Documento de Levantamento de Tecnologias

1. Introdução

1.1 Objetivo

Este documento tem como objetivo descrever as tecnologias selecionadas para o desenvolvimento do projeto de **reconhecimento facial em tempo real**, destacando as linguagens de programação, frameworks, bibliotecas e ferramentas que serão utilizadas em cada etapa do processo.

1.2 Escopo

O levantamento abrange todas as tecnologias necessárias para o desenvolvimento do sistema, desde a captura de vídeos, processamento de imagens, até a infraestrutura necessária para suportar o reconhecimento facial e APIs associadas.

2. Requisitos Técnicos

- **Sistema Operacional**: Windows, Linux ou MacOS
- **Linguagem de Programação**: Python 3.x
- **Hardware**: GPU (se necessário para aceleração de deep learning)
- **Dependências**:
    - OpenCV 4.10.0 (para processamento de imagens)
    - dlib (detecção facial)
    - numpy (manipulação de dados)
    - Flask (para API)
    - SQLite (armazenamento local de dados)
    - TensorFlow ou PyTorch (para deep learning)

3. Tecnologias Selecionadas

3.1 Linguagens de Programação

Python 3.x

Python foi escolhido por sua popularidade e grande quantidade de bibliotecas que facilitam o desenvolvimento de sistemas de visão computacional, além de sua simplicidade e comunidade ativa.

3.2 Frameworks e Bibliotecas de Visão Computacional

- **OpenCV**: Usado para captura de vídeo, processamento de imagens e detecção de faces em tempo real.
- **dlib**: Ferramenta auxiliar na detecção facial usando métodos como HOG e landmarks faciais.
- **MTCNN** (opcional): Para detecção de faces com redes neurais, caso seja necessária uma abordagem mais robusta.

3.3 Frameworks de Aprendizado de Máquina e Deep Learning

- **TensorFlow** ou **PyTorch**: Para o treinamento e utilização de redes neurais avançadas, como FaceNet, que oferece precisão superior para a geração de embeddings faciais.
- **Scikit-learn**: Para a avaliação de classificadores e outras tarefas de aprendizado de máquina.

3.4 Armazenamento de Dados

- **SQLite**: Escolhido por sua simplicidade e facilidade de integração com Python, sendo ideal para projetos menores e protótipos que não necessitam de alta escalabilidade.
- **PostgreSQL** (opcional): Se o projeto crescer, um banco de dados relacional mais robusto pode ser necessário para armazenar grandes volumes de dados.

3.5 Infraestrutura e Performance

- **CUDA** e **cuDNN**: Para aceleração por GPU em modelos de deep learning, caso as redes neurais sejam utilizadas para o reconhecimento facial.

3.6 Desenvolvimento de APIs

- **Flask**: Microframework em Python para criação de APIs RESTful que permitirão a integração do sistema de reconhecimento facial com outros sistemas e dispositivos.
- **FastAPI** (opcional): Alternativa ao Flask, com foco em desempenho e facilidade de criação de APIs rápidas.

3.7 Ferramentas de Controle de Versão e Colaboração

- **Git**: Controle de versão utilizado para garantir a rastreabilidade e a colaboração no desenvolvimento do projeto.
- **GitHub/GitLab**: Plataforma para armazenamento do código-fonte e colaboração entre desenvolvedores.

3.8 Ferramentas de Teste e Integração Contínua

- **PyTest**: Para testes unitários e garantir a confiabilidade do código.
- **Jenkins**: Utilizado para automação de testes e integração contínua.

4. Justificativa das Escolhas

Cada tecnologia foi selecionada considerando fatores como:

- **Facilidade de uso**: Python e suas bibliotecas facilitam o desenvolvimento rápido e eficiente.
- **Desempenho**: A aceleração via GPU (CUDA/cuDNN) foi considerada para lidar com o processamento de deep learning em tempo real.
- **Escalabilidade**: Ferramentas como PostgreSQL foram incluídas como opção para projetos em crescimento.
- **Composição do time**: A familiaridade do time com Python e suas bibliotecas foi determinante para a escolha da linguagem e dos frameworks.

5. Possíveis Riscos e Mitigações

5.1 Desempenho

- **Risco**: O desempenho pode ser insuficiente ao utilizar apenas CPU para processamento de deep learning em tempo real.
- **Mitigação**: Implementação de aceleração via GPU (NVIDIA CUDA).

5.2 Compatibilidade

- **Risco**: Algumas bibliotecas podem ter problemas de compatibilidade entre diferentes sistemas operacionais.
- **Mitigação**: Garantir a utilização de bibliotecas multiplataforma e testar o sistema em diferentes ambientes (Windows, Linux, MacOS).

5.3 Privacidade e Segurança

- **Risco**: O projeto lida com dados biométricos (faces), o que pode levantar questões de privacidade.
- **Mitigação**: Implementação de criptografia para armazenamento de dados sensíveis e conformidade com regulamentos como o GDPR.

6. Conclusão

Este levantamento de tecnologias fornece uma visão detalhada das ferramentas e bibliotecas que serão utilizadas no desenvolvimento do sistema de reconhecimento facial em tempo real. As tecnologias foram escolhidas com base no desempenho, facilidade de uso, escalabilidade e compatibilidade com os objetivos do projeto.