Faculdade: Universidade de Rio verde - UNIRV

Cidade: Rio verde

Curso: Engenharia de Software

Trabalho: Notix - MindBoard

Matéria: Gerência de Configuração de Software

Professor: WAYRONE KLAITON LUIZ SILVA

Integrantes: Rian Guedes Rodrigues, Julio Cezar Rodrigues Correia, Augusto Wobelo, Eduardo Machado Martins.

### **1. Introdução**

#### **1.1 Objetivo do Plano**

Este plano estabelece os procedimentos, ferramentas e responsabilidades para o gerenciamento de configuração dos artefatos do projeto **Notix**. O objetivo é garantir a integridade, rastreabilidade e controle das versões do software e da documentação, aplicando as melhores práticas de GCS como parte da avaliação da disciplina.

#### **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 (post-its virtuais) em um quadro. A aplicação foi projetada para ser simples e intuitiva, com as notas persistindo no navegador do usuário. O público-alvo são usuários que necessitam de uma ferramenta ágil para organização de ideias e tarefas.

#### **1.3 Justificativa da GCS**

Sendo um projeto desenvolvido por uma equipe de 4 integrantes, a aplicação da GCS é fundamental para organizar o fluxo de trabalho, evitar conflitos de código, controlar o histórico de alterações e garantir que todos os membros tenham uma visão clara do progresso e das versões do sistema. A GCS assegura a qualidade e a rastreabilidade do projeto.

### **2. Estrutura Organizacional e Papéis**

#### **2.1 Composição da Equipe**

| **Nome** | **RA** | **GitHub** | **Papel no Projeto** | **Papel na GCS** |
| --- | --- | --- | --- | --- |
| Rian Guedes Rodrigues | [preencher] | https://github.com/riangrodrigues | Dev. Full-Stack | Gerente de Configuração |
| Augusto Wobeto Carvalho | [preencher] | https://github.com/Otsugya | Dev. Frontend | Controlador de Mudanças |
| Júlio Cezar Rodrigues Correia | [preencher] | https://github.com/muddyorc | Dev. Backend/JS | Revisor de Código |
| Eduardo Machado | [preencher] | https://github.com/M4chado | Dev. Frontend | Revisor de Documentação |

#### **2.2 Responsabilidades Específicas**

* **Gerente de Configuração (Rian Guedes Rodrigues):**
  + Responsável por criar e manter a estrutura do repositório no GitHub.
  + Aprova Pull Requests críticos e realiza o merge para a branch main.
  + Cria e gerencia as tags e releases a cada baseline estabelecida.
* **Controlador de Mudanças (Augusto Wobeto Carvalho):**
  + Gerencia o ciclo de vida das Issues no GitHub, desde a criação até o fechamento.
  + Garante que toda mudança no código esteja vinculada a uma Issue.
  + Facilita a comunicação entre os membros sobre as mudanças solicitadas.
* **Revisor de Código / Documentação (Júlio Cezar e Eduardo Machado):**
  + Realizam a auditoria do código-fonte nos Pull Requests, garantindo a conformidade com os padrões.
  + Verificam a clareza e a atualização da documentação (README, CHANGELOG).
  + Validam se os critérios de aceitação das Issues foram atendidos.

### **3. Ferramentas e Ambientes**

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

### **4. Identificação de Itens de Configuração (ICs)**

#### **4.1 Itens Controlados**

| **Tipo** | **Nome / Descrição** | **Localização** |
| --- | --- | --- |
| Código-Fonte | index.html, style.css, script.js | /src/ |
| Documentação | README.md | Raiz do repositório |
| Documentação | CHANGELOG.md | Raiz do repositório |
| Documentação | Plano de GC.pdf (Este documento) | /docs/ |
| Evidências | Screenshots do processo de GCS | /docs/evidencias/ |

#### **4.2 Convenções de Nomes**

* **Branches:**
  + Novas funcionalidades: feature/<nome-da-feature> (ex: feature/deletar-nota)
  + Correções de bugs: fix/<nome-da-correcao> (ex: fix/erro-salvamento)
  + Documentação: docs/<descricao> (ex: docs/ajuste-readme)
* **Commits (Padrão Conventional Commits):**
  + feat: para novas funcionalidades.
  + fix: para correções de bugs.
  + docs: para alterações na documentação.
  + style: para formatação de código.
  + refactor: para refatoração de código sem alteração de funcionalidade.
  + **Exemplo:** feat(notes): adiciona botão para deletar notas (#2)

### **5. Controle de Versão**

#### **5.1 Estratégia de Branching (GitHub Flow)**

* **main**: Contém a versão estável e de produção do projeto. Todo código na main deve ser funcional. Merges só são permitidos via Pull Requests aprovados.
* **Branches de feature e fix**: São criados a partir da main para desenvolver novas funcionalidades ou corrigir bugs. Após a conclusão e revisão, são integrados de volta à main e depois excluídos.

