Instituto Federal de Educação, Ciência e Tecnologia da Paraíba

Campus João Pessoa

Programa de Pós-Graduação em Tecnologia da Informação

Nível Mestrado Profissional

TIAGO FERREIRA DA ROCHA

INVESTIGANDO O IMPACTO DE CONFIGURAÇÕES DE ARQUITETURA EM UM SISTEMA DE RECONHECIMENTO FACIAL EM TEMPO REAL: UM CASO DE USO NO IFPB SANTA RITA

DISSERTAÇÃO DE MESTRADO

JOÃO PESSOA

2026

Tiago Ferreira da Rocha

Investigando o impacto de configurações de arquitetura em um sistema de reconhecimento facial em tempo real: um caso de uso no IFPB Santa Rita

Dissertação de Mestrado apresentada como requisito parcial para obtenção do título de Mestre em Tecnologia da Informação, pelo Programa de Pós-Graduação em Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB.

Orientador: Prof. Dr. Diego Ernesto Rosa Pessoa

Coorientador: Prof. Dr. Thiago Gouveia da Silva

João Pessoa

2026

Dados Internacionais de Catalogação na Publicação – CIP
Biblioteca Nilo Peçanha – IFPB, *Campus* João Pessoa

R672i Rocha, Tiago Ferreira da
Investigando o impacto de configurações de arquitetura em um sistema de reconhecimento facial em tempo real: um caso de uso no IFPB Santa Rita / Tiago Ferreira da Rocha. – 2026.
95 f.

Dissertação (Mestrado em Tecnologia da Informação) - Instituto Federal da Paraíba – IFPB / Programa de Pós-Graduação em Tecnologia da Informação – PPGTI.
Orientador: Prof. Dr. Diego Ernesto Rosa Pessoa
Coorientador: Prof. Dr. Thiago Gouveia da Silva
1. Aprendizado profundo. 2. Arquitetura de microsserviços. 3. Processamento de imagem. 4. Reconhecimento facial. 5. Controle de presença. I. Título.
CDU 004.932:004.273

Bibliotecária responsável Ivanise Andrade Melo de Almeida – CRB15/96

TIAGO FERREIRA DA ROCHA

INVESTIGANDO O IMPACTO DE CONFIGURAÇÕES DE ARQUITETURA EM UM SISTEMA DE RECONHECIMENTO FACIAL EM TEMPO REAL: UM CASO DE USO NO IFPB SANTA RITA

DISSERTAÇÃO submetida Programa de Pós- Graduação em Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - IFPB, Campus João Pessoa, como requisito para obtenção do título de **Mestre em Tecnologia da Informação**.

Aprovado em 13/05/2026 08:00.

Membros da Banca Examinadora:

Prof. Dr. DIEGO ERNESTO ROSA PESSOA
Instituto Federal da Paraíba (IFPB)
Orientador(a)

Prof. Dr. THIAGO GOUVEIA DA SILVA
Instituto Federal da Paraíba (IFPB)
Coorientador(a)

Prof. Dr. CLEYTON CAETANO DE SOUZA
Universidade Federal da Paraíba (IFPB)
Examinador(a)

Profa. Dra. DAMIRES YLUSKA DE SOUZA FERNANDES
Instituto Federal da Paraíba (IFPB)
Examinador(a)

Documento assinado eletronicamente por:

- **Diego Ernesto Rosa Pessoa**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 26/05/2026 12:06:37.
- **Damires Yluska de Souza Fernandes**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 26/05/2026 12:09:40.
- **Cleyton Caetano de Souza**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 26/05/2026 12:20:17.
- **Thiago Gouveia da Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 11/06/2026 20:16:18.

Este documento foi emitido pelo SUAP em 26/05/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código 883915
Verificador: 98de8e10c4
Código de Autenticação:

Este trabalho é dedicado a todos que acreditam no poder da inovação e na capacidade de superar desafios com criatividade e determinação.

"Não podemos resolver problemas usando o mesmo tipo de pensamento que usamos quando os criamos."– Albert Einstein

AGRADECIMENTOS

Este trabalho é o resultado de uma jornada de aprendizado e crescimento que não teria sido possível sem o apoio e a orientação de muitas pessoas.

Em primeiro lugar, gostaria de expressar minha profunda gratidão à minha esposa Nataly, por todo o amor, paciência e compreensão ao longo deste caminho. Seu apoio incondicional foi fundamental para que eu pudesse alcançar esta conquista.

À minha filha Clarice, que chegou enquanto estas páginas ainda estavam sendo escritas, deixo um agradecimento diferente de todos os outros. Você nasceu entre capítulos, revisões e noites de estudo, como se a vida tivesse decidido escrever, ao mesmo tempo, sua própria obra. Seu primeiro sorriso tornou os dias mais leves e fez com que cada esforço encontrasse um sentido novo. Se esta dissertação representa o fim de uma longa caminhada, sua chegada foi o mais bonito dos começos.

Agradeço de coração aos meus pais, Antônia e Luiz, que sempre acreditaram em mim e me incentivaram a seguir em frente, mesmo nos momentos mais difíceis. Sua dedicação e ensinamentos foram a base sobre a qual construí meus sonhos.

Às minhas irmãs, Thamires e Thais, e aos meus sobrinhos Luis e Matheus, deixo meu carinho e reconhecimento. Vocês estiveram ao meu lado em cada passo desta caminhada, oferecendo apoio, alegria e inspiração.

Agradeço também aos meus professores e especialmente aos meus orientadores Diego e Thiago, cujas palavras de sabedoria e conselhos foram fundamentais para a realização deste projeto. Cada discussão, crítica construtiva e orientação foi essencial para moldar este trabalho e para meu desenvolvimento como profissional e pesquisador.

Sou grato aos meus colegas de estudo e trabalho, com quem compartilhei inúmeros desafios e conquistas. A troca de ideias, as discussões enriquecedoras e o apoio mútuo tornaram esta jornada mais leve e significativa.

Por fim, agradeço a todos que, de alguma forma, contribuíram para que este trabalho fosse possível. Seja através de uma palavra de apoio, uma ideia inspiradora, ou simplesmente pela presença constante. Este trabalho é dedicado a todos vocês, que estiveram comigo ao longo deste percurso.

RESUMO

O reconhecimento facial em tempo real impõe desafios técnicos significativos. Variações de iluminação, ângulo de captura das imagens, fluxo contínuo de pessoas e restrições de latência comprometem abordagens arquiteturais simplistas. Este trabalho propõe um modelo de sistema de reconhecimento facial e descreve o desenvolvimento de cinco configurações arquiteturais incrementais que o implementam, testadas com vídeos reais captados pela câmera da portaria do Campus Santa Rita do Instituto Federal da Paraíba (IFPB). Cada versão foi concebida para superar limitações identificadas na anterior, evoluindo de uma prova de conceito ingênua até configurações distribuídas com microsserviços, cache em memória e fragmentação demográfica de dados. A pesquisa caracteriza-se como uma investigação experimental, conduzida segundo uma abordagem iterativa e incremental. A avaliação foi realizada com o processamento de 30 vídeos de aproximadamente um minuto, com métricas organizadas em três grupos: classificação, agrupamento e operacionais. A arquitetura com microsserviços e mensageria assíncrona representou uma evolução relevante, alcançando acurácia de 0,92, F1-Score de 0,95 e razão tempo/duração de 0,69. A adição de cache mostrou-se estatisticamente equivalente à arquitetura de microsserviços, preservando a qualidade de classificação e de agrupamento sem regressão, com ganho de velocidade não significativo — caracterizando uma otimização de baixo risco. A fragmentação demográfica, embora mantivesse a acurácia de classificação, degradou o agrupamento (V-Measure 0,56), evidenciando um desafio no uso de estimativas demográficas como critério de particionamento. Os resultados demonstram que decisões arquiteturais exercem impacto direto sobre a viabilidade operacional do sistema, e a sequência de versões documentada constitui um referencial prático para implementações similares.

Palavras-chaves: Aprendizado profundo; controle de presença; microsserviços; reconhecimento facial.

ABSTRACT

Real-time facial recognition poses significant technical challenges. Variations in illumination, image capture angle, continuous flow of people, and latency constraints compromise simplistic architectural approaches. This work proposes a model of a facial recognition system and describes the development of five incremental architectural configurations that implement it, tested with real footage captured by the entrance gate camera of the Santa Rita Campus of the Federal Institute of Paraíba (IFPB). Each version was designed to overcome limitations identified in the previous one, evolving from a naïve proof of concept to distributed configurations with microservices, in-memory cache, and demographic data fragmentation. The research is characterized as an experimental investigation, conducted under an iterative and incremental approach. The evaluation was carried out by processing 30 videos of approximately one minute each, with metrics organized into three groups: classification, clustering, and operational. The microservices architecture with asynchronous messaging represented a relevant advance, achieving an accuracy of 0.92, an F1-Score of 0.95, and a time-to-duration ratio of 0.69. The addition of cache proved statistically equivalent to the microservices architecture, preserving classification and clustering quality without regression, with a non-significant speed difference — characterizing a low-risk optimization. Demographic fragmentation, although it maintained classification accuracy, degraded clustering (V-Measure 0.56), evidencing a challenge in using demographic estimates as a partitioning criterion. The results demonstrate that architectural decisions have a direct impact on the operational viability of the system, and the documented sequence of versions constitutes a practical reference for similar implementations.

Keywords: Attendance control; deep learning; facial recognition; microservices.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de Programação de Aplicações (*Application Programming Interface*)
CFTV	Circuito Fechado de Televisão
CNN	Redes Neurais Convolucionais (*Convolutional Neural Networks*)
FN	Falso Negativo (*False Negative*)
FP	Falso Positivo (*False Positive*)
FPS	Quadros por Segundo (*Frames per Second*)
HOG	Histograma de Gradientes Orientados (*Histogram of Oriented Gradients*)
JWT	JSON Web Token
LBP	Padrões Binários Locais (*Local Binary Patterns*)
LDA	Análise Discriminante Linear (*Linear Discriminant Analysis*)
LFW	Labeled Faces in the Wild
LGPD	Lei Geral de Proteção de Dados
MTCNN	Multi-Task Cascaded Convolutional Network
PCA	Análise de Componentes Principais (*Principal Component Analysis*)
ReLU	Rectified Linear Unit
RNA	Redes Neurais Artificiais
RNN	Redes Neurais Recorrentes (*Recurrent Neural Networks*)
SVM	Máquina de Vetores de Suporte (*Support Vector Machine*)
TN	Verdadeiro Negativo (*True Negative*)
TP	Verdadeiro Positivo (*True Positive*)

SUMÁRIO

1 INTRODUÇÃO

O reconhecimento facial em tempo real é hoje uma tecnologia consolidada, mas sua adoção em ambientes reais depende de fatores que vão além da precisão do modelo. Este capítulo contextualiza o problema, delimita a questão de pesquisa e apresenta os objetivos e a estrutura do trabalho.

1.1 Motivação e Definição do Problema

O Instituto Federal da Paraíba (IFPB), Campus Santa Rita, que oferta cursos técnicos e superiores, lida cotidianamente com o desafio de monitorar o fluxo de pessoas em suas dependências. Atualmente, esse monitoramento apoia-se em registro manual de presença e vigilância humana na portaria, abordagem suscetível a falhas, ineficiências e fraudes, além de impor carga operacional significativa sobre docentes e corpo administrativo. Como o campus já dispõe de um sistema de Circuito Fechado de Televisão (CFTV) instalado na portaria, a automação desse processo por meio de reconhecimento facial sobre o vídeo já capturado representa uma alternativa capaz de atender simultaneamente a múltiplas demandas institucionais: controle de acesso, segurança patrimonial, registro de frequência de alunos e monitoramento do fluxo de pessoas nas dependências do campus. O registro em tempo real abre ainda a possibilidade de informar os responsáveis sobre a presença dos alunos, fortalecendo o acompanhamento acadêmico.

Entretanto, o reconhecimento facial em tempo real impõe desafios que vão além da qualidade do modelo de reconhecimento. O processamento contínuo de vídeo pelo sistema de Circuito Fechado de Televisão (CFTV) exige arquiteturas de software capazes de equilibrar velocidade e acurácia sob restrições de latência. Em ambientes com múltiplas fontes de captura, o volume de dados cresce rapidamente, exigindo soluções escaláveis que mantenham baixa latência e alta disponibilidade. O avanço do aprendizado profundo e das redes neurais convolucionais elevou a acurácia dos modelos a níveis próximos ao humano; ainda assim, a viabilidade operacional de sistemas em produção depende fundamentalmente das decisões arquiteturais que sustentam o pipeline de processamento — e não apenas da escolha do modelo.

Apesar dessa constatação, a literatura sobre reconhecimento facial concentra-se majoritariamente na comparação de modelos, tratando a arquitetura de software que sustenta o pipeline como uma premissa fixa — escolhida de forma pragmática e raramente avaliada como objeto de estudo. Há, portanto, uma lacuna quanto à compreensão sistemática de como diferentes decisões arquiteturais impactam a viabilidade operacional desses sistemas em condições reais de operação. Diante disto, este trabalho é orientado pela seguinte questão de pesquisa: qual o impacto de diferentes configurações de arquitetura de software sobre a viabilidade operacional de um sistema de reconhecimento facial em tempo real no IFPB Campus Santa Rita?

1.2 Objetivos

O objetivo geral deste trabalho é propor, implementar e avaliar comparativamente um conjunto de configurações arquiteturais para um sistema de reconhecimento facial em tempo real voltado ao controle de presença no IFPB Campus Santa Rita, mensurando o impacto de cada decisão arquitetural sobre a viabilidade operacional do sistema — avaliada por métricas de classificação, agrupamento e custo operacional.

Como objetivos específicos, definem-se: projetar um modelo básico de sistema de reconhecimento facial que sirva como referência conceitual para as configurações avaliadas; implementar cinco configurações incrementais, em que cada uma introduz uma decisão arquitetural que busca superar uma limitação da anterior: da Ingênua (prova de conceito sem otimizações) à Monolítica (consolidação do pipeline em um único serviço), à de Microsserviços (desacoplamento com mensageria assíncrona), à de Cache (priorização por aparição para reduzir reprocessamento) e à de Fragmentação (particionamento dos dados por estimativa demográfica); construir um *dataset* próprio a partir de vídeos reais captados pelo sistema CFTV da portaria do IFPB Campus Santa Rita, representando as condições reais de operação do sistema; avaliar e comparar estatisticamente as configurações com métricas organizadas em três grupos — classificação, agrupamento e operacionais —, a fim de verificar as hipóteses de pesquisa; e sistematizar os impactos das decisões arquiteturais em uma tabela comparativa — contemplando desempenho de classificação, qualidade de agrupamento, custo operacional e implicações éticas — que sirva de referencial prático para projetos similares de reconhecimento facial.

1.3 Contribuições

As principais contribuições deste trabalho são: (i) um modelo de sistema de reconhecimento facial em tempo real e cinco configurações arquiteturais incrementais que o implementam, documentadas de modo que a decisão de projeto por trás de cada uma fique explícita; (ii) um *dataset* próprio, construído a partir de vídeos reais do CFTV da portaria do IFPB Campus Santa Rita, representativo das condições operacionais do sistema; (iii) uma avaliação experimental comparativa das configurações, com verificação estatística de hipóteses sobre métricas de classificação, agrupamento e custo operacional; e (iv) a sistematização dos impactos das decisões arquiteturais em um referencial prático para projetos similares de reconhecimento facial.

1.4 Estrutura do Documento

Os capítulos subsequentes estão organizados da seguinte maneira: O **Capítulo 2** apresenta a fundamentação teórica que embasa o trabalho. O **Capítulo 3** descreve a metodologia, incluindo o desenho experimental, as configurações arquiteturais avaliadas e o método de análise. O **Capítulo 4** apresenta os resultados obtidos nos experimentos. O **Capítulo 5** interpreta esses

resultados e verifica as hipóteses de pesquisa. Por fim, o **Capítulo 6** apresenta as considerações finais e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos necessários para fundamentar a investigação experimental conduzida neste trabalho sobre o impacto de diferentes configurações arquiteturais. São apresentadas as Redes Neurais Artificiais (RNA), com destaque para as Redes Neurais Convolucionais (CNNs) e sua relevância para o reconhecimento facial. Em seguida, detalham-se as etapas de detecção e reconhecimento de faces, descrevendo os principais métodos empregados na literatura e as escolhas adotadas neste trabalho. Na sequência, são apresentadas as arquiteturas de software para sistemas em tempo real, com ênfase nas abordagens monolítica e de microsserviços que fundamentam as decisões arquiteturais avaliadas neste trabalho. As métricas de avaliação utilizadas nos experimentos são então definidas e organizadas em três grupos: classificação, agrupamento e operacionais. Por fim, são apresentados os trabalhos relacionados.

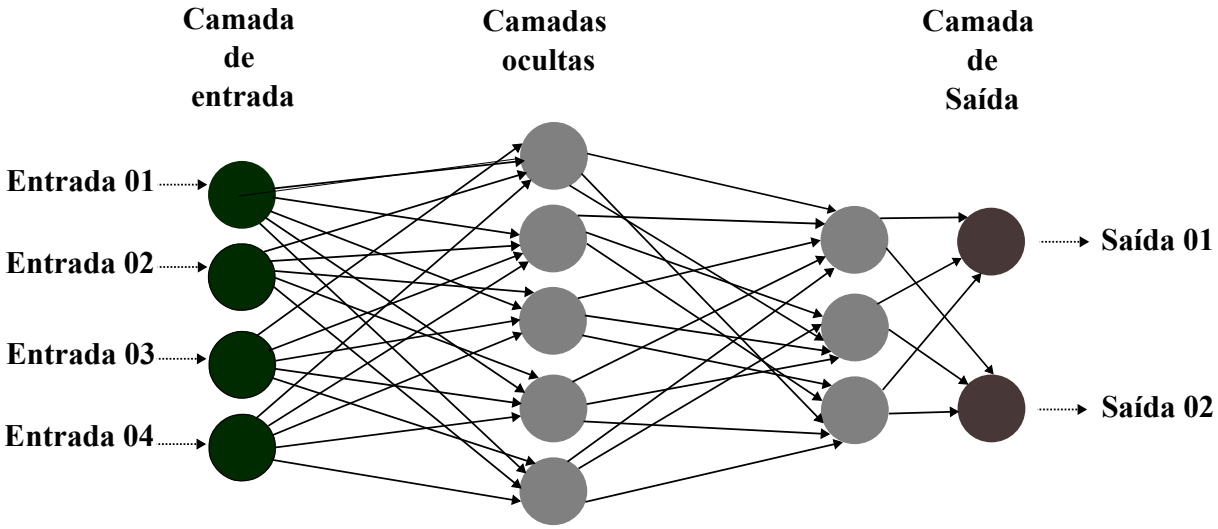
2.1 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados no funcionamento do cérebro humano, capazes de aprender padrões complexos a partir de dados. Sua relevância para este trabalho reside na capacidade de extrair representações faciais robustas a partir de imagens, viabilizando o reconhecimento de indivíduos em condições variadas de iluminação, pose e distância de captura.

2.1.1 Funcionamento das Redes Neurais Artificiais

As RNAs são compostas de unidades básicas chamadas neurônios artificiais, que são inspiradas nos neurônios biológicos. Cada neurônio recebe um conjunto de entradas, as processa e produz uma saída. Estes neurônios são organizados em camadas. Existem camadas de entrada que recebem dados brutos, camadas ocultas que processam esses dados, e camadas de saída que produzem o resultado final. As RNAs podem ter várias camadas ocultas, o que as torna capazes de aprender representações complexas dos dados (MIENYE; SWART, 2024). A Figura 1 exibe um exemplo de uma arquitetura de múltiplas camadas.

Figura 1 – Exemplo de esquema de arquitetura de múltiplas camadas



Fonte: adaptado de (FREITAS; SANTOS; REZENDE, 2022)

Os neurônios em redes neurais utilizam funções de ativação para introduzir não-linearidades no processo de aprendizado, o que permite que a rede aprenda padrões mais complexos. Exemplos comuns de funções de ativação incluem a sigmóide, a tangente hiperbólica, complemento log-log e probit (MIENYE; SWART, 2024).

O processo de aprendizado em RNAs é realizado através do ajuste dos pesos sinápticos (conexões entre neurônios). Isso é feito utilizando um algoritmo chamado retropropagação, que propaga o erro da saída de volta pela rede para ajustar os pesos, em combinação com um método de otimização, como o gradiente descendente (MIENYE; SWART, 2024).

Durante o treinamento, a rede é exposta a um grande número de exemplos, ajustando seus pesos para reduzir a diferença entre a saída prevista e a real. Técnicas de regularização e o treinamento com parada antecipada são usadas para evitar o *overfitting*, que ocorre quando a rede se torna muito boa em prever os dados de treinamento, mas falha em generalizar para novos dados. Desta forma é possível melhorar a capacidade de generalização da RNA (MIENYE; SWART, 2024).

2.1.2 Arquitetura das RNA

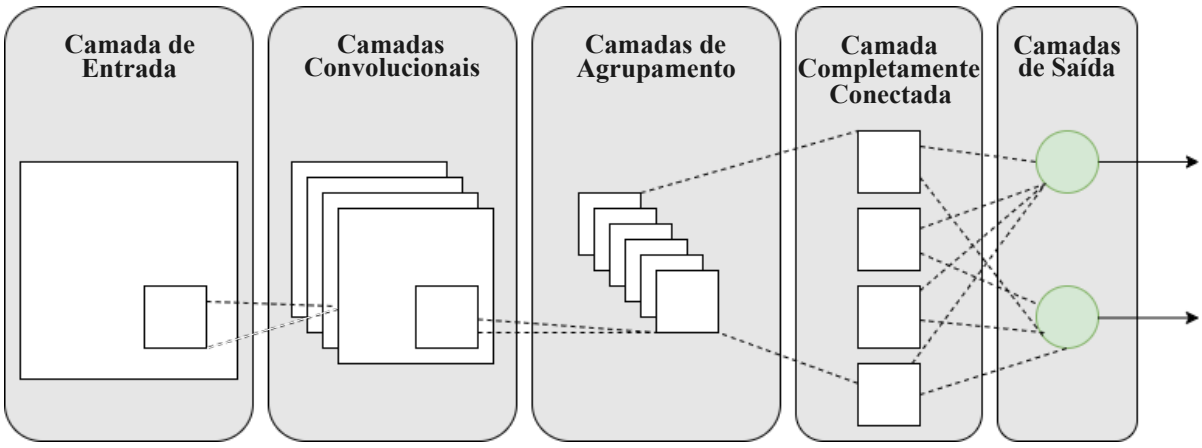
As RNAs podem assumir diferentes arquiteturas conforme o tipo de problema a ser resolvido. As redes *feedforward* organizam os neurônios em camadas onde a informação flui exclusivamente da entrada para a saída, sendo adequadas para classificação de padrões e aproximação de funções. As redes recorrentes introduzem conexões de retroalimentação entre neurônios, tornando-as apropriadas para o processamento de sequências temporais. Já as Redes Neurais Convolucionais (CNNs) são especialmente projetadas para o processamento de dados com estrutura espacial, como imagens, e constituem a arquitetura central para o reconhecimento facial abordado neste trabalho (MIENYE; SWART, 2024).

2.1.2.1 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (CNNs) representam um avanço significativo no campo das redes neurais artificiais, especialmente em tarefas relacionadas ao processamento de imagens. São uma espécie de rede neural artificial inspiradas na organização do córtex visual dos animais e córtex de seres humanos. Este tipo de rede é particularmente eficaz em identificar detalhes em camadas, começando com padrões simples nas primeiras camadas e progredindo para padrões mais complexos nas camadas posteriores, tornando-as ideais para aplicações de visão computacional (LIU; ZHOU; CHEN, 2021).

Uma CNN típica é composta por várias camadas, incluindo camadas convolucionais, camadas de *pooling* (camadas de agrupamento), camadas totalmente conectadas. As camadas convolucionais são responsáveis por extrair características locais das imagens aplicando filtros que geram mapas de características. As camadas de *pooling* reduzem a dimensionalidade dos mapas de características ao combinar informações de regiões adjacentes, enquanto as camadas totalmente conectadas integram essas características para formar um vetor de alta dimensionalidade. A Figura 2 ilustra uma típica arquitetura de rede convolucional com essas camadas.

Figura 2 – Uma típica arquitetura de rede convolucional



Fonte: adaptado de (BARBOSA et al., 2021)

O funcionamento das CNNs baseia-se na aplicação de convoluções, que são operações matemáticas que combinam dois conjuntos de informações. Na prática, cada neurônio em uma camada convolucional é conectado apenas a uma pequena região da camada anterior, chamada de campo receptivo. Este processo permite que a CNN extraia características locais específicas, como bordas e texturas. As funções de ativação, como ReLU (Rectified Linear Unit), são aplicadas para introduzir não-linearidades, permitindo que a rede aprenda representações mais complexas (LIU; ZHOU; CHEN, 2021).

O treinamento das CNNs envolve a otimização dos pesos dos filtros através de um processo iterativo conhecido como retropropagação do erro. Durante o treinamento, a rede é exposta a um grande número de exemplos de treinamento, e os pesos são ajustados para minimizar

a diferença entre as previsões da rede e os valores reais. Algoritmos de otimização, como o gradiente descendente, são utilizados para este ajuste. Técnicas de regularização, como *dropout* — que desativa aleatoriamente uma fração dos neurônios durante o treinamento, forçando a rede a aprender representações mais robustas e independentes — e normalização por lote, são frequentemente empregadas para evitar *overfitting* e melhorar a generalização da rede (ALMABDY; ELREFAEI, 2019).

Embora as CNNs tenham revolucionado o campo do reconhecimento de padrões, ainda existem desafios significativos, como a necessidade de grandes quantidades de dados rotulados para treinamento e o alto custo computacional. Avanços recentes incluem o desenvolvimento de arquiteturas mais eficientes, como redes convolucionais leves que utilizam menos recursos computacionais, e a incorporação de técnicas de aprendizado por transferência para aproveitar modelos pré-treinados em novas tarefas com menos dados (LIU; ZHOU; CHEN, 2021).

2.1.2.2 Aplicação de Redes Neurais Convolucionais no Reconhecimento Facial

As CNNs são particularmente eficazes em reconhecimento facial devido à sua capacidade de aprender representações invariantes às transformações comuns em imagens faciais, como variações de pose e iluminação. (ALMABDY; ELREFAEI, 2019) investigaram o uso de arquiteturas CNN pré-treinadas — especificamente AlexNet e ResNet-50 — combinadas com *Support Vector Machines* (SVM) para classificação, demonstrando taxas de acurácia entre 91,03% e 99,01% em conjuntos de dados como LFW (HUANG et al., 2007) e YouTube Faces (WOLF; HASSNER; MAOZ, 2011). Os resultados evidenciam que a combinação de arquiteturas CNN com técnicas de aprendizado por transferência representa uma abordagem eficaz para o reconhecimento facial em condições variadas (ALMABDY; ELREFAEI, 2019).

Devido à alta demanda de processamento, os sistemas de reconhecimento facial necessitam de diversas otimizações para melhorar sua eficiência. Essas melhorias podem abranger desde o pré-processamento de imagens até a implementação de técnicas avançadas de aprendizado de máquina. Conforme (APPATI et al., 2021), otimizar o reconhecimento facial pode incluir técnicas de alinhamento de imagens e redução de dimensionalidade, que contribuem para reduzir a carga computacional e aumentar a precisão dos sistemas, demonstrando ainda como arquiteturas especializadas lidam de forma mais eficaz com variáveis complexas, como variações de iluminação e expressões faciais. Esses estudos reforçam a importância de inovações contínuas para superar os desafios de desempenho e escalabilidade.

