UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE TECNOLOGIA E GEOCIÊNCIAS

DEPARTAMENTO DE ELETRÔNICA E SISTEMAS

Welington Gabriel Oliveira de Carvalho

Vitor Maciel da Silva

CONTROLE DE ACESSO ATRAVÉS DE RECONHECIMENTO FACIAL POR IA

RELATÓRIO DE PROJETO

2025

Recife

Sumário

1. Resumo

O presente relatório descreve as atividades desenvolvidas para implementação de um sistema de controle de acesso, baseado na estrutura física do Departamento de Engenharia Elétrica (DEE) da UFPE, integrando conhecimentos das disciplinas de Inteligência Artificial (IA) com Banco de Dados (BD). O sistema realiza reconhecimento facial a partir de imagens capturadas por câmera e autoriza ou nega acessos com base em um cadastro prévio de usuários treinados no modelo. A persistência dos dados é realizada via banco MySQL, garantindo registro simples e eficiente dos logs de acesso em tempo real, com baixo custo. A aplicação foi desenvolvida em Python, baseado em OpenCV, com interface de consulta implementada em Streamlit. O protótipo do sistema foi executado localmente em um notebook com webcam, com banco de dados MySQL via cloud Free Database SQL

2. Introdução

A segurança e o controle de acesso a espaços acadêmicos são preocupações constantes. A ausência de mecanismos de controle de acesso eficiente, eleva o risco à integridade da comunidade acadêmica e dos recursos públicos, contando com altos custos e burocracia de aquisição de equipamentos básicos e especializados aplicados ao superior. O projeto proposto visa modernizar o controle de acesso ao DEE, substituindo métodos de RFID por reconhecimento facial automatizado. Além de registrar entradas e saídas, o sistema permitirá o monitoramento em tempo real das pessoas presentes no local, com uma interface aberta através do email institucional, útil em situações de emergência ou comodidade de visualização por alunos e professores.

3. Objetivo Geral

Demonstrar domínio e conhecimento dos assuntos vistos em sala de aula, nas disciplinas de Inteligência Artificial e Banco de Dados, através de um projeto com aplicação prática.

3.1. Objetivos Específicos

- Aplicar técnicas de inteligência artificial para identificação de pessoas para controle de acesso ao ambiente, utilizando aprendizado de máquina treinado previamente;
- Projetar e estruturar um banco de dados (MySQL) para armazenamento de dados dos usuários, com registros de acesso, com controle de validade e rastreabilidade;

- Gerar estatísticas e relatórios com informações úteis à comunidade acadêmica e gestores dos departamentos, com monitoramento em tempo real através de uma interface visual;
- Assegurar a modularidade do sistema, permitindo escalabilidade da aplicação;
- Preservar a integridade acadêmica, registrando acessos negados, incluindo captura de imagem e timestamp da tentativa de acesso;

4. Tecnologias Utilizadas

4.1. Backend e Reconhecimento Facial

- **Python 3:** Linguagem principal do projeto, escolhida pela simplicidade e disponibilidade de materiais voltados à integração com inteligência artificial, banco de dados e interface gráfica.
- **OpenCV:** Biblioteca de código aberto voltada à captura de imagem e detecção de imagem em tempo real
- **Haar Cascade Classifier:** Algoritmo de detecção de objetos por aprendizagem de máquinas, aplicado para separação de imagens facial e do ambiente.
- **Git e GitHub:** Plataformas de controle de versão, armazenamento e compartilhamento de código.

4.2. Frontend e Interface Gráfica

- **CustomTkinter:** Biblioteca GUI (Graphical User Interface) para Python, utilizada para criação das interfaces gráficas do sistema de reconhecimento facial.
- **Streamlit:** Framework voltado para construção de aplicações web e dashboards interativos, permitindo a visualização em tempo real das informações armazenadas no banco de dados.