#### **5.2 Commits e Pull Requests (PRs)**

1. Todo desenvolvimento se inicia com uma Issue aberta.
2. Um branch é criado a partir da main, seguindo a convenção de nomenclatura.
3. Os commits devem ser atômicos e seguir a convenção de mensagens, referenciando a Issue correspondente.
4. Ao final do desenvolvimento, um Pull Request é aberto em direção à main.
5. O PR deve ser revisado por pelo menos um outro membro da equipe.
6. Após a aprovação e a passagem de quaisquer verificações automáticas, o PR é "mergeado".

### **6. Controle de Mudança**

O processo de mudança é gerenciado inteiramente pelo GitHub Issues:

1. **Solicitação:** Um membro cria uma Issue detalhando a necessidade (bug, nova funcionalidade, melhoria).
2. **Análise:** O Controlador de Mudanças e a equipe discutem a viabilidade e prioridade da Issue.
3. **Desenvolvimento:** Um responsável é atribuído e inicia o trabalho em um branch separado.
4. **Validação:** A mudança é validada através do processo de Pull Request e code review.
5. **Conclusão:** Após o merge, a Issue é fechada automaticamente.

### **7. Auditoria e Status de Configuração**

* **Auditorias de Código:** Realizadas continuamente durante a revisão de Pull Requests.
* **Auditorias de Documentação:** Realizadas semanalmente para garantir que o README.md e o CHANGELOG.md reflitam o estado atual do projeto.
* **Relatório de Status:** O status do projeto pode ser verificado em tempo real através do painel de Issues e Projects (Kanban) do GitHub.

### **8. Linhas de Base (Baselines)**

As baselines representam versões estáveis e significativas do projeto. Elas são formalizadas através de tags e releases no GitHub.

| **Marco** | **Tag** | **Descrição** | **Data Prevista** |
| --- | --- | --- | --- |
| Estrutura do projeto | v0.1.0 | Repositório inicial com estrutura de pastas e docs. | [preencher] |
| Funcionalidade de Adicionar | v0.2.0 | Usuário consegue criar notas. | [preencher] |
| Funcionalidade de Deletar | v0.3.0 | Usuário consegue deletar notas. | [preencher] |
| Versão Final com Persistência | v1.0.0 | Notas são salvas no navegador (versão de entrega). | [preencher] |

### **9. Rastreabilidade**

A rastreabilidade é garantida pela vinculação entre os artefatos:

* Um **Requisito** (ex: "deletar nota") é traduzido em uma **Issue** (ex: #2).
* A **Issue** é implementada em um **Branch** (ex: feature/deletar-nota).
* O **Branch** resulta em **Commits** (ex: feat: ... (#2)) e um **Pull Request** (ex: PR #5).
* O **Pull Request** é "mergeado", resultando em uma nova versão estável, que pode ser incluída em uma **Release** (ex: v1.0.0).

### **10. Evidências**

*Esta seção deve ser preenchida com prints de tela do repositório GitHub, mostrando:*

* *O histórico de commits com mensagens padronizadas.*
* *Um exemplo de Pull Request com discussão e aprovação.*
* *O quadro de Issues sendo utilizado.*
* *A página de Releases com as tags criadas.*

### **11. Lições Aprendidas**

*Esta seção deve ser preenchida ao final do projeto.*

#### **11.1 Reflexão Individual**

* **Rian Guedes Rodrigues:** [Descrever aprendizados e dificuldades]
* **Augusto Wobeto Carvalho:** [Descrever aprendizados e dificuldades]
* **Júlio Cezar Rodrigues Correia:** [Descrever aprendizados e dificuldades]
* **Eduardo Machado:** [Descrever aprendizados e dificuldades]

#### **11.2 Conclusão do Grupo**

[Descrever em um parágrafo o que o grupo aprendeu como um todo, quais foram os maiores desafios e como a GCS ajudou no projeto.]

### **12. Cronograma**

| **Semana** | **Atividades Principais** | **Entregáveis Esperados** |
| --- | --- | --- |
| 1 | Setup do repositório, preenchimento do Plano de GC, dev da feature 'add-note'. | v0.2.0, Plano de GC versão inicial. |
| 2 | Dev da feature 'delete-note' e 'save-notes', coleta de evidências, gravação do vídeo. | v1.0.0, Plano de GC final, vídeo e slides. |