UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Marcos Paulo Souto Monteiro

Portaria eletrônica com reconhecimento facial

São Carlos

2020

Marcos Paulo Souto Monteiro

Portaria eletrônica com reconhecimento facial

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Maximiliam Luppe

São Carlos

2020

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Monteiro, Marcos Paulo Souto
M313p Portaria eletrônica com reconhecimento facial / Marcos Paulo Souto Monteiro; orientador Maximiliam Luppe. São Carlos, 2020.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2020.

1. Controle de acesso. 2. Reconhecimento facial. 3. Sistemas embarcados. 4. Detecção facial. 5. Multithreading. 6. Sistemas Web. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Marcos Paulo Souto Monteiro

Título: “Portaria eletrônica com reconhecimento facial”

Trabalho de Conclusão de Curso defendido e aprovado em 08 / 12 / 2020 ,

com NOTA 9,3 (nove , três), pela Comissão Julgadora:

Prof. Dr. Maximiliam Luppe - Orientador - SEL/EESC/USP

Prof. Associado Evandro Luis Linhari Rodrigues - Professor Aposentado - SEL/EESC/USP

Mestre Jovander da Silva Freitas - Professor do IFSP - Instituto Federal de São Paulo, Campus Barretos

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

À minha mãe, Luciana, e ao meu pai, Alan

RESUMO

MONTEIRO, M. P. S. **Portaria eletrônica com reconhecimento facial**. 2020. 63p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

Este projeto se propôs a desenvolver um sistema que controle a entrada de indivíduos a um determinado estabelecimento por meio do reconhecimento facial. O sistema também conta com uma plataforma Web de alteração e visualização de dados e funcionalidades como interação com o usuário. Durante o trabalho, foram realizados uma série de experimentos para definir a melhor implementação de detecção e reconhecimento facial de forma embarcada. Como resultado, foi obtido um tempo de resposta médio entre $0,40s$ e $0,77s$, uma acurácia positiva entre 76% e 100% e uma acurácia negativa acima de 95%. Além disso, as funcionalidades adicionais foram implementadas de forma integrada com o sistema, mas ainda havendo espaço para melhorias, especialmente com relação à confiabilidade.

Palavras-chave: Sistemas Embarcados. Reconhecimento Facial. Detecção Facial. Multithreading. Sistemas Web.

ABSTRACT

MONTEIRO, M. P. S. **Electronic door with facial recognition**. 2020. 63p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

The proposition of this project was to develop a system that controls the entrance of individuals to a certain facility through the recognition of their faces. The system should also be accompanied by a platform for alteration and visualization of data and features such as interaction with the user. Throughout the project, a series of experiments have been conducted to define the best implementation of facial detection and recognition in an embedded form. As a result, it was obtaines an average of response time between $0,40s$ e $0,77s$, a positive accuracy between 76% e 100% and a negative accuracy above 95%. Besides, additional functionalities have been implemented integrated with the system, but still keeping space for improvements, specially when it comes to reliability.

Keywords: Embedded Systems. Facial Recognition. Facial Detection. Multithreading. Web Systems.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Ao longo da revolução de tecnológica em curso já há algumas décadas, o potencial de aplicação de algoritmos e eletrônica para aprimorar processos e tornar as atividades cotidianas mais práticas tem crescido imensamente. Dentre essas aplicações, temos a evolução e automatização de sistemas de controle de acesso. Saindo do uso de chaves convencionais, passou-se para a utilização de cartões ou senhas e, nos últimos anos, tem-se uma expansão do uso da biometria (DIVYA; MATHEW, 2017).

Atualmente, a forma mais popular de biometria é a de varredura de impressão digital, que consegue eliminar problemas de tecnologias anteriores, uma vez que senhas podem ser adivinhadas ou esquecidas e cartões podem ser roubados ou perdidos. Entretanto, essa abordagem traz consigo alguns problemas (ALSAADI, 2012), como a necessidade de um hardware específico para leituras de alta qualidade, a precisão dependente da limpeza do dedo e a falta de praticidade em certos casos, como quando se está usando luvas. Além disso, o sistema com essa característica demanda a presença física de uma pessoa para que possa ser feito o seu cadastro, o que pode ser um problema em cenários em que o acesso da pessoa ao ambiente controlado não é frequente.

Com isso, outros modos de biometria vem crescendo de modo a suprimirem essas complicações, dentre os quais temos o reconhecimento facial. Em termos de hardware, basta uma câmera, a qual é um dispositivo comum, e em termos de usabilidade, basta que o usuário pare em frente à câmera, tornando a experiência ainda mais prática que a que ocorre com impressão digital. Além disso, para cadastrar uma pessoa, basta obter fotos da mesma, o que pode ser feito de forma remota, seja por imagens públicas em redes sociais ou pela própria pessoa enviando uma foto.

1.1 Motivação

Na seção anterior, apresentou-se os benefícios do controle de acesso por reconhecimento facial em termos gerais. De forma aplicada, um sistema com essa característica pode ainda solucionar problemas específicos de um determinado estabelecimento. Nesse sentido, este projeto foi desenvolvido no contexto do escritório de uma empresa de tecnologia. Para dissertar sobre o controle de acesso atual e como poderia ser aprimorado, descreve-se, a seguir, como se dá o acesso de um cliente que vá atender a uma reunião presencial na empresa.

O escritório da empresa em São Paulo é distribuído ao longo de alguns andares de um edifício. Dessa forma, para que um cliente possa atender a uma reunião marcada, ele deve primeiramente passar pela portaria do edifício. Assim, um dos funcionários da

portaria solicita o nome do cliente e utiliza um interfone para entrar em contato com uma funcionária específica do departamento financeiro, a qual será referida como Maria no restante do texto.

Passada a portaria do edifício, o cliente deve se direcionar para o 2º andar, onde se encontram as salas de reunião principais. A entrada no escritório é controlada por um aparelho que faz leitura de impressões digitais; porém, a digital geralmente é cadastrada apenas para funcionários, que são aqueles que tem de ter acesso recorrente. De outra forma, o que acontece é que a Maria abre a porta, de modo que o cliente pode esperar na antessala, ou avisa o gerente responsável pela reunião que o faça.

Nesse cenário, uma das atribuições do sistema proposto é de substituir a Maria nas tarefas de recepcionista para as quais ela é improvisada, já que o seu trabalho está atrelado ao departamento financeiro. Uma outra complicação que pretende-se eliminar é a dependência da portaria do edifício da autorização da Maria, que pode estar ausente da sua sala quando for necessária. Com isso, o cliente acaba tendo que esperar ou que buscar a autorização por meios indiretos, como um telefone celular.

Essa trajetória de acesso descrita para o cliente pode acontecer também para outros indivíduos, como funcionários recém-contratados e prestadores de serviço ocasionais, como um profissional de entregas ou de reparos. Além disso, como já citado, o sistema proposto torna a experiência dos funcionários ainda mais prática.

1.2 Objetivos

Dado o contexto geral de acesso por biometria, o sistema desenvolvido neste projeto teve como principal objetivo realizar o controle de acesso de indivíduos a um estabelecimento a partir de um equipamento posicionado ao lado da porta de entrada, tal qual ocorre na Figura 1.

Figura 1 – Exemplo de equipamento de controle de acesso

Fonte: (ASSISTA INFOCOMM, 2020)

Além disso, dado o cenário específico de aplicação, este projeto propôs-se a atingir os seguintes objetivos adicionais:

- interagir com o usuário por meios audiovisuais de modo a recepcioná-lo ou direcionar a resolução de eventuais complicações;
- notificar os funcionários interessados com relação à chegada de usuários específicos;
- disponibilizar uma plataforma para cadastro e atualização de dados de pessoas autorizadas;
- permitir a consulta ao banco de dados de pessoas cadastradas pela portaria do edifício.

Além disso, por estar relacionado à segurança de um estabelecimento, deveria dispor de um alto grau de confiabilidade, possuindo, dentre outros elementos, mecanismos de auto-teste e de registro de movimentações.

