**1. INTRODUÇÃO**

1.1 OBJETIVO DO PLANO

Este plano estabelece os procedimentos e responsabilidades para controle de itens de configuração, versionamento, auditoria e gestão de mudanças no projeto Easy Menu, como parte da avaliação final da disciplina de GCS.

1.2 ESCOPO DO PROJETO

O Easy Menu é um sistema web para gerenciamento de produtos e geração de cardápios digitais. Permite cadastro, atualização e exclusão de produtos, além da geração de um cardápio visual para o usuário final.

Público-alvo: donos de restaurantes e lanchonetes.

1.3 JUSTIFICATIVA DA GCS

Como o grupo é composto por 5 integrantes com papéis técnicos a Gerência de Configuração é fundamental para garantir controle de versões e rastreabilidade de mudanças, evitando conflitos de código e documentando o ciclo de vida do projeto.

## 2. ESTRUTURA ORGANIZACIONAL E PAPÉIS

2.1 COMPOSIÇÃO DA EQUIPE

| **Nome** | **RA** | **GitHub** | **Papel no Projeto** | **Papel na GCS** |
| --- | --- | --- | --- | --- |
| Gabriel Lopes | 111111 | gabelopesz | Dev Fullstack | Desenvolvedor |
| Lucas Matheus | 222222 | LucasS1L | Dev Backend | Gerente de Configuração |
| Nícollas Silva | 333333 | NicollasSilvaG | Dev Fullstack | Revisor de Documentação |
| Washington Araújo | 444444 | Washington-A | Dev Frontend | Gerente de Configuração |
| Yasmim Andrade | 555555 | Yasmim-Melo | Dev Frontend | Desenvolvedor |

2.2 RESPONSABILIDADES ESPECÍFICAS

**Gerente de Configuração**

* Garantir qualidade antes de merges
* Aprovar merges nas branches principais
* Garantir que o repositório siga as convenções de nomeação
* Manter o documento de GCS atualizado

**Revisor de Documentação**

* Garantir a atualização da documentação
* Corrigir possíveis incoerências entre código e docs
* Fazer revisões antes de releases

**Desenvolvedor**

* Cria, modifica e mantém itens de configuração (como código-fonte, scripts, documentação).
* Segue os procedimentos de versionamento e controle de configuração definidos.

**3. FERRAMENTAS E AMBIENTES**

3.1 PRINCIPAIS FERRAMENTAS USADAS

| **Ferramenta** | **Finalidade** | **Responsável** |
| --- | --- | --- |
| Git | Controle de versão local | Lucas e Washington |
| GitHub | Repositório remoto e tarefas | Lucas e Washington |
| |  | | --- | | WebStorm | | |  | | --- | | Ambiente de desenvolvimento | | Todos |
| Node + React | Execução do frontend | Washington |
| Node + Express | Execução do backend | Lucas |

3.2 ESTRUTURA DO REPOSITÓRIO

easy-menu/

├── backend/ → API em Node

├── frontend/ → Interface web com React

├── docs/ → Documentação (incluindo Plano de GCS)

├── README.md → Apresentação do projeto

├── .gitignore → Boas práticas de versionamento

**4. IDENTIFICAÇÃO DE ITENS DE CONFIGURAÇÃO**

4.1 ITENS CONTROLADOS

| **Tipo** | **Nome** | **Localização** | **Status** | **Responsável** |
| --- | --- | --- | --- | --- |
| Código | app.ts | /backend | Finalizado | Lucas |
| Código | Home.tsx | /frontend | Finalizado | Washington |
| Documento | Plano\_GCS.docx | /docs | Em andamento | Nícollas |

4.2 CONVENÇÕES DE NOMES

**Branches:** feature/<nome>, fix/<bug>, hotfix/<urgência>

**Commits:** Seguir padrão Conventional Commits

exemplo: feat: adiciona tela de login

**5. CONTROLE DE VERSÃO**

5.1 ESTRATÉGIA DE BRANCHING

GitHub Flow com as seguintes regras:

* main: código estável
* feature/\*: novas funcionalidades
* fix/\*: correções
* PRs obrigatórios antes de merge

5.2 COMMITS E PULL REQUESTS

* Commits devem seguir o padrão Conventional Commits
* PRs só serão aceitos por gerentes de configuração

**6. CONTROLE DE MUDANÇA**

6.1 PROCESSO DE MUDANÇA

* Criar uma Issue
* Discutir entre a equipe
* Criar branch específica
* Abrir PR
* Revisar
* Fazer merge
* Fechar a Issue