2.2 Detecção de Faces

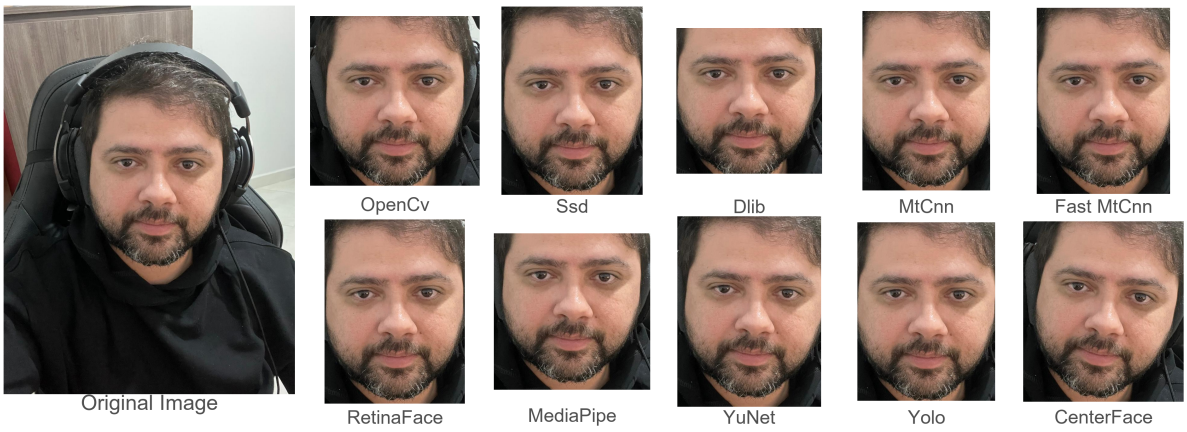
A detecção de faces constitui a etapa inicial de qualquer sistema de reconhecimento facial, sendo responsável por identificar e localizar as regiões de interesse em uma imagem ou sequência de vídeo onde há presença humana. Essa fase é determinante para o desempenho

global do sistema, pois erros de detecção tendem a se propagar nas etapas seguintes de extração de características e reconhecimento (KAUR et al., 2020).

Os primeiros métodos amplamente utilizados foram baseados em técnicas clássicas de visão computacional, como o algoritmo Viola-Jones (VIOLA; JONES, 2001), que introduziu o uso de características Haar e classificadores em cascata, permitindo a detecção de rostos em tempo real. Posteriormente, métodos baseados em Histogram of Oriented Gradients (HOG) e Support Vector Machines (SVM) melhoraram a precisão em ambientes controlados, mas apresentaram limitações em cenários com variações de iluminação e posição.

A Figura 3 apresenta uma comparação visual entre diferentes detectores faciais amplamente utilizados em frameworks modernos. Observa-se que, embora todos identifiquem corretamente a face, há variações na precisão do recorte e no alinhamento, especialmente em métodos clássicos como OpenCV (BRADSKI, 2000) e Dlib (KING, 2009), quando comparados a modelos mais recentes, como RetinaFace (DENG et al., 2020) e MediaPipe (LUGARESI et al., 2019).

Figura 3 – Comparação entre detectores faciais utilizados em frameworks modernos



Fonte: Elaboração própria, imagem gerada pelo Face Detector, disponível em: <https://github.com/tiagorocha1/detector-face>

Com o avanço do aprendizado profundo, surgiram modelos baseados em Redes Neurais Convolucionais (CNNs), capazes de aprender representações mais robustas e invariantes a essas condições. (DEWI; MARYADI, 2025) destacam entre os métodos mais relevantes o Multi-Task Cascaded Convolutional Network (MTCNN), que combina detecção e alinhamento facial em um único pipeline, e o RetinaFace, que prevê simultaneamente as *bounding boxes* e os pontos de referência faciais, alcançando alta precisão mesmo em condições variadas de escala e pose.

Essas abordagens baseadas em CNNs tornaram possível realizar a detecção facial de forma eficiente mesmo em fluxos contínuos de vídeo, com alta taxa de quadros e variações significativas de iluminação, o que as torna essenciais para aplicações em tempo real como as propostas neste trabalho. No sistema desenvolvido, foi adotado o MediaPipe (LUGARESI et

al., 2019) como detector facial por sua leveza e capacidade de operação em tempo real, sendo compatível com o processamento contínuo de vídeo. Especificamente, foi utilizado o modelo *BlazeFace Full-range* (Google LLC, 2023), que oferece maior alcance de detecção e robustez a variações de pose e escala em relação ao modelo *Short-range* — características essenciais para uma câmera CFTV posicionada a distância. Por não ser compatível com o ambiente do navegador, o *BlazeFace Full-range* é executado exclusivamente no backend, o que motivou a migração da detecção para o servidor a partir da Arquitetura Monolítica — decisão discutida em detalhe no Capítulo 3.

2.2.1 Pré-processamento

O pré-processamento é uma etapa fundamental no reconhecimento facial, pois prepara as imagens antes de serem analisadas pelos algoritmos de reconhecimento. As técnicas de pré-processamento ajudam a reduzir as variações nas imagens e melhorar a precisão do reconhecimento.

A normalização de iluminação trata das variações de luz, situação em que a imagem de um rosto pode aparecer diferente devido à mudança nas condições de iluminação. Essa mudança pode ser maior que as diferenças entre indivíduos, fazendo com que sistemas de reconhecimento facial falhem ao comparar imagens. A iluminação afeta a localização, a forma das sombras e a reversão de gradientes de contraste. Humanos reconhecem rostos apesar das mudanças na iluminação, mas isso é um desafio para sistemas automatizados. Diversos algoritmos foram propostos para normalizar imagens de rosto em diferentes condições de iluminação, incluindo transformações de intensidade gamma, modelos faciais 3D e uso de histogramas (GUO et al., 2016).

O alinhamento facial é uma abordagem de pré-processamento que envolve a localização de pontos fiduciais, como boca, olhos, queixo e nariz. Essa abordagem melhora o sistema de reconhecimento facial, embora ainda seja um desafio em ambientes não controlados. A melhoria da imagem é outra abordagem de pré-processamento frequentemente negligenciada na literatura, cujo objetivo é obter uma imagem facial aprimorada a partir da original, melhorando o desempenho do sistema de reconhecimento facial (GUO et al., 2016).

A remoção de ruído é crucial para melhorar a precisão dos sistemas de reconhecimento facial, especialmente em ambientes não controlados. Técnicas avançadas utilizam redes neurais convolucionais (CNN) para eliminar vários tipos de ruído, como sal e pimenta e ruído gaussiano. A combinação de módulos de estimativa e remoção de ruído ajuda a produzir imagens mais limpas. Esses processos não apenas preservam detalhes importantes das imagens faciais, mas também melhoram significativamente o desempenho dos sistemas de reconhecimento facial (ALI et al., 2021a).

Essas técnicas de pré-processamento são essenciais para garantir que as imagens de entrada estejam em um formato consistente e de alta qualidade, facilitando a tarefa de reconheci-

mento facial pelos modelos de aprendizado de máquina. No contexto de sistemas em tempo real, como o proposto neste trabalho, a escolha das técnicas de pré-processamento deve equilibrar qualidade e custo computacional, uma vez que cada etapa adicional impacta diretamente a latência do pipeline.

2.3 Reconhecimento de Faces

Os sistemas de reconhecimento facial evoluíram por duas direções principais. De um lado, métodos clássicos foram combinados em abordagens híbridas com o objetivo de superar limitações como variações de iluminação, oclusão e envelhecimento (KORTLI et al., 2020). De outro, o ressurgimento do aprendizado profundo a partir dos anos 2010 transformou o campo ao substituir a extração manual de características por representações aprendidas diretamente dos dados, elevando o desempenho dos sistemas a níveis próximos ao humano. As subseções a seguir detalham cada uma dessas abordagens.

2.3.1 Métodos híbridos de reconhecimento

Os métodos híbridos combinam abordagens holísticas, que identificam o rosto como um todo através de características globais, com técnicas de extração de características específicas, como boca, nariz e olhos (ALI et al., 2021b) . Entre os algoritmos populares estão: Eigenface (PCA), que utiliza PCA para decompor imagens em subimagens chamadas de eigenfaces, focando nas características essenciais e reduzindo as dimensões da imagem original (RESENDE; PEREIRA, 2015) ; Fisherface (LDA), que utiliza a técnica de Análise Discriminante Linear para agrupar imagens em classes após a rotulação das faces (COSTA, 2019) ; e LBP, que compara cada pixel com seus vizinhos, resistindo a variações de iluminação, escala, posição e ruído, além de ser computacionalmente menos complexo (FERREIRA; PAGLIARI, 2018). O método Viola-Jones, usado na detecção de faces, é eficiente por sua rapidez e flexibilidade, consistindo de quatro estágios: criação de uma imagem integral, extração de características Haar, classificador em cascata e treinamento AdaBoost (ELIAS et al., 2019) (YAAKUB et al., 2021) .

(CHAUDHRY; ELGAZZAR, 2019) também compararam a performance individual dos algoritmos Eigenface, Fisherface e LBP, além de suas combinações (classificador EFL). Eles verificaram que o classificador híbrido aumentou a precisão, mas também aumentou o tempo de processamento. Eles sugerem que otimizações podem melhorar a performance desse algoritmo híbrido. (ZHANG et al., 2019) observaram que o pré-processamento das imagens, como a remoção de variações de iluminação, melhora a eficiência dos algoritmos de reconhecimento facial.

Outros desafios no reconhecimento facial incluem variações de iluminação, posição, oclusão (obstáculos que ocultam características faciais), expressões faciais e envelhecimento (ABUZAR; AHMAD; AHMAD, 2020) (ALI et al., 2021b) (KAUR et al., 2020) (ADJABI et al.,

2020). Para mitigar esses problemas, técnicas de reconhecimento facial 3D têm sido exploradas. (MARTINO et al., 2020) propuseram uma abordagem modular e eficaz utilizando câmeras de alta definição de smartphones modernos para aprimorar essas técnicas.

2.3.2 Reconhecimento Facial com Aprendizado Profundo

Nas décadas seguintes, o uso do aprendizado profundo transformou completamente o reconhecimento facial que passou a empregar redes neurais convolucionais (CNNs), elevando o desempenho para níveis próximos ao humano (KAUR et al., 2020). Estudos mostram que métodos híbridos, que combinam abordagens globais e locais, podem alcançar melhores resultados, embora exijam mais processamento (ALI et al., 2021b; CHAUDHRY; ELGAZZAR, 2019).

Mesmo com esses avanços, o reconhecimento facial ainda enfrenta dificuldades em condições reais — variações de luz, ângulo, oclusões, expressões e envelhecimento continuam impactando a precisão dos modelos (KORTLI et al., 2020). Para reduzir esses efeitos, novas abordagens, como o reconhecimento facial 3D, têm sido exploradas (MARTINO et al., 2020). De forma geral, o processo segue quatro etapas principais: captura, detecção, extração de características e identificação, que estruturam o pipeline básico dos sistemas de reconhecimento facial.

Nos sistemas atuais, esse pipeline evoluiu para modelos capazes de aprender automaticamente cada etapa do processo a partir dos dados, reduzindo a dependência de intervenções manuais (ALI et al., 2021b). O reconhecimento facial atual é amplamente dominado por modelos de aprendizado profundo, que substituíram as abordagens clássicas ao extrair representações mais precisas e robustas das faces. Esses sistemas utilizam redes neurais convolucionais (CNNs) e funções de perda baseadas em margem, como as empregadas no FaceNet e ArcFace, para gerar embeddings capazes de distinguir indivíduos com alta precisão (SERENGIL; ÖZPINAR, 2024). O avanço dos modelos de deep learning, especialmente redes neurais convolucionais, tem contribuído significativamente para a melhoria da precisão e eficiência dos sistemas de reconhecimento facial, impulsionado também pela disponibilidade de grandes volumes de dados (FADEL, 2025).

(SERENGIL; ÖZPINAR, 2024) conduziram um estudo de benchmark abrangente utilizando a biblioteca DeepFace — uma biblioteca Python de código aberto que unifica diferentes modelos e detectores de reconhecimento facial em uma interface comum. O estudo avaliou múltiplas combinações de modelos de reconhecimento facial, detectores faciais e métricas de distância sobre o dataset LFW (*Labeled Faces in the Wild*) (HUANG et al., 2007), com o objetivo de quantificar a contribuição individual de cada componente do pipeline sobre a precisão final do sistema.

Os resultados demonstraram que tanto o detector quanto o modelo de reconhecimento exercem influência significativa sobre a precisão final. Entre os modelos avaliados, FaceNet-512 e ArcFace — baseados em CNNs profundas que geram embeddings faciais comparados

por métricas de distância — atingiram desempenho superior ao nível humano, com destaque para a combinação FaceNet-512/RetinaFace, que alcançou 98,4% de acurácia (SERENGIL; ÖZPINAR, 2024). Abordagens mais leves — modelos com menos camadas e menor número de parâmetros, que exigem menos memória e poder de processamento —, como *OpenFace* e *DeepID* — modelos com arquiteturas menos profundas e menor custo computacional —, embora apresentem acurácia inferior, mantêm relevância em cenários onde eficiência computacional é prioritária (SERENGIL; ÖZPINAR, 2024).

A consolidação dessas bibliotecas e *benchmarks* ilustra a maturidade alcançada pelos sistemas de reconhecimento facial baseados em aprendizado profundo. No entanto, a aplicação prática desses modelos em tempo real exige arquiteturas computacionais escaláveis, capazes de processar grandes volumes de dados provenientes de múltiplas fontes de vídeo — tema discutido na próxima seção.

2.4 Arquiteturas de Software

A aplicação prática de modelos de reconhecimento facial não depende apenas da precisão do modelo utilizado, mas também da arquitetura de software que sustenta o pipeline de processamento. Em cenários com fluxo contínuo de vídeo e múltiplas fontes de captura, decisões arquiteturais como o modelo de processamento adotado, a estratégia de comunicação entre serviços e o mecanismo de armazenamento exercem impacto direto sobre a viabilidade operacional do sistema.

2.4.1 Arquitetura Monolítica

A arquitetura monolítica é caracterizada pela concentração de todas as funcionalidades do sistema em um único processo ou unidade de implantação (NEWMAN, 2015). Nesse modelo, os módulos de captura, processamento, reconhecimento e armazenamento compartilham o mesmo ambiente de execução e são acionados de forma sequencial e síncrona. Embora essa abordagem simplifique o desenvolvimento inicial e a depuração, ela apresenta limitações significativas em cenários de alta carga: qualquer gargalo em uma etapa do pipeline bloqueia todas as demais, e a escalabilidade do sistema como um todo fica condicionada à capacidade do componente mais lento (NEWMAN, 2015).

2.4.2 Arquitetura de Microsserviços

A arquitetura de microsserviços estrutura o sistema como um conjunto de serviços pequenos e independentes, cada um responsável por uma função específica e comunicando-se com os demais por meio de interfaces bem definidas (NEWMAN, 2015). Segundo Newman (2015), os principais benefícios dessa abordagem são o baixo acoplamento entre os componentes, a possibilidade de escalar cada serviço de forma independente conforme a demanda e a maior

tolerância a falhas — já que a indisponibilidade de um serviço não necessariamente compromete os demais.

Em sistemas de reconhecimento facial, essa separação é especialmente relevante: as etapas de captura de frames, detecção de faces, extração de embeddings e armazenamento de registros possuem características computacionais distintas e podem se beneficiar de estratégias de escalabilidade independentes. A comunicação assíncrona entre os serviços, implementada por meio de filas de mensagens, permite que cada componente opere em seu próprio ritmo, possibilitando absorver picos de carga sem ocasionar degradação (NEWMAN, 2015).

2.4.3 Comunicação Assíncrona e Mensageria

A forma como os serviços se comunicam é uma decisão arquitetural central em sistemas distribuídos. Na comunicação síncrona, o serviço que invoca outro permanece bloqueado à espera da resposta, de modo que a lentidão ou a indisponibilidade de um componente se propaga aos demais. Já na comunicação assíncrona, o serviço emissor não aguarda resposta imediata: entrega a tarefa e prossegue sua execução, enquanto o receptor a processa em seu próprio ritmo (NEWMAN, 2015).

A mensageria assíncrona materializa esse modelo por meio de filas de mensagens intermediadas por um *message broker*: serviços produtores publicam mensagens na fila e serviços consumidores as retiram e processam de forma independente, sem que ambos precisem estar ativos simultaneamente. Esse desacoplamento temporal traz benefícios diretos para sistemas sob carga variável — a fila absorve picos de demanda, evitando a sobrecarga do emissor, e a indisponibilidade temporária de um consumidor não implica perda de tarefas, que permanecem na fila até poderem ser processadas (NEWMAN, 2015).

2.5 Métricas

A avaliação do sistema proposto necessita de um conjunto de métricas capazes de capturar tanto a qualidade das detecções individuais quanto a coerência dos agrupamentos produzidos. Conforme (SUNDARAM; MANI, 2016), precisão, recall, acurácia e F1-Score formam um conjunto complementar de indicadores para a avaliação de sistemas de reconhecimento facial, cada um capturando uma dimensão distinta do comportamento do sistema, sendo que erros de diferentes naturezas — como falsos positivos e falsos negativos — podem ter consequências práticas distintas dependendo da aplicação. As métricas adotadas neste trabalho foram organizadas em três grupos: métricas de classificação, métricas de agrupamento e métricas operacionais.

2.5.1 Métricas de Classificação

As métricas de classificação partem da matriz de confusão, construída a partir da comparação entre as saídas do sistema e um conjunto de dados de referência, denominado *ground truth*.

Neste trabalho, o *ground truth* é composto pelas melhores faces extraídas manualmente de cada vídeo do dataset. A partir dessa referência, quatro situações são definidas:

- **True Positive (TP):** o sistema reconheceu corretamente uma face como pertencente a um indivíduo cadastrado no *ground truth*;
- **True Negative (TN):** o sistema rejeitou corretamente uma face por ela não corresponder a nenhum indivíduo cadastrado no *ground truth*;
- **False Positive (FP):** o sistema reconheceu incorretamente uma face como pertencente a um indivíduo cadastrado, quando na verdade ela não pertence ao *ground truth*;
- **False Negative (FN):** o sistema rejeitou incorretamente uma face que, na verdade, pertencia a um indivíduo cadastrado no *ground truth*.

Essas quatro categorias fundamentam os indicadores derivados. A **acurácia** (*accuracy*) representa a proporção de predições corretas — tanto positivas quanto negativas — em relação ao total de casos avaliados, conforme a Equação 1:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

A **precisão** (*precision*) calcula o percentual de acertos dentre todas as predições positivas realizadas pelo sistema, indicando sua capacidade de evitar falsos alarmes (Equação 2):

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

O **recall** representa a capacidade do sistema de identificar corretamente todas as ocorrências positivas reais, penalizando fortemente os falsos negativos (Equação 3):

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

O **F1-Score** é a média harmônica entre precisão e recall, equilibrando acurácia e sensibilidade em uma única medida. (SUNDARAM; MANI, 2016) destacam que, em sistemas de reconhecimento facial, o F1-Score é particularmente valioso quando os dados são desbalanceados, pois considera simultaneamente os falsos positivos e os falsos negativos. Sua fórmula é apresentada na Equação 4:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{4}$$

2.5.2 Métricas de Agrupamento

Além do reconhecimento individual, o sistema realiza o agrupamento (*clustering*) das faces detectadas ao longo dos vídeos, associando múltiplas ocorrências de uma mesma pessoa em um único grupo. A avaliação desse processo requer métricas específicas que analisam tanto a estrutura interna dos grupos quanto a sua coerência em relação às identidades reais.

A **distância intra-cluster** (*intra-clustering distance*) mensura a distância média entre os pontos pertencentes a um mesmo grupo. Valores menores indicam maior coesão interna, ou seja, faces de uma mesma pessoa estão agrupadas de forma compacta. A **distância inter-cluster** (*inter-clustering distance*), por sua vez, mede a distância média entre os centros dos diferentes grupos. Quanto maior esse valor, maior a separação entre identidades distintas, o que indica uma melhor discriminação pelo sistema. (ROUSSEEUW, 1987) define o índice **Silhouette** como um indicador que combina ambas as dimensões — coesão e separação — avaliando a qualidade geral do agrupamento. Seu valor varia de -1 a 1: valores próximos a 1 indicam clusters bem definidos e bem separados; valores próximos a 0 indicam sobreposição entre grupos; e valores negativos indicam atribuições incorretas.

O cenário avaliado neste trabalho corresponde ao modelo de *open universe* descrito por (PHILLIPS et al., 2000), no qual não há garantia de que toda identidade de interesse esteja presente no conjunto de consultas nem que todas as identidades presentes na cena sejam corretamente capturadas pelo sistema. A partir dessa premissa, definimos operacionalmente a métrica *Covering* como a razão entre o número de identidades distintas do *ground truth* efetivamente capturadas pelo sistema — isto é, que aparecem corretamente rotuladas ao menos uma vez nas presenças registradas — e o número total de identidades esperadas no vídeo, refletindo a capacidade do sistema de cobrir as identidades verdadeiras presentes no vídeo.

A **Homogeneidade** (*Homogeneity*) avalia se cada cluster contém amostras de uma única classe, ou seja, se um grupo reúne somente faces de uma mesma pessoa. A **Completude** (*Completeness*) avalia o aspecto complementar: se todas as faces de uma mesma pessoa foram reunidas em um único cluster. Rosenberg e Hirschberg (2007) definem a **V-Measure** como a média harmônica entre homogeneidade e completude, integrando esses dois critérios em um único indicador de consistência do agrupamento, conforme a Equação 5:

$$\text{V-Measure} = 2 \times \frac{\text{Homogeneity} \times \text{Completeness}}{\text{Homogeneity} + \text{Completeness}} \tag{5}$$

Todos os indicadores — Homogeneidade, Completude e V-Measure — assumem valores entre 0 e 1, onde 1 representa o resultado perfeito, facilitando a comparação entre diferentes configurações do sistema (ROSENBERG; HIRSCHBERG, 2007).

2.5.3 Métricas Operacionais

Além da qualidade das detecções e dos agrupamentos, a viabilidade prática do sistema depende de indicadores que reflitam sua eficiência computacional e o custo de armazenamento associado.

A razão **tempo de processamento / duração total do vídeo** (*time to complete / video total time*) representa a eficiência temporal do sistema: valores inferiores a 1 indicam que o processamento ocorre mais rápido do que o vídeo transcorre em tempo real, o que é condição necessária para a operação em tempo real. Essa relação é diretamente afetada pela resolução das imagens, pelo hardware utilizado e pelo modelo de detecção empregado (PEREZ-MONTES et al., 2023), o que reforça a importância de modelar explicitamente essa tensão entre acurácia e velocidade no contexto das arquiteturas avaliadas.

Por fim, o **tamanho do banco de dados auxiliar** (*auxiliary DB size*) quantifica o volume total de dados armazenados pelo sistema ao longo do processamento — incluindo embeddings, metadados e resultados intermediários — servindo como indicador do custo de armazenamento associado a cada arquitetura avaliada. A Tabela 1 apresenta um resumo das métricas adotadas.

Tabela 1 – Resumo das métricas de avaliação adotadas

Métrica	**Grupo**	**Intervalo**
Accuracy	Classificação	$[0, 1]$
Precision	Classificação	$[0, 1]$
Recall	Classificação	$[0, 1]$
F1-Score	Classificação	$[0, 1]$
Covering	Agrupamento	$[0, 1]$
Intra-Clustering Distance	Agrupamento	$[0, +\infty)$
Inter-Clustering Distance	Agrupamento	$[0, +\infty)$
Silhouette	Agrupamento	$[-1, 1]$
Homogeneity	Agrupamento	$[0, 1]$
Completeness	Agrupamento	$[0, 1]$
V-Measure	Agrupamento	$[0, 1]$
Time / Video Duration	Operacional	$[0, +\infty)$
Auxiliary DB Size	Operacional	$[0, +\infty)$

Fonte: Elaboração própria.

2.6 Análise Estatística de Experimentos

A comparação entre diferentes configurações de um sistema — tratadas como tratamentos de um experimento — não pode apoiar-se apenas na inspeção das médias das métricas, pois diferenças numéricas podem decorrer de variação aleatória entre as unidades avaliadas. Por isso, recomenda-se o uso de testes de hipóteses para decidir se as diferenças observadas são estatisticamente significativas ou compatíveis com o acaso (RAINIO; TEUHO; KLÉN, 2024).

Um teste de hipóteses confronta uma hipótese nula (H_0), que postula a ausência de efeito, com uma hipótese alternativa (H_1). A partir dos dados, calcula-se um p-valor: a probabilidade de se observar um resultado tão ou mais extremo que o obtido, caso H_0 seja verdadeira. Quando o p-valor é inferior a um nível de significância α previamente fixado (tipicamente 0,05), rejeita-se H_0; caso contrário, não há evidência suficiente para rejeitá-la. A não rejeição de H_0 não prova a igualdade entre os grupos, indicando apenas ausência de evidência de diferença na amostra avaliada.

A escolha do teste depende das propriedades dos dados. Testes paramétricos, como o teste t, pressupõem normalidade e homogeneidade de variância; quando esses pressupostos não se sustentam — caso de métricas limitadas a intervalos fechados, distribuições assimétricas ou amostras pequenas —, recomendam-se testes não-paramétricos, baseados em postos em vez dos valores brutos. Para comparar duas condições aplicadas às mesmas unidades experimentais (medições pareadas), o teste de Wilcoxon dos postos sinalizados é a alternativa não-paramétrica ao teste t pareado, recomendado para a comparação de modelos e configurações avaliados sobre os mesmos dados (RAINIO; TEUHO; KLÉN, 2024).

Quando várias comparações são conduzidas sobre os mesmos dados, a probabilidade de se obter ao menos um falso positivo cresce com o número de testes — o problema das comparações múltiplas. Para controlá-lo, aplicam-se correções como a de Bonferroni, que divide o nível de significância α pelo número de comparações realizadas, tornando o critério de rejeição mais conservador.

2.7 Trabalhos Relacionados

Esta seção apresenta uma revisão de trabalhos que abordam sistemas de reconhecimento facial em tempo real, com ênfase em aplicações de controle de presença em ambientes educacionais e nas decisões arquiteturais que impactam seu desempenho operacional.

Sistemas de reconhecimento facial em tempo real exigem arquiteturas capazes de lidar com grandes volumes de dados, múltiplas fontes de vídeo e processamento contínuo de alta intensidade. Para atender a essas demandas, a escalabilidade torna-se um requisito central, permitindo distribuir as tarefas entre diferentes componentes e manter o desempenho mesmo sob picos de carga.

Entre as abordagens mais recorrentes está o uso de mensageria assíncrona, que possibilita o desacoplamento entre as etapas de captura, detecção, reconhecimento e armazenamento. Esse modelo, amplamente discutido em arquiteturas distribuídas, permite que os módulos se comuniquem de forma independente, garantindo maior tolerância a falhas e controle do fluxo de processamento (VELEPUCHA; FLORES, 2023; SETIAWAN; HERMAWAN; SETIAWAN, 2025).

A adoção de microsserviços é outra estratégia consolidada para sistemas que demandam

escalabilidade e flexibilidade. Cada serviço é responsável por uma função específica — como detecção, extração de características ou persistência de dados — e pode ser escalado separadamente conforme a necessidade. Essa separação reduz o acoplamento e favorece a evolução contínua do sistema, uma característica essencial em aplicações baseadas em aprendizado de máquina (NEWMAN, 2015).

Outro aspecto fundamental em arquiteturas escaláveis é o armazenamento distribuído de dados, especialmente quando há múltiplas fontes de captura e grande volume de imagens processadas. Sistemas de arquivos distribuídos permitem que diferentes nós compartilhem e acessem dados de forma simultânea, mantendo consistência, disponibilidade e tolerância a falhas. Em aplicações de reconhecimento facial, esse tipo de estrutura garante alta taxa de leitura e escrita, essencial para o processamento contínuo de vídeos e registros de presença. Além disso, soluções modernas adotam estratégias de replicação e balanceamento automático de carga, otimizando o desempenho e reduzindo o tempo de acesso a arquivos em ambientes de larga escala (SETIAWAN; HERMAWAN; SETIAWAN, 2025).

(SHARZIZI et al., 2026) avaliaram quatro modelos de reconhecimento facial — FaceNet, SFace, OpenFace e DeepFace — em um protótipo web desenvolvido no Google Colab, medindo acurácia, distância de similaridade, latência de processamento e robustez sob variações de pose, iluminação e oclusão. Os experimentos utilizaram imagens estáticas carregadas manualmente pelo usuário — uma imagem de referência e uma foto de grupo —, sem processamento de vídeo contínuo. Os resultados demonstraram que não existe um modelo universalmente superior: FaceNet apresentou maior acurácia mas latência elevada (12,92 s), enquanto OpenFace ofereceu melhor equilíbrio entre velocidade (6,02 s) e precisão. O estudo conclui que a escolha do modelo deve ser orientada pelo equilíbrio entre acurácia e velocidade requerido pela aplicação.

Diferentemente de (SHARZIZI et al., 2026), o presente trabalho não trata a escolha do modelo como variável independente — o FaceNet512 foi fixado em todas as arquiteturas avaliadas. O foco recai sobre as decisões arquiteturais de software que sustentam o pipeline de processamento contínuo de vídeo CFTV em tempo real, dimensão não abordada por aquele estudo. Enquanto (SHARZIZI et al., 2026) operam sobre imagens estáticas em um ambiente síncrono e monolítico, este trabalho avalia como estratégias de desacoplamento assíncrono, armazenamento distribuído e fragmentação de dados impactam a viabilidade operacional do sistema em condições reais de fluxo contínuo.

