Manual Técnico de Implementação

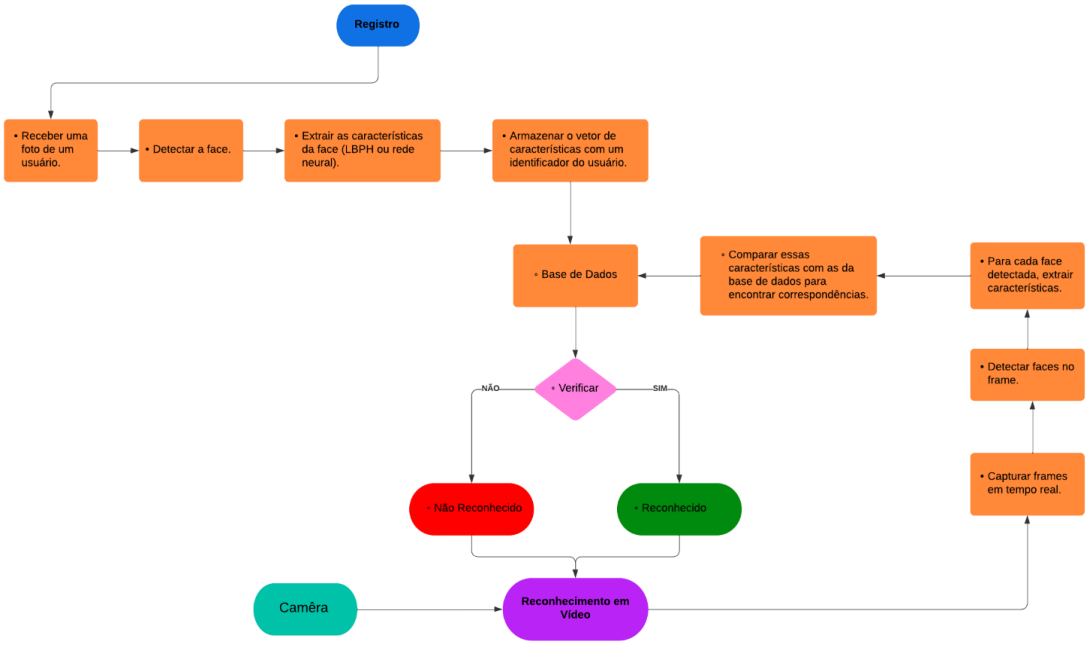
1. Requisitos

- **Sistema Operacional**: Windows, Linux ou MacOS
- **Linguagem de Programação**: Python 3.x
- **Dependências**:
    - opencv-python
    - dlib
    - numpy
    - matplotlib
    - Flask (para API, se necessário)
    - SQLite (ou outro banco de dados)
    - **Outros requisitos**: Webcam ou vídeo ao vivo

2. Arquitetura do Sistema

- **1. Captura de Imagens/Vídeos**: A câmera captura o vídeo ou a aplicação recebe um vídeo em tempo real.
- **2. Detecção de Faces**: A detecção de faces é realizada em cada frame utilizando OpenCV (Haar Cascades, HOG, etc.).
- **3. Extração de Características**: Usamos LBPH ou uma rede neural para gerar vetores de características de cada face detectada.
- **4. Comparação**: A face detectada é comparada com as características previamente armazenadas no banco de dados para encontrar correspondências.
- **5. Resultado**: A aplicação retorna o identificador da pessoa reconhecida ou uma mensagem de que não foi possível identificar.

Fluxo de trabalho:



3. Configuração do Ambiente

1. **Clone o repositório**:

```
git clone https://github.com/seu-usuario/seu-repositorio.git
cd seu-repositorio
```

2. **Instale as dependências**:

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

3. **Configuração do Banco de Dados**: Se estiver usando SQLite, crie o banco de dados para armazenar as características faciais:

```
sqlite3 facial_recognition.db
```

4. Uso

Passo 1: Registro de Usuário

- Para registrar um novo usuário, utilize a API ou o sistema de registro da aplicação. Um exemplo de código:

```
import cv2
from recognition_module import register_face
```

```
# Carregar a imagem do usuário
image = cv2.imread('foto_usuario.jpg')

# Chamar função para registro
register_face('nome_usuario', image)
```

Passo 2: Reconhecimento em Vídeo

- A aplicação deve capturar o vídeo ao vivo e reconhecer as faces registradas.

```
import cv2
from recognition_module import recognize_faces_in_video

# Iniciar captura de vídeo
video_capture = cv2.VideoCapture(0)

# Função para reconhecimento
recognize_faces_in_video(video_capture)
```

Passo 3: Atualização da Base de Dados

- Novos usuários podem ser adicionados sem precisar parar a aplicação. Eles serão reconhecidos automaticamente nas próximas tentativas.

5. API (se houver)

Endpoint 1: Registro de Usuário

- **Método**: POST
- **URL**: /register
- **Parâmetros**:
  - image: Imagem da pessoa
  - name: Nome do usuário
- **Resposta**:
  - Sucesso ou erro ao registrar.

```
curl -X POST -F 'image=@path/to/image.jpg' -F 'name=Lucas'
http://localhost:5000/register
```

Endpoint 2: Reconhecimento Facial em Tempo Real

- **Método**: GET
- **URL**: /recognize
- **Parâmetros**: Nenhum (vídeo capturado diretamente)
- **Resposta**: ID da pessoa reconhecida ou 'Não reconhecido'.

8. Estrutura de Arquivos

- /src/: Código-fonte do projeto
  - main.py: Arquivo principal para iniciar a aplicação.
  - recognition_module.py: Módulo com as funções de reconhecimento.
- /models/: Modelos treinados ou configurados (LBPH, FaceNet).
- /data/: Imagens ou dados de faces para registro.
- requirements.txt: Lista de dependências do projeto.
- README.md: Documentação do projeto.

9. Considerações Finais

- **Performance**: O desempenho pode variar dependendo da quantidade de faces no banco de dados e da resolução das imagens.
- **Privacidade**: Assegure-se de seguir as regulamentações de privacidade de dados, como a GDPR, ao armazenar e processar dados faciais.
- **Melhorias Futuras**: Utilizar redes neurais mais avançadas como FaceNet para melhorar a precisão do reconhecimento.