# MANUAL DE RODAGEM

# AlagAlert

# 1. Objetivo e escopo

O avaliador deve conseguir executar a API FastAPI (backend) e o aplicativo Flutter (frontend) localmente, com os mesmos dados utilizados pelo grupo. Este documento lista todos os passos obrigatórios, variáveis de ambiente e comandos esperados.

# 2. Estrutura do repositório

- **backend/** → API FastAPI responsável por geocodificação, consulta de previsão e cálculo de risco.
- **frontend/** → App Flutter utilizado pelos avaliadores (Android/iOS/Web/Desktop).
- **docker-compose.yaml** → opcional para subir tudo em contêineres (ajustes descritos na Seção 8).

Não há banco de dados dedicado; todo armazenamento é em memória ou vindo de APIs externas (Open-Meteo, Nominatim, IBGE).

# 3. Pré-requisitos

## 3.1 Sistema operacional (qualquer um dos abaixo funciona):

- Windows 11 ou 10 22H2 com PowerShell 5.1+.
- macOS 13 Ventura+ com Homebrew.
- Ubuntu 22.04+ (ou distro equivalente com systemd).

## 3.2 Ferramentas comuns:

Acesso à internet liberado para https://api.open-meteo.com, https://nominatim.openstreetmap.org e https://servicodados.ibge.gov.br.

## 3.3 Backend:

- Python 3.13.9
- pip 23+ (vem com o Python oficial).
- (Opcional) virtualenv para isolamento do ambiente.

## 3.4 Frontend:

- Flutter SDK 3.35.7 (Dart 3.9.2).
- Android Studio Giraffe+ com Android SDK 34 e um emulador habilitado; ou Xcode 15+ para iOS; ou Chrome para execução web.
- Drivers USB habilitados caso utilize um dispositivo físico.

### 3.5 Recursos opcionais:

- Docker 24+ e Docker Compose v2, caso opte por utilizar contêineres.

## 4. Passo a passo resumido (checklist)

1. cd alagAlert
2. Configurar a API (Seção 5) e garantir que http://localhost:8000/health responde {"ok": true}.
3. Configurar o app Flutter (Seção 6) apontando para a API.
4. Validar as telas listadas na Seção 7 (resultados e mapa).

# 5. Backend FastAPI

## 5.1 Preparar ambiente virtual (recomendado)

**Linux/macOS:**

```
cd backend python3 -m venv .venv source .venv/bin/activate python -m pip install --upgrade pip
```

**Windows (PowerShell):**

```
cd backend python -m venv .venv .\.venv\Scripts\Activate.ps1 python -m pip install --upgrade pip
```

## 5.2 Variáveis de ambiente

Copie o arquivo de exemplo e ajuste os valores:

```
cp .env.example .env # Linux/macOS copy .env.example .env # Windows
```

| Variável | Obrigatório | Descrição |
|---|---|---|
| BRASIL_ABERTO_API_KEY | Não | Chave para API Brasil Aberto; sem ela a rota /risk/neighborhoods usa mock |
| HOST | Não | Interface escutada pelo uvicorn (padrão 0.0.0.0) |
| PORT | Não | Porta do backend (padrão 8000) |
| RATE_LIMIT | Não | Limite SlowAPI (padrão 60/minute) |
| WEB_DIR | Não | Caminho opcional para build Flutter Web a ser servido pela API |

## 5.3 Instalar dependências

Com o ambiente ativo e dentro de backend/ execute:

```
pip install -r requirements.txt
```

## 5.4 Executar o servidor

Modo desenvolvimento (recarga automática):

```
uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
```

Ou utilizando o módulo python:

```
python -m uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
```

A API fica disponível em http://localhost:8000 e a documentação Swagger/OpenAPI em http://localhost:8000/docs.

## 5.5 Testes rápidos do backend

Com o servidor em execução:

```
curl http://localhost:8000/health curl "http://localhost:8000/risk/by-city?city=Santos&uf;=SP"
```

Respostas 200 indicam que os acessos às APIs externas estão funcionando.