(KANAGAMALLIGA et al., 2024) propuseram o HC-CNN, um método de reconhecimento facial em tempo real para videovigilância que combina Haar Cascade — responsável pela detecção rápida — com Redes Neurais Convolucionais para o reconhecimento, sobre uma arquitetura híbrida *edge*-nuvem em que a detecção inicial ocorre localmente, com baixa latência, e as operações mais custosas são delegadas à nuvem. O modelo foi treinado nos *datasets* WIDER Face e CelebA, que reúnem variações de pose, iluminação e oclusão. Avaliado contra uma CNN de referência, o HC-CNN elevou o F1-Score de 0,90 para 0,97, a precisão de 0,87 para 0,94 e o

recall de 0,89 para 0,96, processando vídeo a 30 *FPS* em dispositivos de borda. Assim como nos demais trabalhos, no entanto, a variável avaliada é o modelo de reconhecimento, enquanto a arquitetura de software permanece como meio para viabilizar o desempenho em tempo real, não como objeto de investigação.

(IMRAN et al., 2024) propuseram o FaceEngine, um framework baseado em rastreamento para reconhecimento facial em tempo real em sistemas de vigilância por câmeras CCTV. Os autores identificaram que abordagens convencionais apresentam dificuldades para reconhecer indivíduos em deslocamento físico pelo campo de visão da câmera (i.e., caminhando em velocidade normal) e exigem grande volume de imagens de treinamento por pessoa. O sistema proposto incorpora mecanismos de otimização de entrada, rastreamento e votação temporal, alcançando 90,91% de acurácia com 10 votos temporais e otimização de entrada ativa em condições reais. Sem otimização, a acurácia cai para 86,36% nas mesmas condições. A presença de imagens anômalas no banco de dados reduziu a acurácia para 77,27%, enquanto imagens desfocadas resultaram em 86,36%, evidenciando a sensibilidade do sistema à qualidade dos dados de entrada.

(SON et al., 2020) implantaram um sistema de controle de presença baseado em câmeras CFTV em ambiente acadêmico real no FPT Polytechnic College, Vietnã, cobrindo 120 estudantes em cinco salas. O sistema estrutura o processamento em um modelo *mestre-escravo*: um componente central agrupa os frames em *jobs* de 2 minutos e os distribui a *workers* paralelos responsáveis pela detecção via MTCNN, extração de *embeddings* com ArcFace e classificação por KNN. Para mitigar erros causados por desfoque de movimento e variações de pose, os autores propõem um algoritmo de sumarização temporal baseado em rastreamento de *bounding boxes* entre frames consecutivos, que elevou a acurácia de 91,3% para 92,7%, com taxa de falsos positivos reduzida a 1%. Excluindo amostras classificadas como desconhecidas, a acurácia atinge 98,5%. Entre as limitações reconhecidas pelos autores, destacam-se a inviabilidade de atingir a precisão de *benchmarks* controlados em ambientes reais, a degradação causada por variações de iluminação, desfoque e resolução das câmeras, e a baixa revocação (*recall*) em condições subótimas. Além disso, o sistema foi avaliado apenas com 120 estudantes, aquém da capacidade projetada de 500, e o processamento paralelo em uma única máquina mostrou ganho de velocidade não linear ao aumentar o número de *workers*.

O trabalho de (SON et al., 2020) compartilha com o presente estudo o uso de vídeo CFTV em ambiente institucional real como base experimental. No entanto, há uma diferença fundamental de foco: (SON et al., 2020) tratam a arquitetura de *software* como premissa fixa e concentram a avaliação no desempenho do modelo de reconhecimento. O presente trabalho inverte essa perspectiva — fixa o modelo (FaceNet512 com MediaPipe) e trata as decisões arquiteturais como variável independente, avaliando sistematicamente como cada escolha de *software* impacta a viabilidade operacional do sistema em tempo real.

(PATEL et al., 2025) investigaram sistemas de controle de presença baseados em CCTV em salas de aula reais, utilizando ResNet-50 e VGG-16 com transfer learning, e observaram

que mesmo com aumento de dados e redução de dimensionalidade via PCA, o ambiente não controlado impõe limitações ao desempenho que não se manifestam em datasets estáticos. Esses resultados reforçam que a qualidade do reconhecimento em cenários reais depende não apenas do modelo empregado, mas das decisões arquiteturais.

Os trabalhos analisados avançam na direção de modelos mais precisos e robustos para o reconhecimento facial em ambientes reais: desde a comparação sistemática de detectores e modelos de reconhecimento (SHARZIZI et al., 2026; SERENGIL; ÖZPINAR, 2024), passando por frameworks baseados em rastreamento (IMRAN et al., 2024), até aplicações diretas em controle de presença acadêmico (SON et al., 2020; PATEL et al., 2025). Contudo, em todos esses trabalhos, a infraestrutura de software que sustenta o pipeline de processamento é tratada como um dado — escolhida pragmaticamente, mas nunca avaliada como variável independente.

Essa ausência configura uma lacuna relevante: não há, na literatura consultada, uma avaliação sistemática de como decisões arquiteturais de software impactam a viabilidade operacional de sistemas de reconhecimento facial em tempo real. É justamente essa lacuna que o presente trabalho se propõe a preencher, partindo da hipótese de que a arquitetura de software é um fator tão determinante quanto o modelo utilizado.

A Tabela 2 sintetiza os trabalhos discutidos. Em todos eles, a variável de interesse é o modelo de reconhecimento, enquanto a arquitetura de software permanece como premissa fixa — contraste que delimita a lacuna investigada neste trabalho.

Tabela 2 – Comparação entre os trabalhos relacionados.

Trabalho	**Técnica de reconhecimento**	**Ambiente de dados**	**Vídeo contínuo em tempo real**	**Variável de interesse**
Sharzizi et al. (2026)	FaceNet, SFace, OpenFace, DeepFace	Imagens estáticas (web/Colab)	Não	Modelo
Kanagamalliga et al. (2024)	Haar Cascade + CNN	Videovigilância (edge/nuvem)	Sim	Modelo
Imran et al. (2024)	YOLOv5 + ArcFace (rastreamento e votação temporal)	CCTV de vigilância	Sim	Modelo
Son et al. (2020)	MTCNN + ArcFace + KNN	CFTV acadêmico real	Não	Modelo
Patel et al. (2025)	ResNet-50 e VGG-16 (*transfer learning*)	CCTV em sala de aula real	Sim	Modelo
Este trabalho	FaceNet512 + MediaPipe	CFTV institucional real	Sim	**Arquitetura de software**

Fonte: Elaboração própria.

3 METODOLOGIA

Este capítulo descreve a caracterização metodológica da pesquisa, apresenta o *dataset* construído, o ambiente de execução, o modelo básico e as cinco configurações arquiteturais avaliadas no experimento — Ingênua, Monolítica, Microsserviços, Cache e Fragmentação —, incluindo as considerações éticas e legais sobre o uso de atributos demográficos.

O sistema foi concebido de forma modular e independente da fonte de captura de imagens, o que permite sua aplicação em diferentes contextos que demandem identificação contínua de indivíduos. No contexto educacional, ele pode apoiar o controle de acesso às dependências do campus, o monitoramento do fluxo de pessoas na portaria, o registro automático de frequência de alunos, a segurança patrimonial e o monitoramento de fluxo em espaços públicos.

As cinco configurações arquiteturais foram construídas de forma incremental, partindo de uma Arquitetura Ingênua — que executa a detecção facial no navegador do cliente — e evoluindo progressivamente até configurações distribuídas com mensageria assíncrona, armazenamento distribuído e fragmentação de dados, cada uma concebida para superar limitações identificadas na anterior. O detalhamento dessa progressão e seu papel no desenho experimental são apresentados na Seção 3.1.

Os experimentos foram conduzidos sobre um *dataset* próprio, construído a partir de vídeos reais captados pelo sistema de CFTV da portaria do IFPB Campus Santa Rita. Todas as configurações foram avaliadas sobre o mesmo conjunto de 30 vídeos de teste, com os parâmetros mantidos fixos, de modo que as variações de desempenho observadas possam ser atribuídas às diferenças arquiteturais — ressalvadas as condições de teste específicas das arquiteturas Ingênua e Monolítica, detalhadas na Seção 3.1. As seções a seguir detalham cada um desses elementos.

3.1 Caracterização Metodológica

Quanto à natureza, esta pesquisa classifica-se como aplicada, por gerar conhecimento dirigido à solução de um problema concreto — a viabilização operacional de um sistema de reconhecimento facial em tempo real no IFPB Campus Santa Rita. Quanto aos objetivos, é explicativa, pois busca estabelecer relações de causa e efeito entre as decisões arquiteturais e o desempenho do sistema em suas múltiplas dimensões — classificação, agrupamento e custo operacional. Quanto ao procedimento, caracteriza-se como uma **investigação experimental** (WOHLIN et al., 2012).

3.1.1 Natureza Experimental

Em um experimento controlado, o pesquisador manipula deliberadamente um fator de interesse, mantendo as demais variáveis constantes, a fim de isolar e medir seu efeito sobre

indicadores previamente definidos (WOHLIN et al., 2012). Neste trabalho, o fator manipulado é a configuração arquitetural, avaliada sob condições fixas de modelo de reconhecimento, detector, limiares e conjunto de dados, com o objetivo de testar hipóteses formuladas a priori sobre seu impacto no desempenho do sistema. Os vídeos reais de CFTV constituem o material experimental sobre o qual as configurações são comparadas em igualdade de condições, aproximando os resultados das circunstâncias efetivas de operação sem reduzir o controle sobre o fator avaliado.

3.1.2 Variáveis

O experimento organiza-se em torno de uma variável independente, três grupos de variáveis dependentes e um conjunto de variáveis mantidas sob controle, sintetizados na Tabela 3.

Tabela 3 – Variáveis do experimento

Tipo	**Definição**
Independente	Configuração arquitetural, com cinco níveis: Ingênua, Monolítica, Microsserviços, Cache e Fragmentação.
Dependentes	*Classificação*: acurácia, precisão, recall, F1-Score. *Agrupamento*: Covering, Silhouette, Homogeneidade, Completude, V-Measure, distâncias intra e inter-cluster. *Operacionais*: razão tempo/duração e tamanho do banco de dados auxiliar.
Controladas	Modelo de reconhecimento (FaceNet512); detector facial (MediaPipe BlazeFace Full-range); limiar de similaridade (0,30); confiança mínima de detecção (0,80); resolução de entrada (1344×760); conjunto de teste (os mesmos 30 vídeos); intervalo de envio (`um frame a cada dez`).

Fonte: Elaboração própria

Cabe registrar uma ameaça à validade interna: as arquiteturas Ingênua e Monolítica não preservaram integralmente as variáveis de controle — a Ingênua executou a detecção no navegador com o modelo *BlazeFace Short-range*, e ambas utilizaram entrada emulada por *webcam* via OBS Studio. Por essa razão, são tratadas como *baselines* de referência, enquanto o experimento estritamente controlado — no qual apenas a arquitetura varia — concentra-se nas configurações Microsserviços, Cache e Fragmentação.

3.1.3 Hipóteses de Pesquisa

A investigação é orientada por uma hipótese nula e três hipóteses alternativas dirigidas, cada uma associada a uma intervenção arquitetural específica: $\mathbf{H_0}$**:** a configuração arquitetural não exerce efeito estatisticamente significativo sobre as métricas de viabilidade operacional do sistema — isto é, não há diferença significativa entre as configurações avaliadas; $\mathbf{H_1}$**:** a adoção de processamento distribuído com mensageria assíncrona (Microsserviços) eleva significativamente o desempenho de classificação em relação à *baseline* monolítica síncrona; $\mathbf{H_2}$**:** a priorização dinâmica de candidatos por aparição recente (Cache) reduz a razão tempo/duração em relação à

arquitetura de Microsserviços, sem degradar as métricas de classificação e de agrupamento; **H_3:** a fragmentação do banco por gênero estimado (Fragmentação) melhora o desempenho operacional em relação à arquitetura com Cache, sem degradar a qualidade do agrupamento. A verificação de cada hipótese é apresentada no Capítulo 5.

3.1.4 Abordagem Iterativa e Incremental

A abordagem iterativa e incremental, consolidada na engenharia de software por Pressman e Maxim (2021), consiste em desenvolver o sistema por meio de ciclos sucessivos, nos quais cada iteração produz uma versão funcional que incorpora melhorias identificadas na iteração anterior. Essa abordagem é adequada quando os requisitos evoluem ao longo do desenvolvimento e quando se deseja observar o impacto isolado de cada decisão de projeto sobre o comportamento do sistema.

Neste trabalho, a abordagem se manifesta na progressão das cinco arquiteturas desenvolvidas, cada uma concebida para superar uma limitação identificada na anterior. A Arquitetura Ingênua estabelece a linha de base como prova de conceito, sem otimizações. A Monolítica consolida o pipeline disperso da prova de conceito em um único serviço. A de Microsserviços introduz o desacoplamento entre as etapas por meio de mensageria assíncrona, permitindo que captura, detecção, reconhecimento e armazenamento operem de forma independente. A de Cache acrescenta a priorização por aparição, evitando o reprocessamento de indivíduos já reconhecidos. Por fim, a de Fragmentação investiga o particionamento dos dados por estimativa demográfica como estratégia para reduzir o espaço de busca. Essa progressão permite atribuir as variações de desempenho observadas às decisões arquiteturais introduzidas em cada etapa, conferindo rastreabilidade às conclusões.

3.1.5 Desenvolvimento Incremental das Configurações

As cinco configurações comparadas no experimento foram construídas de forma incremental, segundo a abordagem iterativa e incremental consolidada na engenharia de software por Pressman e Maxim (2021), na qual cada iteração produz uma versão funcional que incorpora melhorias identificadas na anterior. Neste trabalho, a progressão das arquiteturas — Ingênua, Monolítica, Microsserviços, Cache e Fragmentação — foi conduzida de modo que cada versão fosse concebida para superar uma limitação específica da precedente. Cada versão consolidada constitui um nível da variável independente, o que permite atribuir as variações de desempenho observadas à decisão arquitetural introduzida em cada etapa, conferindo rastreabilidade às conclusões.

3.1.6 Protocolo de Coleta de Dados e Procedimentos Experimentais

A coleta de dados e os procedimentos experimentais adotados neste trabalho seguem um protocolo estruturado, cujos detalhes — fontes de captação, segmentação dos vídeos, construção

do *ground truth* e configuração do ambiente de execução — são descritos na Seção 3.2. Em termos metodológicos, cabe destacar três decisões de protocolo que conferem validade interna aos resultados.

Primeiro, todas as arquiteturas foram avaliadas sobre o mesmo conjunto de 30 vídeos de teste (A01-ENTRADA a A30-ENTRADA), mantendo fixos os parâmetros de configuração — modelo de reconhecimento, detector facial, limiar de similaridade e resolução de entrada. A partir da Arquitetura com Microsserviços, os vídeos foram alimentados no pipeline por meio do worker de captura, configurado com intervalo de envio de 1 frame a cada 10 capturados (`frame_skip` $= 10$), garantindo condições de entrada equivalentes para as arquiteturas com Microsserviços, Cache e Fragmentação. Para as arquiteturas Ingênua e Monolítica, os vídeos foram emulados como entrada de webcam via OBS Studio, condição que difere das demais e deve ser considerada na interpretação comparativa dos resultados — conforme discutido nas respectivas seções de cada arquitetura.

Segundo, os resultados foram organizados em três grupos de métricas complementares — classificação, agrupamento e operacionais —, descritos na Seção 2.5, permitindo uma avaliação multidimensional da viabilidade de cada arquitetura.

Terceiro, o código-fonte de cada arquitetura foi disponibilizado publicamente no GitHub, viabilizando a replicação dos experimentos por trabalhos futuros.

3.1.7 Análise Estatística

Cada hipótese foi associada a uma comparação planejada entre duas configurações, definida a priori. Como o mesmo conjunto de 30 vídeos foi submetido a todas as configurações, as observações são pareadas vídeo a vídeo, o que controla a heterogeneidade de dificuldade entre eles. Em cada comparação, aplicou-se o teste de Wilcoxon dos postos sinalizados — alternativa não-paramétrica ao teste t pareado —, adequado a métricas limitadas ao intervalo $[0,1]$, assimétricas e com empates (WOHLIN et al., 2012).

Foram realizadas três comparações, uma por hipótese: Monolítica *versus* Microsserviços (H_1), Microsserviços *versus* Cache (H_2) e Cache *versus* Fragmentação (H_3). Na H_1, a comparação utilizou a acurácia em vez do F1-Score, por ser sempre definida: o F1-Score torna-se indefinido quando não há nenhum verdadeiro positivo ($TP = 0$), cenário que a acurácia contorna por ser sempre calculável. Para controlar o erro acumulado dos três testes, aplicou-se a correção de Bonferroni, com limiar de significância de $\alpha = 0{,}0167$ ($0{,}05/3$).

É importante ressaltar que a não rejeição de H_0 indica ausência de evidência de diferença na escala de dados avaliada, e não prova de equivalência entre as configurações — distinção relevante para a interpretação de qualquer comparação cujo resultado não seja significativo.

Os testes foram aplicados à métrica que operacionaliza diretamente a afirmação de cada hipótese — a acurácia para o desempenho de classificação, a razão tempo/duração para o custo

operacional e o V-Measure para a qualidade do agrupamento —, complementadas, quando pertinente, por testes de apoio sobre as demais dimensões. Optou-se por não submeter todas as métricas a teste, tanto para evitar a inflação do erro decorrente de comparações múltiplas quanto pela forte correlação entre métricas de um mesmo grupo (o V-Measure, por exemplo, integra homogeneidade e completude). As médias de todas as métricas permanecem reportadas de forma descritiva nas tabelas consolidadas e no Apêndice A.

Os testes foram computados por meio de um script em Python com a biblioteca SciPy, apresentado integralmente no Apêndice B, juntamente com a saída de execução, para fins de transparência e verificação da análise.

3.2 Dataset

A escolha do conjunto de dados é um dos elementos mais determinantes para o desempenho de qualquer sistema de reconhecimento facial. Na literatura, existem diversos *datasets* amplamente utilizados para treinamento e avaliação de modelos, como *Labeled Faces in the Wild* (LFW) (HUANG et al., 2007), VGGFace2 (CAO et al., 2018). Esses conjuntos são essenciais para a pesquisa acadêmica, pois oferecem grande volume de imagens e ampla diversidade de indivíduos, expressões e condições de iluminação. Entretanto, esses datasets apresentam limitações importantes quando se busca avaliar um sistema voltado a cenários específicos, como o controle de presença em ambientes educacionais. Em geral, tratam-se de bases formadas por imagens estáticas, capturadas em condições controladas e, portanto, distantes da realidade operacional de um sistema que processa vídeo em tempo real, onde há variação constante de iluminação, movimento e qualidade de captura.

Diante dessa diferença entre os contextos experimentais e o ambiente real de aplicação, optou-se por criar um dataset próprio, utilizando vídeos reais captados pelo sistema de CFTV da portaria do Instituto Federal da Paraíba — Campus Santa Rita. Essa decisão permitiu representar com fidelidade o comportamento do sistema em condições autênticas, incluindo fluxo contínuo de pessoas, diferentes horários do dia e mudanças nas condições de iluminação. Por envolver imagens de pessoas em ambiente institucional, o dataset não é disponibilizado publicamente, em conformidade com a Lei Geral de Proteção de Dados (LGPD).

3.2.1 Coleta e Segmentação dos Vídeos

As gravações foram realizadas em seis dias distintos — 20/03/2025 (07:00 às 08:00), 31/03/2025 (07:00 às 08:00), 01/04/2025 (06:00 às 08:00), 02/04/2025 (06:00 às 08:00), 03/04/2025 (06:00 às 08:00) e 04/04/2025 (06:00 às 08:00) — período correspondente à entrada dos alunos no turno da manhã, momento de maior fluxo de pessoas na portaria. Os vídeos foram capturados em resolução de 2688×1520 pixels, a 20 quadros por segundo (fps), configurando um material de boa qualidade visual e temporalmente consistente para análise de movimento e

detecção facial.

O sistema de CFTV da instituição gera vídeos com duração padrão de 30 minutos. Para adequar esse material ao formato desejado de experimentação, foi utilizado o aplicativo LosslessCut, que permite cortes rápidos e sem recompressão. Os vídeos originais foram segmentados em diferentes durações, dando origem ao dataset final, composto por 125 vídeos de aproximadamente 1 minuto, 70 vídeos de cerca de 5 minutos e 12 vídeos de 30 minutos. Essa distribuição foi planejada para representar diferentes níveis de carga de processamento, possibilitando observar o comportamento do sistema tanto em análises curtas quanto em períodos prolongados de monitoramento contínuo.

Cabe registrar que não foram adotados critérios formais de inclusão ou exclusão de vídeos — a intenção inicial era processar todos os segmentos disponíveis, mas restrições de tempo inviabilizaram a formalização desse protocolo. Essa ausência constitui uma limitação metodológica que deve ser considerada na interpretação dos resultados.

3.2.2 Conjunto de Teste e Distribuição Temporal

Os experimentos foram conduzidos sobre um subconjunto de 30 (trinta) vídeos de aproximadamente 1 minuto, identificados de A01-ENTRADA a A30-ENTRADA. Esses vídeos correspondem a dois dias de captação, conforme detalhado na Tabela 4.

Tabela 4 – Distribuição temporal dos vídeos do conjunto de teste

Vídeo	Data	Horário	Pessoas Únicas	Frames Vazios (%)
A01-ENTRADA	20/03/2025	07:00	4	76,8
A02-ENTRADA	20/03/2025	07:01	2	92,6
A03-ENTRADA	20/03/2025	07:03	2	87,9
A04-ENTRADA	20/03/2025	07:05	2	85,2
A05-ENTRADA	20/03/2025	07:21	4	52,5
A06-ENTRADA	20/03/2025	07:27	2	87,5
A07-ENTRADA	20/03/2025	07:30	5	70,5
A08-ENTRADA	20/03/2025	07:31	1	72,7
A09-ENTRADA	20/03/2025	07:38	7	54,6
A10-ENTRADA	20/03/2025	07:40	58	20,3
A11-ENTRADA	20/03/2025	07:41	8	51,3
A12-ENTRADA	20/03/2025	07:42	62	16,6
A13-ENTRADA	20/03/2025	07:44	52	20,1
A14-ENTRADA	20/03/2025	07:45	37	52,8
A15-ENTRADA	20/03/2025	07:55	2	90,2
A16-ENTRADA	01/04/2025	06:10	1	82,7
A17-ENTRADA	01/04/2025	06:12	1	96,0
A18-ENTRADA	01/04/2025	06:38	4	86,2
A19-ENTRADA	01/04/2025	06:51	5	89,3
A20-ENTRADA	01/04/2025	06:52	9	78,2
A21-ENTRADA	01/04/2025	06:53	3	94,0
A22-ENTRADA	01/04/2025	06:54	2	97,7
A23-ENTRADA	01/04/2025	07:15	2	96,6
A24-ENTRADA	01/04/2025	07:01	5	82,5
A25-ENTRADA	01/04/2025	07:03	2	89,4
A26-ENTRADA	01/04/2025	07:05	2	90,5
A27-ENTRADA	01/04/2025	07:06	3	64,6
A28-ENTRADA	01/04/2025	07:11	7	87,6
A29-ENTRADA	01/04/2025	07:13	2	96,1
A30-ENTRADA	01/04/2025	07:14	14	60,1

Pessoas Únicas: número de indivíduos distintos identificados manualmente no vídeo. Frames Vazios: percentual de frames sem detecção de faces. Fonte: Elaboração própria.

A Tabela 5 apresenta as estatísticas descritivas do conjunto de teste, evidenciando a heterogeneidade da carga de processamento entre os vídeos avaliados.

Tabela 5 – Estatísticas descritivas do conjunto de teste (A01-ENTRADA a A30-ENTRADA)

Atributo	Mín.	Máx.	Média	Mediana
Duração (s)	59,65	61,90	60,97	61,10
Pessoas únicas	1	62	9,77	3,50
Total de frames	1193	1238	1221	1222
Frames vazios (%)	16,6	97,7	74,4	83,9

Fonte: Elaboração própria.

Ressalta-se que não foi realizado um levantamento formal do número total de indivíduos únicos no dataset consolidado, uma vez que cada vídeo foi tratado de forma independente, com seu próprio *ground truth*, sem cruzamento de identidades entre vídeos. A partir da planilha de controle das amostras, é possível estimar que os vídeos do conjunto de teste correspondem a aproximadamente 248 indivíduos distintos no dia 20/03/2025 (07:00–07:45) e aproximadamente 62 indivíduos no dia 01/04/2025 (06:10–07:14). Essa estimativa contextualiza a escala do problema de agrupamento enfrentado pelo sistema e auxilia na interpretação das métricas de Completude e V-Measure reportadas no Capítulo 4.

3.2.3 Protocolo de Construção do Ground Truth

O *ground truth* foi construído por meio de uma etapa manual de curadoria, realizada com o auxílio do software Kinovea[1]. Por suas funcionalidades voltadas à análise quadro a quadro, o Kinovea permitiu navegar pelos frames de cada vídeo com precisão, possibilitando ao pesquisador selecionar, para cada indivíduo, os quadros mais adequados à identificação facial segundo três critérios de qualidade visual: nitidez (ausência de desfoque de movimento), iluminação adequada e enquadramento frontal do rosto.

Para cada vídeo, foi criada uma pasta específica contendo subpastas numeradas sequencialmente de 1 a n, onde n corresponde ao número de indivíduos distintos identificados no vídeo. Cada subpasta contém exatamente uma imagem — a melhor face disponível do respectivo indivíduo —, de modo que o *ground truth* adota uma abordagem *one-shot*: uma única imagem de referência por indivíduo por vídeo.

O critério de seleção adotado foi a qualidade visual geral da imagem, avaliada subjetivamente pelo pesquisador quadro a quadro. Para cada indivíduo, buscou-se o frame que apresentasse a melhor combinação dos três critérios — nitidez, iluminação e enquadramento frontal. Nos casos em que nenhum frame disponível apresentava qualidade visual satisfatória — por oclusão parcial, desfoque de movimento ou ângulo desfavorável —, o indivíduo foi descartado do *ground truth* daquele vídeo.

Cabe destacar as limitações desse protocolo. Não foram estabelecidos critérios objetivos e mensuráveis de qualidade de imagem — como resolução mínima da região facial em pixels, limiar numérico de nitidez ou ângulo máximo de rotação da cabeça aceito —, nem foi registrada a frequência de descartes por vídeo. A curadoria foi realizada por um único pesquisador, sem verificação de concordância entre avaliadores por meio de coeficientes como o Kappa de Cohen. Essas escolhas introduzem subjetividade na referência utilizada para o cálculo das métricas de classificação e restringem a plena reprodutibilidade do experimento. A definição de critérios formais e a adoção de múltiplos avaliadores constituem direções prioritárias para trabalhos futuros.

[1] Disponível em: <https://www.kinovea.org>. Acesso em: [PREENCHER: data].

3.2.4 Anonimização

O processo de anonimização adotado neste trabalho é estrutural: ao processar os vídeos, o sistema atribui automaticamente a cada indivíduo detectado um identificador único universal (UUID) gerado aleatoriamente, sem qualquer vínculo com dados pessoais reais — nome, matrícula ou qualquer outra informação identificável. Os identificadores sequenciais utilizados nas subpastas do dataset (1 a *n*) são independentes dos UUIDs gerados pelo sistema, de modo que não há correspondência possível entre os dois conjuntos. Não foram aplicados procedimentos técnicos de desfoque ou supressão de faces nos vídeos originais, uma vez que o material não é disponibilizado publicamente e permaneceu sob acesso exclusivo do pesquisador responsável, em conformidade com a autorização concedida pela Direção-Geral da instituição exclusivamente para fins de pesquisa científica.

3.3 Ambiente de Execução

Os experimentos foram conduzidos em um único servidor com as configurações descritas na Tabela 6. Os serviços de infraestrutura — MongoDB, RabbitMQ e MinIO — foram executados em containers Docker. O código-fonte de cada arquitetura está disponível publicamente no GitHub, conforme indicado na Tabela 6.