1.3 Organização

Este trabalho é organizado nos seguintes capítulos:

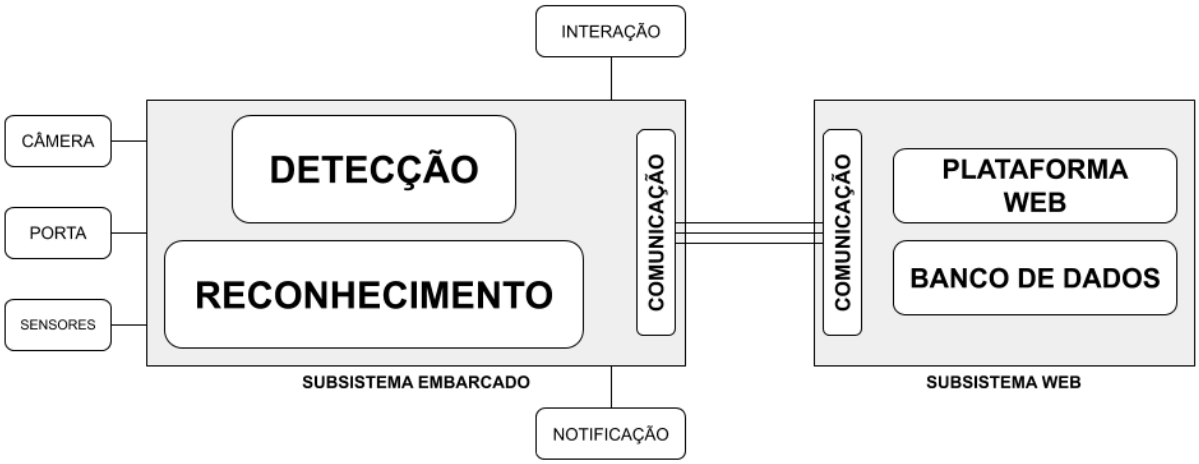
1. Introdução: introduz o contexto da aplicação, descreve o problema e traça os objetivos.
2. Desenvolvimento: define a arquitetura do sistema, analisando, justificando a escolha e determinando os parâmetros de cada um dos componentes, o que é feito por meio de revisão da literatura e da realização de experimentos.
3. Implementação: descreve a implementação do sistema e expõe os resultados obtidos.
4. Conclusão: reflete sobre o trabalho realizado, ponderando acerca do cumprimento dos objetivos e das possibilidades de melhoria para futuros trabalhos.

2 DESENVOLVIMENTO

Os requisitos levantados para o sistema permitiram que se esboçasse um diagrama em alto nível do mesmo, o qual é usado como referência para a forma como esta seção se organiza. Para que possa haver reconhecimento facial, é preciso, primeiramente, que haja captura de imagens por uma câmera e, após isso, a análise de cada imagem por um detector facial a fim de localizar a face a ser identificada. Além disso, junto com os elementos associados à abertura da porta, podem ser necessários sensores adicionais.

Para cadastrar e alterar dados dos usuários, optou-se pelo desenvolvimento de uma plataforma Web; esses dados, por sua vez, devem estar armazenados em um banco de dados. Configuraram-se, portanto, dois subsistemas distintos, os quais tem de se comunicar de alguma maneira. Por fim, eram necessários ainda mecanismos de interação com o usuário e notificação a interessados, que inicialmente são associados ao subsistema embarcado mas que podem depender também de procedimentos no outro subsistema. Com isso, temos o diagrama da Figura 2, de modo que esta seção partirá dos componentes do embarcado, passando para o núcleo do Web e, por fim, tratará dos elementos de integração.

Figura 2 – Diagrama em alto nível do sistema



Fonte: própria

2.1 Hardware Principal

2.1.1 Computador

A fim de projetar o núcleo do subsistema embarcado, iniciou-se pela escolha do computador que será responsável pelo processamento central e controle dos demais elementos. Desejava-se que o hardware do sistema compusesse, ao final, um aparelho

portátil, que pudesse ser instalado, por exemplo, na parede em que se encontra a porta; assim, tornou-se necessário que o computador fosse de pequeno porte. Além disso, ele precisava possuir uma capacidade de processamento considerável, uma vez que esse projeto poderia envolver algoritmos de processamento de imagens e de inteligência artificial, eliminando a possibilidade de uso de um microcontrolador. Com isso, a melhor opção acabou sendo a utilização de um computador de placa única.

Figura 3 – Raspberry Pi 4B

Fonte: (TOM'S HARDWARE, 2020)

Nesse sentido, o sistema utilizou uma placa *Raspberry Pi* 4B (Figura 3), que faz parte de uma de linha de computadores de placa única de baixo custo, utilizada mundialmente em projetos de sistemas embarcados e robótica. Dentre outras características, pode-se conectar por *Wi-Fi* e *Bluetooth*, suporta a execução de sistemas operacionais derivados da família Linux e possui os seguintes conectores: *Ethernet*, portas USB, portas de entrada e saída de uso geral, saída de áudio, saídas de vídeo HDMI, entrada de câmera serial, saída de vídeo serial e alimentação USB-C.

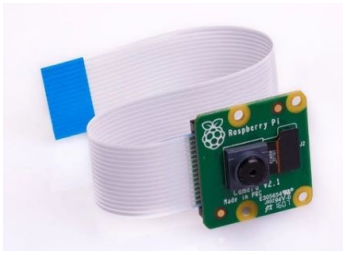
O ambiente de desenvolvimento foi baseado no sistema operacional *Raspbian*, da família Linux, e na linguagem de programação *Python*. Esses dois elementos são os oficialmente apoiados pelo fabricante, de forma que possuem a maior quantidade de recursos para desenvolvimento, característica importante para um projeto que precisa agregar uma gama de componentes de hardware e software distintos.

2.1.2 Câmera

A fabricante da *Raspberry Pi* suporta também uma linha de módulos de câmera chamada *Pi Camera*. A vantagem de se utilizar um desses módulos é que eles são conectados na entrada serial e, com isso, gerenciados pela placa gráfica da *Raspberry*, permitindo uma resolução e taxa de quadros maiores do que *webcams* tradicionais, que se conectam às portas USB. Neste projeto, foi utilizado o modelo V1 (Figura 4), que, além do custo menor, suporta imagens com resoluções de até 2592×1944 e captura vídeos HD em até $30\,FPS$ (JONES, 2016b). Além disso, é pequeno, leve e possui um cabo de comunicação

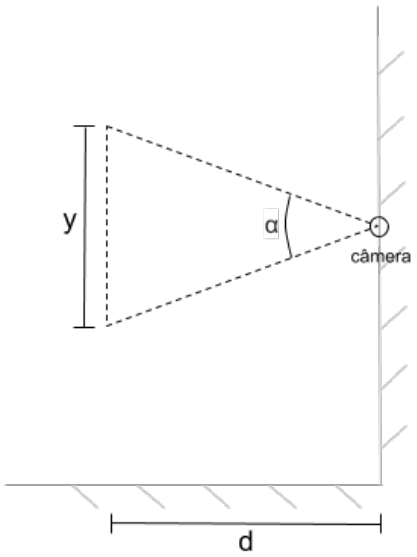
flexível, o que permite ser ajustado a qualquer invólucro que venha a conter a *Raspberry Pi*.

Figura 4 – Pi Camera



Fonte: (TOM'S HARDWARE, 2020)

Figura 5 – Campo de visão vertical da câmera



Fonte: própria

Dado que o equipamento deve se encontrar fixado na parede ao lado da porta, uma especificação que deve ser levada em conta é a de campo de visão, já que deve ser possível capturar imagens de pessoas de diferentes tamanhos. Essa especificação é dada como o ângulo de "abertura" da lente da câmera, retratado na Figura 5 como α. Assim, a uma distância d, a câmera registra imagens em uma faixa $y = 2d\tan(\frac{\alpha}{2})$. Com isso, considerando uma distância de $1,5m$ e dado que o campo de visão vertical da *Pi Camera* é de $41,41°$, temos uma faixa $y = 113cm$, o que é suficiente para atender às estaturas de adultos (OUR WORLD IN DATA, 2019).

Com relação à interface de programação da câmera, tem-se disponível a biblioteca `picamera`, a qual, em sua documentação, apresenta uma série de exemplos descrevendo diversas formas e condições de captura de uma imagem. Dentre elas, temos duas funções que se aproximam ao que era necessário para a aplicação neste projeto: `capture_continuous` e `capture_sequence`. A primeira seria a escolha mais intuitiva, pois permite que se obtenha imagens continuamente, sendo possível, a cada iteração, fazer o processamento da imagem e encerrar o laço de execução caso uma determinada condição seja atingida. No entanto, utilizar esse método faz com que o codificador da câmera seja reinicializado a cada iteração, diminuindo sensivelmente o FPS que pode ser obtido (JONES, 2016a). Assim, o que é proposto na documentação para um melhor desempenho neste tipo de aplicação é executar `capture_sequence` de forma concorrente com o restante do programa, destinando uma *thread* especificamente para a captura de imagens.

