UNIVERSIDADE PAULISTA

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

GABRIELLE MARTINS DIAS RA:G8447E6

GRETZEL KATTIA LAURA PENALOZA RA:N158930

MATHEUS MARCELINO DE OLIVEIRA RA:G79GHB8

SOLANGE ARAUJO DE SANTANA RA:N069228

DESENVOLVIMENTO DE UM SISTEMA DE IDENTIFICAÇÃO E AUTENTICAÇÃO BIOMÉTRICA

SÃO PAULO

2025

GABRIELLE MARTINS DIAS RA:G8447E6

GRETZEL KATTIA LAURA PENALOZA RA:N158930

MATHEUS MARCELINO DE OLIVEIRA RA:G79GHB8

SOLANGE ARAUJO DE SANTANA RA:N069228

DESENVOLVIMENTO DE UM SISTEMA DE IDENTIFICAÇÃO E AUTENTICAÇÃO BIOMÉTRICA

Trabalho apresentado a UNIP – Universidade Paulista como requisito para conclusão da disciplina Atividades Práticas Supervisionadas.

Orientador: Prof. Rafael Santo

SÃO PAULO

2025

SUMÁRIO

OBJETIVO DO TRABALHO

O presente trabalho tem como objetivo principal o desenvolvimento de um sistema de reconhecimento facial voltado ao controle de acesso, utilizando técnicas modernas de visão computacional e inteligência artificial. Busca-se proporcionar uma solução automatizada, segura e eficiente, capaz de substituir ou complementar métodos tradicionais de autenticação, como o uso de senhas ou cartões de identificação, reduzindo riscos de fraudes e aumentando a confiabilidade dos processos de verificação de identidade.

O sistema foi projetado para realizar a captura, processamento e comparação de imagens faciais em tempo real, por meio de algoritmos de aprendizado profundo integrados à biblioteca *face_recognition*. Além disso, propõe-se uma arquitetura modular e escalável, construída com o framework Django, que permite fácil manutenção, reuso de código e futuras expansões, como o acréscimo de novas funcionalidades de segurança ou relatórios de acesso.

De forma complementar, o projeto busca explorar a integração entre ciência da computação e inteligência artificial aplicada, demonstrando o potencial das tecnologias emergentes na automatização de processos institucionais. Pretende-se, portanto, oferecer uma ferramenta prática e inovadora, que alie desempenho, precisão e usabilidade, contribuindo para o avanço de soluções tecnológicas voltadas à autenticação biométrica e controle de informações sensíveis.

1. INTRODUÇÃO

Em uma era dominada pela informação e marcada pela constante transformação tecnológica, a segurança digital tornou-se um dos pilares fundamentais para o funcionamento de instituições públicas e privadas. O avanço da conectividade global e o aumento exponencial do volume de dados armazenados em sistemas computacionais tornaram a proteção de informações sensíveis um desafio estratégico e indispensável. Nesse contexto, os métodos de autenticação tradicionais — como senhas, cartões magnéticos e tokens — vêm se mostrando insuficientes diante da sofisticação das ameaças cibernéticas e da facilidade com que podem ser fraudados ou compartilhados indevidamente.

Com o crescimento da digitalização de processos administrativos, financeiros e institucionais, a demanda por soluções tecnológicas mais eficazes, seguras e automatizadas impulsionou o surgimento e aprimoramento de técnicas de autenticação biométrica, que têm como princípio a identificação de indivíduos a partir de suas características físicas ou comportamentais. Entre as diferentes modalidades de biometria existentes, como impressões digitais, reconhecimento de voz e leitura da íris, destaca-se o reconhecimento facial, por sua praticidade, confiabilidade e ampla aplicabilidade em sistemas computacionais modernos.

A relevância desse tema é evidenciada pela sua crescente presença em contextos de segurança crítica, tais como aeroportos, instituições financeiras, ambientes corporativos e órgãos governamentais. Nessas situações, a integridade das informações e o controle rigoroso de acesso são fatores decisivos para a proteção de dados sigilosos e para a prevenção de incidentes de segurança. Pode-se ilustrar essa importância com o exemplo de um cofre de segurança máxima instalado em um ministério, contendo documentos estratégicos relacionados a políticas ambientais e acordos internacionais. A exposição indevida desses dados poderia gerar consequências de escala global, comprometendo não apenas a segurança nacional, mas também o equilíbrio ecológico e econômico de diversas nações.

Diante desse cenário, torna-se indispensável a adoção de mecanismos de defesa que transcendam as limitações dos métodos convencionais. A biometria facial apresenta-se, portanto, como uma tecnologia de ponta capaz de associar a identidade digital à

identidade física do indivíduo, reduzindo significativamente o risco de fraudes e de acessos indevidos. Diferentemente das senhas ou cartões, que podem ser esquecidos, perdidos ou clonados, as feições humanas são únicas e praticamente imutáveis, conferindo à autenticação biométrica um grau superior de confiabilidade e eficiência.

O reconhecimento facial baseia-se na análise de pontos específicos da face humana, como a distância entre os olhos, o formato do queixo e a curvatura do nariz, que são traduzidos em representações matemáticas chamadas de encodings faciais. Esses vetores numéricos permitem que sistemas computacionais identifiquem ou confirmem a identidade de uma pessoa com elevada precisão, mesmo sob variações de iluminação, ângulo ou expressão. O avanço de técnicas de aprendizado profundo (deep learning) e de redes neurais convolucionais (CNNs) tem potencializado de forma expressiva a precisão desses modelos, possibilitando aplicações em tempo real com alto desempenho e confiabilidade.

A utilização de bibliotecas de código aberto, como *OpenCV* e *face_recognition*, associadas ao framework Django, representa uma oportunidade de aplicar conceitos teóricos de inteligência artificial e engenharia de software em uma solução prática, acessível e reprodutível. O emprego dessas ferramentas permite o desenvolvimento de um sistema modular, escalável e compatível com múltiplas plataformas, atendendo às demandas atuais por eficiência, usabilidade e segurança.

Além do aspecto tecnológico, este projeto também se insere em um contexto social e institucional relevante. Em um cenário de crescente preocupação com a privacidade e a proteção de dados pessoais, especialmente após a promulgação da Lei Geral de Proteção de Dados (LGPD – Lei nº 13.709/2018), torna-se essencial que as soluções tecnológicas adotem princípios de transparência, segurança e responsabilidade no tratamento das informações coletadas. Assim, o desenvolvimento de um sistema de reconhecimento facial precisa considerar não apenas sua eficiência técnica, mas também os aspectos éticos e legais que envolvem o uso de dados biométricos.

Do ponto de vista acadêmico e científico, o trabalho contribui para a integração entre os campos da visão computacional, aprendizado de máquina e segurança da informação, áreas que têm se mostrado fundamentais para o avanço da computação moderna. Ao empregar redes neurais pré-treinadas para detecção e codificação facial,

o projeto evidencia a aplicabilidade prática da inteligência artificial em problemas reais, demonstrando como modelos matemáticos podem ser incorporados a sistemas de autenticação de maneira funcional e eficiente.

Além disso, o desenvolvimento deste sistema visa promover o aprendizado prático sobre a construção de aplicações web seguras e inteligentes, reforçando competências de programação, modelagem de dados e integração de bibliotecas de IA. O projeto também contribui para a compreensão de conceitos essenciais como tratamento de imagens digitais, extração de características, análise de similaridade vetorial e controle hierárquico de acesso, todos essenciais para profissionais da área de Ciência da Computação.