Cabe observar que, diferentemente de ambientes Linux — nos quais containers Docker são executados nativamente —, o Windows 11 executa containers por meio de uma camada de virtualização (WSL 2 ou Hyper-V), o que pode introduzir overhead adicional de I/O e memória. Essa característica do ambiente de execução deve ser considerada na interpretação das métricas operacionais reportadas, uma vez que os tempos de processamento observados podem ser superiores aos que seriam obtidos em uma implantação nativa em Linux.

Tabela 6 – Configuração do ambiente de execução dos experimentos

Componente	**Configuração**
Hardware	
Processador	Intel Core i7-11800H @ 2,30 GHz (16 CPUs)
Memória RAM	32 GB
GPU	NVIDIA GeForce RTX 3060 Laptop (6 GB VRAM)
Armazenamento	NVMe Samsung MZVL21T0HCLR (1 TB)
Sistema Operacional	Windows 11 Home 64 bits
Software	
Python	3.10.0
DeepFace	0.0.93
TensorFlow	2.11.0
MongoDB	8.0.12 (standalone) / 6.x (cluster sharding)
RabbitMQ	3-management
MinIO	RELEASE.2025-04-08T15-41-24Z
Parâmetros	
Modelo de reconhecimento	FaceNet512
Detector facial	MediaPipe (BlazeFace Full-range)
Detector demográfico	MTCNN
Métrica de similaridade	Distância cosseno
SIMILARITY_THRESHOLD	0,30
MIN_DETECTION_CONFIDENCE	0,80
Resolução de entrada	1344×760 px (reduzida de 2688×1520)
Paralelismo	ProcessPoolExecutor (4 workers)
Priorização por aparição	Ordenação por `last_appearance` decrescente
Chave de shard (MongoDB)	`sexo` (gênero estimado)
Número de shards	2
Intervalo de envio de frames	1 frame a cada 10 capturados (`frame_skip` = 10)
Repositórios públicos (GitHub)	
Ingênua (V1)	<github.com/tiagorocha1/Prova-de-conceito-mestrado-V1>
Monolítica (V3)	<github.com/tiagorocha1/Prova-de-conceito-mestrado-V3>
Microsserviços (V5)	<github.com/tiagorocha1/Prova-de-conceito-mestrado-V5>
Cache (V7)	<github.com/tiagorocha1/Prova-de-conceito-mestrado-V7>
Fragmentação (V9)	<github.com/tiagorocha1/Prova-de-conceito-mestrado-V9>

Fonte: Elaboração própria.

Os repositórios disponibilizados correspondem às versões ímpares (V1, V3, V5, V7, V9), que representam as versões consolidadas de cada arquitetura. As versões pares (V2, V4, V6, V8) correspondem a versões de transição, utilizadas durante o processo de refatoração entre duas arquiteturas consecutivas, e não foram incluídas por não constituírem configurações estáveis para fins de avaliação.

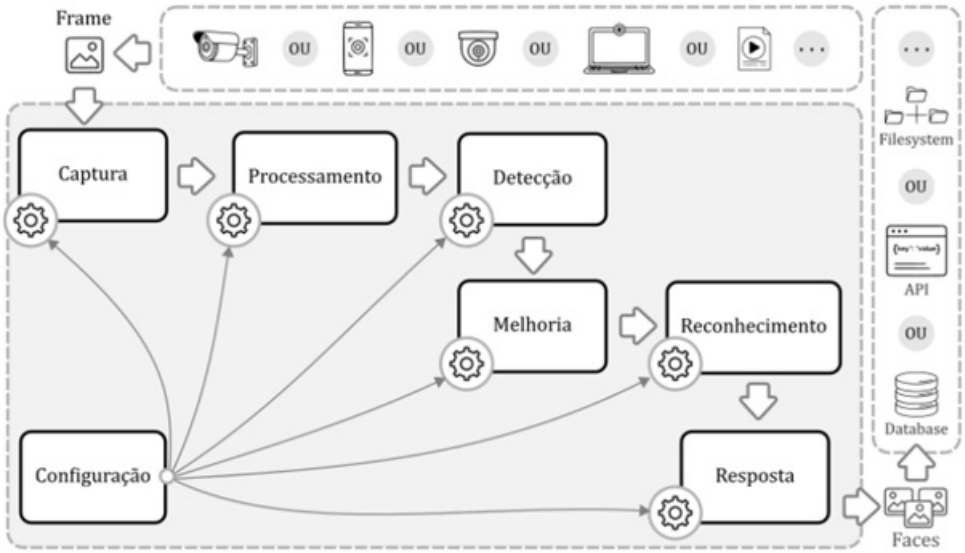
3.4 Modelo Básico

O modelo básico foi a ideia inicial construída a partir da análise dos trabalhos dos demais pesquisadores da área de visão computacional. Neste modelo conceitual foram definidas as etapas existentes no processo de reconhecimento facial e a ordem de interação entre elas sem

ainda entrar em aspectos técnicos ou de implementação. O objetivo foi estabelecer um ponto de partida sólido que pudesse atender uma ampla gama de contextos de sistemas que utilizam o reconhecimento facial em alguma de suas funcionalidades.

Nesse modelo de software descentralizado, cada etapa do processo de reconhecimento facial é executada por um agente independente. A Figura 4 apresenta o modelo proposto, ilustrando o fluxo de dados entre os módulos e as diferentes fontes de captura de vídeo. Nela é possível observar como o modelo organiza as etapas do processo de reconhecimento facial, onde cada uma delas é desenvolvida em um modulo especifico, por exemplo, o módulo de captura apenas capta frames da fonte configurada e os encaminha para o módulo seguinte.

Figura 4 – Modelo básico



Fonte: Elaboração própria.

O módulo de captação é o responsável por receber as imagens que alimentam o sistema, podendo vir de diferentes fontes, como câmeras fixas, webcams ou dispositivos móveis. O módulo de processamento, por sua vez, atua como uma etapa intermediária, responsável por preparar os frames capturados e encaminhá-los adequadamente para as fases seguintes. Já o módulo de detecção identifica as regiões da imagem onde há rostos, enquanto o módulo de melhoria aplica ajustes como alinhamento facial, correção de iluminação e remoção de ruídos, otimizando a qualidade da imagem antes do reconhecimento. O módulo de reconhecimento compara as características extraídas das imagens capturadas com aquelas armazenadas na base de dados, determinando a identidade dos indivíduos. O módulo de registro consolida o resultado, armazenando as presenças identificadas e garantindo a persistência das informações. Em seguida, o sistema gera uma resposta, que representa o resultado do processo — seja a confirmação da presença, o retorno de logs ou a atualização de dados sobre as faces reconhecidas.

Além desses componentes, o modelo também prevê um módulo de configuração, que atua de forma transversal a todos os demais. Esse módulo é responsável por definir parâmetros

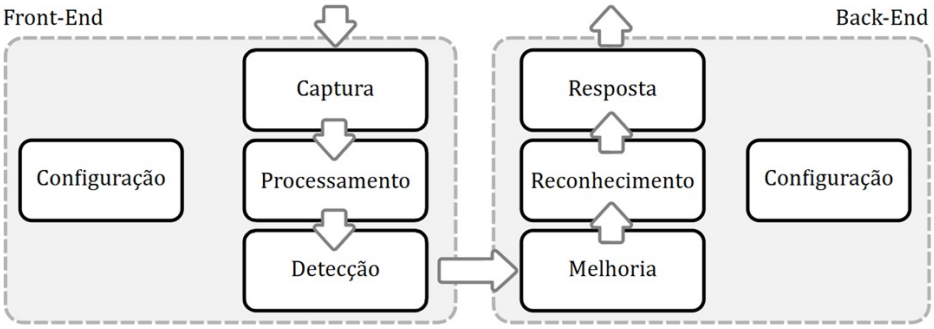
de operação, como fontes de captura, limiares de similaridade e políticas de armazenamento, garantindo a flexibilidade e adaptabilidade do sistema a diferentes contextos. Essa organização modular facilita a compreensão do processo como um todo e cria uma base flexível para evolução. A partir dela, foi possível aprimorar o modelo em diferentes direções, introduzindo melhorias de desempenho, escalabilidade e precisão, que serão detalhadas nas próximas subseções.

As cinco configurações arquiteturais apresentadas nas seções seguintes são propostas originais dos autores, concebidas a partir desse modelo básico para o problema de reconhecimento facial em tempo real em ambiente institucional.

3.5 Arquitetura Ingênua

A Arquitetura Ingênua[2] constitui a prova de conceito inicial do sistema: uma implementação reduzida cujo objetivo é demonstrar a viabilidade técnica do reconhecimento facial aplicado ao controle de presença, sem compromisso com completude funcional, robustez de produção ou validação por usuários finais. Por isso, a arquitetura adotada foi propositalmente simples, priorizando a clareza do fluxo de execução e a integração básica entre os componentes essenciais. A Figura 5 ilustra a arquitetura, apresentando o fluxo básico. Nela, é possível visualizar a comunicação direta entre o *frontend* e o *backend*, onde todas as etapas — desde a captura até o reconhecimento facial — ocorrem de forma sequencial e centralizada.

Figura 5 – Arquitetura Ingênua



Fonte: Elaboração própria.

Essa versão inicial, foi implementada com tecnologias amplamente conhecidas e de fácil integração. O frontend foi desenvolvido em React, responsável pela interface web e pela captura das imagens. Para a detecção e o recorte das faces diretamente no navegador, foi utilizado o MediaPipe (LUGARESI et al., 2019), um framework de código aberto desenvolvido pelo Google para construção de pipelines de percepção em tempo real. O backend foi construído em

[2] Uma versão inicial deste sistema de reconhecimento facial para registro de presença foi premiada como melhor Resumo Expandido da temática Tecnologia da Informação e Comunicação, modalidade Iniciação Científica e Tecnológica, no 4º Simpósio de Pesquisa, Inovação e Pós-Graduação do IFPB (SIMPIF), em 2021.

FastAPI, que serviu como API REST para gerenciar o cadastro de pessoas, processar as imagens recebidas e registrar as presenças. O reconhecimento facial propriamente dito foi realizado por meio da biblioteca DeepFace, utilizando o modelo FaceNet512 para extração e comparação de embeddings. Os dados de pessoas e presenças foram armazenados em um banco MongoDB, enquanto as imagens capturadas eram salvas em um sistema de arquivos local.

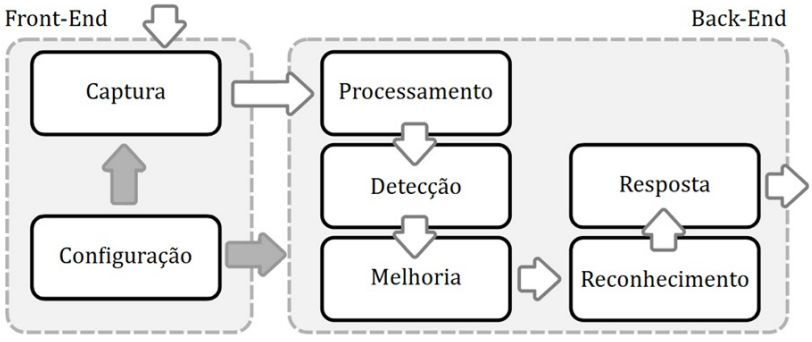
É importante destacar uma limitação que afeta a comparabilidade direta desta arquitetura com as demais: como os vídeos de teste são gravações, foi necessário emulá-los como entrada de webcam utilizando o OBS Studio. Adicionalmente, o modelo *BlazeFace Full-range* do MediaPipe não é compatível com o ambiente do navegador, de modo que a detecção facial foi realizada com o modelo *BlazeFace Short-range* — mais leve e com menor alcance de detecção. Essas condições de teste diferem das demais arquiteturas, nas quais o processamento ocorre integralmente no backend com o modelo *BlazeFace Full-range*, o que deve ser considerado na interpretação comparativa dos resultados.

3.6 Arquitetura Monolítica

A segunda arquitetura desenvolvida tem como objetivo eliminar a dependência do *hardware* cliente identificada na Arquitetura Ingênua, migrando todas as etapas do pipeline de reconhecimento facial — detecção, recorte e reconhecimento — para o *backend*. Essa centralização caracteriza o modelo monolítico, no qual todos os componentes essenciais do sistema compartilham o mesmo ambiente de execução no servidor.

A Figura 6 apresenta a arquitetura monolítica, destacando a centralização das etapas de detecção, recorte e reconhecimento facial no servidor. O diagrama evidencia a simplificação do cliente e a unificação do processamento no *backend*.

Figura 6 – Arquitetura Monolítica



Fonte: Elaboração própria.

Com essa abordagem, o frontend passou a ter uma função mais simples: apenas capturar e enviar os frames para o servidor. Todo o restante do processamento passou a ocorrer no backend, o que trouxe ganhos significativos em termos de organização e controle do fluxo. Essa

centralização também facilitou o gerenciamento dos logs e a depuração de falhas, já que todas as etapas críticas do processo ficaram concentradas em um único ambiente de execução.

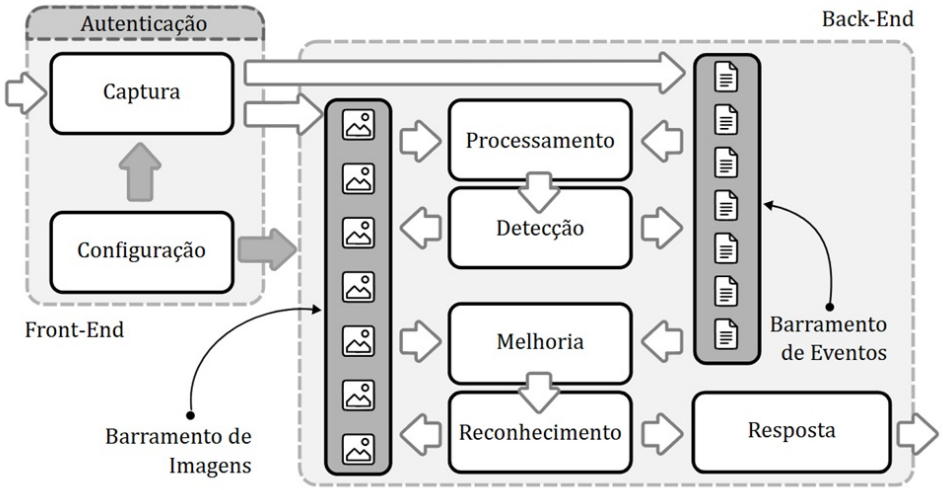
No servidor, a detecção de faces foi executada utilizando o MediaPipe. O reconhecimento facial continuou sendo feito pela biblioteca DeepFace, integrada diretamente à API construída em FastAPI, o que permitiu que todo o processo fosse orquestrado de maneira mais coesa. O armazenamento de dados e imagens permaneceu baseado no MongoDB e no sistema de arquivos local, respectivamente. A centralização do processamento proporcionou uma experiência de uso mais estável, uma vez que o navegador deixou de ser responsável por operações intensivas de processamento, o que também ampliou a compatibilidade com diferentes dispositivos.

3.7 Arquitetura com Microsserviços

Após avaliar o funcionamento da Arquitetura Monolítica, evidenciou-se sua principal limitação: por executar as etapas de forma sequencial e síncrona em um único processo, um gargalo em qualquer etapa — como a detecção facial — bloqueava todo o pipeline, e a escalabilidade ficava condicionada ao componente mais lento. Para superar essa limitação, adotou-se uma arquitetura baseada em microsserviços, que permite distribuir o processamento e escalar cada etapa de forma independente.

A Figura 7 ilustra a arquitetura baseada em microsserviços. Nela, observam-se os módulos independentes de captura, detecção, reconhecimento e armazenamento, conectados por meio do Barramento de Imagens e do Barramento de Eventos.

Figura 7 – Arquitetura Microsserviços



Fonte: Elaboração própria.

O backend foi reestruturado para adotar um modelo modular e distribuído. As etapas de captura de frames, detecção de faces, reconhecimento facial e armazenamento de dados foram

divididas em serviços especializados, que se comunicavam entre si por meio de mensageria assíncrona, implementada com o RabbitMQ. Essa mudança reduziu o acoplamento entre os módulos e aumentou a tolerância a falhas: caso algum serviço ficasse temporariamente indisponível, as tarefas pendentes eram mantidas na fila até que pudessem ser processadas.

O worker de captura é o ponto de entrada do pipeline: implementado em Python com interface gráfica em Tkinter e OpenCV, ele abstrai as diferentes fontes de vídeo — arquivo local, webcam e câmera IP via stream —, permitindo que gravações do dataset sejam processadas pelo sistema da mesma forma que uma câmera em operação real, sem modificações nos demais workers. Antes do envio, cada frame é redimensionado de 2688×1520 para 1344×760 pixels e codificado em PNG. O frame é então armazenado no MinIO e uma mensagem de metadados é publicada na fila do RabbitMQ contendo o caminho do frame, o instante de captura, a taxa de quadros, a duração do vídeo e uma *tag* identificadora do vídeo utilizada para rastreamento durante os experimentos. O worker também permite configurar o intervalo de envio — 1 frame a cada N frames capturados —, o que possibilitou controlar a carga de processamento sem modificar o pipeline das demais arquiteturas.

Diferentemente das arquiteturas Ingênua e Monolítica — nas quais a captura era realizada no próprio *frontend*, no navegador —, a partir desta arquitetura a captura passou a ser conduzida por este worker dedicado, integrado ao RabbitMQ e ao MinIO. A interface gráfica do worker de captura — por meio da qual foram configurados a fonte de vídeo (webcam, arquivo local ou câmera IP via stream), a taxa de quadros, a duração e o intervalo de envio de frames utilizado nos experimentos — é apresentada no Apêndice C (Figura 13).

O armazenamento de imagens também foi aprimorado. Em vez de utilizar o sistema de arquivos local, o sistema passou a contar com um repositório distribuído baseado no MinIO, garantindo maior disponibilidade, segurança e escalabilidade dos arquivos. Essa abordagem permitiu que múltiplas instâncias do backend pudessem acessar o mesmo conjunto de imagens de forma síncrona, sem dependência de um servidor físico específico.

Em relação à segurança, foram introduzidos mecanismos de autenticação e proteção de dados mais robustos. O sistema passou a utilizar OAuth2 e JWT para autenticação de usuários e controle de acesso, além de adotar o algoritmo Argon2 para a criptografia de senhas. Essas melhorias reforçaram a confiabilidade da aplicação e a proteção das informações sensíveis processadas pelo sistema.

No frontend, o foco foi a ampliação das funcionalidades de controle e visualização. Foram incluídos painéis analíticos para monitorar presenças, exibir estatísticas e acompanhar o fluxo de processamento de frames em tempo real. Também foi implementado um mecanismo de login, garantindo que apenas usuários autorizados pudessem acessar os dados. As telas do sistema — autenticação, listagem de registros de presença e as três visões do dashboard analítico — são apresentadas no Apêndice C (Figuras 14, 15, 16, 17 e 18).

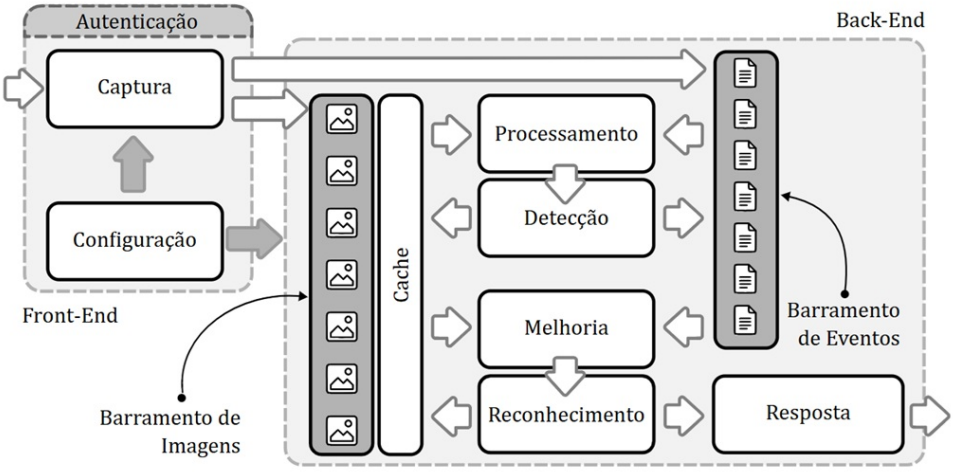
3.8 Arquitetura com Cache

Embora denominada “Arquitetura com Cache”, esta versão não utiliza um mecanismo de cache tradicional com estrutura dedicada de armazenamento temporário (como Redis ou Memcached). O termo foi adotado por aproximação conceitual: a estratégia implementada consiste na priorização dinâmica dos candidatos no banco de dados, ordenando-os pelo campo `last_appearance` em ordem decrescente antes de iniciar as comparações de embeddings.

Com a estrutura distribuída consolidada, o foco passou a ser o desempenho do processo de reconhecimento facial. O objetivo principal foi reduzir o tempo médio de resposta, tornando o sistema mais fluido durante o processamento contínuo de frames. Para isso, foi introduzida uma estratégia de priorização dinâmica de candidatos por aparição.

A Figura 8 representa a arquitetura com priorização por aparição. O mecanismo consiste em ordenar os candidatos no banco de dados pelo campo `last_appearance` em ordem decrescente antes de iniciar as comparações de embeddings. Com isso, indivíduos vistos mais recentemente são comparados primeiro, aumentando a probabilidade de encontrar um match nas primeiras iterações — especialmente em ambientes como uma portaria, onde a mesma pessoa tende a reaparecer em frames consecutivos. Essa abordagem reduz o tempo médio até o primeiro match sem necessidade de estrutura de cache dedicada ou armazenamento temporário adicional.

Figura 8 – Arquitetura com Cache



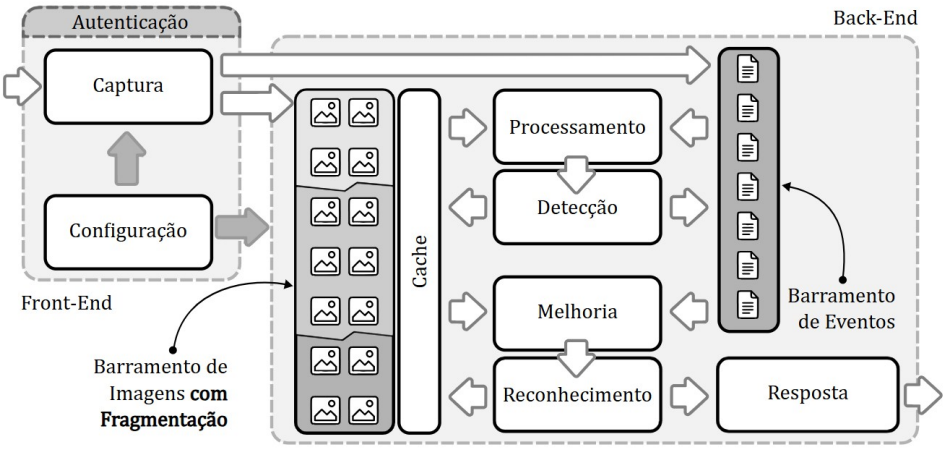
Fonte: Elaboração própria.

3.9 Arquitetura com Fragmentação

Com o desempenho já otimizado pelas estratégias de cache e priorização dinâmica, o objetivo agora passou a ser reduzir o volume de comparações desnecessárias e diminuir a latência das consultas ao banco de dados. Para isso, foram introduzidos dois novos elementos: o filtro demográfico e o mecanismo de fragmentação (sharding) no banco de dados MongoDB.

A Figura 9 ilustra a arquitetura com fragmentação, destacando o uso do filtro demográfico antes da etapa de reconhecimento facial e a aplicação do sharding no MongoDB. O diagrama enfatiza como o fluxo de dados foi reorganizado para reduzir comparações e otimizar consultas em larga escala. O filtro demográfico foi incorporado ao fluxo de reconhecimento como uma etapa anterior à comparação dos embeddings.

Figura 9 – Arquitetura com Fragmentação



Fonte: Elaboração própria.

A ideia é dar subsídio ao mecanismo de sharding do banco de dados. Para isso, cada face detectada passa por uma etapa de análise demográfica executada pelo DeepFace com o detector MTCNN, que estima atributos como idade aproximada, gênero, raça predominante e emoção. Desses atributos, apenas o gênero estimado — armazenado no campo `sexo` — é utilizado como chave de particionamento no MongoDB. O banco de dados passou a utilizar fragmentação (sharding) com dois shards, distribuindo os registros de acordo com o gênero estimado, com o objetivo de reduzir o espaço de busca nas comparações de embeddings e melhorar o desempenho das consultas em momentos de maior demanda.

3.10 Considerações Éticas e Legais

Este trabalho envolve o processamento de dados biométricos — reconhecimento facial de indivíduos em ambiente institucional —, o que exige atenção às disposições da Lei Geral de Proteção de Dados (Lei nº 13.709/2018) em duas dimensões: o uso geral das imagens captadas pelo sistema de CFTV e o uso específico de atributos demográficos na Arquitetura com Fragmentação.

3.10.1 Uso Geral das Imagens e Base Legal

O uso das imagens captadas pelo sistema de CFTV da portaria do IFPB Campus Santa Rita foi autorizado pela Direção-Geral da instituição, exclusivamente para fins de pesquisa científica, com aplicação de medidas de anonimização. Essa autorização enquadra o tratamento dos dados na base legal prevista no Art. 11, II, alínea *c* da LGPD, que admite o tratamento de dados pessoais sensíveis — incluindo dados biométricos — para fins de pesquisa científica, desde que observadas as salvaguardas necessárias à proteção dos titulares. Em conformidade com esse enquadramento, o dataset não é disponibilizado publicamente.

3.10.2 Salvaguarda de Privacidade Aplicada em Todas as Arquiteturas

Em todas as arquiteturas desenvolvidas, as faces detectadas são recortadas e armazenadas no MinIO vinculadas ao registro de presença correspondente, porém associadas exclusivamente a um identificador único universal (UUID) gerado aleatoriamente — sem qualquer vínculo com dados cadastrais reais como nome ou matrícula. No contexto experimental, os indivíduos presentes no dataset recebem identificadores sequenciais independentes, sem correspondência com os UUIDs gerados pelo sistema, o que impede o cruzamento entre dados experimentais e registros operacionais. Além das imagens das faces, são persistidos os *embeddings* faciais — representações matemáticas que não permitem a reconstrução da imagem original — e metadados como o instante de detecção.

Ressalta-se ainda que toda a infraestrutura permaneceu ativa exclusivamente durante a execução dos experimentos, sendo desativada ao término de cada sessão, e que apenas o pesquisador responsável teve acesso ao ambiente de execução e aos dados coletados. Essas medidas configuram pseudonimização nos termos do Art. 13, §4º da LGPD e minimizam o risco de exposição indevida das informações biométricas dos indivíduos captados.

3.10.3 Uso de Atributos Demográficos na Arquitetura com Fragmentação

A Arquitetura com Fragmentação introduz um elemento adicional de atenção ética: a análise demográfica executada pelo DeepFace com o detector MTCNN estima atributos de idade, gênero, raça predominante e emoção para cada face detectada. Desses atributos, apenas o gênero estimado — armazenado no campo `sexo` — é utilizado como chave de particionamento no MongoDB. Os demais atributos, incluindo raça, são estimados mas não utilizados para particionamento nem para decisões sobre indivíduos. Ainda assim, o simples armazenamento desses atributos estimados no banco de dados requer atenção às exigências do Art. 11 da LGPD, que regula o tratamento de dados pessoais sensíveis — entre os quais a LGPD classifica dado referente à origem racial ou étnica (Art. 5º, II).

Do ponto de vista técnico, modelos de estimativa demográfica apresentam taxas de erro sistematicamente maiores para determinados grupos populacionais (SUMSION et al., 2024).No presente trabalho, como apenas o gênero estimado é utilizado como chave de particionamento,

erros nessa estimativa específica fazem com que faces de uma mesma pessoa sejam distribuídas entre fragmentos distintos — por exemplo, quando o mesmo indivíduo é classificado como masculino em um frame e feminino em outro —, o que tende a fragmentar os agrupamentos de uma mesma identidade. O impacto desse efeito sobre a qualidade do agrupamento é analisado no Capítulo 4.

O uso de um atributo sensível como chave de particionamento — ainda que sem efeito sobre decisões a respeito de indivíduos — constitui, portanto, uma limitação ética e técnica do design desta arquitetura, cujas implicações e possíveis salvaguardas são retomadas no Capítulo 6.

