UNIVERSIDADE PAULISTA – UNIP

CIÊNCIA DA COMPUTAÇÃO

TRABALHO SEMESTRAL APS

PAULO HENRIQUE MESSIAS SILVA – G5463D-8

LUCAS MELO PESSOA – N016159

HENRIQUE KRIIGEN BAIONE MIRANDA – N0128G-0

Reconhecimento Facial com Hierarquia: Linguagem Java

Versão 1

Jundiaí/SP

15/06/2025

GRUPO DE TRABALHO APS

Reconhecimento Facial com Hierarquia: Linguagem Java

Versão 1

Trabalho de Conclusão de Semestre apresentado ao Professor Caco da Universidade Paulista – Unip do Curso de Ciência da Computação.

Orientador: Prof. Dr/Ms. Marcos Paulo

Orientador: Gerson Oliveira.

Jundiaí/SP

15/06/2025

**RESUMO**

Este documento detalha o desenvolvimento de uma aplicação de software em Java, cujo objetivo é realizar o reconhecimento facial para autenticação e controle de acesso de usuários. O sistema utiliza a biblioteca JavaCV para implementar as funcionalidades de visão computacional e emprega o algoritmo FisherFaces para o treinamento do modelo de reconhecimento. Um diferencial do projeto é a implementação de um sistema de controle de acesso com múltiplos níveis hierárquicos, que restringem as funcionalidades disponíveis a cada tipo de usuário, garantindo a segurança e a integridade dos dados gerenciados pela aplicação. A arquitetura do software segue o padrão de projeto MVC (Model-View-Controller), com classes de modelo para representar as entidades, classes de visão para a interface gráfica e classes de controle para a lógica de negócio e persistência de dados em um banco de dados SQL Server.

Sumário

[2.1 Objetivo Geral 5](#_Toc201011012)

[2.2 Objetivos Específicos 5](#_Toc201011013)

[3.1 Ferramentas e Tecnologias 6](#_Toc201011014)

[3.2 Processo de Reconhecimento Facial 6](#_Toc201011015)

[3.2.1 Detecção e Captura de Amostras 6](#_Toc201011016)

[3.2.2 Treinamento do Modelo com FisherFaces 6](#_Toc201011017)

[3.2.3 Reconhecimento e Autenticação em Tempo Real 7](#_Toc201011018)

[3.3 Controle de Acesso e Níveis Hierárquicos 7](#_Toc201011019)

[4.1 Camada de Modelo (Model) 7](#_Toc201011020)

[4.2 Camada de Persistência de Dados (DAO) 8](#_Toc201011021)

[4.3 Módulos de Controle e Lógica de Negócio 8](#_Toc201011022)

[4.4 Camada de Visão (View) 8](#_Toc201011023)

[4.5 Fluxo de Operação Integrado 8](#_Toc201011024)

1 INTRODUÇÃO

O reconhecimento facial é uma tecnologia de visão computacional que permite identificar ou verificar a identidade de um indivíduo a partir de uma imagem ou vídeo. Suas aplicações são vastas, abrangendo sistemas de segurança, autenticação de usuários e monitoramento.

Este trabalho apresenta o desenvolvimento de uma aplicação em Java que implementa funcionalidades de reconhecimento facial integradas a um sistema de controle de acesso baseado em níveis hierárquicos. O projeto foi concebido para oferecer uma solução robusta e segura, onde o acesso às funcionalidades é determinado pelo perfil do usuário autenticado.

2 OBJETIVOS

2.1 Objetivo Geral

O objetivo principal deste trabalho foi desenvolver uma aplicação desktop em Java que utilize técnicas de visão computacional para realizar o reconhecimento facial e, com base na identidade confirmada, gerenciar o acesso dos usuários a diferentes funcionalidades do sistema por meio de uma estrutura hierárquica de permissões.

2.2 Objetivos Específicos

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

* Utilizar a linguagem Java e a biblioteca JavaCV para as funcionalidades de visão computacional.
* Implementar um sistema de captura e treinamento de imagens faciais utilizando o algoritmo **FisherFaces**.
* Estruturar um sistema de acesso com quatro níveis hierárquicos distintos.
* Desenvolver a persistência dos dados de usuários e clientes em um banco de dados SQL Server.
* Criar interfaces gráficas intuitivas para cada nível de acesso.

