Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Proposta de Arquitetura para um Sistema de Controle de Atividades utilizando Reconhecimento Facial por Dispositivos Móveis

Kelvin William Moreira Lima

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Flávio de Barros Vidal

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Proposta de Arquitetura para um Sistema de Controle de Atividades utilizando Reconhecimento Facial por Dispositivos Móveis

Kelvin William Moreira Lima

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Flávio de Barros Vidal (Orientador)
CIC/UnB

Prof. Dr. Marcus Vinícius Chaffim Costa
Universidade de Brasília

Prof. Dr. Eduardo Adilo Pelinson Alchieri
Universidade de Brasília

Prof. Dr. João José Costa Gondim
Coordenadora do Curso de Engenharia da Computação

Brasília, 25 de Maio de 2021

Dedicatória

Dedico este trabalho à minha família, que sempre se dedicou ao máximo para que se tornassem realidade os sonhos de todos os seus filhos, mesmo diante das dificuldades encontradas pelo caminho.

Dedico também à minha esposa Arícia e à minha filha Aurora, por terem sido grandes motivações neste nesta significativa trajetória.

Por fim, dedico aos meus avós, Agostinho, Creuza, Walter(em memória) e Maria(em memória) por terem sido os precursores dessa família.

Agradecimentos

Agradeço a Universidade de Brasília e o corpo docente do Departamento de Ciência da Computação por toda a formação provida.

Agradeço a toda a minha família por todo o suporte durante essa jornada especialmente aos meus pais Fernando e Alessandra bem como aos meus avós Agostinho e Creuza, que forneceram apoio e dedicação desde os primeiros momentos em Brasília.

Agradeço aos meus irmãos Mattheus e Wiviann por todo empenho e parceira prestados, além da revisão de texto.

Agradeço à minha esposa Arícia por ter sido meu ombro amigo e um grande ponto de equilíbrio em vários momentos.

Agradeço à minha filha Aurora por ser a fagulha final de motivação para que esse sonho esteja se tornando realidade.

Agradeço à Lurielly, Ludmilla, Mattheus e Wiviann pelas revisões textuais.

Agradeço ao professor Dr. Flavio Vidal por ter me acolhido como seu orientando e, mais do que isso, ter prestado todo o suporte pessoal quando necessário.

Agradeço à LoopKey, especialmente Pedro Salum e Daniel Sandoval, por terem participado ativamente dessa jornada e terem aberto oportunidades que não seriam possíveis sem eles.

Resumo

Este trabalho apresenta uma proposta de uma arquitetura para um produto mínimo viável na validação de usuários a partir da utilização de algoritmos de reconhecimento facial. As imagens faciais capturadas são padronizadas a partir da Norma ICAO 9303, onde são utilizados arquivos de treinamento individuais para cada usuário do sistema de modo a ser aferida a identidade do usuário. O modelo foi designado para realizar o processo de reconhecimento facial com a apresentação de um pequeno conjunto de imagens e usuários para testes, devido à limitação do uso pela pandemia de Sars-Cov-2, para mostrar seu funcionamento em situações reais da aplicação. Optou-se para a demonstração de funcionamento a abordagem por ablação das partes dos componentes individuais da arquitetura proposta. A partir dos resultados obtidos, foram identificados os pontos positivos e negativos, sendo as inovações encaminhadas para trabalhos futuros.

Palavras-chave: Reconhecimento Facial, Dispositivos Móveis, Controle de Atividades

Abstract

This work presents a proposal for an architecture for a minimum viable product to validate users using facial recognition algorithms. The captured facial images are standardized based on the ICAO Standard 9303, where individual training files are used for each user of the system to verify the user's identity. The model was designed to perform the facial recognition process by presenting a small set of images and users for testing, due to your limited use on the Sars-Cov-2 pandemic, to show its operation in real situations. We opted to demonstrate functioning by the ablation approach of the parts of the individual components of the proposed architecture. All positive and negative issues were identified from the results, and proposed innovations were forwarded to future works.

Keywords: Face Recognition, Mobile Devices, Activities Management

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

Em salas de aula espalhadas pela Universidade de Brasília, professores têm como uma de suas responsabilidades atestar as frequências dos alunos presentes em cada dia letivo, de acordo com as informações disponíveis em [7], o que é realizado por meio de uma folha impressa na qual os alunos assinam a mesma em frente aos seus respectivos nomes. Em um momento posterior, cabe ao professor a contabilização dos dados e posterior lançamento desses no sistema da universidade para que assim, o mesmo se torne um dado oficial. Além disso, é designado ao professor o armazenamento desses dados por um período mínimo fixado em Lei [8].

Dito isto, podemos identificar alguns problemas associados ao fluxo previamente citado com um exemplo simples, a elevada probabilidade de erro humano durante o processo de assinatura, onde os dados não podem ser facilmente auditados ou da possibilidade de burlamento por parte dos atores envolvidos no processo.

Outrossim, surge a motivação para este trabalho que foca em tornar o processo menos suscetível a erros humanos e cria um processo auditável por seguir uma série padronizada de passos, além de tirar responsabilidades do professor que, idealmente, deveria focar somente no processo de ensino.

1.2 Definição do Problema de Pesquisa

Esta proposta consiste em projetar uma arquitetura de um sistema para atestar a viabilidade de algoritmos de reconhecimento facial para computação de presenças em fotos padronizadas, além da proposição de uma modelagem de software que possa atender a todos os atores envolvidos no processo em questão.

1.3 Objetivos

Os objetivos gerais e específicos deste trabalho são apresentados a seguir:

1.3.1 Objetivos Gerais

O objetivo geral a ser alcançado é a proposição e construção de um produto mínimo viável para a atestar a presenças de indivíduos de forma automatizada, de modo que torne o processo livre de erros humanos ou o mais próximo disto possível, permitindo a identificação unívoca do usuário da aplicação.

1.3.2 Objetivos Específicos

Os objetivos são listados a seguir:

- Atestar a proposta de utilização de algoritmos de reconhecimento facial com arquivos individuais de treinamento para cada usuário de modo que seja possível identificar quão similar um indivíduo é de quem diz ser, usando no arquivo de entrada um único arquivo global de treinamento por indivíduos.
- A construção de um sistema composto por painel administrativo em plataforma web, sistema de *backend* em nuvem como interface entre as plataformas e aplicativo móvel para alunos.
- Plataforma administrativa web para professores, com papéis específicos para professores e gestores (ex.: coordenadores de curso) para consolidar a estrutura organizacional de disciplinas, departamentos, turmas, professores e adição de alunos a cada período além de solicitar a aferição de presenças dos alunos numa dada turma.
- Uma plataforma móvel deve ser utilizada por alunos para que possam capturar imagens das suas faces de maneira padronizada com o intuito de atestarem a sua presença em um dado dia letivo de uma turma, estes previamente planejados e cadastrados.
- A plataforma de *backend* como ponto de sincronia entre as plataformas, além de armazenar todos os dados e ser o responsável pela a parte de reconhecimento facial a ser executada em momento oportuno.
- Ao final propor um sistema que compute as presenças em classes de aula de forma ágil e que seja de grande valor agregados, funcional e com praticidade para os atores envolvidos no processo.

1.4 Justificativa

Com o avanço das tecnologias e com a alta disponibilidades de *smartphones*, a cada momento mais presentes no cotidiano das pessoas, torna-se possível a utilização de ferramentas tecnológicas para as mais diversas tarefas. Portanto, esse trabalho se justifica com a criação de uma plataforma que traga mais funcionalidade, praticidade, confiabilidade e comodidade para os atores envolvidos no processo de avaliação de presenças, trazendo assim mais segurança ao processo, além de inúmeros dados úteis para gestores, professores, alunos e instituições de ensino ou capacitação.

1.5 Organização do Manuscrito

O trabalho é estruturado como descrito a seguir: No Capítulo 2, são apresentados o histórico dos algoritmos de reconhecimento facial e as principais técnicas utilizadas na implementação desse trabalho. No Capítulo 3, é apresentado um breve resumo dos principais trabalhos que se propuseram a realizar trabalhos semelhantes aos propostos neste estudo. No Capítulo 4, a proposta de solução a ser implementada sob a forma de uma arquitetura, e os principais módulos de funcionamento. Em seguida, no Capítulo 5, são mostrados os resultados da implementação da proposta bem como os resultados específicos do reconhecimento facial. Por fim, no Capítulo 6 são apresentadas as conclusões que podem ser tiradas do trabalho construído, bem como as propostas de trabalhos futuros para que a solução se torne ainda mais robusta.

Capítulo 2

Fundamentação Teórica

Neste capítulo serão apresentados os principais conceitos e métodos a respeito das técnicas a serem desenvolvidas e propostas nesse trabalho. Adicionalmente, serão apresentadas a evolução e as principais contribuições dos métodos aqui trabalhados.

O capítulo será iniciado com um breve histórico sobre os sistemas de reconhecimento facial, bem como a cerca dos utilizados para fins específicos, a exemplo sistemas de reconhecimento facial para fins forenses, e seguido de conceitos mais detalhados sobre os métodos a serem utilizados nesse trabalho, sendo eles as técnicas de reconhecimento facial *Eigen Faces*, *Fisher Faces* e *Local Binary Patterns*.

2.1 Sistemas de Reconhecimento Facial

2.1.1 Histórico

A primeiro proposta de um sistema de reconhecimento facial é datada dos anos 60, introduzida por Bledsoe, Wolf e Bisson [1], que propuseram um método de reconhecimento facial automatizado. Em seu trabalho classificado como homem-máquina, necessitava que as coordenadas de uma série de características fossem extraídas e usadas em um computador para a realização do processo de reconhecimento. As características normalmente utilizadas passavam por coordenadas do centro das pupilas, da parte interna do contorno dos olhos, tamanho da boca, além de várias outras. Partindo do pressuposto, todas essas informações eram guardadas em um banco de dados e associadas a uma pessoa e, posteriormente, comparadas com outras fotografias e as mais semelhantes eram retornadas como resultado do processo de reconhecimento.

De acordo com [1], problemas eram notados em diferentes frentes como iluminação, rotação de cabeça, iluminação, escala das imagens, além do incômodo processo de manu-

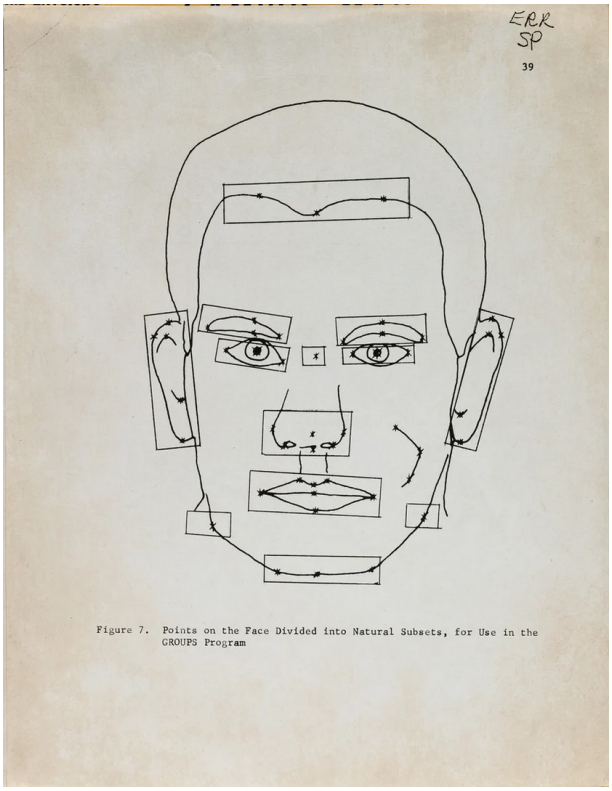


Figura 2.1: Divisão proposta por Woody Bledsoe para características da face a serem analisadas [1].

almente ter de se extrair as coordenadas das características a serem utilizadas. Com isso, tem-se a evolução dos conceitos de reconhecimento facial.

Seguindo a linha proposta em [9] nos meados anos 70, Takeo Kanade propôs um sistema para extração automática das características utilizadas em sistemas de reconhecimento facial, a partir de um banco de 850 fotografias. O que fazia o processo de reconhecimento facial ser totalmente automatizado já que eliminava a necessidade de ação humana em qualquer nível do processo.

A partir desses avanços, os estudos passaram a se dedicar em resolver os problemas já encontrados no primeiro sistema proposto e melhorar a acurácia da detecção.

Desde então, em meados dos anos 90 a ideia da utilização de reconhecimento facial começou a ser largamente utilizada, principalmente pela grande popularização dos sistemas computacionais por conta da redução de custos.

Com isso, inúmeras técnicas foram desenvolvidas e apresentadas no decorrer dos anos, como:

1. Eigen Faces [10]
2. Local Binary Patterns [11]
3. Fisher Faces [12]

4. Scale Invariante Feature Transform(SIFT) [13]
5. Speed Up Robust Features (SURF) [14]
6. FaceNet [14]
7. Multi-task Cascaded Convolutional Networks (MTCNN) [2]

A partir da popularização da uso de sistemas de reconhecimento facial, suas aplicações foram inúmeras. Hoje, as técnicas deste método são presentes nos mais variados eletrônicoscomo no desbloqueio de um *smartphone* ou em sistemas mais complexos de reconhecimento para fins forenses.

2.1.2 Principais Métodos

Eigen Faces

O método foi proposto em 1991 por *M. Turk* e *A. Pentland*, com o objetivo de ser um algoritmo de reconhecimento facial baseado em aparência, que procura e captura a variação numa coleção de imagens e usa essas informações para codificar e comparar imagens de maneira holística [10].

As motivações para o trabalho realizado por Turk e Pentland foram, primeiramente, de se extrair informações relevantes da face, podendo ou não estar diretamente relacionadas com a intuição humana de características como olhos, nariz e lábios, por meio da variação estatística entre imagens e segundo, de representar imagens de uma maneira eficiente, reduzindo a complexidade e o espaço computacional para que a imagem possa ser representada por um pequeno número de parâmetros.

Portanto, o algoritmo *Eigen Faces* pode ser considerado um pequeno conjunto de características da face que designam uma variação entre elas. Cada imagem pode ser aproximada utilizando um pequeno conjunto de *Eigenface* associados a grandes conjunto de *Eigenvalues*.

O processo para se computar as *Eigenfaces* é feito considerando uma imagem 2D, representada por I(x,y), de dimensões NxN com valores de intensidade, partindo do método PCA, que tem por objetivo achar os melhores vetores para distribuição de imagens de faces no espaço. Esses vetores são definidos como subespaços de imagens, chamado, pelos autores de "Espaço de Face". Cada vetor tem um tamanho N^2 que descreve uma imagem de dimensões NxN e a sua combinação linear das imagens originais de face.

Portanto, a utilização do algoritmo no processo de reconhecimento facial passa pela extração de características em uma imagem de face, codificando-as de maneira mais eficiente possível e comparando-a com outras previamente codificadas e guardadas de forma similar.

O processo de reconhecimento facial utilizando-se do *Eigen Faces* pode ser descrito em apenas alguns passos:

1. Inicialização: Aquisição das faces de teste e cálculo de sua *eigenface*, a qual define o seu espaço facial;
2. Quando uma nova face é encontrada, calcula-se um conjunto de pesos baseado na imagem de entrada e nas M *eigenfaces*, projetando a imagem de entrada para cada *eigenface*;
3. Determina-se se a imagem como um todo, checando para descobrir se a imagem está suficientemente próxima ao espaço facial;
4. Se realmente é uma face, classifica-se o modelo de pesos como uma pessoa conhecida ou desconhecida;
5. (Opcional) Se uma mesma face desconhecida é apresentada várias vezes, calcula-se suas características e a incorpora-se esta ao conjunto de faces conhecidas, isto é, aprende-se a reconhece-la.

Fisher Faces

O método foi proposto em 1997 por Peter N. Belhumeur *et al.*, com o objetivo de apresentar uma técnica na qual o reconhecimento facial ocorra de maneira insensível a intensas variações na direção da iluminação e a variação de expressões faciais [12].

Todos os algoritmos apresentados antes do *Fisher Faces* tomam como base a presença de condições ideais de iluminação, posição e expressões faciais. Quando essas condições ideias não se fazem presentes, as taxas de erro crescem, deixando evidentes as limitações dos métodos.

O fato de as imagens serem previamente rotuladas faz com que a utilização de métodos lineares para redução da dimensionalidade do espaço de características seja interessante, fresultando, assim em melhores taxas de reconhecimento em métodos como *Eigenfaces*.

A utilização de técnicas de classe específica, como o *Fisher's Linear Discriminant (FLD)*, tenta adicionar uma forma à dispersão, de modo a torna-lá mais confiável para classificação [12].

A melhora fica evidente ao observar-se os resultados experimentais propostos no artigo, onde a taxa de erro do método proposto fica em 4,6%, enquanto o método *Eigenfaces* 2.1.2 fica em 47,7% para um experimento em que são realizadas variações de condições de iluminação.

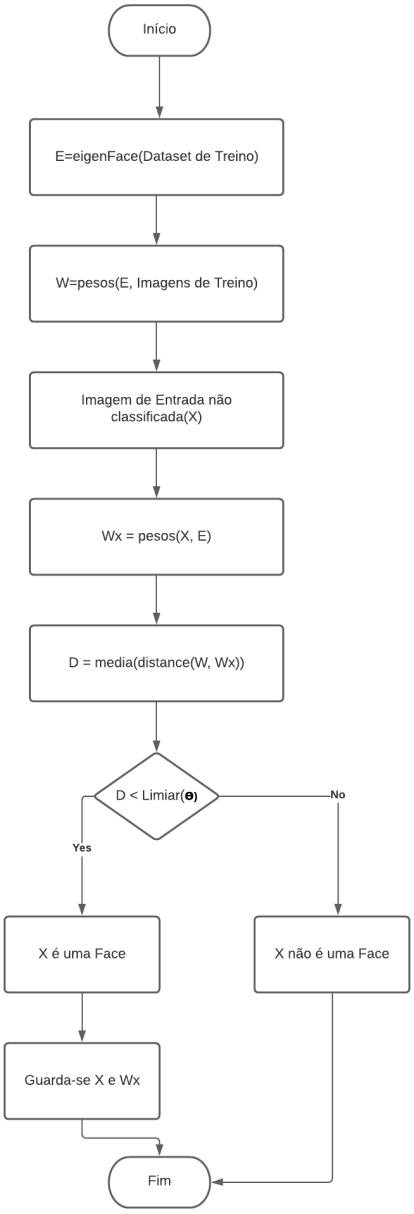


Figura 2.2: Fluxograma de Alto nível da utilização do Eigenfaces para reconhecimento facial

Para um experimento com variação de expressões faciais, iluminação e utilização de óculos, os resultados são ainda melhores haja visto que onde o *Fisherfaces* apresenta uma taxa de erro de 1,7% e o *Eigenfaces*2.1.2 uma taxa de erro de 52,9%.

Local Binary Patterns

O método foi proposto por Timo Ahonen *et al.* com o objetivo de considerar tanto as informações de formato quanto de textura para a representação de uma face. Dividindo

a face em áreas pequenas com o qual o histogramas de Local Binary Patterns (LBP) são extraídos e concatenados em um único e espacialmente melhorado histograma de características para representar uma imagem facial [11].

Ao utilizarmos o operador LBP, vários padrões podem ser adotados. Para o caso do método propostos nesse estudo, foi-se utilizado um operador chamado "padrão uniforme". Um LBP é dito uniforme se ele contém pelo menos duas alterações bit a bit de 0 para 1 ou vice-versa quando a *string* é considerada circular [11].

Como o histograma contém informações sobre a distribuição local de micro padrões como bordas, pontos e áreas planas em toda a imagem, para que seja mais eficiente o algoritmo deve reter informações espaciais e para que isso seja feito é então dividida entre regiões. Partindo do pressuposto, o histograma melhorado passa a ter uma descrição da face com três níveis de localidade: os rótulos do histograma contêm informações sobre os padrões a nível de pixel, onde são somados por via de pequenas regiões para produzir informação em nível regional, e estas regiões do histograma são concatenadas para construir uma descrição global da face [11].

Sob o olhar da classificação de padrões, um problema comum no reconhecimento de imagens é a abundância de classes e somente um pequeno e, possivelmente, único exemplo destas [11]. Nesse contexto, são propostos inúmeros métodos para classificação, sendo necessário apenas a utilização de um classificador de vizinho mais próximo(*nearest-neighbor*). Várias possíveis medidas de dissimilaridade são propostas para histogramas, como: Interseção de Histogramas, Estatística de log-verossimilhança e Estatística de Chi Quadrado.

Quando a imagem é dividida em regiões, é esperado que algumas dessas contenham mais informações relevantes que outras para a distinção entre pessoas. Com isso, para que seja tirada vantagem dessa característica, pesos são utilizados para cada uma das áreas, se baseando na informação obtida.

A qualidade de reconhecimento facial é atrelada, diretamente, ao tamanho da janela escolhida e dos pesos para cada região determinada. Todos os parâmetros são testados e calibrados seguindo a modelagem proposta pelo *CSU Face Identification Evaluation System*, visando a melhor utilização do recurso computacional disponível e o aumento da taxa de precisão de reconhecimento.

Olhando sob o ponto de vista de busca por um valor ótimo para tamanho de janela e de operador LBP, a representação é bem robusta com respeito a seleção de parâmetros. Alterações nesses podem causar grandes diferenças no tamanho do vetor de características, porém a performance geral não é, necessariamente, afetada. Como esperado, janelas muito grandes induzem a um decréscimo na taxa de reconhecimento por conta da perca de informação espacial. Desta forma, um operador com janela dimensões 18x21 é selecionada

como um bom *trade-off* entre desempenho de reconhecimento e tamanho do vetor de características [11].

Pelo método de análise proposto, ficou claro que a técnica de reconhecimento proposto por esse estudo tem uma performance melhor que os algoritmos previamente propostos, alcançando uma taxa de reconhecimento de 97% em *datasets* com diferentes expressões faciais. Sob variações de iluminação, o algoritmo teve uma taxa de reconhecimento de 79% o que o torna superior a todos os demais [11].

Multi-task Cascaded Convolutional Networks

O método proposto por Kapieng Zhang *et al.* para detecção de face e alinhamento em um ambiente natural é bastante desafiador por fatores como iluminação, poses e oclusões. No estudo é apresentada uma técnica multi-tarefa profunda em cascata que explora a correlação herdada entre elas para melhorar a performance. O algoritmo é dividido em três etapas cuidadosamente desenhadas em redes neurais convolucionais, as quais conseguem predizer a face e os pontos de interesse de uma maneira ruim para ótima com o passar dos estágio [2].

Neste trabalho, a principal contribuição é a junção das tarefas de detecção de pontos de interesse e de face através de uma cascata unificada de CNN com um de aprendizado multitarefa. A proposta consiste em três estágios: no primeiro, são produzidas janelas candidatas rápidas mediante uma CNN rasa, posteriormente, as janelas são refinadas por uma rede CNN mais complexa e, por fim, que tem os pontos de interesse facial como produtos dessa saída.

O abordagem utilizada consiste no redimensionamento da imagem para diferentes escalas afim de que seja assim seja construída uma pirâmide de imagens que serão o os objetos de entrada do *framework*.

Na primeira etapa é explorada uma rede totalmente convolucional chamada *Proposal Network* (P-Net), a qual obtém as janelas candidatas e seus vetores de regressão de caixa delimitadora. Após isso, é empregado um algoritmo de Supressão Não Máxima(NMS) para que sejam mesclados candidatos altamente sobrepostos.

Na etapa seguinte, todos os candidatos produzidos na etapa anterior são utilizados para alimentar uma outra CNN chamada de *Refined Network*(R-Net), a qual rejeita um grande número de falsos candidatos e realiza uma regreção de caixa delimitadora e uma nova mescla por meio do NSM.

Por fim, é destacável que a última etapa apresenta similaridade com a etapa anterior, mas visando atingir o objetivo de atingir uma face descrita em mais detalhes. Particularmente, o produto dessa rede são 5 pontos de interesse facial. Um exemplo gráfico é exposto na Figura 2.3.

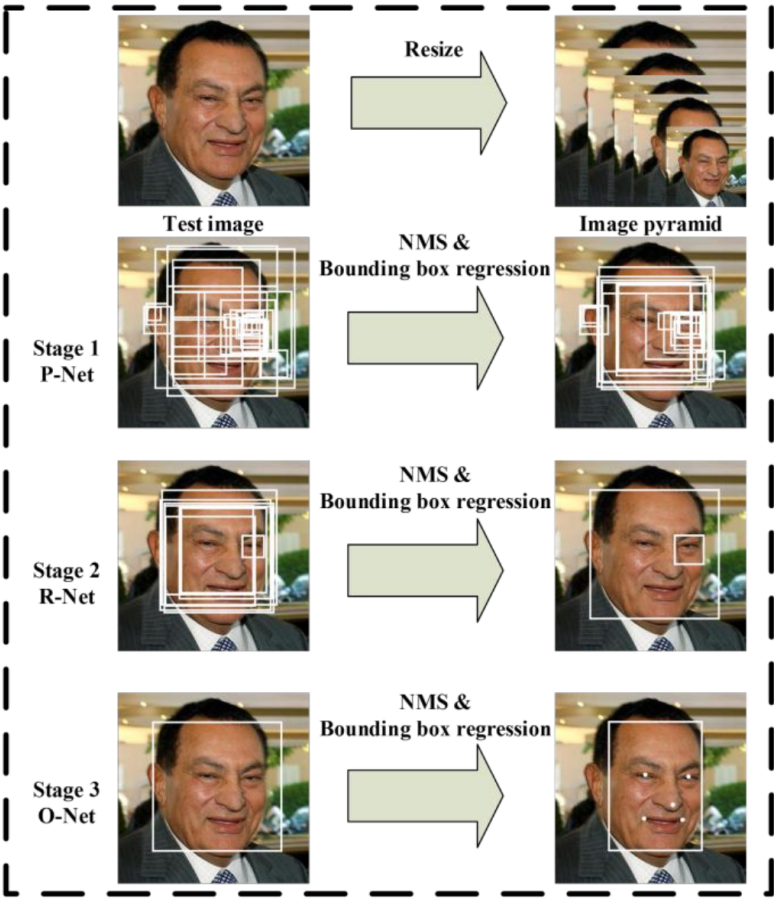


Figura 2.3: Representação gráfica dos passos utilizados para a detecção da face e seus pontos de interesse diante da técnica MTCNN [2]. Figura retirada de [2].

Com a técnica proposta, apresentou-se um algoritmo com taxa de reconhecimento de 94%, sendo assim uma taxa acima do estado da arte até o momento em que foi apresentado. Outro ponto de vantagem é que o algoritmo foi construído para utilização em tempo real, com isso, a quantidade de aplicações que podem tirar proveito desse método se expande de maneira considerável.

2.2 Reconhecimento Facial para Dispositivos Móveis

Com o avanço crescente dos dispositivos móveis, várias aplicações começaram a ser disponibilizadas, testadas e exploradas. Uma área em franca expansão é a do Reconhecimento Facial, a qual se beneficia bastante com o avanço signifcativo dos *hardwares* de câmera, além da utilização de processadores cada vez mais potentes, o que faz com as suas operações serem realizadas em tempo real(ou quase).

Neste viés, várias tecnologias começaram a ser fornecidas e melhoradas com foco em extrair o máximo disponibilizado pela plataforma móvel. Esforços como o Vision Api [3],

que faz parte do ML Kit no Android, e o Vision Framework [15], nas plataformas da Apple, além do OpenCV[16], que é uma ferramenta clássica de desenvolvimento nesse campo, demonstram o empenho disponível em se obter o melhor disponível.

2.2.1 Principais Tecnologias e Arquiteturas Disponíveis

O OpenCV [16] é a plataforma mais tradicional para o aprendizado e desenvolvimento de técnicas de processamento digital de imagens, sendo amplamente utilizado no meio acadêmico e em muitos ambientes de produção. Suas APIs são implementadas em C e C++ com interfaces para as linguagens mais comumente utilizadas, como Python, Javascript, Java, além de outras portabilidades não oficiais feitas pela comunidade de desenvolvimento de software. Em plataformas móveis, é notável pela sua ampla documentação e suporte por parte da comunidade.

Saltando para as APIs totalmente voltadas para os dispositivos móveis, no Android a utilização da *Vision Api* [3] disponível por meio do MLKit, oferecendo uma série de funcionalidades prontas para facilitar a utilização das diferentes técnicas desenvolvidas de reconhecimento facial, como por exemplo a detecção de faces e reconhecimento de expressões 2.4.

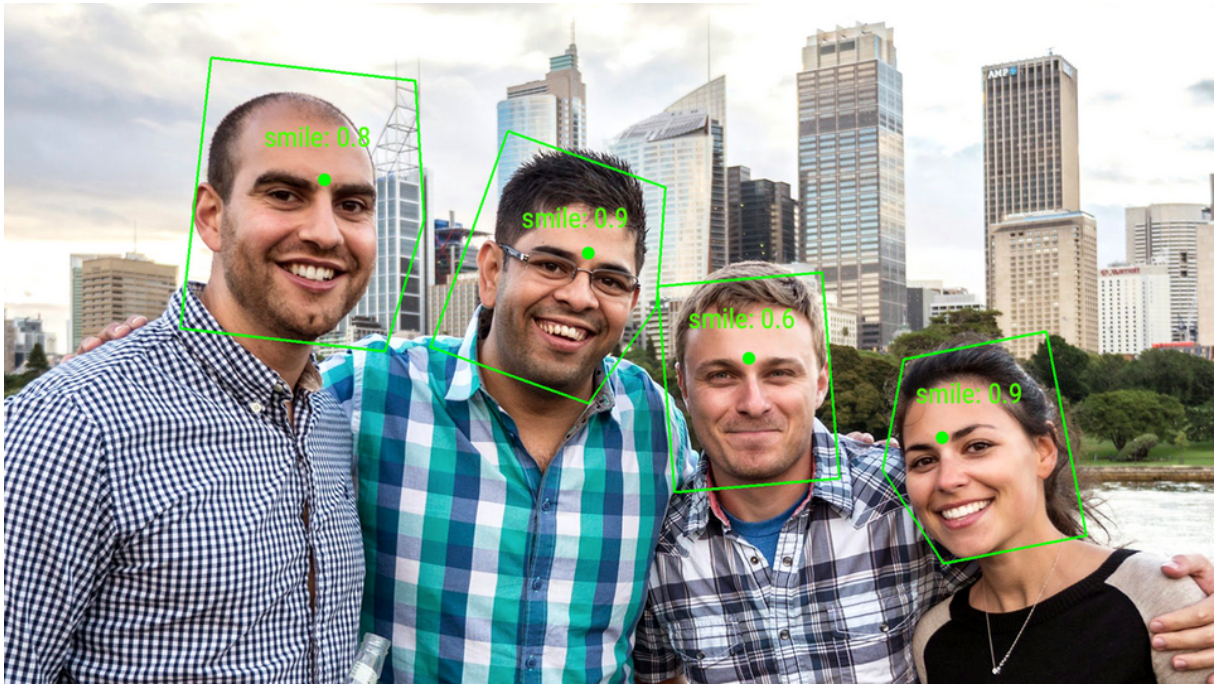


Figura 2.4: Representação do reconhecimento de expressões faciais feita pela VisionApi [3]

O *Vision Framework* [15], disponível para as plataformas Apple, tem o mesmo foco do *framework* disponível no Android. Dentre as funcionalidades disponibilizadas estão

o rastreamento de objetos, análise de saliências, análise de imagens estáticas, detecção facial e de corpo, detecção de trajetória, detecção de contorno, dentre várias outras.

Além da utilização desses frameworks já construídos, existem ferramentas que permitem a utilização de modelos baseados em redes neurais, como por exemplo o CoreML [17] e MLKit [18] e TensorFlowLite [19]. A partir do uso dessas ferramentas, torna-se possível a utilização de técnicas como MTCNN [2] e tantas outras.

Por fim, não são mencionadas em nenhuma das ferramentas citadas a filtragem a ataques em imagens, como o *anti-spoofing*, de maneira automática ficando a cargo do desenvolvedor o utilização destas técnicas.

Capítulo 3

Trabalhos Relacionados

Com o avanço das técnicas de reconhecimento facial, inúmeros estudos começaram a surgir demonstrando suas aplicações. Posteriormente, com o aumento da mobilidade, conectividade e portabilidade além do crescente poder computacional em dispositivos cada vez menores fizeram com que as essas técnicas fossem popularizadas. Com isso, inúmeros estudos relacionados apareceram e mostraram a eficácia de técnicas de reconhecimento facial em dispositivos móveis, com destaque para que seja computada a presença de alunos, professores e outros profissionais em cursos de capacitação, aulas, palestras ou quaisquer eventos em que sejam necessários. Estudos prévios, como os mostrados em [5], [20], [21] e [22], embasam a eficácia das técnicas de reconhecimento facial e seus sistemas para a melhora da fluidez em sala de aula, ao tornar o processo mais fácil e menos suscetível a erros humanos, além de mais amigável.

Diante disso, aplicações similares já foram construídas e descritas em publicações que são responsáveis por embasar o estudo aqui proposto. Foi-se gerado um grafo de referências a partir da ferramenta *Connected Papers* [4], utilizando-se como artigo de entrada o proposto por Sunaryono *et al.* [5], para o qual são referenciados com círculos maiores aqueles com maior número de citações e o grafo gerado é exposto em 3.1. Portanto, navegando-se por meio desse grafo, são expostos a seguir os trabalhos mais relevantes obtidos.

Em [5], Sunaryono *et al.* propõe um sistema para computação de presença que depende de uma imagem captura do estudante e de um QR Code utilizando o seu *smartphone*. Logo ambos os parâmetros são enviados para um servidor que realiza o reconhecimento e a computação da presença. Entrando em mais detalhes, o processo de captura é realizado no *smartphone* e captura a imagem do usuário com diferentes expressões, posteriormente, a imagem é cortada e redimensionada para ter 224x224 pixels e de modo a conter toda a face, então, a imagem é enviada para um servidor localizado no próprio campus. Todo o processo utiliza o algoritmo de Viola Jones. Na parte do servidor, PHP foi a linguagem

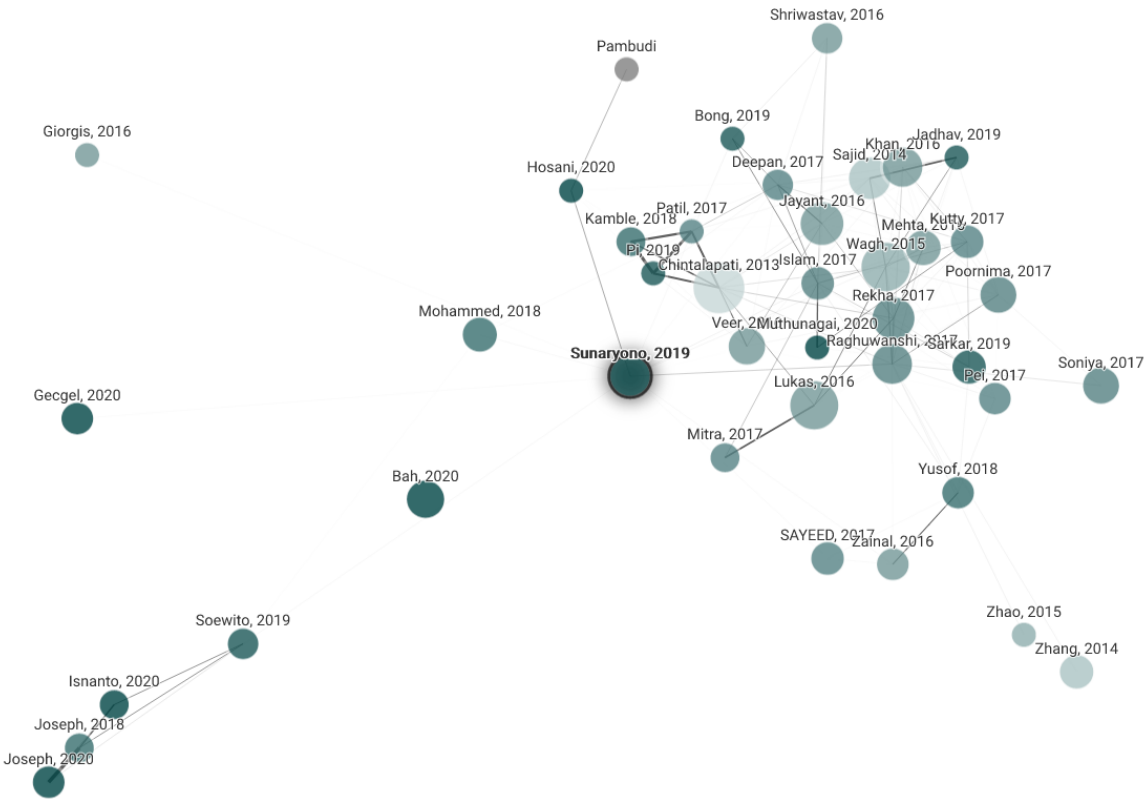


Figura 3.1: Grafo de referências geradas através do Connected Papers[4] utilizando-se como fonte de entrada o artigo proposto por Sunaryono *et al.* [5]

escolhida para construção do processo, durante a parte de processamento todas as imagens são redimensionadas para 96x96, convertidas para escala de cinza e em seguida, para um vetor que será utilizado como entrada para o processo de classificação. Os resultados apresentados no estudo são satisfatórios e apresentam uma taxa de reconhecimento de 97% além de um tempo de reconhecimento de 0.000096s no servidor.

Em [20], Samuel Lukas *et al.* propõe a utilização de *Tranformada Discreta de Wavelets* (DWT) e *Transformada Discreta de Cossenos(DCT)* para extração de características da face dos estudantes e o uso de *Função de Base Radial* (RBF) para a classificação das características extraídas. Visando esse processo, todas as imagens passam por normalização em escala de cinza e equalização de histograma, além de um redimensionamento para que todas as imagens tenham exatamente o tamanho 64x64 pixels. A arquitetura proposta consiste em um *dataset* de treino como entrada, extração das características da face e treinamento da *Redes de Função de Base Radial* (RBFNN). Logo após, é então dada a entrada no sistema de uma imagem, detecta-se sua face, extrai-se suas características por meio da DCT e DWT e, então, usa-se a rede treinada para reconhecimento, que tem como saída o identificador do estudante reconhecido. Os resultados apresentados apon-

tam para uma taxa de reconhecimento de 82% que fica aquém do desejado. É apontado como problema a técnica utilizada no estudo, a qual configura-se como ponto de melhoria futura.

Em [21], Wagh *et al.*, diferentemente dos outros estudos aqui citados, propõe a utilização da detecção de múltiplas faces em uma imagem ao invés de uma única face por imagem. Ou seja, uma imagem para computar a presença de toda a turma, em oposição a cada aluno computar a sua. Desse modo, propões-se então a utilização de uma câmera de alta definição posicionada em um lugar que toda a classe possa ser coberta por apenas uma imagem. Posteriormente, a imagem é convertida, em escala de cinza e passa por uma melhoria utilizando-se a técnica de equalização de histograma em seguida, é realizada a detecção facial, que é o recorte das diferentes faces presentes na imagem que então são comparadas com as imagens presentes no banco de dados, cada face reconhecida computa a presença de um estudante. Para que seja realizada a detecção facial é utilizado o algoritmo de Viola-Jones, e para reconhecimento facial o método *Eigenfaces*. No estudo, não são apresentados dados sobre eficiência, porém menciona-se que os problemas de variação de intensidade de iluminação e poses da cabeça são resolvidos.

Em [22], N. K. Jayant e S. Borra propõe a criação de um sistema automatizado para gerenciamento de presença, baseado em técnicas de detecção e reconhecimento facial. Para a detecção facial é utilizada uma versão modificada do algoritmo de Viola-Jones e um algoritmo de reconhecimento parcial de face sem alinhamento. Após a efetivação do processo, o sistema automaticamente atualiza a presença em uma planilha do Excel. A metodologia proposta consiste no posicionamento de uma câmera de alta resolução acima do quadro negro, em que todas as imagens capturadas são transmitidas para um servidor, armazenadas e processadas em tempo real. Subsequentemente, a imagem é convertida em escala de cinza, as características das imagens são computadas utilizando-se do algoritmo de Viola-Jones e as mesmas são classificadas utilizando-se a técnica de aprendizado *Adaboost*, para que sejam selecionadas as mais importantes a partir das características previamente computadas. Após a detecção, todas as faces são passadas por um filtro para que sejam identificados os seus tamanhos e caso estejam entre 30x20 e 200x200, são recortadas e guardadas para reconhecimento facial, sendo esse processo é repetido até o fim da classe.

Ademais, após o processo de detecção todas as imagens são enviadas para o algoritmo de reconhecimento facial, que utiliza-se do método de *Multi-Key point Descriptor* (MKD), no qual as faces são representadas por um esparso e grande dicionário de galeria de descritores. Por fim, os passos de detecção de reconhecimento facial são combinados e, a cada vez que um estudante é reconhecido, o sistema computa a presença, e esse processo se repete até o fim da classe. Os resultados apresentados mostram que a taxa de

reconhecimento varia bastante, de acordo com a atenção dos alunos ao quadro por conta do posicionamento da câmera, em que os melhores resultados são obtidos quando a maior parte dos estudantes está com a face posicionada frontalmente para o quadro.

Capítulo 4

Arquitetura Proposta

4.1 Contextualização da Proposta de Arquitetura

O trabalho aqui proposto consiste em uma solução para que seja computada de maneira automática a presença de alunos em uma classe de aula, retirando do professor essa responsabilidade que em turmas extensas demanda um tempo excessivo e é passível de erros humanos. Então, o trabalho é desenvolvido com base em 3 atores, a saber:

- Gestor (Seção 4.3): Responsável por toda a criação da estrutura e suas associações, além de computar todas as frequências;
- Professor (Seção 4.4): Responsável pela criação de dias letivos, disponibilização do código QR identificador do dia letivo e acompanhamento de suas turmas;
- Aluno (Seção 4.5): Ator com menos atribuições e responsável por computar sua própria presença;

A estrutura em faixas demonstrada na Figura 4.1 concede uma visão geral acerca da arquitetura proposta e os detalhes de cada ator são representados nas seções seguintes.

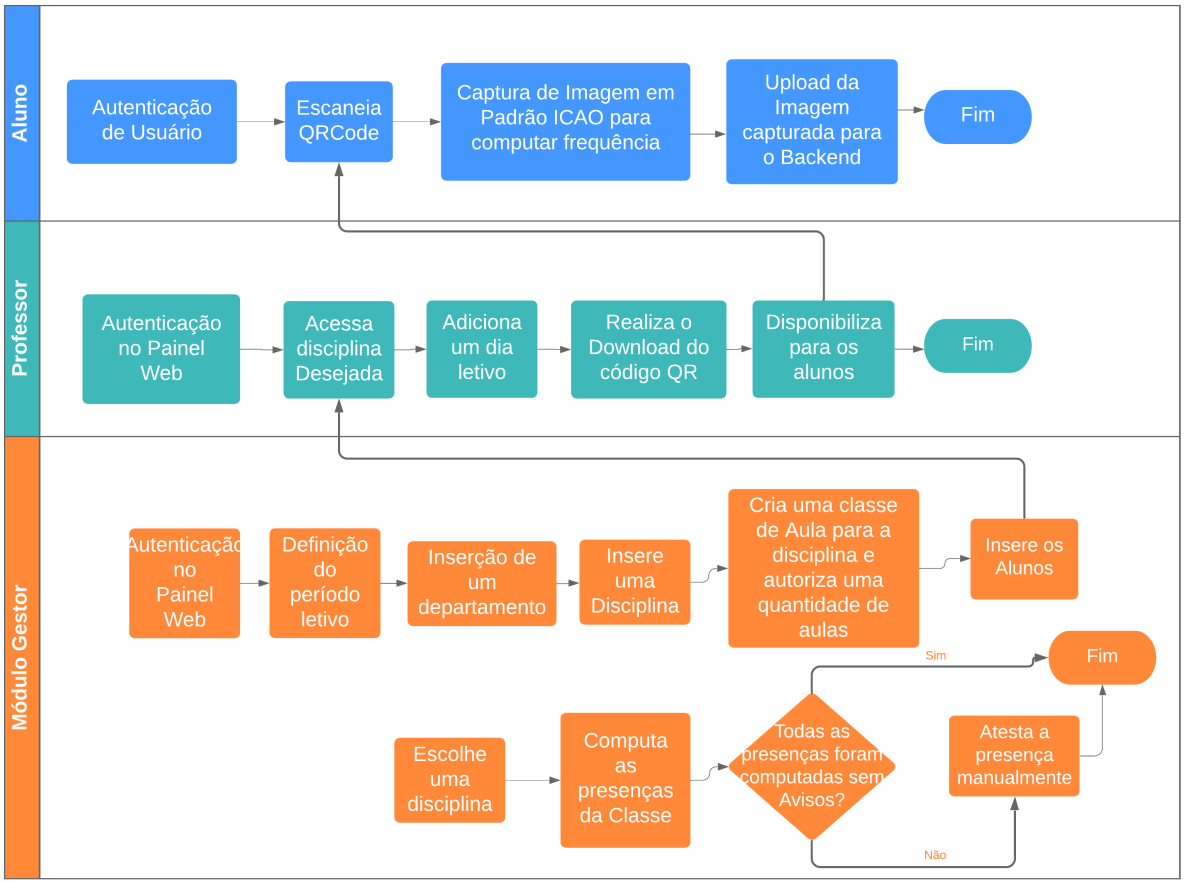


Figura 4.1: Modelo de Arquitetura Proposta.

4.2 Modelo de Arquitetura Proposta

A arquitetura proposta consiste na utilização de 3 plataformas diferentes, cada qual com suas responsabilidades e papéis atribuídos na organização da arquitetura, sendo elas:

- Móvel: Disponível para o aluno realizar a marcação da sua presença em aulas;
- Web: Disponível para o professor gerar seus dias letivos e para o Gestor realizar todo o gerenciamento organizacional;
- Servidor: Responsável por gerenciar os dados e implementar todos os algoritmos de reconhecimento facial e atestar as frequências.

Toda a arquitetura baseia-se num processo cliente-servidor [23], em que o servidor é responsável por prover todos os dados necessários para os seus clientes. Nesse caso, os clientes são o painel Web e os dispositivos móveis, o processo é feito baseando-se em boas práticas criptográficas, de modo a mitigar ataques e exposição de dados. Um exemplo básico é visto em 4.2.

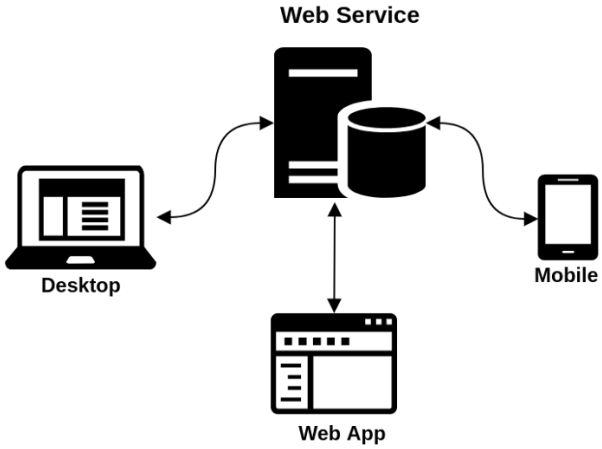


Figura 4.2: Modelo básico de Arquitetura Proposta.

De posse de todas as informações básicas de arquitetura temos, que o trabalho é coletivo dos 3 atores e pode ser explicado, sequencialmente, como a criação de todo o processo gerencial por parte do gestor. Bem como pela criação de dias letivos para uma classe no qual o docente disponibiliza um código de barras bidimensional (QR) aos alunos que, dentro do horário permitido para a computação da chamada, capturam o código gerado, realizam a captura de uma imagem de suas faces dentro do padrão estabelecido e enviam essa imagem ao servidor. Em um momento posterior, o gestor é o responsável por disparar, manualmente, o processo de computação de chamadas, tendo como resultado final a apuração das frequências em um dado momento ou, coletivamente, para todas os dias letivos.

O processo foi desenhado para que seja simples para todos os atores envolvidos, retirando, assim, a burocracia e o trabalho extra-classe, tornando o processo transparente e auditável, diminuindo a capacidade de erros humanos e introduzindo uma maior confiabilidade.

4.3 Módulo Gestor

O gestor representa o topo da hierarquia nesse processo, possuindo, assim, um maior número de atribuições e responsabilidades. Para definir-se um escopo que atenda a todas as necessidades de gerenciamento, ficaram definidas como requisitos para o gestor:

- Criação de Semestres Letivos;
- Criação de Departamentos
- Criação de Disciplinas
- Criação/Remoção de Classes para Disciplinas em um Semestre Letivo
- Disparar o processo de Computação de Frequência das Classes

O processo fica ilustrado na figura 4.3, notar-se-á todas as responsabilidades elencadas anteriormente.

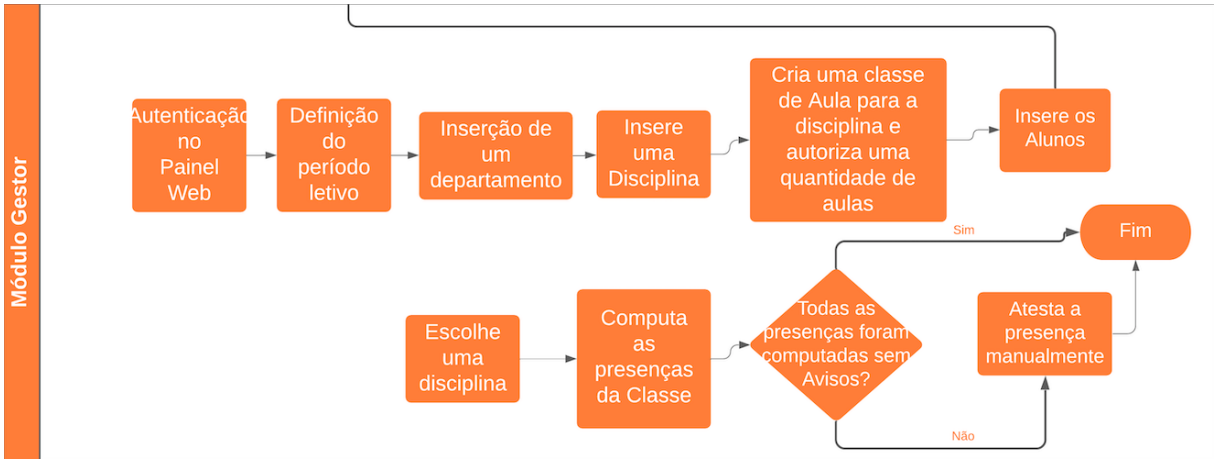


Figura 4.3: Representação gráfica do fluxo de atividades do ator Gestor.

Para que toda essa estrutura garanta um ator confiável, é realizado o processo de autenticação conforme mostrado em 4.6. Após a autenticação, todas as suas funcionalidades são devidamente liberadas.

Posteriormente, é necessária a criação da estrutura organizacional para que todas as associações possam ser feitas. Então, cria-se a organização de departamentos, que contém

professores e disciplinas, para que posteriormente sejam criadas classes de disciplinas com os alunos matriculados.

Após toda essa estrutura criada, se torna possível a utilização do sistema pelos outros atores, que têm suas responsabilidades definidas em 4.4 e 4.5.

4.4 Módulo Professor

O professor tem atuação hierárquica abaixo do gestor e acima do aluno e as suas funcionalidades foram pensadas de modo a atender suas demandas decorrentes do processo letivo, tanto em sala de aula, quanto em posterior gerenciamento e acompanhamento. Então, esse módulo traz como requisitos:

- Criação de dias letivos para presença
- Visualização de alunos matriculados
- Acompanhamento de frequência média
- Acompanhamento de frequência de um dia letivo específico

A criação de dias letivos compreende à definição de limites mínimos e máximos para a computação de presenças por parte dos alunos. Com isso, alunos fora do período autorizado não podem atestar presença, de modo a tornar o processo confiável dentro dos limites propostos pelo docente. Após a criação do dia letivo é disponibilizado um código QR contendo as informações necessárias para que um aluno possa, então, computar a sua presença, haja visto que este código deve ser fornecido para os alunos, que escanearão com os seus *smartphones* e atestarão presença no dia letivo.

O processo a ser feito pelo professor é ilustrado na Figura 4.4. O mesmo demonstra sua simplicidade e atendimento as demandas básicas de sala de aula e do seu acompanhamento.

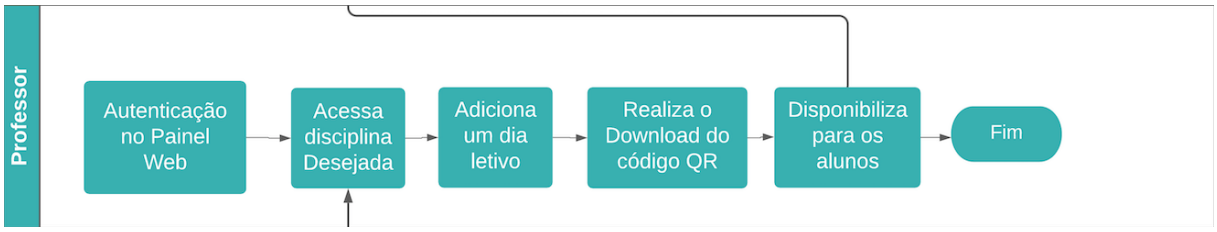


Figura 4.4: Representação Gráfica do Fluxo de Atividades do Professor

4.5 Módulo Aluno

O aluno, por sua vez é, o ator de nível mais baixo na hierarquia, sendo disponibilizado a ele, somente, a possibilidade de capturar o código QR fornecido pelo professor e realizar a captura da imagem de sua face para atestar uma presença. Deste modo, seu acesso é cedido por um aplicativo de *smartphone* disponibilizado sempre que necessário.

O processo de atestar a sua presença é muito bem ilustrado pela Figura 4.5, que mostra todos os seus passos de maneira sequencial.

Vale detalhar nesse processo os requisitos necessários para que seja atestada uma frequência, sendo eles a captura de um código válido e uma imagem de sua face, seguindo um subconjunto de regras definidas no ICAO 9303 [24] e implementadas em um trabalho prévio [25], o qual serviu como base para todo o procedimento aqui desenvolvido. Após a captura que garante o padrão de qualidade necessário para o processamento que será realizado, então, é feito o *upload* da imagem para o servidor que valida e então o procedimento é finalizado com sucesso para o aluno.

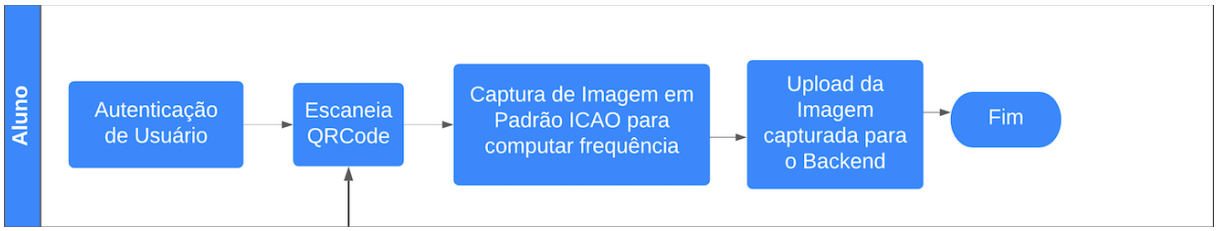


Figura 4.5: Representação Gráfica do Fluxo de Atividades do Aluno

4.6 Autenticação e Autorização

Com o grande avanço dos sistemas baseados em nuvens computacionais, a necessidade por meios de autenticação e autorização robustos torna-se cada vez mais necessária. Portanto, é uma área de grande estudo e com inúmeras técnicas bem difundidas.

Neste cenário, é evidente a necessidade de conhecimento dos diferentes meios de autenticação. Para a escolha de um bom meio de autenticação, faz-se necessário o entendimento de suas premissas, custo e aplicação. Diante do entendimento desses pontos, pode-se escolher um bom método de autenticação, de modo que se as premissas foram corretamente seguidas, o ator autorizado é de fato quem diz ser.

Partindo de todas as informações mostradas nos parágrafos anteriores, temos que levar em consideração o protocolo utilizado na construção do serviço de nuvem. Neste caso, é utilizada a arquitetura de Transferência Representacional de Estado(REST) [26], a qual tem como principal característica o fato de uma requisição ter qualquer tipo de

conexão com as outras, fazendo assim com que nenhum dado do cliente seja armazenado ou transferido entre requisições.

De posse da premissa arquitetural utilizada, temos que ter um sistema de autenticação de baixo custo e seguro, principalmente por ser uma proposta para validação do estudo aqui proposto. Inúmeras técnicas poderiam ser utilizadas, haja vista que a autenticação por biometria facial seria a alternativa mais viável, principalmente pelo escopo do trabalho, entretanto, deve-se ter em mente o custo de transferência de uma imagem entre as partes. Portanto, para tal foi escolhida uma autenticação HTTP Básica [27] utilizando usuário e senha.

Posteriormente, se seguíssemos apenas com a autenticação básica, teríamos que mandar estes mesmos dados em todas as requisições, tendo em vista que o protocolo REST não mantém estado do cliente. De modo a contornarmos esse problema, podemos utilizar um *token* de autorização, o qual é definido na RFC7519 [28].

A RFC7519 [28] define um formato autocontido e compacto para transmissão de segura da informação entre partes como um objeto JSON. A informação é confiável por ser digitalmente assinada, utilizando um segredo HMAC ou um par de chaves público-privada com algoritmos RSA ou ECDSA.

O *JWT* consiste em três partes separadas por ponto, sendo elas cabeçalho, carga e assinatura. O cabeçalho consiste em duas partes, onde a primeira é o tipo do *token* e a segunda o algoritmo de assinatura utilizado. A parte de carga é composta pelas afirmações sobre as entidades(geralmente sobre o usuário) e dados adicionais, em que essas afirmações podem ser privadas, públicas ou registradas. Por fim, a assinatura é criada por meio do cabeçalho codificado em Base64, carga codificada em Base64, um segredo e o algoritmo especificado no cabeçalho e, então, assinado. O resultado final é um texto codificado em Base64 separados por pontos como por exemplo a *xxxxx.yyyyy.zzzzz*.

Deste modo, temos um modelo simples e eficiente de autenticação seguido por autorização, sendo de fácil manuseio por todos os serviços que irão consumir as interfaces disponibilizadas.

Capítulo 5

Resultados

A construção da proposta apresentada no decorrer desse estudo baseou-se nos requisitos básicos com a estrutura e porte da Universidade de Brasília, fazendo assim com que a mesma fosse o ambiente indicado para testes de campo, com o qual seria realizado em momento posterior ao aperfeiçoamento dos algoritmos e arquitetura do sistema como um todo.

Contudo, vale ressaltar que todos os testes de campo foram suspensos em virtude da pandemia de Sars-Cov-2(Covid-19), a qual afeta o mundo desde o fim de 2019. Com isso, especialmente no Brasil, no momento da construção e apresentação desse trabalho todas as universidades encontram-se fechadas para evitar a propagação do vírus e contribuir para o contingenciamento deste problema. A reabertura da Universidade acontecerá somente quando o comitê gestor de crise entender que é seguro para todas as pessoas envolvidas, tal informações são divulgadas sempre que necessário no sítio do Comitê Gestor [29].

Desta feita, para a demonstração da funcionalidade e viabilidade da arquitetura proposta, optou-se para uma estrutura de apresentação por ablação, em que os módulos, onde a implementação construída é apresentada. Por fim, foi desenvolvido um vídeo demonstração para que seja apresentado todo o funcionamento da arquitetura proposta, este disponível no seguinte endereço web Video - Apresentação Prática, bem como todas as rotas disponibilizadas em formato de coleção no endereço web Rotas para o Postman.

Na subseção 5.2 é detalhado todo o funcionamento do sistema em nuvem implementado, bem como a construção do seu banco de dados e da sua execução integral. Em seguida, na subseção 5.3 é detalhada a implementação do sistema web construído para ser utilizado por professores e gestores. Por fim, na subseção 5.4 é caracterizada a construção da aplicação móvel que objetiva o uso de alunos para atender a frequências.

5.1 Tecnologias empregadas no desenvolvimento da Implementação

Todas as tecnologias foram escolhidas com base em critérios importantes, os quais são mostrados a seguir.

- Documentação disponível;
- Difusão na comunidade de desenvolvimento;
- Estar ativamente manutenido;
- Compatibilidade com as plataformas;

Ditos os motivos que embasam as escolhas de tecnologias, temos então que as escolhidas são expostas na Tabela 5.1.

Tecnologia	**Versão**	**Módulo**	**Justificativa**
OpenCV[16]	3.4.14	*Back-end* e Móvel	Utilização dos algoritmos de processamentos de imagens e reconhecimento facial
Node.js [30]	14.16.1	*Back-end*	Criação dos módulos do *back-end*
Koa [31]	7.6.0	*Back-end*	Adição de funcionalidades de servidor Web ao Node.js
opencv4nodejs [32]	5.6.0	*Back-end*	Adição de interface mais amigável do OpenCV para Node.js
Swift [33]	5.4	Móvel	Criação de aplicações para iOS
Angular [34]	11.2.11	*Front-end*	Criação de painel web
Visual Studio Code [35]	1.5.3	*Front-end* e *Backend*	IDE de desenvolvimento utilizada no *Front-end* e *Back-end*
Xcode [36]	12.5	Móvel	IDE para desenvolvimento de aplicativos para iOS
PostgreSQL [6]	13.2	*Back-end*	Armazenamento de dados

Tabela 5.1: Tecnologias utilizadas para a construção de cada módulo

5.2 Implementações de *Back-End*

O módulo de *back-end* foi construído para ser o ponto de sincronia entre as aplicações Web e móvel, sendo então responsável por armazenamento e processamento de todos

os dados gerados. Por consequência, é o designado para a execução de algoritmos de reconhecimento facial e atestar as frequências dos alunos.

Para que o objetivo seja alcançado foram escolhidas tecnologias altamente difundidas, como descrito em 5.1. Portanto, os resultados de construção podem ser mostrados seguindo uma abordagem *bottom-up*, onde será apresentado o desenvolvimento do banco de dados e seus relacionamentos até que seja mostrada a interface programável que é exposta para o consumo do cliente.

Seguindo a abordagem citada, temos que o armazenamento dos dados é feito utilizando a *Standard Query Language(SQL)*, a qual é implementada pela ferramenta PostgreSQL, que é *open-source* e amplamente utilizada em projetos comerciais. Portanto, os requisitos são modelados e normalizados seguindo a terceira forma normal, de modo a termos um diagrama entidade relacionamento que atenda as necessidades do projeto. O diagrama é mostrado em 5.1.

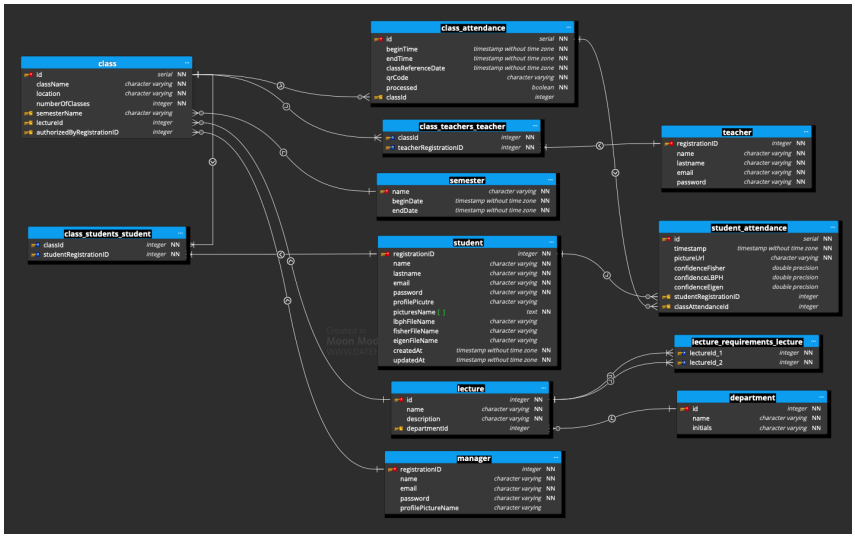


Figura 5.1: Diagrama entidade-relacionamento implementado em banco de dados Postgresql [6]

Analisando-se o diagrama gerado, devemos olhar atentamente para a estrutura organizacional construída, na qual notamos a replicação do modelo atual utilizado pela Universidade de Brasília. Vale notar que as relações de maior importância são as das tabelas "professor", "chamada"e "aluno", onde elas representam o *core* da solução proposta e todas as outras estruturas podem ser adaptadas de acordo com a implantação.

Notamos que os professores são associados a um departamento(*department*) e esse contém disciplinas(*lecture*). Cada uma dessas apresenta classes(*class*) ou turmas, as quais são, individualmente, associadas a um semestre(*semester*) e a 1 ou mais professores(*teacher*). Por fim, turmas contêm alunos(*students*) e dias letivos(*class_attendance*) onde os alunos podem enviar as suas solicitações de presenças(*student_attendance*).

Partindo-se para a parte de criação de tabelas e acesso ao banco de dados, foi utilizado um mapeador objeto-relacional(ORM) para simplificar o trabalho e melhorar a qualidade de manutenção, o escolhido para este caso sendo o TypeOrm [37]. As anotações de código e configurações de acesso a banco, corretamente definidas no arquivo *ormconfig.json*, fazem com que o *framework* seja capaz de acessar corretamente e criar todas as tabelas e propriedades necessárias, inclui-se no processo de gênese de relações.

Todas as informações para a construção do banco de dados ficaram contidas no diretório *data/entities*, sendo que cada arquivo define, define uma tabela no banco de dados e suas anotações provêm informações suficientes acerca das relações que devem ser construídas no momento da execução do servidor.

De posse de todo o banco construído e devidamente estruturado, são arquitetados serviços que consomem os bancos de dados ou outros para a construção de lógicas de negócios. A maior relevância é dada ao serviço de processamento de imagens (*imageProcessing.service.ts*), o qual é consumido por outros serviços, mas, principalmente, pelo serviço de gestão (*manager.service.ts*), que é responsável por realizar o pedido de computação de presença dos alunos de uma determinada classe, sendo necessário apenas o envio do identificador de uma turma ou de um dia letivo para que se obtenha o resultado.

O processo de reconhecimento facial é realizado 3 vezes para que diferentes algoritmos apresentem seus resultados, dando uma maior confiabilidade ao processo. Os algoritmos utilizados são o Fisherfaces [12], EigenFaces [10] e o LBPH [11], haja visto que cada um desses algoritmo tem as suas particularidades conforme apresentado em 2.1.

Com a utilização de múltiplos algoritmos podemos validar, com testes de campo, qual traz o melhor resultado de acordo com as necessidades que serão apresentadas, vale lembrar que os testes em larga escala não foram possíveis por conta da pandemia do Sars-Cov-2.

Diferentemente do que é realizado em trabalhos tradicionais, em que são realizados treinamentos para todo o conjunto de pessoas nos algoritmos, optamos por treiná-los e armazenar os resultados para cada pessoa separadamente, então, ao invés de esperar que o algoritmo diga com quem a pessoa é mais similar, temos que ele nos dirá o quanto a pessoa se parece com quem diz ser e, a partir daí, é definido um limiar de aceitação.

Após o cálculos das devidas distâncias, todos os resultados são persistidos no banco de dados na tabela *student_attendance* e armazenados nas propriedades *confidenceFisher*, *confidenceLBPH*, *confidenceEigen*, as quais são recuperadas a posteriori e exibidas para o gestor, conforme descrito em 5.3.

Seguindo a abordagem, temos que outro serviço importante é o de autenticação, em que é autenticado e autorizado um dado usuário. Seguindo os padrões previamente descritos em 4.6, todas as senhas são armazenadas durante o processo de cadastro com um *hash*

gerado após a concatenação da senha com um *salt* de tamanho 10.

Na autenticação, são buscados usuários com o identificador fornecido e são comparada as senhas fornecidas com a armazenadas. Caso a autenticação ocorra com sucesso, é retornado ao usuário um *token* seguindo o padrão JWT [28], sendo que no *payload* são adicionadas informações extras como o papel do usuário e suas informações básicas. Um exemplo de retorno é mostrado em 5.1. Como dito em 4.6, o *token* JWT é gerado utilizando um par de chaves criptográficas e padrão *Hashed Message Authentication Mode (HMAC)* [38] em combinação com *significa secure hash algorithm(SHA)* [39] de 64 Bytes, mais conhecido como SHA512. Segundo o pontuam em 4.6 nas requisições posteriores à autenticação, é necessário que o *token* fornecido seja enviado para que haja a autorização do recurso a ser utilizado, o qual é baseado no *token* e no papel do usuário.

Listing 5.1: Exemplo de retorno para autenticação de usuário

```
1  {
2      "token": "eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.
          eyJ1c2VySWQiOjEsInJvbGVzIjpbIk1BTkFHRVIiXSwiaWF0IjoxNjE5NzMwMDEyI
          .SOv94zacYRzmhjtw9FeuoF5K2eHxABtogGaXU-
          AepHMx9GkaIyFADPlHfKK4U3G5WTzXCLYM1Q7tCRSwRHEplw",
3      "role": "MANAGER",
4      "user": {
5          "registrationID": 1,
6          "name": "Kelvin",
7          "email": "kelvinwml@gmail.com",
8          "profilePictureName": null
9      }
10 }
```

Além dos serviços de maior importância detalhados nos parágrafos anteriores, temos os serviços que foram desenvolvidos para realizar os processos de criação, recuperação, atualização e deleção(CRUD) das informações, que endossam as regras de negócios da aplicação e são utilizados pela camada de controle, os quais são comentados em 5.2.

- class.attendance.ts: Responsável por realizar o CRUD das frequências de uma dada classe;
- class.service.ts: Responsável por realizar o CRUD das informações de uma classe;
- department.service.ts: Responsaável por realizar o CRUD das informações de um departamento;
- file.service.ts: Responsável por realizar o CRUD de arquivos;

- lecture.service.ts: Responsável por realizar um CRUD das informações de uma disciplina;
- manager.service.ts: Responsável por realizar um CRUD das informações de um gestor, além de realizar trabalhos de gestão como requisição de processamento de frequência ou estatísticas de uma turma;
- semester.service.ts: Responsável por realizar um CRUD das informações de um semestre letivo;
- student.attendance.service.ts: Responsável por realizar o CRUD das frequências computadas por um estudante;
- student.service.ts: Responsável por realizar o CRUD das informações de um estudante;
- teacher.service.ts: Responsável por realizar o CRUD das informações de um professor;

Por fim, chegamos à última camada, a qual é de controle responsável por definição de rotas que podem ser acessadas por quem consome o serviço. Por simplicidade, todas as rotas foram definidas em um único arquivo chamado *server.ts*, o qual tem como responsabilidades adicionais a configuração da aplicação.

Como adição à camada de controle, temos a inserção de *Middlewares*, que são aplicações posiconadas no intermédio entre o recurso desejado e o cliente. Neste caso, são adicionados *Middlewares* para checagem de autorização, suporte a requisições com formato de comunicação *multipart/formdata*, além de um último responsável por interceptar todas as condições de erros e as retornarem no formato desejado.

Dito isto, temos que algumas definições devem ser ditas de modo a não causar confusão. Toda a representação de informações é feita utilizando-se o formato JSON [40], por ser mais simples que outras alternativas como XML, além do melhor suporte provido em todas as linguagens utilizadas no escopo desse trabalho.

Posteriormente às validações providas pelos *Middlewares*, todas as rotas que recebem parâmetros passam por validação para que seja garantida com exatidão os seus tipos, bem formados, além de validade do parâmetro de acordo com regras específicas, como pode ser o caso de endereços de e-mail. Para que este objetivo seja alcançado, foi utilizado o *framework Joi.js* [41], tido como uma das melhoras ferramentas de validação e de larga utilização em projetos comerciais. As regras são construídas de maneira simples, como exemplificado em 5.2

Listing 5.2: Exemplo de validação de parâmetros utilizando Joi [41]

```
let schema = Joi.object({
        id: Joi.number().min(0).required(),
        name: Joi.string().required(),
        lastname: Joi.string().required(),
        email: Joi.string().email().required(),
        password: Joi.string().min(4).max(32).required()
    })

    let { error } = schema.validate(ctx.request.body);
```

Em seguida, é válido mencionar a organização das rotas que são agrupadas de acordo com as classes sobre as quais as operações são realizadas. Por exemplo, pode-se obter todos os departamentos por meio de uma requisição do tipo *GET* ao *endpoint /department*, da mesma forma, pode-se obter as informações de um departamento em específico, passando-se um parâmetro no corpo da requisição com o nome *id*.

Para que fosse facilitado todo o processo de construção de requisitos a serem utilizados nos testes, foi então criada uma coleção para o *software* Postman [42] afim de que essa pudesse ser facilmente exportada e importada por outras pessoas que tenham interesse em utilizá-la.

5.2.1 Execução

Para que seja executado o serviço de *back-end*, é necessário que seja fornecida uma chave secreta a ser utilizada na criação e validação dos *tokens* JWT, além das criação de credenciais para acesso ao banco de dados.

A primeira pré-etapa a ser realizada é a obtenção de uma chave simétrica que seja usada na criação e assinatura dos *tokens* JWT [28], a qual é mostrada em 5.3.

Listing 5.3: Criação de chave simétrica.

```
$ openssl rand 512 | base64
```

Por fim, devemos garantir que todas as dependências do projeto sejam corretamente instaladas antes de executarmos a aplicação, para tal, são necessárias duas etapas, em que a primeira etapa é a instalação do OpenCV 4.5.2 que pode ser feita como mostrado em 5.4.

Listing 5.4: Instalação do *framework* OpenCV [16] em sitema operacional mac

```
$ brew update
$ brew install opencv@3
```

```
$ brew link --force opencv@3
```

Posteriormente, devemos desabilitar o *autobuild* da biblioteca OpenCV4NodeJs [32] para que a mesma, corretamente, referencie o OpenCV [16]. A execução desse processo é mostrada 5.5.

Listing 5.5: Comando a ser executado para desabilitar o *autobuild* do OpenCV [16].

```
$ export OPENCV4NODEJS_DISABLE_AUTOBUILD=1
```

Em seguida, precisamos garantir que todas as dependências do projeto sejam corretamente configuradas, para tal, basta executarmos o comando exibido em 5.6.

Listing 5.6: Instalação de dependências Node.js [30] do OpenCV [16].

```
$ npm install --save
```

Por fim, necessita-se garantir que a conexão com o banco de dados funcione corretamente, para tal é necessária a criação de um usuário que forneça acesso a um conjunto limitados de funcionalidades, mitigando, assim, possíveis problemas por má utilização da aplicação. Ademais, tem-se que o padrão de criação aqui utilizado deu-se pelos comandos mostrados em 5.7

Listing 5.7: Criação das credenciais para acesso ao banco de dados

```
1 $ create database db_attendances_face;
2 $ create user db_access with encrypted password '_uc;3z!.d
  /\Q,/xr';
3 $ grant all privileges on database db_attendances_face to
  db_access;
```

Após executados todos os passos, podemos então executar a aplicação *back-end*, de modo a tê-la rodando de maneira correta por meio do comando exibido em 5.8

Listing 5.8: Execução do serviço back-end

```
$ nodemon src/server.ts JWT_SECRET=<CHAVE_SECRETA>
```

5.3 Implementações de *Front-End*

O módulo *front-end* traz o consumo por professores e gestores, trazendo a possibilidade de criação de disciplinas, turmas e da apuração de frequências por parte do gestor e a criação de dias letivos para classes por professores, além de algumas estatísticas.

Para que o objetivo seja alcançado, tendo por base todas as premissas definidas no módulo *back-end* descrito na seção 5.2, o usuário é primeiramente conduzido à tela de login, na qual o mesmo entra com as suas credenciais de acesso e, então, é autenticado. A tela é exibida na Figura 5.2.

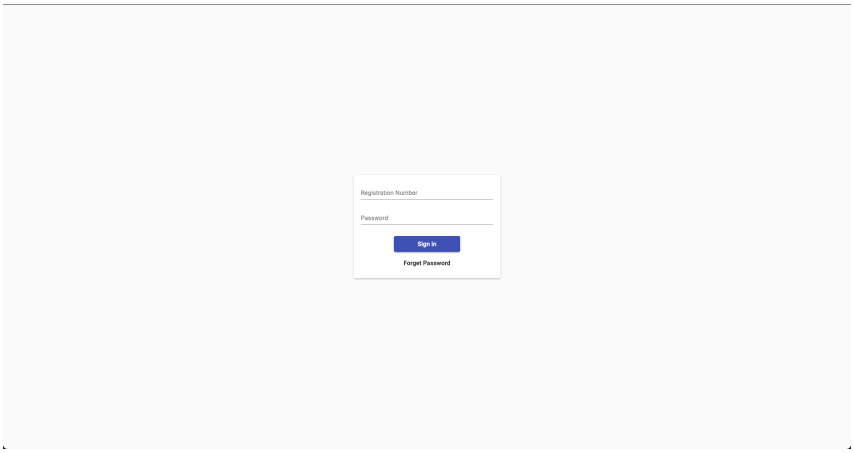


Figura 5.2: Página de autenticação

Após o processo de autenticação, é utilizada a resposta para determinação do papel que o usuário autenticado ocupa na organização, com isso a tela seguinte a ser apresentada é diferente a depender do papel. Para o professor é apresentada a tela mostrada na Figura 5.3b e para o gestor é apresentada a tela na Figura 5.3a.

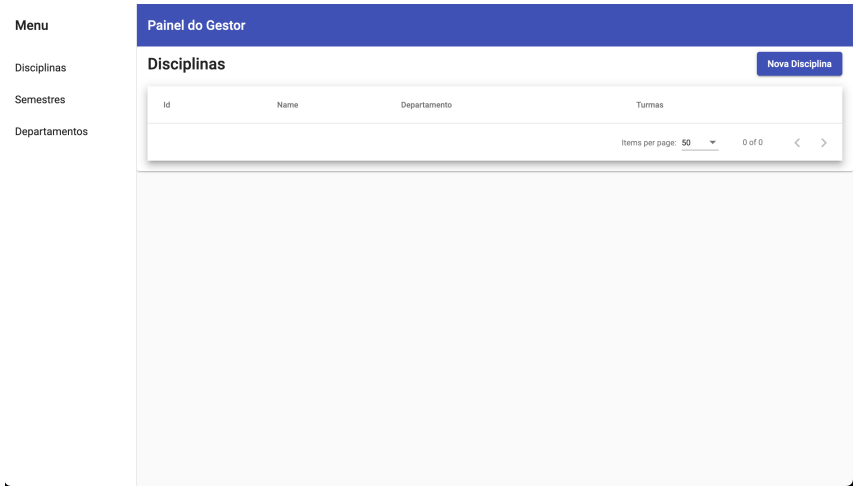
5.3.1 Gestor

Seguindo primeiramente o caminho para o papel de gestor, por ser o papel com maior nível de responsabilidades, além de ser fundamental para a criação de toda a estrutura organizacional e administrar integralmente o processo de criação de turmas e apuração das mesmas.

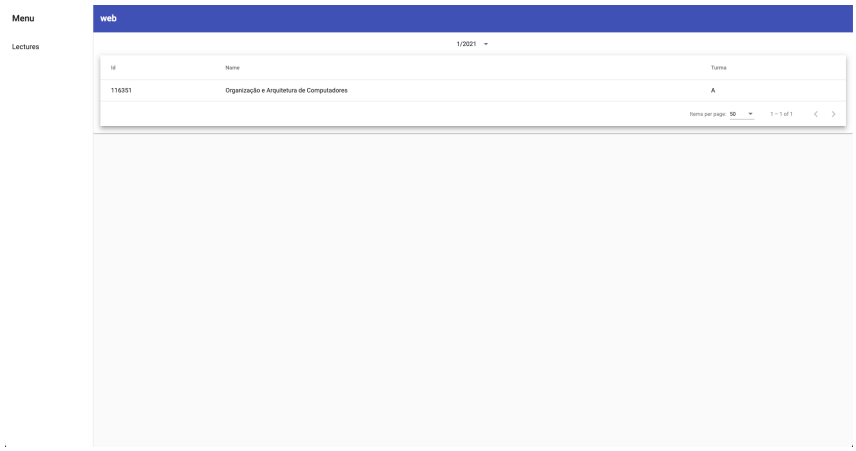
Tendo que segundo as regras organizacionais, todas as disciplinas pertencem a departamentos, portanto, o primeiro passo é a criação do mesmo, que pode ser feito pela opção "departamentos"na barra lateral, a qual exibe então a tela mostrada na Figura 5.4 que exibe uma listagem com todos que estão disponíveis.

Nota-se, na imagem, que no lado direito da parte superior existe um botão para a criação de um novo departamento. Ao ser clicado, o botão exibe um formulário com as informações básicas que devem ser preenchidas para a criação de um departamento. A tela é mostrada emna Figura 5.5.

Após a criação do novo departamento, a página de formulário some e a lista de departamentos é atualizada, conforme mostrado na Figura 5.6.



(a) Página inicial pós-autenticação para o papel de gestor



(b) Página inicial pós-autenticação para o papel de professor

Figura 5.3: Exibição de páginas pós-autenticação para gestor e professor.

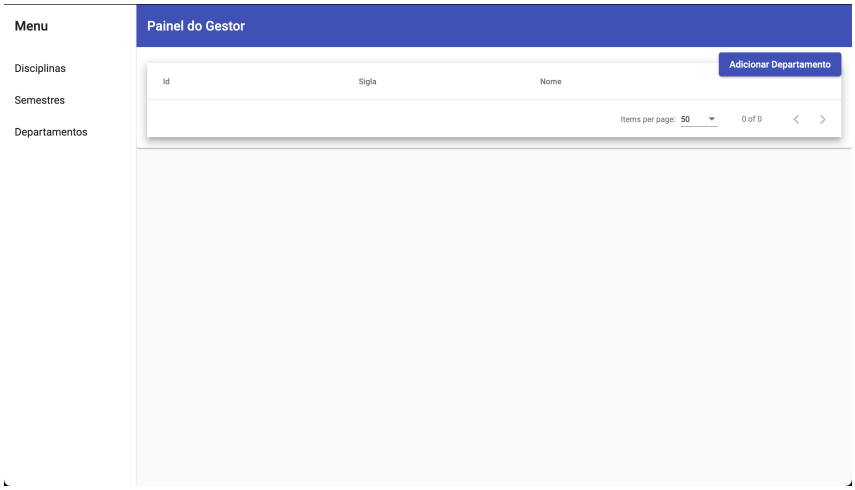


Figura 5.4: Página com a listagem de departamentos

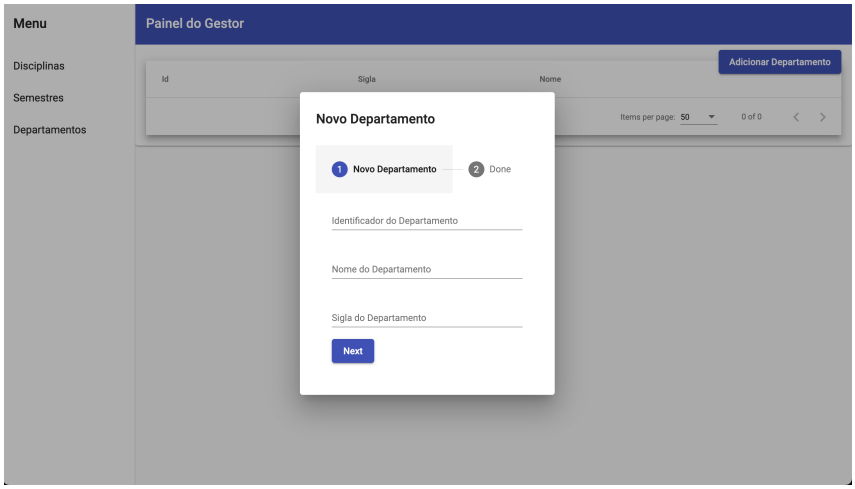


Figura 5.5: Página com formulário para a criação de um novo departamento

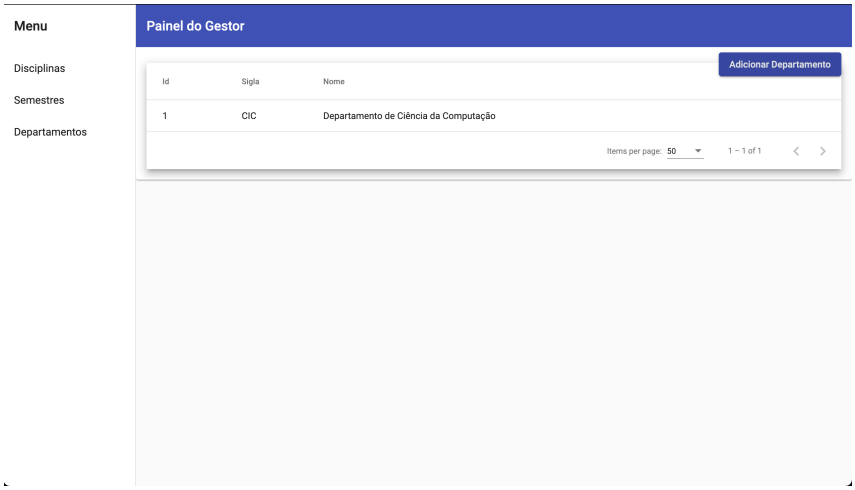


Figura 5.6: Página com listagem de departamentos preenchido

Em momento posterior, é necessária a inclusão de um novo semestre letivo para que sejam criadas classes de disciplinas, para tal, a opção semestres no menu lateral é utilizada, a qual exibe uma listagem das mesmas. Seguindo o padrão adotado, existe um botão no canto superior do lado direito para essa adição, como mostrado na Figura 5.7.

Em seguida, é apresentado o formulário para a criação do novo semestre que conta com data de início e data de fim, além de um identificador para o mesmo, conforme apresentado na Figura 5.8.

Após o preenchimento, também é recarregada a listagem que exibe os semestres e suas informações básicas em formato de cartão, conforme exibido na Figura 5.9.

Por fim, restou-se apenas a criação de professores para o cumprimento de todos os pré-requisitos e criação de uma disciplina, entretanto, por se tratar de um cargo foi deixada uma rota para criação de professores com suas respectivas credenciais de acesso. A rota(*POST /teacher*) está disponível para teste através da coleção disponibilizada na

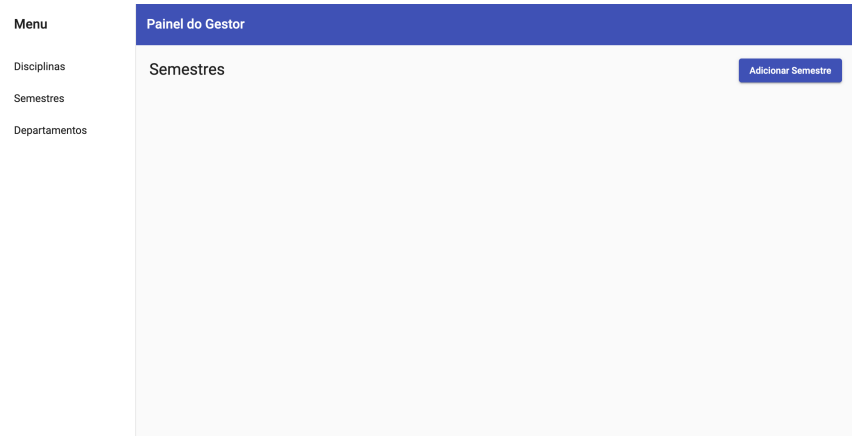


Figura 5.7: Listagem de semestres letivos.

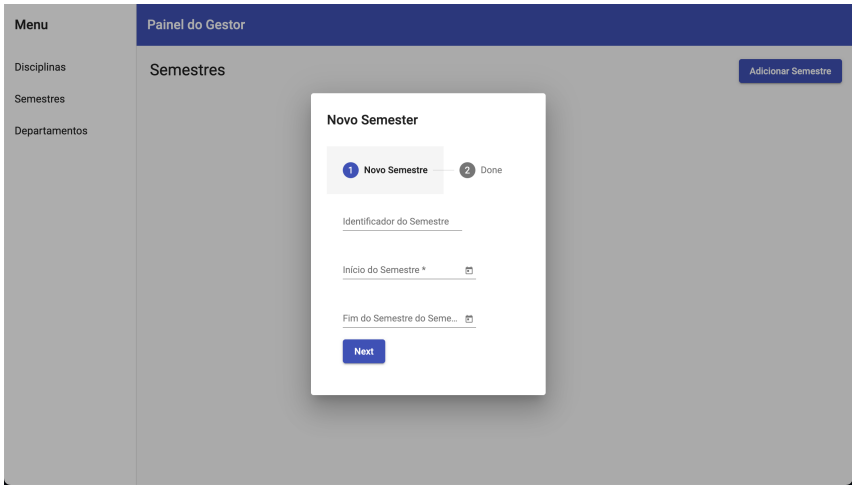


Figura 5.8: Listagem de semestres letivos.

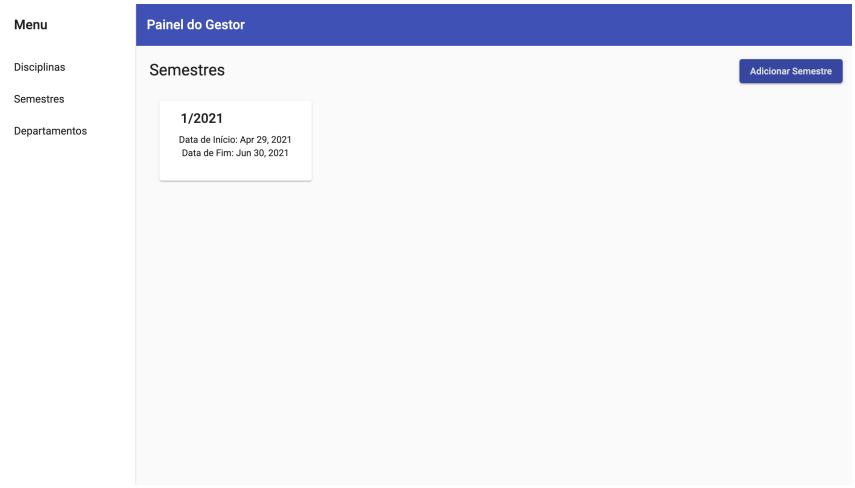


Figura 5.9: Listagem de semestres letivo com exemplo.

seção 5.2.

A criação de alunos é feita através do módulo implementado para sistema móvel e

é detalhado na seção 5.4. Entretanto é papel do gestor adiciona-lo ou removê-lo em disciplinas, processo o qual é detalhado nos parágrafos a seguir.

Para a criação de disciplinas, devemos utilizar a opção na barra lateral correspondente, o qual apresenta uma listagem de todas as disciplinas cadastradas no sistema com a quantidade de turmas ativas para o semestre corrente, e acessarmos o botão posicionado no canto superior do lado direito. A tela é ilustrada na Figura 5.10.

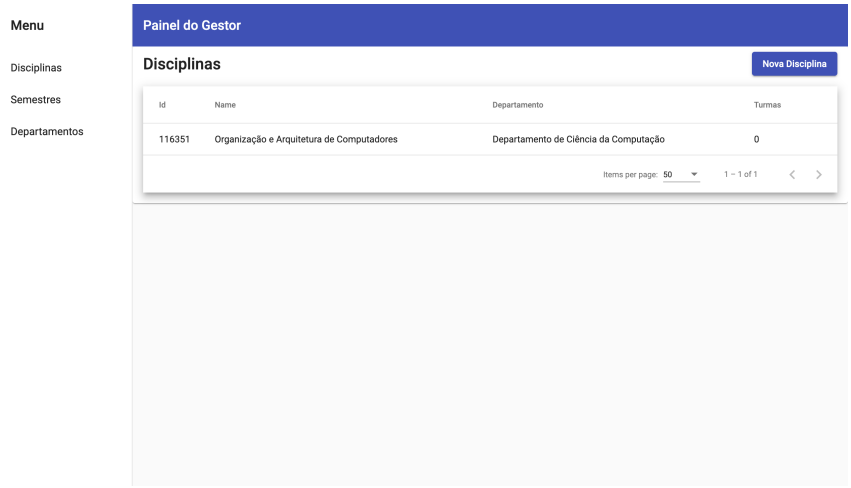


Figura 5.10: Exemplo de Listagem de disciplinas

Após o clique no botão, é apresentado o formulário para adição de uma nova disciplina, como mostrado na Figura 5.11. Pré-requisitos podem ser adicionados por intermédio dos seus códigos os quais são preenchidos no formulário em forma de bolhas e, após a criação com sucesso, a lista é atualizada com a nova disciplina conforme mostrado na Figura 5.9.

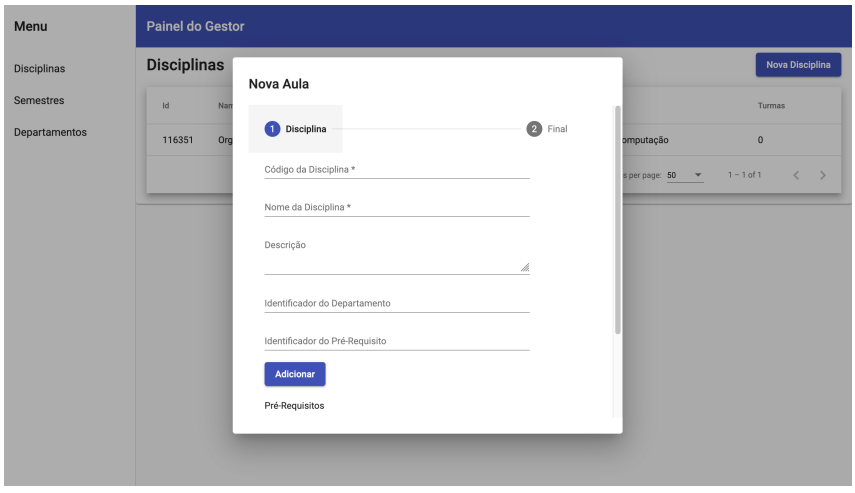
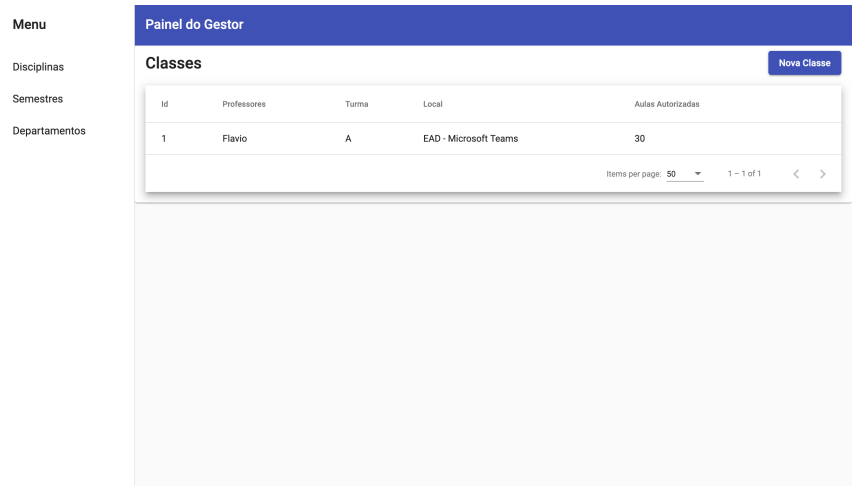


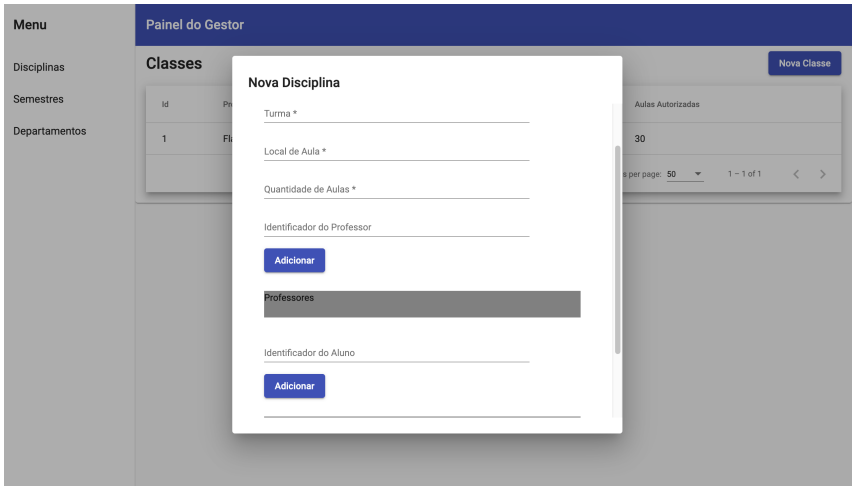
Figura 5.11: Formulário para inserção de uma nova disciplina

Ao selecionarmos uma disciplina, verificamos as turmas para o semestre em vigor, conforme mostrado na Figura 5.12a. Seguindo o mesmo padrão, é possível adicionar

uma disciplina por meio do botão localizado na parte superior e no canto direito. Logo é exibido o seu formulário conforme mostrado na Figura 5.12b, onde é obrigatório o fornecimento do identificador de um ou mais professores, sendo possível adicionar alunos nesta etapa ou em um momento posterior.



(a) Listagem de classes para uma disciplina previamente escolhida



(b) Formulário para inserção de uma nova classe

Figura 5.12: Operações sobre classes no painel administrativo.

Por fim, ao clicarmos em uma classe, são mostradas informações básicas em formato de painel administrativo, algo exemplificado na Figura 5.13, que contém número de alunos matriculados, frequência média para as aulas já ministradas e a quantidade de aulas ministradas versus a quantidade de aulas autorizadas pelo gestor. Logo abaixo, é mostrada a lista de aulas dadas. Ao clicar-se no botão "processar aulas", é realizada a contabilização de frequência para todas as que aulas já ministradas.

Por fim, o último nível de particularidades disponível é a possibilidade de detalhes acerca de um dia letivo específico, a partir do qual puderam ser vistos os alunos que

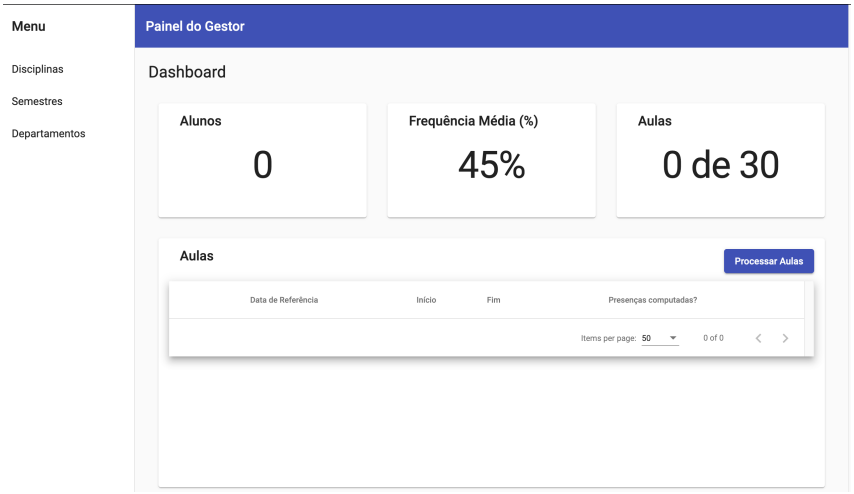


Figura 5.13: Detalhes de uma classe.

enviaram as frequências, bem como os índices dos algoritmos de reconhecimento facial obtidos, conforme mostrado em 5.14.

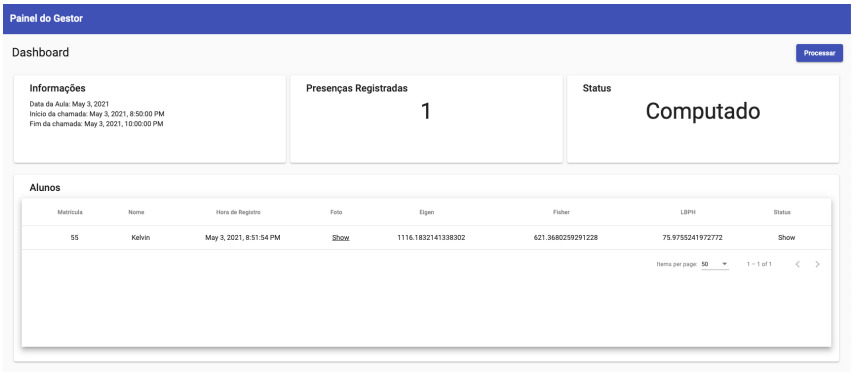


Figura 5.14: Detalhamento de um dia letivo para uma classe

Outro detalhe possível é a listagem de alunos por disciplina, que fica disponível na tela ao interagir com o cartão "Alunos", bem como a possibilidade de mostrar sua foto,, além de adicioná-los e removê-los. A tela é mostrada na Figura 5.15.

5.3.2 Professor

Após toda a construção organizacional por parte do gestor, podemos detalhar o fluxo disponível para que um professor veja suas disciplinas e crie dias letivos nos quais seus alunos computem suas presenças.

Para que isso seja possível, após o gestor criar uma classe, vincular o professor e seus alunos, a mesma fica disponível para o professor seguida da tela de autenticação, conforme mostrado na Figura 5.16. Vale notar, assim que um professor pode ver suas

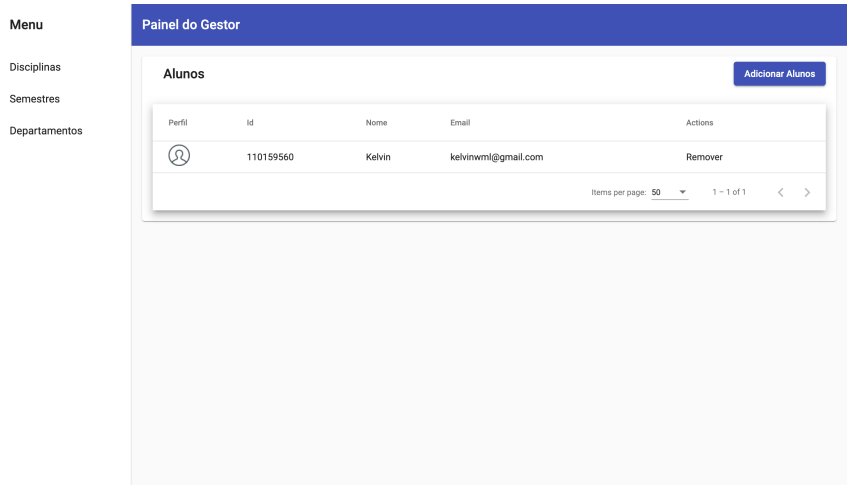


Figura 5.15: Detalhes de alunos de uma classe

classes ministradas em semestres anteriores, facilitando a comparação de estatísticas com o passar do tempo.

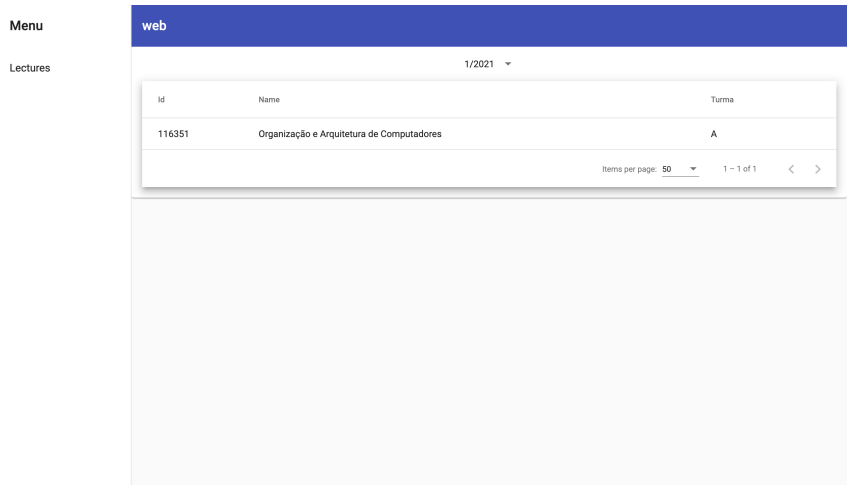


Figura 5.16: Classes ministradas por um professor em um dado semestre letivo

Diante da escolha de uma classe, são exibida informações básicas acerca da mesma. Assim como no painel disponível para o gestor, é possível a visualização da quantidade de alunos matriculados, frequência média e quantidade de aulas ministradas versus autorizadas, além da listagem de aulas já criadas. Adicionalmente, ao invés da presença de um botão para que sejam processadas as frequências, conforme exibido na Figura 5.14, temos um botão para criação de um novo dia letivo. Os detalhes podem ser visualizados na Figura 5.17.

Ao ocorrer a interação com o botão de nova aula, é exibido o formulário para a criação da mesma, conforme mostrado nas Figuras 5.18a e 5.18b, nas quais são exigidos campos como data da aula e restrições para as que as presenças possam ser feitas pelos alunos.

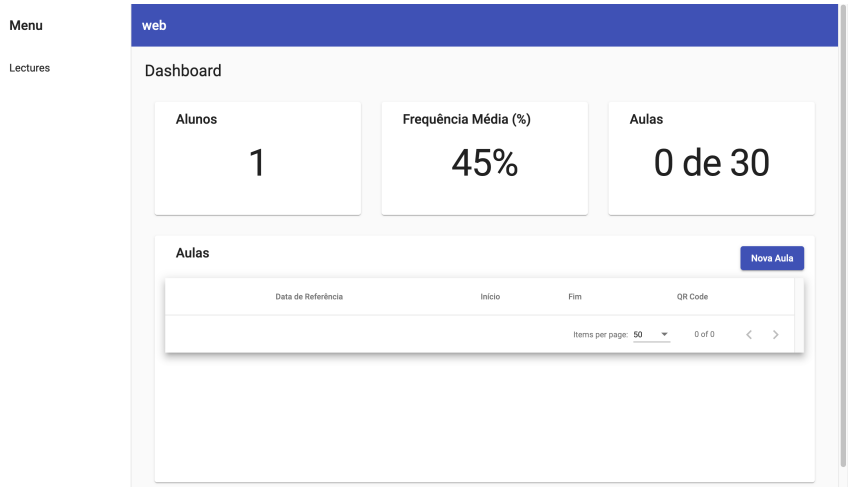
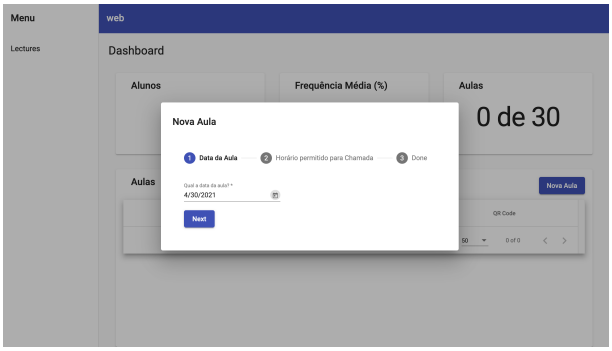
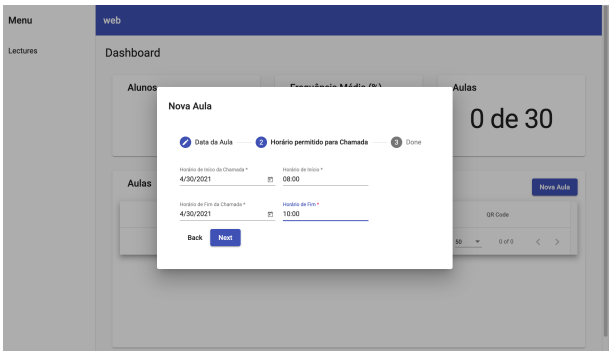


Figura 5.17: Detalhes painel administrativo de uma classe para o professor

Consecutivamente, essas restrições seguem o formato dia/hora, principalmente por conta do modelo "online", que é amplamente utilizado na situação atual.



(a) Formulário para a criação de uma nova aula



(b) Formulário para a criação de uma nova aula

Figura 5.18: Exemplo de operações sobre aulas realizadas por um professor em painel administrativo

Após a criação de um dia letivo, fica disponível na tabela de aulas o código QR para que o professor possa salvá-lo e exibi-lo aos alunos da forma que for conveniente, conforme mostrado na Figura 5.19.

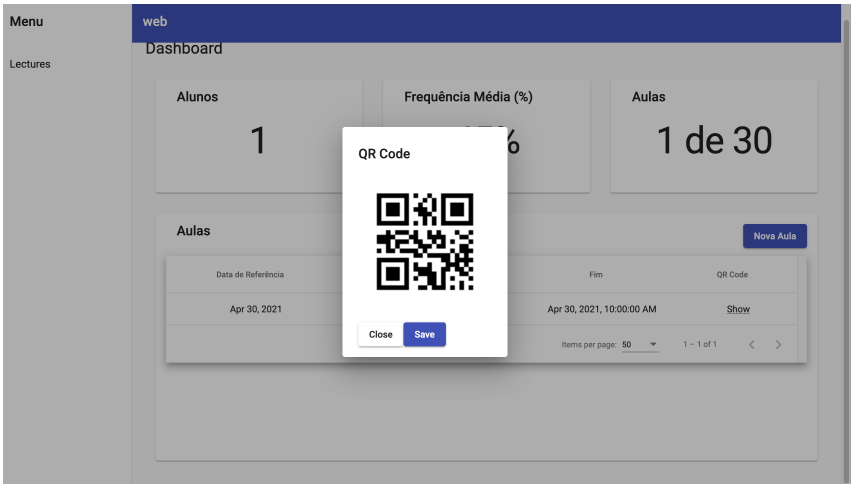


Figura 5.19: Exemplo de código QR gerado para que alunos atendam a chamada

Seguindo o detalhamento do painel, similarmente ao que é exibido para o gestor, é possível a visualização de alunos e suas fotos, como mostrado na Figura 5.20, porém é vedada ao professor a permissão para remoção de alunos de sua turma, algo de responsabilidade do gestor.

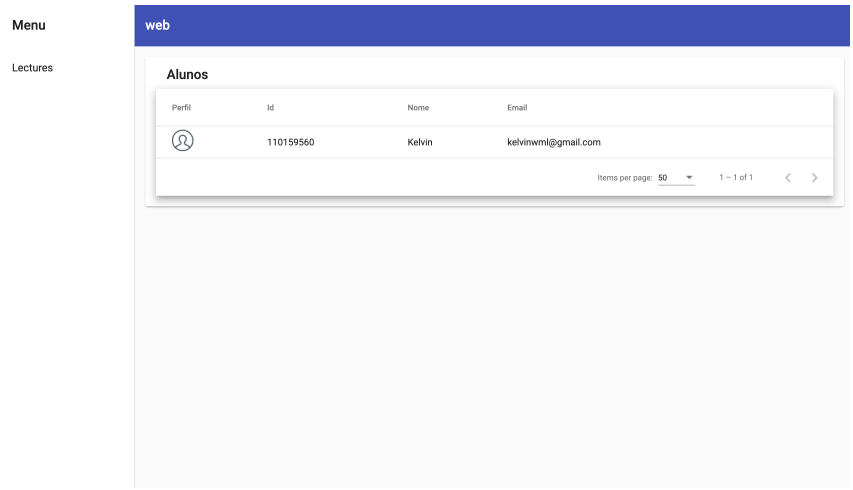


Figura 5.20: Formulário para a criação de uma nova aula

Por fim, ao professor selecionar um dia letivo é são exibidas as suas estatísticas, para o qual podemos ver um exemplo na Figura 5.21. São mostradas as informações básicas como dia letivo e restrições para a chamada, além de, quantidade de presenças registradas para a turma e o status que informa se o gestor já computou ou não aquele dia letivo.

Dessarte a criação de toda a estrutura organizacional e de dias letivos por parte de um professor, resulta, apenas que alunos computem suas presenças, o qual é feito por meio do módulo desenvolvido para dispositivos móveis e detalhado na seção 5.4.

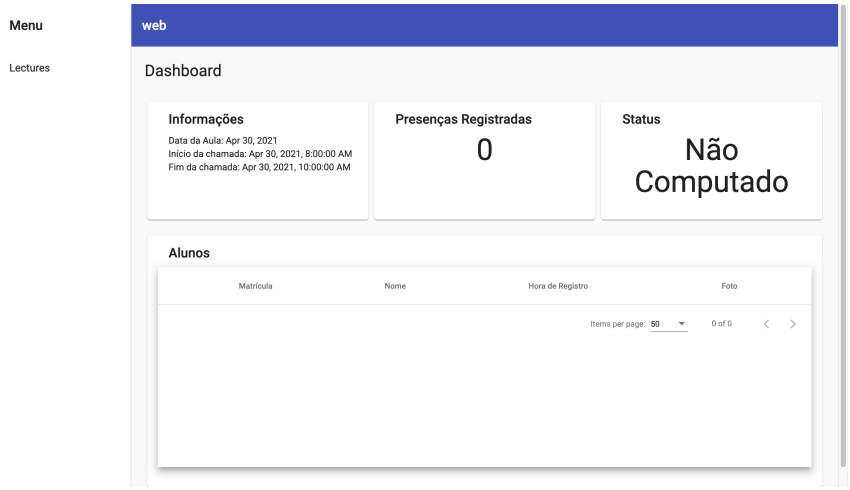


Figura 5.21: Painel administrativo para um dia letivo

5.4 Implementação Móvel

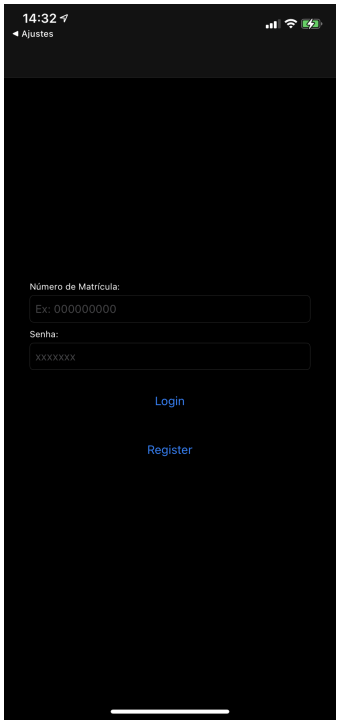
O módulo destinado à aplicação móvel foi criado para que os alunos possam computar suas presenças em dias letivos, de modo que as imagens capturadas no processo atendam aos requisitos necessários para que sejam processadas por algoritmos de reconhecimento facial.

Para que o objetivo seja alcançado, foi utilizado o conjunto de regras detalhados em 4.5 e, então, construída toda a interface de usuário, de modo que os requisitos sejam garantidos, além da comunicação com o serviço de *back-end* mostrado na seção 5.2.

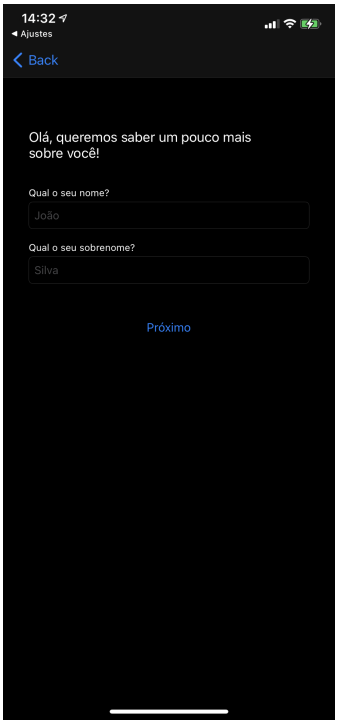
Inicialmente, assim como o gestor e o professor, o aluno se depara com uma tela para autenticação mostrada na Figura 5.22a, em que devem ser fornecidos seu número de matrícula e senha, caso o mesmo já tenha se cadastrado. Mediante necessidade de registro, esse deve tocar no botão de criação de conta e seguir o fluxo determinado, tendo em vista o preenchimento de informações como número de matrícula, senha, nome e, ao fim do processo, são capturadas algumas imagens para o treinamento inicial dos algoritmos de reconhecimento facial. Todo o processo é mostrado nas Figuras 5.22b, 5.22c, 5.22d, 5.22e, 5.22f.

Posteriormente ao registro ou à autenticação de um usuário do tipo aluno, é então disponibilizado o fluxo para captura de um código QR de disciplina e, logo após, a captura da sua imagem respeitando também o formato pré-definido, com dicas para que o usuário consiga alcançar a captura da melhor imagem. Após a captura com sucesso, a imagem é enviada para o *back-end* e é retornada para o aluno uma mensagem de sucesso, o que encerra o seu fluxo. Todo o processo é mostrado nas Figuras 5.23a, 5.23b, 5.23c, 5.23d, 5.23e.

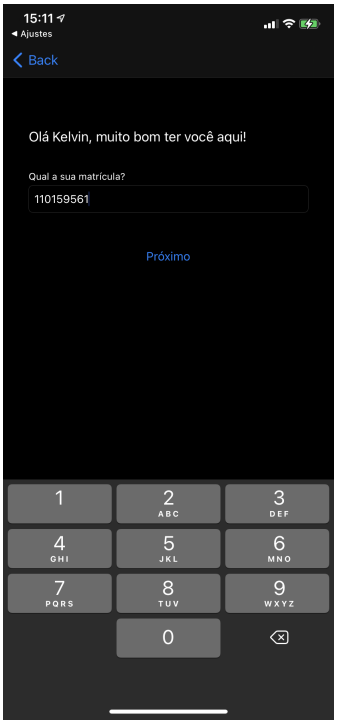
Seguindo, ao inserir informações no banco de dados, é possível validarmos o fluxo



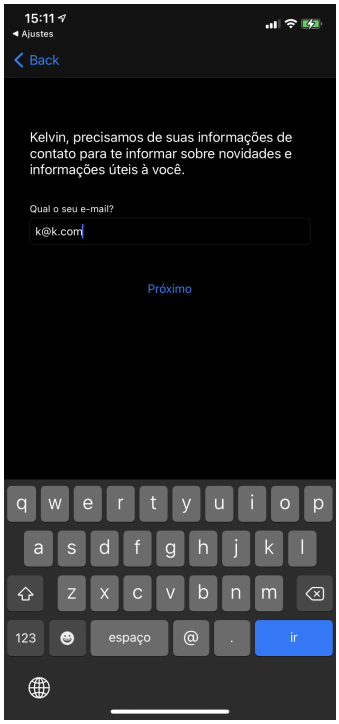
(a) Tela para autenticação do Aluno



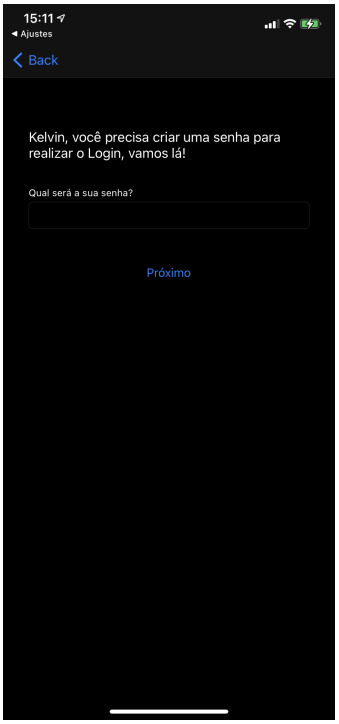
(b) Tela para preenchimento de nome e sobrenome do aluno



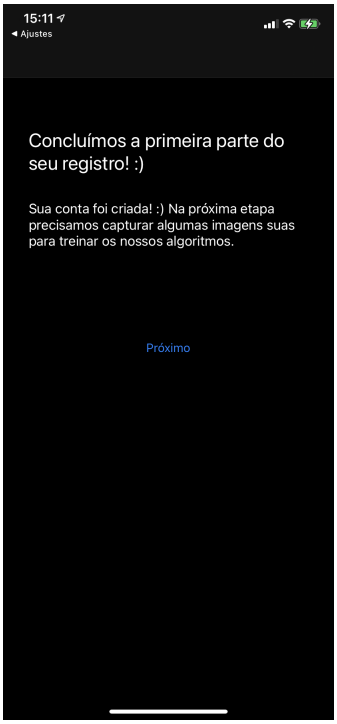
(c) Tela para inserção de número de matrícula



(d) Tela para inserção de e-mail



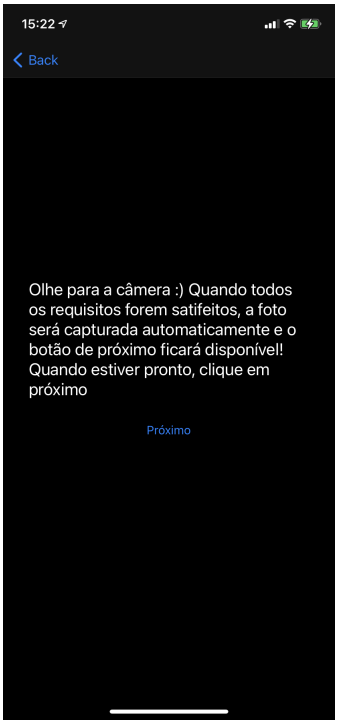
(e) Tela para inserção de senha



(f) Tela de aviso de captura de imagem no registro do aluno

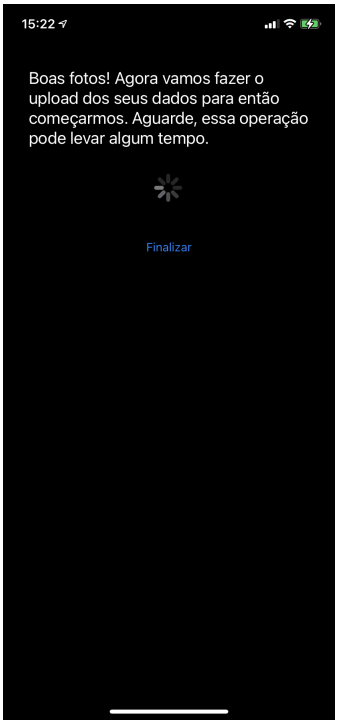
Figura 5.22: Fluxo de autenticação e registro de alunos.

(a) Tela de captura de código QR para chamada

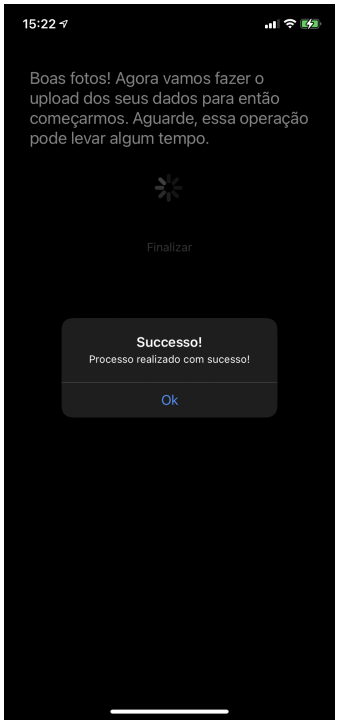


(b) Tela com instruções para captura de imagem do usuário

(c) Tela de captura de imagem do usuário



(d) Tela de informação sobre a captura com sucesso e upload para o *back-end*



(e) Tela de processamento móvel finalizado com sucesso

Figura 5.23: Exemplificação do processo de captura de presença de alunos em aplicativo móvel.

criado com um pequeno conjunto de testes e então exibir os seus resultados. O processo é detalhado em 5.5.

5.5 Resultados Experimentais

Como ressaltado em parágrafos anteriores, não foi possível a realização de testes em larga escala para que tanto o algoritmo quanto a arquitetura fossem ajustados de modo a apresentar os melhores resultados possíveis, ficando assim como um trabalho futuro à ser melhor detalhado na subseção vindoura 6.1.

Com isso, temos que os testes foram executados em 5 faces, onde os indivíduos 1 e 2 são irmãos e primos do indivíduo 3 e os indivíduos 3 e 4 são irmãos e sem parentescos com os anteriores, Os indivíduos 2, 3, 4 são do sexo feminino e os indivíduos 1 e 5 de sexo masculino. Os testes foram realizados de modo que todos capturam imagens prévias para treinamento, conforme detalhado na subseção 5.4, e realizaram a computação de chamadas em seus próprios cadastros e em cadastros dos outros usuários de modo que possa ser checada a variação das distâncias reconhecimento. Os resultados obtidos são mostrados nas Tabelas 5.2, 5.3, 5.4.

Tabela 5.2: Resultados de reconhecimento facial utilizando o algoritmo FisherFaces

	Sujeito 1	**Sujeito 2**	**Sujeito 3**	**Sujeito 4**	**Sujeito 5**
Sujeito 1	148.56	2147.72	2127.67	1454.77	1740.69
Sujeito 2	106.70	2049.36	3197.08	2651.36	-1
Sujeito 3	1471.23	2830.53	737.21	2183.43	3321.71
Sujeito 4	353.89	2129.91	1777.64	2174.10	1848.82
Sujeito 5	1670.82	-1	-1	1551.52	1196.08

Tabela 5.3: Resultados de reconhecimento facial utilizando o algoritmo LBPH

	Sujeito 1	**Sujeito 2**	**Sujeito 3**	**Sujeito 4**	**Sujeito 5**
Sujeito 1	80.07	142.35	97.11	144.08	97.16
Sujeito 2	96.25	86.09	93.67	95.84	101.73
Sujeito 3	94.18	95.33	79.90	92.62	95.65
Sujeito 4	93.30	93.67	88.94	105.10	102.05
Sujeito 5	100.32	102.32	99.30	94.72	69.19

Como notado na tabela, os resultados apresentam valores fora de um intervalo de 0 a 100% e sim a distância entre as imagens de treino e a imagem para a qual a predição é desejada, fazendo assim com que seja necessário a definição de um limiar para assumirmos que uma pessoa é de fato quem ela diz ser.

Tabela 5.4: Resultados de reconhecimento facial utilizando o algoritmo EigenFaces

	Sujeito 1	**Sujeito 2**	**Sujeito 3**	**Sujeito 4**	**Sujeito 5**
Sujeito 1	671.10	1133.47	616.70	1048.56	2478.49
Sujeito 2	1146.91	931.97	1144.48	862.55	2121.46
Sujeito 3	2061.09	2275.76	4434.15	1781.72	2904.54
Sujeito 4	1062.67	1176.22	473.84	1153.52	2478.09
Sujeito 5	1289.84	1068.20	917.96	1210.46	460.62

Para o algoritmo FisherFaces [12], quando o resultado é -1, temos que a pessoa não é reconhecida pelo algoritmo como quem diz ser.

Como resultado geral, temos que as distâncias tendem a ser menores quando a pessoa é quem diz ser. Entretanto, vale notar que expressões faciais e qualidade da captura influenciam diretamente no resultado obtido, como detalhados nas seção 2.1.2.

Mesmo com essa pequena observação, seria muita pretensão realizar qualquer afirmação sobre um valor aceitável para um delta de reconhecimento sob o qual o valor para a mesma pessoa tende a variar e com isso é preciso definir quão permissivo o mesmo deve ser. Portanto, é deixado como trabalho futuro 6.1 a definição deste.

Para que melhoras afirmações possam ser feitas torna-se necessária a realização de testes em larga escala de modo a checar-se possíveis condições de contorno e mapear-se todos os casos de teste para o refinamento de algoritmos tornando-o robusto.

Capítulo 6

Conclusões e Trabalhos futuros

O trabalho consistiu na proposição e construção de um produto mínimo viável para a computação automática de frequências utilizando-se algoritmos de reconhecimento facial, com os quais foram definidas regras para a entrada de imagens na base, de modo a se garantir um padrão de qualidade em todas as imagens utilizadas como entrada.

Identifica-se, também, a necessidade de melhores testes em pontos críticos, como a proposição de separação em arquivos individuais de treinamento e seus efeitos sobre os resultados finais. Como dito, esses testes não foram possíveis por conta da pandemia de SARS-CoV-2 que atinge o mundo como um todo, bem como pelas medidas sanitárias exigidas pelo fato supracitado.

Por fim, temos que melhorias podem ser aplicadas a todos os módulos e muitos estudos podem derivar desse trabalho inicial, os quais são detalhados na seção 6.1.

6.1 Trabalhos Futuros

Trabalhos futuros podem ser concebidos para atestar que o sistema aqui proposto apresenta viabilidade e aplicabilidade nos mais diferentes setores. Adicionalmente, temos que os algoritmos de reconhecimento facial utilizados são clássicos e em sua maior parte concebidos nos anos 90, após os quais apareceram novas técnicas que podem agregar ainda mais viabilidade e qualidade no reconhecimento, a exemplo temos o MTCNN [2].

Trazendo especificamente para cada plataforma apresentada, no módulo de *back-end* apresentado na subseção 5.2 poderia ser utilizada uma outra linguagem que apresente melhor integração com o OpenCV [16] ou outras tecnologias que forneçam uma integração mais simples, o que facilitaria sua manutenibilidade e crescimento.

Já no módulo de *front-end* mostrado na subseção 5.3, a aplicação pode ser melhor organizada mediante estudos de usabilidade e interface do usuário, de modo a trazer mais clareza e coesão nas ações para se tornar ainda amigável para o usuário final.

Por fim, para o módulo móvel apresentado na subseção 5.4, assim como no módulo de *front-end* apresentado em 5.3, um trabalho de levantamento de requisitos, melhoria de usabilidade e de interface pode ser proposto, de modo a atender as necessidades dos alunos que utilizarão o sistema. Além disso, é possível que, assim como no *back-end* mostrado em 5.2, sejam utilizados algoritmos mais eficientes de processamento de imagens, de modo a reduzir o consumo energético e de CPU por parte destes dispositivos. Finalizamos as sugestões ao identificarmos a necessidade de implementação contra ataques ou fradues que possam vir a ser realizadas.

Dessarte, deve-se levar ainda em consideração os pontos de melhoria citados em [25] que afetam, diretamente, além de que constituem-se como um ponto de partida para novos trabalhos.

Referências

[1] Bledsoe, W.W.: *The model method in facial recognition.* Technical report pri 15„ Panoramic Research, Inc, Palo Alto, California. ix, 4, 5

[2] Zhang, Kaipeng, Zhanpeng Zhang, Zhifeng Li e Yu Qiao: *Joint face detection and alignment using multitask cascaded convolutional networks.* IEEE Signal Processing Letters, 23, abril 2016. ix, 6, 10, 11, 13, 48

[3] *Mobile vision, google developers.* `https://developers.google.com/vision`. ix, 11, 12

[4] *Find and explore academic papers.* `https://www.connectedpapers.com/`. ix, 14, 15

[5] Sunaryono, Dwi, Joko Siswantoro e Radityo Anggoro: *An android based course attendance system using face recognition.* Journal of King Saud University - Computer and Information Sciences, 33:304–312, março 2021. ix, 14, 15

[6] *PostgreSQL*, Apr 2021. `https://www.postgresql.org`, [Online; accessed 23. Apr. 2021]. ix, 26, 27

[7] *UnB - Boas Vindas - Início*, May 2021. `https://boasvindas.unb.br`, [Online; accessed 3. May 2021]. 1

[8] *L13709compilado*, Sep 2020. `http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709compilado.htm`, [Online; accessed 3. May 2021]. 1

[9] Kanade, Takeo: *Computer recognition of human faces.* Interdisciplinary Systems Research, 47, January 1977. 5

[10] Turk, M. A. e A. P. Pentland: *Face recognition using eigenfaces.* Em *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, páginas 586–591, 1991. 5, 6, 28

[11] Ahonen, Timo, Abdenour Hadid e Matti Pietikäinen: *Face recognition with local binary patterns.* Volume 3021, páginas 469–481, maio 2004, ISBN 978-3-540-21984-2. 5, 9, 10, 28

[12] Belhumeur, P. N., J. P. Hespanha e D. J. Kriegman: *Eigenfaces vs. fisherfaces: recognition using class specific linear projection.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):711–720, 1997. 5, 7, 28, 47

[13] Lindeberg, Tony: *Scale Invariant Feature Transform*, volume 7. maio 2012. 6

[14] Bay, Herbert, Tinne Tuytelaars e Luc Van Gool: *Surf: Speeded up robust features*. Volume 3951, páginas 404–417, julho 2006, ISBN 978-3-540-33832-1. 6

[15] `https://developer.apple.com/documentation/vision`. 12

[16] Bradski, G.: *The OpenCV Library*. Dr. Dobb's Journal of Software Tools, 2000. 12, 26, 31, 32, 48

[17] *Core ml*. `https://developer.apple.com/documentation/coreml`. 13

[18] *Ml kit | google developers*. `https://developers.google.com/ml-kit`. 13

[19] Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu e Xiaoqiang Zheng: *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. `https://www.tensorflow.org/`, Software available from tensorflow.org. 13

[20] Lukas, S., A. R. Mitra, R. I. Desanti e D. Krisnadi: *Student attendance system in classroom using face recognition technique*. Em *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, páginas 1032–1035, 2016. 14, 15

[21] Wagh, P., R. Thakare, J. Chaudhari e S. Patil: *Attendance system based on face recognition using eigen face and pca algorithms*. Em *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, páginas 303–308, 2015. 14, 16

[22] Jayant, N. K. e S. Borra: *Attendance management system using hybrid face recognition techniques*. Em *2016 Conference on Advances in Signal Processing (CASP)*, páginas 412–417, 2016. 14, 16

[23] Sulyman, Shakirat: *Client-server model*. IOSR Journal of Computer Engineering, 16:57–71, janeiro 2014. 20

[24] 2015. `https://www.icao.int/publications/documents/9303_p9_cons_en.pdf`. 23

[25] PEDREIRA, Matheus Rosendo e il f.: *Fotografias padronizadas utilizando dispositivo móvel*, volume x. Bacharelado em Engenharia de Computação)—Universidade de Brasília, Brasília. 23, 49

[26] Fielding, Roy Thomas e Irvine: *Chapter 5: Representational state transfer (rest)». architectural styles and the design of network-based software architectures (ph.d.). university of california.* 23

[27] Zeilenga, Kurt e Alexey Melnikov: *Simple Authentication and Security Layer (SASL).* RFC 4422, junho 2006. `https://rfc-editor.org/rfc/rfc4422.txt`. 24

[28] Jones, Michael, John Bradley e Nat Sakimura: *JSON Web Token (JWT).* RFC 7519, maio 2015. `https://rfc-editor.org/rfc/rfc7519.txt`. 24, 29, 31

[29] *Comitê gestor do plano de contingência da covid-19.* `http://repositoriocovid19.unb.br/comite-gestor-do-plano-de-contingencia-da-covid-19/`. 25

[30] nodejs: *node*, Apr 2021. `https://github.com/nodejs/node`, [Online; accessed 23. Apr. 2021]. 26, 32

[31] koajs: *koa*, Apr 2021. `https://github.com/koajs/koa`, [Online; accessed 23. Apr. 2021]. 26

[32] justadudewhohacks: *opencv4nodejs*, Apr 2021. `https://github.com/justadudewhohacks/opencv4nodejs`, [Online; accessed 23. Apr. 2021]. 26, 32

[33] Inc., Apple: *Swift.org*, Apr 2021. `https://swift.org`, [Online; accessed 23. Apr. 2021]. 26

[34] angular: *angular*, Apr 2021. `https://github.com/angular/angular`, [Online; accessed 23. Apr. 2021]. 26

[35] *Visual Studio Code*, Apr 2021. `https://code.visualstudio.com`, [Online; accessed 23. Apr. 2021]. 26

[36] Inc., Apple: *Xcode - Apple Developer*, Feb 2021. `https://developer.apple.com/xcode`, [Online; accessed 23. Apr. 2021]. 26

[37] *TypeORM - Amazing ORM for TypeScript and JavaScript (ES7, ES6, ES5). Supports MySQL, PostgreSQL, MariaDB, SQLite, MS SQL Server, Oracle, WebSQL databases. Works in NodeJS, Browser, Ionic, Cordova and Electron platforms.*, Apr 2021. `https://typeorm.io/`, [Online; accessed 23. Apr. 2021]. 28

[38] Krawczyk, Hugo, Ran Canetti e Mihir Bellare: *HMAC: Keyed-Hashing for Message Authentication*, Feb 1997. `https://tools.ietf.org/html/rfc2104`, [Online; accessed 25. Apr. 2021]. 29

[39] Gueron, Shay, Simon Johnson e Jesse Walker: *Sha-512/256.* Em *2011 Eighth International Conference on Information Technology: New Generations*, páginas 354–358, 2011. 29

[40] Bray, Tim: *The JavaScript Object Notation (JSON) Data Interchange Format.* RFC 8259, dezembro 2017. `https://rfc-editor.org/rfc/rfc8259.txt`. 30

[41] sideway: *joi*, Apr 2021. `https://github.com/sideway/joi`, [Online; accessed 25. Apr. 2021]. 30

[42] *Postman The Collaboration Platform for API Development*, Apr 2021. `https://www.postman.com`, [Online; accessed 25. Apr. 2021]. 31