4 RESULTADOS

Este capítulo apresenta os resultados obtidos nos experimentos realizados com as cinco arquiteturas desenvolvidas. Os testes foram conduzidos sobre um conjunto de 30 vídeos de aproximadamente 1 minuto, identificados de A01-ENTRADA a A30-ENTRADA, extraídos do dataset descrito no Capítulo 3. Para as arquiteturas Ingênua e Monolítica, são reportadas apenas as métricas de classificação: o número de faces detectadas e reconhecidas foi tão baixo — com ausência total de detecções na maioria dos vídeos — que não há agrupamentos significativos a avaliar, e as métricas de agrupamento e operacionais pressupõem um volume mínimo de reconhecimentos bem-sucedidos, condição não atendida por essas duas arquiteturas. Esse comportamento é coerente com a natureza de cada uma — a Ingênua como prova de conceito sem otimizações e a Monolítica sem mecanismos de controle de fluxo —, de modo que a inviabilidade do cálculo das demais métricas era esperada e, em si, já constitui um resultado sobre os limites dessas configurações. As arquiteturas com Microsserviços, Cache e Fragmentação, por sua vez, apresentaram desempenho suficiente para o cálculo do conjunto completo de métricas definido na Seção 2.5.

4.1 Arquitetura Ingênua

A Arquitetura Ingênua apresentou desempenho muito limitado. Em 22 dos 30 vídeos avaliados não houve nenhuma detecção bem-sucedida ($TP = 0$), resultando em acurácia média de 0,04. Nos 8 vídeos em que ao menos uma face foi reconhecida (A06, A07, A10–A14 e A30), a precisão atingiu 1,00, com recall médio de 0,04 e F1-Score médio de 0,27. As métricas de agrupamento e operacionais não foram calculadas para esta arquitetura.

Cabe registrar que o MediaPipe foi utilizado para detecção de faces em todas as arquiteturas. Na Arquitetura Ingênua, o modelo é executado no cliente (navegador), onde apenas o modelo *BlazeFace Short-range* funcionou — o modelo *BlazeFace Full-range* não é compatível com o ambiente do navegador. Foram realizados testes de compatibilidade nos navegadores Firefox, Chrome, Opera e Microsoft Edge, mas nenhuma das tentativas resultou em êxito. Segundo a documentação oficial do MediaPipe (Google LLC, 2023), o modelo *Short-range* foi projetado para detectar faces a curta distância (até aproximadamente 2 metros), enquanto o *Full-range* estende esse alcance para distâncias maiores e oferece maior robustez a variações de pose e escala. Essa diferença entre os modelos, combinada com a necessidade de emular os vídeos como entrada de webcam via OBS Studio, contribui para explicar o desempenho significativamente inferior da Arquitetura Ingênua em relação às demais. Os resultados individuais por vídeo constam no Apêndice A (Tabela 13).

4.2 Arquitetura Monolítica

A centralização da detecção no *backend* elevou a acurácia média para 0,13. A precisão média caiu para 0,43, o recall médio foi de 0,18 e o F1-Score de 0,31. O número de vídeos sem nenhuma detecção caiu em relação à arquitetura anterior, e surgiram os primeiros falsos positivos. As métricas de agrupamento e operacionais não foram calculadas para esta arquitetura. Os resultados individuais por vídeo constam no Apêndice A (Tabela 14).

4.3 Arquitetura com Microsserviços

A arquitetura atingiu acurácia média de 0,92, precisão de 0,99, recall de 0,93 e F1-Score de 0,95, com 12 dos 30 vídeos obtendo F1-Score igual a 1,00. As quedas de desempenho nas demais amostras concentraram-se em vídeos com maior densidade de pessoas, refletidas nos valores de FN. O Covering médio foi de 0,82, V-Measure de 0,82, Completeness de 0,77 e razão tempo/duração de 0,688. Os resultados individuais por vídeo constam no Apêndice A (Tabelas 15, 16 e 17).

4.4 Arquitetura com Cache

A arquitetura obteve acurácia de 0,91, precisão de 0,99, recall de 0,92 e F1-Score de 0,95. As métricas de agrupamento permaneceram praticamente idênticas às de Microsserviços — Covering 0,81, V-Measure 0,82 e Silhouette 0,34. A razão tempo/duração caiu para 0,679. Os resultados individuais por vídeo constam no Apêndice A (Tabelas 18, 19 e 20).

4.5 Arquitetura com Fragmentação

A arquitetura obteve acurácia de 0,921, precisão de 0,996, recall de 0,925 e F1-Score de 0,955. O V-Measure caiu para 0,56, o Silhouette para 0,06 e a Completeness para 0,45. A razão tempo/duração subiu para 0,91. Os resultados individuais por vídeo constam no Apêndice A (Tabelas 21, 22 e 23).

4.6 Comparação entre as Arquiteturas

As Tabelas 7, 8 e 9 consolidam as médias das métricas de classificação, agrupamento e operacionais para todas as arquiteturas avaliadas, permitindo a comparação direta entre as configurações desenvolvidas.

Tabela 7 – Médias das métricas de classificação por arquitetura

Arquitetura	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Ingênua	0,04*	1,00	0,04*	0,27
Monolítica	0,13*	0,43	0,18	0,31
Microsserviços	0,92	0,99	**0,93**	0,95
Cache	0,91	0,99	0,92	0,95
Fragmentação	**0,92**	**1,00**	0,92	**0,96**

Os valores representam a média de cada métrica sobre os 30 vídeos avaliados.* Desempenho inviável para o cenário proposto: a ausência de controle de fluxo e a dependência do hardware cliente resultaram em acurácia abaixo de 0,15 e recall abaixo de 0,20, tornando essas arquiteturas inadequadas para uso em produção. Fonte: Elaboração própria.

Tabela 8 – Médias das métricas de agrupamento (Microsserviços, Cache e Fragmentação)

Arquitetura	**Cover.**	**Inter.**	**Intra.**	**Silh.**	**Homog.**	**Compl.**	**V-Meas.**
Microsserviços	**0,819**	**23,466**	5,661	0,332	0,961	**0,766**	**0,818**
Cache	0,812	23,457	**5,620**	**0,343**	0,961	0,764	0,817
Fragmentação	0,813	21,345	3,317*	0,056*	**0,966**	0,446*	0,555*

Cover.: Covering; Inter.: Inter-Distância; Intra.: Intra-Distância; Silh.: Silhouette; Homog.: Homogeneity; Compl.: Completeness; V-Meas.: V-Measure. Silhouette calculada sobre os 25 vídeos com valor definido em cada arquitetura. * Queda significativa em relação às arquiteturas anteriores: o filtro demográfico distribuiu faces de uma mesma pessoa entre fragmentos distintos, reduzindo a Completeness de 0,77 para 0,45 e o V-Measure de 0,82 para 0,56. Fonte: Elaboração própria.

Tabela 9 – Médias das métricas operacionais (Microsserviços, Cache e Fragmentação)

Arquitetura	**Tempo / Duração**	**Tamanho BD Auxiliar (MB)**
Microsserviços	0,688	0,42
Cache	0,679	0,42
Fragmentação	0,911*	0,41

* Razão próxima ao limite do tempo real (1,00): a sobrecarga das consultas fragmentadas elevou o tempo de processamento em $\approx 34\%$ em relação à arquitetura com Cache. As arquiteturas Ingênua e Monolítica não são incluídas por não terem atingido desempenho suficiente para o cálculo dessas métricas. Fonte: Elaboração própria.

A visualização gráfica desses resultados consolidados é apresentada no Apêndice A (Figuras 10, 11 e 12).

Para verificar se as diferenças observadas nas médias são estatisticamente significativas, aplicaram-se testes de Wilcoxon pareados, conforme o procedimento descrito na Seção 3.1.7. A Tabela 10 apresenta a estatística W e o p-valor de cada comparação. A interpretação desses resultados — a verificação de cada hipótese — é apresentada no Capítulo 5.

Tabela 10 – Resultados dos testes de hipóteses (Wilcoxon pareado)

Hip.	**Comparação**	**Métrica**	W	p
H_1	Monolítica × Microsserviços	**Acurácia***	0	$< 0{,}001$
H_2	Microsserviços × Cache	**Tempo/duração***	122	0,107
H_2	Microsserviços × Cache	Acurácia	3	0,465
H_2	Microsserviços × Cache	V-Measure	27	0,959
H_3	Cache × Fragmentação	**V-Measure***	0	$< 0{,}001$
H_3	Cache × Fragmentação	Tempo/duração	4	$< 0{,}001$

* Teste confirmatório da hipótese; as demais linhas são testes de apoio. Limiar de significância de Bonferroni: $\alpha = 0{,}0167$. Fonte: Elaboração própria.

A maior diferença entre as arquiteturas ocorre na transição da Monolítica para a Microsserviços, com o F1-Score subindo de 0,31 para 0,95. As Arquiteturas com Microsserviços e Cache apresentam métricas de classificação e de agrupamento praticamente idênticas, diferindo apenas marginalmente na razão tempo/duração. A Fragmentação, por sua vez, embora alcance o maior F1-Score (0,96), registra a menor qualidade de agrupamento (V-Measure 0,56) e o maior custo operacional (razão 0,91). A interpretação desses padrões e a verificação das hipóteses são desenvolvidas no Capítulo 5.

5 DISCUSSÃO

Este capítulo interpreta os resultados apresentados no Capítulo 4, verificando as hipóteses de pesquisa, discutindo as implicações práticas das decisões arquiteturais avaliadas, contextualizando os resultados frente à literatura e apresentando as limitações do trabalho.

5.1 Verificação das Hipóteses

A investigação foi orientada pela questão de qual o impacto de diferentes configurações de arquitetura de software sobre a viabilidade operacional de um sistema de reconhecimento facial em tempo real no IFPB Campus Santa Rita. Essa questão foi operacionalizada em três hipóteses dirigidas, verificadas a partir dos testes estatísticos reportados na Tabela 10 e sintetizadas na Tabela 11. Em conjunto, elas a respondem: as decisões arquiteturais exercem impacto determinante sobre a viabilidade do sistema, sendo o modelo de comunicação entre serviços o fator mais crítico observado.

Tabela 11 – Verificação das hipóteses de pesquisa

Hip.	**Hipótese (resumo)**	**Veredicto**
H_1	A mensageria assíncrona melhora o desempenho de classificação	Suportada
H_2	A priorização por aparição reduz o tempo sem degradar a qualidade	Parc. suportada
H_3	A fragmentação por gênero melhora o desempenho operacional sem degradar o agrupamento	Refutada

Resultados dos testes na Tabela 10. Fonte: Elaboração própria.

A hipótese H_1 foi suportada: a adoção de mensageria assíncrona elevou a acurácia de forma estatisticamente significativa em relação à *baseline* monolítica. A hipótese H_2 foi parcialmente suportada: a priorização por aparição preservou integralmente a qualidade — sem diferença significativa em classificação nem em agrupamento —, mas a redução da razão tempo/duração não atingiu significância, caracterizando equivalência operacional em vez do ganho previsto. A hipótese H_3 foi refutada: a fragmentação por gênero estimado degradou significativamente o V-Measure em relação ao Cache, contrariando a expectativa de preservar a qualidade do agrupamento.

Em conjunto, a hipótese nula H_0 é rejeitada: as diferenças significativas observadas em H_1 (classificação) e H_3 (agrupamento) demonstram que a configuração arquitetural afeta as métricas de viabilidade. A não significância detectada em H_2 não sustenta H_0, restringindo-se a uma equivalência entre duas configurações específicas (Microsserviços e Cache) no único nível de carga avaliado — vídeos curtos e base de indivíduos limitada. Como o ganho da priorização por aparição tende a crescer com o número de candidatos comparados por frame, não se pode

descartar que, sob demanda mais elevada (base maior ou operação prolongada), essa diferença se torne significativa — o que se registra como direção para trabalhos futuros.

5.2 Interpretação dos Resultados

As arquiteturas Ingênua e Monolítica demonstraram ser inviáveis para o cenário proposto. A execução da detecção facial no cliente, na Arquitetura Ingênua, resultou em acurácia de apenas 0,04, evidenciando que a dependência do hardware do usuário final inviabiliza o reconhecimento em condições reais. A centralização no backend, na Arquitetura Monolítica, elevou a acurácia para 0,13, mas sem mecanismos de controle de fluxo, o sistema permaneceu inconsistente e impreciso.

A transição para microsserviços com mensageria assíncrona foi a decisão arquitetural de maior impacto: a acurácia saltou para 0,92 e o F1-Score para 0,95, tornando o sistema viável pela primeira vez — resultado que sustenta a hipótese H_1 (Seção 5.1). Como o modelo de reconhecimento (FaceNet512) e o detector (MediaPipe BlazeFace Full-range) foram mantidos idênticos entre as duas configurações, o salto não pode ser atribuído ao modelo, mas à maneira como a arquitetura passou a lidar com o tempo. Esse resultado confirma que o desacoplamento entre as etapas de captura, detecção, reconhecimento e armazenamento — viabilizado pelo RabbitMQ — é condição necessária para absorver o fluxo contínuo de frames em tempo real sem degradar a qualidade do reconhecimento.

Embora a transição para microsserviços tenha envolvido múltiplas mudanças simultâneas — mensageria assíncrona via RabbitMQ, armazenamento distribuído com MinIO, paralelismo com *ProcessPoolExecutor* (4 workers) e mecanismos de autenticação —, a contribuição central pode ser atribuída ao desacoplamento do fluxo de processamento viabilizado pelo RabbitMQ. Nas arquiteturas anteriores, a ausência de controle de fila forçava o sistema a processar cada frame de forma síncrona e sequencial, criando um gargalo estrutural que nenhuma melhoria pontual de hardware ou modelo poderia resolver isoladamente. A 20 fps, esse comportamento síncrono se traduz em descarte de frames ou em atraso crescente, derrubando a cobertura das detecções independentemente da qualidade do modelo. A introdução da mensageria assíncrona eliminou esse gargalo ao permitir que os serviços de captura, detecção e reconhecimento operassem em ritmos independentes, absorvendo picos de carga sem degradar a qualidade do reconhecimento. Os demais componentes — MinIO, paralelismo e autenticação — ampliam a robustez e a escalabilidade do sistema, mas operam sobre uma fundação que só se tornou viável com o controle de fluxo assíncrono. Em trabalhos futuros, a condução de experimentos controlados, introduzindo cada componente isoladamente, permitiria quantificar essa contribuição com maior precisão.

A introdução da priorização dinâmica de candidatos por aparição — ordenando as comparações pelo campo `last_appearance` decrescente — preservou integralmente a qualidade

do reconhecimento, sem diferença significativa em relação à Arquitetura com Microsserviços nas métricas de classificação e de agrupamento; a redução média da razão tempo/duração (0,688 $\rightarrow$ 0,679), por sua vez, não atingiu significância estatística, de modo que a estratégia se caracteriza como uma otimização segura — sem regressão de qualidade — e operacionalmente equivalente à configuração anterior na escala avaliada. Esse padrão de equivalência é o que sustenta apenas parcialmente a hipótese H_2 (Seção 5.1). A preservação da qualidade decorre de a priorização atuar apenas sobre a ordem das comparações, sem alterar o limiar de similaridade que define o critério de reconhecimento. Em termos conceituais, essa estratégia guarda similaridade com o princípio de priorização do algoritmo LRU (*Least Recently Used*) aplicado em sistemas de cache (TANENBAUM; BOS, 2015): assim como o LRU mantém no topo da fila os itens acessados mais recentemente — por serem os mais prováveis de serem requisitados novamente em breve —, a abordagem adotada posiciona na frente das comparações os candidatos vistos mais recentemente, explorando o mesmo princípio de localidade temporal. Em ambientes como uma portaria, onde a mesma pessoa tende a reaparecer em frames consecutivos, essa ordenação reduz o número médio de comparações necessárias por frame até o primeiro *match*. Vale ressaltar que a analogia é parcial: diferentemente do LRU, a estratégia adotada não envolve descarte de candidatos, apenas reordenação da fila de comparação. Essa decisão demonstrou que otimizações locais de desempenho podem ser introduzidas de forma incremental sem risco de regressão, consolidando essa arquitetura como configuração de referência para o cenário avaliado.

Uma avaliação restrita às métricas de classificação apontaria a Fragmentação como a configuração mais avançada (F1-Score 0,96, precisão 1,00); é a análise conjunta dos três grupos de métricas que revela os custos ocultos dessa configuração. Embora tenha mantido a acurácia de classificação em 0,92, a fragmentação demográfica degradou significativamente o agrupamento — V-Measure de 0,82 para 0,56 — e elevou a razão tempo/duração para 0,91, próxima ao limite do tempo real — desfecho que refuta a hipótese H_3 (Seção 5.1). Como apenas o gênero estimado é utilizado como chave de particionamento, a degradação observada está diretamente associada à inconsistência nas estimativas de gênero produzidas pelo DeepFace entre frames distintos de um mesmo indivíduo — quando o gênero estimado varia entre frames, as faces do mesmo indivíduo são distribuídas em shards distintos e deixam de ser comparadas entre si durante o reconhecimento. O aumento do tempo, por sua vez, decorre de o overhead das consultas ao cluster de sharding do MongoDB superar o ganho obtido com a redução do espaço de busca, invertendo o efeito pretendido. Cabe ressaltar, no entanto, que a ausência de um experimento de controle com fragmentação por critério neutro — como um identificador numérico ou hash — impede isolar o efeito do sharding em si do efeito do uso do gênero estimado como chave de particionamento. Não é possível afirmar, portanto, se a degradação observada é inerente à estratégia de fragmentação ou específica ao uso de estimativas de gênero como critério de partição. Essa arquitetura evidencia que decisões de particionamento baseadas em estimativas demográficas introduzem um risco sistêmico que supera os ganhos de escalabilidade pretendidos, ao menos nas condições avaliadas.

A viabilidade operacional demonstrada pela Arquitetura com Cache — razão tempo/duração de 0,68, acurácia de 0,91 e F1-Score de 0,95 — indica que o reconhecimento facial em tempo real é tecnicamente alcançável, desde que a arquitetura de software adote processamento assíncrono e distribuído.

Contudo, o comportamento ultraconservador do sistema merece atenção. A precisão próxima a 1,00 combinada com recall de 0,92 significa que aproximadamente 8% das faces de indivíduos cadastrados não são reconhecidas pelo sistema.

Quatro fatores operam simultaneamente e cada um contribui, em grau não isolável a partir dos experimentos realizados, para a taxa de não-reconhecimento observada: o limiar de similaridade, mantido no valor padrão da biblioteca DeepFace para FaceNet-512 sem ajuste fino para o dataset; o redimensionamento dos frames de 2688×1520 para 1344×760 pixels, que reduz pela metade a resolução de entrada e pode degradar a qualidade dos embeddings extraídos; o limiar de confiança de detecção ($MIN\_DETECTION\_CONFIDENCE = 0{,}80$), que descarta faces detectadas com baixa confiança; e as variações de pose e iluminação inerentes ao ambiente CFTV, que introduzem variabilidade nas representações faciais.

Para isolar a contribuição de cada fator seria necessária uma análise de sensibilidade controlada, variando cada parâmetro de forma independente, essa análise será realizada em trabalhos futuros.

A Arquitetura com Fragmentação, apesar dos melhores resultados de classificação, não é recomendada para implantação no estado atual. A degradação do agrupamento e o aumento do tempo de processamento, combinados com as implicações éticas e legais do uso do gênero estimado como chave de particionamento — conforme discutido na Seção 3.10.3 —, tornam essa arquitetura inadequada para uso institucional sem salvaguardas adicionais.

A Tabela 12 sistematiza esses resultados em um referencial de impactos arquiteturais destinado a orientar decisões de projeto em contextos similares.

Tabela 12 – Tabela comparativa de impactos arquiteturais

Arquitetura	**Classificação**	**Agrupamento**	**Custo Operacional**
Ingênua	Inviável (F1 = 0,27)	Não avaliado	Baixo
Monolítica	Inviável (F1 = 0,31)	Não avaliado	Baixo–Médio
Microsserviços	Alta (F1 = 0,95)	Boa (V-Measure = 0,82)	Médio (razão 0,69)
Cache	Alta (F1 = 0,95)	Boa (V-Measure = 0,82)	Médio (razão 0,68)
Fragmentação	Alta (F1 = 0,96)	Degradada (V-Measure = 0,56)	Alto (razão 0,91) + riscos LGPD

Fonte: Elaboração própria.

5.3 Comparação com Trabalhos Relacionados

A interpretação dos resultados obtidos requer contextualização frente à literatura, uma vez que o desempenho de sistemas de reconhecimento facial varia substancialmente conforme as condições do ambiente de captura. Benchmarks realizados em datasets controlados, como o LFW (*Labeled Faces in the Wild*), atingem acurácias superiores a 98% com modelos como FaceNet-512 e RetinaFace (SERENGIL; ÖZPINAR, 2024). Contudo, esses resultados são obtidos sobre imagens estáticas, frontais e de alta qualidade, sendo pouco representativos de sistemas operando sobre vídeo CFTV em condições reais.

Conforme discutido na Seção 2.7, os trabalhos que operam sobre vídeo CFTV em ambientes reais relatam, de forma consistente, degradação de desempenho frente aos benchmarks controlados. Son et al. (2020), em controle de presença com CCTV acadêmico, elevaram a acurácia de 91,3% para 92,7% com sumarização temporal. Imran et al. (2024) reportam 90,91% de acurácia sobre CCTV de vigilância, também abaixo de seus próprios resultados em condições controladas. Patel et al. (2025), em salas de aula, observaram que nem o *transfer learning* nem o aumento de dados eliminam as limitações impostas pelo ambiente não controlado. Sharzizi et al. (2026), por sua vez, reforçam que não existe modelo universalmente superior para aplicações em tempo real, dependendo o desempenho do equilíbrio entre acurácia e velocidade imposto pelas restrições operacionais.

Kanagamalliga et al. (2024) investigaram algoritmos de reconhecimento facial em tempo real para videovigilância operando sob variações de iluminação, pose e oclusão, e destacaram que o equilíbrio entre desafios computacionais e escalabilidade exige decisões arquiteturais explícitas — como a integração de edge computing e soluções em nuvem —, reforçando que a viabilidade operacional de sistemas reais não depende apenas da precisão do modelo, mas da arquitetura que sustenta o pipeline de processamento.

Esses resultados, em conjunto, indicam que a acurácia de 0,92 obtida pela Arquitetura com Microsserviços é competitiva para um sistema operando sobre vídeo CFTV real, com fluxo contínuo de pessoas, câmera de longa distância e sem controle sobre iluminação ou posicionamento dos indivíduos. A diferença em relação aos 98,4% reportados por Serengil e Özpınar (2024) não reflete limitação do modelo, mas sim um conjunto de decisões técnicas motivadas pelos requisitos de operação em tempo real. Enquanto o benchmark LFW utiliza RetinaFace como detector — combinação que atinge 98,4% —, o presente trabalho adotou MediaPipe, que, segundo os próprios dados de Serengil e Özpınar (2024), atinge 96,1% com FaceNet-512 já em condições controladas. Essa escolha foi motivada pela necessidade de detecção em tempo real sobre vídeo contínuo a 20 fps. Adicionalmente, os frames foram redimensionados de 2688×1520 para 1344×760 pixels antes do processamento, reduzindo pela metade a resolução de entrada para viabilizar a operação dentro da janela temporal de cada frame. O limiar de similaridade coseno foi mantido no valor padrão da biblioteca, sem ajuste fino para o dataset utilizado.O limiar de confiança de detecção foi definido em 0,80 com base em

observação empírica durante os ensaios preliminares: foi a partir desse valor que o volume de faces corretamente detectadas passou a superar consistentemente o de detecções incorretas e de regiões sem face, favorecendo a precisão das detecções em detrimento do volume total de faces capturadas.

5.4 Limitações

Este trabalho apresenta limitações que devem ser consideradas na interpretação dos resultados.

O *ground truth* foi construído por um único avaliador, sem critérios formais de qualidade de imagem nem verificação de concordância entre avaliadores, o que introduz subjetividade na referência utilizada para o cálculo das métricas de classificação e restringe a reprodutibilidade do experimento.

A análise da Arquitetura com Fragmentação não incluiu uma configuração de controle com fragmentação por critério neutro — como identificador numérico ou hash —, o que impede isolar o efeito do sharding em si do efeito do uso do gênero estimado pelo DeepFace como chave de particionamento. Não é possível afirmar, a partir dos dados disponíveis, em que medida a regressão observada no agrupamento é inerente à estratégia de fragmentação ou específica à inconsistência das estimativas de gênero produzidas pelo DeepFace entre frames distintos de um mesmo indivíduo.

O limiar de similaridade coseno ($SIMILARITY\_THRESHOLD = 0{,}30$) foi mantido no valor padrão da biblioteca DeepFace, sem ajuste fino para o dataset e o cenário avaliados. A ausência de uma análise de sensibilidade impede afirmar se o desempenho reportado é próximo ao ótimo para este cenário específico — valores distintos poderiam alterar substancialmente o equilíbrio entre precisão e recall. Recomenda-se que trabalhos futuros avaliem ao menos quatro pontos de corte: 0,25, 0,30, 0,35 e 0,40, cobrindo uma faixa representativa em torno do valor padrão adotado.

Por fim, os experimentos foram conduzidos sobre vídeos de aproximadamente um minuto, extraídos de um único ponto de captura — a portaria do Campus Santa Rita. Embora esse cenário seja representativo do caso de uso pretendido, os resultados podem não se generalizar para ambientes com múltiplas câmeras, maior densidade de pessoas ou condições de iluminação distintas.

6 CONSIDERAÇÕES FINAIS

Este trabalho investigou como decisões arquiteturais de software impactam a viabilidade operacional de um sistema de reconhecimento facial em tempo real, tendo como ambiente experimental a portaria do IFPB Campus Santa Rita. Foram propostas, implementadas e avaliadas cinco arquiteturas incrementais — Ingênua, Monolítica, Microsserviços, Cache e Fragmentação — sobre um *dataset* próprio construído a partir de vídeos reais de CFTV, com métricas organizadas em três grupos: classificação, agrupamento e operacionais.

Os resultados demonstram que a arquitetura de *software* é um fator tão determinante quanto o modelo de reconhecimento adotado. As Arquiteturas Ingênua e Monolítica mostraram-se inviáveis para o cenário proposto — com F1-Score de 0,27 e 0,31, respectivamente —, evidenciando que nem a centralização do processamento no *backend* é suficiente sem mecanismos de controle de fluxo. A transição para microsserviços com mensageria assíncrona via RabbitMQ foi a decisão de maior impacto, elevando o F1-Score para 0,95 ao eliminar o gargalo síncrono que impedia o sistema de operar em sua capacidade plena.A Arquitetura com Cache introduziu uma otimização incremental — priorização de candidatos por aparição recente — estatisticamente equivalente à de Microsserviços em qualidade e em tempo, sem qualquer regressão, caracterizando-se como otimização de baixo risco e configuração de referência para o cenário avaliado. A Arquitetura com Fragmentação, embora tenha apresentado o melhor F1-Score (0,96) — margem de apenas 0,01 em relação às Arquiteturas com Microsserviços e Cache —, degradou significativamente o agrupamento (V-Measure de 0,82 para 0,56) e elevou o tempo de processamento, revelando que o uso de estimativas demográficas como chave de particionamento introduz riscos sistêmicos que superam o ganho marginal de classificação obtido.

Entre os pontos fortes do trabalho, destacam-se a condução dos experimentos sobre vídeo CFTV real em ambiente institucional — em contraste com a maioria dos trabalhos relacionados que operam sobre datasets estáticos controlados —, a avaliação sistemática de cinco arquiteturas incrementais com métricas multidimensionais, e a disponibilização pública do código-fonte de todas as arquiteturas, viabilizando a replicação dos experimentos. Entre as limitações, destacam-se a construção do *ground truth* por único avaliador, a ausência de análise de sensibilidade do limiar de similaridade e a restrição dos experimentos a um único ponto de captura, o que pode limitar a generalização dos resultados.

6.1 Trabalhos Futuros

A partir das limitações identificadas e dos resultados obtidos, destacam-se as seguintes direções para pesquisas futuras.

A investigação do impacto de diferentes valores de *SIMILARITY_THRESHOLD* sobre o

equilíbrio entre precisão e recall é a direção mais imediata, podendo melhorar a usabilidade do sistema sem comprometer sua confiabilidade. Valores como 0,35 ou 0,40 representam pontos de partida naturais para essa análise.

A integração do sistema com o SUAP (*Sistema Unificado de Administração Pública*) para registro automático de presença constitui um desdobramento natural, assim o sistema passaria a alimentar diretamente a base acadêmica institucional, eliminando a etapa de lançamento manual de frequência pelos docentes.

A avaliação de uma configuração alternativa de fragmentação por critério neutro — como identificador numérico ou intervalo temporal — permitiria separar o impacto do particionamento em si do impacto das estimativas demográficas, fornecendo evidências mais precisas sobre a viabilidade da estratégia de *sharding*.