3 METODOLOGIA

A metodologia adotada para o desenvolvimento do projeto abrange a seleção de tecnologias, o processo de treinamento do modelo de reconhecimento e a implementação da lógica de controle de acesso.

3.1 Ferramentas e Tecnologias

A linguagem de programação escolhida foi o **Java**, devido à sua portabilidade, robustez e ao vasto ecossistema de bibliotecas. Para a implementação do reconhecimento facial, foi utilizada a biblioteca de código aberto **JavaCV**, que atua como um wrapper para a popular biblioteca OpenCV. A interface gráfica do usuário (GUI) foi desenvolvida com a biblioteca **Swing**, e a persistência de dados foi realizada em um banco de dados **Microsoft SQL Server**.

3.2 Processo de Reconhecimento Facial

O funcionamento do sistema de reconhecimento é composto por duas fases fundamentais: o cadastro e treinamento, onde o sistema aprende a identificar um usuário, e o reconhecimento em tempo real para autenticação.

3.2.1 Detecção e Captura de Amostras

O processo de cadastro de um novo usuário é iniciado pelo Administrador. Ao acionar a funcionalidade de captura, o sistema utiliza um classificador **Haar Cascade** (haarcascade\_frontalface\_alt.xml) para detectar, em tempo real, a presença de um rosto no feed de vídeo da câmera. Uma vez que o rosto é localizado, a região correspondente é processada e armazenada. Para garantir a robustez do modelo, um total de **25 amostras** faciais são capturadas. Cada amostra passa por um pré-processamento que consiste em:

1. **Conversão para escala de cinza:** A informação de cor é removida para reduzir a complexidade computacional.
2. **Redimensionamento:** A imagem é padronizada para uma resolução de 160x160 pixels, assegurando consistência para a etapa de treinamento.

As imagens processadas são salvas em disco com uma nomenclatura padronizada (pessoa.[ID].[numero\_da\_amostra].jpg), que vincula inequivocamente cada imagem ao ID único do funcionário no banco de dados.

3.2.2 Treinamento do Modelo com FisherFaces

Com as amostras faciais coletadas, o Administrador aciona o módulo de treinamento. Este módulo varre o diretório de imagens e constrói duas estruturas de dados: um vetor contendo todas as imagens faciais (MatVector) e uma matriz de rótulos (labels) contendo os IDs correspondentes extraídos dos nomes dos arquivos.

Essas estruturas são então passadas para o reconhecedor **FisherFaceRecognizer**. Este algoritmo, baseado em Análise Discriminante Linear (LDA), analisa o conjunto de dados para encontrar as características faciais que melhor distinguem uma pessoa da outra. Ao final, o conhecimento adquirido é serializado e salvo no arquivo **classificadosFisherFaces.yml**. Este arquivo representa o modelo de reconhecimento treinado, pronto para ser usado.

3.2.3 Reconhecimento e Autenticação em Tempo Real

Na tela de login, a câmera é ativada e o modelo classificadosFisherFaces.yml é carregado na memória. O sistema detecta o rosto do usuário em tempo real e, para cada frame, o método **predict()** do reconhecedor é invocado. Este método compara o rosto detectado com os padrões aprendidos e retorna dois valores: o **ID** do usuário mais provável e uma **pontuação de confiança**.

Uma pontuação de confiança baixa indica uma alta semelhança com o modelo treinado. Se essa pontuação estiver abaixo de um limiar pré-definido, a identidade do usuário é confirmada.

3.3 Controle de Acesso e Níveis Hierárquicos

Uma vez que a autenticação facial é bem-sucedida, o ID do usuário é utilizado para consultar seu cargo no banco de dados. Com base no cargo, o sistema libera o acesso a diferentes módulos, conforme a hierarquia de quatro níveis definida:

* **Nível 1 - Funcionário:** Acesso para cadastro de clientes e atualização de status de serviços.
* **Nível 2 - Gerente:** Permissões do Nível 1, mais a capacidade de consultar registros e gerenciar dados sensíveis de clientes.
* **Nível 3 - Diretor:** Visão geral completa de dados, com permissão para gerenciar valores financeiros.
* **Nível 4 - Administrador:** Acesso irrestrito para gerenciamento de usuários e configurações do sistema.