2.1.3 Periféricos

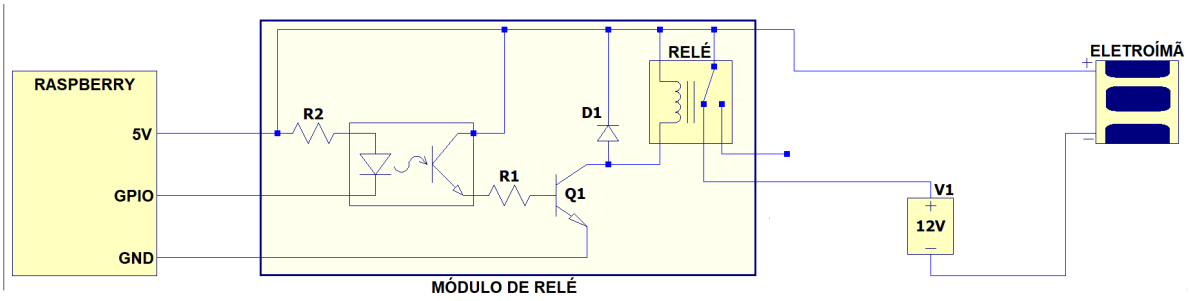
Um outro componente vital para os fins deste projeto é justamente a porta que se quer fazer o controle, ou, mais especificamente, o eletroímã que atua como fechadura dessa porta, cujo exemplo pode ser observado na Figura 6. Para manter a porta fechada, deve-se aplicar uma corrente elétrica no eletroímã (INTELBRAS, 2020), de modo que o campo gerado provoca uma força magnética de atração entre as partes metálicas; para abrir a porta, basta cortar a corrente de alimentação.

Figura 6 – Eletroímã como fechadura de porta

Fonte: (SEGURANÇAJATO, 2018)

Nesse sentido, foi utilizado um módulo relé para chavear a corrente, a qual foi fornecida por uma bateria $12V$, cujo esquemático pode ser observado na Figura 7. O módulo em questão foi alimentado pela *Raspberry Pi* e controlado por um de seus pinos de uso geral, sendo que o estado normal do relé mantém o circuito de alimentação fechado. Assim, para abrir a porta, coloca-se o pino digital de entrada em nível baixo, acionando o opto-acoplador, o qual ativa o transistor, que por sua vez aciona o relé, abrindo o circuito.

Figura 7 – Esquemático de ligações para fazer abertura da porta por meio de relé

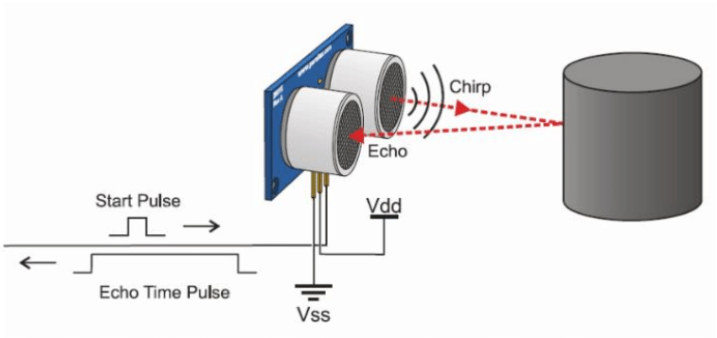


Fonte: própria

Foi adicionado, ainda, um sensor ultrassônico de distância ao projeto, de modo a limitar o alcance do aparelho, evitando que a porta abra para uma pessoa que se encontra

no ambiente em frente à porta mas não pretende entrar. O sensor utilizado foi o HC-SR04 (Figura 8), o qual possui um alcance entre $2cm$ e $4m$. Esse sensor funciona emitindo uma onda de ultrassom e medindo quanto tempo ela demora para ser refletida pelo objeto e voltar ao sensor. Nesse sentido, a função do controlador é enviar um pulso de gatilho ao sensor e receber um pulso de retorno com duração proporcional à distância percorrida, fazendo o cálculo a partir da velocidade do som no ar.

Figura 8 – Sensor de distância HC-SR04



Fonte: (TOM'S HARDWARE, 2020)

Ambos os periféricos descritos trabalham com $5V$, fazendo com que fosse preciso fazer alguns ajustes para utilizá-los com a *Raspberry* sem danificar os pinos digitais, que são $3,3V$. No caso do módulo relé, o fechamento da porta foi feito colocando o pino em estado de alta impedância ao invés de nível alto. Já no caso do sensor, foi utilizado um divisor de tensão com resistores para o sinal de eco, o qual corresponde ao período entre a transmissão e a recepção da onda. Por fim, tratando da programação, esses pinos podem ser comandados a partir da biblioteca `RPi.GPIO`; um detalhe é que para obter o estado de alto impedância é necessário configurar o pino como entrada sem resistores *pull-up/down*.

2.2 Reconhecimento Facial

2.2.1 Aprendizagem de Máquina

Estabelecido o hardware, partiu-se para a definição do principal elemento do núcleo embarcado, o reconhecimento facial. Este foi realizado por meio de algoritmos de aprendizagem de máquina, a qual é uma subárea da inteligência artificial que trabalha com técnicas que permitem que computadores possam "aprender" (PAIVA, 2013). Nesse sentido, modelos de aprendizagem de máquina são treinados a extrair informações de dados automaticamente através de técnicas computacionais e estatísticas. Esse treinamento consiste em fornecer amostras de dados pré-classificadas de modo a "calibrar" os parâmetros internos do modelo, permitindo que o mesmo possa fazer previsões quando novos dados forem apresentados.

Para avaliar o desempenho de um modelo, existe uma gama de métricas utilizadas na literatura, muitas delas se baseando nos estados que podem ser assinalados ao teste de uma amostra, os quais são:

- verdadeiro positivo (VP), em que a amostra pertence a uma determinada classe e é identificada pelo modelo como tal;
- falso negativo (FN), em que a amostra pertence a uma determinada classe mas é identificada como não-pertencente;
- verdadeiro negativo (VN), em que a amostra não pertence a uma determinada classe e é identificada como tal;
- falso positivo (FP), em que a amostra não pertence a uma determinada classe mas é identificada como pertencente.

Neste trabalho, foram utilizadas as métricas sensibilidade, especificidade e acurácia (PAIVA, 2013). Entretanto, considerou-se que os dois primeiros termos não são intuitivos ao leitor no sentido de informar o significado da métrica associada, preferindo-se adotar uma nomenclatura própria, em que sensibilidade é chamada de acurácia positiva e especificidade é chamada de acurácia negativa. Com isso, temos as fórmulas das métricas utilizadas listadas a seguir.

$$\text{acurácia positiva} = \frac{VP}{VP + FN}$$

$$\text{acurácia negativa} = \frac{VN}{VN + FP}$$

$$\text{acurácia} = \frac{VP + VN}{VP + VN + FN + FP}$$

2.2.2 OpenCV

Primeiramente, analisou-se a viabilidade de se empregar a *OpenCV*, a biblioteca de processamento de imagens mais utilizada para projetos embarcados, a qual dispõe de implementações de três algoritmos: *Eigenfaces*, *Fisherfaces* e *LBPH* (OPENCV, 2020b). Os dois primeiros não são adequados para utilização em casos em que se dispõe de apenas uma imagem por pessoa (TAN, 2006). Com isso, foram descartados previamente, uma vez que, neste trabalho, pretende-se que o cadastro no sistema seja prático, não sendo preciso coletar várias imagens distintas do indivíduo a ser cadastrado.

Com relação ao *LBPH*, não foram encontradas informações assertivas sobre a sua precisão nas condições em questão. Nesse sentido, decidiu-se usar um banco de imagens para verificar o seu desempenho. O banco escolhido foi o *Labeled Faces in the Wild* (HUANG, 2007), ou LFW, da Universidade de Massachusetts, que possui 13233 imagens pré-existentes de 5749 pessoas diferentes, personalidades famosas e líderes mundiais, em

geral. A escolha se deu devido ao grande número de pessoas com duas fotos ou mais (1680) e à variedade étnica do banco, como pode ser observado na Figura 9. Esse último requisito se mostrou difícil de atender já que muitos bancos são compostos por imagens capturadas na própria universidade.

Figura 9 – Exemplos do banco de imagens LFW

Fonte: própria

Durante o desenvolvimento do teste, notou-se que várias fotos possuíam mais de uma face, às vezes havendo pessoas com rostos lado-a-lado; entretanto, cada foto estava assinalada à apenas uma pessoa, o que prejudicava o desempenho do algoritmo no teste. Assim, para este projeto, foi construído um subconjunto do LFW fazendo-se o recorte das imagens que possuíam mais de uma face. Foram escolhidas 200 pessoas, de modo que o teste possua um número de faces de referência próximo àquele da aplicação final; a escolha se deu de modo aleatório desde que cada pessoa possuísse pelo menos 2 fotos, totalizando 622 amostras de faces para teste. A partir disso, pôde-se realizar testes com várias combinações de parâmetros do LBPH, sendo que o melhor resultado obtido foi uma acurácia de 19%, atestando que esse algoritmo também não atende às condições de projeto.