A substituição do gênero estimado por atributos menos sensíveis e mais estáveis como critério de particionamento — como faixa etária ampla ou características geométricas neutras — reduziria os riscos éticos e legais associados à Arquitetura com Fragmentação, mantendo o potencial de ganho operacional.

A adoção de uma política de minimização de dados, descartando as estimativas demográficas tão logo cumprido o particionamento, reduziria a exposição de atributos sensíveis e fortaleceria a conformidade com o princípio da necessidade previsto na LGPD.

A avaliação do impacto diferencial das estimativas demográficas por subgrupo populacional — verificando se determinados grupos são sistematicamente prejudicados pela imprecisão do modelo de estimativa — constitui direção relevante, sobretudo diante da evidência de que tais modelos apresentam taxas de erro desiguais entre populações (SUMSION et al., 2024).

A construção de um *ground truth* com critérios objetivos e múltiplos avaliadores — incluindo resolução mínima da região facial, limiar de nitidez e ângulo máximo de rotação da cabeça — fortaleceria a reprodutibilidade dos experimentos e permitiria comparações mais rigorosas com trabalhos futuros.

A avaliação do sistema em cenários com múltiplas câmeras, maior densidade de pessoas e condições de iluminação variadas permitiria verificar a generalização dos resultados obtidos neste estudo de caso.

Por fim, a avaliação do sistema em condições adversas — como oclusão parcial por máscara facial, captura noturna com câmera infravermelha e movimentação rápida dos indivíduos — forneceria evidências sobre os limites operacionais de cada arquitetura em cenários de maior exigência, complementando os resultados obtidos nas condições naturais avaliadas neste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

ABUZAR, M.; AHMAD, A. b.; AHMAD, A. A. b. A survey on student attendance system using face recognition. In: *8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. [s.n.], 2020. p. 1252–1257. Acesso em: 01 maio 2024. Disponível em: <https://ieeexplore.ieee.org/document/9197815>. Citado na página 25.

ADJABI, I. et al. Past, present, and future of face recognition: A review. *Electronics (Switzerland)*, MDPI AG, v. 9, n. 8, p. 1–53, 2020. Acesso em: 01 maio 2024. Disponível em: <https://doi.org/10.3390/electronics9081188>. Citado na página 26.

ALI, W. et al. Methods for image denoising using convolutional neural network: a review. *Complex & Intelligent Systems*, 2021. Acesso em: 15 jun. 2024. Disponível em: <https://link.springer.com/article/10.1007/s40747-021-00428-4>. Citado na página 24.

ALI, W. et al. Classical and modern face recognition approaches: a complete review. *Multimedia tools and applications*, Springer, v. 80, n. 3, p. 4825–4880, 2021. Disponível em: <https://doi.org/10.1007/s11042-020-09850-1>. Acesso em: 23 jul. 2021. Citado 2 vezes nas páginas 25 e 26.

ALMABDY, S.; ELREFAEI, L. Deep convolutional neural network-based approaches for face recognition. *Applied Sciences*, MDPI, v. 9, n. 4397, p. 1–21, 2019. Acesso em: 01 maio 2024. Disponível em: <https://www.mdpi.com/2076-3417/9/20/4397>. Citado na página 22.

APPATI, J. K. et al. Analysis and implementation of optimization techniques for facial recognition. *Applied Computational Intelligence and Soft Computing*, Hindawi, v. 2021, March 2021. Acesso em: 01 maio 2024. Disponível em: <https://doi.org/10.1155/2021/6672578>. Citado na página 22.

BARBOSA, G. N. N. et al. Segurança em redes 5g: Oportunidades e desafios em detecção de anomalias e predição de tráfego baseadas em aprendizado de máquina. In: *XXI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg 2021)*. Sociedade Brasileira de Computação, 2021. cap. 4. Disponível em: <https://www.researchgate.net/publication/355261609_Seguranca_em_Redes_5G_Oportunidades_e_Desafios_em_Deteccao_de_Anomalias_e_Predicao_de_Trafego_Baseadas_em_Aprendizado_de_Maquina>. Citado na página 21.

BRADSKI, G. The OpenCV library. *Dr. Dobb's Journal of Software Tools*, v. 25, n. 11, p. 120–123, 2000. Disponível em: <https://elibrary.ru/item.asp?id=4934581>. Citado na página 23.

CAO, Q. et al. VGGFace2: A dataset for recognising faces across pose and age. In: *Proceedings of the IEEE International Conference on Automatic Face & Gesture Recognition*. [s.n.], 2018. p. 67–74. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8373813>. Citado na página 40.

CHAUDHRY, A.; ELGAZZAR, H. Design and implementation of a hybrid face recognition technique. In: IEEE. *2019 IEEE 9th Annual Computing and Communication*

Workshop and Conference (CCWC). 2019. p. 0384–0391. Disponível em: <https://ieeexplore.ieee.org/document/8666587>. Acesso em: 30 jul. 2023. Citado 2 vezes nas páginas 25 e 26.

COSTA, V. J. da. *Reconhecimento de padrões faciais: uma síntese*. 2019. Trabalho de Conclusão de Curso (Graduação) - Universidade Tecnológica Federal do Paraná, Santa Helena, Paraná, Brasil. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/11953>. Citado na página 25.

DENG, J. et al. RetinaFace: Single-shot multi-level face localisation in the wild. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [s.n.], 2020. p. 5203–5212. Disponível em: <https://openaccess.thecvf.com/content_CVPR_2020/html/Deng_RetinaFace_Single-Shot_Multi-Level_Face_Localisation_in_the_Wild_CVPR_2020_paper.html>. Citado na página 23.

DEWI, I. A.; MARYADI, N. A. T. Multiscale facial detection using retinaface architecture with loss function. *Journal of Computer Networks, Architecture and High Performance Computing*, v. 7, n. 3, p. 700–710, 2025. Disponível em: <https://jurnal.itscience.org/index.php/CNAPC/article/view/6161>. Acesso em: 14 out. 2025. Citado na página 23.

ELIAS, S. J. et al. Face recognition attendance system using local binary pattern (lbp). *Bulletin of Electrical Engineering and Informatics*, 2019. Acesso em: 01 maio 2024. Disponível em: <https://doi.org/10.11591/eei.v8i1.1439>. Citado na página 25.

FADEL, N. E. Facial recognition algorithms: A systematic literature review. *Journal of Imaging*, MDPI, v. 11, n. 2, p. 58, 2025. Disponível em: <https://www.mdpi.com/2313-433X/11/2/58>. Acesso em: 14 out. 2025. Citado na página 26.

FERREIRA, F. R. T.; PAGLIARI, C. L. Detecção e reconhecimento de faces distorcidas por artefatos de compressão. *Revista Militar de Ciência e Tecnologia*, v. 35, n. 2, p. 31–37, 2018. Acesso em: 01 maio 2024. Disponível em: <http://ebrevistas.eb.mil.br/CT/article/view/1957>. Citado na página 25.

FREITAS, E. J. de R.; SANTOS, S. S.; REZENDE, T. M. Redes neurais artificiais: Uma visão histórica. In: GUIMARãES, O. S. (Ed.). *Engenharia, Gestão e Inovação - Volume 1*. 1. ed. Belo Horizonte: Poisson, 2022. p. 123–134. ISBN 978-65-5866-159-7. Acesso em: 01 maio 2024. Citado na página 20.

Google LLC. *MediaPipe Face Detection — Model Card: BlazeFace*. 2023. <https://storage.googleapis.com/mediapipe-assets/MediaPipe%20BlazeFace%20Model%20Card%20(Short%20Range).pdf>. Acesso em: abr. 2025. Citado 2 vezes nas páginas 24 e 55.

GUO, Y. et al. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. *European Conference on Computer Vision*, p. 87–102, 2016. Disponível em: <https://link.springer.com/article/10.1007/s11042-018-6510-8>. Citado na página 24.

HUANG, G. B. et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. [S.l.], 2007. Citado 3 vezes nas páginas 22, 26 e 40.

IMRAN, A. et al. FaceEngine: A tracking-based framework for real-time face recognition in video surveillance system. *SN Computer Science*, Springer, v. 5, n. 609, 2024. Disponível em:

<https://link.springer.com/article/10.1007/s42979-024-02922-1>. Citado 3 vezes nas páginas 34, 35 e 63.

KANAGAMALLIGA, S. et al. Advancements in real-time face recognition algorithms for enhanced smart video surveillance. *Procedia Computer Science*, Elsevier, p. 486–492, 2024. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050923021099?via%3Dihub>. Citado 3 vezes nas páginas 33, 35 e 63.

KAUR, P. et al. Facial-recognition algorithms: A literature review. *Medicine, Science and the Law*, SAGE Publications Sage UK: London, England, v. 60, n. 2, p. 131–139, 2020. Disponível em: <https://doi.org/10.1177/0025802419893168>. Acesso em: 23 jul. 2023. Citado 3 vezes nas páginas 23, 25 e 26.

KING, D. E. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, v. 10, p. 1755–1758, 2009. Disponível em: <https://www.jmlr.org/papers/volume10/king09a/king09a.pdf>. Citado na página 23.

KORTLI, Y. et al. Face recognition systems: A survey. *Sensors*, MDPI, v. 20, n. 2, p. 342, 2020. Disponível em: <https://doi.org/10.3390/s20020342>. Acesso em: 14 out. 2025. Citado 2 vezes nas páginas 25 e 26.

LIU, W.; ZHOU, L.; CHEN, J. Face recognition based on lightweight convolutional neural networks. *Information*, 2021. Acesso em: 01 maio 2024. Disponível em: <https://www.mdpi.com/2078-2489/12/5/191>. Citado 2 vezes nas páginas 21 e 22.

LUGARESI, C. et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019. Disponível em: <https://arxiv.org/abs/1906.08172>. Acesso em: 10 ago. 2025. Citado 3 vezes nas páginas 23, 24 e 47.

MARTINO, J. M. D. et al. Differential 3d facial recognition: Adding 3d to your state-of-the-art 2d method. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 42, n. 7, p. 1582–1593, 2020. Disponível em: <https://doi.org/10.1109/tpami.2020.2986951>. Acesso em: 30 jul. 2024. Citado na página 26.

MIENYE, I. D.; SWART, T. G. A comprehensive review of deep learning: Architectures, recent advances, and applications. *Information*, v. 15, n. 12, p. 755, 2024. Citado 2 vezes nas páginas 19 e 20.

NEWMAN, S. *Building Microservices: Designing Fine-Grained Systems*. [S.l.]: O'Reilly Media, 2015. ISBN 978-1491950357. Citado 3 vezes nas páginas 27, 28 e 33.

PATEL, J. et al. Enhancing classroom attendance systems with face recognition through cctv using deep learning. *Procedia Computer Science*, Elsevier, v. 258, p. 3031–3041, 2025. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050925016655>. Acesso em: 23 jul. 2023. Citado 3 vezes nas páginas 34, 35 e 63.

PEREZ-MONTES, F. et al. Analysis of real-time face-verification methods for surveillance applications. *Journal of Imaging*, v. 9, n. 2, p. 21, 2023. Disponível em: <https://doi.org/10.3390/jimaging9020021>. Citado na página 31.

PHILLIPS, P. J. et al. The FERET evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, n. 10, p. 1090–1104, 2000. Acesso em: fev. 2026. Disponível em: <https://doi.org/10.1109/34.879790>. Citado na página 30.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de Software: Uma Abordagem Profissional*. 9. ed. Porto Alegre: AMGH, 2021. Citado na página 38.

RAINIO, O.; TEUHO, J.; KLÉN, R. Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, v. 14, n. 1, p. 6086, 2024. Disponível em: <https://www.nature.com/articles/s41598-024-56706-x>. Citado 2 vezes nas páginas 31 e 32.

RESENDE, C. A. de P.; PEREIRA, M. H. R. Visão computacional aplicada em reconhecimento facial na busca por pessoas desaparecidas. *Revista E-xacta*, Editora UniBH, v. 8, n. 2, p. 95–107, 2015. Acesso em: 01 maio 2024. Disponível em: <http://dx.doi.org/10.18674/exacta.v8i2.1661>. Citado na página 25.

ROSENBERG, A.; HIRSCHBERG, J. V-measure: A conditional entropy-based external cluster evaluation measure. In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. [s.n.], 2007. p. 410–420. Disponível em: <https://aclanthology.org/D07-1043/>. Acesso em: 10 ago. 2025. Citado na página 30.

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, Elsevier, v. 20, p. 53–65, 1987. Disponível em: <https://www.sciencedirect.com/science/article/pii/0377042787901257>. Acesso em: 30 jul. 2025. Citado na página 30.

SERENGIL, S.; ÖZPINAR, A. A benchmark of facial recognition pipelines and co-usability performances of modules. *Bilişim Teknolojileri Dergisi*, Gazi University, v. 17, n. 2, p. 95–107, 2024. Disponível em: <https://dergipark.org.tr/en/pub/gazibtd/article/1399077>. Acesso em: 14 out. 2025. Citado 4 vezes nas páginas 26, 27, 35 e 63.

SETIAWAN, R.; HERMAWAN, W.; SETIAWAN, A. T. Scalable microservices architecture for face recognition-based employee attendance systems. *Journal of Electrical, Electronic, Information, and Communication Technology*, v. 7, n. 2, p. 53–58, 2025. Disponível em: <https://jurnal.uns.ac.id/jeeict/article/view/108430>. Acesso em: 23 jul. 2023. Citado 2 vezes nas páginas 32 e 33.

SHARZIZI, A. F. et al. Evaluating accuracy latency and robustness of face recognition models for real-time web applications. *Journal of Informatics and Web Engineering*, v. 5, n. 1, p. 335–343, 2026. Disponível em: <https://journals.mmupress.com/index.php/jiwe/article/view/2476>. Citado 3 vezes nas páginas 33, 35 e 63.

SON, N. T. et al. Implementing cctv-based attendance taking support system using deep face recognition: A case study at fpt polytechnic college. *Symmetry*, MDPI, v. 12, n. 2, p. 307, 2020. Disponível em: <https://www.mdpi.com/2073-8994/12/2/307>. Acesso em: 30 jul. 2025. Citado 3 vezes nas páginas 34, 35 e 63.

SUMSION, A. et al. Surveying racial bias in facial recognition: Balancing datasets and algorithmic enhancements. *Electronics*, MDPI, v. 13, n. 12, p. 2317, 2024. Disponível em: <https://www.mdpi.com/2079-9292/13/12/2317>. Acesso em: 30 jul. 2025. Citado 2 vezes nas páginas 53 e 66.

SUNDARAM, M.; MANI, A. Face recognition: demystification of multifarious aspect in evaluation metrics. *Face Recognition-Semisupervised Classification, Subspace Projection and Evaluation Methods*, Intech, p. 75–92, 2016. Disponível em: <https:

//www.intechopen.com/chapters/50437>. Acesso em: 30 jul. 2025. Citado 2 vezes nas páginas 28 e 29.

TANENBAUM, A. S.; BOS, H. *Modern Operating Systems*. 4. ed. Upper Saddle River: Pearson, 2015. Citado na página 61.

VELEPUCHA, V.; FLORES, P. A survey on microservices architecture: Principles, patterns and migration challenges. *IEEE access*, IEEE, v. 11, p. 88339–88358, 2023. Disponível em: <https://ieeexplore.ieee.org/abstract/document/10220070>. Acesso em: 23 jul. 2023. Citado na página 32.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: IEEE. *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. 2001. v. 1, p. I–I. Disponível em: <https://ieeexplore.ieee.org/abstract/document/990517>. Acesso em: 30 jul. 2025. Citado na página 23.

WOHLIN, C. et al. *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer, 2012. ISBN 978-3-642-29044-2. Disponível em: <https://doi.org/10.1007/978-3-642-29044-2>. Citado 3 vezes nas páginas 36, 37 e 39.

WOLF, L.; HASSNER, T.; MAOZ, I. Face recognition in unconstrained videos with matched background similarity. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2011. p. 529–534. Citado na página 22.

YAAKUB, N. D. A. et al. Class attendance system using viola-jones algorithm and principal component analysis. In: *Journal of Physics: Conference Series*. [s.n.], 2021. Acesso em: 01 maio 2024. Disponível em: <https://doi.org/10.1088/1742-6596/1962/1/012053>. Citado na página 25.

ZHANG, W. et al. Improving shadow suppression for illumination robust face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 41, n. 3, p. 611–624, 2019. Acesso em: 01 maio 2024. Disponível em: <https://doi.org/10.1109/TPAMI.2018.2803179>. Citado na página 25.

Apêndices

APÊNDICE A – RESULTADOS DETALHADOS

Este apêndice reúne os resultados detalhados dos experimentos: os resultados individuais de cada um dos 30 vídeos do conjunto de teste, organizados por arquitetura e por grupo de métricas, e os gráficos comparativos consolidados. As médias correspondentes são discutidas no Capítulo 4.

A.1 Arquitetura Ingênua

Tabela 13 – Métricas de classificação — Arquitetura Ingênua

Vídeo	**TP**	**TN**	**FP**	**FN**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
A01-ENTRADA	0	0	0	4	0,0000	–	0,0000	–
A02-ENTRADA	0	0	0	2	0,0000	–	0,0000	–
A03-ENTRADA	0	0	0	2	0,0000	–	0,0000	–
A04-ENTRADA	0	0	0	2	0,0000	–	0,0000	–
A05-ENTRADA	0	0	0	5	0,0000	–	0,0000	–
A06-ENTRADA	1	0	0	1	0,5000	1,0000	0,5000	0,6667
A07-ENTRADA	1	0	0	4	0,2000	1,0000	0,2000	0,3333
A08-ENTRADA	0	0	0	1	0,0000	–	0,0000	–
A09-ENTRADA	0	0	0	7	0,0000	–	0,0000	–
A10-ENTRADA	7	0	0	51	0,1207	1,0000	0,1207	0,2154
A11-ENTRADA	1	0	0	7	0,1250	1,0000	0,1250	0,2222
A12-ENTRADA	9	0	0	53	0,1452	1,0000	0,1452	0,2535
A13-ENTRADA	4	0	0	48	0,0769	1,0000	0,0769	0,1429
A14-ENTRADA	1	0	0	36	0,0270	1,0000	0,0270	0,0526
A15-ENTRADA	0	0	0	2	0,0000	–	0,0000	–
A16-ENTRADA	0	0	0	1	0,0000	–	0,0000	–
A17-ENTRADA	0	0	0	1	0,0000	–	0,0000	–
A18-ENTRADA	0	0	0	4	0,0000	–	0,0000	–
A19-ENTRADA	0	0	0	5	0,0000	–	0,0000	–
A20-ENTRADA	0	0	0	9	0,0000	–	0,0000	–
A21-ENTRADA	0	0	0	3	0,0000	–	0,0000	–
A22-ENTRADA	0	0	0	2	0,0000	–	0,0000	–
A23-ENTRADA	0	0	0	2	0,0000	–	0,0000	–
A24-ENTRADA	0	0	0	5	0,0000	–	0,0000	–
A25-ENTRADA	0	0	0	2	0,0000	–	0,0000	–
A26-ENTRADA	0	0	0	2	0,0000	–	0,0000	–
A27-ENTRADA	0	0	0	3	0,0000	–	0,0000	–
A28-ENTRADA	0	0	0	7	0,0000	–	0,0000	–
A29-ENTRADA	0	0	0	2	0,0000	–	0,0000	–
A30-ENTRADA	2	0	0	12	0,1429	1,0000	0,1429	0,2500
Média	26	0	0	9,5	0,0446	1,0000*	0,0446	0,2671*

* As médias de *Precision* e *F1-Score* foram calculadas apenas sobre os vídeos com valores definidos. Como $FP = 0$ em todos os 30 vídeos, ambas as métricas tornam-se indefinidas sempre que $TP = 0$, situação verificada em 22 vídeos (A01–A05, A08–A09, A15–A29). As médias de *Precision* e *F1-Score* foram portanto apuradas sobre os 8 vídeos com $TP > 0$ (A06, A07, A10, A11, A12, A13, A14 e A30). Fonte: Elaboração própria

A.2 Arquitetura Monolítica

Tabela 14 – Métricas de classificação — Arquitetura Monolítica

Vídeo	**TP**	**TN**	**FP**	**FN**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
A01-ENTRADA	1	0	0	3	0,2500	1,0000	0,2500	0,4000
A02-ENTRADA	1	0	4	1	0,1667	0,2000	0,5000	0,2857
A03-ENTRADA	0	0	0	20	0,0000	–	0,0000	–
A04-ENTRADA	0	0	10	10	0,0000	0,0000	0,0000	–
A05-ENTRADA	0	0	4	5	0,0000	0,0000	0,0000	–
A06-ENTRADA	1	0	0	1	0,5000	1,0000	0,5000	0,6667
A07-ENTRADA	0	0	13	5	0,0000	0,0000	0,0000	–
A08-ENTRADA	0	0	44	10	0,0000	0,0000	0,0000	–
A09-ENTRADA	1	0	1	6	0,1250	0,5000	0,1429	0,2222
A10-ENTRADA	2	0	15	56	0,0274	0,1176	0,0345	0,0533
A11-ENTRADA	2	0	0	6	0,2500	1,0000	0,2500	0,4000
A12-ENTRADA	4	0	0	58	0,0645	1,0000	0,0645	0,1212
A13-ENTRADA	2	0	3	50	0,0364	0,4000	0,0385	0,0702
A14-ENTRADA	1	0	0	36	0,0270	1,0000	0,0270	0,0526
A15-ENTRADA	0	0	3	20	0,0000	0,0000	0,0000	–
A16-ENTRADA	0	0	13	1	0,0000	0,0000	0,0000	–
A17-ENTRADA	0	0	8	1	0,0000	0,0000	0,0000	–
A18-ENTRADA	1	0	0	3	0,2500	1,0000	0,2500	0,4000
A19-ENTRADA	0	0	10	5	0,0000	0,0000	0,0000	–
A20-ENTRADA	1	0	0	8	0,1111	1,0000	0,1111	0,2000
A21-ENTRADA	1	0	0	2	0,3333	1,0000	0,3333	0,5000
A22-ENTRADA	1	0	0	1	0,5000	1,0000	0,5000	0,6667
A23-ENTRADA	2	0	5	0	0,2857	0,2857	1,0000	0,4444
A24-ENTRADA	1	0	12	4	0,0588	0,0769	0,2000	0,1111
A25-ENTRADA	1	0	0	1	0,5000	1,0000	0,5000	0,6667
A26-ENTRADA	0	0	3	2	0,0000	0,0000	0,0000	–
A27-ENTRADA	1	0	6	2	0,1111	0,1429	0,3333	0,2000
A28-ENTRADA	3	0	1	4	0,3750	0,7500	0,4286	0,5455
A29-ENTRADA	0	0	0	20	0,0000	–	0,0000	–
A30-ENTRADA	1	0	4	13	0,0556	0,2000	0,0714	0,1053
Média	0,93	0	5,3	9,43	0,1300	0,4224*	0,1800	0,3216*

* As médias de *Precision* e *F1-Score* foram calculadas apenas sobre os vídeos com valores definidos. *Precision* é indefinida nos vídeos A03 e A29 (onde $TP = FP = 0$), sendo sua média apurada sobre os 28 vídeos restantes. *F1-Score* é indefinido nos vídeos em que $Precision + Recall = 0$ (A03, A04, A05, A07, A08, A15, A16, A17, A19, A26 e A29), sendo sua média calculada sobre os 19 vídeos com valor definido. Fonte: Elaboração própria

A.3 Arquitetura com Microsserviços

Tabela 15 – Métricas de classificação — Arquitetura com Microsserviços

Vídeo	**TP**	**TN**	**FP**	**FN**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
A01-ENTRADA	27	0	0	0	1,0000	1,0000	1,0000	1,0000
A02-ENTRADA	2	0	0	0	1,0000	1,0000	1,0000	1,0000
A03-ENTRADA	24	1	1	0	0,9620	0,9600	1,0000	0,9800
A04-ENTRADA	11	0	0	0	1,0000	1,0000	1,0000	1,0000
A05-ENTRADA	15	0	0	0	1,0000	1,0000	1,0000	1,0000
A06-ENTRADA	9	0	0	0	1,0000	1,0000	1,0000	1,0000
A07-ENTRADA	20	0	0	2	0,9090	1,0000	0,9090	0,9520
A08-ENTRADA	8	0	0	0	1,0000	1,0000	1,0000	1,0000
A09-ENTRADA	12	0	0	3	0,8000	1,0000	0,8000	0,8890
A10-ENTRADA	191	2	0	16	0,9230	1,0000	0,9230	0,9600
A11-ENTRADA	22	0	2	3	0,8150	0,9170	0,8800	0,8980
A12-ENTRADA	248	0	0	16	0,9390	1,0000	0,9390	0,9690
A13-ENTRADA	232	0	0	10	0,9590	1,0000	0,9590	0,9790
A14-ENTRADA	130	0	0	11	0,9220	1,0000	0,9220	0,9590
A15-ENTRADA	11	0	0	0	1,0000	1,0000	1,0000	1,0000
A16-ENTRADA	8	0	1	0	0,8890	0,8890	1,0000	0,9410
A17-ENTRADA	6	0	0	0	1,0000	1,0000	1,0000	1,0000
A18-ENTRADA	14	0	0	0	1,0000	1,0000	1,0000	1,0000
A19-ENTRADA	7	0	0	2	0,7780	1,0000	0,7780	0,8750
A20-ENTRADA	31	0	0	2	0,9390	1,0000	0,9390	0,9690
A21-ENTRADA	15	0	0	1	0,9380	1,0000	0,9380	0,9680
A22-ENTRADA	4	0	0	1	0,8000	1,0000	0,8000	0,8890
A23-ENTRADA	1	0	0	1	0,5000	1,0000	0,5000	0,6670
A24-ENTRADA	15	0	0	1	0,9380	1,0000	0,9380	0,9680
A25-ENTRADA	13	0	0	0	1,0000	1,0000	1,0000	1,0000
A26-ENTRADA	14	0	0	0	1,0000	1,0000	1,0000	1,0000
A27-ENTRADA	14	0	0	0	1,0000	1,0000	1,0000	1,0000
A28-ENTRADA	10	0	0	3	0,7690	1,0000	0,7690	0,8700
A29-ENTRADA	5	0	0	1	0,8330	1,0000	0,8330	0,9090
A30-ENTRADA	56	0	1	2	0,9490	0,9820	0,9660	0,9740
Média	39,20	0,10	0,17	2,50	0,9187	0,9916	0,9264	0,9539

Fonte: Elaboração própria.

Tabela 16 – Métricas de agrupamento — Arquitetura com Microsserviços

Vídeo	**Cover.**	**Inter.**	**Intra.**	**Silh.**	**Homog.**	**Compl.**	**V-Meas.**
A01-ENTRADA	1,000	23,513	6,082	0,400	1,000	0,866	0,928
A02-ENTRADA	1,000	30,338	0,000	–	1,000	1,000	1,000
A03-ENTRADA	1,000	24,925	4,999	0,319	1,000	0,616	0,762
A04-ENTRADA	1,000	29,093	11,040	0,472	1,000	1,000	1,000
A05-ENTRADA	1,000	24,379	2,450	0,098	1,000	0,679	0,809
A06-ENTRADA	1,000	26,806	5,047	0,273	1,000	0,565	0,722
A07-ENTRADA	0,600	25,583	3,483	0,220	1,000	0,691	0,818
A08-ENTRADA	1,000	27,137	5,142	0,400	1,000	0,000	0,000
A09-ENTRADA	0,571	27,105	3,262	0,369	1,000	0,906	0,951
A10-ENTRADA	0,729	26,829	4,826	0,272	0,982	0,879	0,928
A11-ENTRADA	0,625	27,247	4,378	0,265	1,000	0,683	0,811
A12-ENTRADA	0,746	27,701	4,397	0,262	0,978	0,914	0,945
A13-ENTRADA	0,811	27,778	5,673	0,300	0,985	0,925	0,954
A14-ENTRADA	0,703	27,447	4,950	0,277	0,996	0,878	0,934
A15-ENTRADA	1,000	34,939	8,842	0,632	1,000	1,000	1,000
A16-ENTRADA	1,000	22,767	5,315	0,356	1,000	0,000	0,000
A17-ENTRADA	1,000	0,000	10,808	–	1,000	1,000	1,000
A18-ENTRADA	1,000	25,334	5,226	0,311	1,000	0,888	0,941
A19-ENTRADA	0,600	25,323	4,904	0,418	1,000	1,000	1,000
A20-ENTRADA	0,778	27,244	8,158	0,298	1,000	0,900	0,947
A21-ENTRADA	0,667	30,422	6,980	0,372	1,000	0,722	0,838
A22-ENTRADA	0,500	0,000	10,745	–	1,000	1,000	1,000
A23-ENTRADA	0,500	0,000	0,000	–	0,000	0,000	0,000
A24-ENTRADA	0,800	26,293	4,627	0,205	0,899	0,582	0,706
A25-ENTRADA	1,000	26,919	5,285	0,426	1,000	1,000	1,000
A26-ENTRADA	1,000	26,958	10,312	0,496	1,000	1,000	1,000
A27-ENTRADA	1,000	27,557	2,798	0,309	1,000	0,571	0,727
A28-ENTRADA	0,571	27,290	5,717	0,419	1,000	1,000	1,000
A29-ENTRADA	0,500	0,000	10,391	–	1,000	1,000	1,000
A30-ENTRADA	0,857	27,050	3,995	0,121	0,978	0,724	0,832
Média	0,819	23,466	5,661	0,332*	0,961	0,766	0,818

Cover.: Covering; Inter.: Inter-Distância; Intra.: Intra-Distância; Silh.: Silhouette; Homog.: Homogeneity; Compl.: Completeness; V-Meas.: V-Measure. * *Silhouette* indefinida nos vídeos A02, A17, A22, A23 e A29, onde apenas um cluster foi gerado ou todas as faces pertencem ao mesmo grupo. Média calculada sobre os 25 vídeos com valor definido. Fonte: Elaboração própria.