6.2 TEMPLATE DE ISSUE

### Mudança Solicitada

Tipo: [bug, melhoria, nova funcionalidade]

Descrição: [detalhes]

Impacto: [Baixo / Médio / Alto]

Critérios de Aceitação:

- [ ] Item 1

- [ ] Item 2

**7. AUDITORIA E STATUS DE CONFIGURAÇÃO**

7.1 AUDITORIAS

* **Código:** Revisado via PRs
* **Documentação:** Revisão semanal pelo Revisor de Documentação
* **Estrutura de pastas:** Conferida antes de cada release

7.2 RELATÓRIO DE STATUS

markdown

# Semana 2

- Commits: 12

- Issues abertas: 2

- Releases: v0.1.0 criada

- Problemas: conflito em feature/login resolvido

**8. LINHAS DE BASE (BASELINES)**

8.1 MARCOS E RELEASES

| **Marco** | **Tag** | **Descrição** | **Data** |
| --- | --- | --- | --- |
| Estrutura inicial | v0.1.0 | Versão base do repositório e docs | 15/06 |
| Segunda versão | v0.1.1 | Versão com documentação atualizada | 18/06 |

8.2 CRITÉRIOS PARA NOVA BASELINE

* PRs fechados
* Doc atualizada
* Tag criada no GitHub

**9. RASTREABILIDADE**

9.1 COMO CONECTAR ITENS?

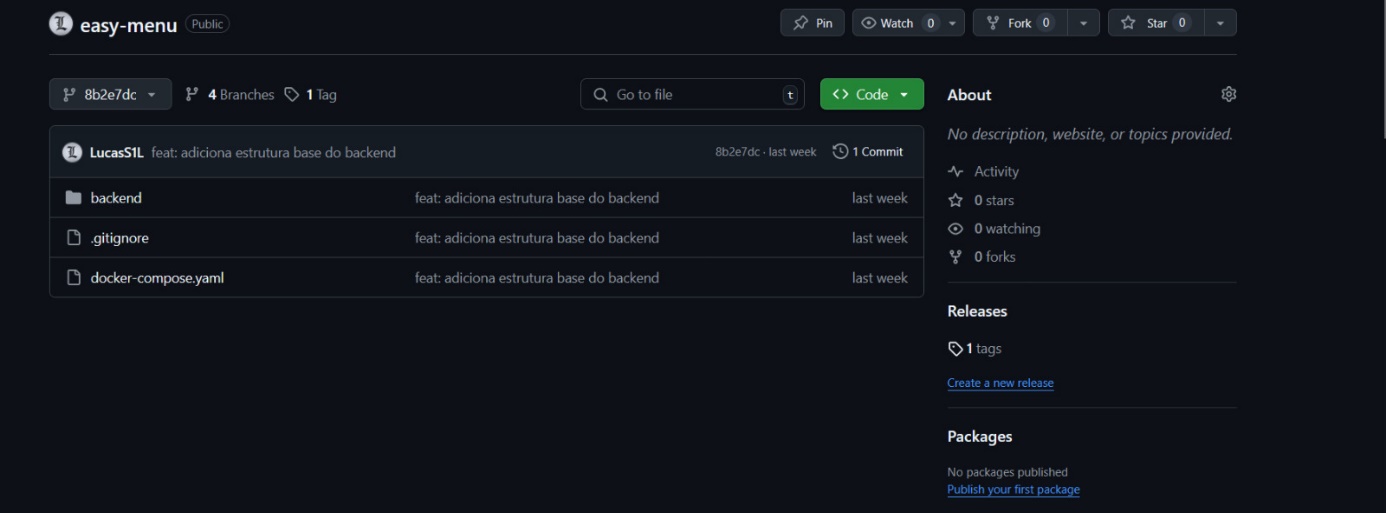
* Issue #2 cria feature/primeira-logo
* PR #3 resolve a issue
* Commit final: feat: adiciona logo na pagina inicial