4.3. Banco de dados e Cloud

- **MySQL 8.0:** Sistema de gerenciamento de banco de dados relacional, utilizado para armazenamento de dados de usuários, logs de acesso, imagens codificadas em base64 e metadados. A estrutura do banco permite manter a aplicação funcionando com baixo custo, com pequenas aplicações descentralizadas, assim como a estrutura de governança da UFPE, dividida em departamentos.
- **MySQL Connector:** Biblioteca para conexão direta entre aplicações Python e servidores MySQL, utilizada para inserção, atualização e consulta dos dados no sistema.
- **Freesqldatabase:** Servidor MySQL gratuito para testes em micro escala, responsável por armazenar todos os dados gerados na concepção do projeto.

4.4. Hardware

- **Arduino Uno:** Microcontrolador para controle físico de travas e sensores de movimento.

5. Arquitetura Geral do Sistema e Repositório no GitHub

Todos os dados, incluindo código do projeto, se encontram no repositório do GitHub: https://github.com/Gabriel-o-Carvalho/facedetector. Abaixo é apresentado a estrutura principal de funcionamento do sistema, seguidos as estruturas em árvores das aplicações desenvolvidas em Python para o modelo de reconhecimento facial, incluindo dashboard e arquitetura em árvore do banco de dados.

Figura 1 e 2: Fluxograma de funcionamento do sistema feito no Microsoft Visio e arquitetura em árvore do projeto

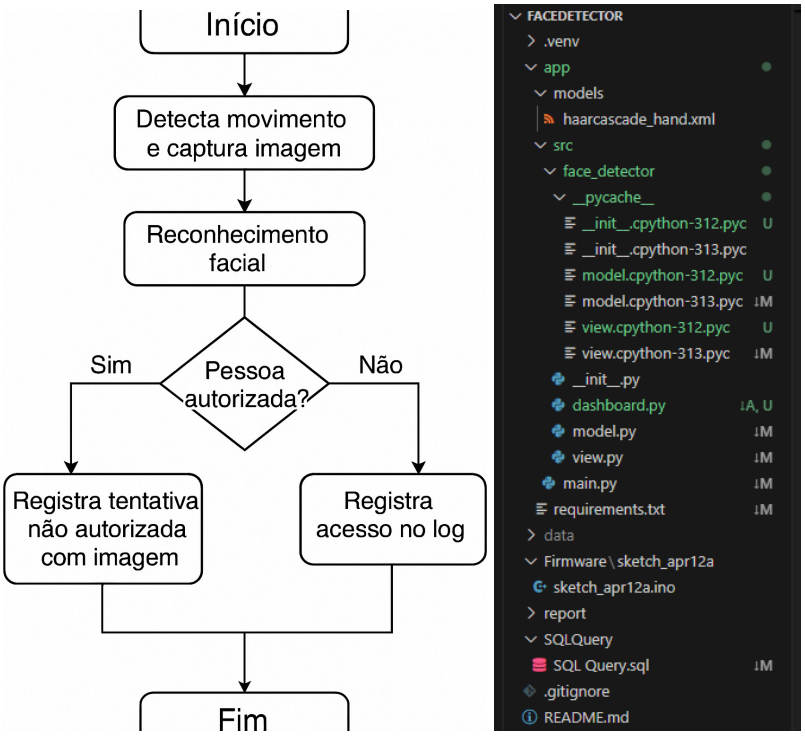


Figura 3: Estrutura do banco de dados

```
Banco de Dados MySQL (controle_acesso_dee)
    Tabela: usuarios
        id - INT (Primary Key)
        nome - VARCHAR (60)
        CPF - VARCHAR (11)
        tipo - Enum
        cadastro_valido_ate DATETIME
        criado_em DATETIME

    Tabela: log_acesso
        id - INT (Primary Key)
        usuario_id (Foreign Key → usuarios.id)
        autorizado - BOOLEAN
        timestamp - TIMESTAMP
        imagem_base64 - LONGTEXT
```

6. Metodologia

O processo de desenvolvimento do projeto foi dividido em etapas operacionais, planejadas de acordo com a lógica de dependência entre os componentes do sistema. Abaixo é apresentado o fluxo de construção do projeto, as tecnologias empregadas e as decisões de projeto estabelecidas.