# 6. Frontend Flutter

## 6.1 Validar instalação do Flutter

```
flutter --version flutter doctor
```

Garanta que todos os checkboxes estão verdes (especialmente Android toolchain e Chrome se for usar Web).

## 6.2 Configurar a origem da API

O app consome a constante `ApiService.baseUrl`, que por padrão usa `http://localhost:8000`. Caso rode o backend em outro host (ex.: emulador Android acessando a máquina), ajuste com `--dart-define`:

- Emulador Android: `flutter run --dart-define=API_URL=http://10.0.2.2:8000`
- Dispositivo Android físico na mesma rede: `flutter run --dart-define=API_URL=http://SEU_IP:8000`
- iOS Simulator: `flutter run --dart-define=API_URL=http://127.0.0.1:8000`
- Web: `flutter run -d chrome --dart-define=API_URL=http://localhost:8000`

Também é possível alterar o valor padrão em `frontend/lib/services/api_service.dart`, mas o uso de `--dart-define` evita modificações de código.

## 6.3 Instalar dependências Flutter

```
cd frontend flutter pub get
```

## 6.4 Executar o aplicativo

- Android (emulador): `flutter run -d emulator-5554`
- Android (dispositivo): `flutter run -d --release`
- iOS: `flutter run -d ios`
- Web (debug): `flutter run -d chrome` ou `flutter run -d edge`

O Flutter abrirá uma URL local (por exemplo http://127.0.0.1:51119) quando em modo web; utilize-a no navegador.

## 6.5 Testes rápidos do frontend

```
flutter test
```

O projeto possui testes básicos em `frontend/test/services_test.dart` para validar chamadas de serviços.

## 7. Execução integrada e URLs esperadas

1. Suba o backend (Seção 5) e aguarde a mensagem `Application startup complete` no console.

2. Suba o frontend (Seção 6) apontando para a mesma URL.

3. Fluxo mínimo a validar:

- Na tela inicial selecione UF "SP" e cidade "Santos" e toque em "Ver risco".

- Abra o mapa aprimorado e altere os filtros de dia (+1, +2, +3) e intensidade (Baixo/Médio/Alto).

4. URLs úteis durante a validação:

- Backend API: http://localhost:8000

- Swagger: http://localhost:8000/docs

- App Web (quando rodando via `flutter run -d chrome`): URL mostrada pelo CLI

## 8. Opcional: Execução via Docker

O repositório contém `docker-compose.yaml` com dois serviços (backend e frontend). Atualize o caminho `context` do serviço `frontend` para `./frontend` (o arquivo atual ainda aponta para ./mobile). Depois rode:

```
cp backend/.env.example backend/.env # preencha BRASIL_ABERTO_API_KEY se possuir FRONTEND_API_URL=http://backend:8000 docker compose up --build
```

Após o build, o backend exporta a porta 8000 e o frontend web ficará em http://localhost:8080.

## 9. Solução de problemas rápidos

• **Porta 8000 já em uso:** altere PORT no backend/.env e rode o Flutter com `--dart-define=API_URL=http://localhost:NOVA_PORTA`.

• **Timeout ao consultar cidades:** confirme acesso a https://nominatim.openstreetmap.org e verifique se não há bloqueio de rede.

• **Erros SSL no macOS:** execute `Install Certificates.command` dentro do diretório do Python framework.

• **App Flutter não encontra assets IBGE:** garanta que o comando `flutter pub get` foi executado e que `frontend/assets/ibge` permanece dentro do projeto.

• **Erro de CORS em build web externo:** defina a variável `HOST=0.0.0.0` e utilize o mesmo IP no `API_URL`.

Este manual garante os mesmos passos usados pelo grupo para preparar e rodar o AlagAlert localmente. Em caso de dúvidas, consulte o README principal para contexto adicional.