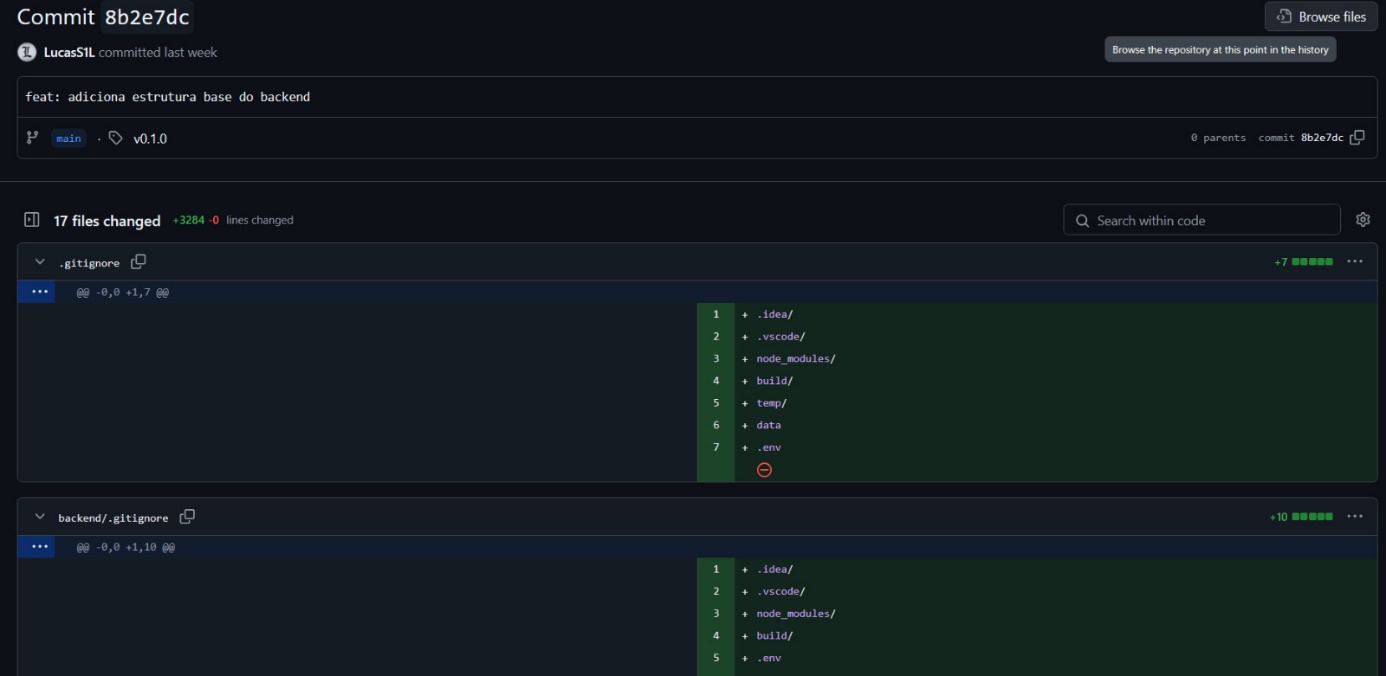
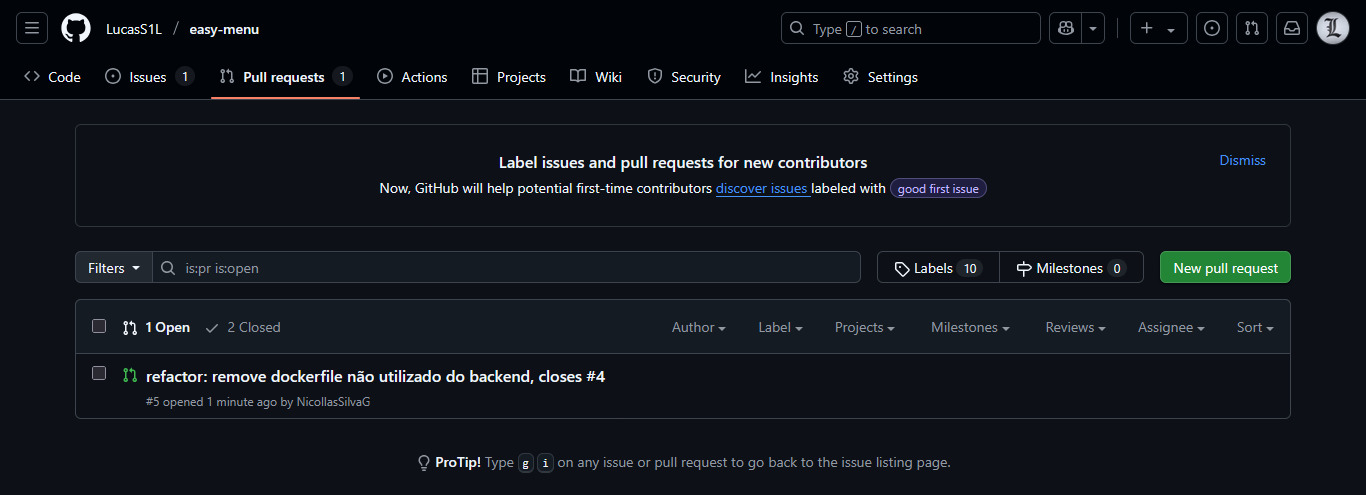
9.2 MATRIZ DE RASTREAMENTO