6.1. Concepção de Projeto

A escolha de desenvolver um sistema de controle de acesso por reconhecimento facial para o Departamento de Engenharia Elétrica (DEE) da UFPE surgiu da união entre viabilidade técnica da aplicação, escopos das disciplinas cursadas e conhecimento dos discentes envolvidos no projeto. O DEE foi um pensando como contexto de estudo devido a existência de um sistema similar de controle de acesso por crachás com RFID. Além disso, caso o projeto trouxesse resultados satisfatórios, seria possível ter o apoio dos professores para testar e validar um possível acesso em um ambiente controlado dentro do departamento.

Do prisma de viabilidade financeira, as aplicações foram selecionadas visando manter o menor custo possível para testes em pequenas escalas. A utilização da linguagem Python, com a biblioteca OpenCV e armazenamento gratuito com MySQL e FreeSQLDatabase, tornaram possível a construção do projeto sem a necessidade de investimento financeiro pelos alunos, mas com uma estrutura modular que permite a troca de cada item do projeto em uma possível aplicação futura

Por fim, é notório mencionar a familiaridade prévia da equipe com as tecnologias empregadas, acelerando o desenvolvimento e integração do projeto, aproveitando-se da grande gama de documentações na internet em projetos semelhantes.

6.2. Interface de Captura de Imagem

A segunda etapa consistiu na construção de uma interface gráfica destinada à coleta de imagens dos usuários. A interface foi desenvolvida em Python, utilizando as bibliotecas CustomTkinter e OpenCV para coleta de imagens dos usuários.

Figura 4: Parte inicial do código da interface

```
def _create_btn_frame(self):
    self.btn_frame = customtkinter.CTkFrame(self, fg_color="transparent")

    self.btn_frame.columnconfigure(0, weight=1)
    self.btn_frame.columnconfigure(1, weight=1)
    self.btn_frame.columnconfigure(2, weight=1)
    self.btn_frame.columnconfigure(3, weight=1)


    def toggle_face_detection():
        self.face_detection = not self.face_detection
    btn1 = customtkinter.CTkButton(self.btn_frame, text="Iniciar", command=toggle_face_detection)
    btn1.grid(row=0, column=0, sticky="nsew", padx=10)

    btn2 = customtkinter.CTkButton(self.btn_frame, text="Cadastrar", command=self.controller.handle_sign_up)
    btn2.grid(row=0, column=1, sticky="nsew", padx=10)

    btn3 = customtkinter.CTkButton(self.btn_frame, text="Remover")
    btn3.grid(row=0, column=2, sticky="nsew", padx=10)

    btn4 = customtkinter.CTkButton(self.btn_frame, text="Acessos")
    btn4.grid(row=0, column=3, sticky="nsew", padx=10)

    self.btn_frame.grid(row=1, column=0, sticky="ew", pady=10)


def _create_img_frame(self):
    self.img_frame = customtkinter.CTkFrame(self, fg_color="black")

    self.cam_src = 0
    self.cap = cv2.VideoCapture(self.cam_src)

    self.label = customtkinter.CTkLabel(master=self.img_frame, text="")
    self.label.pack(expand=True, fill="both")

    self.img_frame.grid(row=0, column=0, sticky="nsew")
```

Com a construção da interface inicial para coleta de imagens, o próximo passo é permitir que o sistema consiga identificar rostos nas imagens.

6.3. Identificação Facial com Haar Cascade

Após o desenvolvimento da interface de captura, foi implementado no sistema a separação da face frontal do restante da imagem, necessário para melhorar a qualidade do treinamento e garantir a precisão necessária do sistema de reconhecimento facial. Com isso, foi implementado o classificador Haar Cascade para localizar regiões faciais e extrair apenas essa porção da imagem a ser utilizada no treinamento do algoritmo.

Figura 5: Código para detecção de rosto

```
def _create_img_frame(self):
    self.img_frame = customtkinter.CTkFrame(self, fg_color="black")

    self.cam_src = 0
    self.cap = cv2.VideoCapture(self.cam_src)

    self.label = customtkinter.CTkLabel(master=self.img_frame, text="")
    self.label.pack(expand=True, fill="both")

    self.img_frame.grid(row=0, column=0, sticky="nsew")
```

6.4. Reconhecimento Facial e Treinamento

Com as imagens faciais separadas anteriormente, foi utilizado o método LBPH (Local Binary Pattern Histogram) para o treinamento do modelo de reconhecimento facial. O sistema pré-treinado, foi alimentado com os rostos de dois alunos, para testar se o algoritmo iria associar os rostos corretamente. Apesar de algumas flutuações no reconhecimento facial, principalmente com algumas variações na iluminação ou expressões faciais, o sistema teve um bom desempenho na classificação dos usuários.