Portanto, a criação de um sistema de reconhecimento facial para controle de acesso não se limita a um exercício técnico, mas constitui uma iniciativa que alia inovação, segurança e aplicabilidade social. A proposta busca não apenas automatizar o processo de autenticação, mas também oferecer um modelo de arquitetura computacional robusto, capaz de servir como base para futuras pesquisas e aprimoramentos na área de biometria.

Em síntese, este trabalho propõe o desenvolvimento de uma aplicação prática que utiliza a inteligência artificial como ferramenta para fortalecer os mecanismos de segurança digital. Ao combinar técnicas de visão computacional, aprendizado de máquina e engenharia de software, o projeto busca contribuir para a evolução das soluções de autenticação e controle de acesso, evidenciando o potencial das tecnologias emergentes na construção de sistemas mais seguros, eficientes e confiáveis para o mundo contemporâneo.

2. PRINCIPAIS TÉCNICAS DE RECONHECIMENTO FACIAL

2.1 Abordagem Holística

A abordagem holística no reconhecimento facial consiste em uma técnica que considera o rosto como uma unidade integral e indivisível, tratando-o como um padrão global em vez de analisar características isoladas, como a distância entre os olhos ou o formato do nariz. Essa perspectiva parte do princípio de que a identificação facial é mais eficiente quando o sistema leva em conta a configuração e a correlação entre todos os elementos que compõem a imagem do rosto.

Funcionamento da Abordagem Holística

O reconhecimento baseado em abordagem holística fundamenta-se na análise da distribuição geral de luz, sombra e textura que caracteriza a imagem facial. Assim, o rosto é representado por um vetor de alta dimensão, no qual cada valor descreve aspectos do padrão visual global.

Entre suas principais características, destacam-se:

- Ênfase no padrão global: em vez de se restringir a pontos nodais específicos, a técnica considera a imagem completa do rosto, tratando-a como um conjunto de informações interdependentes.
- Robustez a pequenas variações: por utilizar o rosto como um todo, essa abordagem tende a ser menos sensível a alterações sutis de pose, expressão facial ou ruído de fundo.
- Redução de dimensionalidade: devido à complexidade computacional envolvida, são empregados algoritmos capazes de reduzir o espaço de dados, preservando as informações mais relevantes para a distinção entre indivíduos.

Principais Técnicas Holísticas

As técnicas mais representativas dessa abordagem são Eigenfaces e Fisherfaces, amplamente utilizadas em sistemas clássicos de reconhecimento facial.

1. **Eigenfaces** (PCA)
   Baseada na *Análise de Componentes Principais (PCA)*, essa técnica representa cada rosto como uma combinação linear de um conjunto reduzido de “rostos base”, denominados *eigenfaces*. Esses rostos base capturam as principais variações encontradas em um conjunto de imagens de treinamento. Assim, cada indivíduo é descrito por um vetor de características que reflete a aparência geral de seu rosto, sem a necessidade de medir regiões específicas.
2. **Fisherfaces** (LDA)
   Derivada da *Análise Discriminante Linear (LDA)*, a técnica Fisherfaces aprimora o desempenho do método anterior ao priorizar a separação entre classes (diferentes indivíduos) e reduzir a variação dentro da mesma classe (imagens distintas de uma mesma pessoa). Dessa forma, mostra-se mais resistente a variações de iluminação e condições ambientais.

Embora as técnicas de *Deep Learning*, especialmente as redes neurais convolucionais (CNNs), representem atualmente o padrão de excelência em reconhecimento facial, é importante ressaltar que elas mantêm os princípios da abordagem holística. O aprendizado profundo pode ser considerado uma evolução dessa metodologia, incorporando mecanismos mais sofisticados de extração e generalização de padrões a partir da imagem facial completa.

2.2 Abordagem geométrica

A abordagem geométrica no reconhecimento facial fundamenta-se na análise das proporções, formas e estruturas do rosto humano, constituindo uma das técnicas mais tradicionais e pioneiras nesse campo. Diferentemente da abordagem holística — que considera o rosto como um todo indivisível —, a metodologia geométrica enfatiza o estudo das relações métricas entre pontos específicos da face, conhecidos como pontos nodais ou fiduciais. Essa perspectiva parte do princípio de que a disposição e a distância relativa entre tais pontos formam uma assinatura geométrica única, capaz de identificar um indivíduo de maneira confiável.

Princípios da Abordagem Geométrica

O processo inicia-se com a detecção e extração de pontos nodais, localizados em regiões faciais estáveis e facilmente reconhecíveis, como os cantos internos e externos dos olhos, a ponta do nariz, o centro da boca e o contorno do queixo. Uma vez identificados esses pontos de referência, o algoritmo procede à medição das distâncias e ângulos que os interligam — por exemplo:

- distância entre os centros dos olhos;
- largura e altura da boca;
- distância entre a ponta do nariz e o centro do queixo;
- proporção entre a largura e a altura facial.

Essas medições são então convertidas em um vetor de características numérico, conhecido como *impressão facial* (*faceprint*) geométrica. Esse vetor sintetiza a estrutura facial do indivíduo e é posteriormente comparado a registros armazenados em um banco de dados, permitindo tanto a identificação (quem é o usuário) quanto a verificação (se o usuário é quem afirma ser).

Vantagens e Desafios da Abordagem Geométrica

Entre as principais vantagens dessa técnica, destaca-se a clareza interpretativa, pois os parâmetros utilizados baseiam-se em medidas físicas diretamente observáveis. Além disso, o baixo custo computacional e a eficiência de armazenamento tornam o método atrativo para aplicações em dispositivos com recursos limitados, já que os vetores geométricos exigem menos espaço do que imagens completas, como ocorre nas abordagens holísticas (TURK; PENTLAND, 1991).

Entretanto, a abordagem geométrica enfrenta desafios significativos. A variação de pose — mudanças na orientação do rosto — pode distorcer as distâncias medidas e comprometer a precisão. De modo semelhante, expressões faciais (como sorrir ou franzir a testa) e oclusões parciais (provocadas por óculos, barba ou cabelos) alteram a posição de pontos nodais críticos, reduzindo a confiabilidade das medições (BRUNELLI; POGGIO, 1993).

Apesar dessas limitações, a abordagem geométrica mantém relevância nos sistemas modernos de reconhecimento facial. Em arquiteturas baseadas em *Deep Learning*,

como as redes neurais convolucionais (CNNs), seus princípios continuam a ser aplicados nas etapas iniciais de detecção, alinhamento e normalização facial, servindo como pré-processamento para modelos mais complexos (PARKHI; VEDALDI; ZISSERMAN, 2015).

Assim, ainda que o método geométrico puro tenha sido amplamente superado por técnicas baseadas em aprendizado profundo, seu legado permanece essencial para a compreensão e o aprimoramento das tecnologias contemporâneas de autenticação biométrica.

2.3 Redes Neurais Convolucionais (CNNs)

As Redes Neurais Convolucionais (CNNs) representam atualmente a principal tecnologia empregada em sistemas de reconhecimento facial, constituindo um dos pilares do Aprendizado Profundo (Deep Learning). Essa abordagem revolucionou a área da visão computacional ao substituir técnicas tradicionais, baseadas em medições geométricas ou representações holísticas, por métodos capazes de aprender automaticamente características discriminantes a partir de grandes volumes de dados (LECUN; BENGIO; HINTON, 2015).

