**Plano de Configuração**

**1. Objetivo do Plano de Configuração**

Este plano de configuração estabelece diretrizes e práticas para garantir a organização, controle de mudanças e qualidade do software desenvolvido. Ele é fundamental para melhorar a comunicação entre a equipe e facilitar a manutenção do projeto.

**2. Importância do Plano de Configuração**

* **Controle de Mudanças**: Gerenciar e documentar alterações de forma eficaz, evitando conflitos e retrabalhos.
* **Consistência**: Estabelecer padrões claros e práticas que garantem uniformidade na nomenclatura, estrutura e implementação de componentes.
* **Rastreabilidade**: Facilitar a identificação e o rastreamento de itens de configuração, crucial para auditorias e manutenção.
* **Comunicação**: Melhorar a comunicação entre a equipe, fornecendo um guia claro que todos podem seguir.
* **Redução de Riscos**: Minimizar a probabilidade de erros e problemas, aumentando a qualidade do produto final.
* **Eficiência**: Acelerar o desenvolvimento e a manutenção ao oferecer um quadro organizado para gerenciar componentes e mudanças.

**3. Estrutura de Pastas**

Mobile

- lib

- screens

- ip\_detection\_screen.dart

- menu\_screen.dart

- url\_detection.dart

- main.dart

- test

- ip\_detection\_screen\_test.dart

- url\_detection\_test.dart

- menu\_screen\_test.dart

**4. Nomenclatura**

**4.1 Nome de Variáveis**

* Utilizar nomes altamente intuitivos, exemplo: resultadoMultiplicacao = 10 \* 5.

**4.2 Nome dos Arquivos e Pastas**

* Se o nome do arquivo conter espaço, utilizar ‘\_’ (underline).
* Nome do arquivo + ‘\_(nome da pasta)’, exemplo:
  + ip\_detection\_screen.dart (dentro da pasta screens).
* Nomes das pastas devem ser altamente intuitivos, exemplo: ‘screens’.

**5. Procedimentos**

* **Utilização do Git**: Todas as alterações devem ser versionadas utilizando Git.
* **Utilização do GitHub**: O repositório deve estar hospedado no GitHub para facilitar a colaboração e o controle de versões.
* **Documentação de Mudanças**: Todas as alterações significativas devem ser documentadas.
* **Revisão de Código**: Processo de revisão de código: a pessoa que fez o código não poderá revisar, apenas outro membro da equipe, para garantir que seja melhor avaliado.
* **Gerenciamento de Dependências**: Documentar e controlar as dependências do projeto, especificando como e quando atualizá-las.
* **Backup e Recuperação**:   
  - Frequência de Backup: **Backup Incremental**: Diariamente (ex.: todos os dias às 23:00).  
  - Realizar testes de recuperação trimestrais para garantir que os backups estão funcionando corretamente e que a equipe está familiarizada com os procedimentos.  
  - Manter documentação detalhada de todos os backups realizados, incluindo data, tipo de backup, e local de armazenamento.  
  - Armazenamento de Backup: Utilização de serviços de armazenamento na nuvem.

**6. Estratégia de Testes**

* **Testes Unitários**: Criar testes para cada unidade de código, garantindo que as funções operem como esperado.
* **Testes de Integração**: Garantir que os diferentes módulos funcionem juntos corretamente.
* **Testes de Interface do Usuário**: Validar a usabilidade e a interação do usuário com a aplicação.

**7. Versão do Software**

* Versão atual do Software: **1.0**

**8. Versão da Documentação do Plano de Configuração**

* Versão atual da Documentação do Plano de Configuração: **1.0**