Figura 6: Trecho de código do modelo de reconhecimento facial

```python
class Model:
    def __init__(self):
        self.db = FaceDetectionDB()
        self.locker = Locker("/dev/ttyUSB0", 9600)
        self.face_model = FaceDetectorModel()
        self.curr_idx = 0

    def detect_face(self, frame):
        return self.face_model.face_identifier(frame)

    def add_user_to_db(self, user_data):
        return 0

    def update_dataset(self, user_data, frame, faces):
        if not os.path.isdir(f"data/user.{user_data['id']}"):
            os.makedirs(f"data/user.{user_data['id']}")
            self.curr_idx = 0
        if self.curr_idx < 100:
            for (x,y,w,h) in faces:
                cv2.imwrite(f"data/user.{user_data['id']}/user.{user_data['id']}.{self.curr_idx}.png", frame[y:y+h, x:x+w])
                self.curr_idx += 1
            return False
        else:
            return True
```

6.5. Criação do Banco de Dados MySQL

Tendo a implementação funcional do modelo de reconhecimento facial, a próxima etapa foi a construção das tabelas do banco de dados para armazenar as informações dos usuários, registros de acessos (em formato de logs) e tentativas não autorizadas de acesso, isto é, usuários não reconhecidos capturados pela câmera.

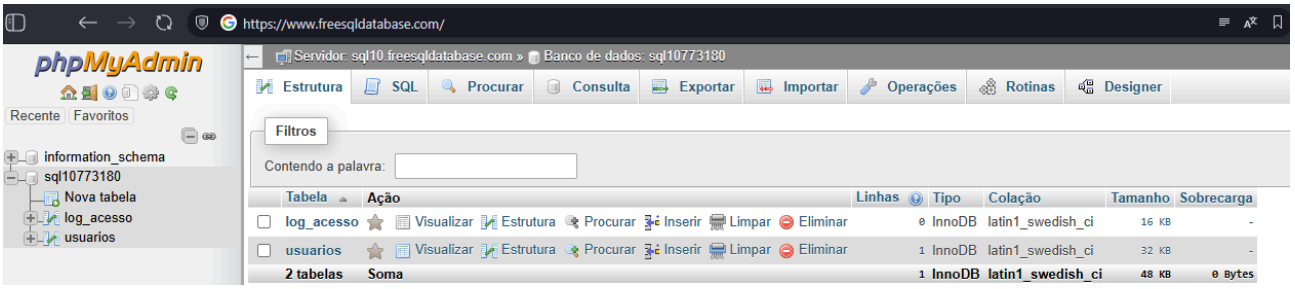
Figura 6: Trecho de código de interação com o banco de dados

```python
class FaceDetectionDB:
    def __init__(self):
        self.conexao = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            user="root",
            password="root",
            database="controle_acesso_dee"
        )
        self.cursor = self.conexao.cursor()

    def add_user(self, user):
                # O campo 'cadastro_valido_ate' deve estar no formato 'YYYY-MM-DD'
        # Os tipos de cadastro são: Aluno, Professor, Técnico, Terceirizado ou Visitante
        query = """
        INSERT INTO usuarios (nome, CPF, tipo, cadastro_valido_ate)
        VALUES (%s, %s, %s, %s)
        """
        values = (user.name, user.cpf, user.tipo, user.cadastro_valido_ate)
        self.cursor.execute(query, values)
        self.conexao.commit()
        return self.cursor.lastrowid  # retorna o id do novo usuário
```

6.6. Hospedagem em Servidor

Visando a centralização dos dados gerados e aplicações de acesso público, considerando também a disparidade de utilização de sistemas operacionais para desktop entre os membros (Windows OS e Linux OS), optou-se por um servidor web capaz de armazenar os dados via MySQL Connection. Apesar do Free Database SQL oferecer apenas 5MB em nuvem, mesmo com um sistema simples, havia a limitação do armazenamento de imagens em base64. Com isso, não foi possível realizar o guardar as imagens de tentativas indeferidas de acesso pelo sistema.