O funcionamento das CNNs baseia-se em uma sequência de camadas convolucionais que processam a imagem em diferentes níveis de abstração. Nas primeiras camadas, a rede identifica padrões de baixo nível, como bordas, texturas e contornos, enquanto as camadas mais profundas aprendem estruturas complexas, como olhos, nariz e boca, e as relações espaciais entre essas partes (GOODFELLOW; BENGIO; COURVILLE, 2016). Essa hierarquia de aprendizado permite que as CNNs sejam altamente robustas a variações de iluminação, pose e expressão facial.

Após a extração das características, as informações são condensadas em um vetor numérico de alta dimensão, denominado *face embedding* ou *faceprint*, que atua como uma assinatura matemática da face. De acordo com Schroff, Kalenichenko e Philbin (2015), embeddings gerados por redes como o FaceNet apresentam a propriedade de manter representações de uma mesma pessoa próximas entre si no espaço vetorial, mesmo sob condições distintas, enquanto as de indivíduos diferentes permanecem afastadas.

O reconhecimento facial é então realizado por meio da comparação entre embeddings, utilizando métricas como a distância euclidiana ou a similaridade de cosseno para avaliar o grau de proximidade entre as representações vetoriais. Essa metodologia tem alcançado níveis de acurácia superiores à capacidade humana em diversos contextos práticos (TAIGMAN et al., 2014; PARKHI; VEDALDI; ZISSERMAN, 2015).

Entre as principais vantagens das CNNs estão a robustez a condições ambientais adversas, a alta precisão e a capacidade de aprendizado hierárquico, que permite a construção de representações faciais ricas e discriminativas. Modelos consagrados como o DeepFace, o FaceNet e o VGG-Face estabeleceram marcos no desenvolvimento de sistemas modernos de reconhecimento facial, consolidando as CNNs como o paradigma dominante nessa área.

2.4 Reconhecimento Facial 3D

O reconhecimento facial tridimensional (3D) constitui uma evolução natural das abordagens bidimensionais, ao incorporar informações de profundidade e forma geométrica da face humana. Em vez de analisar apenas a projeção plana de uma imagem, essa técnica reconstrói um modelo tridimensional detalhado do rosto, utilizando sensores que capturam dados espaciais por meio de luz estruturada, laser ou infravermelho (BOWYER; CHANG; FLYNN, 2006).

Durante o processo de captura, os sensores projetam padrões luminosos sobre a face e registram a deformação desses padrões, permitindo calcular a posição tridimensional de milhares de pontos. A partir dessas informações, é gerado um modelo digital que descreve com alta precisão as curvaturas, saliências e depressões da superfície facial. As informações geométricas e topológicas extraídas desse modelo são utilizadas para comparação com dados previamente armazenados, permitindo a identificação individual com elevado grau de confiabilidade.

Uma das principais vantagens do reconhecimento facial 3D está na sua independência em relação às condições de iluminação e pose, já que a geometria facial é uma característica intrínseca e estável, independente das sombras ou ângulos de captura (BOWYER; CHANG; FLYNN, 2006). Outro aspecto fundamental é a resistência a tentativas de fraude (spoofing), visto que a reprodução fiel da

profundidade facial em uma imagem bidimensional é extremamente difícil. Além disso, a análise de curvatura facial, que avalia o grau de convexidade e concavidade de cada região do rosto, constitui um parâmetro biométrico único e praticamente impossível de ser replicado com precisão (GOODFELLOW; BENGIO; COURVILLE, 2016).

Graças à sua robustez e segurança aprimorada, o reconhecimento facial 3D tem sido amplamente adotado em sistemas de autenticação de alta confiabilidade, como controle de fronteiras e identificação forense, além de aplicações comerciais, como o desbloqueio biométrico de dispositivos móveis. Essa tecnologia representa, portanto, um avanço significativo rumo a sistemas biométricos mais precisos e seguros.

3. PLANO DE DESENVOLVIMENTO DA APLICAÇÃO

3.1 Desenvolvimento do Sistema

O sistema de reconhecimento facial foi concebido com o propósito de oferecer um controle de acesso automatizado, seguro e eficiente, empregando técnicas modernas de autenticação biométrica.

Durante sua implementação, utilizou-se o Miniconda, uma distribuição leve do Anaconda, responsável pela criação e gerenciamento de ambientes virtuais e pacotes Python. Essa escolha se deve à sua leveza e flexibilidade, possibilitando a instalação exclusiva dos pacotes e dependências realmente necessários ao projeto, sem a necessidade de todo o conjunto de ferramentas da distribuição principal.

Além disso, o Miniconda facilita o gerenciamento de bibliotecas de difícil instalação direta — como as dependentes de C++ — evitando conflitos de versão e garantindo a integridade do ambiente de desenvolvimento.

Para o armazenamento de dados, adotou-se o banco SQLite, utilizado para registrar informações dos usuários, encodings faciais e respectivos níveis de permissão de acesso.

3.2 Tecnologias e Bibliotecas Utilizadas

O sistema foi desenvolvido com o uso de diversas tecnologias e bibliotecas integradas, conforme descrito a seguir:

- views.py (em registro/): responsável pelo processamento das imagens e pela lógica de comparação facial.
- OpenCV (cv2): realiza a captura e manipulação de imagens em tempo real por meio da webcam.
- face_recognition: executa a detecção e codificação facial, gerando vetores numéricos que representam as características de cada rosto.
- NumPy: realiza operações matemáticas sobre os vetores de características faciais.
- Django: provê a estrutura de backend, gerenciamento de rotas e controle de autenticação.
- Pillow (PIL): encarregada do tratamento e conversão de imagens antes do processamento facial.

Essa combinação de ferramentas assegurou um ambiente de desenvolvimento organizado, reprodutível e compatível com diferentes sistemas operacionais, favorecendo tanto os testes quanto a implantação da aplicação final. As bibliotecas atuam de forma integrada para capturar a imagem facial, extrair suas características e compará-las com os dados previamente armazenados no banco.

3.3 Lógica de Funcionamento

O fluxo de execução do sistema segue as seguintes etapas:

1. O usuário com nível máximo de acesso (Nível 3) realiza o cadastro de novos usuários, enviando uma imagem facial.
2. A imagem é processada e convertida em um encoding facial (assinatura numérica única), armazenada no banco de dados.
3. No momento do login, o usuário pode optar entre o método tradicional (usuário e senha) ou o login via reconhecimento facial, utilizando a câmera para capturar a imagem em tempo real.
4. O sistema gera um novo encoding e o compara com os já cadastrados. Caso a similaridade esteja dentro do limite aceitável, o acesso é concedido.

Níveis de Acesso:

- Nível 1 – Acesso Geral: destinado a usuários com permissões básicas e acesso restrito a funções limitadas do sistema.
- Nível 2 – Acesso de Diretor: concedido a diretores de divisões específicas, com acesso ampliado a relatórios e controles.
- Nível 3 – Acesso de Ministro: reservado ao administrador principal, com privilégios totais de cadastro e gerenciamento de usuários e níveis de acesso.

Essa hierarquia de permissões assegura o controle refinado sobre os dados e previne acessos não autorizados a informações sensíveis.

3.4 Tratamento de Erros e Garantia de Qualidade

Foram implementados diversos mecanismos de tratamento de exceções e controle de qualidade, dentre os quais:

- Verificação automática da disponibilidade da webcam e da captura de imagem;
- Exibição de mensagens de alerta em casos de falha na detecção facial;
- Validação de credenciais incorretas e autenticações inválidas.

