**PLANO DE GERENCIAMENTO DE CONFIGURAÇÃO PARA O APLICATIVO SMS SEGURO**

VERSÃO 1.0

MARINGÁ, 2024

1. **INTRODUÇÃO**

**1.1. OBJETIVO**

Este documento tem como objetivo estabelecer diretrizes para a gestão eficaz dos itens de configuração do projeto de desenvolvimento do aplicativo ‘SMS seguro’, garantindo que o desenvolvimento seja feito de forma uniforme e que todos os componentes sejam monitorados e controlados de forma adequada.

**1.2. ESCOPO**

O escopo deste documento é abranger os itens de configuração, dentre eles, o código fonte, a documentação externa consultada, os arquivos de configuração e as dependências, bem como estabelecer as políticas de versionamento e protocolos para o controle de versões e de registro de mudanças, detalhar o processo para a solicitação de mudança e como será feita a sua avaliação e aprovação. Por fim, este documento definirá as ferramentas de controle de versão, de gerenciamento de mudanças e a documentação do projeto.

1. **ITENS DE CONFIGURAÇÃO**

**2.1. CÓDIGO FONTE**

Os arquivos de código fonte estarão disponíveis no link:

**2.2. DOCUMENTAÇÃO EXTERNA**

No presente projeto foram utilizadas as seguintes ferramentas externas:

**2.3. ARQUIVOS DE CONFIGURAÇÃO**

No ambiente de configuração foram utilizados os seguintes arquivos de configuração:

|  |  |
| --- | --- |
| .env |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**2.4. DEPENDÊNCIAS**

Foram instaladas as seguintes dependências:

|  |  |
| --- | --- |
| Dependência | Versão |
|  |  |
|  |  |
|  |  |
|  |  |

1. **POLÍTICAS DE VERSIONAMENTO**

**3.1. CONTROLE DE VERSÕES**

**3.1.1. Repositório**

O código-fonte e os arquivos associados serão gerenciados em um repositório Git hospedado no Github, o qual poderá ser acessado através do link:

Todas as operações de versionamento, colaboração e controle de mudanças serão realizadas através dessa plataforma, que também permite o gerenciamento de *issues*, *pull requests* e fornece integração contínua.

**3.1.2. Versões**

**3.2. REGISTRO DE MUDANÇAS**

**3.2.2. Padronização de commits**

Cada commit deve conter uma mensagem clara e descritiva que explique a alteração realizada, facilitando o entendimento e rastreamento das mudanças no projeto.

Os commits deverão observar o padrão:

Número do card + tipo de alteração + mensagem descritiva.

Ex: 01-feat-desenvolvimento-tela-inicial.

Os tipos de alterações serão classificados conforme a seguinte convenção:

|  |  |
| --- | --- |
| ADD | Para adição de arquivos ou dependências externas ao projeto |
| CHORE | Para tarefas rotineiras ou mudanças que não afetam diretamente a funcionalidade do código |
| DOCS | Para alterações ou melhorias na documentação |
| FEAT | Para inclusão de novas funcionalidades |
| FIX | Para correção de bugs |
| IMPROVE | Para melhorias incrementais em funcionalidades existentes, como ajustes de usabilidade ou eficiência |
| MERGE | Para commits resultantes de fusões de branches |
| REFACTOR | Para refatorações de código |
| TEST | Para adição ou alteração de testes automatizados |
| STYLE | Para mudanças de estilo no frontend que não afetem a funcionalidade do código |

1. **CONTROLE DE MUDANÇAS**

**4.1. PROCESSO DE SOLICITAÇÃO DE MUDANÇA**

Caso haja necessidade de mudanças no projeto, deverá ser criado um novo card no Trello detalhando a mudança e a justificativa.

**4.2. AVALIAÇÃO E APROVAÇÃO**

As mudanças serão avaliadas pelos integrantes do grupo e caso sejam aprovadas, serão implementadas no ambiente de desenvolvimento e testadas. Após o teste, as mudanças realizadas serão documentadas, devendo conter a descrição, o responsável e a data da mudança.

1. **FERRAMENTAS E RECURSOS**

**5.1. FERRAMENTAS DE CONTROLE DE VERSÃO**

Git e Github.

**5.2. SISTEMA DE GERENCIAMENTO DE MUDANÇA**

Trello

**5.3. DOCUMENTAÇÃO**

1. **CRONOGRAMA**