**Documentação de Gerenciamento de Configuração**

**1. Introdução**

**1.1 Objetivo**

Este documento descreve as práticas de gerenciamento de configuração para o projeto AEP-IA-CYBERSECURITY, incluindo a estrutura de branches, o fluxo de trabalho de desenvolvimento e os procedimentos para anotações de commits.

**1.2 Escopo**

Esta documentação se aplica a todos os desenvolvedores e stakeholders envolvidos no desenvolvimento do projeto, utilizando GitHub como plataforma de versionamento.

**2. Estrutura de Branches**

**2.1 Branch Principal**

* **main**: Esta é a branch principal que contém a versão estável do código. Somente código testado e aprovado deve ser mesclado nesta branch.

**2.2 Branches de Desenvolvimento**

* **Branch de Dev**: Cada desenvolvedor deve criar uma branch de desenvolvimento a partir da branch main, nomeando-a como dev/joao. Esta branch é destinada ao trabalho contínuo do desenvolvedor.

**2.3 Branches de Funcionalidade**

* **Branch de Feature**: Quando um desenvolvedor começa a trabalhar em uma nova funcionalidade, ele deve criar uma branch de funcionalidade a partir da sua branch de dev, nomeando-a como feature/adicionar-autenticacao.

**3. Fluxo de Trabalho**

**3.1 Criação de Branches**

1. O desenvolvedor cria uma branch de dev a partir da branch main.
2. Para cada nova funcionalidade, o desenvolvedor cria uma branch de feature a partir de sua branch de dev.

**3.2 Desenvolvimento**

* Os desenvolvedores devem realizar commits frequentemente, garantindo que cada commit inclua mensagens descritivas e siga o padrão de anotação.

**3.3 Mesclagem para a Branch Principal**

1. Após a conclusão da funcionalidade, o desenvolvedor deve abrir um Pull Request (PR) da branch de feature para a branch main.
2. O PR deve ser revisado por pelo menos um outro desenvolvedor antes de ser aprovado e mesclado à branch main.

**3.4 Monitoramento de Releases**

* Um membro designado da equipe será responsável por monitorar a branch main e coordenar o processo de lançamento das releases.

## 4. Anotações de Commits

### 4.1 Tipos de Mudanças

Os commits de manutenção devem utilizar a anotação fix: para indicar que se trata de uma correção ou alteração. Além disso, as mudanças devem ser categorizadas no GitHub com labels que indicam o tipo de manutenção:

* **Label: Corretiva**: Para correção de bugs.
* **Label: Adaptativa**: Para adaptações a novos requisitos ou ambientes.
* **Label: Perfectiva**: Para melhorias na funcionalidade existente.
* **Label: Preditiva**: Para mudanças antecipatórias ou preparatórias.

### 4.2 Uso de Labels no GitHub

No GitHub, os commits de manutenção devem ser acompanhados de labels apropriadas para facilitar a categorização e rastreamento. Quando um Pull Request é aberto, o desenvolvedor deve adicionar a label correspondente ao tipo de manutenção:

* **Adicionar label Corretiva**: Para commits que corrigem bugs.
* **Adicionar label Adaptativa**: Para adaptações a novos requisitos.
* **Adicionar label Perfectiva**: Para melhorias.
* **Adicionar label Preditiva**: Para mudanças antecipatórias.

### 4.3 Importância das Anotações e Labels

O uso da anotação fix: nos commits e das labels no GitHub ajuda a manter um registro claro e organizado das alterações, facilitando a revisão do histórico e a comunicação entre a equipe sobre a natureza das mudanças.