Figura 7: Interface do Free Database SQL



6.7. Dashboard com Streamlit

Realizada a implementação em nuvem do banco de dados, resta a consulta aos registros em tempo real. Visando utilizar uma biblioteca modular com framework, foi desenvolvido um painel web interativo com o Streamlit, para exibição dos últimos registros de acesso, usuários presentes e estatística de acesso ao departamento.

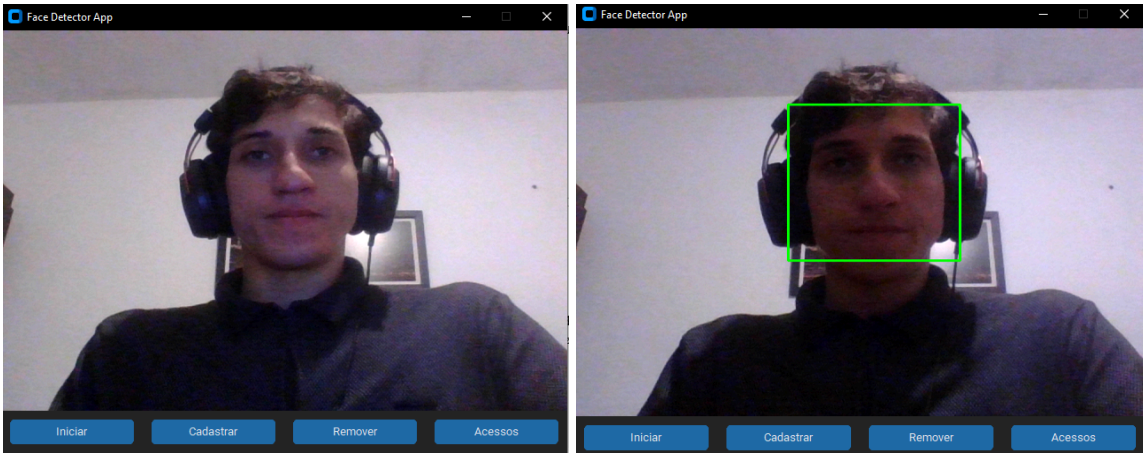
Figura 8: Trecho de código com consulta ao banco e interface com Streamlit

```python
# Comportamento por tipo
def acessos_por_tipo():
    conn = conectar_banco()
    query = """
        SELECT u.tipo, COUNT(*) as acessos
        FROM log_acesso l
        JOIN usuarios u ON l.usuario_id = u.id
        WHERE l.autorizado = TRUE
        GROUP BY u.tipo
    """
    df = pd.read_sql(query, conn)
    conn.close()
    return df

# Interface
st.set_page_config(page_title="Painel de Acesso - DEE", layout="wide")
st.title("🔐 Painel de Acesso - DEE/UFPE")
st.markdown("Este painel mostra os acessos mais recentes e as pessoas atualmente no local.")

aba = st.sidebar.radio("Navegação", ["Dashboard Principal", "Frequência Mensal", "Horários de Pico", "Cadastros a Vencer", "Acessos por Tipo"])

if aba == "Dashboard Principal":
    st.subheader("📋 Últimos registros de acesso")
    logs_df = buscar_logs()
    if not logs_df.empty:
        logs_df['timestamp'] = logs_df['timestamp'].apply(lambda x: x.strftime('%d/%m/%Y %H:%M:%S'))
        logs_df['autorizado'] = logs_df['autorizado'].map({1: '✅ Autorizado', 0: '❌ Negado'})
        st.dataframe(logs_df.rename(columns={
            'nome': 'Nome', 'tipo': 'Tipo', 'autorizado': 'Status', 'timestamp': 'Data/Hora'
        }), use_container_width=True)
    else:
        st.info("Nenhum acesso registrado ainda.")
```