Tabela 17 – Métricas operacionais — Arquitetura com Microsserviços

Vídeo	**Tempo / Duração do Vídeo**	**Tamanho BD Auxiliar (MB)**
A01-ENTRADA	0,192	0,33
A02-ENTRADA	0,158	0,16
A03-ENTRADA	0,535	0,34
A04-ENTRADA	0,355	0,22
A05-ENTRADA	0,204	0,25
A06-ENTRADA	0,420	0,21
A07-ENTRADA	0,331	0,29
A08-ENTRADA	0,105	0,20
A09-ENTRADA	1,649	0,23
A10-ENTRADA	1,623	1,51
A11-ENTRADA	1,031	0,31
A12-ENTRADA	2,155	1,90
A13-ENTRADA	1,593	1,79
A14-ENTRADA	1,830	1,06
A15-ENTRADA	0,369	0,22
A16-ENTRADA	0,301	0,21
A17-ENTRADA	0,030	0,18
A18-ENTRADA	0,356	0,24
A19-ENTRADA	1,080	0,19
A20-ENTRADA	1,704	0,36
A21-ENTRADA	1,403	0,25
A22-ENTRADA	0,028	0,17
A23-ENTRADA	0,000	0,15
A24-ENTRADA	0,326	0,25
A25-ENTRADA	0,212	0,24
A26-ENTRADA	0,206	0,25
A27-ENTRADA	1,101	0,24
A28-ENTRADA	0,435	0,22
A29-ENTRADA	0,043	0,18
A30-ENTRADA	0,869	0,55
Média	0,688	0,42

Fonte: Elaboração própria.

A.4 Arquitetura com Cache

Tabela 18 – Métricas de classificação — Arquitetura com Cache

Vídeo	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-Score
A01-ENTRADA	27	0	0	0	1,0000	1,0000	1,0000	1,0000
A02-ENTRADA	2	0	0	0	1,0000	1,0000	1,0000	1,0000
A03-ENTRADA	27	0	1	0	0,9640	0,9640	1,0000	0,9820
A04-ENTRADA	11	0	0	0	1,0000	1,0000	1,0000	1,0000
A05-ENTRADA	15	0	0	0	1,0000	1,0000	1,0000	1,0000
A06-ENTRADA	9	0	0	0	1,0000	1,0000	1,0000	1,0000
A07-ENTRADA	20	0	0	2	0,9090	1,0000	0,9090	0,9520
A08-ENTRADA	8	0	0	0	1,0000	1,0000	1,0000	1,0000
A09-ENTRADA	12	0	0	3	0,8000	1,0000	0,8000	0,8890
A10-ENTRADA	193	0	0	16	0,9230	1,0000	0,9230	0,9600
A11-ENTRADA	22	0	2	3	0,8150	0,9170	0,8800	0,8980
A12-ENTRADA	248	0	0	16	0,9390	1,0000	0,9390	0,9690
A13-ENTRADA	232	0	0	9	0,9630	1,0000	0,9630	0,9810
A14-ENTRADA	130	0	0	11	0,9220	1,0000	0,9220	0,9590
A15-ENTRADA	11	0	0	0	1,0000	1,0000	1,0000	1,0000
A16-ENTRADA	8	0	1	0	0,8890	0,8890	1,0000	0,9410
A17-ENTRADA	6	0	0	0	1,0000	1,0000	1,0000	1,0000
A18-ENTRADA	14	0	0	0	1,0000	1,0000	1,0000	1,0000
A19-ENTRADA	7	0	0	2	0,7780	1,0000	0,7780	0,8750
A20-ENTRADA	31	0	0	2	0,9390	1,0000	0,9390	0,9690
A21-ENTRADA	15	0	0	1	0,9380	1,0000	0,9380	0,9680
A22-ENTRADA	4	0	0	1	0,8000	1,0000	0,8000	0,8890
A23-ENTRADA	1	0	0	1	0,5000	1,0000	0,5000	0,6670
A24-ENTRADA	15	0	0	2	0,8820	1,0000	0,8820	0,9380
A25-ENTRADA	13	0	0	0	1,0000	1,0000	1,0000	1,0000
A26-ENTRADA	14	0	0	0	1,0000	1,0000	1,0000	1,0000
A27-ENTRADA	14	0	0	0	1,0000	1,0000	1,0000	1,0000
A28-ENTRADA	7	0	0	3	0,7000	1,0000	0,7000	0,8240
A29-ENTRADA	5	0	0	1	0,8330	1,0000	0,8330	0,9090
A30-ENTRADA	56	0	1	2	0,9490	0,9820	0,9660	0,9740
Média	39,23	0,00	0,17	2,50	0,9148	0,9917	0,9224	0,9515

Fonte: Elaboração própria.

Tabela 19 – Métricas de agrupamento — Arquitetura com Cache

Vídeo	**Cover.**	**Inter.**	**Intra.**	**Silh.**	**Homog.**	**Compl.**	**V-Meas.**
A01-ENTRADA	1,000	23,452	5,589	0,465	1,000	0,850	0,919
A02-ENTRADA	1,000	30,338	0,000	–	1,000	1,000	1,000
A03-ENTRADA	1,000	24,925	4,999	0,319	1,000	0,616	0,762
A04-ENTRADA	1,000	29,093	11,040	0,472	1,000	1,000	1,000
A05-ENTRADA	1,000	23,472	3,548	0,083	1,000	0,642	0,782
A06-ENTRADA	1,000	26,806	5,047	0,273	1,000	0,565	0,722
A07-ENTRADA	0,600	25,583	3,483	0,220	1,000	0,691	0,818
A08-ENTRADA	1,000	27,137	5,142	0,400	1,000	0,000	0,000
A09-ENTRADA	0,571	27,105	3,262	0,369	1,000	0,906	0,951
A10-ENTRADA	0,729	26,769	4,862	0,256	0,989	0,892	0,938
A11-ENTRADA	0,625	27,476	3,755	0,338	1,000	0,725	0,841
A12-ENTRADA	0,746	27,777	4,373	0,308	0,988	0,916	0,951
A13-ENTRADA	0,827	27,897	5,393	0,331	0,993	0,924	0,957
A14-ENTRADA	0,703	27,352	4,510	0,260	1,000	0,880	0,936
A15-ENTRADA	1,000	34,939	8,842	0,632	1,000	1,000	1,000
A16-ENTRADA	1,000	22,767	5,315	0,356	1,000	0,000	0,000
A17-ENTRADA	1,000	0,000	10,808	–	1,000	1,000	1,000
A18-ENTRADA	1,000	25,334	5,226	0,311	1,000	0,888	0,941
A19-ENTRADA	0,600	25,323	4,904	0,418	1,000	1,000	1,000
A20-ENTRADA	0,778	27,465	8,077	0,348	1,000	0,891	0,943
A21-ENTRADA	0,667	30,422	6,980	0,372	1,000	0,722	0,838
A22-ENTRADA	0,500	0,000	10,745	–	1,000	1,000	1,000
A23-ENTRADA	0,500	0,000	0,000	–	0,000	0,000	0,000
A24-ENTRADA	0,600	26,293	4,627	0,205	0,884	0,496	0,636
A25-ENTRADA	1,000	26,919	5,285	0,426	1,000	1,000	1,000
A26-ENTRADA	1,000	26,958	10,312	0,496	1,000	1,000	1,000
A27-ENTRADA	1,000	27,557	2,798	0,309	1,000	0,571	0,727
A28-ENTRADA	0,571	27,290	5,717	0,419	1,000	1,000	1,000
A29-ENTRADA	0,500	0,000	10,391	–	1,000	1,000	1,000
A30-ENTRADA	0,857	27,265	3,577	0,185	0,974	0,738	0,840
Média	0,812	23,457	5,620	0,343*	0,961	0,764	0,817

Cover.: Covering; Inter.: Inter-Distância; Intra.: Intra-Distância; Silh.: Silhouette; Homog.: Homogeneity; Compl.: Completeness; V-Meas.: V-Measure. * *Silhouette* indefinida nos vídeos A02, A17, A22, A23 e A29, onde apenas um cluster foi gerado ou todas as faces pertencem ao mesmo grupo. Média calculada sobre os 25 vídeos com valor definido. Fonte: Elaboração própria.

Tabela 20 – Métricas operacionais — Arquitetura com Cache

Vídeo	**Tempo / Duração do Vídeo**	**Tamanho BD Auxiliar (MB)**
A01-ENTRADA	0,320	0,34
A02-ENTRADA	0,143	0,16
A03-ENTRADA	0,505	0,34
A04-ENTRADA	0,351	0,22
A05-ENTRADA	0,190	0,25
A06-ENTRADA	0,412	0,21
A07-ENTRADA	0,346	0,29
A08-ENTRADA	0,095	0,20
A09-ENTRADA	1,688	0,23
A10-ENTRADA	1,669	1,51
A11-ENTRADA	1,059	0,31
A12-ENTRADA	2,114	1,91
A13-ENTRADA	1,518	1,79
A14-ENTRADA	1,925	1,07
A15-ENTRADA	0,345	0,22
A16-ENTRADA	0,314	0,21
A17-ENTRADA	0,030	0,18
A18-ENTRADA	0,346	0,24
A19-ENTRADA	1,012	0,20
A20-ENTRADA	1,602	0,36
A21-ENTRADA	1,360	0,25
A22-ENTRADA	0,028	0,17
A23-ENTRADA	0,000	0,15
A24-ENTRADA	0,322	0,25
A25-ENTRADA	0,193	0,24
A26-ENTRADA	0,207	0,25
A27-ENTRADA	1,052	0,24
A28-ENTRADA	0,406	0,22
A29-ENTRADA	0,038	0,18
A30-ENTRADA	0,793	0,55
Média	0,679	0,42

Fonte: Elaboração própria.

A.5 Arquitetura com Fragmentação

Tabela 21 – Métricas de classificação — Arquitetura com Fragmentação

Vídeo	**TP**	**TN**	**FP**	**FN**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
A01-ENTRADA	26	1	0	0	1,0000	1,0000	1,0000	1,0000
A02-ENTRADA	2	0	0	0	1,0000	1,0000	1,0000	1,0000
A03-ENTRADA	27	0	1	0	0,9640	0,9640	1,0000	0,9820
A04-ENTRADA	11	0	0	0	1,0000	1,0000	1,0000	1,0000
A05-ENTRADA	15	0	0	0	1,0000	1,0000	1,0000	1,0000
A06-ENTRADA	9	0	0	0	1,0000	1,0000	1,0000	1,0000
A07-ENTRADA	20	0	0	2	0,9090	1,0000	0,9090	0,9520
A08-ENTRADA	8	0	0	0	1,0000	1,0000	1,0000	1,0000
A09-ENTRADA	12	0	0	3	0,8000	1,0000	0,8000	0,8890
A10-ENTRADA	193	0	0	16	0,9230	1,0000	0,9230	0,9600
A11-ENTRADA	22	0	2	3	0,8150	0,9170	0,8800	0,8980
A12-ENTRADA	248	0	0	15	0,9430	1,0000	0,9430	0,9710
A13-ENTRADA	231	1	0	11	0,9550	1,0000	0,9550	0,9770
A14-ENTRADA	130	0	0	10	0,9290	1,0000	0,9290	0,9630
A15-ENTRADA	11	0	0	0	1,0000	1,0000	1,0000	1,0000
A16-ENTRADA	9	0	0	0	1,0000	1,0000	1,0000	1,0000
A17-ENTRADA	6	0	0	0	1,0000	1,0000	1,0000	1,0000
A18-ENTRADA	14	0	0	0	1,0000	1,0000	1,0000	1,0000
A19-ENTRADA	7	0	0	2	0,7780	1,0000	0,7780	0,8750
A20-ENTRADA	31	0	0	2	0,9390	1,0000	0,9390	0,9690
A21-ENTRADA	15	0	0	1	0,9380	1,0000	0,9380	0,9680
A22-ENTRADA	4	0	0	1	0,8000	1,0000	0,8000	0,8890
A23-ENTRADA	1	0	0	1	0,5000	1,0000	0,5000	0,6670
A24-ENTRADA	15	0	0	2	0,8820	1,0000	0,8820	0,9380
A25-ENTRADA	13	0	0	0	1,0000	1,0000	1,0000	1,0000
A26-ENTRADA	14	0	0	0	1,0000	1,0000	1,0000	1,0000
A27-ENTRADA	14	0	0	0	1,0000	1,0000	1,0000	1,0000
A28-ENTRADA	10	0	0	3	0,7690	1,0000	0,7690	0,8700
A29-ENTRADA	5	0	0	1	0,8330	1,0000	0,8330	0,9090
A30-ENTRADA	57	0	0	2	0,9660	1,0000	0,9660	0,9830
Média	39,33	0,07	0,10	2,50	0,9214	0,9960	0,9248	0,9553

Fonte: Elaboração própria.

Tabela 22 – Métricas de agrupamento — Arquitetura com Fragmentação

Vídeo	**Cover.**	**Inter.**	**Intra.**	**Silh.**	**Homog.**	**Compl.**	**V-Meas.**
A01-ENTRADA	1,000	21,061	8,735	−0,010	1,000	0,570	0,726
A02-ENTRADA	1,000	28,314	4,107	0,295	1,000	1,000	1,000
A03-ENTRADA	1,000	25,118	2,798	0,029	1,000	0,274	0,430
A04-ENTRADA	1,000	25,267	4,844	−0,052	1,000	0,373	0,544
A05-ENTRADA	1,000	23,826	2,917	0,039	1,000	0,558	0,716
A06-ENTRADA	1,000	28,298	1,413	−0,017	1,000	0,336	0,503
A07-ENTRADA	0,600	25,181	3,086	−0,044	1,000	0,346	0,514
A08-ENTRADA	1,000	23,725	5,328	0,081	1,000	0,000	0,000
A09-ENTRADA	0,571	26,324	1,338	−0,001	1,000	0,704	0,826
A10-ENTRADA	0,729	27,428	2,458	0,001	1,000	0,770	0,870
A11-ENTRADA	0,625	26,681	1,167	0,066	1,000	0,534	0,696
A12-ENTRADA	0,762	28,359	2,924	0,045	1,000	0,803	0,891
A13-ENTRADA	0,788	28,447	2,952	0,018	0,989	0,770	0,866
A14-ENTRADA	0,730	28,523	2,580	0,065	0,986	0,759	0,857
A15-ENTRADA	1,000	14,842	3,842	0,151	1,000	0,446	0,617
A16-ENTRADA	1,000	22,132	4,135	0,124	1,000	0,000	0,000
A17-ENTRADA	1,000	8,662	10,385	−0,066	1,000	0,000	0,000
A18-ENTRADA	1,000	23,719	3,656	0,192	1,000	0,667	0,800
A19-ENTRADA	0,600	23,496	3,460	0,169	1,000	0,748	0,856
A20-ENTRADA	0,778	27,504	5,133	0,098	1,000	0,709	0,830
A21-ENTRADA	0,667	24,630	6,442	0,002	1,000	0,305	0,468
A22-ENTRADA	0,500	0,000	0,000	–	1,000	0,000	0,000
A23-ENTRADA	0,500	0,000	0,000	–	0,000	0,000	0,000
A24-ENTRADA	0,600	27,486	2,173	0,201	1,000	0,452	0,623
A25-ENTRADA	1,000	0,000	0,000	–	1,000	0,220	0,361
A26-ENTRADA	1,000	23,323	3,858	−0,019	1,000	0,296	0,456
A27-ENTRADA	1,000	25,943	0,000	–	1,000	0,446	0,617
A28-ENTRADA	0,571	24,202	1,516	−0,009	1,000	0,665	0,799
A29-ENTRADA	0,500	0,000	6,222	–	1,000	0,000	0,000
A30-ENTRADA	0,857	27,869	2,054	0,034	1,000	0,642	0,782
Média	0,813	21,345	3,317	0,056*	0,966	0,446	0,555

Cover.: Covering; Inter.: Inter-Distância; Intra.: Intra-Distância; Silh.: Silhouette; Homog.: Homogeneity; Compl.: Completeness; V-Meas.: V-Measure. * *Silhouette* indefinida nos vídeos A22, A23, A25, A27 e A29, onde apenas um cluster foi gerado ou todas as faces pertencem ao mesmo grupo. Média calculada sobre os 25 vídeos com valor definido. Fonte: Elaboração própria.

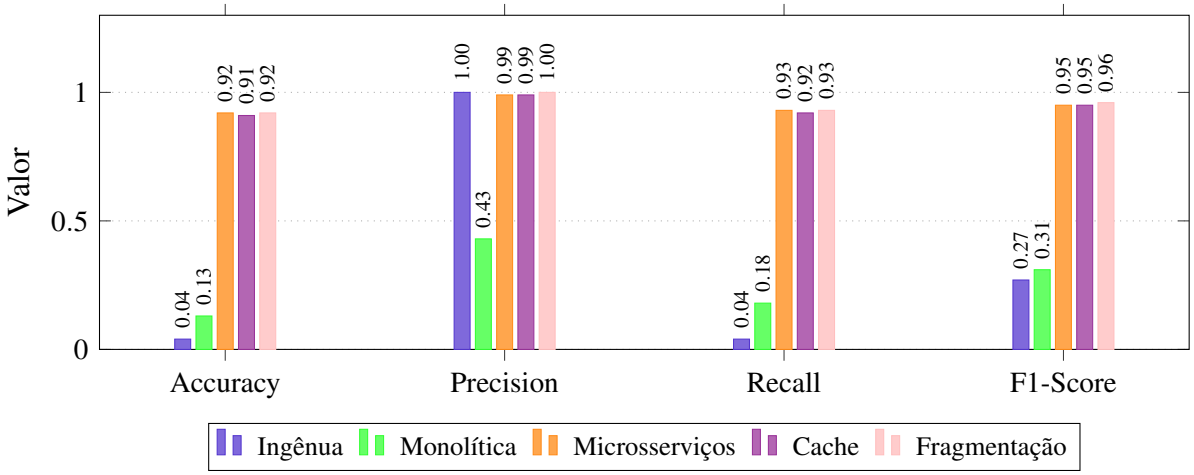
Tabela 23 – Métricas operacionais — Arquitetura com Fragmentação

Vídeo	**Tempo / Duração do Vídeo**	**Tamanho BD Auxiliar (MB)**
A01-ENTRADA	0,490	0,60
A02-ENTRADA	0,162	0,16
A03-ENTRADA	0,668	0,28
A04-ENTRADA	0,409	0,24
A05-ENTRADA	0,276	0,27
A06-ENTRADA	0,476	0,20
A07-ENTRADA	0,392	0,29
A08-ENTRADA	0,157	0,18
A09-ENTRADA	1,743	0,22
A10-ENTRADA	2,775	1,76
A11-ENTRADA	1,028	0,31
A12-ENTRADA	3,527	1,67
A13-ENTRADA	3,578	1,66
A14-ENTRADA	2,776	0,86
A15-ENTRADA	0,451	0,18
A16-ENTRADA	0,355	0,21
A17-ENTRADA	0,072	0,20
A18-ENTRADA	0,388	0,24
A19-ENTRADA	1,053	0,19
A20-ENTRADA	1,645	0,35
A21-ENTRADA	1,402	0,25
A22-ENTRADA	0,042	0,14
A23-ENTRADA	0,000	0,15
A24-ENTRADA	0,363	0,23
A25-ENTRADA	0,225	0,15
A26-ENTRADA	0,294	0,24
A27-ENTRADA	1,109	0,19
A28-ENTRADA	0,455	0,20
A29-ENTRADA	0,058	0,16
A30-ENTRADA	0,954	0,50
Média	0,911	0,41

Fonte: Elaboração própria.

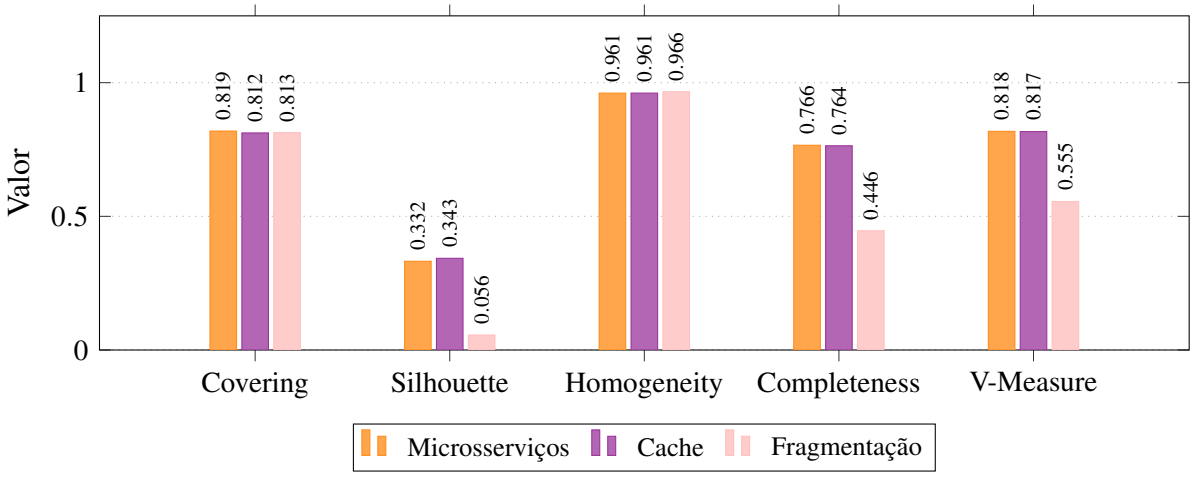
A.6 Gráficos comparativos

Figura 10 – Comparação das métricas de classificação por arquitetura



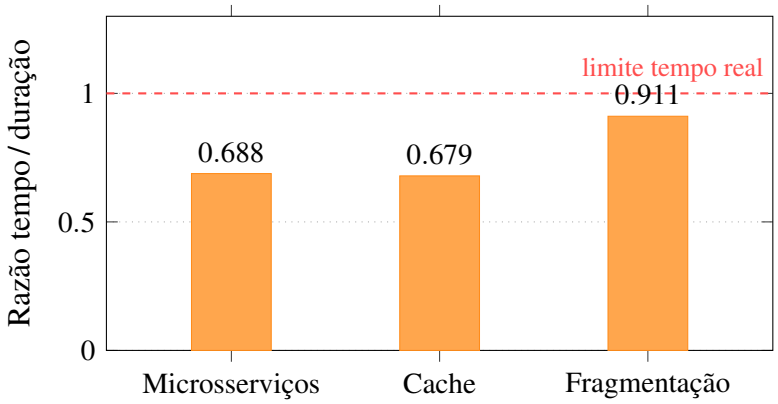
Fonte: Elaboração própria.

Figura 11 – Comparação das métricas de agrupamento por arquitetura



Fonte: Elaboração própria.

Figura 12 – Comparação da razão tempo de processamento / duração do vídeo por arquitetura



Nota: a linha tracejada indica o limite do tempo real (razão = 1,00). Fonte: Elaboração própria.

APÊNDICE B – SCRIPT DE ANÁLISE ESTATÍSTICA E SAÍDA DE EXECUÇÃO

Este apêndice apresenta o script em Python, baseado na biblioteca SciPy, utilizado para os testes de hipóteses descritos na Seção 3.1.7. Os dados de entrada correspondem aos valores por vídeo reportados no Apêndice A. O Código B.1 reproduz a implementação; em seguida, apresenta-se a saída obtida na sua execução, cujos valores de W e p coincidem com os da Tabela 10.

```

# -*- coding: utf-8 -*-
"""
Testes de hipóteses da dissertação.

Comparações planejadas (pareadas, mesmos 30 vídeos), uma por hipótese,
via teste de Wilcoxon dos postos sinalizados (não-paramétrico):

H1  Monolítica x Microsserviços -> métrica: Acurácia
H2  Microsserviços x Cache      -> métrica: Razão tempo/duração
H3  Cache x Fragmentação        -> métrica: V-Measure

Como são 3 testes confirmatórios, aplica-se correção de Bonferroni:
limiar de significância = 0,05 / 3 = 0,0167.

Dependência: scipy
"""

from scipy.stats import wilcoxon
from statistics import median

VIDEOS = [f"A{i:02d}" for i in range(1, 31)]

# ---------------------------------------------------------------------
# DADOS (extraídos das tabelas por vídeo do Capítulo de Resultados)
# ---------------------------------------------------------------------

# --- Acurácia ---
acc_mono = [0.2500,0.1667,0.0000,0.0000,0.0000,0.5000,0.0000,0.0000,0.1250,0.0274,
0.2500,0.0645,0.0364,0.0270,0.0000,0.0000,0.0000,0.2500,0.0000,0.1111,
0.3333,0.5000,0.2857,0.0588,0.5000,0.0000,0.1111,0.3750,0.0000,0.0556]

acc_micro = [1.0000,1.0000,0.9620,1.0000,1.0000,1.0000,0.9090,1.0000,0.8000,0.9230,
0.8150,0.9390,0.9590,0.9220,1.0000,0.8890,1.0000,1.0000,0.7780,0.9390,
```

```
0.9380,0.8000,0.5000,0.9380,1.0000,1.0000,1.0000,0.7690,0.8330,0.9490]

acc_cache = [1.0000,1.0000,0.9640,1.0000,1.0000,1.0000,0.9090,1.0000,0.8000,0.9230,
0.8150,0.9390,0.9630,0.9220,1.0000,0.8890,1.0000,1.0000,0.7780,0.9390,
0.9380,0.8000,0.5000,0.8820,1.0000,1.0000,1.0000,0.7000,0.8330,0.9490]

# --- Razão tempo/duração ---
time_micro = [0.192,0.158,0.535,0.355,0.204,0.420,0.331,0.105,1.649,1.623,
1.031,2.155,1.593,1.830,0.369,0.301,0.030,0.356,1.080,1.704,
1.403,0.028,0.000,0.326,0.212,0.206,1.101,0.435,0.043,0.869]

time_cache = [0.320,0.143,0.505,0.351,0.190,0.412,0.346,0.095,1.688,1.669,
1.059,2.114,1.518,1.925,0.345,0.314,0.030,0.346,1.012,1.602,
1.360,0.028,0.000,0.322,0.193,0.207,1.052,0.406,0.038,0.793]

time_frag = [0.490,0.162,0.668,0.409,0.276,0.476,0.392,0.157,1.743,2.775,
1.028,3.527,3.578,2.776,0.451,0.355,0.072,0.388,1.053,1.645,
1.402,0.042,0.000,0.363,0.225,0.294,1.109,0.455,0.058,0.954]

# --- V-Measure ---
vm_cache = [0.919,1.000,0.762,1.000,0.782,0.722,0.818,0.000,0.951,0.938,
0.841,0.951,0.957,0.936,1.000,0.000,1.000,0.941,1.000,0.943,
0.838,1.000,0.000,0.636,1.000,1.000,0.727,1.000,1.000,0.840]

vm_frag = [0.726,1.000,0.430,0.544,0.716,0.503,0.514,0.000,0.826,0.870,
0.696,0.891,0.866,0.857,0.617,0.000,0.000,0.800,0.856,0.830,
0.468,0.000,0.000,0.623,0.361,0.456,0.617,0.799,0.000,0.782]

# (apoio) V-Measure Microsserviços, para checar estabilidade Micro x Cache
vm_micro = [0.928,1.000,0.762,1.000,0.809,0.722,0.818,0.000,0.951,0.928,
0.811,0.945,0.954,0.934,1.000,0.000,1.000,0.941,1.000,0.947,
0.838,1.000,0.000,0.706,1.000,1.000,0.727,1.000,1.000,0.832]

ALPHA = 0.05 / 3 # Bonferroni para 3 testes confirmatórios


def teste(nome, metrica, grupo_a, x, grupo_b, y, esperado):
stat, p = wilcoxon(x, y) # bilateral, pareado
ma, mb = median(x), median(y)
sig = "SIGNIFICATIVO" if p < ALPHA else "nao significativo"
direcao = "A > B" if ma > mb else ("A < B" if ma < mb else "A = B")
print(f"[{nome}] {grupo_a} x {grupo_b} --metrica: {metrica}")
print(f"   mediana {grupo_a}: {ma:.3f} |  mediana {grupo_b}: {mb:.3f} ({direcao})")
print(f"   W = {stat:.1f} p = {p:.5f}  (limiar Bonferroni = {ALPHA:.4f}) -> {sig}")
print(f"   esperado: {esperado}\n")


```

```
print("=" * 70)
print("TESTES CONFIRMATORIOS (Wilcoxon pareado, alpha de Bonferroni = "
f"{ALPHA:.4f})")
print("=" * 70 + "\n")

teste("H1", "Acuracia", "Mono", acc_mono, "Micro", acc_micro,
"Micro >> Mono (mensageria assincrona elimina o gargalo)")

teste("H2", "Razao tempo/duracao", "Micro", time_micro, "Cache", time_cache,
"Cache <= Micro (priorizacao reduz o tempo)")

teste("H3", "V-Measure", "Cache", vm_cache, "Frag", vm_frag,
"Frag < Cache (fragmentacao por genero degrada o agrupamento)")

print("=" * 70)
print("TESTES DE APOIO (contexto; nao entram na correcao dos 3 principais)")
print("=" * 70 + "\n")

# H2 --"sem degradar": acuracia e V-Measure devem permanecer estaveis
teste("H2-apoio", "Acuracia (estabilidade)", "Micro", acc_micro, "Cache", acc_cache,
"sem diferenca significativa = qualidade preservada")
teste("H2-apoio", "V-Measure (estabilidade)", "Micro", vm_micro, "Cache", vm_cache,
"sem diferenca significativa = qualidade preservada")

# H3 --degradacao tambem no tempo
teste("H3-apoio", "Razao tempo/duracao", "Cache", time_cache, "Frag", time_frag,
"Frag > Cache (overhead do sharding eleva o tempo)")
```

Código B.1 – Script de análise estatística: testes de Wilcoxon pareados com correção de Bonferroni.