O sistema foi submetido a testes sob diferentes condições de luminosidade e distância, comprovando sua robustez e confiabilidade. Sua estrutura modular permite futuras expansões, como geração de relatórios de acesso, alteração dinâmica de níveis de permissão e implementação de técnicas de prova de vida (*liveness detection*).

3.5 Documentação e Execução

O repositório do projeto contém um arquivo README.md com instruções detalhadas para instalação e execução, incluindo:

- Criação do ambiente virtual Conda;
- Instalação das dependências listadas em requirements.txt (incluindo a biblioteca dlib);

- Inicialização do servidor local via python manage.py runserver;
- Acesso à aplicação pelo endereço padrão http://127.0.0.1:8000/.

Embora utilize a biblioteca face_recognition, que incorpora um modelo de rede neural pré-treinado, o sistema não requer o arquivo trainer.yml, uma vez que não realiza treinamento próprio — ele apenas utiliza o modelo existente para geração e comparação dos encodings faciais.

4. FASES DE PROCESSAMENTO DE IMAGEM

4.1 Sensor (Aquisição)

Bibliotecas envolvidas: OpenCV (cv2) e JavaScript.
Descrição: O sistema captura imagens diretamente da webcam. No frontend, o JavaScript acessa a câmera e envia os quadros para o backend, onde o OpenCV realiza o processamento inicial.
Função: Etapa responsável pela aquisição da imagem facial, etapa fundamental para o início do reconhecimento.

4.2 Pré-processamento

Bibliotecas: face_recognition, cv2, Pillow.
Descrição: Após a captura, a imagem é convertida para o formato RGB (padrão requerido pela biblioteca face_recognition) e redimensionada, garantindo uniformidade e qualidade na análise.
Função: O pré-processamento reduz ruídos e padroniza as imagens, otimizando a precisão do modelo.

4.3 Segmentação

Bibliotecas: face_recognition.
Descrição:

face_locations = face_recognition.face_locations(rgb_frame)

Essa função identifica as regiões da imagem que contêm rostos, retornando coordenadas que delimitam as faces detectadas.
Função: Etapa de segmentação, responsável por isolar a região facial relevante para as análises seguintes.

4.4 Extração de Características

Bibliotecas: face_recognition, NumPy, pickle.
Descrição:

face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

Cada rosto é transformado em um vetor de 128 dimensões, representando numericamente suas características únicas. Esses vetores são armazenados e utilizados nas comparações de reconhecimento.
Função: Responsável pela conversão da face em uma assinatura digital exclusiva, base do processo de autenticação.

4.5 Classificação e Interpretação

Bibliotecas: face_recognition, NumPy, Django ORM.
Descrição:

matches = face_recognition.compare_faces(known_face_encodings, face_encoding)

O sistema compara o encoding atual com os armazenados no banco de dados. Quando há correspondência, o usuário é identificado e autenticado.
Função: Etapa final do processo, na qual o sistema interpreta a identidade do usuário e concede ou nega o acesso conforme a validação.

5. PROJETO (ESTRUTURA E MÓDULOS DESENVOLVIDOS)

5.1 Arquitetura do Sistema

O projeto foi estruturado utilizando o framework Django, conforme o seguinte esquema:

```
projeto_reconhecimento_facial/
├── gestao/                 # Configurações principais
│   ├── settings.py
│   └── urls.py
├── registro_recognition/
│   └── registro/
│       ├── models.py       # Modelos de dados (Funcionario, ColetaDeFaces)
│       ├── views.py        # Lógica de negócio e processamento de imagem
│       ├── forms.py        # Formulários
│       └── templates/      # Interfaces HTML
├── static/
│   ├── css/
│   └── js/
└── media/                  # Arquivos enviados (fotos dos funcionários)
```

5.2 Sistema de Níveis de Acesso

O sistema implementa um modelo hierárquico de permissões, definido no modelo Funcionario, conforme exemplo:

```
class Funcionario(models.Model):
    NIVEL_CHOICES = [
        (1, 'Nível 1: Acesso Geral'),
        (2, 'Nível 2: Acesso de Diretor'),
        (3, 'Nível 3: Acesso de Ministro'),
    ]
    nivel_acesso = models.IntegerField(choices=NIVEL_CHOICES, default=1)
```

Nos templates e views, o atributo funcionario.nivel_acesso é utilizado para controlar a visibilidade de informações e funções. Por exemplo, o botão *“Cadastrar Funcionário”* é exibido apenas para usuários com nível de acesso superior a 1.

5.3 Aplicação de Inteligência Artificial

O sistema integra técnicas de Inteligência Artificial por meio da biblioteca face_recognition, baseada em redes neurais convolucionais (CNNs) pré-treinadas para detecção e reconhecimento facial.

Etapas de IA:

- Detecção de rostos: identifica automaticamente as regiões faciais nas imagens capturadas.
- Extração de características: converte as feições em vetores numéricos de 128 dimensões (*face encodings*).
- Comparação e reconhecimento: calcula a similaridade entre o vetor atual e os armazenados, determinando a identidade do usuário.

Esses processos caracterizam o uso de técnicas de aprendizado supervisionado, reconhecimento de padrões visuais e decisão automatizada, evidenciando a integração entre visão computacional, aprendizado de máquina e engenharia de software.

6. APRESENTAÇÃO DO PROGRAMA EM FUNCIONAMENTO

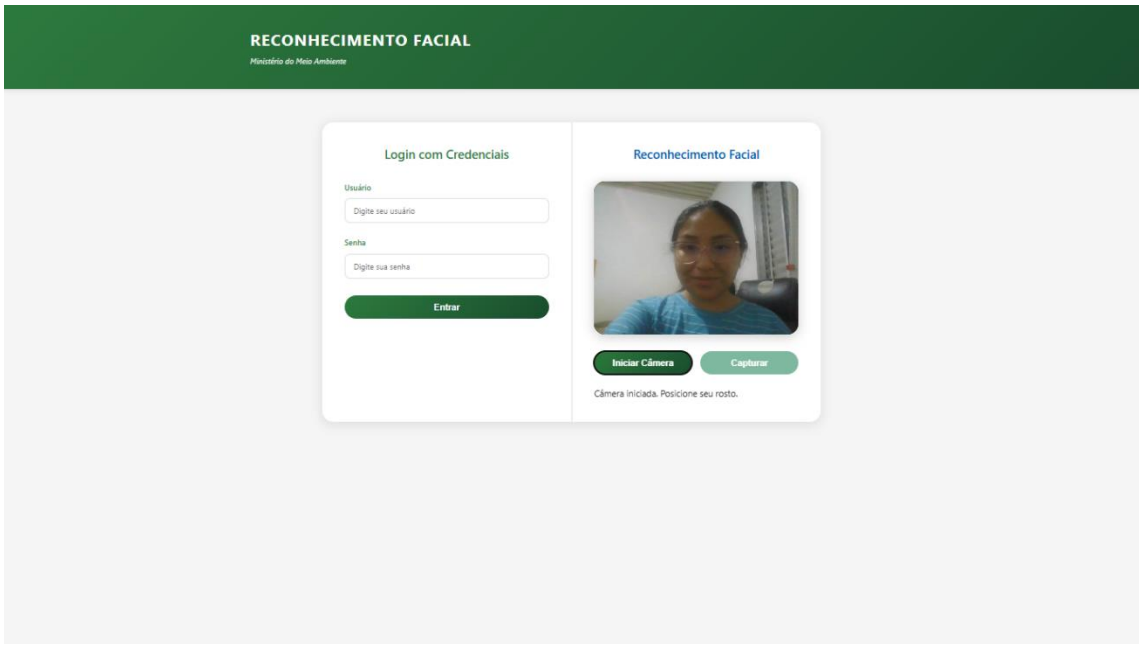
1. Tela de Login

Descrição:
Interface inicial do sistema, responsável por autenticar os usuários. Oferece dois métodos de acesso:

1. Autenticação tradicional com nome de usuário e senha;
2. Login por reconhecimento facial, que utiliza a webcam para identificar o usuário automaticamente.