4 ARQUITETURA E FLUXO DE OPERAÇÃO DO SOFTWARE

A aplicação foi estruturada em camadas lógicas para promover a organização e manutenibilidade, seguindo princípios do padrão MVC.

4.1 Camada de Modelo (Model)

Representa as entidades do sistema:

* **Pessoa.java:** Classe base com atributos comuns (id, nome, cpf, etc.).
* **Funcionario.java:** Herda de Pessoa e adiciona o atributo cargo.
* **Cliente.java:** Herda de Pessoa e adiciona atributos como terreno e status.

4.2 Camada de Persistência de Dados (DAO)

Responsável pela comunicação com o banco de dados SQL Server:

* **ConnectionFactory.java:** Centraliza e gerencia a criação de conexões com o banco.
* **FuncionarioDAO.java e ClienteDAO.java:** Contêm os métodos para realizar operações CRUD nas tabelas correspondentes.

4.3 Módulos de Controle e Lógica de Negócio

Concentra a lógica de visão computacional e as regras de negócio:

* **CapturaFoto.java:** Controla o hardware da câmera e a captura de imagens.
* **Treinamento.java:** Orquestra o processo de treinamento do modelo.
* **Reconhecer.java:** Implementa a lógica de autenticação facial em tempo real.

4.4 Camada de Visão (View)

Composta pelas interfaces gráficas em Java Swing:

* **Janela\_Login.java:** Ponto de entrada do sistema para autenticação.
* **Janela\_Admin.java, Frame\_Diretor.java, Frame\_Gerente.java, Frame\_funcionario.java:** Telas específicas para cada nível hierárquico, exibindo apenas as funcionalidades permitidas.

4.5 Fluxo de Operação Integrado

O fluxo de operação do sistema integra todas as camadas:

1. Um usuário se posiciona em frente à câmera na **Janela\_Login (View)**.
2. A tela aciona a classe **Reconhecer (Controller)**, que usa o modelo yml para identificar o usuário.
3. Se a identificação for bem-sucedida, o Controller utiliza o **FuncionarioDAO (DAO)** para buscar no banco de dados o perfil do usuário, que é carregado em um objeto **Funcionario (Model)**.
4. Com base no atributo cargo do modelo, o Controller determina qual tela da **View** (ex: Frame\_Gerente) deve ser instanciada e exibida, finalizando o processo de login e liberando o acesso.

5 CONCLUSÃO

O desenvolvimento deste projeto resultou em uma aplicação funcional e robusta que integra com sucesso a tecnologia de reconhecimento facial a um sistema de gestão com controle de acesso hierárquico. A escolha da linguagem Java, juntamente com a biblioteca JavaCV, provou-se adequada para os requisitos de processamento de imagem e para a construção de uma aplicação desktop multiplataforma.

O sistema de autenticação biométrica elimina a necessidade de senhas tradicionais, aumentando a segurança e a conveniência para o usuário. A arquitetura em camadas e o uso de padrões de projeto garantem que o software seja escalável e de fácil manutenção.

Como trabalhos futuros, sugere-se a migração da aplicação para uma plataforma web, a implementação de outros algoritmos de reconhecimento para fins de comparação de desempenho e a adição de um módulo de auditoria para registrar todas as ações realizadas pelos usuários no sistema.

6 IMAGENS

**Tela de cadastro**

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

Tela de Login  
Interface gráfica do usuário, Site

O conteúdo gerado por IA pode estar incorreto.

Tela de Trabalho

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Fichas de Trabalho

Tabela

O conteúdo gerado por IA pode estar incorreto.

Tabela

O conteúdo gerado por IA pode estar incorreto.

Tabela

O conteúdo gerado por IA pode estar incorreto.