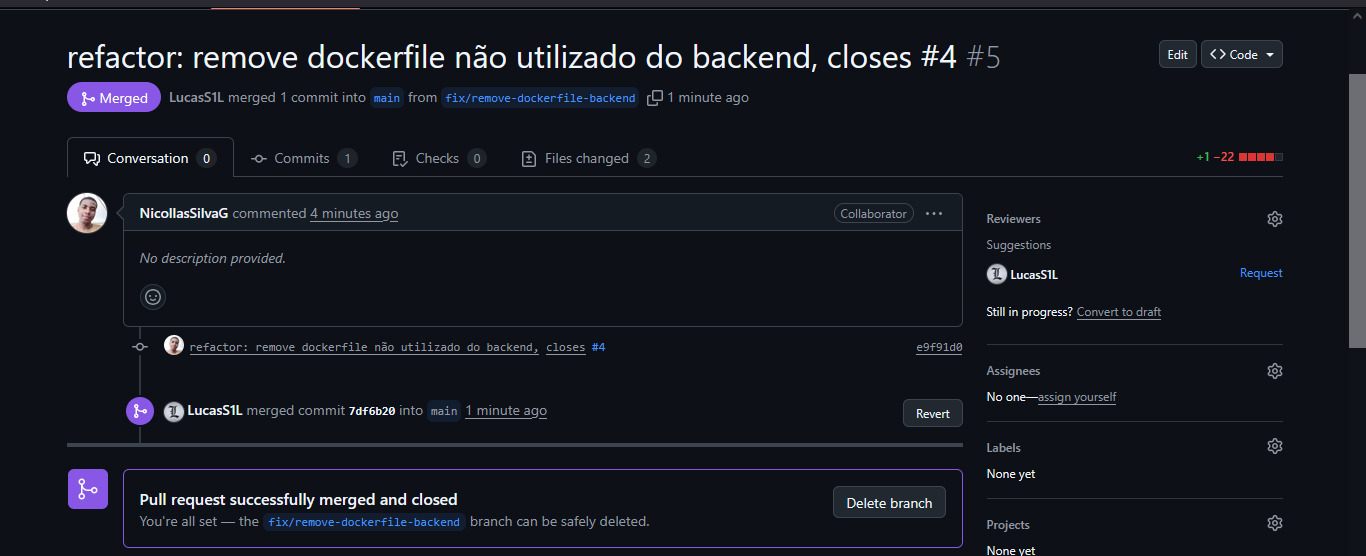
| **Requisito** | **Issue** | **Branch** | **PR** | **Status** |
| --- | --- | --- | --- | --- |
| Adicionar Logo na Tela Inicial | #2 | feature/primeira-logo | #3 | Concluído |
| Remover Dockerfile | #4 | Fix/remove-dockerfile-backend | #5 | Concluído |