7. Resultados

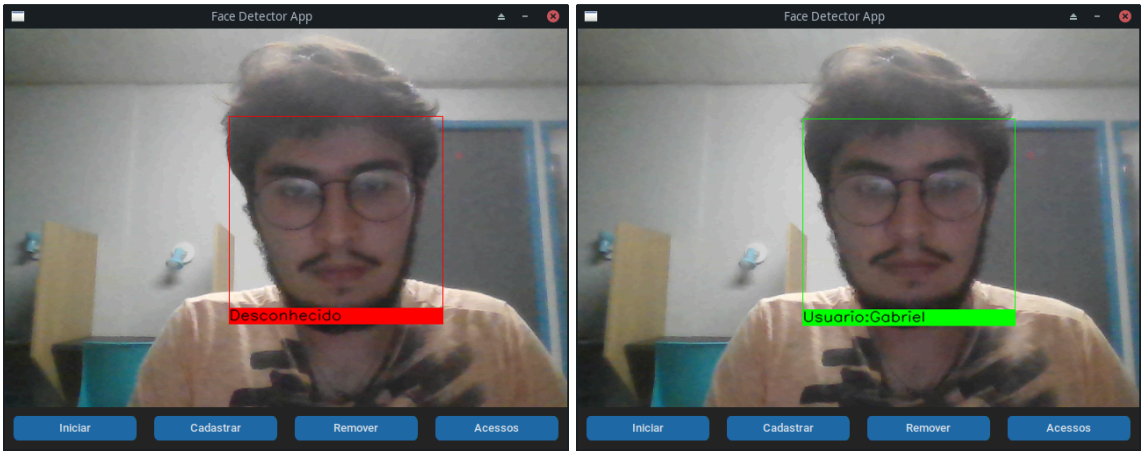
A implementação do sistema apresentado trouxe resultados animadores de validação do sistema como controle de acesso. A interface simplificada e intuitiva permite a navegação dos usuários sem a necessidade de treinamento prévio. Um registro do sistema funcionando antes da implementação dos algoritmos está ilustrada na figura 9. Após implementação da detecção de rosto com Haar Cascade, chegamos ao resultado apresentado na figura 10.

Figuras 9 e 10: Interface de captura de imagem com e sem detecção de rosto



Ao término da implementação do reconhecimento facial, a aplicação era capaz de identificar rostos desconhecidos em seu modelo interno de treinamento, apresentando a tag “Desconhecido” (figura 11) para rostos não reconhecidos. Realizado o cadastro, o algoritmo passa a informar o nome do usuário na tela (figura 12).

Figuras 11 e 12: Interface de captura de imagem antes e depois do treinamento do cadastro do usuário



A função também realizava o registro dos usuários no banco de dados, conforme apresentado na figura 14.

Figura 13: Cadastros presentes o banco de dados

Mostrando registros 0 - 2 (3 no total, Consulta levou 0.1752 segundos.)

```
SELECT * FROM `usuarios`
```

Carregando...

Mostrar tudo | Número de linhas: 25 Filtrar linhas: Procurar nesta tabela Ordenar pela chave: Nenhum

+ Opções

	id	nome	CPF	tipo	cadastro_valido_ate	criado_em
Editar Copiar Remover	0	Gabriel	12345678901	Aluno	2025-12-12	2025-04-14 15:57:35
Editar Copiar Remover	1	Vitor	2345678901	Aluno	2025-12-12	2025-04-14 15:59:11
Editar Copiar Remover	2	Neemias	12312312300	Aluno	2025-12-12	2025-04-14 14:05:54

Marcar todos *Com marcados:* Editar Copiar Remover Exportar

A fim de avaliar de maneira mais sistemática a precisão do modelo, foram criados o relatório de classificação e a matriz de confusão, a partir das classes *classification_report* e *confusion_matrix* do no módulo OpenCV. Os resultados pode ser observados na figuras 13 e 14 respectivamente. É possível notar que o modelo apresentou uma alta acurácia nas métricas apresentadas, contudo deve-se ressaltar que o dataset disponibilizado possuia apenas 400 imagens, sendo 100 do usuário 0 e 300 do usuário 12. Provavelmente houve a ocorrência de um *overfitting* de dados. Os *datasets* de treino e teste foram divididos igualmente para reduzir esse efeito.

Figura 14: Matriz confusão do teste realizado

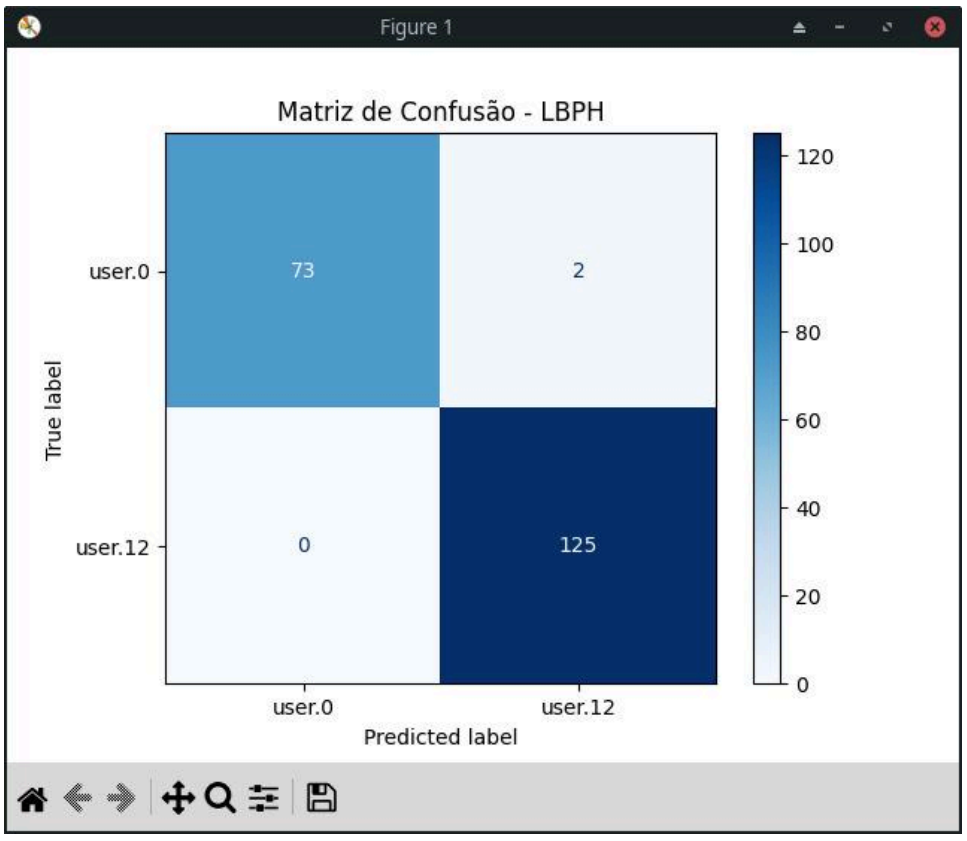
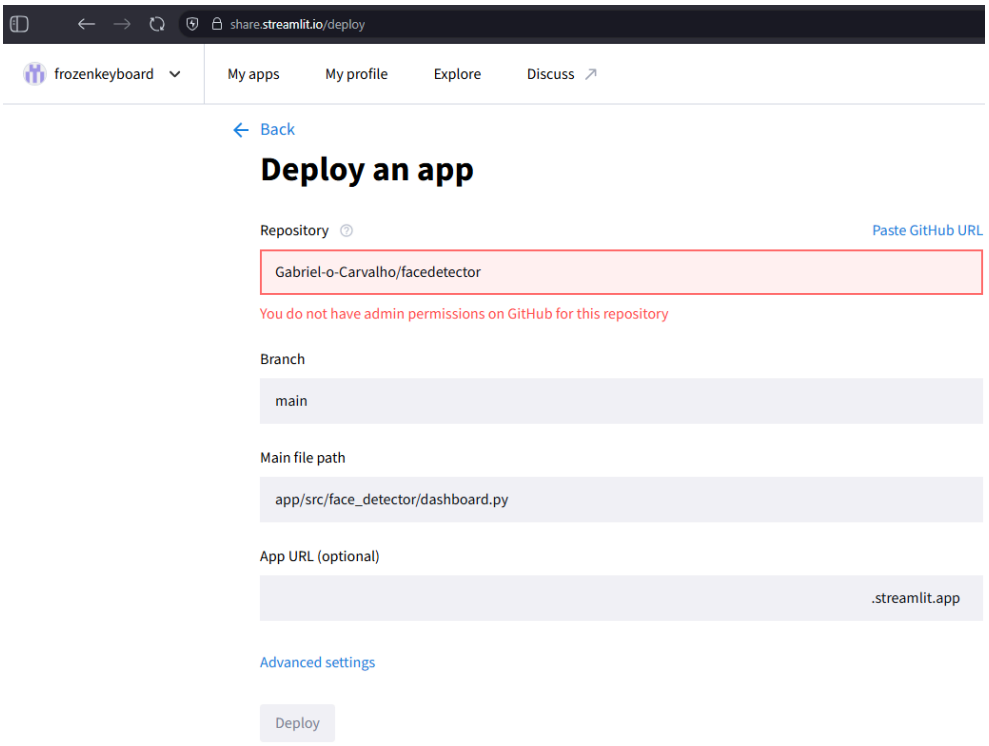