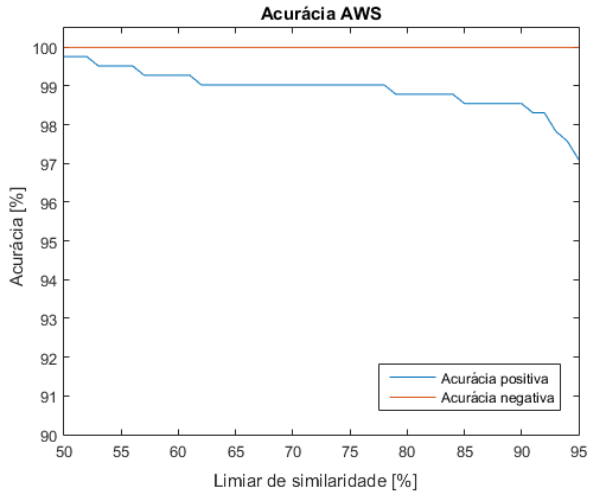
2.2.3 AWS

Além do *OpenCV*, existem outras bibliotecas de código aberto comumente utilizadas para reconhecimento facial como a *OpenFace*, *FaceRecLib* e a *OpenBR*; porém, essas opções apresentaram pouco ou nenhum suporte à instalação no *Raspbian*, impossibilitando a execução na *Raspberry* Pi. A partir disso, a ideia inicial deste projeto era de se utilizar o serviço de reconhecimento facial da *Amazon Web Services*, ou AWS, o qual realiza sua computação na nuvem. Nesse sentido, decidiu-se implementar testes para verificar o seu desempenho em função do banco de imagens construído anteriormente, para os quais foi utilizada a biblioteca `boto3` disponibilizada pela AWS para desenvolvimento em *Python*.

Com relação à acurácia, foram realizados dois testes, também utilizando o subconjunto do LFW construído anteriormente. No primeiro, analisou-se a acurácia positiva,

em que espera-se que o algoritmo seja capaz de identificar a face de um indivíduo que se encontra presente no banco de de referência; assim, as 622 amostras de faces eram analisadas com relação às 200 faces de referência, esperando que a saída seja, de fato, o nome da pessoa presente na amostra. Já no segundo analisou-se a acurácia negativa, no qual espera-se que o algoritmo trate como desconhecido um indivíduo que não se encontra no banco; nesse sentido, cada subconjunto de amostras de uma mesma pessoa era analisado com relação a um conjunto de referência com 199 faces, excluindo a pessoa em análise.

Figura 10 – Resultados dos testes de acurácia com a AWS



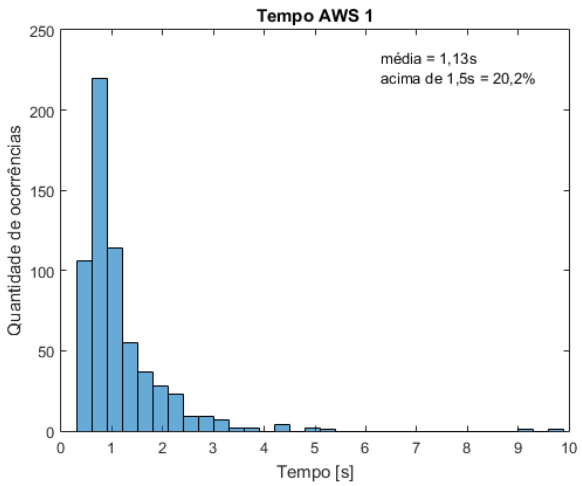
Fonte: própria

Os resultados podem ser observados na Figura 10, nos quais o eixo horizontal corresponde ao nível de similaridade, um parâmetro que representa o quanto se tem certeza de que a face apresentada de fato corresponde à pessoa identificada. Esse parâmetro é fornecido para ser um limiar configurável, de forma que o serviço identifica como "desconhecido" um caso no qual a similaridade fica abaixo do mesmo; o seu valor padrão é definido como 80% na documentação da própria AWS (AMAZON WEB SERVICES, 2020). Pode-se notar que, para o valor padrão, o serviço de reconhecimento apresentou uma acurácia positiva de $98,8\%$ e uma acurácia negativa de 100%, o que certamente é adequado para este projeto.

Durante os testes citados, coletou-se também o tempo de resposta do algoritmo, ou seja, o período decorrido entre o envio da imagem à AWS e o recebimento do resultado, cuja média foi de $1,13s$. Entretanto, utilizando o serviço em outros momentos, observou-se que essa média aparentava ter um resultado distinto. Sendo assim, foi elaborado um novo teste que consistia em analisar 100 imagens diferentes de uma mesma pessoa, a qual foi adicionada às 200 faces do LFW cadastradas na AWS. Para isso, o conjunto *Raspberry*/câmera foi fixado em uma parede a uma altura que seria adequada para a

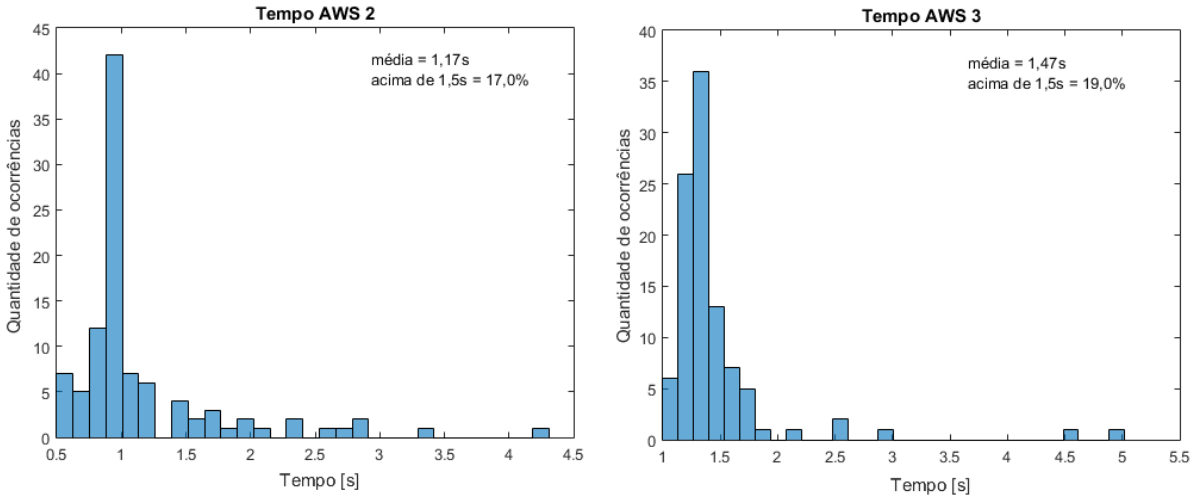
implementação da portaria e capturava imagens frontais da pessoa a $1,5m$ do mesma a partir de um detector de faces; a cada captura, a pessoa mudava um pouco de posição para diferenciação das fotos. A ideia de se utilizar imagens capturadas pela própria câmera surgiu de uma suposição de que estas imagens poderiam ter uma resolução e nitidez sensivelmente diferentes das encontradas no LFW, o que poderia influenciar no tempo de resposta.

Figura 11 – Resultado do teste de tempo com a AWS para imagens do LFW



Fonte: própria

Figura 12 – Resultados dos testes de tempo com a AWS para imagens da câmera



Fonte: própria

O novo teste foi realizado em dois dias distintos e os resultados podem ser observados nos histogramas das Figuras 11 e 12, que contém também aquele relativo aos tempos

coletados com o LFW. O ponto que mais chama a atenção é a variabilidade dos valores registrados, tanto nos tempos individuais, que podem chegar a até $10s$, quanto na média dos testes, que são diferentes entre si. Para explicar esse comportamento, poderia-se levantar a hipótese de que o reconhecimento demora mais para imagens de pior qualidade; entretanto, as fotos capturadas nos testes com a câmera possuíam resolução e nitidez semelhantes entre si e, ainda assim, os tempos variaram entre $0,6s$ e $5s$.

Essa alta dispersão traz dois problemas com relação à experiência do usuário. O mais latente deles é que, em alguns casos, o tempo de resposta poderia ser muito alto, o que talvez poderia ser mitigado com o envio de uma nova requisição ao servidor a partir de um limiar de *timeout*. Um outro ponto importante é que os usuários podem sentir que a implementação do novo equipamento piorou o desempenho da portaria. Atualmente, o acesso é controlado a partir da biometria por impressão digital, cujo tempo de resposta fica em torno de $1s$; em contrapartida, como é exposto nos histogramas, o reconhecimento facial analisado apresenta uma parcela de tempos acima de $1,5s$ entre 16% e 20%. Portanto, a utilização do serviço da AWS carrega consigo uma insegurança com relação à aprovação dos usuários, os quais poderiam ficar insatisfeitos mesmo com os benefícios decorrentes da mudança do tipo de biometria apresentados na Introdução.

