# **Política de Gerência de Configuração**

## **1. Informações Gerais**

**Repositório do Grupo:**

* **URL:** [GitHub - Umaps](https://github.com/ArthurAlvesPereira/Umaps)
* **Estrutura do Repositório:**
  + **Mainline:** Contém o código de desenvolvimento ativo.
  + **Branches de Desenvolvimento:** Criados para grandes evoluções ou correções específicas.
  + **Tags:** Usadas para identificar releases específicas.
  + **Diretórios Principais:**
    - **/src:** Código-fonte do projeto.
    - **/docs:** Documentação.
    - **/scripts:** Scripts de build e automatização.
    - **/tests:** Testes automatizados.

## **2. Itens de Configuração**

### **2.1. Itens de Configuração do Projeto**

Os seguintes artefatos serão considerados itens de configuração e estarão sujeitos ao controle de versão:

* Código-fonte (/src)
* Documentação (/docs)
* Scripts de build e automação (/scripts)
* Testes automatizados (/tests)
* Arquivos de configuração do ambiente e ferramentas utilizadas no projeto.

## **3. Gerenciamento de Branches**

### **3.1. Quando abrir uma Branch?**

* **Critério:** Uma branch será aberta para cada tarefa ou por desenvolvedor.
* **Nomenclatura das Branches:**
  + Branches de tarefas: feature/<nome\_da\_tarefa>
  + Branches de correções: fix/<descrição\_da\_correção>
  + Branches de desenvolvedores: dev/<nome\_do\_dev>

### **3.2. Base das Branches**

* Todas as branches devem ser baseadas na branch principal (main).

## **4. Controle de Alterações**

### **4.1. Merge de Branches**

* **Quando realizar o Merge:** Após a conclusão e validação das tarefas/correções implementadas.
* **Quem realiza o Merge:** O merge será realizado por um dos membros designados do Comitê de Controle de Configuração, e deve ser revisado por, no mínimo, uma outra pessoa do grupo.
* **Tratamento de Conflitos:**
  + Conflitos de merge devem ser resolvidos colaborativamente pelo(s) desenvolvedor(es) responsável(is) pela branch e pela pessoa que realiza o merge.
  + Caso o conflito seja complexo, a decisão final será tomada pelo Comitê de Controle de Configuração.

## **5. Mensagens de Commit**

### **5.1. Padrão de Mensagens**

* Cada commit deve conter uma mensagem que descreva a modificação realizada e referencie a tarefa correspondente.
* **Padrão de Mensagens:** [Tipo]: Breve descrição da mudança - (Ref. #ID\_Tarefa)
  + Exemplo: feat: Adiciona funcionalidade de login - (Ref. #23)

## **6. Versionamento e Releases**

### **6.1. Padrão de Versionamento**

* O projeto seguirá o padrão de **versionamento semântico** (SemVer):
  + MAJOR.MINOR.PATCH

### **6.2. Lançamento de Releases**

* **Quando é lançada uma release:**
  + Uma release será lançada ao final de um ciclo de desenvolvimento, após a aprovação pelo Comitê de Controle de Configuração.
* **Quem aprova a release:**
  + A aprovação será feita pelo Comitê de Controle de Configuração.
* **Identificação de Elementos de Release:**
  + Releases serão identificadas utilizando tags no repositório, seguindo o formato vMAJOR.MINOR.PATCH.

## **7. Controle de Mudanças**

### **7.1. Processo de Controle de Mudanças**

* Alterações nos itens de configuração devem ser iniciadas por uma solicitação formal e aprovada pelo Comitê de Controle de Configuração.
* Cada mudança passará por uma **avaliação** que inclui análise de custos, cronograma e esforço.
* O plano de mudanças será documentado e revisado antes da implementação.