Funcionalidades Principais:

- Captura e análise facial em tempo real;
- Comparação de encodings para validação do acesso;
- Integração com o sistema de níveis de permissão.
- Formulário para login com credenciais
- Botão para ativar a câmera e iniciar o processo de reconhecimento facial
- Feedback visual em tempo real durante a tentativa de reconhecimento (ex: "Aguardando captura...", "Rosto não reconhecido")



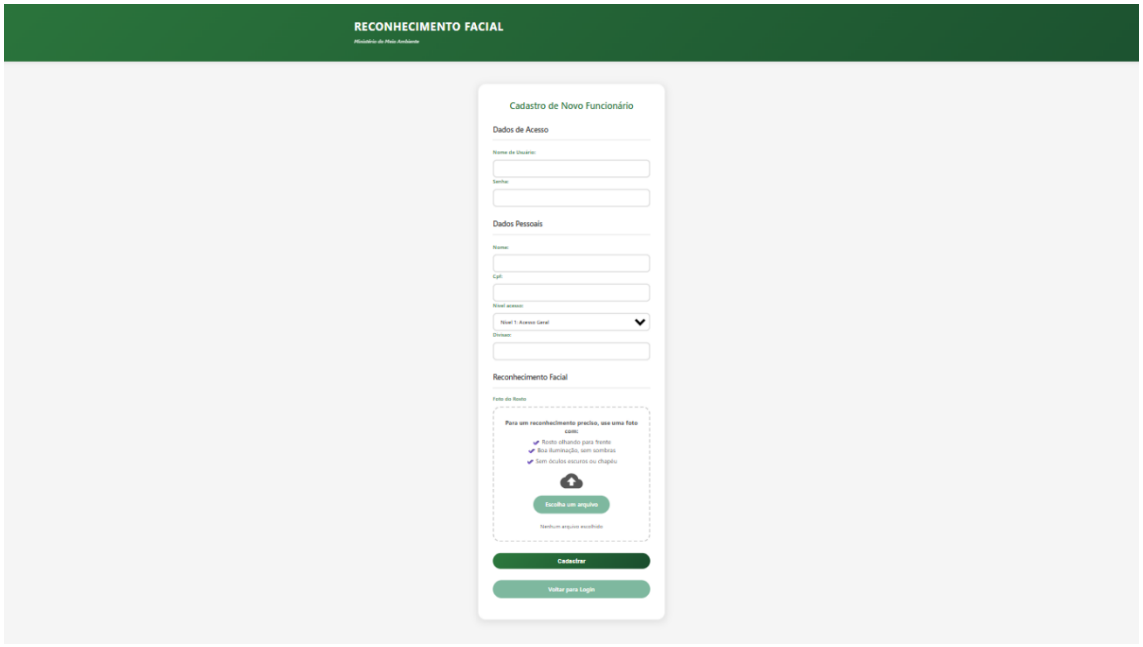
2. Tela de Cadastro de Funcionário

Descrição:

Acessível apenas por usuários com nível de acesso máximo (Nível 3 - Ministro), esta tela permite o registro de novos funcionários no sistema.

Funcionalidades Chave:

- Formulário para inserção de dados pessoais (nome, CPF, etc.)
- Campo para upload da foto do funcionário, que será usada para gerar a assinatura facial
- Seleção do nível de acesso do novo usuário (Nível 1, 2 ou 3)



3. Dashboard do Usuário

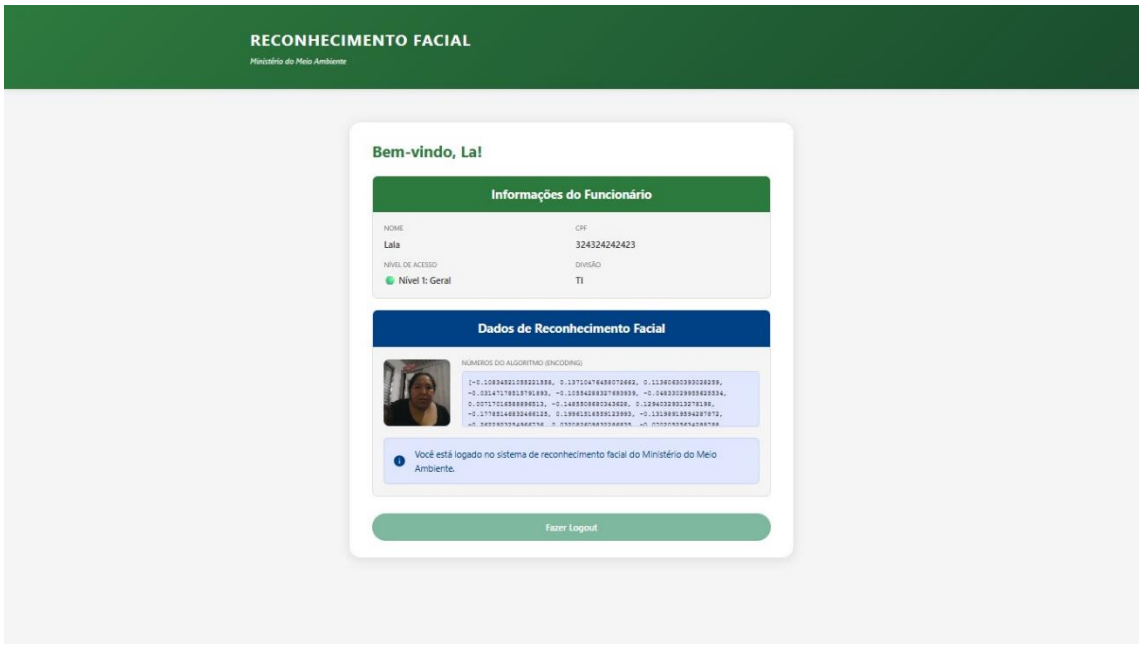
Descrição:

Após o login, o usuário é direcionado para seu painel pessoal, que exibe suas informações de forma clara e segura, adaptando o conteúdo exibido com base no seu nível de acesso.

