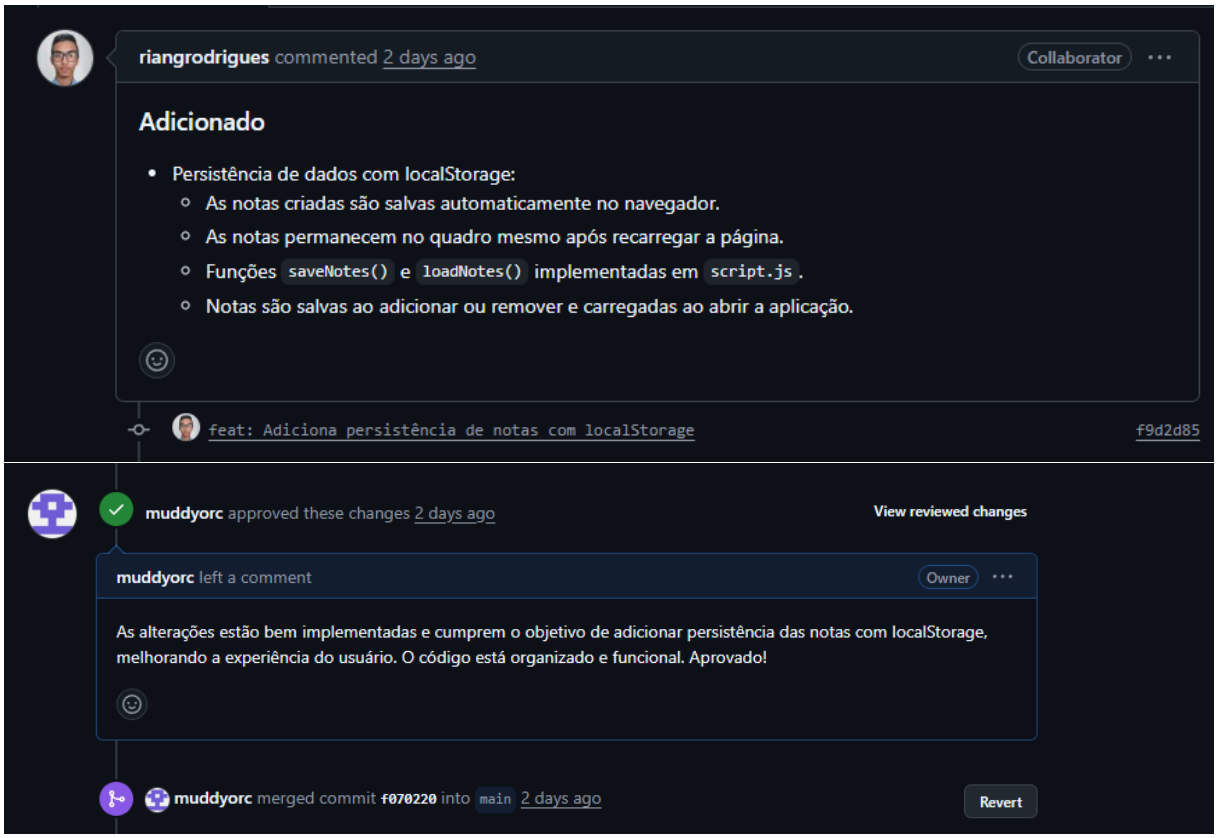


Figura 3: Exemplo de Pull Request com revisão, aprovação e merge.

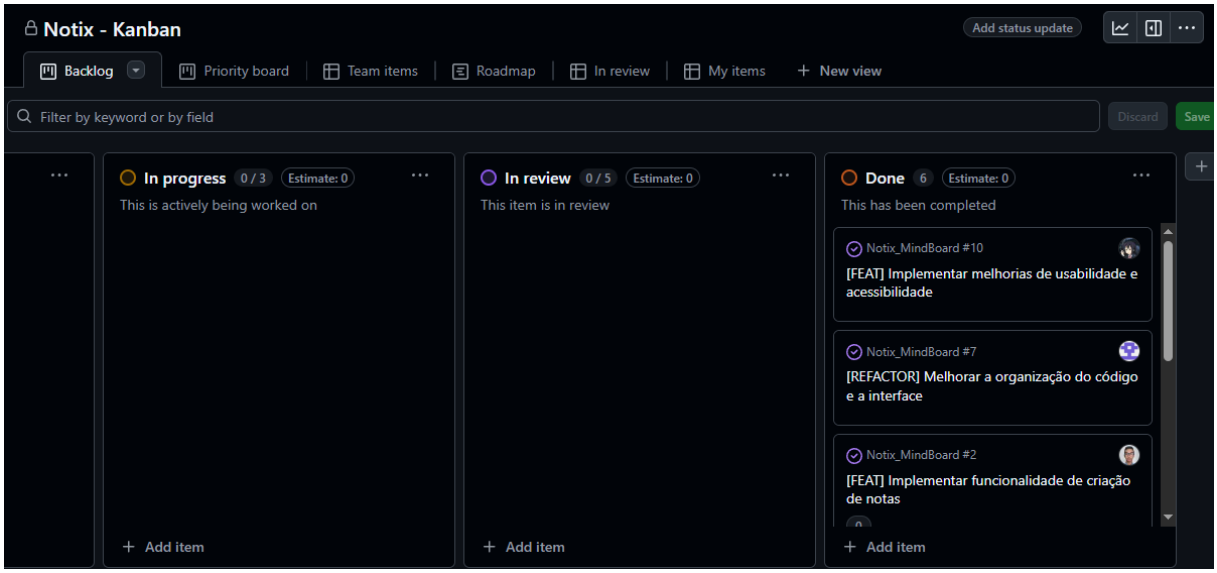


Figura 4: Quadro Kanban no GitHub Projects para acompanhamento do status das tarefas.

# 12 Lições Aprendidas

## 12.1 Reflexão Individual

- **Rian Guedes Rodrigues**: Aprendi a importância de manter um repositório bem organizado e como as práticas de controle de versão podem facilitar o trabalho em equipe, evitando conflitos e retrabalho. O uso das ferramentas do GitHub foi essencial para acompanhar o andamento do projeto.

- **Augusto Wobeto Carvalho**: O projeto reforçou minha habilidade em trabalhar com branches e pull requests, além de destacar a relevância da comunicação clara na hora de propor e revisar mudanças no código.

- **Júlio Cezar Rodrigues Correia**: Durante o desenvolvimento, pude aprimorar meus conhecimentos em versionamento e revisão de código. Percebi como as auditorias constantes ajudam a manter a qualidade do projeto.

- **Eduardo Machado Martins**: Aprendi mais sobre a importância da documentação atualizada e de como o registro das mudanças facilita o entendimento do projeto como um todo. Além disso, melhorei minha prática na escrita e revisão de documentação técnica.

## 12.2 Conclusão do Grupo

O desenvolvimento do **Notix – MindBoard** foi uma experiência enriquecedora para todos os integrantes. Destacamos a organização do repositório, o uso eficiente das branches e pull requests e a integração do time por meio das ferramentas de versionamento. A divisão clara dos papéis contribuiu para um fluxo de trabalho mais fluido, e o uso do GitHub Issues ajudou a manter o controle das atividades e prazos.

Aprendemos a lidar melhor com a comunicação em equipe, especialmente na hora de revisar e aprovar mudanças. A prática constante de versionamento e documentação nos mostrou como pequenas falhas no processo podem gerar retrabalho, reforçando a importância da GCS em projetos colaborativos.

Se tivéssemos que fazer algo diferente, teríamos investido mais tempo no planejamento inicial das funcionalidades e na definição dos critérios de aceitação. Além disso, acreditamos que um maior uso de automação (como actions para testes e builds) poderia ter otimizado ainda mais o processo. Essa experiência nos deixou mais preparados para enfrentar desafios em futuros projetos.

# 13 Cronograma de Desenvolvimento

Tabela 6: Cronograma de Atividades do Projeto Notix

| **Dias** | **Atividades** | **Entregável(eis)** |
|---|---|---|
| 15/06 | Setup inicial, criação da estrutura, implementação da criação e exclusão de notas. | Plano GC, v0.1.0, v0.2.0, v0.3.0 |
| 16-17/06 | Implementação da persistência com localStorage, merge de PRs. | v0.4.0 |
| 18-19/06 | Refatoração da UI/UX, melhorias de usabilidade, ajuste de contraste, melhorias de qualidade de vida, finalização da documentação. | v0.5.0, v1.0.0 (versão final) |