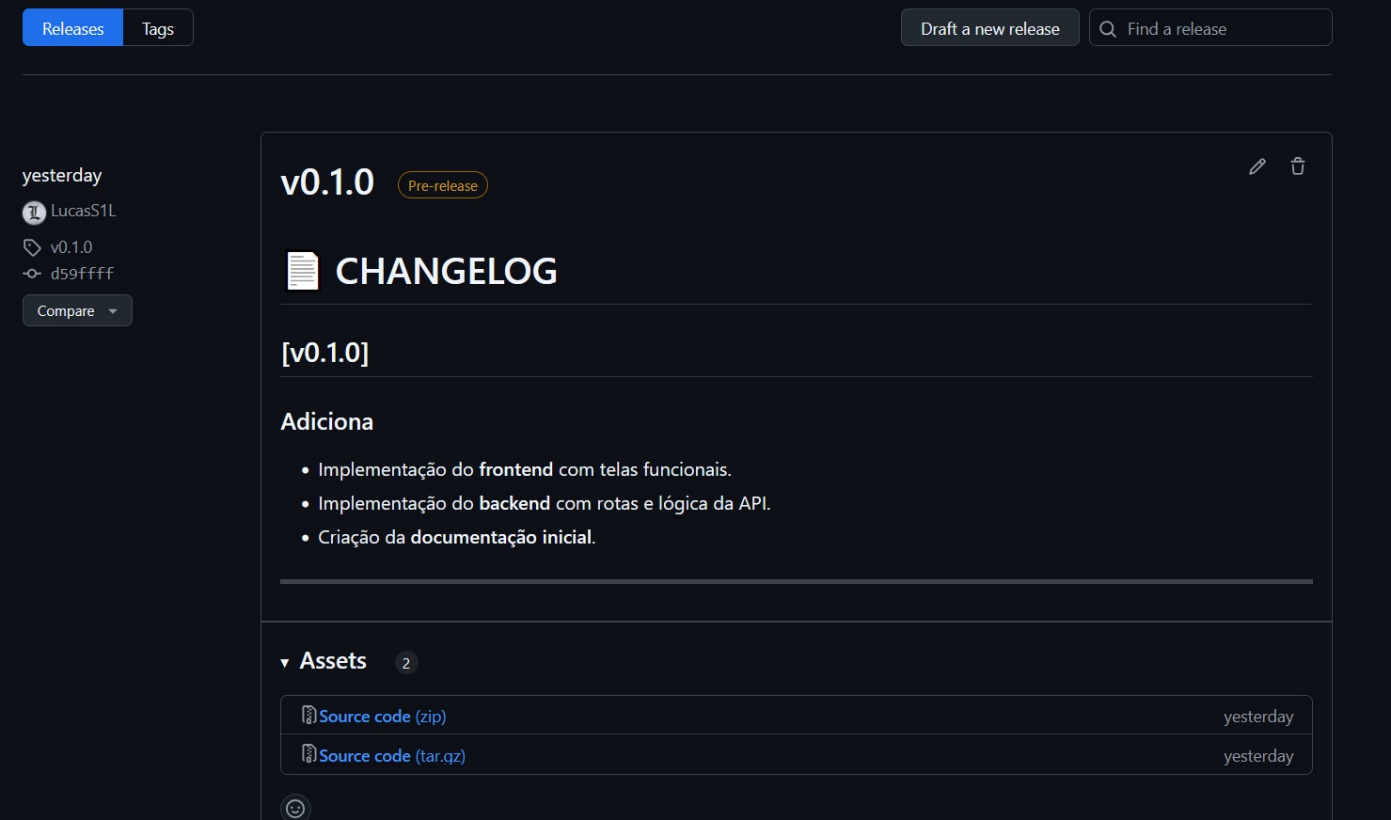
**10. EVIDÊNCIAS**

**10.1 Imagens**









**10.2 Organização sugerida**

📁 evidencias/

├── semana-01/

│ └── print-repo.png

├── semana-02/

│ └── print-pr-login.png

**11. Lições Aprendidas**

**11.1 Reflexão Individual**

**Lucas (Gerente GC)**  
Aprendi a usar tags e releases no GitHub e a criar issues.

**Washington (Gerente GC)**  
Aprendi a importância de manter o repositório organizado e padronizado e também a realização de PR e merges.

**Nícollas (Revisor de Documentação)**  
Aprendi a criar issues e notei a importância de manter a documentação atualizada.

**Gabriel (Desenvolvedor)**  
Aprendi a seguir práticas de configuração e utilização de commits padronizados.

**Yasmim (Desenvolvedor)**  
Aprendi a criar e organizar branches para cada tarefa, facilitando o controle do desenvolvimento.

**11.2 Conclusão do Grupo**

No nosso projeto, conseguimos manter uma boa organização no repositório, com commits bem estruturados. Aprendemos a importância de manter a documentação sempre atualizada e de integrar as práticas de GCS no nosso fluxo de trabalho. A comunicação dentro da equipe foi essencial para resolver conflitos e alinhar as tarefas. Contudo, poderíamos ter melhorado a sincronização entre os membros antes de novos commits. Além disso, a compatibilidade de ferramentas poderia ter sido mais bem gerenciada anteriormente.

**12. Cronograma**

| **Semana** | **Atividades** | **Entregável** |
| --- | --- | --- |
| 1 | Setup, plano GC | Plano-gc.docx |
| 2 | Codificação | app.ts e Home.tsx |