2.2.4 Dlib

Durante o desenvolvimento deste projeto, foi encontrada uma biblioteca que não havia sido considerada na pesquisa feita inicialmente, a *Dlib*. Esta é composta por uma série de algoritmos de aprendizagem de máquina, dentre os quais se encontra um de reconhecimento facial feito a partir de uma rede neural do tipo residual (DLIB, 2020b), ou *ResNet*. Efetuando o mesmo teste de tempo descrito anteriormente, obteve-se um tempo de processamento praticamente constante de $0,3s$, sendo 4 vezes mais rápida do que a AWS, em geral.

Essa rede é composta por camadas para identificação de características físicas da mesma forma que o cérebro humano o faz (HE, 2016). Assim, uma camada pode captar desde atributos mais simples, como bordas, até características mais complexas, como texturas. Ao passar pela rede, a imagem, que vai sendo decomposta por suas características mais relevantes até que seja reduzida a um vetor de 128 elementos, o descritor da face. A partir disso, pode-se fazer o reconhecimento ao calcular distância Euclidiana normalizada entre dois descritores de face; caso ela seja menor que um determinado limiar, considera-se que as faces são da mesma pessoa. Este limiar é configurável, sendo definido neste trabalho por meio de experimentos.

Figura 13 – Esquema de processamento da imagem em um descritor de face

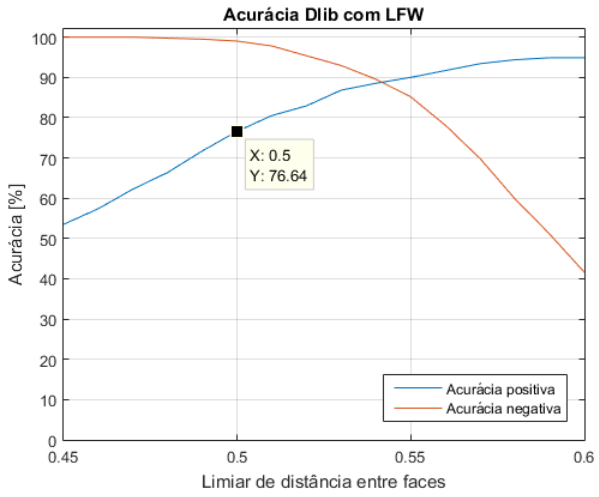
Fonte: própria

Além disso, visando aumentar a performance do mesmo, o autor recomenda que se utilize dois algoritmos como pré-processamento. O primeiro é um preditor de formato de face, que recebe a localização da face na imagem (vinda de um detector facial) e detecta pontos de referência associados a partes do corpo como olhos, boca e queixo (DLIB, 2020a). O segundo, a partir do resultado do primeiro, realiza o alinhamento, reposicionamento e redimensionamento da face para a resolução 150×150, a qual foi utilizada para treinar o modelo de reconhecimento. Com isso, o esquema do processamento de uma imagem em um descritor de face pode ser observado na Figura 13.

Para que a aplicação da *Dlib* pudesse ser viável, seria necessário que também que se obtivesse um bom desempenho com relação à taxa de acerto na identificação. Inicialmente, esse aspecto aparentou ser um problema também resolvido, uma vez que a biblioteca foi atestada oficialmente como tendo uma acurácia de $99,4\%$ para o LFW (UNIVERSITY OF MASSACHUSETTS, 2020). Entretanto, essa avaliação foi feita para o problema de verificação de faces, não podendo ser estendida para o problema de identificação de faces, o qual é tratado neste projeto. Isso porque cada instância de verificação consiste na análise de um par de faces de modo a dizer se são da mesma pessoa ou não; por outro lado, uma instância de identificação compõe-se de uma verificação para cada pessoa cadastrada no banco de referência. Assim, em um acurácia negativa com 200 pessoas cadastradas, por exemplo, caso a face analisada (de uma pessoa que não está no banco) seja tida como semelhante a apenas 1 das faces cadastradas, teríamos uma acurácia de $199/200 = 99,5\%$ para a verificação e de $0/1 = 0\%$ para a identificação.

Nesse sentido, a acurácia da Dlib foi avaliada da mesma forma que ocorreu com a AWS, com o subconjunto do banco LFW. Foram obtidos os resultados exibidos na Figura 14, na qual o eixo horizontal corresponde à distância Euclidiana normalizada entre as faces, que, da mesma forma que o nível de similaridade da AWS, pode ser usada como limiar

Figura 14 – Resultados dos testes de acurácia com a Dlib para o LFW



Fonte: própria

configurável. Nota-se que existe um claro compromisso na escolha do limiar com relação às acurácias analisadas. Examinando esse elemento, temos que uma acurácia positiva baixa gera muitos falso negativos, ou seja, faces de pessoas que estão no banco mas são tidas como desconhecidas; de forma análoga, uma acurácia negativa baixa gera muitos falso positivos, ou seja, faces de pessoas que não estão no banco mas são tidas como conhecidas. A consequência prática do primeiro caso é um tempo de espera maior do usuário para conseguir o acesso, enquanto a do segundo é uma pessoa desconhecida obtendo acesso, o que julga-se um ponto mais crítico. Sendo assim, considera-se que a melhor escolha de limiar seria $0,5$, em que tem-se uma acurácia positiva de $76,6\%$ e uma acurácia negativa de $99,03\%$.

Com os resultados obtidos, tem-se que a acurácia negativa da Dlib se encontra com um valor próximo do observado na AWS. Porém, pode-se argumentar que a acurácia positiva da Dlib faz com que o seu tempo de resposta real seja maior do que apresentado anteriormente, uma vez que seria preciso mais de uma imagem para que o algoritmo atestasse que uma pessoa é conhecida. Mais do que isso, como as imagens capturadas de uma pessoa em frente ao equipamento seriam semelhantes entre si, pode-se supor que, em uma parte dos casos, o algoritmo não conseguiria identificar o usuário de forma alguma, o que também causaria uma grande insatisfação.

No entanto, durante os testes de tempo realizados a partir de capturas da câmera, notou-se que a frequência de falha do reconhecimento aparentava ser menor que aquela revelada pelo experimento. Esse comportamento levantou a suposição de que esse algoritmo tem um desempenho superior para o tipo de imagem gerado pela aplicação com a câmera, em que temos apenas registros frontais das faces, com uma pequena angulação no eixo

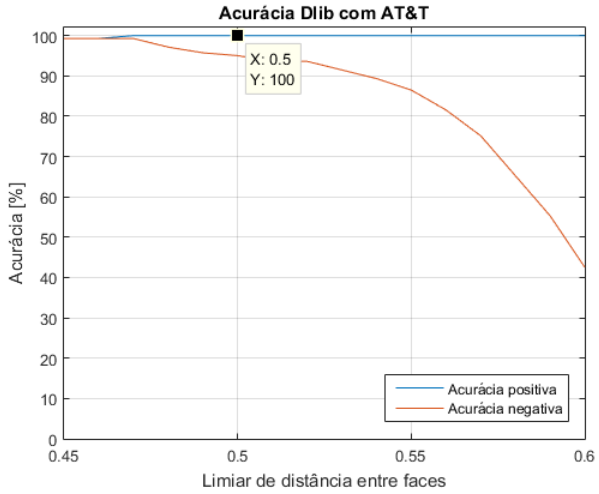
normal ao topo da cabeça proveniente da diferença de altura entre a câmera e a pessoa. No caso das imagens do LFW, temos faces com diferentes orientações e poses, além de obstruções por objetos como óculos escuros e microfones, o que certamente não aconteceria na aplicação deste trabalho.

Figura 15 – Exemplos do banco de imagens AT&T

Fonte: (RUBIOLO; STEGMAYER; MILONE, 2013)

Nesse sentido, buscou-se realizar o mesmo teste utilizando um banco de imagens que tivesse duas ou mais fotos frontais de uma grande quantidade de pessoas; idealmente, o banco também deveria possuir um grande diversidade étnica. Entretanto, não foi encontrada nenhuma opção que atendesse plenamente a todos os critérios; o que julgou-se mais adequado foi o banco de imagens AT&T (GEORGIA INSTITUTE OF TECHNOLOGY, 2020), produzido originalmente pela Universidade de Cambridge, que possui 10 fotos de 40 pessoas distintas. Nem todas as fotos eram frontais, mas, fazendo uma filtragem, pôde-se obter um subconjunto desse banco com 2 a 5 fotos de cada uma das 40 pessoas, algumas das quais podem ser observadas na Figura 15.