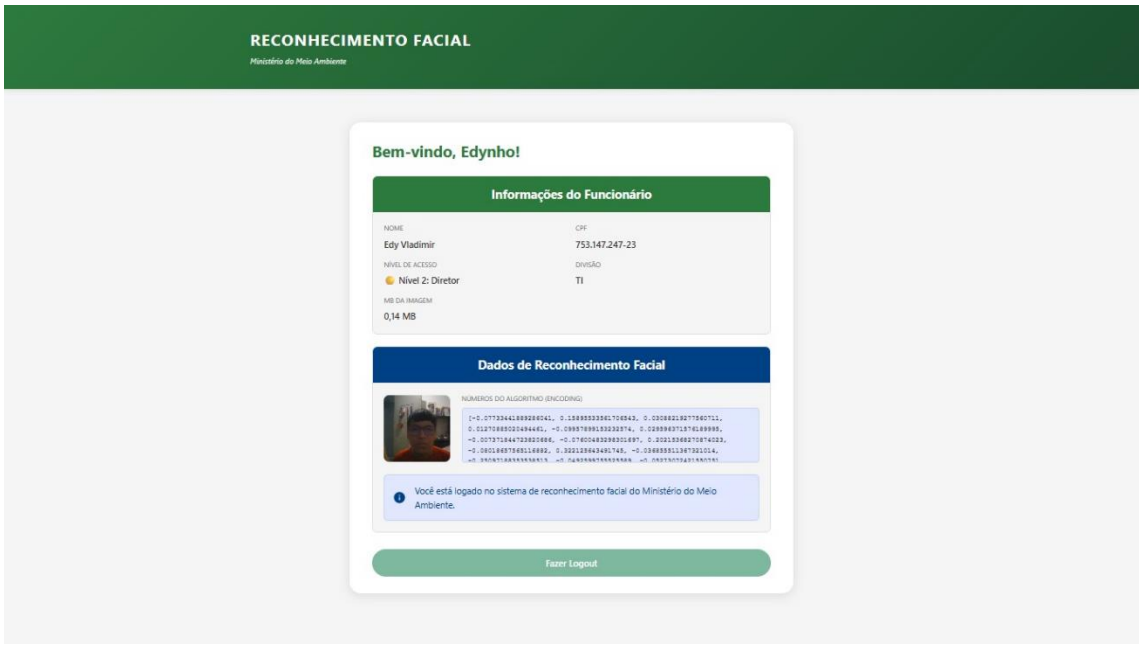
Funcionalidades Chave:

- Exibição de informações como nome, CPF, divisão e nível de acesso
- Para níveis de acesso mais altos, exibe dados técnicos como o tamanho da imagem e o encoding facial
- O botão "Cadastrar Funcionário" aparece apenas para usuários de Nível 3

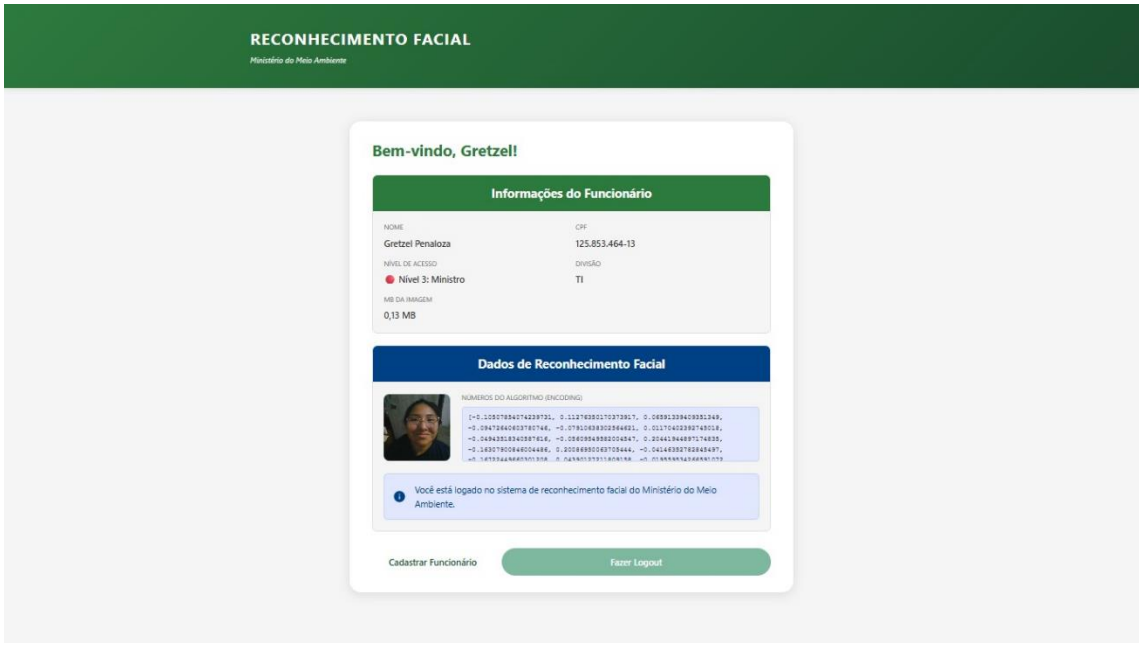
NÍVEL 1



NÍVEL 2



NÍVEL 3


RECONHECIMENTO FACIAL
Ministério do Meio Ambiente
Bem-vindo, Gretzel!
Informações do Funcionário
NOME
Gretzel Penaloza
CPF
125.853.464-13
NÍVEL DE ACESSO
Nível 3: Ministro
DIVISÃO
TI
MB DA IMAGEM
0,13 MB
Dados de Reconhecimento Facial
NÚMEROS DO ALGORITMO (ENCODING)
Você está logado no sistema de reconhecimento facial do Ministério do Meio Ambiente.
Cadastrar Funcionário
Fazer Logout

7. PROJETO DO PROGRAMA

Manage.py

```python
import os
import sys
from django.core.management import execute_from_command_line

def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'gestao.settings')
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

E o **controlador de comandos administrativos** do Django. Ele configura o ambiente e executa qualquer comando que você digitar relacionado ao projeto — como iniciar o servidor, gerenciar o banco de dados, criar apps, etc.

Asgi.py

```python
"""
ASGI config for gestao project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'gestao.settings')

application = get_asgi_application()
```

É o ponto de entrada assíncrono **do projeto usado quando o Django roda com servidores que suportam** ASGI (Asynchronous Server Gateway Interface)**.**

Settings.py

Guarda **todas as configurações principais** do sistema e define **como o Django deve funcionar** desde o banco de dados e pastas até autenticação, idioma, arquivos estáticos e apps do projeto.

```python
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = 'django-insecure-s0f78v3c(%cy5+mh-6z!1-v%efkb9*dt+)@*0)ixuv4q+wcgn5'

DEBUG = True

ALLOWED_HOSTS = []  # ou ['*'] se quiser acessar de qualquer lugar localmente

# Apps instalados
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'registro_recognition.registro',
]

# Middlewares
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'gestao.urls'

WSGI_APPLICATION = 'gestao.wsgi.application'  # <-- CORRETO AQUI
```

```python
import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'registro_recognition' / 'registro' / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]



# Banco de dados SQLite
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

```python
85
86      # Validação de senha
87      AUTH_PASSWORD_VALIDATORS = [
88          {'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator'},
89          {'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator'},
90          {'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator'},
91          {'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator'},
92      ]
93
94      # Localização
95      LANGUAGE_CODE = 'pt-br'
96      TIME_ZONE = 'America/Sao_Paulo'
97      USE_I18N = True
98      USE_TZ = True
99
100     # Arquivos estáticos
101     STATIC_URL = '/static/'
102     STATICFILES_DIRS = [BASE_DIR / 'static']  # onde você coloca seus arquivos estáticos durante o desenvolvimento
103
104     # Para collectstatic funcionar
105     STATIC_ROOT = BASE_DIR / 'staticfiles'  # pasta que receberá os arquivos coletados
106
107
108     # Arquivos de mídia (ex: imagens capturadas)
109     MEDIA_URL = '/media/'
110     MEDIA_ROOT = BASE_DIR / 'media'
111
112     # Chave primária padrão
113     DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
114
115     # Redirecionamento de login/logout
116     LOGIN_URL = 'login'
117     LOGIN_REDIRECT_URL = 'dashboard'
```

Urls.py

```python
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from django.views.generic import TemplateView
from registro_recognition.registro import views



urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('registro_recognition.registro.urls')),

]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

É **o** mapa de navegação do projeto Django ele conecta os endereços (URLs) às funções (views) que exibem as páginas.

Wsgi.py