Figura 15: Resultados de acurácia obtidos

```
[Relatório de Classificação]
              precision    recall  f1-score   support

     user.0       1.00      0.97      0.99        75
    user.12       0.98      1.00      0.99       125

    accuracy                           0.99       200
   macro avg       0.99      0.99      0.99       200
weighted avg       0.99      0.99      0.99       200
```

Por erros de limitação de permissão de acesso do streamlit, aliado ao curto prazo para entrega da atividade, não foi possível solucionar o problema de permissão de acesso. Entretanto, o código funcional, incluindo consulta ao banco de dados e construção da interface se encontram disponíveis no GitHub para verificação. Apesar dos erros apresentados, todos os usuários estavam com todas as permissões necessárias.

Figura 16: Tentar de deploy da dashboard do streamlit



8. Conclusão

Em suma, o projeto alcançou seus objetivos ao propor e desenvolver uma solução prática, envolvendo os conceitos vistos em sala de aula, permitindo integrar conhecimentos das áreas de Inteligência Artificial e Banco de Dados para solucionar um problema do controle de acesso por reconhecimento facial. Baseando-se em tecnologias amplamente empregadas, como Python, OpenCV e MySQL, foi possível construir uma aplicação funcional, com potencial de uso prático e adaptável a diferentes cenários.

A escolha pelo reconhecimento facial como método de autenticação se mostrou eficiente, apesar de serem necessários ajustes no ambiente para lidar com a mudança de luminosidade, eliminando a necessidade de cartões físicos, muitas vezes esquecidos pelos alunos. A coleta e o treinamento de imagens, somados à detecção em tempo real, permitiram montar um sistema ágil com boa margem de precisão. Além disso, o uso de uma interface gráfica para o cadastro de usuários e de um dashboard online para monitoramento reforça a usabilidade e a aplicabilidade do projeto.

Acerca do banco de dados, o sistema foi estruturado de forma clara simplificada, podendo adicionar mais tabelas conforme necessidades futuras. A utilização de um servidor online gratuito, viabilizou testes em nuvem e validações remotas da aplicação, evidenciando a escalabilidade do sistema.

9. Referências Bibliográficas

Analytics Website Dashboard using Python and Streamlit Library with mysql database (Data Science). *YouTube*. Disponível em: https://www.youtube.com/watch?v=pWxDxhWXJos. Acesso em: 13 abr. 2025.

Connect to MySQL. Streamlit. Disponível em: https://docs.streamlit.io/develop/tutorials/databases/mysql. Acesso em: 13 abr. 2025.

Controlando um servo motor com Arduino. MakerHero. Disponível em: https://www.makerhero.com/blog/controlando-um-servo-motor-com-arduino/. Acesso em: 10 abr. 2025.

CustomTkinter., c2024. Disponível em: https://github.com/TomSchimansky/CustomTkinter. Acesso em: 05 abr. 2025.

MySQL Connector/Python Developer Guide.: MySQL, 2024. Disponível em: https://dev.mysql.com/doc/connector-python/en/. Acesso em: 09 abr. 2025.

Real time face detection in OpenCV with python p. 1. . *YouTube*. Disponível em: https://www.youtube.com/watch?v=mkAx_81Pcww. Acesso em: 07 abr. 2025.