Figura 16 – Resultados dos testes de acurácia com a Dlib para o AT&T



Fonte: própria

Os resultados podem ser observados na Figura 16; apesar de não poderem ser tomados como definitivos, dado o número menor de pessoas e a baixa variedade étnica, pode-se observar que o desempenho do algoritmo se torna melhor quando se trata de faces frontais, com uma acurácia positiva de 100% para o limiar escolhido. Portanto, considerando todos os aspectos, o reconhecimento facial da Dlib se mostrou a melhor opção para utilização neste projeto.

2.3 Detecção Facial

2.3.1 Modelo

Em geral, a ação do reconhecimento facial é precedida pela de um algoritmo de detecção facial, o qual recebe as imagens capturadas pela câmera e retorna as caixas delimitadoras contendo a localização das faces. Uma alternativa muito usada para este fim é a utilização dos algoritmos fornecidos pela biblioteca de processamento de imagens *OpenCV*; nesse caso, tem-se disponíveis os classificadores *Haar* e *Local Binary Patterns*, ou LBP (OPENCV, 2020a).

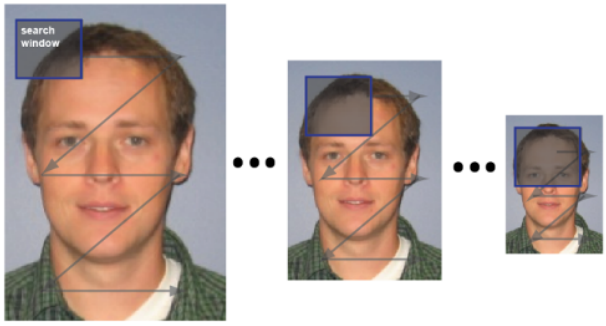
As implementações dos mesmos foram comparadas em (KADIR, 2014), utilizando a mesma metodologia para treinamento e testes. Com os resultados obtidos, o LBP obteve um tempo médio de processamento 140% menor e uma taxa de acerto 4% maior que o *Haar*; além disso, o LBP se mostrou mais robusto à mudanças na iluminação. Ambos os algoritmos apresentaram uma taxa de acerto acima de 80% e não apresentaram distinção significativa de desempenho para diferentes tipos de face, como aquelas com óculos ou com muitos pelos.

O classificador LBP consiste em utilizar uma janela de pixels 3×3 que é convertida em um binário de 8 bits a partir da comparação entre o pixel central com os demais (AHONEN; HADID; M., 2006). Nessa operação, caso o pixel em questão seja mais claro que o central, é assinalado um bit 1; caso contrário, assinala-se um bit 0. Repetindo esse processo ao longo de toda a imagem, tem-se uma nova representação da imagem em 256 níveis referenciadas no pixel central, a qual é utilizada como um descritor de textura a partir de histogramas. Assim, pode-se fazer a detecção comparando as texturas de imagem com a de uma face humana.

Na implementação desse algoritmo no *OpenCV*, temos que a imagem é varrida por uma janela de tamanho determinado buscando faces a partir da comparação de texturas. Para que possa ser possível detectar faces com diferentes dimensões, faz-se um redimensionamento sucessivo da imagem a cada iteração de varredura. Assim, não é a janela de busca que diminui e, sim, a própria imagem; essa dinâmica pode ser observada na Figura 17. Com a análise de várias sub-imagens, temos a possibilidade de detectar uma mesma face várias vezes. Quanto mais vezes uma vizinhança de uma face é marcada com uma detecção, mais certeza se tem de que, de fato, a região marcada contém uma face;

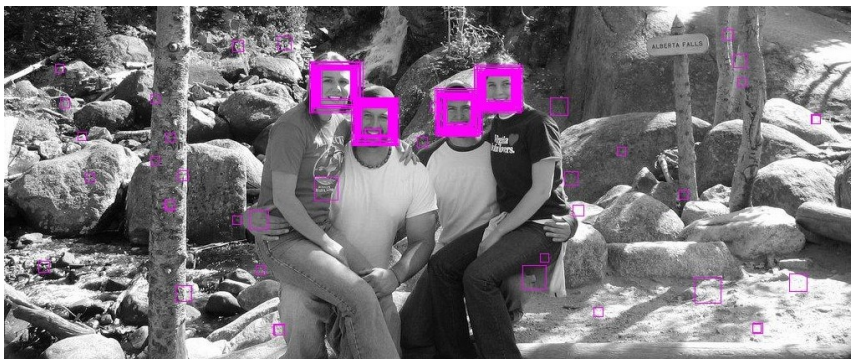
essa característica é exibida na Figura 18.

Figura 17 – Varredura da imagem pela janela ao longo de sucessivos redimensionamentos no algoritmo de detecção facial



Fonte: (MATHWORKS, 2020)

Figura 18 – Marcações de possíveis faces em vizinhanças de uma imagem analisada pela detecção facial



Fonte: (STACKOVERFLOW, 2014)

Com isso, essa implementação é regida por três parâmetros, os quais estão diretamente associados com o desempenho em termos de acurácia e tempo de processamento:

- tamanho mínimo do objeto, o qual define que tamanhos de face serão descartadas pelo algoritmo e, com isso, o tamanho da janela de varredura (OPENCV, 2016); este influencia no tempo de processamento, já que uma janela maior percorre as imagens mais rapidamente;
- fator de escala, pelo qual a imagem é reduzida durante o redimensionamento sucessivo (OPENCV, 2013); este influencia em ambos os aspectos, uma vez que se aumento faz com que o processo de redimensionamento tenha menos etapas, o que gera menos sub-imagens a serem analisadas, melhorando o tempo de processamento mas prejudicando a acurácia;

- número mínimo de vizinhos, que determina o número de vezes que uma vizinhança de *pixels* deve ser marcada para se considerar que ela contém uma face (STACKOVERFLOW, 2014); este último influencia apenas na acurácia, já que sua diminuição aumenta a detecção de positivos, sejam eles verdadeiros ou falsos.

Para alimentar o algoritmo, escolheu-se o modelo desenvolvido por (PUTTEMANS; ERGUN; GOEDEMÉ, 2017), em que foram propostas modificações no modo como o detector facial do *OpenCV* é treinado visando aprimorar a taxa de acerto do modelo padrão dessa biblioteca. Como resultado, obteve-se uma acurácia positiva de 68% para uma acurácia negativa de 90%, enquanto o padrão do *OpenCV* apresentava acurácias de 40% e 40%, respectivamente. Definido o modelo, é preciso determinar quais valores serão utilizados para os parâmetros do algoritmo.

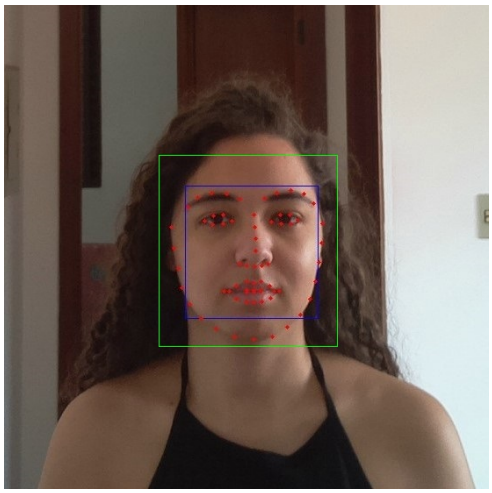
2.3.2 Parâmetros

Com relação ao tamanho mínimo, o objeto que estamos tratando é a face de uma pessoa posicionada em frente à câmera. Assim, é preciso definir qual o tamanho mínimo, em *pixels*, dessa face, o que está relacionado com a distância à câmera, o que já foi definido anteriormente, a resolução da imagem e a altura da pessoa. Para a resolução, o ideal é que a escolha faça com que uma face tenha um tamanho próximo ao que é utilizado como entrada para o modelo de reconhecimento facial da *Dlib* (150×150), fazendo com que haja pouca modificação na informação contida na imagem original por meio do redimensionamento. Nesse sentido, foi preciso levantar as resoluções suportadas pela câmera e verificar qual delas estaria mais próxima dessa condição.