```python
"""
WSGI config for gestao project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'gestao.settings')

application = get_wsgi_application()
```

É um dos arquivos principais do Django serve **como** ponto de entrada do servidor web para rodar a aplicação

Views.py

```python
from django.shortcuts import render, redirect
from django.contrib.auth import login, authenticate, logout
from django.contrib.auth.decorators import login_required
from django.http import JsonResponse
from .models import Funcionario, ColetaDeFaces
from django.contrib.auth.models import User
from .forms import FuncionarioForm, UserForm, ColetaDeFacesForm
import base64
from django.db import transaction
import cv2
import numpy as np
from PIL import Image
import face_recognition
import json
import os

def login_view(request):
    """Página de login com usuário/senha e reconhecimento facial."""
    if request.method == 'POST':
        data = request.POST
        username = data.get('username')
        password = data.get('password')
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('dashboard')
        else:
            return render(request, 'registro/login.html', {'error': 'Usuário ou senha inválidos'})
```

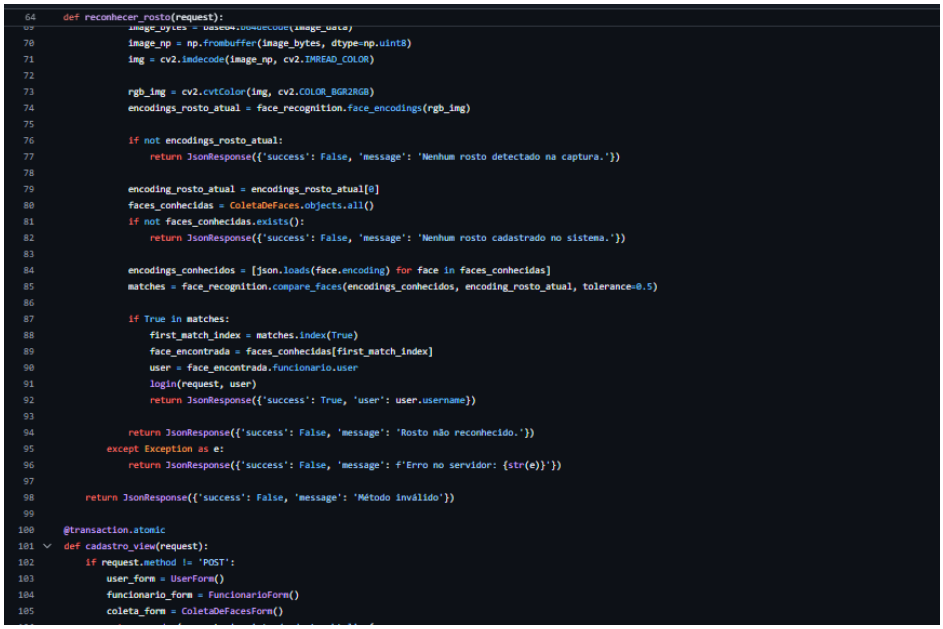
```python
@login_required
def dashboard(request):
    """Dashboard do usuário após o login."""
    try:
        funcionario = request.user.funcionario
        face_data = ColetaDeFaces.objects.get(funcionario=funcionario)
    except Funcionario.DoesNotExist:
        funcionario = None
        face_data = None
    except ColetaDeFaces.DoesNotExist:
        face_data = None

    # Calcular tamanho da imagem em MB
    mb_image = None
    if face_data and face_data.image:
        path = face_data.image.path
        if os.path.exists(path):
            size_bytes = os.path.getsize(path)
            size_mb = size_bytes / (1024 * 1024)
            mb_image = round(size_mb, 2)

    return render(request, 'registro/dashboard.html', {
        'funcionario': funcionario,
        'face_data': face_data,
        'mb_image': mb_image,
    })

@login_required
def logout_view(request):
    """Faz o logout do usuário."""
    logout(request)
    return redirect('login')

def reconhecer_rosto(request):
    if request.method == 'POST':
        try:
```



```
def reconhecer_rosto(request):
            image_bytes = base64.b64decode(image_data)
            image_np = np.frombuffer(image_bytes, dtype=np.uint8)
            img = cv2.imdecode(image_np, cv2.IMREAD_COLOR)

            rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            encodings_rosto_atual = face_recognition.face_encodings(rgb_img)

            if not encodings_rosto_atual:
                return JsonResponse({'success': False, 'message': 'Nenhum rosto detectado na captura.'})

            encoding_rosto_atual = encodings_rosto_atual[0]
            faces_conhecidas = ColetaDeFaces.objects.all()
            if not faces_conhecidas.exists():
                return JsonResponse({'success': False, 'message': 'Nenhum rosto cadastrado no sistema.'})

            encodings_conhecidos = [json.loads(face.encoding) for face in faces_conhecidas]
            matches = face_recognition.compare_faces(encodings_conhecidos, encoding_rosto_atual, tolerance=0.5)

            if True in matches:
                first_match_index = matches.index(True)
                face_encontrada = faces_conhecidas[first_match_index]
                user = face_encontrada.funcionario.user
                login(request, user)
                return JsonResponse({'success': True, 'user': user.username})

            return JsonResponse({'success': False, 'message': 'Rosto não reconhecido.'})
        except Exception as e:
            return JsonResponse({'success': False, 'message': f'Erro no servidor: {str(e)}'})

    return JsonResponse({'success': False, 'message': 'Método inválido'})

@transaction.atomic
def cadastro_view(request):
    if request.method != 'POST':
        user_form = UserForm()
        funcionario_form = FuncionarioForm()
        coleta_form = ColetaDeFacesForm()
```

É onde você cria as **funções (ou classes)** que **recebem as requisições do usuário**, **processam os dados** e **devolvem uma resposta.**

Models.py

```python
# ...existing code...
from django.db import models
from django.contrib.auth.models import User
from django.utils.text import slugify

class Funcionario(models.Model):
    """Modelo para armazenar dados dos funcionários"""
    NIVEL_CHOICES = [
        (1, 'Nível 1: Acesso Geral'),
        (2, 'Nível 2: Acesso restrito para Diretor'),
        (3, 'Nível 3: Acesso exclusivo para Ministro'),
    ]

    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='funcionario')
    nome = models.CharField(max_length=255)
    cpf = models.CharField(max_length=14, unique=True)
    nivel_acesso = models.IntegerField(choices=NIVEL_CHOICES, default=1)
    divisao = models.CharField(max_length=255, blank=True, null=True)
    slug = models.SlugField(unique=True, blank=True, help_text="Deixe em branco para gerar automatica
    criado_em = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.nome

    def save(self, *args, **kwargs):
        if not self.slug:
            base_slug = slugify(self.nome)
            self.slug = base_slug
        super().save(*args, **kwargs)

class ColetaDeFaces(models.Model):
    """Modelo para armazenar encodings faciais"""
    funcionario = models.OneToOneField(Funcionario, on_delete=models.CASCADE, related_name='face_data
    image = models.ImageField(upload_to='faces/', blank=True, null=True)
    encoding = models.TextField()  # JSON string do encoding facial
    criado_em = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Face de {self.funcionario.nome}"
```

É onde você **define as tabelas e os campos** que o Django vai criar e gerenciar automaticamente.

Forms.py

