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

VERSÃO 1.1

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 e documentação estarão disponíveis no link: [Github\_projeto](https://github.com/cassiab13/aep-6)

**2.2. FERRAMENTAS EXTERNAS**

O projeto fará uso das seguintes ferramentas externas:

1. [API do Virus Total](https://docs.virustotal.com/): utilizada para verificar a segurança das URLs fornecidas pelo usuário.

**2.3. ARQUIVOS DE CONFIGURAÇÃO**

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

|  |  |
| --- | --- |
| .env | Contém as variáveis de ambiente. Neste projeto foram utilizados:  “API\_URL” = [URL da API]  “API\_KEY” = [Chave gerada para acesso à API] |
| pubspec.yaml | Função: Gerencia dependências do projeto Flutter/Dart e contém as seguintes informações:  **Nome e Descrição do Projeto**: Identificação do aplicativo.  **Versão do Aplicativo**: Versão atual do projeto.  **Dependências Necessárias**: Lista das bibliotecas e pacotes utilizados no projeto.  **Configurações Específicas do Projeto**: Informações adicionais relevantes, como assets e configurações do Flutter. |
|  |  |

**2.3.1 Instruções de uso**

**2.3.1.1 Configuração do arquivo .env**

1. No diretório backend-api do projeto, crie um arquivo chamado .env.

2. Preencha as variáveis do ambiente necessárias no seguinte formato:

API\_URL=[Insira a URL da API]

API\_KEY=[Insira a chave da API]

3. No código use a biblioteca dotenv para carregar as variáveis do .env:

from dotenv import load\_dotenv

load\_dotenv()

**2.3.1.1 Configuração do arquivo pubspec.yaml**

1. Dependências: este arquivo gerencia as dependências do projeto Flutter. Para adicionar novas dependências, inclua na seção *dependencies*.

2. Gerenciamento de assets: definir arquivos estáticos

3. Executar comando: [flutter pub get] para baixar e instalar as dependências.

**2.4. DEPENDÊNCIAS**

No back-end foram instaladas as seguintes dependências, conforme listado no documento requirements.txt:

|  |  |
| --- | --- |
| Dependência | Versão |
| asgiref | 3.8.1 |
| certifi | 2024.8.30 |
| charset-normalizer | 3.3.2 |
| colorama | 0.4.6 |
| Django | 5.1.2 |
| django-cors-headers | 4.5.0 |
| idna | 3.10 |
| iniconfig | 2.0.0 |
| joblib | 1.4.2 |
| numpy | 2.1.2 |
| packaging | 24.1 |
| pandas | 2.2.3 |
| pluggy | 1.5.0 |
| pytest | 8.3.3 |
| python-dateutil | 2.9.0.post0 |
| python-dotenv | 1.0.1 |
| pytz | 2024.2 |
| requests | 2.32.3 |
| scikit-learn | 1.5.2 |
| scipy | 1.14.1 |
| six | 1.16.0 |
| sqlparse | 0.5.1 |
| threadpoolctl | 3.5.0 |
| tzdata | 2024.2 |
| urllib3 | 2.2.3 |

No front-end foram instaladas as seguintes dependências:

|  |  |
| --- | --- |
| Dependência | Versão |
| http | 0.13.6 |
| telephony | 0.2.0 |

Além disso, o front-end faz uso do Flutter SDK versão 3.6.0-255.0.dev, que oferece suporte às funcionalidades essenciais para a interface gráfica e outras funcionalidades do aplicativo.

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.

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**

Cada versão do projeto será identificada de com o padrão **Major.Minor.Patch**, seguindo o padrão abaixo:

* **Major:** grandes alterações ou alterações incompatíveis com versões anteriores.
* **Minor:** para adição de funcionalidades que mantenham a compatibilidade com versões anteriores.
* **Patch:** para correções de bugs e pequenas melhorias.

**3.2. REGISTRO DE MUDANÇAS**

**3.2.1. 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**

Para solicitar mudanças no projeto, deverá ser criado um *card* no Trello detalhando a mudança e a justificativa para a mudança. O *card* deve ser rotulado como “Solicitação de mudança”.

**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**

O projeto utilizará Git como sistema de controle de versão, hospedado na plataforma Github. Essa ferramenta permite o versionamento do código-fonte e o gerenciamento colaborativo do projeto.

**5.2. SISTEMA DE GERENCIAMENTO DE MUDANÇA**

O gerenciamento de mudanças será realizado utilizando o Trello, onde cada solicitação será registrada em um *card* que incluirá os detalhes da mudança, como descrição e justificativa.

**5.3. DOCUMENTAÇÃO**

Toda documentação do projeto será mantida no repositório do Github, garantindo a atualização constante e o acesso por todos os membros da equipe.

1. **CRONOGRAMA**

O cronograma a seguir descreve as principais atividades e marcos do projeto SMS Seguro:

|  |  |
| --- | --- |
| Cronograma “SMS Seguro” | |
| Data | Atividade |
| 09/09 | Escolha do projeto e elaboração do resumo. |
| 16/09 | Definição das funcionalidades e discussão sobre telas, cores e logo. |
| 23/09 | Criação dos cards no Trello e divisão de tarefas. |
| 30/09 | Elaboração do Plano de Gerenciamento de Configuração. |
| 14/10 a 04/11 | Desenvolvimento de funcionalidades, algoritmos de aprendizado de máquina e estrutura do front-end. |
| 11/11 | Entrega final do projeto |