A execução do script produz a seguinte saída:

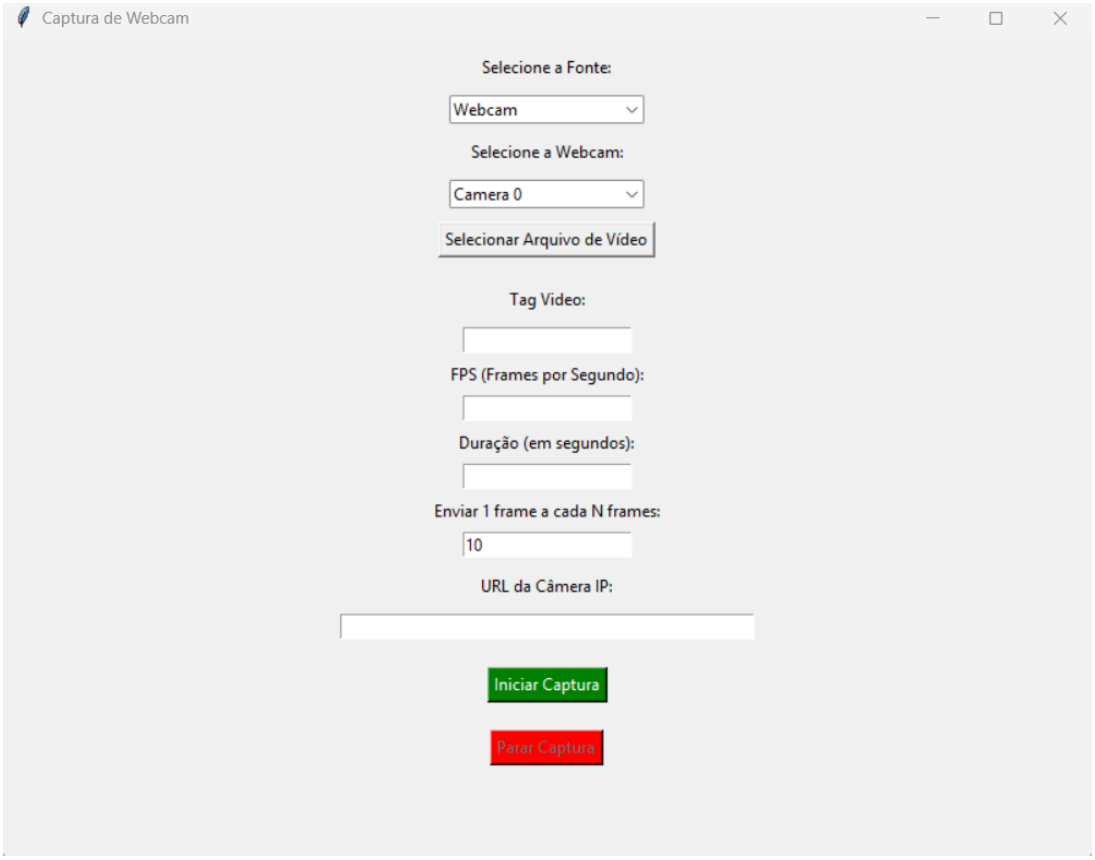
```
======================================================================
TESTES CONFIRMATORIOS (Wilcoxon pareado, alpha de Bonferroni = 0.0167)
======================================================================

[H1] Mono x Micro --metrica: Acuracia
mediana Mono: 0.057 |  mediana Micro: 0.944 (A < B)
W = 0.0  p = 0.00000  (limiar Bonferroni = 0.0167) -> SIGNIFICATIVO
esperado: Micro >> Mono (mensageria assincrona elimina o gargalo)

[H2] Micro x Cache --metrica: Razao tempo/duracao
mediana Micro: 0.362 |  mediana Cache: 0.348 (A > B)
W = 122.0  p = 0.10746  (limiar Bonferroni = 0.0167) -> nao significativo
esperado: Cache <= Micro (priorizacao reduz o tempo)

[H3] Cache x Frag --metrica: V-Measure
mediana Cache: 0.940 |  mediana Frag: 0.620 (A > B)
W = 0.0  p = 0.00001  (limiar Bonferroni = 0.0167) -> SIGNIFICATIVO
esperado: Frag < Cache (fragmentacao por genero degrada o agrupamento)
```

```
======================================================================
TESTES DE APOIO (contexto; nao entram na correcao dos 3 principais)
======================================================================

[H2-apoio] Micro x Cache --metrica: Acuracia (estabilidade)
mediana Micro: 0.944 |  mediana Cache: 0.944 (A = B)
W = 3.0  p = 0.46521  (limiar Bonferroni = 0.0167) -> nao significativo
esperado: sem diferenca significativa = qualidade preservada

[H2-apoio] Micro x Cache --metrica: V-Measure (estabilidade)
mediana Micro: 0.938 |  mediana Cache: 0.940 (A < B)
W = 27.0  p = 0.95935  (limiar Bonferroni = 0.0167) -> nao significativo
esperado: sem diferenca significativa = qualidade preservada

[H3-apoio] Cache x Frag --metrica: Razao tempo/duracao
mediana Cache: 0.348 |  mediana Frag: 0.453 (A < B)
W = 4.0  p = 0.00000  (limiar Bonferroni = 0.0167) -> SIGNIFICATIVO
esperado: Frag > Cache (overhead do sharding eleva o tempo)
```

APÊNDICE C – TELAS DO SISTEMA IMPLEMENTADO

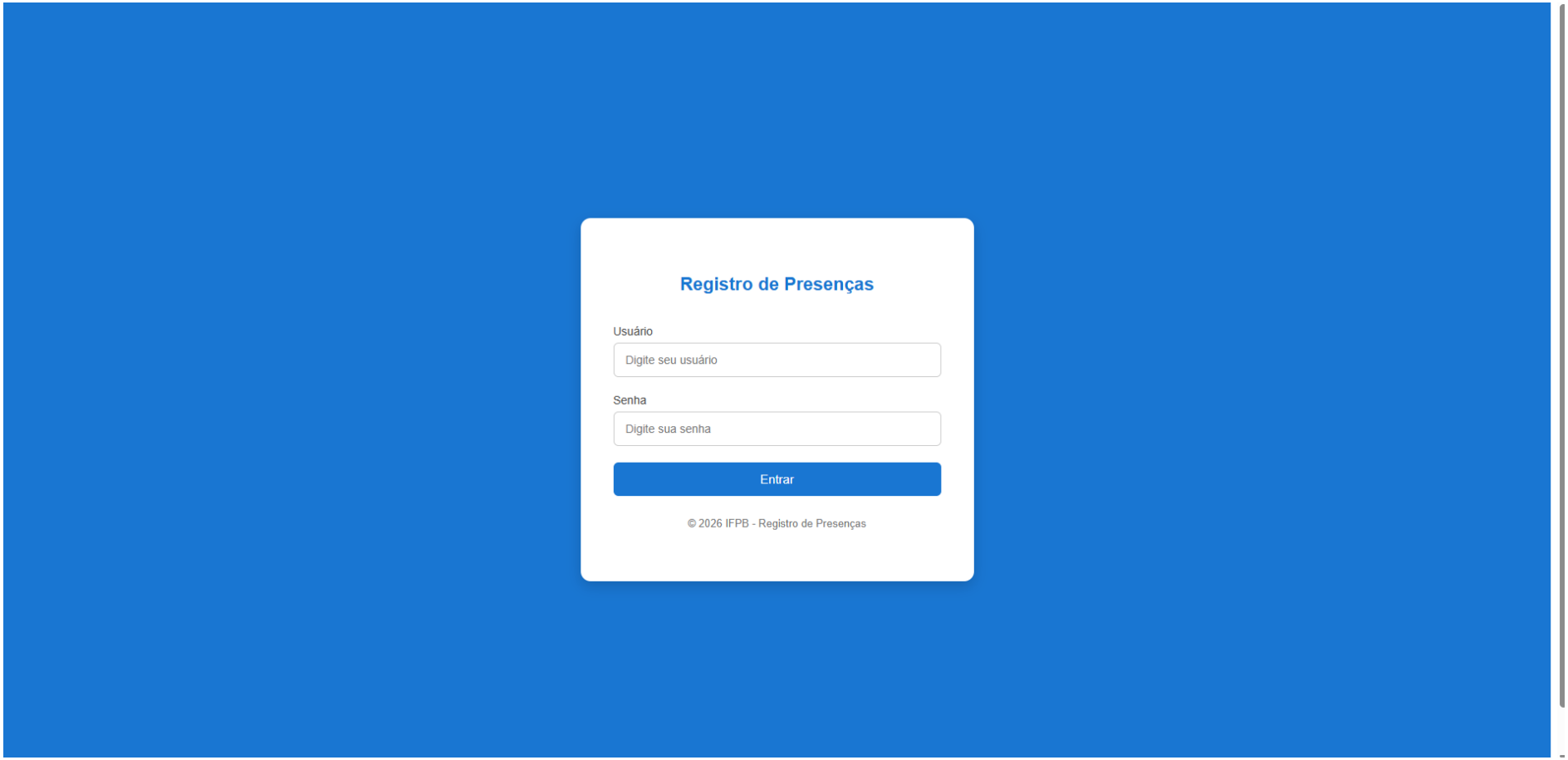
Este apêndice reúne as capturas de tela da interface do sistema implementado na Arquitetura com Microsserviços, referenciadas no Capítulo 3.

Figura 13 – Interface gráfica do worker de captura



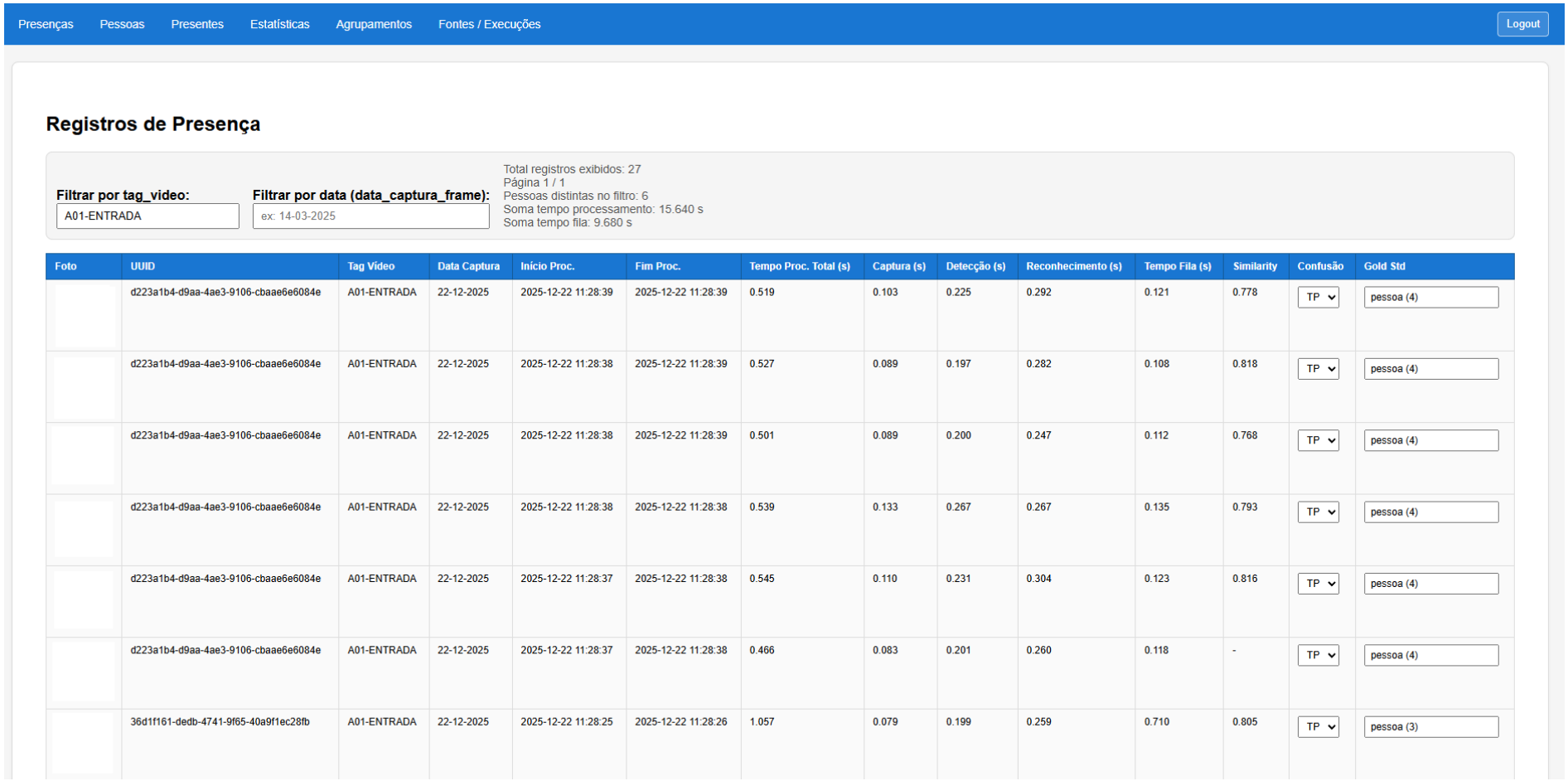
Fonte: Elaboração própria.

Figura 14 – Tela de autenticação do sistema



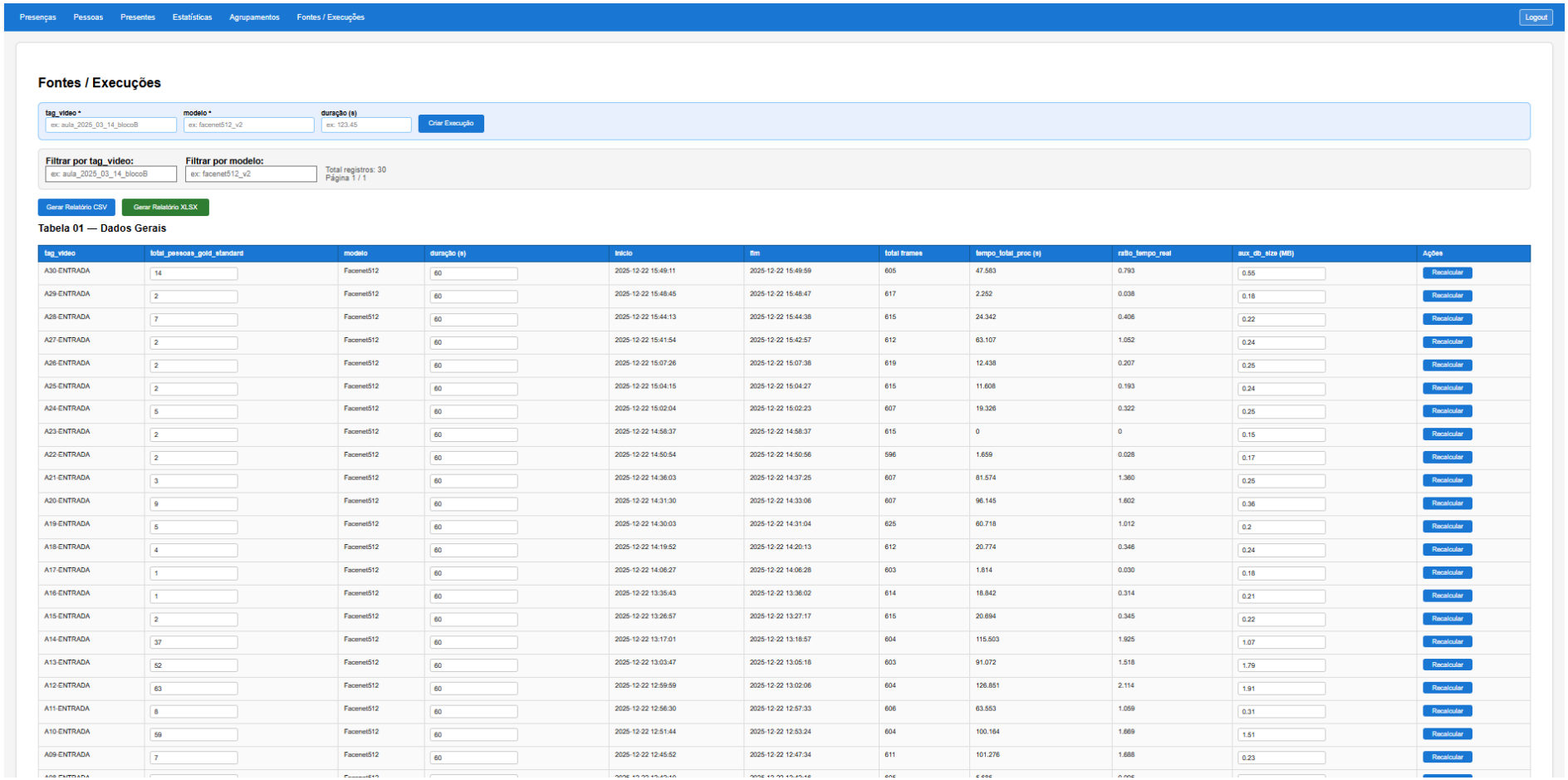
Fonte: Elaboração própria.

Figura 15 – Tela de listagem de registros de presença



Fonte: Elaboração própria.

Figura 16 – Dashboard — Dados Gerais



Fonte: Elaboração própria.

Figura 17 – Dashboard — Métricas de Eficiência

Tabela 02 — Métricas de Eficiência

tag_video	faces analisadas	clusters_gerados	faces_nao_reconhecidas	TP	TN	FP	FN	accuracy	precision	recall	f1
A30-ENTRADA	57	27	2	56	0	1	2	0.949	0.982	0.966	0.974
A29-ENTRADA	5	1	1	5	0	0	1	0.833	1	0.833	0.909
A28-ENTRADA	10	4	3	7	0	0	3	0.700	1	0.700	0.824
A27-ENTRADA	14	7	0	14	0	0	0	1	1	1	1
A26-ENTRADA	14	2	0	14	0	0	0	1	1	1	1
A25-ENTRADA	13	2	0	13	0	0	0	1	1	1	1
A24-ENTRADA	15	8	2	15	0	0	2	0.882	1	0.882	0.938
A23-ENTRADA	1	1	1	1	0	0	1	0.500	1	0.500	0.667
A22-ENTRADA	4	1	1	4	0	0	1	0.800	1	0.800	0.889
A21-ENTRADA	15	3	1	15	0	0	1	0.938	1	0.938	0.968
A20-ENTRADA	31	9	2	31	0	0	2	0.939	1	0.939	0.969
A19-ENTRADA	7	3	2	7	0	0	2	0.778	1	0.778	0.875
A18-ENTRADA	14	5	0	14	0	0	0	1	1	1	1
A17-ENTRADA	6	1	0	6	0	0	0	1	1	1	1
A16-ENTRADA	9	3	0	8	0	1	0	0.889	0.889	1	0.941
A15-ENTRADA	11	2	0	11	0	0	0	1	1	1	1
A14-ENTRADA	130	42	11	130	0	0	11	0.922	1	0.922	0.959
A13-ENTRADA	232	64	9	232	0	0	9	0.963	1	0.963	0.981
A12-ENTRADA	248	71	16	248	0	0	16	0.939	1	0.939	0.969
A11-ENTRADA	24	11	3	22	0	2	3	0.815	0.917	0.880	0.898
A10-ENTRADA	193	69	16	193	0	0	16	0.923	1	0.923	0.960
A09-ENTRADA	12	5	3	12	0	0	3	0.800	1	0.800	0.889
A08-ENTRADA	8	2	0	8	0	0	0	1	1	1	1
A07-ENTRADA	20	6	2	20	0	0	2	0.909	1	0.909	0.952
A06-ENTRADA	9	4	0	9	0	0	0	1	1	1	1
A05-ENTRADA	15	8	0	15	0	0	0	1	1	1	1
A04-ENTRADA	11	2	0	11	0	0	0	1	1	1	1
A03-ENTRADA	28	6	0	27	0	1	0	0.964	0.964	1	0.982
A02-ENTRADA	2	2	0	2	0	0	0	1	1	1	1
A01-ENTRADA	27	6	0	27	0	0	0	1	1	1	1

Fonte: Elaboração própria.

Figura 18 – Dashboard — Métricas de Agrupamento

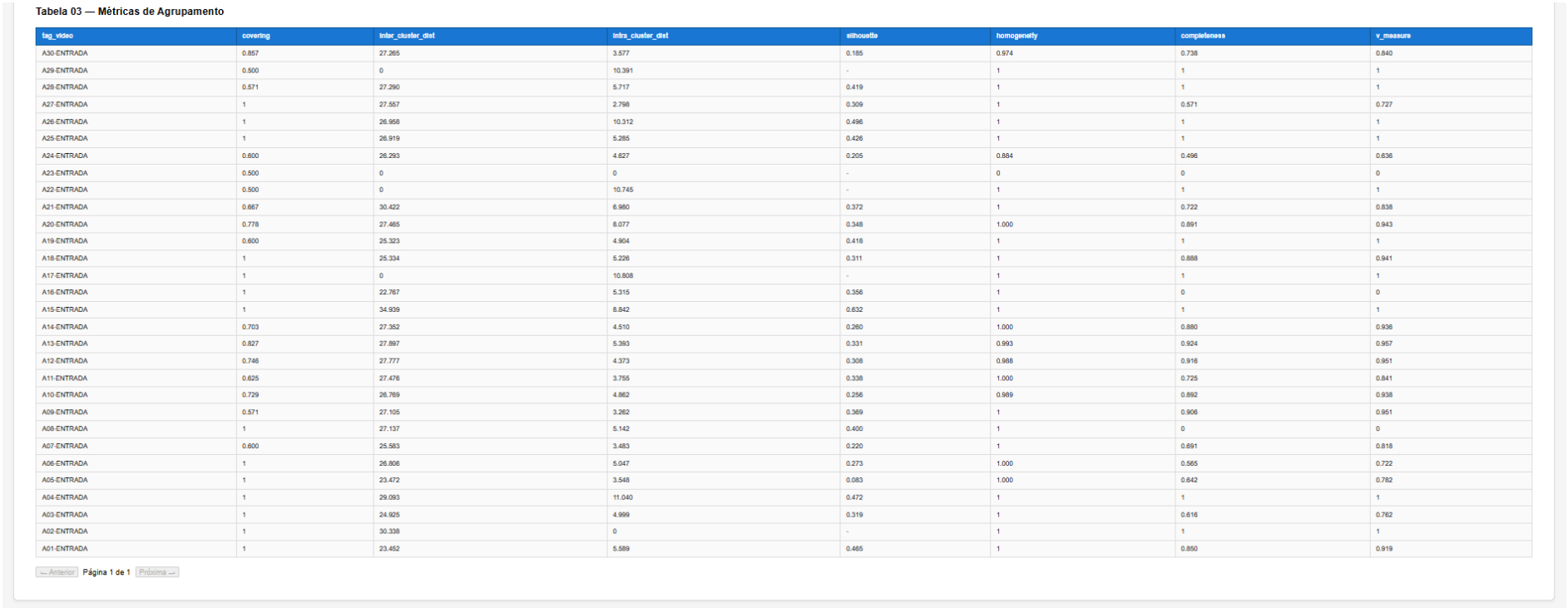


Tabela 03 — Métricas de Agrupamento

tag_video	covering	inter_cluster_dist	intra_cluster_dist	silhouette	homogeneity	completeness	v_measure
A30-ENTRADA	0.857	27.265	3.577	0.185	0.974	0.738	0.840
A29-ENTRADA	0.500	0	10.391	-	1	1	1
A28-ENTRADA	0.571	27.290	5.717	0.419	1	1	1
A27-ENTRADA	1	27.557	2.798	0.309	1	0.571	0.727
A26-ENTRADA	1	26.958	10.312	0.496	1	1	1
A25-ENTRADA	1	26.919	5.285	0.426	1	1	1
A24-ENTRADA	0.600	26.293	4.627	0.205	0.884	0.496	0.636
A23-ENTRADA	0.500	0	0	-	0	0	0
A22-ENTRADA	0.500	0	10.745	-	1	1	1
A21-ENTRADA	0.667	30.422	6.980	0.372	1	0.722	0.838
A20-ENTRADA	0.778	27.465	8.077	0.348	1.000	0.891	0.943
A19-ENTRADA	0.600	25.323	4.904	0.418	1	1	1
A18-ENTRADA	1	25.334	5.226	0.311	1	0.888	0.941
A17-ENTRADA	1	0	10.808	-	1	1	1
A16-ENTRADA	1	22.767	5.315	0.356	1	0	0
A15-ENTRADA	1	34.939	8.842	0.632	1	1	1
A14-ENTRADA	0.703	27.352	4.510	0.260	1.000	0.880	0.936
A13-ENTRADA	0.827	27.897	5.393	0.331	0.993	0.924	0.957
A12-ENTRADA	0.746	27.777	4.373	0.308	0.988	0.916	0.951
A11-ENTRADA	0.625	27.476	3.755	0.338	1.000	0.725	0.841
A10-ENTRADA	0.729	26.769	4.862	0.256	0.989	0.892	0.938
A09-ENTRADA	0.571	27.105	3.262	0.369	1	0.906	0.951
A08-ENTRADA	1	27.137	5.142	0.400	1	0	0
A07-ENTRADA	0.600	25.583	3.483	0.220	1	0.691	0.818
A06-ENTRADA	1	26.806	5.047	0.273	1.000	0.565	0.722
A05-ENTRADA	1	23.472	3.548	0.083	1.000	0.642	0.782
A04-ENTRADA	1	29.093	11.040	0.472	1	1	1
A03-ENTRADA	1	24.925	4.999	0.319	1	0.616	0.762
A02-ENTRADA	1	30.338	0	-	1	1	1
A01-ENTRADA	1	23.452	5.589	0.465	1	0.850	0.919

← Anterior Página 1 de 1 Próxima →

Fonte: Elaboração própria.