```python
from django import forms
from .models import Funcionario

class FuncionarioForm(forms.ModelForm):
    """Formulário para os dados básicos do funcionário."""
    class Meta:
        model = Funcionario
        fields = ['nome', 'cpf', 'nivel_acesso', 'divisao']

class UserForm(forms.Form):
    """Formulário para os dados do usuário (login)."""
    username = forms.CharField(max_length=150, label="Nome de Usu
    password = forms.CharField(widget=forms.PasswordInput, label="

class ColetaDeFacesForm(forms.Form):
    """Formulário para a imagem do rosto."""
    image = forms.ImageField(label="Foto do Rosto")
```

O arquivo no Django serve para **criar e gerenciar formulários** de maneira simples, segura e integrada com os modelos do banco de dados

Admin

```python
from django.contrib import admin
from .models import Funcionario, ColetaDeFaces
admin.site.register(Funcionario)
admin.site.register(ColetaDeFaces)
```

registra os modelos (tabelas) **do seu aplicativo Django no** painel administrativo **do Django**

Train_faces.py

```python
# train_faces.py
# Script para gerar encodings.pkl a partir do modelo ColetaDeFaces
import os, django, pickle
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'gestao.settings')
django.setup()
from registro.models import ColetaDeFaces
import face_recognition

def treinar_faces():
    encodings, nomes, niveis = [], [], []
    coletas = ColetaDeFaces.objects.all()
    for coleta in coletas:
        try:
            imagem_path = coleta.imagem.path
        except Exception as e:
            print('⚠ Não foi possível obter o caminho da imagem para', coleta.nome, e)
            continue
        img = face_recognition.load_image_file(imagem_path)
        faces = face_recognition.face_encodings(img)
        if faces:
            encodings.append(faces[0])
            nomes.append(coleta.nome)
            niveis.append(coleta.nivel)
        else:
            print('⚠ Nenhum rosto detectado em:', coleta.nome)

    with open('encodings.pkl', 'wb') as f:
        pickle.dump((encodings, nomes, niveis), f)
    print('✅ Treinamento concluído. encodings.pkl salvo.')

if __name__ == '__main__':
    treinar_faces()
```

Treinar o modelo com base nas imagens de rosto que estão salvas na pasta de coleta (dentro de media/faces/).

8. CONCLUSÃO

O projeto de reconhecimento facial implementa todas as etapas exigidas, da aquisição à classificação, utilizando bibliotecas robustas e modelos pré-treinados. Ele já integra um controle de níveis de acesso, combinando o processamento de imagem com a lógica de permissões do framework Django.

O sistema desenvolvido se destaca pela integração entre diferentes áreas — inteligência artificial, desenvolvimento web e segurança digital — aplicando conceitos práticos de visão computacional em um ambiente funcional e acessível.

Com um design modular, documentação clara e tratamento de erros eficaz, o sistema demonstra robustez, funcionalidade e boa estrutura técnica, atendendo aos critérios de qualidade exigidos no projeto.

9. REFERÊNCIAS

- TURK, M.; PENTLAND, A. *Eigenfaces for recognition.* Journal of Cognitive Neuroscience, 1991.
- BRUNELLI, R.; POGGIO, T. *Face recognition: Features versus templates.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1993.
- PARKHI, O. M.; VEDALDI, A.; ZISSERMAN, A. *Deep Face Recognition.* Proceedings of the British Machine Vision Conference, 2015.
- BOWYER, K. W.; CHANG, K.; FLYNN, P. *A survey of approaches and challenges in 3D and multi-modal 3D+2D face recognition*. *Computer Vision and Image Understanding*, v. 101, n. 1, p. 1–15, 2006.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. Cambridge: MIT Press, 2016.
- LECUN, Y.; BENGIO, Y.; HINTON, G. *Deep learning*. *Nature*, v. 521, n. 7553, p. 436–444, 2015.
- PARKHI, O. M.; VEDALDI, A.; ZISSERMAN, A. *Deep Face Recognition*. In: *British Machine Vision Conference (BMVC)*. Swansea: BMVA Press, 2015.
- SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. *FaceNet: A Unified Embedding for Face Recognition and Clustering*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston: IEEE, 2015.
- TAIGMAN, Y. et al. *DeepFace: Closing the Gap to Human-Level Performance in Face Verification*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Columbus: IEEE, 2014.

10. FICHAS APS

UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES COMPLEMENTARES

NOME: Solange Araujo de Santana TURMA: CC6P33 RA: N069228

CURSO: Ciência da Computação CAMPUS: Tatuape SEMESTRE: 6 TURNO: Noturno

CÓDIGO DA ATIVIDADE: SEMESTRE: 6 ANO GRADE: 2025

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
26/08	formacao dos grupos	10	Solange Araujo		
30/08	pesquisa sobre projeto	20	Solange Araujo		
27/09	pesquisa sobre framework	10	Solange Araujo		
04/10	codigo do projeto	20	Solange Araujo		
07/11	documentacao	10	Solange Araujo		

(1) Horas atribuídas de acordo com o regulamento das Atividades Complementares do curso.

TOTAL DE HORAS ATRIBUÍDAS:

AVALIAÇÃO:
Aprovado ou Reprovado

NOTA:

DATA:____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Matheus Marcelino de Oliveira TURMA: CC6P33 RA: G79GHB-8

CURSO: Ciência da Computação CAMPUS: Tatuapé SEMESTRE: 6° TURNO: Noturno

CÓDIGO DA ATIVIDADE: SEMESTRE: 6° ANO GRADE: 2025

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
26/08	Organização do Grupo	10 Horas			
29/08	Pesquisa do Projeto	10 Horas			
15/09	Pesquisa de Bibliotecas do Python	08 Horas			
28/09	Reunião do Grupo	10 Horas			
03/11	Pesquisa sobre Django	11 Horas			
04/11	Pesquisa sobre Mineconda	06 Horas			
05/11	Implementação do sistema de análise de Cargo	05 Horas			
09/11	Implementação do sistema de Disposição de Dados	10 Horas			
10/11	Finalizações e Correção de Bugs	05 Horas			

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS:

AVALIAÇÃO:
Aprovado ou Reprovado

NOTA:

DATA:____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Gretzel Kattia Laura Penaloza TURMA: CC5P33 RA: N158930

CURSO: Ciência da Computação CAMPUS: Tatuapé SEMESTRE: 5° TURNO: Noturno

CÓDIGO DA ATIVIDADE: SEMESTRE: 5° ANO GRADE: 2025

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
26/08	organização do grupo	10 horas			
30/08	pesquisa de projeto	10 horas			
28/09	reunião com o grupo	10 horas			
29/09	ajustes na parte back end	12 horas			
3/11	ajustes na parte de front end	8 horas			
2/11	documentação Readme.md	4 horas			
7/11	correção de bug na tela	6 horas			
8/11	rasura da dissertação	5 horas			
9/11	atualização de front-end	6 horas			
10/11	refatorar código	5 horas			

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS:

AVALIAÇÃO:
Aprovado ou Reprovado

NOTA:

DATA: / /

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Gabriele Martins Dias TURMA: CC6P33 RA: 9847E6

CURSO: Ciências da Computação CAMPUS: Tatuapé SEMESTRE: 6ª TURNO: Noturno

CÓDIGO DA ATIVIDADE: SEMESTRE: 6ª ANO GRADE: 2025

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
26/08	organização do grupo	10			
30/08	pesquisa do projeto	20			
28/09	reunião do grupo	10			
05/10	Codigo do projeto	20			
03/11	Documentação	10			
06/11	revisão	5			

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS:

AVALIAÇÃO:
Aprovado ou Reprovado

NOTA:

DATA: / /

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO