MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Cap DANIEL DOS REIS ABREU
1º Ten GABRIEL MOYSÉS DELFINO
1º Ten JOSÉ LUIZ NEVES VOLTAN

TCHAU PAPELETA DE FALTAS!

Rio de Janeiro
2018

INSTITUTO MILITAR DE ENGENHARIA

Cap DANIEL DOS REIS ABREU
1º Ten GABRIEL MOYSÉS DELFINO
1º Ten JOSÉ LUIZ NEVES VOLTAN

TCHAU PAPELETA DE FALTAS!

Projeto de Fim de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador: Ten Cel Anderson Fernandes Pereira dos Santos - D.Sc.

Rio de Janeiro
2018

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 - Praia Vermelha
Rio de Janeiro - RJ CEP 22290-270

005 Abreu, Daniel dos Reis
A162t Tchau Papeleta de Faltas! / Daniel dos Reis Abreu, Gabriel Moysés Delfino, José Luiz Neves Voltan, orientado por Anderson Fernandes Pereira dos Santos - Rio de Janeiro: Instituto Militar de Engenharia, 2018.

74p.: il.

Projeto de Fim de Curso (graduação) - Instituto Militar de Engenharia, Rio de Janeiro, 2018.

1. Curso de Graduação em Engenharia de Computação - projeto de fim de curso. 1. Reconhecimento facial. I. dos Santos, Anderson Fernandes Pereira . II. Título. III. Instituto Militar de Engenharia.

INSTITUTO MILITAR DE ENGENHARIA

Cap DANIEL DOS REIS ABREU
1º Ten GABRIEL MOYSÉS DELFINO
1º Ten JOSÉ LUIZ NEVES VOLTAN

TCHAU PAPELETA DE FALTAS!

Projeto de Fim de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador: Ten Cel Anderson Fernandes Pereira dos Santos - D.Sc.

Aprovado em 1º de outubro de 2018 pela seguinte Banca Examinadora:

Ten Cel Anderson Fernandes Pereira dos Santos - D.Sc. do IME - Presidente

Ten Cel Sérgio dos Santos Cardoso Silva - M.Sc. do IME

Prof. Ronaldo Ribeiro Goldschimdt - D.Sc. do IME

Rio de Janeiro
2018

Ao Instituto Militar de Engenharia, alicerce de nossa formação e aperfeiçoamento.

AGRADECIMENTOS

Agradecemos a todas as pessoas que nos incentivaram, apoiaram e possibilitaram esta oportunidade de ampliar nossos horizontes.

Nossos familiares e mestres que sempre buscaram nos estimular durante o projeto.

Ao meu pai, José Luiz Voltan (*in memorian*), por todo seu amor, exemplo de caráter e dignidade que me permitiram chegar até aqui. Além das palavras de estímulo e orgulho ao longo do projeto.

Em especial ao nosso Professor Orientador Ten. Cel. Anderson Fernandes Pereira dos Santos, por sua disponibilidade e atenção.

"A história de todas as grandes civilizações galácticas tende a atravessar três fases distintas e identificáveis – as da sobrevivência, da interrogação e da sofisticação, também conhecidas como as fases do como, do porquê e do onde. "

DOUGLAS ADAMS

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

AJAX	Asynchronous JavaScript And XML
BD	Banco de Dados
CA	Corpo de Alunos
ER	Entidade-Relacionamento
GNU	Gnu's Not Unix
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
IDC	International Data Corporation
IME	Instituto Militar de Engenharia
IP	Internet Protocol
JPEG	Joint Photographic Experts Group
JSP	Java Server Page
LBPH	Local Binary Patterns Histograms
LBP	Local Binary Pattern
LDA	Linear Discriminant Analysis
NDK	Native Development Kit
NEFSE	Normas de Execução de Funções e Serviços de Escala
NICA	Normas Internas do Corpo de Alunos
PCA	Principal Component Analysis
SDK	Software Development Kit
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

LISTA DE ABREVIATURAS

ABREVIATURAS

OpenCV - Open Source Computer Vision

RESUMO

O objetivo deste projeto consiste em desenvolver um sistema informatizado, empregando reconhecimento facial e tecnologia de desenvolvimento móvel, visando propor uma alternativa para o atual sistema de apuração de faltas dos alunos do IME. Essa alternativa consiste em utilizar dispositivos eletrônicos para registrar a frequência dos alunos, em vez de fazê-lo por meio do preenchimento de papeletas, com a finalidade de facilitar o processo de verificação de presença, evitando o retrabalho recorrente do modelo vigente. Nessa proposta, tirar-se-á uma fotografia da turma, enviando-a posteriormente para um servidor remoto responsável por realizar o reconhecimento facial. O servidor possui um banco de imagens da turma, que são utilizadas para o treinamento do algoritmo de reconhecimento facial. A inteligência gerada pelo treinamento associa os parâmetros de cada foto do banco de imagens a um identificador, nesse caso o nome do aluno. Dessa forma, o reconhecimento consiste em comparar os parâmetros de uma foto tirada com os parâmetros das imagens do banco de imagens. A presença do aluno é registrada quando ele é reconhecido na fotografia. Essa informação é, então, salva em um banco de dados, possibilitando o armazenamento persistente da informação, permitindo, portanto, a correção de eventuais erros. A atualização dos dados pode ser feita no momento em que a foto foi tirada ou em momentos posteriores, podendo-se utilizar a fotografia para eventuais verificações necessárias.

ABSTRACT

The objective of this project is to develop a computerized system, employing facial recognition and mobile development technology, aiming at proposing an alternative to the current IME's student attendance system. This alternative consists of using electronic devices to record the students' attendance, rather than filling a paper form, in order to facilitate the presence verification process, avoiding the recurring rework of the current model. In this proposal, a photograph of the class is taken, which is sent to a remote server responsible for performing facial recognition. The server has a group of images of the class, which is used to train the facial recognition algorithm. The intelligence obtained by this training process associates the parameters of each picture in the image bank to an identifier, in this case the student's name. In this way, recognition consists of comparing the parameters of a photo taken with the parameters of the image bank images. The student's presence is recorded when he is recognized in the photograph. This information is then saved in a database, allowing the persistent storage of the information, thus allowing the correction of any errors. This can be done at the time the photo was taken, as well as at later times, and the photograph can be used for any necessary checks.

1 INTRODUÇÃO

A tarefa de verificar a presença dos alunos é uma constante na sala de aula. Todo início de aula é marcado por esse processo. Ele é extremamente importante para a vida acadêmica dos alunos, uma vez que a falta acarreta na perda de pontos, o que pode levar, em casos extremos, a exclusão do aluno.

1.1 CONTEXTUALIZAÇÃO DO TEMA

Por ser uma instituição de ensino com aulas presenciais e cumprir as exigências do Ministério da Educação, o Instituto Militar de Engenharia, exige, em seu regulamento que trata sobre a frequência escolar (IME, 2009b), que seus alunos mantenham um alto índice de presença nas aulas ao longo do período escolar. De modo a verificar esse objetivo, existe um sistema de apuração de faltas que consiste na entrega da papeleta de faltas, uma papeleta preenchida a caneta contendo a identificação de todo e qualquer estudante ausente em cada um dos tempos de aula do dia. O preenchimento das papeletas é de responsabilidade de um aluno escalado, o qual, na seção de Engenharia da Computação, faz tal tarefa em três vias: duas vias entregues à Seção de Ensino e uma via entregue ao Corpo de Alunos. Além disso, os professores têm de assinar a papeleta, em suas 2 vias. As Seções de Ensino fazem o lançamento das faltas no sistema acadêmico do IME e recebem as justificativas de faltas dos alunos da reserva. O CA recebe as justificativas das faltas dos alunos da ativa e abre processo de apuração de transgressão disciplinar quando ocorre uma falta não justificada.

1.2 MOTIVAÇÃO

Diante do exposto, podemos identificar uma série de problemas nesse processo: a manutenção de arquivos do mesmo documento, sendo um em cada seção; lentidão de consulta ao arquivo, devido ao grande acúmulo de papel; possibilidade de perda de papeletas, comprometendo a apuração das faltas; maior probabilidade de conter informações imprecisas ou incorretas, como erros de lançamento; e aumento do fluxo de documentos em cada seção. Outrossim, o processo depende da interação de diversos atores, o que aumenta a probabilidade de uma falha.

Além disso, a apuração da falta cabe ao chefe de turma, segundo as normas que

especificam as funções e responsabilidades (IME, 2009a). Dessa maneira, cabe ao aluno a verificação de presença dos seus colegas e a sua própria, fugindo assim da idealidade, já que acumula os papeis de fiscalizado e fiscalizador.

É importante ressaltar que (IME, 2009b) dispõem no seguinte sentido: "o acúmulo superior a 120 pontos perdidos, em faltas, resulta na reprovação, desligamento e exclusão do aluno, impedindo-o de concluir seu curso no IME". Assim sendo, fica evidente a necessidade de um sistema ágil e preciso, que simplifique a apuração das faltas pelos responsáveis.

1.3 OBJETIVO

Considerando a motivação apresentada na seção anterior, o objetivo desse trabalho é desenvolver um sistema informatizado capaz de automatizar o processo de apuração e registro de faltas. Para tanto, será feito o uso de reconhecimento facial e de tecnologia de desenvolvimento móvel como principais ferramentas, estando também presentes no sistema uma conexão cliente-servidor e um banco de dados.

A estrutura geral do sistema, como se pode ver na Figura 1.1, consiste na interação entre dois dispositivos, sendo um cliente e um servidor. O primeiro será um dispositivo móvel Android, cuja tarefa é enviar ao servidor uma foto dos alunos presentes no início de cada tempo de aula, bem como as informações sobre a turma e a aula. O segundo dispositivo será um servidor que terá como entrada as informações enviadas pelo cliente e um boleto contendo as faltas como saída. As faltas serão apuradas por meio de algoritmos de reconhecimento facial, os quais irão reconhecer as faces contidas na foto recebida, marcando como presentes os alunos reconhecidos e salvando essa informação no banco de dados. Dessa forma, o registro de frequências ficará armazenado nesse banco de dados, o qual pode ser consultado sempre que for necessário, eliminando a necessidade do uso de papel. Como maneira corretiva para eventuais erros, o professor e o aluno podem atuar. A arquitetura será explicada com mais detalhes em um capítulo próprio.

1.4 JUSTIFICATIVA

Fica evidente, então, que a retirada de falta é um processo muito importante na vida acadêmica do aluno, podendo inclusive acarretar, nos casos mais extremos, em sua reprovação. Dada a importância desse processo, o produto deve ser tão fiel quanto possível à realidade, apresentando de maneira correta e precisa os discentes presentes e ausentes naquele tempo de aula.

Com a proposta de sistema ora feita, os erros causados pelo fator humano e o retrabalho mencionados na motivação são mitigados. Dessa forma, consegue-se um processo mais eficiente com produtos mais fidedignos.

1.5 METODOLOGIA

A metodologia adotada para alcançar o objetivo proposto para este trabalho se divide em: fundamentação teórica, projeto e implementação de prova de conceito. Visando a correta implementação do sistema proposto, a fundamentação teórica faz-se necessária para a correta compreensão de conceitos como reconhecimento e detecção facial, os quais se constituem como o cerne da atividade proposta por esse trabalho. O projeto envolve a modelagem e o planejamento de desenvolvimento do sistema. Na fase de implementação de prova de conceito ocorrerá a realização de testes a fim de verificar a acurácia na verificação de faltas e o comportamento do sistema como um todo.

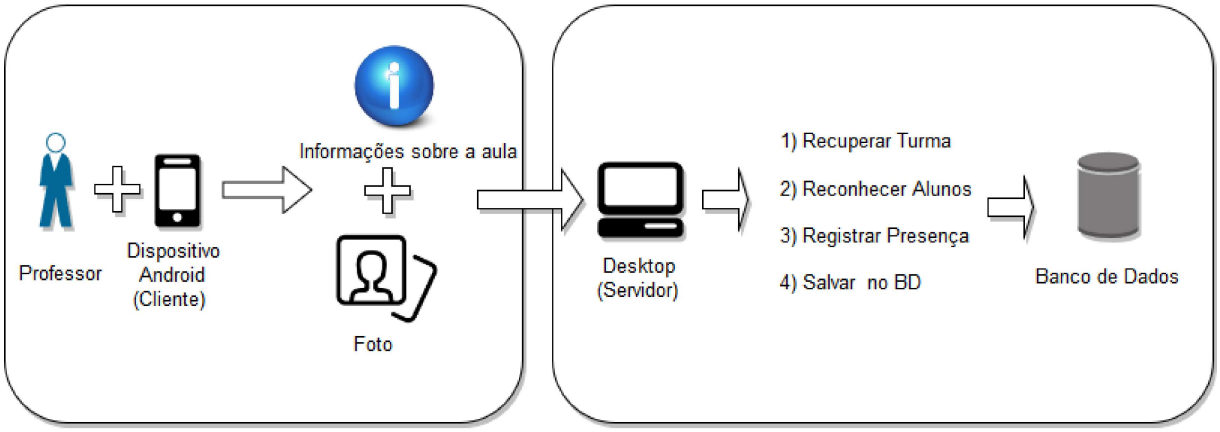

FIG. 1.1: Estrutura do Sistema

1.6 ESTRUTURA

Este trabalho será organizado de acordo com o seguinte formato: O capítulo 1 traz uma breve introdução e contextualização do tema. o capítulo 2 contém os conceitos básicos relacionados à detecção e reconhecimento facial; o capítulo 3 trata da modelagem do sistema; o capítulo 4 aborda a construção do módulo servidor, esse módulo contempla o reconhecimento facial; o capítulo 5 trata do módulo *mobile*, mostrando o aplicativo desenvolvido e o programa que executado no servidor é responsável por receber a fotografia e as informações enviadas pelo professor, como seu desenvolvimento estava intimamente ligado com o aplicativo, optou-se por descrevê-lo nesse capítulo; o capítulo 6 descreve o

módulo BD, e é responsável pela criação do esquema no banco de dados, aborda-se ainda a conexão do módulo servidor com o banco de dados, além de se abordar a interface web (servidor web) para visualização e alteração das papeletas; o capítulo 7 faz a Análise crítica do sistema, e por fim, o capítulo 8 aborda a conclusão do projeto.

2 RECONHECIMENTO FACIAL

O reconhecimento facial desenvolve a tarefa mais importante do projeto, na medida em que ele é o responsável direto pela apuracão de faltas. Essa tarefa possui, basicamente, dois estágios: a detecção facial e o reconhecimento facial propriamente dito.

2.1 DETECÇÃO FACIAL

Nessa seção, são descritas as especificidades básicas dos algoritmos de detecção facial, representados nesse projeto pelos algoritmos *haarcascade frontal face* e *haarcascade eye*, os quais são componentes da biblioteca de código aberto OpenCV, cujas especificações estão disponíveis na documentação (OPENCV, 2018a), e são utilizados como classificadores para detectar faces e olhos, respectivamente. O principal objetivo desses algoritmos é identificar algum objeto em uma imagem, determinando seu tamanho e localização, que será extraído da imagem e processado de forma a facilitar o reconhecimento.

Esses algoritmos, baseados no trabalho de (PAUL VIOLA, 2001), são constituídos por quatro componentes: *Haar-like features*, imagem integral (*Integral Image*), o algoritmo de aprendizado chamado de *Adaboost* e a combinação de classificadores em cascata. As *Haar-like features* (características) são as estruturas utilizadas para identificar objetos em uma imagem. Elas podem ter diferentes formatos e tamanhos e são representadas por figuras retangulares que cobrem uma determinada área da imagem que se deseja analisar. Na figura 2.1 pode-se observar alguns exemplos dessas características:

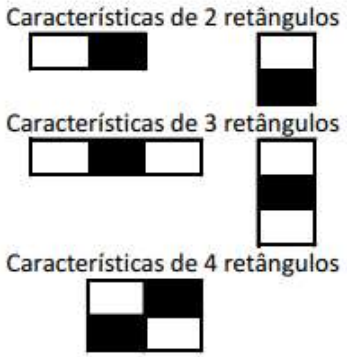

FIG. 2.1: Haar-like Features

Os retângulos pretos correspondem a um grupo de *pixels* com tonalidade escura, ao passo que os retângulos brancos correspondem aos *pixels* com tonalidade clara. Cada

característica é definida por um valor que é obtido pela diferença entre o somatório das intensidades dos *pixels* contidos na área preta e o somatório dos *pixels* da área branca. Quando este valor é superior a um limiar padronizado, diz-se que a característica está presente na imagem. A figura 2.2 é um exemplo de aplicação desse conceito: há a utilização de uma característica de três retângulos (sendo dois pretos e um branco) para verificar se a imagem contem olhos, isso porque, normalmente, a região dos olhos é mais escura que a região superior do nariz. Em outras palavras, para que a existência de olhos seja detectada é necessário que uma característica com aquele formato específico possua um valor superior ao limiar determinado em treinamento.

FIG. 2.2: Exemplo de Haar-like Features

A Imagem Integral é uma técnica desenvolvida por (PAUL VIOLA, 2001) para agilizar o cálculo das características. Para tanto, cria-se uma imagem intermediária tal que o valor de qualquer *pixel* na posição (x,y) é dado pelo somatório dele com todos os *pixels* localizados acima e a esquerda dele. Dessa forma, não é necessário acessar o valor de cada *pixel* para calcular o valor de uma área. A figura 2.3 ilustra o funcionamento dessa técnica: nesse exemplo uma imagem está sendo representada por uma matriz 4x4, a qual contém uma região destacada cujo valor se deseja obter, note que o valor da soma dos *pixels* dessa área já está definido na imagem integral, na última posição que define a área escolhida (sempre localizado no canto inferior direito).

Foi exposto anteriormente que a associação de características a objetos está condicionada a um treinamento. Essa tarefa é realizada pelo algoritmo de aprendizado de máquina supervisionado *Adaboost*, cujo objetivo é construir um conjunto de classificadores simples, sendo cada um composto por apenas uma característica importante para a detecção do objeto, (PAUL VIOLA, 2001). Essa tarefa é importante porque em uma imagem existe uma grande quantidade de características (*Haar-like features*) e, por isso, encontrar os tipos de características que melhor representam o objeto alvo economiza o

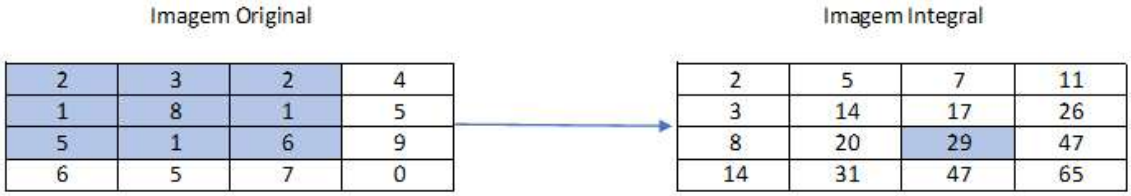

FIG. 2.3: Exemplo de Imagem Integral

esforço de analisar todas as demais. A *Haar-like feature* da figura 2.2 é um exemplo de característica que compõe um classificador simples, pois ela sozinha não define uma face, mas a sua ausência já descarta a possibilidade de ser um rosto. Assim, o treinamento desse algoritmo consiste, segundo (OPENCV, 2018a), na exposição de algumas centenas de amostras de um objeto específico, como um rosto ou um carro, chamados de exemplos positivos, os quais são formatados para terem as mesmas dimensões para só então compará-los com imagens arbitrárias de outros objetos de mesmas dimensões, chamados de exemplos negativos. Dos exemplos positivos, portanto, são definidos os classificadores (características e limiares).

Terminado o treinamento, é possível aplicar o classificador numa imagem de entrada. Quando o valor do conjunto de características que compõem o classificador é superior ao seu limiar, este atribui o valor 1 a imagem, o que indica que o objeto alvo está presente na imagem, caso contrário atribui 0, (OPENCV, 2018a). Na detecção facial, esse processo ocorre diversas vezes por meio da combinação dos classificadores em cascata, que consiste no encadeamento de vários classificadores simples aplicados em sequência em uma imagem, que crescem em complexidade a cada iteração, até que essa região seja rejeitada, por algum classificador, ou aprovada em todos os classificadores, (OPENCV, 2018a). Quando a imagem é rejeitada por algum classificador simples, o processo encerra sem que os outros classificadores sejam aplicados. Esse procedimento aumenta drasticamente o desempenho e a precisão da detecção do objeto, pois as imagens indesejadas são eliminadas nas iterações iniciais, envolvendo menor processamento, sobrando para os classificadores mais complexos apenas aquelas com alta probabilidade de ser o objeto procurado.

Dessa forma, o papel a ser desenvolvido pelos algoritmos citados no início dessa seção é apoiar o reconhecimento facial, a partir da detecção de faces em imagens captadas por uma câmera fotográfica, as quais serão processadas e armazenadas para posterior análise dos algoritmos de reconhecimento facial que serão o objeto da seção seguinte.

2.2 ALGORITMOS DE RECONHECIMENTO FACIAL

Concluída a etapa de detecção facial, tem-se como produto a imagem da face já redimensionada, transformada para a escala de cinza e recortada da imagem original, esse processo é ilustrado na figura 2.4, em (b) ocorre a transformação para a escala cinza (o retângulo vermelho é apenas uma abstração utilizada para representar a borda do que se irá recortar), e em (c) o recorte propriamente dito. A partir daí começa a etapa de reconhecimento que possui, basicamente, duas modalidades: autenticação e identificação. A primeira consiste em uma comparação 1 X 1 entre a imagem de entrada com a imagem da pessoa a ser autenticada. A segunda consiste em uma comparação 1 X N na qual a entrada é comparada com todas as imagens de um determinado conjunto para determinar se alguma pessoa do conjunto corresponde à entrada (SALTON, 2018). No processo de apuração de faltas, convém utilizar apenas a modalidade de identificação.

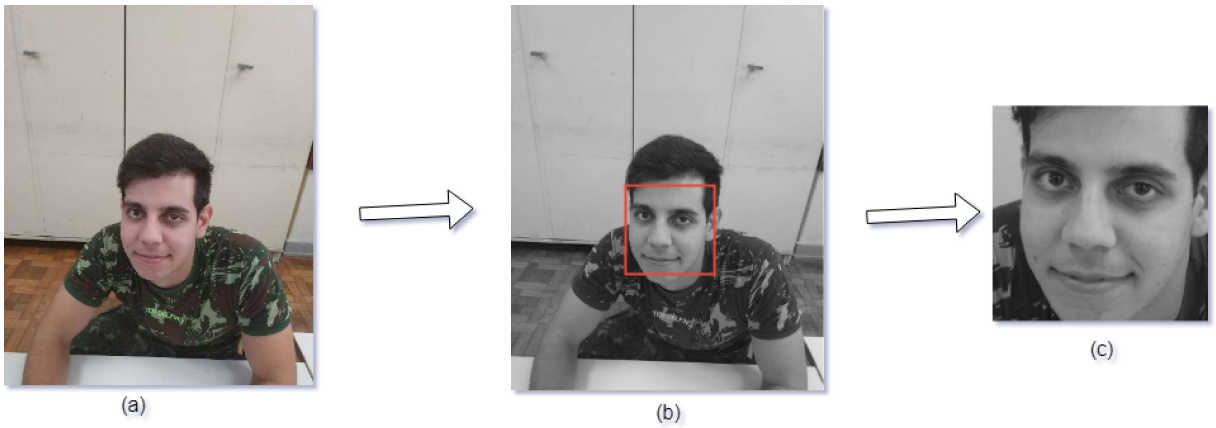

FIG. 2.4: Processo de recorte

Para o cumprimento da tarefa de reconhecimento facial, a biblioteca OpenCV oferece três algoritmos: *Eigenfaces*, *Fischerfaces* e o LBPH. Por fazerem parte de uma biblioteca de código aberto, esses algoritmos são gratuitos para uso e esse fato contribuiu para a sua popularização, tendo como resultado o seu uso frequente em trabalhos científicos. Outro fator motivador para o emprego desses algoritmos é o seu bom desempenho, pois quando comparado ao MATLAB, que também pode ser utilizado para essa função, o OpenCV se destaca por usar menos memória e ser mais rápido que o seu concorrente, pois pode processar 30 quadros por segundo, enquanto o MATLAB executa de 3 a 4 quadros por segundo (SUDHANARANG; AASHNAARORA, 2018).

Quanto aos algoritmos, observou-se que em trabalhos científicos cujo objetivo é realizar o controle de faltas por meio de reconhecimento facial, o LBPH é apontado como a melhor opção dentre os três. Isso porque os algoritmos que utilizam as técnicas de

PCA e LDA (Eigenfaces e Fisherfaces, respectivamente) para o reconhecimento facial apresentam dificuldades para superar problemas como a variação de: dimensionamento, iluminação, rotação e oclusão (MRUNMAYEE SHIRODKAR; NEMADE, 2015). O trabalho de (SUDHANARANG; AASHNAARORA, 2018) também indica a superioridade do LBPH, o qual é classificado como o algoritmo mais eficiente e preciso, do OpenCV, para a tarefa de reconhecimento, após fazer um teste de desempenho comparando os três algoritmos. Esse teste também apontou a invariância à iluminação do LBPH, especialmente quando aplicado em imagens na escala de cinza, bem como a interferência mínima de ruídos.

2.2.1 LOCAL BINARY PATTERNS HISTOGRAMS

Descrito inicialmente em 1994, a operação LBP ficou conhecida como uma poderosa ferramenta de classificação de texturas, a qual teve a sua performance de detecção melhorada consideravelmente quando passou a ser combinada com histogramas de gradientes orientados (SALTON, 2018). Com o uso dos histogramas o algoritmo é chamado de LBPH. O seu funcionamento possui 4 fases: treinamento do algoritmo, aplicação da operação LBP, extração de histogramas e reconhecimento de face.

O treinamento funciona de forma semelhante ao que foi descrito nos algoritmos de detecção, porém as imagens que compõem a base de dados são das pessoas a serem reconhecidas. Além disso, cada foto do arquivo precisa de uma identificação, podendo ser um número ou o nome da pessoa, que dever ser a mesma para todas as fotos. Isso é feito para que o algoritmo associe a uma mesma identificação as diversas expressões que uma pessoa possa fazer, aumentando, portanto, a probabilidade de reconhecimento.

A forma básica da operação LBP consiste em classificar os *pixels* de uma imagem da seguinte maneira: a partir de uma região contendo 9 *pixels*, que são representados por uma matriz 3 X 3, com cada *pixel* assumindo um valor, correspondente a intensidade de cor, entre 0 e 255 (correspondentes aos tons de cinza). Em seguida usa-se o valor central da matriz como limiar para definir um novo valor, agora binário, para os outros 8 vizinhos. Caso o valor do *pixel* vizinho seja menor que o valor do limiar, atribui-se o valor 0, caso contrário o valor é 1. O próximo passo é concatenar os valores para ter uma sequência binária (daí o nome padrão binário local), que é convertida para um número decimal que passa a representar o conjunto de *pixels*, (SALTON, 2018). A figura 2.5 ilustra esse processo. A forma mais genérica da operação LBP usa vizinhanças circulares, figura 2.6 , as quais podem abranger qualquer quantidade de vizinhos com a variação do

raio, definidos pelo usuário, por meio de uma interpolação bilinear dos valores dos *pixels* (TIMO AHONEN, 2006).

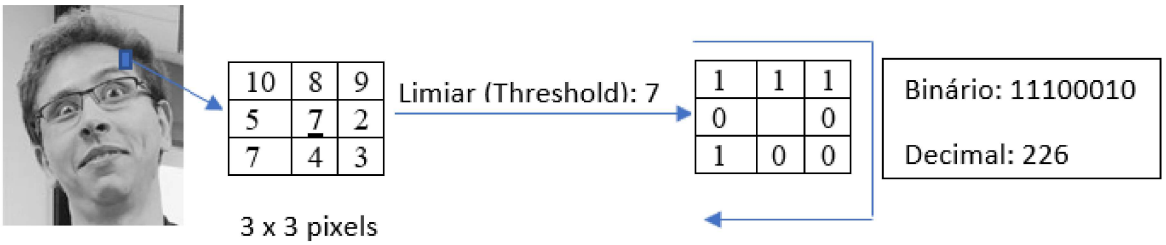

FIG. 2.5: Operação LBP

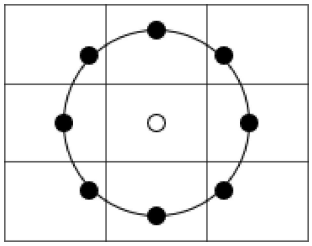

FIG. 2.6: Operação LBP - vizinhança circular

O procedimento seguinte é a extração dos histogramas, que consiste em agrupar os valores dos *pixels* classificados pela operação LBP, de forma que para cada região da imagem haverá um histograma contendo a quantidade de ocorrências de cada tom de cinza presente. Essas regiões que dividem a imagem são chamadas de *grids*, pois têm o mesmo aspecto de uma grade, e podem ter qualquer formatação quanto a quantidade de linhas e colunas: nas linguagens de programação os parâmetros que definem a quantidade de *grids* são *Grid x* e *Grid y*. Os histogramas extraídos de cada *grid* representam características locais, contendo 256 posições (de 0 a 255) relativas aos diversos tons de cinza. Em seguida, todos os histogramas locais são concatenados, gerando um histograma global que representa as características da imagem principal, conforme pode ser vista na figura 2.7 , (TIMO AHONEN, 2006). Dessa forma, para uma imagem de 9 x 9 *grids*, tem-se um histograma de 20.736 posições (9x9x256).

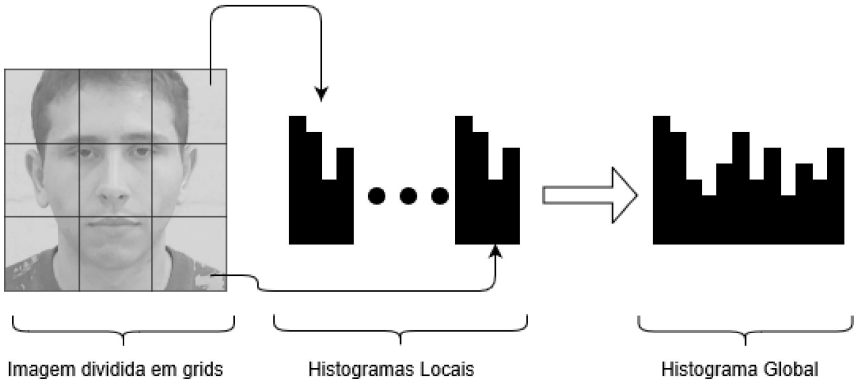

FIG. 2.7: Extração de Histogramas

O reconhecimento facial se dá, portanto, pela comparação entre os histogramas globais: da imagem de entrada com as imagens do conjunto de treino. O parâmetro de comparação utilizado é a distância entre os vetores de características, que são as estruturas de dados usadas para armazenar a informação dos histogramas. Na implementação do LBPH do OpenCV existem 4 métodos para o cálculo dessa distância: a distribuição qui-quadrado, a distância euclidiana, a distância euclidiana normalizada ou a fórmula do valor absoluto (OPENCV, 2018a). Após o cálculo das distâncias entre a imagem de entrada e as imagens do banco de dados, o algoritmo retorna a identificação da imagem que apresenta a menor distância em relação à imagem de entrada.

3 MODELAGEM DO SISTEMA

Este capítulo tem o objetivo de abordar a modelagem proposta para o sistema. Vale destacar que o sistema se propõe a não somente realizar o reconhecimento facial, como também permitir o armazenamento persistente das faltas apuradas, permitir a correção de eventuais erros de apuração e possuir mecanismos de consultas das faltas. O escopo do projeto abrange a apuração e o registro das faltas, sem incluir, portanto, a apuração de pontos perdidos.

3.1 LEVANTAMENTO DE REQUISITOS

O levantamento de requisitos é uma importante atividade para se entender o problema. Segundo (BEZERRA, 2007), essa etapa tem a finalidade de que desenvolvedores e usuários compartilhem da mesma visão do problema a ser solucionado. Buscou-se realizar uma série de procedimentos com a finalidade de levantar os requisitos funcionais e não-funcionais aqui expostos. Dentre as técnicas adotadas merecem destaque a análise de documentos, *Brainstorming* e entrevistas informais.

3.1.1 REQUISITOS FUNCIONAIS

Segundo (BEZERRA, 2007) esse tipo de requisito está ligado às funcionalidades do sistema. A seguir apresenta-se o resultado desse levantamento.

- RF01 - Apurar falta: O sistema deve apurar as faltas de um determinado tempo de aula em uma data, disciplina, para uma turma, usando o reconhecimento facial em uma fotografia da turma. As fotos devem ser tiradas por um dispositivo móvel com câmera ou selecionadas de sua galeria. Deve-se reconhecer várias faces em uma mesma fotografia, isso é, a fotografia é da turma e não individual. O professor deve poder enviar uma ou mais fotografias em um único tempo de aula, em operações de envio distintas. Assim ao enviar uma fotografia, e verificar que nem todos os alunos presentes receberam a presença, ele poderia enviar uma fotografia em adição à apuração anterior. Isso também ajuda, para os casos em que a turma não pode ser enquadrada em uma única fotografia. As informações referentes a data, isso é, dia, mês e ano devem ser extraídas do servidor;

- RF02 - Verificar lista de presença : Um aluno, coordenador ou professor deve ser capaz de exibir a lista de presença referente a um determinado tempo de aula, de um determinado dia para uma turma específica;
- RF03 - Editar lista de presença : O professor deve ser capaz de alterar as faltas de um tempo de aula, isso permite que ele corrija eventuais erros na apuração de faltas que usou o reconhecimento facial; e
- RF04 - Armazenar fotografia : Deve-se armazenar a fotografia com sua apuração no servidor, em uma pasta específica para isso, permitindo uma eventual auditoria.

3.1.2 REQUISITOS NÃO-FUNCIONAIS

Tratam sobre características do sistema, (BEZERRA, 2007) elenca alguns tipos como confiabilidade, desempenho, portabilidade, segurança e usabilidade. A seguir apresenta-se os requisitos não-funcionais levantados.

- RN01 - O Sistema deve apresentar interface com o usuário "Professor" compatível com o sistema Android;
- RN02 - O banco de dados, no qual se fará o armazenamento persistente, deve ser PostgreSQL;
- RN03 - As tarefas de captura de foto, reconhecimento facial e armazenamento de registros devem ser feitas em módulos distintos; e
- RN04 - Um professor deve ser capaz de utilizar o sistema após um treinamento de 15 minutos.

3.2 CASOS DE USO

Segundo (BEZERRA, 2007), o modelo de casos de uso consiste em um refinamento dos requisitos funcionais. Acrescenta que os casos de uso tratam sobre funcionalidades externas do sistema, assim não se abordam mecanismos internos. Assim, com base nos requisitos funcionais apresentados no tópico 3.1.1, tem-se a figura 3.1 que traz o diagrama de casos de uso. Vale destacar que o RF04 foi incorporado ao RF01 gerando um único caso de uso, isso se explica pois o armazenamento da imagem só ocorre posteriormente à apuração das faltas. Os atores **professor** e **usuário** são apresentados utilizando-se o

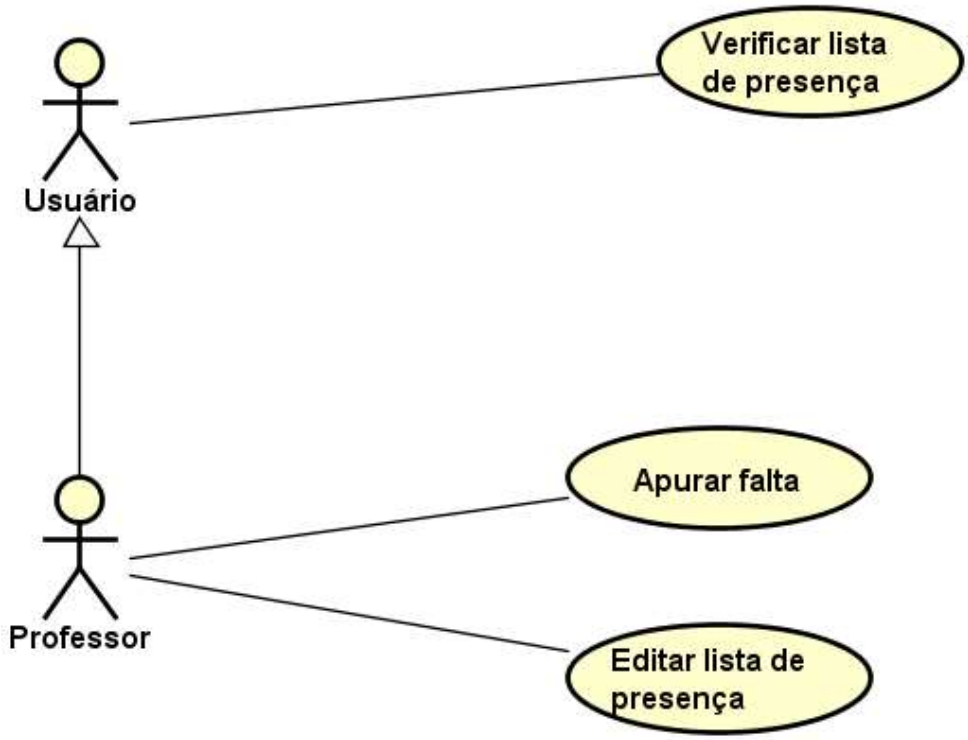

FIG. 3.1: Diagrama de casos de uso

relacionamento de generalização entre atores. Além disso usuário também comporta os alunos e os coordenadores.

Nas subseções seguintes é feito o detalhamento dos casos de uso **apurar falta**, **editar lista de presença**, **verificar lista de presença**.

3.2.1 APURAR FALTA

Identificador e nome: UC-01. Apurar falta.

Sumário: O ator utiliza esta função para apurar as faltas de um tempo de aula específico, através de uma fotografia da turma.

Ator Principal: Professor.

Pré-condição: O ator começa um tempo de aula e inicia o aplicativo Android Tchau Papeletas.

Fluxo Principal:

1- O ator solicita a função apurar faltas.

2- O sistema apresenta os campos código e tempo de aula para preenchimento e seleção, respectivamente. Além do campo para a fotografia.

3- O ator preenche com o código, formado pelo nome da turma + '.' + nome da matéria, e seleciona o tempo de aula.

4- O ator seleciona uma imagem previamente existente ou abre a câmera do dispositivo e fotografa a turma.

5- O ator clica em enviar.

6- O sistema acusa *upload* realizado com sucesso.

Pós-condição: A apuração de faltas daquele tempo de aula foi realizada e registrada no banco de dados, uma imagem com as faces identificadas foi salva no servidor.

3.2.2 EDITAR LISTA DE PRESENÇA

Identificador e nome: UC-02. Editar Lista de presença.

Sumário: O ator utiliza esta função para alterar as faltas apuradas no UC-01. Permitindo assim, correções na apuração pelo reconhecimento facial.

Ator Principal: Professor

Pré-condição: Uma apuração utilizando o reconhecimento facial já foi realizada para aquele tempo de aula.

Fluxo Principal:

1- O ator solicita a função editar lista de presença.

2- O sistema apresenta os campos código e tempo de aula, além de dia, mês e ano.

3- O ator preenche os campo nome da turma, nome da matéria, tempo de aula e a data, formada por dia, mês e ano.

4- O sistema exibe a papeleta de faltas preenchida, com as faltas apuradas pelo UC-01. Essa papeleta contém em seu cabeçalho a disciplina, o tempo de aula, data (dia, mês e ano) e turma. Em seu corpo, o código do aluno (matrícula), o nome do aluno e a presença ou ausência do aluno. Esse último campo é editável. Na parte inferior um botão de confirmar alterações.

5- O ator realiza as alterações de presença que julgar ser necessário. Ao final, pressiona o botão de confirmar alterações.

6- O sistema exibe uma mensagem de alerta, pedindo uma nova confirmação.

7- O ator confirma a operação.

8- O sistema acusa atualização realizada com sucesso.

Pós-condição: A apuração de faltas daquele tempo de aula foi alterada e registrada no banco de dados.

3.2.3 VERIFICAR LISTA DE PRESENÇA

Identificador e nome: UC-03. Verificar Lista de Presença.

Sumário: O ator utiliza esta função com a finalidade de verificar as faltas de um tempo de aula específico. Seria similar a observar a papeleta de faltas preenchida.

Ator Principal: Aluno, Coordenador e Professor

Pré-condição: Uma apuração utilizando o reconhecimento facial já foi realizada para aquele tempo de aula.

Fluxo Principal:

1- O ator solicita a função verificar lista de presença.

2- O sistema apresenta os campos código e tempo de aula, além de dia, mês e ano.

3- O ator preenche os campo nome da turma, nome da matéria, tempo de aula e a data, formada por dia, mês e ano.

4- O sistema exibe a papeleta de faltas preenchida, com as faltas apuradas pelo UC-01 e possivelmente alteradas pelo UC-02 . Essa papeleta contém em seu cabeçalho a disciplina, o tempo de aula, data (dia, mês e ano) e turma. Em seu corpo, o código do aluno (matrícula), o nome do aluno e a presença ou ausência do aluno.

Pós-condição: O sistema permanece inalterado.

3.3 CONCEITO GERAL DO SISTEMA

A partir das etapas anteriores, pode-se depreender o conceito geral do sistema e como ele será trabalhado. Segundo (PRESSMAN; MAXIM, 2016) a engenharia de *software* tem como objetivo principal a entrega do *software* em tempo hábil, com alta qualidade e que atenda aquilo que lhe foi requisitado. Para atender esse objetivo, (PRESSMAN; MAXIM, 2016) elenca uma série de princípios, dentre os quais, se destaca o conhecido princípio dividir para conquistar, que como o nome sugere se trata de subdividir o problema em partes menores. Posteriormente, os mencionados autores depreendem desse princípio um outro, que é o de construir *softwares* apresentando o que chamam de modularidade efetiva. Sobre isso, afirmam que: "cada módulo deve se concentrar exclusivamente em um aspecto bem restrito do sistema – deve ser coeso em sua função e/ou apresentar conteúdo bem preciso" (PRESSMAN; MAXIM, 2016, p. 109).

A modularidade permite que um módulo seja substituído por outro, sem que se tenha de alterar o sistema como um todo, além disso facilita a manutenção, uma vez que cada módulo tem um propósito bem definido. O sistema se dividirá em 3 módulos, a saber:

- *Módulo Servidor:* esse é o responsável por receber a fotografia tirada da turma, assim como os dados como turma, dia, hora e matéria. Ele deverá detectar as faces na fotografia e depois, utilizando o algoritmo escolhido, identificar cada uma delas.

Para cada face reconhecida deve-se atribuir a presença, a qual será enviada para o armazenamento persistente;

- *Módulo Mobile:* ele permitirá que o professor fotografe a turma, insira os dados mencionados acima, e encaminhe para o servidor, oferecendo ao professor uma interface amigável. Além disso, ele permitirá que o professor visualize a papeleta digital após a apuração e a corrija, se for o caso. Para tal, será projetado para o sistema *Android*; e

- *Módulo BD:* é onde se dará o armazenamento persistente das informações. Ele terá as informações sobre os alunos, como nome, turma, matrícula, disciplinas daquela turma, e é claro a presença/falta no tempo de aula.

Pode-se ainda detalhar as atividades envolvidas no processo de reconhecimento facial através de um diagrama de atividades, conforme a figura 3.2.

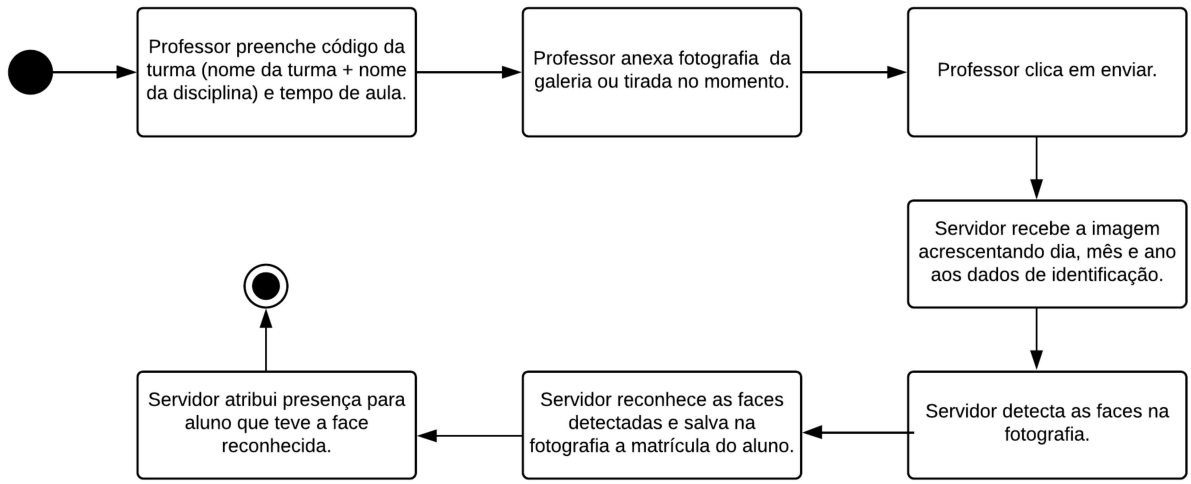

FIG. 3.2: Diagrama de atividades para o reconhecimento facial

4 MÓDULO SERVIDOR

Este capítulo tem o objetivo de abordar o 1º módulo desenvolvido para atender o objetivo proposto, o módulo do servidor. De maneira geral, como já mencionado, ele é o responsável por analisar a fotografia da turma, assim como os dados pertinentes. Ele deverá detectar as faces na fotografia e depois, utilizando o algoritmo LBPH, identificar cada uma delas. Para cada face reconhecida deve-se atribuir a presença, a qual será enviada para o armazenamento persistente, o banco de dados PostgreSQL. Ele está relacionado aos requisitos funcionais *RF01 - Apurar falta* e *RF04 - Armazenar fotografia.* Ele atende ao caso de uso *UC-01 Apurar falta.* Optou-se por abordar o processo de envio e recebimento das fotografias e dados, no módulo *mobile.* Assim a atividade do módulo servidor se inicia ao detectar uma fotografia na pasta faltas.

4.1 CONCEITO GERAL DO MÓDULO

Segundo (ARUBAS, 2013) o reconhecimento compara a face (desconhecida) que se quer reconhecer com um conjunto de treinamento de faces conhecidas. No conjunto de treinamento, são fornecidas as faces e a qual pessoa elas pertencem. Quando é solicitado ao algoritmo que reconheça alguma face desconhecida, ele usa o conjunto de treinamento para fazer o reconhecimento. Ele destaca ainda que, o LBPH analisa cada rosto no conjunto de treinamento de forma individual e independente.

O presente projeto utilizou a biblioteca OpenCV, onde se explorou os algoritmos *HaarCascade*, detecção de olhos e o de detecção de faces, para a detecção com maior grau de certeza das faces e o LBPH para o reconhecimento da face. O OpenCV, conforme (OPENCV, 2018c), é uma biblioteca gratuita para uso no meio acadêmico e comercial. Ela é voltada para visão computacional e aprendizado de máquina, tendo sido escrita primeiramente em C++. Possui interfaces nas linguagens C++, Python, Java e MATLAB.

Optou-se por utilizar a interface na linguagem Python, devido as características da linguagem, tais como facilidade de leitura e compreensão, além da fácil manutenção. Nesse sentido, (PERKOVIC, 2016) diz que Python é uma linguagem caracterizada como sendo de uso geral, que foi idealizada para fazer com que os programas sejam muito legíveis, isso é, priorizou-se a legibilidade. Além disso, Python também possui inúmeras bibliotecas, isso torna possível a escrita de aplicações sofisticadas, mas com aparência simples. Pelas

razões apresentadas, o mencionado autor conclui que "Python tornou-se uma linguagem de desenvolvimento de aplicações popular e também uma preferência como “primeira” linguagem de programação" (PERKOVIC, 2016, p. 22).

Vale destacar nesse momento uma particularidade do Python. Conforme (PERKOVIC, 2016), a versão 2 não é compatível com a 3. Isso significa que um programa escrito em Python 2, provavelmente, não irá funcionar usando um interpretador de Python 3. (PERKOVIC, 2016) considera que a versão 3 apresenta uma maior consistência que a 2. Nesse aspecto, optou-se por utilizar o Python 3.6.3, o qual foi, segundo (PYTHON.ORG, 2018), lançado em 03 de outubro de 2017.

Quanto ao OpenCV, a versão escolhida foi a 3.4.2. Essas considerações sobre as versões se justificam devido ao fato de que por vezes, métodos e formas de trabalho idealizadas para uma determinada versão, acabam sendo modificados em versões seguintes. Justamente para evitar esse tipo de problema, optou-se por utilizar uma única versão ao longo do desenvolvimento do trabalho.

4.2 ETAPAS PRÉ-RECONHECIMENTO

Antes de realizar o reconhecimento facial, faz-se necessário realizar o treinamento, isso é, gerar o conhecimento sobre a turma, gera-se um arquivo que concatene as informações extraídas de cada uma das fotografias, mantendo a identificação (a quem pertence) de cada uma das faces aprendidas. Para isso, segue-se as seguintes etapas ilustradas na figura 4.1

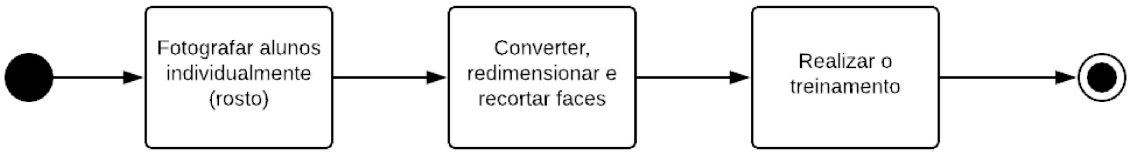

FIG. 4.1: Diagrama de atividades da fase pré-reconhecimento

Na primeira etapa deve-se fotografar o aluno de forma individual, com uma grande variedade de expressões, ângulos de fotografia e incidência de luz. Essa é uma etapa muito importante, pois essas imagens formarão a base de conhecimento (imagens a serem treinadas).

Depois disso, utiliza-se o artefato responsável pela conversão, desenvolvido ao longo do módulo servidor. Ele faz com que a fotografia seja transformada em uma escala cinza,

ocorra a detecção da face utilizando o *haarcascade frontal face*, depois disso, essa região é verificada com o *haarcascade eye*, a fim de eliminar-se resultados falso-positivos, isso é, eliminar a situação em que um objeto é reconhecido erroneamente como uma face. Por fim ocorre o recorte dessa face já na escala cinza e em um mesmo tamanho (720x720). Nesse ponto vale destacar que não são todas as fotografias que irão gerar faces válidas, uma vez que o *haarcascade frontal face* e o *haarcascade eye* podem não detectar nenhuma face, mesmo havendo. O programa recebe do usuário o nome da turma, e converte todas imagens contidas na pasta com esse nome. Oferece ainda ao usuário a opção de escolher se deseja exibir cada uma das fotografias antes da converter ou não. Ao final da conversão das imagens de cada aluno, exibe-se a quantidade de imagens lidas, e a quantidade de faces reconhecidas, isso é, que foram de fato aproveitadas e geraram uma nova imagem. A figura 4.2 ilustra esse procedimento, mostrando além do nome do arquivo, sua luminosidade.

```
ESTATISTICAS:
Matricula: 101
Exibindo arquivo: turma2\aluno.101.teste (1).jpg luminosidade: 153.40066741666666
Exibindo arquivo: turma2\aluno.101.teste (2).jpg luminosidade: 165.597196
Exibindo arquivo: turma2\aluno.101.teste (3).jpg luminosidade: 164.11084166666666
Exibindo arquivo: turma2\aluno.101.teste (4).jpg luminosidade: 168.75224625
Exibindo arquivo: turma2\aluno.101.teste (5).jpg luminosidade: 169.07403333333335
Exibindo arquivo: turma2\aluno.101.teste (6).jpg luminosidade: 170.04043741666666
 de 6 fotos, foram reconhecidas 5 faces.
```

FIG. 4.2: Informações estatísticas exibidas ao final do processo conversor

Uma padronização adotada foi o nome dado às fotografias, optou-se por utilizar a seguinte concatenação de nomes "aluno"+ "."+ o número da matrícula do aluno + "."+ nome da turma + "("+ número da fotografia + ").jpg", por exemplo: "aluno.101.comp18 (1).jpg", na figura 4.2 o nome de turma usado foi "teste". Essa padronização de nomenclatura facilita o processo de codificação e foi utilizada para armazenar o número da foto e a matrícula do aluno. As imagens de alunos de uma mesma turma foram agrupadas em uma mesma pasta. A figura 4.3 ilustra um exemplo de entrada e saída do procedimento descrito acima. Ao final dessa etapa, tem-se as imagens já convertidas, redimensionadas e recortadas.

4.2.1 O TREINAMENTO E SEUS PARÂMETROS

Na etapa seguinte gera-se o arquivo com as informações extraídas das faces, usando-se a metodologia do LBPH, anteriormente apresentada. Assim, através do artefato, desenvolvido ao longo do módulo servidor, "criador_de_inteligencia" irá se gerar o arquivo de treinamento. Esse é o primeiro momento que o algoritmo LBPH é de fato empregado.

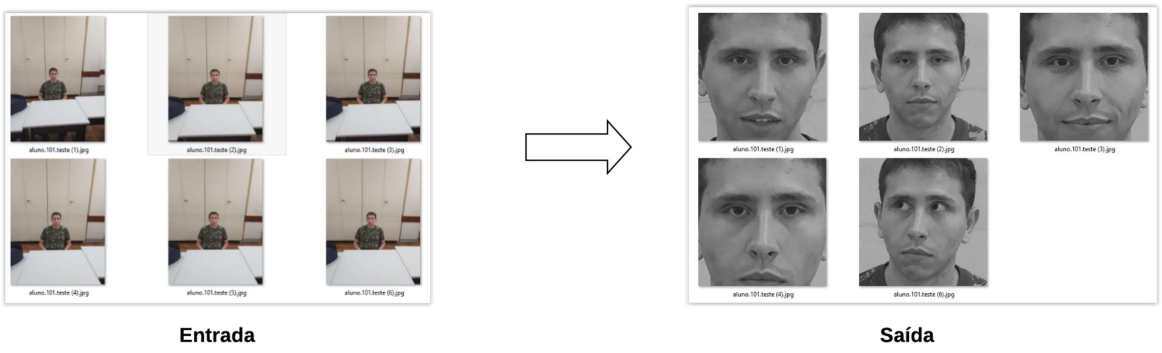

FIG. 4.3: Entrada e saída do conversor

Ao se invocar o método *LBPHFaceRecognizer_create*, tem-se de escolher alguns valores para determinados parâmetros. Quando explicou-se sobre o algoritmo LBPH, eles foram detalhados. São eles: *radius, neighbors, grid_x, grid_y, threshold*

O parâmetro *radius*, segundo (OPENCV, 2018b), consiste no raio usado para construir o Padrão Binário Local Circular. Raios maiores, tornam mais suaves as imagens, porém, carregam mais informações espaciais.

Para o parâmetro *neighbors*, (OPENCV, 2018b) explica que quanto maior o valor usado para a quantidade de vizinhos, maior será o gasto computacional, ela sugere o uso do valor 8. Esse valor corresponde a uma matriz 3X3, onde pode-se pensar no elemento central, isso é, elemento na linha 2 e coluna 2, como sendo o elemento a ser analisado. Os demais elementos dessa matriz, seriam os vizinhos utilizados.

Já para o parâmetro $grid\_x$ e $grid\_y$, número de células na horizontal e vertical respectivamente, apesar de (OPENCV, 2018b) explicar que muitas publicações sugerem o uso do valor 8, os testes realizados no presente projeto obtiveram melhores resultados com o valor 12.

Para melhor escolher os valores a serem utilizados para *neighbors*, $grid\_x$ e $grid\_y$ realizou-se alguns testes. Utilizou-se o banco de imagens disponibilizado pela Universidade de Yale. (UNIVERSITY, 2018) explica que esse banco contém 165 imagens em escala cinza, de 15 pessoas diferentes, havendo portanto 11 imagens por pessoa. Essas imagens variam as expressões da face e condições de iluminação. Além disso, (UNIVERSITY, 2018) esclarece que o banco de imagens está disponível publicamente para uso que não seja comercial. As imagens foram convertidas para o formato jpg, por questões de compatibilidade com a biblioteca OpenCV. Os testes consistiram em variando os parâmetros *neighbors*, $grid\_x$ e $grid\_y$ verificar o índice de acerto no reconhecimento facial.

Sobre os testes, foram escolhidas 2 imagens de cada pessoa para serem reconhecidas,

o que totaliza 30 fotografias. As imagens restantes, isso é, 9 imagens por pessoa, foram utilizadas no treinamento. Como saída do teste, obteve-se a Tabela 4.1, nela X e Y referem-se respectivamente aos valores utilizados em *grid_x* e *grid_y*, a coluna acertos é exibida em percentagem, já tamanho refere-se ao arquivo gerado pelo treinamento em megabytes, tempo 1 e tempo 2 foram obtidos em segundos e referem-se ao gasto no treinamento, e o total nos reconhecimentos, respectivamente. Por fim distância refere-se ao somatório das distâncias entre as imagens em análise (faces a serem reconhecidas) e as imagens que mais se aproximam delas. Vale destacar, que em virtude da apuração de tempo, os testes foram repetidos 5 vezes e o tempo mostrado trata-se de uma média aritmética.

Nesses testes, adotou-se o parâmetro *threshold*, no valor de 1000, nos próximos parágrafos serão feitos mais comentários sobre isso. O que vale destacar é que esse valor extremamente alto, força com que a face em análise seja reconhecida como uma das pessoas cadastradas no treinamento.

TAB. 4.1: Testes com os parâmetros do método *LBPHFaceRecognizer_create*.

Radius	Neighbors	X	Y	Acertos	Tamanho	Tempo 1	Tempo 2	Distância
1	4	8	8	66.66%	2.1	1.73	2.01	3.25
1	5	8	8	66.66%	3.9	1.64	2.17	4.45
1	6	8	8	66.66%	6.6	2.63	2.38	6.9
1	7	8	8	63.33%	10.8	2.83	2.76	6.8
1	8	4	4	63.33%	6.08	2.72	2.5	0.9
1	8	6	6	63.33%	11.6	3.44	2.85	3.39
1	8	8	8	66.66%	18.5	5.27	3.44	10.69
1	8	10	10	66.66%	26.7	5.72	4.15	21.53
1	8	12	12	70%	36.3	7.30	5.10	38.00
1	8	14	14	70%	47.0	8.36	6.09	62.61
1	9	12	12	70%	60.1	10.32	7.32	40.79
1	9	14	14	70%	79.3	11.62	8.69	65.87
2	8	8	8	63.33%	20.4	4.20	3.49	11.72
2	10	10	10	60%	82.8	13.72	8.95	30.55
3	8	8	8	60%	21.3	5.46	3.50	12.32

A partir da Tabela 4.1, concluiu-se pela escolha dos parâmetros da seguinte forma: *radius = 1, neighbors = 8, grid_x = 12, grid_y = 12*. Essa escolha se deu, pois foi o conjunto de parâmetros que obteve a maior taxa de acerto, combinada com um menor somatório de distância.

Por fim, o parâmetro *threshold*, é um dos mais importantes e de difícil escolha. Tem-se que uma situação comum no reconhecimento facial é dizer se uma determinada face

pertence ao conjunto de treinamento ou se é desconhecida. O parâmetro *threshold* refere-se ao limite, no teste cujo resultado foi mostrado acima, o algoritmo calcula a distância entre a face em análise e as que foram treinadas. O menor valor de distância encontrado é comparado com o *threshold*, se o primeiro for maior que o segundo a face em análise será classificada como desconhecida, caso contrário é atribuída ao elemento ao qual pertence a face de treinamento com menor distância.

No caso de uma turma de aula, todas as faces, que aparecem na fotografia tirada no início do tempo de aula, pertencem a um aluno regularmente matriculado. Em outras palavras, não existe a situação de uma face não pertencer a alguém que não esteja no conjunto de treinamento. Essa afirmação poderia levar a falsa ideia de que a solução seria colocar um valor extremamente elevado para o parâmetro *threshold* forçando o algoritmo a não classificar uma face como desconhecida. Percebeu-se em alguns casos, como na figura 4.4, que dado um aluno X, pode acontecer que na fotografia da turma (aquela a se apurar a falta), sua face fique distante das faces previamente utilizadas para a geração do conhecimento (treinamento) da turma. Ao se utilizar o parâmetro *threshold* elevado, o algoritmo acabou por classificar aquele aluno X, como sendo outro aluno, o que mais se aproxima, mesmo esse não sendo o aluno X. A figura 4.4 ilustra um exemplo em que isso aconteceria. Colocando um parâmetro de *threshold* elevado, a face em análise seria reconhecida como sendo a correspondente ao Aluno B, o que seria uma classificação errada.

A dificuldade na escolha desse parâmetro está no fato de quando é melhor classificar a face como desconhecida ao invés de tentar forçar uma classificação com distância maior, e que pode ser errada. Ao utilizar um valor elevado, pode-se forçar classificações erradas. Por outro lado, colocar valores baixos pode fazer com que se classifique uma face como desconhecida enquanto que ela poderia utilizar a classificação que lhe seria dada dentre o conjunto. Pensando no caso da figura 4.5, caso fosse adotado um *threshold* de 0.5, a análise concluiria que a face pertence a um elemento desconhecido. Por outro lado um *threshold* de 0.7 seria suficiente para uma classificação correta.

Feitas as considerações acima, realizou-se o procedimento similar ao descrito no início da subseção, isso é, utilizou-se o banco de imagens disponibilizado pela Universidade de Yale (UNIVERSITY, 2018), contendo 165 imagens, dais quais 30 foram utilizadas para serem reconhecidas e as outras 135 para a geração do arquivo de treinamento. Ao gerar o arquivo de treinamento, utilizou-se como parâmetro os encontrados anteriormente, *radius = 1, neighbors = 8, grid_x = 12, grid_y = 12*. A Tabela 4.2, traz o valor de *threshold*, a quantidade de faces detectadas com acerto, classificadas como desconhecidas e erradas. As quantidades estão em valores absolutos. Perceba que a classificação como desconhecida

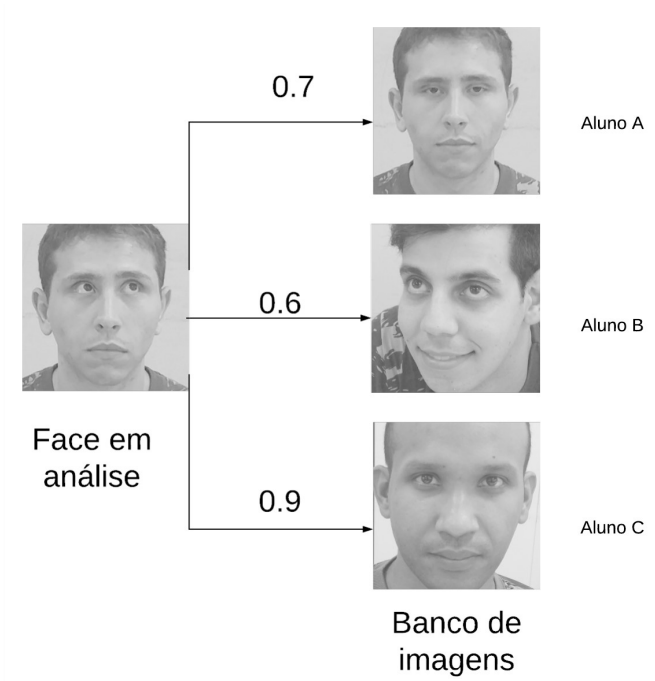

FIG. 4.4: Exemplo de distância menor que o limite, gerando classificação incorreta

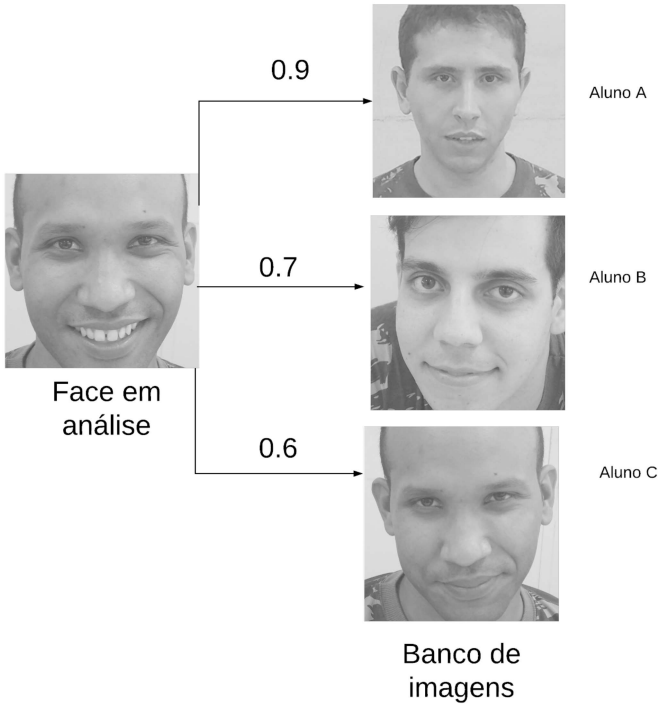

FIG. 4.5: Exemplo de distância maior que o limite, gerando classificação desconhecida

não é considerada como um erro, nem como um acerto. Em cada linha da tabela, a soma de acerto, desconhecida e erro totaliza 30, que é o número de imagens testadas.

TAB. 4.2: Testes para determinação do *threshold*.

threshold	Acerto	Desconhecida	Erro
10	3	27	0
15	3	27	0
20	3	27	0
25	6	24	0
30	7	23	0
35	8	22	0
40	13	17	0
45	17	13	0
50	18	12	0
55	18	12	0
60	18	12	0
65	18	11	1
70	18	11	1
75	19	10	1
80	19	9	2
85	20	7	3
90	20	5	5
95	20	4	6
100	20	3	7
110	20	2	8
130	21	0	9

A Tabela 4.2 corrobora aquilo que foi explicado nos parágrafos anteriores, a medida que se aumenta o valor do *threshold*, o algoritmo é forçado a escolher a opção mais próxima daquela face. Com isso, a partir do valor 85, o que ocorre é a diminuição da quantidade de faces desconhecidas, mas um aumento do reconhecimento incorreto. Apenas no *threshold = 130* é que todas faces são reconhecidas, como uma face conhecida. É nesse valor que se atinge o índice de acerto mencionado na Tabela 4.1, isso é, 70% de acerto e nenhuma imagem classificada como desconhecida.

Na escolha do *threshold*, o que deseja-se é maximilizar a taxa de acerto com uma mínima taxa de desconhecido. Da análise da Tabela 4.2 verifica-se que a máxima quantidade de acertos combinada com a menor taxa de desconhecidos ocorreu com o *threshold* igual a 50; 55 e 60. Isso significa que do conjunto de imagens analisadas, nenhuma obteve uma distância no intervalo entre 50 e 60. Assim o desempenho dos 3 valores foi igual. Optou-se por utilizar o valor 50, por ser o menor valor do conjunto. Nesse caso, distâncias maiores que 50, levariam a uma classificação como desconhecida, evitando-se assim o erro. Assim, o *threshold = 50*, dentre as imagens classificadas como conhecidas, obteve um índice de acerto de 100%, além disso do conjunto de imagens 40% foi classificado como desconhe-

cido. Pensando em uma sala de aula, o professor teria a opção de enviar uma nova foto para ser reconhecida, em adição a que já foi analisada , ou mesmo poderia fazer as alterações manualmente. Procedimentos no momento da fotografia ajudam a diminuir a taxa de desconhecidos, como exibir o rosto o mais similar do que foi cadastrado (imagens do treinamento).

Ainda no artefato "criador_de_inteligencia", pertencente ao módulo *servidor*, como entrada, o usuário digita o nome da turma, então busca-se as fotografias já convertidas na etapa anterior dessa turma, usa-se o método *train* da biblioteca OpenCV correspondente ao algoritmo LBPH. Tem-se então o processo de treinamento, ao final tem-se o arquivo que concatena as informações referentes a todas as fotografias de todos os alunos de uma turma. A figura 4.6 sintetiza esses procedimentos através de um diagrama de atividades. Perceba que os parâmetros pré-definidos referem-se aos valores estabelecidos para o presente projeto.

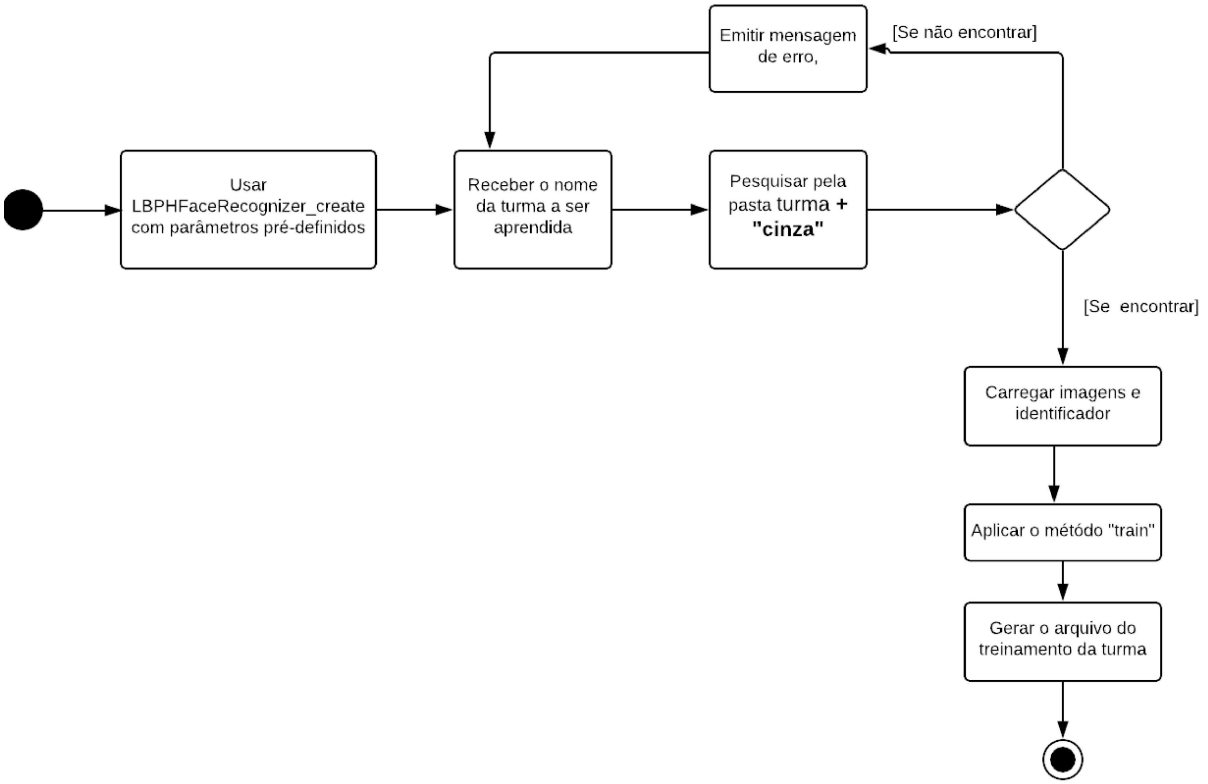

FIG. 4.6: Diagrama de atividades da fase treinamento

4.3 O RECONHECIMENTO

A etapa seguinte consiste em a partir de uma fotografia recebida da turma identificar quem nela encontra-se presente. Ela é realizada através do arquivo "reconhecimento_final.py".

Como mencionado no início do capítulo, inicialmente, ele procura por fotografias em uma pasta denominada "faltas". Quando detecta a presença de uma fotografia na pasta, inicia-se o processo. Do nome da foto, se extrai o nome da turma a qual ela pertence, a disciplina, o dia, mês e ano, o tempo de aula. A figura 4.7 mostra um exemplo com explicação dos campos, perceba que turma2 é o nome da turma e IA o nome da disciplina. A identificação da disciplina e do tempo de aula fazem-se necessárias, uma vez que uma disciplina pode durar mais do que 1 tempo de aula. Assim, um aluno pode ter faltado ao 1º tempo, mas ter comparecido no 2º.

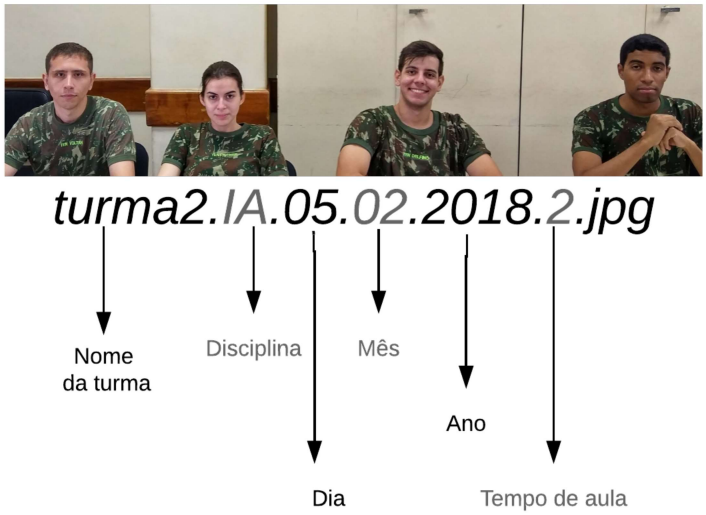

FIG. 4.7: Exemplo de nome da foto

Ao encontrar uma imagem, ela é carregada, tratada, isso é, convertida para escala cinza, redimensionada e tem suas faces frontais detectadas. A partir do nome da turma, busca-se o arquivo que concatena as informações do banco de imagens daquela turma, gerado na etapa treinamento. Tem-se aqui um ponto a ser destacado, o trabalho consiste em comparar características, usando o LBPH, daquela uma face com as faces previamente catalogadas para aquela turma. Como explicado, o algoritmo verifica a menor distância e depois compara com o*threshold*. Cada uma das faces é reconhecida, ou no caso da menor distância ainda ser maior que o *threshold*, ela é atribuída como desconhecida. O retorno do reconhecimento da face é o código do aluno. Aquele que teve sua face reconhecida

tem sua presença computada. Em cada uma das faces detectadas, desenha-se o retângulo delimitador e a matrícula do aluno. Essas informações sobre composição de turma e o registro da presença serão realizadas utilizando-se o banco de dados, e serão abordadas com mais detalhes no capítulo 6, que tratará sobre o banco de dados.

Após a imagem receber o retângulo com o nome do aluno, essas alterações são salvas e o arquivo migra para a pasta *apuradas*. Ao nome da imagem, antes do formato, é feita a adição do caracter '.' e o número daquela imagem, isso é, '.1' se for a primeira, '.2' se for a segunda e assim por diante. Por exemplo, caso a fotografia da figura 4.5 ao ser movida de pasta, verifique-se que foi a 5ª imagem apurada da turma2, na disciplina IA, no dia 05/02/2018 em seu 2º tempo de aula, então ao ser movido para a pasta apuradas, ela passaria a se chamar *turma2.IA.05.02.2018.2.5.jpg*.

A migração da imagem entre pastas garante que o laço *while* não acabe tendo de verificar fotos que já foram apuradas. Além disso, a manutenção da imagem, em uma outra pasta, permite que caso algum aluno conteste a falta, tenha-se um mecanismo alternativo para verificar essa informação. Nesse caso, a imagem da turma poderia ser utilizada, como mecanismo de verificação secundário. A figura 4.8 ilustra um diagrama de atividades para o "reconhecimento_final.py". Nele, é possível verificar que o servidor procura pela fotografia na pasta faltas, quando encontra inicia seu processo de reconhecimento, e ao final transfere a fotografia apurada (com as faces identificadas por um retângulo branco com a matricula do aluno) para a pasta apuradas. Dessa forma ele está sempre verificando a pasta faltas.

A figura 4.9 ilustra em *a)* a imagem de entrada do módulo Servidor, ou seja, a recebida, e em *b)* a imagem de saída, ou seja, a que será salva na pasta *apuradas*. Comparando *a)* e *b)*, pode-se perceber que adicionou-se o retângulo na face acompanhado da matrícula do aluno. Dessa forma atende-se o requisito funcional *RF04 - Armazenar fotografia*.

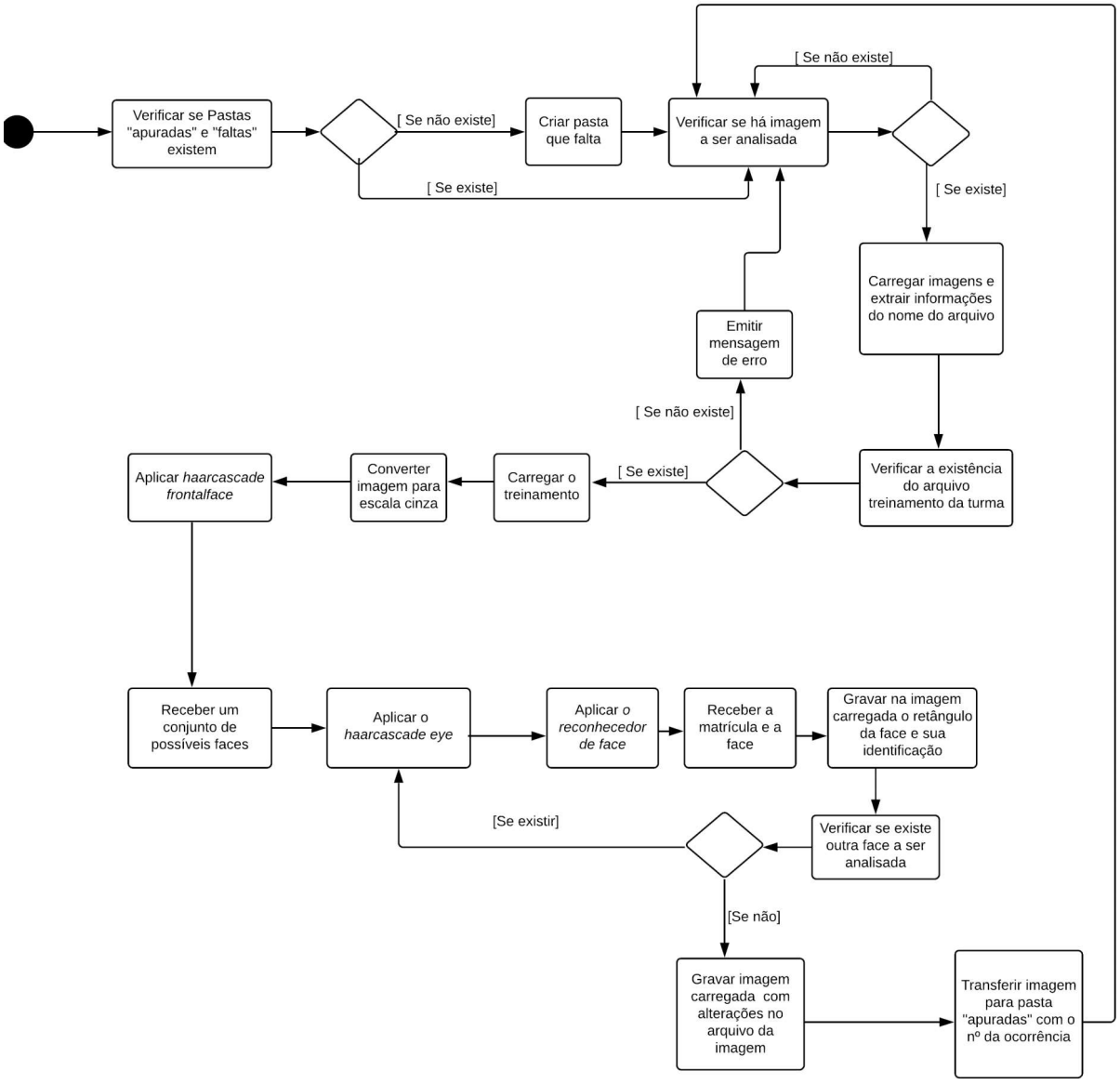

FIG. 4.8: Diagrama de atividades da fase reconhecimento

FIG. 4.9: Entrada e saída do módulo Servidor

5 MÓDULO MOBILE

Este capítulo tem o objetivo de abordar o 2º módulo do projeto, o módulo *mobile*. De maneira geral, ele é o responsável por retirar a fotografia da turma, captar os dados relativos ao código identificador, tempo de aula e, por fim, enviar toda a informação para o servidor e no servidor realizar o recebimento. O servidor conforme apresentado anteriormente, será responsável pela análise da foto.

5.1 CONCEITO GERAL DO MÓDULO

Desenvolvido para plataforma Android, a aplicação deverá proporcionar ao usuário, professor, uma interface simples e intuitiva para a realização das tarefas propostas. O professor deve ser capaz de realizar a seleção de uma fotografia, já salva na galeria ou tirada no próprio momento, completar o registro de informações essenciais e realizar o posterior envio. O diagrama de atividades pode ser visto na figura 5.1

5.2 PLATAFORMA ANDROID

Android é uma plataforma móvel baseada em Linux, cujo desenvolvimento propunha já desde os primórdios um desenvolvimento sob padrão aberto, ou seja, um padrão que está disponível para o público (OFICIAL ANDROID ARQUITETURA DA PLATAFORMA, 2018). O projeto foi desenvolvido pelo consórcio de empresas de tecnologias Open Handset Alliance, contendo gigantes do mercado como Google, Sony, Samsung e outras operadoras de telefonia e fabricantes de dispositivos (ALLIANCE, 2018). A pilha de software do Android pode ser vista em detalhes na figura 5.2, que mostra a maioria dos componentes da plataforma Android (OFICIAL ANDROID ARQUITETURA DA PLATAFORMA, 2018).

A equipe de desenvolvedores lideradas por Andy Rubinera criou no android a capacidade de acessar informações por meio de poucos toques extremamente intuitivos. Uma das grandes vantagens é tornar o sistema acessível para o usuário casual, ou seja, utilizadores que não possuem grande conhecimento tecnológico. De forma fluida, portanto, a plataforma permite a utilização dos mais diversos aplicativos sem demandar um processo de reaprendizado custoso por parte do usuário, visto que as características do dispositivos presente nos aplicativos torna familiar a experiência mesmo se tratando de novos

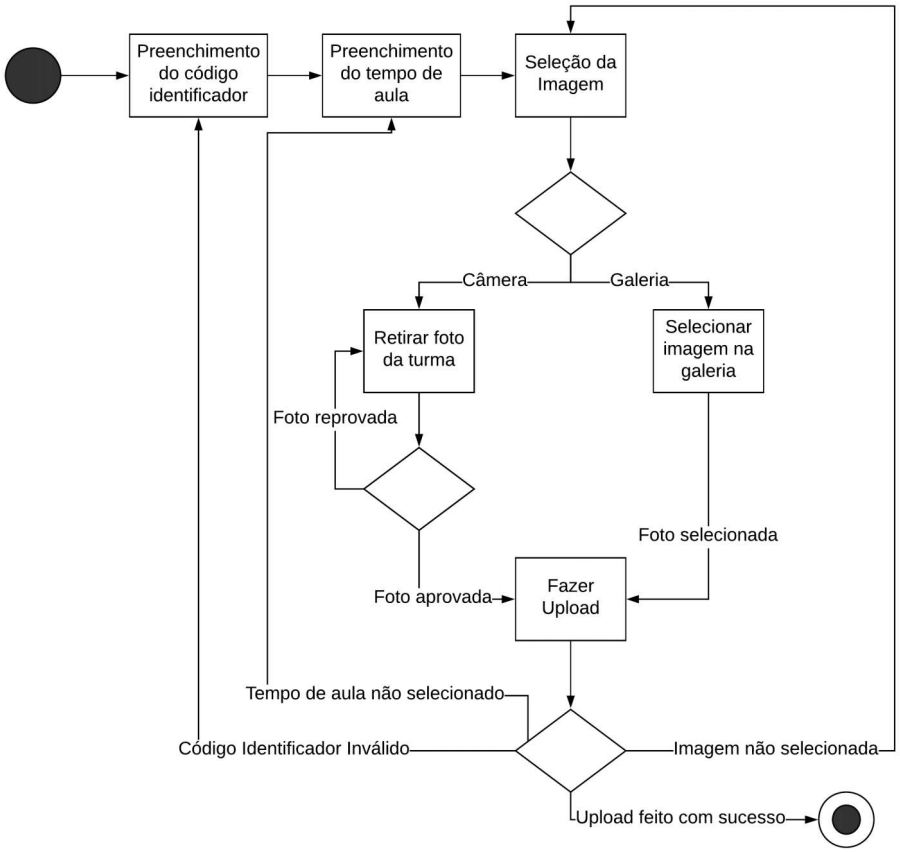

FIG. 5.1: Diagrama de atividades

aplicativos (DE SOUZA, 2018).

A possibilidade de usar componentes já prontos intrínsecos ao sistema operacional, como a interface de controle da câmera fotográfica, diminui o custo de desenvolvimento para o software em questão ao se escolher a plataforma Android, além de aumentar de forma relevante a qualidade do produto desenvolvido, já que os componentes reutilizados dentro da aplicação já foram exaustivamente testados antes de serem homologados e disponibilizados para uso. A utilização da câmera fotográfica através da aplicação pode ser vista na figura 5.6.

Todos esses fatores fizeram com que a plataforma Android dominasse o mercado de maneira avassaladora. Segundo pesquisa liderada pela corporação IDC, International Data Corporation, no primeiro quarto de ano de 2017, a plataforma estava presente em 85% dos aparelhos mobile ao redor do mundo, justificando o rótulo de plataforma mais popular do mundo, presente até mesmo no site oficial Android (IDC, 2018) (ENTERPRISE, 2018). No Brasil, a hegemonia é ainda mais significante, segundo dados da empresa

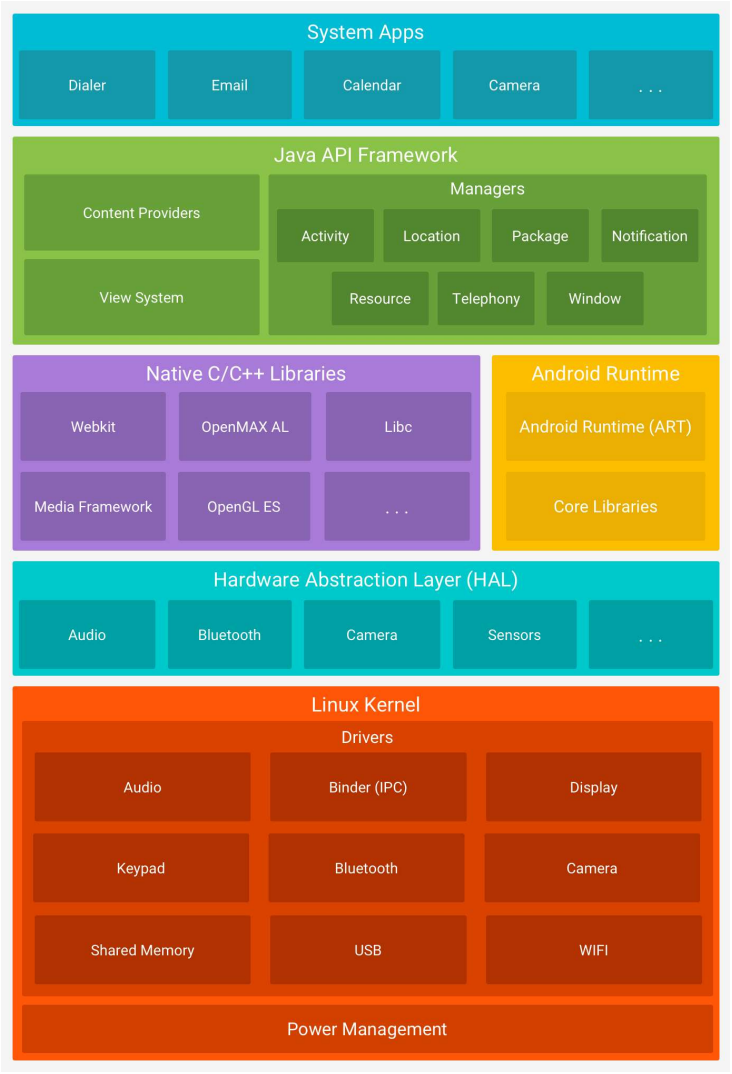

FIG. 5.2: Pilha de software do Android v. abril 2018
(OFICIAL ANDROID ARQUITETURA DA PLATAFORMA, 2018)

Kantar, tendo alcançado 93% do mercado nacional em março de 2018 (KANTAR, 2018).

Esses dados corroboram a escolha da plataforma para construção do aplicativo, que busca ser o mais acessível possível.

5.2.1 AMBIENTE DE DESENVOLVIMENTO

Buscando tornar o aplicativo o mais acessível possível, este foi desenvolvido tendo como base a API 15: Android 4.0.3 (*IceCreamSandwich*), sendo compatível com quase a totalidade de aplicativos existentes, segundo estatísticas do próprio Android Studio.

Utilizou-se a versão estável mais atual do Android Studio, visando tornar o código compatível com os recursos mais recentes oferecidos por esse ambiente de desenvolvimento. A figura 5.3 contém as informações sobre a versão utilizada, contruída em 4 de Junho de

2018.

FIG. 5.3: Versão Android

5.2.2 LINGUAGEM DE PROGRAMAÇÃO

A linguagem de programação utilizada foi Java, utilizada pelo Android Software Development Kit, SDK. A utilização do SDK é a escolha recomendada oficialmente pela Google, visto que a Android Native Development Kit, NDK, só é aconselhável caso verdadeiramente se necessite ganhar maior desempenho, já que se rodaria a aplicação diretamente no processador (OFICIAL ANDROID NDK, 2018). Esse ganho extra de desempenho, que aumentaria a complexidade do código, não é justificável visto que o processamento exigido durante toda a aplicação é extremamente reduzido. Além disso, ao utilizar o SDK se tem a garantia de portabilidade de dispositivo independentemente da arquitetura do processador, corroborando com a tentativa de tornar o aplicativo o mais acessível quanto possível (ANDROIDXCOMMUNITY, 2018).

5.2.3 DISPOSITIVOS DE TESTE

Os dispositivos, ilustrados pelas figuras 5.4 e 5.5, foram utilizados neste projeto para a realização de testes da aplicação desenvolvida.

Nenhuma diferença de desempenho foi observada ao utilizar a aplicação nos diferentes dispositivos. Ambos aparelhos foram testados utilizando o Android 8.0.0.

A única funcionalidade da aplicação verdadeiramente dependente de alguma característica do dispositivo é o acionamento da câmera, cuja utilização através da aplicação só pode ser executada caso o dispositivo possua câmera. Colocou-se uma verificação que libera o clique de acesso a câmera apenas para aqueles dispositivos que possuem o hardware. Devido a incapacidade de tirar fotos caso a câmera não exista, a verificação impede

FIG. 5.4: MOTO Z Play - Processador Snapdragon^{TM} 625 Qualcomm® 3GB de RAM

FIG. 5.5: One Plus 3 - Processador Snapdragon^{TM} 820 Qualcomm® 6GB de RAM

que o aplicativo gere um erro crítico. Ainda assim, contudo, é possível realizar envio completo das informações para retirada de faltas mesmo em um celular sem câmera, bastando utilizar uma imagem salva na galeria. A utilização da câmera através da aplicação pode ser revista na figura 5.6, já mencionada anteriormente.

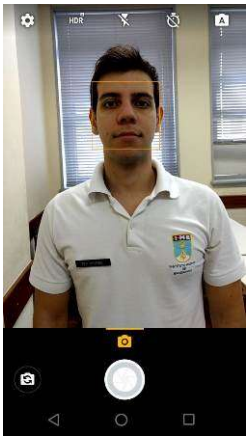

FIG. 5.6: Utilização da câmera através da aplicação

5.3 A APLICAÇÃO

O objetivo da aplicação, conforme descrito no início do capítulo é garantir o preenchimento de informações necessárias para a retirada de faltas de maneira fluída e intuitiva. A figura 5.7 mostra a tela inicial da aplicação.

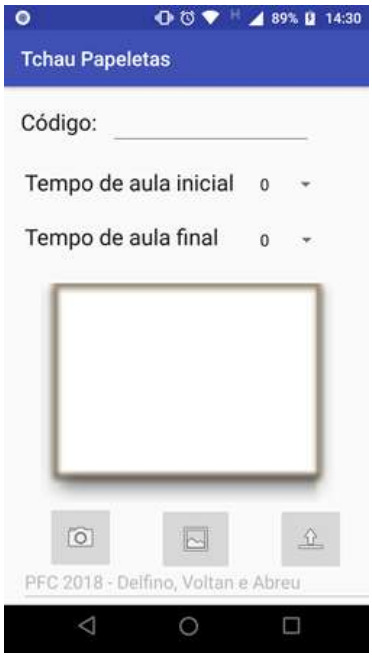

FIG. 5.7: Tela inicial da aplicação

5.3.1 PREENCHIMENTO DO CÓDIGO IDENTIFICADOR

O preenchimento do código identificador, figura 5.8 ocorre ao clicar-se sobre o campo destinado para o mesmo, quando abrir-se-á o teclado e o professor poderá digitar a referência à turma e à matéria por ele ministrada. No exemplo a seguir, ilustradro na figura 6.9, tem-se turma1 e disciplina Redes1, gerando o código identificador "turma1.Redes1".

No atual momento de desenvolvimento optou-se por deixar esse campo com preenchimento livre de tal forma a possibilitar utilização por qualquer turma e qualquer disciplina, não o limitando a escolhas restritas das disciplinas de alguma seção de ensino, por exemplo. O objetivo é tornar o aplicativo compatível com futuras aplicações e diferentes turmas e disciplinas.

5.3.2 PREENCHIMENTO DO TEMPO DE AULA

O preenchimento dos tempos de aula ocorre clicando sobre os tempos de aula correspondentes, figura 5.10. Como os tempos de aula possíveis já estão previamente definidos,

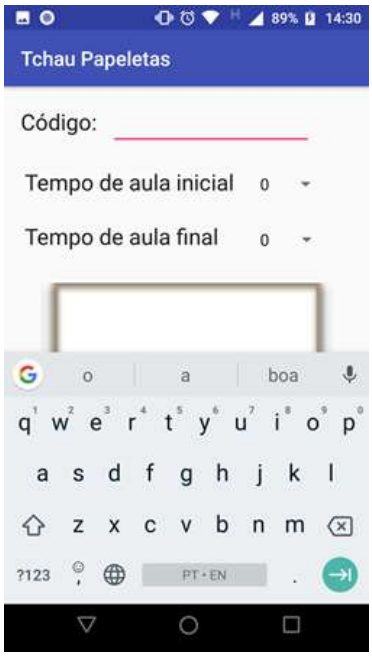

FIG. 5.8: Teclado para digitação

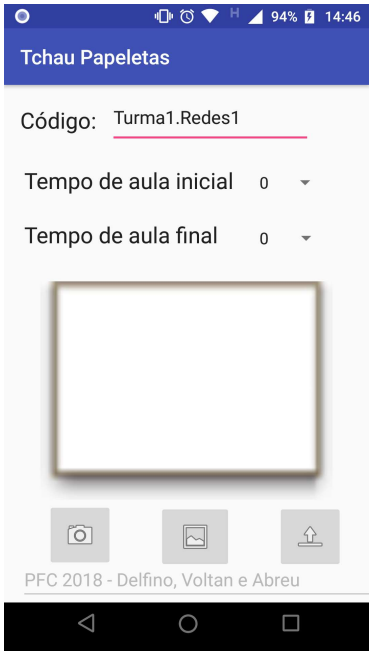

FIG. 5.9: Código preenchido

optou-se por realizar o preenchimento desse campo através de um controle giratório. Basta clicar na opção desejada para o tempo de aula, quando abrir-se-ão opções de tempos do 1 ao 11, intervalo de horário máximo segundo quadro de horários vigente no IME.

5.3.3 AQUISIÇÃO DA IMAGEM

A aquisição da imagem pode ser feita de duas maneiras diferentes:

FIG. 5.10: Preenchimento Tempo de aula

- Por meio do acesso à galeria de imagens do dispositivo; ou
- Por meio da captura de uma imagem através da câmera do dispositivo.

5.3.3.1 CÂMERA

Para iniciar abertura da câmera basta clicar sobre o ícone da câmera. A interface de câmera do sistema operacional Android vigente então abrirá, permitindo a retirada de fotos já familiar ao usuário. Após retirada da foto, o usuário terá opção de confirmar ou rejeitar a foto obtida, figura 5.11. Exibir-se-á na tela inicial da aplicação, então, a foto tirada, caso essa tenha sido aprovada pelo usuário.

5.3.3.2 GALERIA

Para iniciar a escolha de foto através da galeria basta clicar sobre o ícone da galeria. A interface de galeria do sistema operacional Android vigente então abrirá, permitindo a escolha de fotos já familiar ao usuário, conforme ilustrado na figura 5.12. Exibir-se-á na tela inicial da aplicação, então, a foto selecionada pelo usuário. Um exemplo de exibição da foto escolhida na tela inicial da aplicação pode ser vista na figura 5.13.

FIG. 5.11: Aguardando aprovação da foto tirada

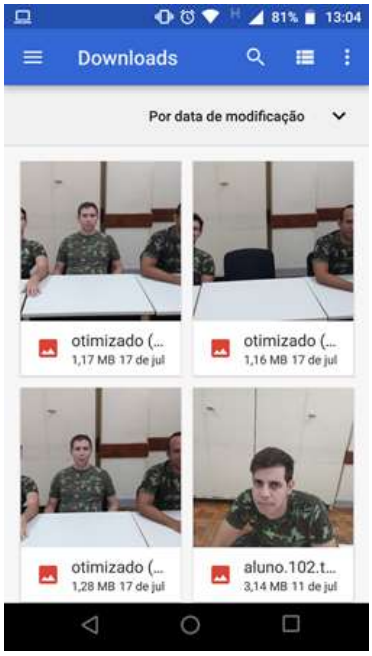

FIG. 5.12: Seleção da foto a partir da galeria

5.3.4 UPLOAD PARA O SERVIDOR

Para enviar todos os dados para o servidor, basta clicar no ícone de *upload*. A aplicação enviará toda a informação coletada caso todos os campos tenham sido preenchidos de maneira adequada. Na impossibilidade dos dados serem aprovados, a aplicação notificará o usuário por meio de notificações qual campo está impossibilitando o sucesso da execução,

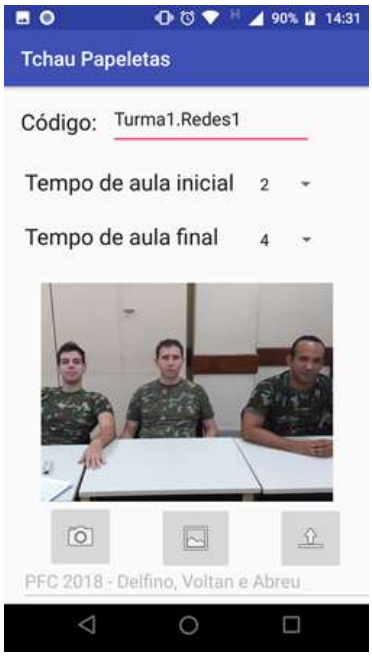

FIG. 5.13: Foto selecionada com sucesso

conforme mostram as figuras 5.14, 5.15 e 5.16.

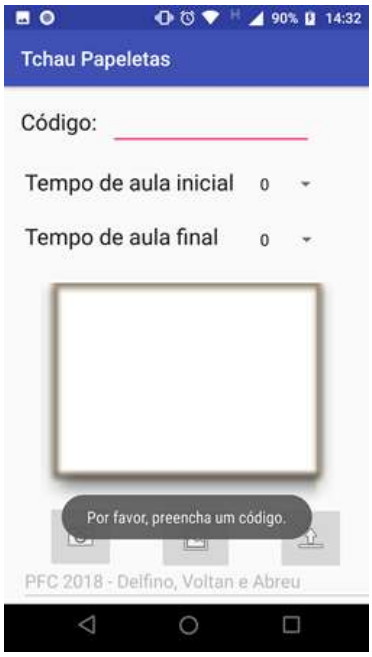

FIG. 5.14: Erro no preenchimento do código

Caso todas as informações tenham sido preenchidas corretamente, notificar-se-á o usuário do upload feito de maneira correta, de maneira temporária e também de maneira persistente, figura 5.17. Além disso, não mais se exibirá a foto enviada, garantindo a percepção de que aquela foto já foi enviada para o servidor, impedindo assim que o professor persista na tentativa de enviar a mesma informação de maneira duplicada.

FIG. 5.15: Erro no preenchimento do tempo de aula

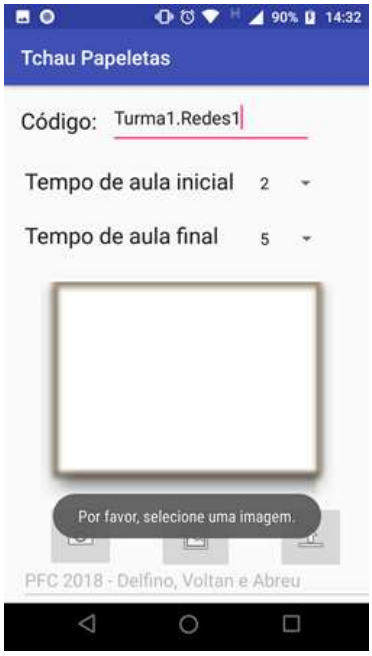

FIG. 5.16: Erro na seleção de Imagem

No módulo 3, módulo do banco de dados, estudar-se-á a exibição dos resultados produzidos pelo reconhecimento facial. O módulo 3 também será o responsável pelo armazenamento persistente dessas informações no banco de dados.

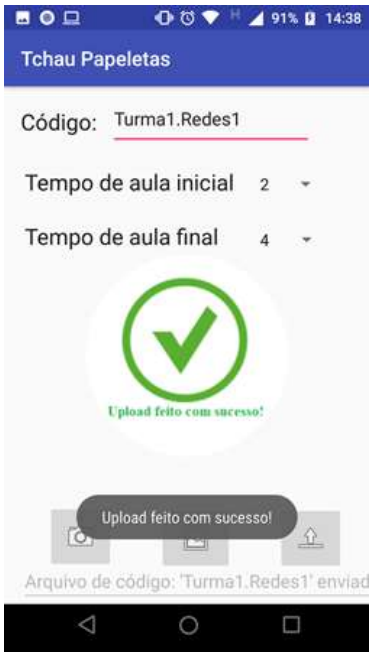

FIG. 5.17: Upload efetuado com sucesso

5.4 ESPECIFICAÇÕES DO ENVIO

O envio das informações dar-se-á através de um *socket*, um ponto final de comunicação entre duas máquinas. Para o caso do trabalho em tela, o *smartphone* utilizado pelo professor contendo a aplicação instalada, cliente, e o computador contendo o servidor.

Utilizar-se-á o protocolo TCP para a transmissão. A escolha pela não utilização do *DatagramSocket*, que utilizaria o protocolo UDP, deu-se fundamentalmente pela verificação de envio correto das informações presente na utilização do protocolo TCP (OFICIAL ANDROID SOCKET, 2018). Apesar de possibilitar um envio mais rápido, o UDP não é confiável, já que não há garantia que o pacote chegou ao destino da maneira correta, fator impeditivo para seu uso na aplicação em questão (JAMES F. KUROSE, 2009). Além disso, o ganho de desempenho proporcionado pelo protocolo UDP não é almejado com afinco, visto que enviar-se-á uma foto única, não executando uma transmissão contínua de vídeo, por exemplo, ou seja, não necessita de alta taxa de transmissão (JAMES F. KUROSE, 2009).

Todas as informações serão enviadas através de uma mensagem única. Para isso, converter-se-á a imagem selecionada. Para isso, criar-se-á o arquivo *Bitmap* da foto da turma especificada, obtida por meio da galeria ou captura de câmera. Tal arquivo será, então, convertido em uma stream de dados, utilizando o formato JPEG para a compressão em um *array* de *bytes*. A escolha do formato JPEG deu-se fundamentalmente por

ser permitido definir a qualidade de imagem após conversão, especificada como máxima e permitindo o reconhecimento abordado ao longo do capítulo 4. A informação agora escrita nesse *array* de *bytes* é finalmente convertida em base 64 como uma *string*, para ser adicionada a mensagem enviada pelo *socket*, conforme já mencionado. Por fim, enviar-se-á a mensagem contendo as *strings* de Código, Tempo de aula e Foto convertida em *String*, sendo essas informações separadas por vírgula e incluídas nessa ordem em uma padronização exigida para restauração das imagens uma vez que essas sejam recebidas no Servidor TCP. A decodificação realizada no Servidor TCP exige tal padronização.

5.4.1 CLIENTE TCP

A execução da tarefa em Android é feita por uma instância da classe *SocketImpl*, através da importação do pacote java.net.Socket. A extensa documentação a respeito da utilização de socket em plataformas android no site oficial de referências, (OFICIAL ANDROID SOCKET, 2018), bem como o cumprimento de todos os requisitos propostos para o trabalho foram decisivas para a escolha do método de transmissão.

5.4.2 SERVIDOR TCP

O servidor TCP fará a coleta da informação recebida, separando os dados recebidos e convertendo a imagem recebida para o formato o JPEG, já colocando toda a informação necessária no nome do arquivo criado, seguindo as padronizações especificadas ao longo do módulo 4. Optou-se por manter a mesma linguagem de programação utilizada para o desenvolvimento da aplicação Android, buscando manter a coerência dentro do módulo desenvolvido. Assim sendo, toda a aplicação do servidor TCP, que conta com interface gráfica, foi desenvolvida em linguagem de programação Java.

Após clicar em "Iniciar serviço", o servidor TCP atuará de maneira contínua, esperando o recebimento de solicitações feitas através da aplicação e salvando as respectivas imagens com os respectivos códigos associados, conforme ilustrado nas figuras 5.18, 5.19 e 5.20. Nenhuma ação é necessária para o recebimento das futuras imagens, sendo o processo de decodificação e posterior registro em disco da imagem recebida com o respectivo nome identificador contínuo.

FIG. 5.18: Iniciando serviço: Passo 1

FIG. 5.19: Iniciando serviço: Passo 2

FIG. 5.20: Iniciando serviço: Passo 3

6 MÓDULO BD

Este capítulo tem o objetivo de abordar o último módulo desenvolvido para atender o objetivo proposto: o módulo do BD. Ele é responsável por oferecer o armazenamento persistente ao sistema e sua edição, possibilitando a visualização das faltas e sua alteração. As atividades de consulta e alteração da papeleta são realizadas através de um servidor web, desenvolvido nesse último módulo. Ele está relacionado aos requisitos funcionais *RF02 - Verificar lista de presença* e *RF03 - Editar lista de presença.* Sobre o banco de dados a ser empregado, o *RN02 - O banco de dados, no qual se fará o armazenamento persistente, deve ser PostgreSQL* torna isso claro. Além disso, o módulo encontra-se relacionado com os 3 casos de uso *UC-01 Apurar falta, UC-02 Editar Lista de presença, UC-03 Verificar Lista de Presença.*

6.1 CONCEITO GERAL DO MÓDULO

Segundo (ELMASRI; NAVATHE, 2010), os bancos de dados e suas tecnologias representam um papel crítico nos mais diversos sistemas nos quais se inserem, sendo que o nome banco de dados está relacionado a um conjunto de dados relacionados. No sistema proposto, esse conjunto de dados seria a composição das informações relacionadas à presença dos alunos, ou seja, disciplina, tempo de aula, matrícula do aluno, entre outros.

O banco de dados escolhido foi o *PostgreSQL*, visando atender ao RN02 que estabelece que "o banco de dados, no qual se fará o armazenamento persistente, deve ser PostgreSQL". Segundo (OFICIAL POSTGRESQL, 2018), o *PostgreSQL* é um banco de dados relacional, com código aberto e que usa e estende a tradicional linguagem SQL. Ele teve origem na Universidade da Califórnia em Berkeley, no ano de 1986. Nesse contexto, o banco de dados relacional conta com mais de 30 anos de desenvolvimento e aperfeiçoamento. A versão escolhida foi a 9.6.10. Optou-se por utilizar o *pgAdmin III*, em sua versão 1.22.2. O *pgAdmin III*, segundo (OFICIAL PGADMIN III, 2018), é uma ferramenta para o gerenciamento de banco de dados *PostgreSQL* que possui interface gráfica e ferramentas que auxiliam no gerenciamento e manutenção do banco. Sobre a licença, (OFICIAL PGADMIN III, 2018) informa que a ferramenta é disponibilizada gratuitamente.

6.2 MODELAGEM DO BANCO DE DADOS

O Modelo Entidade-Relacionamento, segundo (ELMASRI; NAVATHE, 2010), é um modelo conceitual, muito popular, que trabalha com uma abstração de alto nível, empregando entidades, relacionamentos e atributos em sua modelagem.(ELMASRI; NAVATHE, 2010) explica ainda que o Diagrama Entidade-Relacionamento (ER) é a representação gráfica para um esquema. Buscou-se modelar um banco de dados de maneira simplificada, mas que atenda ao propósito do sistema de registrar a apuração das faltas por tempo de aula. Para a construção do Diagrama ER, utilizou-se o software brModelo 3.0. (CÂNDIDO, 2018) informa que ele foi desenvolvido por Carlos Henrique Cândido, e trata-se de uma ferramenta para a modelagem relacional. Destaca-se o fato do software desenvolvido por Carlos Cândido ser nacional e gratuito.

A figura 6.1 ilustra o Diagrama ER para o sistema. Nele, tem-se as entidades *aluno, turma, disciplina, professor, aula.* Cada aluno está matriculado em uma turma e cada turma cursa uma disciplina. Cabe aqui observar que, no Instituto Militar de Engenharia, os alunos devem cursar, obrigatoriamente, as disciplinas previstas para aquele semestre. Assim sendo, uma turma tem sua grade de disciplinas rígida, tendo sido determinadas previamente as disciplinas a serem cursadas em um dado semestre. Um ou mais professores podem ministrar uma disciplina ao longo do semestre. Ao evento específico no qual um professor ministra uma disciplina, durante um intervalo predefinido de tempo, deu-se o nome de aula. Optou-se por considerar aula como sendo uma entidade devido à ênfase do sistema. Por fim, o aluno pode comparecer a uma aula ou não. Pode-se notar que aluno e professor guardam certa semelhança, sendo possível para ambos herdarem da entidade pessoa, o que não foi feito no diagrama apenas com a finalidade de torná-lo mais legível. Quanto a cardinalidade do relacionamento, optou-se por seguir a terminologia proposta por (BATINI et al., 1991), por exemplo, um aluno está matriculado em uma única turma, podendo uma turma ter um ou mais alunos.

Na fase seguinte, usando o modelo ER, construiu-se o modelo de dados relacional. (ELMASRI; NAVATHE, 2010) explica que esse modelo trata o banco de dados como sendo um conjunto de relações. Representou-se através da figura 6.2 o Diagrama para o esquema do banco de dados relacional tchau_papeleta. Quanto ao formato da representação, adotou-se o que (ELMASRI; NAVATHE, 2010) sugere em seu capítulo 5, página 98.

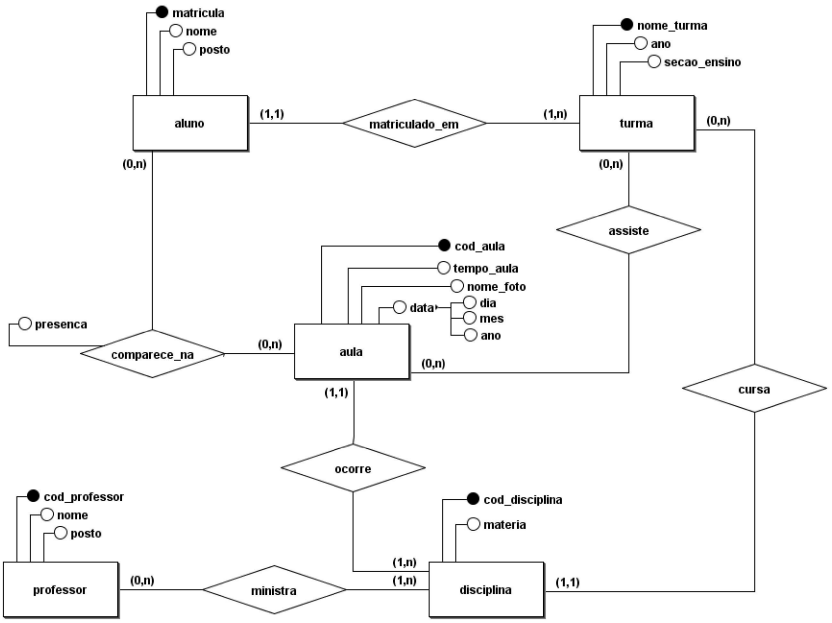

FIG. 6.1: Diagrama ER do sistema

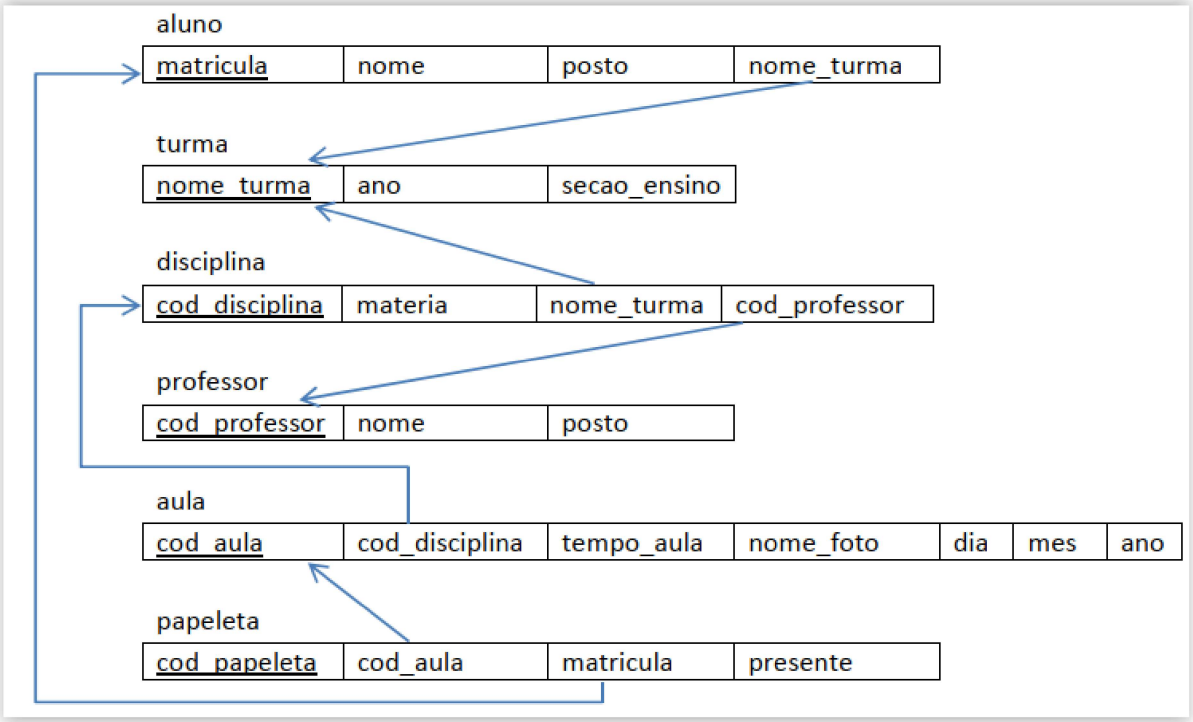

FIG. 6.2: Diagrama para o esquema do banco de dados relacional tchau_papeleta

6.3 CRIAÇÃO DO ESQUEMA

Na fase seguinte, criou-se um banco de dados relacional do tipo postgreSQL. No banco de dados proposto, a tabela **aluno** tem as colunas matrícula, nome, posto e cod_turma.

A matrícula, por ser um identificador único, é a chave primária. Para a coluna posto, os alunos militares têm a abreviação de sua graduação ou posto. Os alunos da reserva receberiam a palavra *civil*. O nome_turma faz referência à turma, sendo uma chave estrangeira. A **turma** seria composta pelo nome_turma, um identificador único da turma que funciona como chave primária, além do ano e da secao_ensino. O ano se refere ao ano da graduação na qual a turma se encontra e.g: o 5º ano teria ano igual a 5. A secao_ensino representa a seção de ensino, sendo essa uma sigla sigla, e.g: SE8. O **professor** é composto por um cod_professor, chave primária, além do nome e posto. A **disciplina** é formada por cod_disciplina, matéria (nome por extenso da disciplina), nome_turma (chave estrangeira fazendo referência à turma), cod_professor1 e cod_professor2 (chaves estrangeiras que referenciam professor). Nesse momento, vale destacar que preferiu-se utilizar 2 campos para professor, priorizando o desempenho ao espaço, isso se explica pois no IME, as disciplinas tem, em geral, no máximo 2 professores encarregados de uma disciplina. Criar uma tabela relacionando professor e disciplina, nesse caso teria um impacto negativo no desempenho, e traria um benefício de evitar redundância pequeno. A **aula** é composta por cod_aula (chave primária), dia, mês e ano (compondo a data), cod_disciplina (chave estrangeira que se refere à disciplina), tempo_aula e nome_foto. Por fim, a **papeleta** é composta por cod_papeleta (chave primária), cod_aula (chave estrangeira que refere aula), matrícula (chave estrangeira que refere aluno). Além desses, ter-se-á a coluna presente, um *boolean* no qual *true* representa que o aluno compareceu à aula e *false* que o aluno faltou.

Criou-se um *Database* denominado IME. A partir dele, criou-se o esquema e suas tabelas. A figura 6.3 mostra a criação do esquema denominado *tchau_papeleta*. A seguir, as tabelas *turma, aluno, professor, disciplina, aula e papeleta* são criadas. Após a criação do esquema e tabelas, fez-se a inserção de dados, conforme ilustrado na figura 6.4. Para o funcionamento do sistema, os professores, alunos, disciplinas e turmas já devem ter sido previamente cadastrados, conforme ilustra a figura 6.4.

Quando o professor, utilizando o aplicativo "Tchau Papeletas", fizer o envio da fotografia da turma, ele também enviará nome da turma, representado no esquema como *nome_turma*, e o nome da disciplina, que aparecerá como *materia*. Essas duas informações serão necessárias para identificar a turma de maneira inequívoca, uma vez que uma mesma matéria pode ser oferecida para mais de uma turma. Por exemplo, a matéria "Fenômenos de Transporte"é ministrada com o mesmo nome tanto para o 3º ano de Engenharia Química quanto para o 3º ano de Engenharia da computação. Trata-se, contudo, de disciplinas distintas, com códigos (*cod_disciplina*) e conteúdo diferentes. A figura 6.5

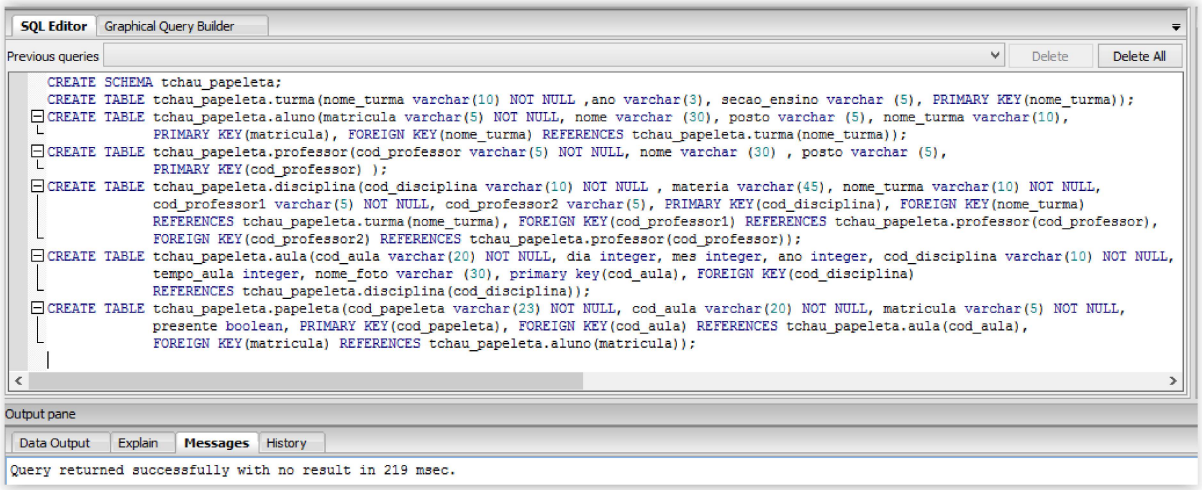

FIG. 6.3: Criação usando o pgAdmin III

exibe a consulta para, a partir dessas duas informações, encontrar-se o *cod_ disciplina*. No exemplo, *nome_ turma* = 'turma1' e *materia* = 'Logica', retornando como resultado o *cod_ disciplina* = 19023.

Além desses, o professor também informará os tempo de aula, sendo a data composta por dia, mês e ano incluída pelo servidor. Com essas informações, ter-se-á parâmetros suficientes para criação da aula. A figura 6.6 demonstra a inserção de uma aula ministrada no dia 05, mês 09 e ano 2018, referente à disciplina Lógica, ministrada para a turma1, (*cod_ disciplina* = '19023'). Preencher-se-á o atributo *nome_foto* conforme apresentado no capítulo 4, referente ao módulo servidor.

Após a inserção da aula, pode-se gerar para cada membro da turma a presença ou falta naquela aula. A figura 6.7 mostra como obter a *matrícula* dos alunos de uma turma. No exemplo, utiliza-se turma1. Por fim, a figura 6.8 demonstra a inserção da presença ou falta de cada aluno dessa turma.

6.4 CONEXÃO COM O BANCO DE DADOS ATRAVÉS DO PYTHON DB API

Depois que a fotografia é enviada para o servidor e as faces são reconhecidas, registrar-se-á essas informações no banco de dados. Para a integração entre o código escrito em Python e o PostgreSQL, fez-se uso do *Psycopg*. Segundo (OFICIAL PSYCOPG, 2018), o *Psycopg* é o adaptador mais popular para essa finalidade. O *"Python DB API 2.0"* possui licença gratuita e segue a política *GNU Lesser General Public License*. A versão escolhida foi a 2.7.5.

Optou-se, então, por criar uma classe denominada Conecta (arquivo conexao.py),

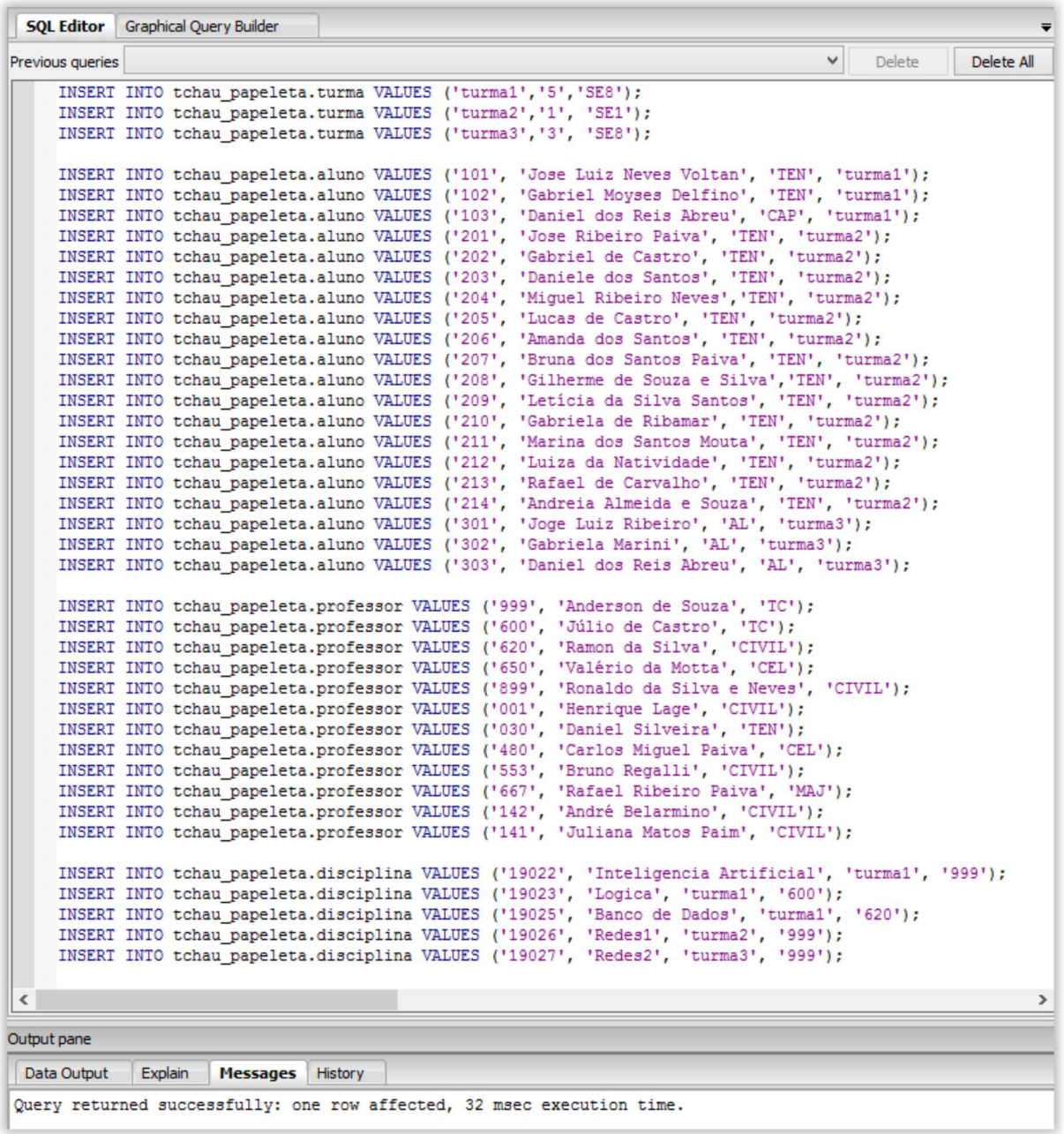

FIG. 6.4: Inserção dos dados no esquema *tchau_papeleta*

que apresenta os métodos necessários para a manipulação dos dados. Em seu construtor, realizar-se-á a conexão com o banco de dados, usando como parâmetros *host, database, usuário* e *senha*. A classe possui ainda, dentre outos, os métodos genéricos para consulta e inserção (cláusulas *select* e *insert*).

Quando uma fotografia de uma turma é recebida, o primeiro passo é identificar se aquela fotografia é a primeira ou não. Caso seja a primeira, deve-se criar a linha aula correspondente àquela fotografia e suas informações, além da papeleta de cada aluno com o atributo presente recebendo *false*. Caso não seja a primeira isso não deve ser feito sob

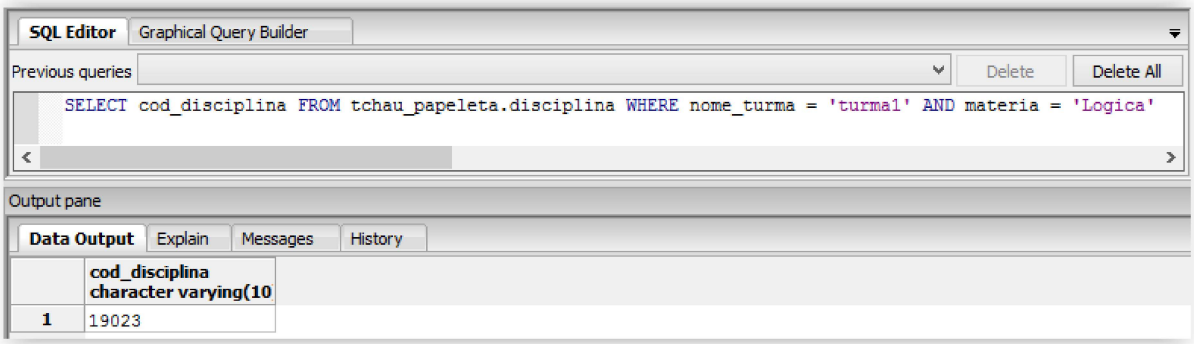

FIG. 6.5: Consulta para obtenção da *cod_ disciplina*

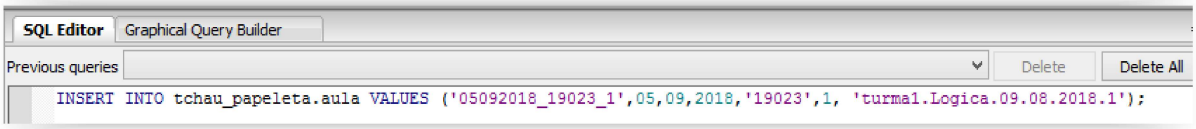

FIG. 6.6: Inserção de uma aula

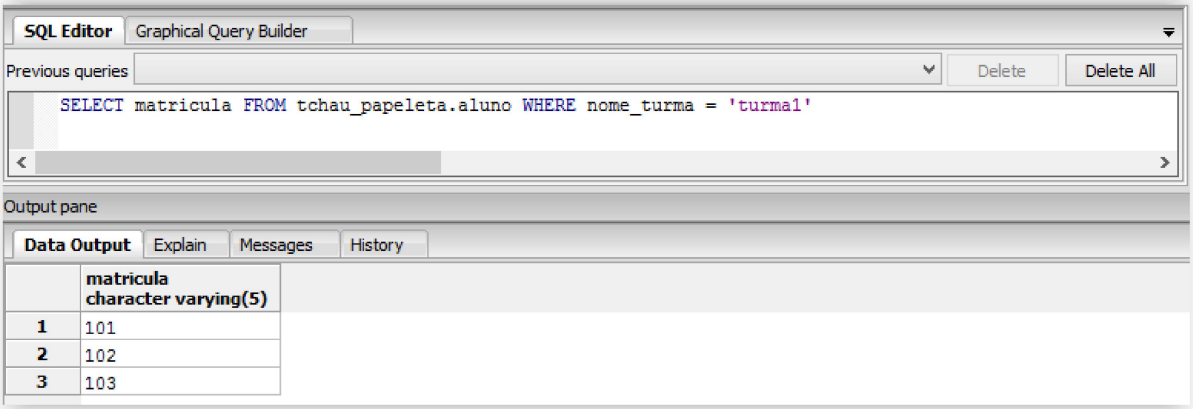

FIG. 6.7: Obtenção da relação de aluno (*matrícula*) de uma turma

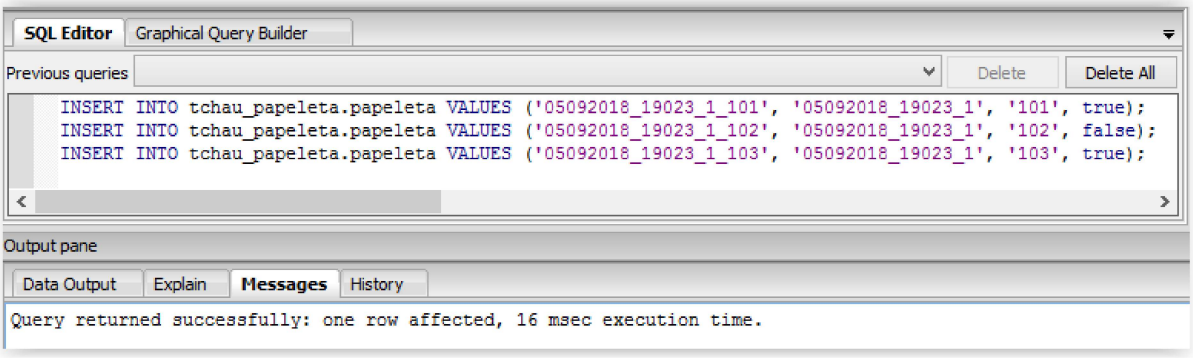

FIG. 6.8: Inserção da falta

pena de apagar a presença de um aluno apurada anteriormente. Para essa verificação, optou-se por, dentro do método *inserir_ aula*, consultar se, de fato, aquela inserção é

necessária ou não. Caso necessário, além da inserção, cria-se a papeleta de cada aluno como mencionado. O retorno do referido método é o *cod_ aula*. O passo seguinte é criar um método nessa classe tal que, recebendo a matrícula do aluno que teve sua face detectada, seja permitido alterar, na linha correspondente da tabela papeleta, o atributo presente de *false* para *true*.

6.5 ACESSO AO BANCO DE DADOS PARA CONSULTA E ATUALIZAÇÃO

Para o atendimento dos requisitos RF02 e RF03, definidos no capítulo de modelagem do sistema, foi implementado um serviço *web*, no qual o cliente consiste em um formulário HTML e o servidor em uma página JSP. No formulário, constam os seguintes campos de preenchimento: dia, mês, ano, tempo de aula, turma e disciplina . Com essa informação é possível fazer uma consulta que retorna a matrícula, o posto e o nome de todos os alunos da turma, bem como a situação de cada um quanto a frequência ao tempo de aula especificado. A estrutura desse serviço *web* caracteriza-se por uma página inicial, ilustrada na figura 6.9, que exibe dois *links* de acesso: *Link* de acesso para a atualização, que permite a consulta e atualização do banco de dados, isso é, os dados são exibidos na forma de uma tabela e o usuário pode alterar a presença ou falta do aluno; e *Link* que permite apenas consulta, voltado para alunos e coordenadores. Ambos os *links* direcionam o usuário para a página que contém o formulário, figura 6.10, o qual, ao ser enviado, direciona para a página JSP, que se encarrega de fazer a manipulação do banco de dados, conforme mostra a figura 6.11. Caso o formulário seja enviado pelo professor, o servidor exibe uma tabela com o resultado da consulta e contendo uma coluna com botões cuja função é modificar a situação do aluno, de presente para ausente ou o inverso. Se for aluno ou coordenador, o servidor retorna retorna a mesma tabela do caso anterior, porém sem as permissões de alteração e exibição dos botões.

Conforme mencionado anteriormente, a transação de atualização é feita pelos botões. Cada botão, ao ser pressionado, atualiza a frequência da linha a qual se refere. Quando a operação é bem sucedida, o usuário é notificado do sucesso por uma mensagem de texto, exibida abaixo da tabela, e também pela mudança de cor do campo presença e do próprio botão acionado. É importante mencionar que durante essas atualizações e notificações a página não precisa ser recarregada, evitando-se, assim, a necessidade de uma nova consulta para obter a tabela a cada atualização feita. Isso é possível por meio de requisições HTTP assíncronas, conhecidas como AJAX, que permitem a troca de dados entre cliente e servidor em segundo plano e a atualização de partes da página sem

FIG. 6.9: Página inicial do serviço(*web*)

FIG. 6.10: Página de preenchimento e envio de formulário html

a necessidade de recarregar a página como um todo, (W3SCHOOLS, 2018). A figura 6.12 mostra um exemplo de notificação.

Perceba que devido à questões temporais limitantes do projeto, não se projetou um sistema de autenticação de usuário. Como oportunidade de melhoria, pode se ter um cadastro com login e senha dos usuários, que permita a separação das operações conforme

FIG. 6.11: Página de exibição do resultado da operação Atualização

FIG. 6.12: Exemplo de atualização

a função do usuário, assim um aluno não poderia atualizar a lista de presença de uma turma, mas um professor sim. Além disso, poderia se dar um tratamento especial aos campos, evitando tentativas de injeção SQL.

7 ANÁLISE CRÍTICA DO SISTEMA

Realizar-se-á nesse capítulo uma análise crítica do sistema, verificando primeiramente se todos os requisitos propostos foram satisfatoriamente atendidos e, posteriormente, analisando oportunidades de melhoria para futuros projetos que tenham como base temas próximos ao do trabalho em tela.

7.1 VERIFICAÇÃO DOS REQUISITOS

Cabe, nesse momento, verificar se os requisitos levantados no início do projeto (Capítulo 3) foram satisfatoriamente atendidos.

7.1.1 REQUISITOS FUNCIONAIS

Tratar-se-á primeiro sobre os requisitos funcionais:

- RF01 - Apurar falta;
- RF02 - Verificar lista de presença;
- RF03 - Editar lista de presença; e
- RF04 - Armazenar fotografia.

Após o envio da fotografia da turma, e recebimento pelo servidor tem-se a apuração das faltas. Vale destacar que a fotografia é selecionada pelo professor a partir do acesso à sua galeria ou obtida através da câmera no momento de aula. A análise da foto permite identificar as faces do alunos, possibilitando atribuir presença para os alunos reconhecidos no processo. O sistema consegue reconhecer várias faces em uma mesma fotografia, e permite o envio de várias fotografias para um mesmo tempo (em operações de envio distintas). Assim sendo, apura-se satisfatoriamente as faltas e cumpre-se o RF01.

A verificação da lista de presença é possível devido ao módulo BD, que busca, no banco de dados, as informações referentes a presença e as exibe no ambiente Web, permitindo consulta. Além disso, o módulo BD também permite, através da interface web, que a presença em uma papeleta de faltas seja editada. Cumpre-se, dessa maneira, os requisitos RF02 e RF01. Cabe mais uma vez destacar a limitação mencionada ao final do capítulo 6. Por questões de limitações temporais do projeto, não foi projetado um sistema de login

e senha. Com o sistema atual o professor pode fazer a edição/atualização da papeleta, com um sistema de login e senha, com um sistema de login e senha, somente o professor poderia fazer essa edição/atualização.

Por fim, quando o módulo servidor terminar a análise da fotografia, a mesma será transferida para a pasta de armazenamento (denominada apuradas). Permite-se, assim, uma auditoria no futuro caso essa seja necessária. O armazenamento da fotografia, então, consolida o cumprimento do RF04.

7.1.2 REQUISITOS NÃO FUNCIONAIS

Tratar-se-á, agora, os requisitos não funcionais:

- RN01 - O Sistema deve apresentar interface com o usuário "Professor" compatível com o sistema *Android*;
- RN02 - O banco de dados, no qual se fará o armazenamento persistente, deve ser PostgreSQL;
- RN03 - As tarefas de captura de foto, reconhecimento facial e armazenamento de registros devem ser feitas em módulos distintos; e
- RN04 - Um professor deve ser capaz de utilizar o sistema após um treinamento de 15 minutos.

Conforme descrito descrito ao longo da descrição da aplicação mobile (Capítulo 5), o aplicativo teve seu desenvolvimento focado no sistema operacional *Android*, satisfatoriamente cumprindo o RN01. De maneira semelhante, desenvolveu-se o armazenamento persistente das informações em banco de dados utilizando PostgreSQL, o que foi descrito ao longo do (Capítulo 6). Como sugestão de melhoria, poderia-se acrescentar ao aplicativo uma tela que redirecionasse para a tela de consulta, dessa forma, após o envio da fotografia, o professor já iria observar quais alunos foram reconhecidos e quais não, e assim optar por enviar uma nova fotografia ou simplesmente atualizar a papeleta.

Quanto a modularização, tem-se que a leitura do trabalho torna claro o fato de que a realização do projeto foi inteiramente feita de maneira modular. Isso permite que trabalhos futuros reaproveitem o projeto, substituindo apenas alguns módulos. Assim sendo, concluí-se que o RN03 também foi atendido.

Por fim, a interface amigável das aplicações *Android* e *Web* tornam a utilização de todo o sistema extremamente intuitiva. Dessa forma, garante-se que a utilização do sistema

não apresentará dificuldades para maior parte do grupo de utilizadores mesmo dentro do contexto da utilização acontecer sem nenhum treinamento prévio. Além disso, espera-se que, em uma eventual necessidade, dúvidas sobre a utilização do sistema por determinado professor possam ser sanadas em um tempo inferior a 15 minutos. Garante-se, assim, o cumprimento do RN04.

7.2 OPORTUNIDADES DE MELHORIA

Entende-se que o módulo de reconhecimento facial, apresentado ao longo do capítulo 4, pode ser modificado de tal forma a melhorar os índices de reconhecimento, ou seja, pode-se buscar apurar as faltas com mais eficiência a partir da fotografia analisada. A utilização da biblioteca *OpenCV*, biblioteca gratuita para uso no meio acadêmico e comercial, foi satisfatória para os objetivos do trabalho em tela, mas pode-se explorar formas de melhorar os algoritmos, sendo o *HaarCascade* ou até mesmo o LBPH. Essa melhoria de algoritmos, que fugiria do escopo do trabalho em tela, pode resultar em melhores índices no número de faces reconhecidas e também em uma maior certeza associada a cada um dos reconhecimentos.

Entende-se também que fotografar a turma pode ser uma tarefa demandadora de tempo. Tal ação poderia ser substituída por um sistema automático de retirada de fotos. Uma câmera, por exemplo, poderia ser colocada na sala de aula em posição estratégica e, ao longo da aula, fotografar-se-ia os alunos presentes, sempre atribuindo presença aqueles que foram reconhecidos. Nesse contexto, não seria mais responsabilidade do professor fotografar a turma e economizaria-se tempo de aula, já que não seria mais necessário utilizar parte do tempo de aula para enviar a fotografia para o servidor. Destaca-se ainda o fato de que a automatização do processo permitiria que várias fotografias fossem tiradas em diferentes momentos da aula, melhorando os índices de reconhecimento já que criar-se-ia diferentes oportunidades de reconhecimento.

Por fim, destaca-se oportunidades de melhoria da aplicação *Web*, cuja interface gráfica pode ser modificada e cuja segurança de acesso pode ser intensificada. As consequências seriam uma experiência de utilização mais agradável e um aumento de segurança, impedindo, por exemplo, que modificações dos status de presença dos alunos fossem realizados por alguém que não o professor que ministrou a respectiva aulas.

8 CONCLUSÃO

Este trabalho teve como objetivo desenvolver um sistema informatizado, que utilizasse o reconhecimento facial em conjunto com a tecnologia de desenvolvimento móvel, capaz de ser uma alternativa para o atual sistema de apuração de faltas dos alunos do IME. Após o estudo conceitual sobre métodos de detecção e reconhecimento facial, foi possível entender melhor as possibilidades oferecidas pela utilização dos diferentes algoritmos pesquisados. Tal entendimento tornou possível a escolha do algoritmo que melhor atenderia ao objetivo do projeto: o LBPH.

O sistema desenvolvido integra uma aplicação *Android*, um banco de dados, um servidor de reconhecimento facial e um serviço *web* para a realização da apuração de faltas dos alunos do IME. Apesar de trabalharem para o mesmo objetivo, esses elementos são independentes entre si, de forma que qualquer um deles pode ser modificado ou substituído sem afetar o sistema como um todo. Dessa forma, a manutenibilidade do sistema fica facilitada permitindo que futuros projetos deem continuidade ao estudo e proponham soluções de melhoria a partir do aprimoramento dos módulos conforme a necessidade ou o surgimento de técnicas novas.

Por fim, o resultado alcançado mostrou-se compatível ao objetivo proposto, apresentando uma interface amigável e de rápido entendimento para os usuários . Além disso, funções como a correção da papeleta, realizada pelo serviço *web*, onde o professor pode alterar manualmente a presença de um aluno, ajudam a aumentar a confiabilidade do sistema, uma vez que o reconhecimento facial pode falhar.

9 REFERÊNCIAS BIBLIOGRÁFICAS

OPEN HANDSET ALLIANCE. Alliance members. Disponível em: <https://www.openhandsetalliance.com/oha_members.html>. Acesso em: 24 de julho de 2018.

ANDROIDXCOMMUNITY. Android NDK vs Android SDK, What is the Difference?. Disponível em: <http://androiddeveloper.galileo.edu/2017/03/08/android-ndk-vs-android-sdk/>. Acesso em: 24 de julho de 2018.

EYAL ARUBAS. Face Detection and Recognition (Theory and Practice). Disponível em: <http://eyalarubas.com/face-detection-and-recognition.html>. Acesso em: 20 jun. de 2018.

BATINI, C.; CERI, S. ; NAVATHE, S. B. **Conceptual Database Design: An Entity-Relationship Approach**. 1. ed. Redwood City: Addison-Wesley, 1991. 470 p.

BEZERRA, E. **Princípios de análise e projeto de sistemas com UML**. 2. ed. Rio de Janeiro: Elsevier, 2007. 369 p.

CARLOS HENRIQUE CÂNDIDO. brModelo - Ferramenta de Ensino: Modelagem de Dados (MER). Disponível em: <http://www.sis4.com/brModelo/>. Acesso em: 6 de setembro de 2018.

RICARDO ROMANO LEONEL DE SOUZA. Android: uma análise prática sobre a plataforma. Disponível em: <https://www.devmedia.com.br/android-uma-analise-pratica-sobre-a-plataforma/37603>. Acesso em: 24 de julho de 2018.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco De Dados**. 6. ed. São Paulo: Pearson Brasil, 2010. 808 p.

ANDROID ENTERPRISE. Android enterprise. Disponível em: <https://www.android.com/enterprise/>. Acesso em: 24 de julho de 2018.

IDC. Smartphone OS market share. Disponível em: <https://www.idc.com/promo/smartphone-market-share/os>. Acesso em: 24 de julho de 2018.

IME. **Normas Internas do Corpo de Alunos**. Rio de Janeiro: IME, 2009.

IME. **Normas internas para controle de frequência de alunos do IME**. Rio de Janeiro: IME, 2009. 14 p.

JAMES F. KUROSE, K. W. R. Camada de aplicação. In: JAMES F. KUROSE, K. W. R. (Org.). **Redes de computadores e a Internet**. São Paulo: Pearson Education, 2009. p. 119–129.

KANTAR. Android vs. iOS - Smartphone OS sales market share evolution. Disponível em: <https://www.kantarworldpanel.com/global/smartphone-os-market-share/intro>. Acesso em: 24 de julho de 2018.

MRUNMAYEE SHIRODKAR, VARUN SINHA, U. J.; NEMADE, B. Automated attendance management system using face recognition. **International Journal of Computer Applications**, v. 2, 2015. Disponível em: <https://research.ijcaonline.org/icwet2015/number2/icwet5026.pdf>. Acesso em: 11 jul de 2018.

DOCUMENTAÇÃO OFICIAL ANDROID - ARQUITETURA DA PLATAFORMA. Arquitetura da plataforma - Android Developers. Disponível em: <https://developer.android.com/guide/platform/?hl=pt-br>. Acesso em: 24 de julho de 2018.

DOCUMENTAÇÃO OFICIAL ANDROID - NDK. Primeiros passos com o NDK. Disponível em: <https://developer.android.com/ndk/guides/>. Acesso em: 24 de julho de 2018.

DOCUMENTAÇÃO OFICIAL ANDROID - SOCKET. Socket - Android Developers. Disponível em: <https://developer.android.com/reference/java/net/Socket>. Acesso em: 24 de julho de 2018.

DOCUMENTAÇÃO OFICIAL PGADMIN III. pgAdmin III 1.22.2 documentation. Disponível em: <https://www.pgadmin.org/docs/pgadmin3/1.22/>. Acesso em: 2 de setembro de 2018.

DOCUMENTAÇÃO OFICIAL POSTGRESQL. PostgreSQL: About. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 2 de setembro de 2018.

DOCUMENTAÇÃO OFICIAL PSYCOPG. Psycopg – PostgreSQL database adapter for Python. Disponível em: <http://initd.org/psycopg/docs/>. Acesso em: 6 de setembro de 2018.

DOCUMENTATION OPENCV. Haar Feature-based Cascade Classifier for Object Detection. Disponível em: <https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html>. Acesso em: 16 abr. de 2018.

DOCUMENTATION OPENCV. OpenCV: cv::face::LBPHFaceRecognizer Class Reference. Disponível em: <https://bit.ly/2LJAlIE>. Acesso em: 16 abr. de 2018.

DOCUMENTATION OPENCV. OpenCV library. Disponível em: <https://opencv.org/>. Acesso em: 16 abr. de 2018.

PAUL VIOLA, M. J. Rapid object detection using a boosted cascade of simple features. In: ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2001., 2001. **Proceedings**... Kauai, HI, USA: IEEE, 2001, p. I511–I518.

PERKOVIC, L. **Introdução à computação usando Python: um foco no desenvolvimento de aplicações**. 1. ed. Rio de Janeiro: LTC, 2016.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software: uma abordagem profissional**. Porto Alegre: AMGH Editora Ltda., 2016. 107–109 p.

PYTHON.ORG. Download Python | Python.org. Disponível em: <https://www.python.org/downloads/>. Acesso em: 05 jul. de 2018.

KEVIN SALTON. Face Recognition: Understanding LBPH Algorithm. Disponível em: <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>. Acesso em: 28 abr. de 2018.

SUDHANARANG, KRITI JAIN, M.; AASHNAARORA. Comparison of face recognition algorithms using opencv for attendance system. **International Journal of Scientific and Research Publications**, v. 8, 2018. Disponível em: <http://www.ijsrp.org/research-paper-0218/ijsrp-p7433.pdf>. Acesso em: 09 jul de 2018.

TIMO AHONEN, ABDENOUR HADID, M. P. Face description with local binary patterns: Application to face recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 28, n. 12, p. 2037–2041, 2006.

YALE UNIVERSITY. Yale Face Database. Disponível em: <http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html>. Acesso em: 26 maio. de 2018.

W3SCHOOLS. AJAX Introduction. Disponível em: <https://www.w3schools.com/js/js_ajax_intro.asp>. Acesso em: 18 setembro de 2018.