Segundo a documentação da *Pi Camera*, a câmera captura em alguns tamanhos determinados mas pode-se solicitar qualquer um que se queira uma vez que é feito uma conversão pelo próprio periférico da câmera. Assim, temos que, na verdade, a imagem original passar por dois redimensionamentos até a sua análise final: um pela própria câmera e outro pelo algoritmo de reconhecimento facial. Julga-se que o segundo é mais prejudicial, uma vez que recebe uma imagem já modificada, fazendo com que gere uma imagem que não é baseada diretamente na informação contida na imagem original, como é feito pelo primeiro; com isso, mantém-se a condição definida no parágrafo anterior, de modo que a face resultante do processamento da câmera tenha um tamanho próximo a 150×150.

Nesse sentido, foi capturada uma imagem com resolução 1856×1392 de uma pessoa de $1,64m$ de altura posicionada a $1,5m$ da câmera para ser tomada como referência; considera-se que esse tamanho de face estaria próximo ao da média da população, já que as médias de altura feminina e masculina de um adulto no Brasil são de $1,62cm$ e $1,74cm$, respectivamente. Essa imagem é exposta na Figura 19, em que é feita a distinção entre a caixas delimitadoras usadas pelo *OpenCV* (em azul), de tamanho 151×151, e pela *Dlib* (em verde), de tamanho 204×219. Essa última possui dimensão média de 211, de modo

Figura 19 – Face de uma pessoa de $1,64m$ de altura a $1,5m$ da câmera



Fonte: própria

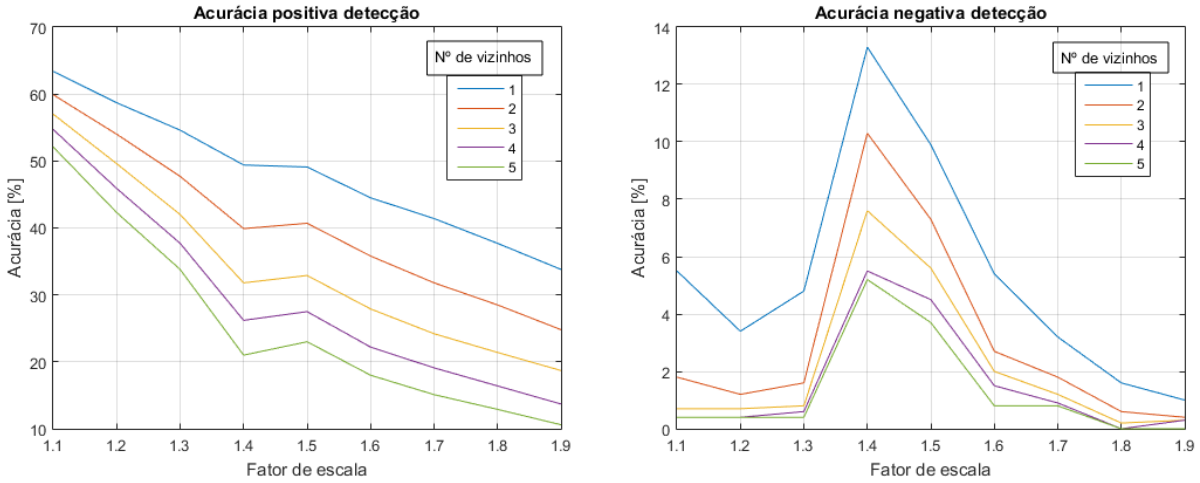
que, para atingir a condição estabelecida, a resolução deve ser $1,41$ vezes menor, passando para 1316×987, o que faz com que a caixa delimitadora do *OpenCV* passe para 107×107. Para definir o tamanho mínimo do objeto, deve-se levar em conta a menor face que seria capturada pelo sistema; como não foi encontrado nenhum estudo que correlacionasse altura com tamanho da face, considerou-se que o tamanho mínimo deveria ser 90% do tamanho da caixa delimitadora de uma pessoa de $1,64m$ de altura, sendo, portanto, 96×96.

Para definir os outros dois parâmetros da detecção, é preciso fazer uma análise conjunta da acurácia positiva, da acurácia negativa e do tempo de processamento para cada combinação; nesse sentido, de forma semelhante ao que foi feito para o reconhecimento, foram feitos testes para avaliar essas grandezas. Os testes para as acurácia se deram com relação ao banco de imagens *Face Detection Dataset and Benchmark* (JAIN; LEARNED-MILLER, 2010), ou FDDB, da Universidade de Massachusetts, que contém 5171 faces em 2845 imagens retiradas do banco *Faces in the Wild* (que deu origem também ao LFW). Esse banco disponibiliza um programa em C++ para gerar a avaliação de desempenho, o que não é algo trivial como no reconhecimento, uma vez que é preciso fazer uma associação entre as caixas delimitadoras reais e detectadas, o que feito a partir da resolução de um problema de emparelhamento em um grafo bipartido.

Infelizmente, não foi possível instalar esse programa, uma vez que ele requeria o *OpenCV* e *drivers* associados em versões específicas, com problemas de compatibilidade sendo reportados pela comunidade acadêmica. Assim, foi necessário implementar uma versão própria desse programa, fazendo a associação das caixas delimitadoras por uma varredura sequencial do grafo, assinalando cada caixa real à caixa detectada mais próxima; essa simplificação faz com que as acurácias obtidas sejam inferiores às reais. A partir disso,

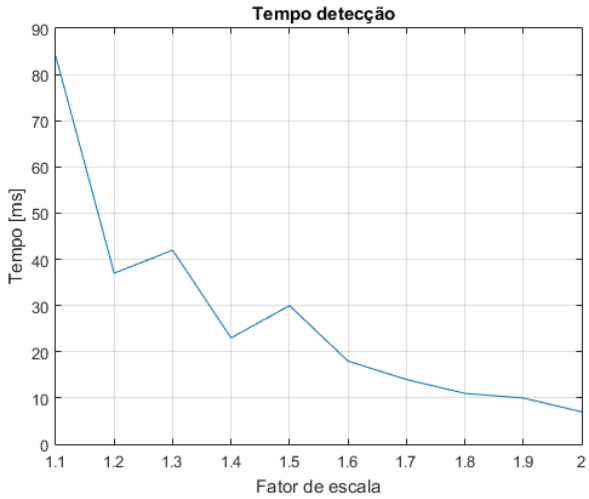
foram avaliadas todas as combinações para um fator de escala entre $1,1$ e $1,9$, com passo de $0,1$, e um número mínimo de vizinhos de 1 a 5. Os resultados podem ser observados na Figura 20.

Figura 20 – Resultado dos testes de acurácia da detecção



Fonte: própria

Figura 21 – Resultado do teste de tempo da detecção



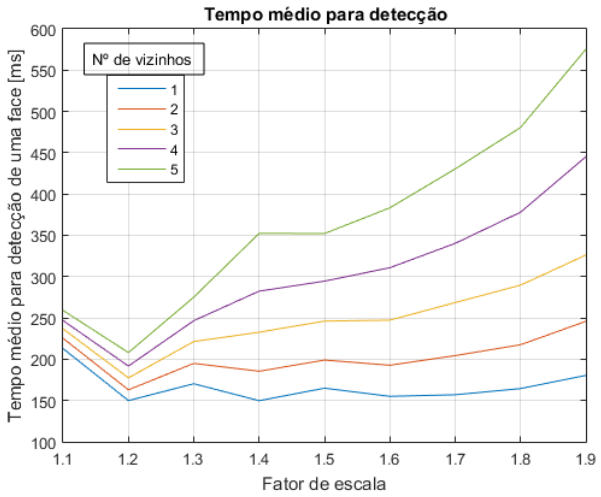
Fonte: própria

Antes de examinar os resultados das acurácias, é preciso também avaliar o tempo de processamento em função dos fatores de escala em questão (o número de vizinhos não influencia no tempo), de forma a fazer um compromisso entre as grandezas avaliadas. Para esse novo teste, o algoritmo de detecção processou 50 imagens para cada um dos fatores de escala, utilizando o tamanho mínimo definido anteriormente, já que este também influencia

no tempo; os resultados são exibidos na Figura 21. Vale destacar que, a partir deste teste, passou-se a recortar as imagens horizontalmente antes da detecção de modo a conterem apenas o terço central, fazendo com que a imagem a ser analisada seja menor. Isso se justifica pelo fato de que o sistema pretende analisar apenas um indivíduo que se encontra à sua frente, não sendo preciso capturar todo o ambiente.

Com isso, se interpretarmos a acurácia positiva como a chance de que, dada uma face em uma imagem, o algoritmo seja capaz de detectá-la, podemos calcular o tempo médio para que uma face seja detectada como sendo $t = \frac{T_P + T_C}{A_+}$, em que T_P é o tempo de processamento, A_+ a acurácia positiva e T_C é o tempo de captura de uma imagem, que por sua vez é $T_C = \frac{1}{FPS}$, sendo que, para a resolução escolhida, $FPS = 19,44$. A partir disso, traça-se o gráfico dessa grandeza na Figura 22; nota-se que temos um ponto de mínimo para o fator de escala $1,2$, o qual se torna o escolhido. Com relação ao número de vizinhos, temos uma queda na acurácia negativa com o mesmo crescendo de 1 para 4; de 4 para 5 não há alteração, de modo que define-se o valor de 4 para esse parâmetro.

Figura 22 – Gráfico do tempo médio para detecção de uma face



Fonte: própria

2.3.3 Filtro de Nitidez

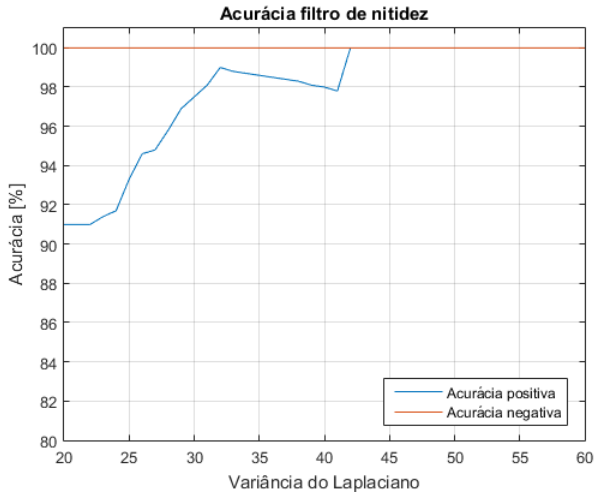
Durante testes com a Dlib, notou-se que os poucos casos em que havia-se um erro no reconhecimento geralmente se davam com imagens em que a face da pessoa está borrada devido à sua movimentação. Assim, visando evitar esse tipo de erro, buscou-se uma forma de mensurar a nitidez das imagens e eliminar aquelas que estivessem borradas a ponto de "enganar" o algoritmo de reconhecimento. Nesse sentido, em (PERTUZ; PUIG; GARCIA, 2013) são listadas uma série de métricas para a grandeza em questão, sendo que optou-se por utilizar a variância do Laplaciano de uma imagem, uma vez que o filtro

Laplaciano é justamente utilizado como detector de bordas, fazendo com que a variância acabe representando o quão marcantes essas bordas são na imagem, ou seja, o quão nítida essa imagem é.

Figura 23 – Exemplos do banco próprio de imagens com diferentes graus de nitidez

Fonte: própria

Figura 24 – Resultado dos testes de acurácia com o filtro de nitidez



Fonte: própria

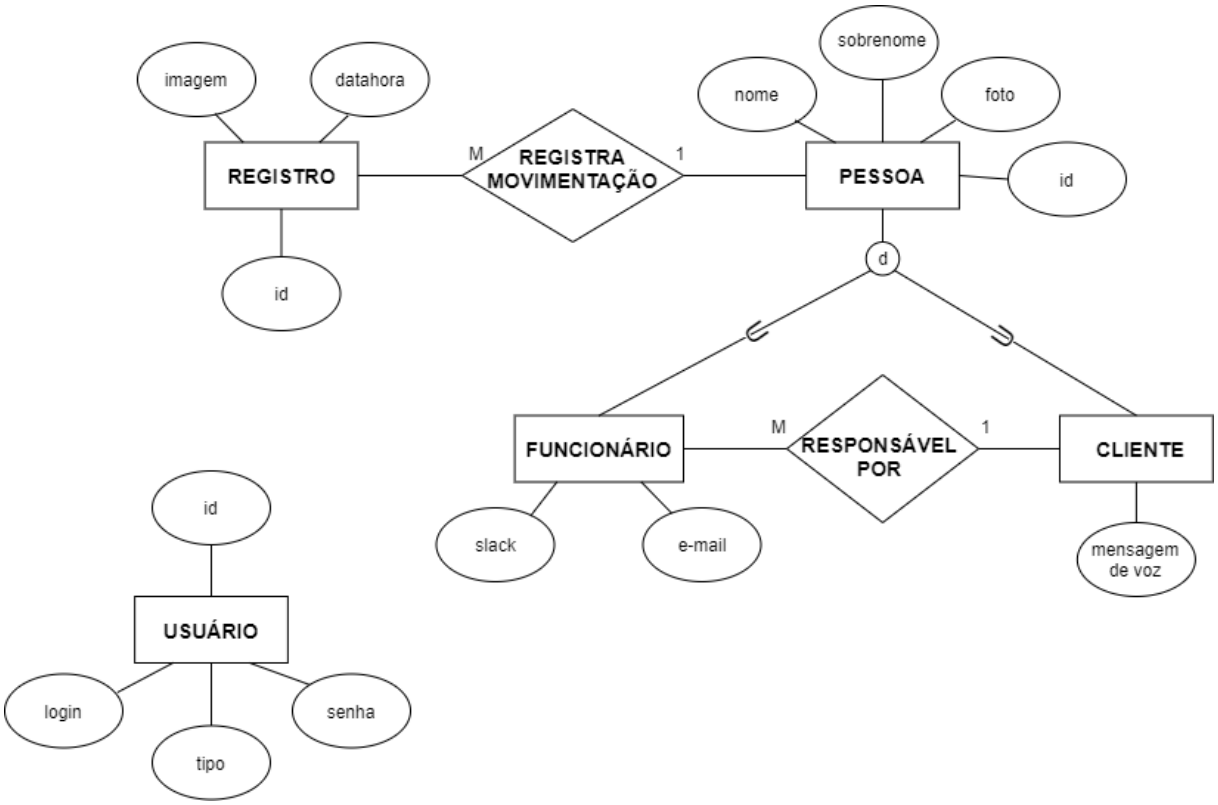
Para verificar a acurácia do reconhecimento em função da nitidez, foi construído um banco de imagens próprio, já que não se encontrou um banco de faces com amostras borradas por movimento. Para isso, foram capturadas 200 imagens de 2 pessoas, as quais ficavam andando para em frente e para trás em frente à câmera fixada na parede, variando a velocidade da caminhada e fazendo alguns movimentos com a cabeça em determinados momentos. Com isso, foi possível obter imagens com variados níveis de nitidez, sendo que alguns exemplares podem ser vistos na Figura 23.

Feito isso, passou-se para o reconhecimento de cada uma das imagens, no qual se utilizou o limiar de $0,5$ (definido anteriormente para a Dlib) e foram registrados, para cada imagem, o grau de nitidez e se o reconhecimento foi correto. Realizando esse procedimento de forma a obter as acurácias positiva e negativa, foram obtidos os resultados exibidos na Figura 24. Nota-se que as acurácias são de 100% a partir do valor 42, de modo que é escolhido 45 como limiar de corte para a implementação deste filtro.

2.4 Núcleo Web

2.4.1 Banco de Dados

Figura 25 – Diagrama entidade-relacionamento do banco de dados



Fonte: própria

O banco de dados foi estabelecido em um computador pessoal por meio do gerenciador de banco de dados relacional MySQL. Um gerenciador implementa um banco no sistema de armazenamento de um computador e realizam funções que tornam a utilização mais prática, como a integração com outros componentes e a possibilidade de acesso pela rede. Como a maioria dos gerenciadores, utiliza a linguagem SQL para fazer acessos ao banco. Além disso, o MySQL é disponibilizado de forma gratuita e apresenta uma interface gráfica que permite operações e visualização dos dados de forma mais prática. O diagrama

entidade-relacionamento que representa o banco deste projeto pode ser observado na Figura 25.

A entidade principal do banco de dados é PESSOA, a qual contém os campos id, nome, sobrenome e foto, sendo id um número de identificação para cada entrada do banco. Para categorizar um indivíduo cadastrado, PESSOA possui as derivações FUNCIONÁRIO e CLIENTE. O primeiro possui os campos adicionais *slack* e *e-mail*, para fins de notificação (*Slack* é a plataforma de mensagens utilizada internamente na empresa), e o segundo possui uma mensagem de voz para sua recepção. Temos ainda um relacionamento entre eles, indicando que cada cliente deve possuir um funcionário responsável, sendo que um funcionário pode ser responsável por qualquer quantidade de clientes (incluindo nenhum).

Além disso, temos a entidade REGISTRO, que registra as tentativas de acesso de indivíduos ao estabelecimento; nesse sentido, essa entidade deve estar relacionada a uma instância de PESSOA, a qual pode aparecer em múltiplos registros. Além disso, o registro deve conter a imagem com a face utilizada para reconhecimento e a data/hora em que a tentativa ocorreu. Por fim, deve-se registrar o login e a senha dos usuários que podem acessar a plataforma Web, além do tipo de permissão que eles possuem, o que é feito pela entidade USUÁRIO.

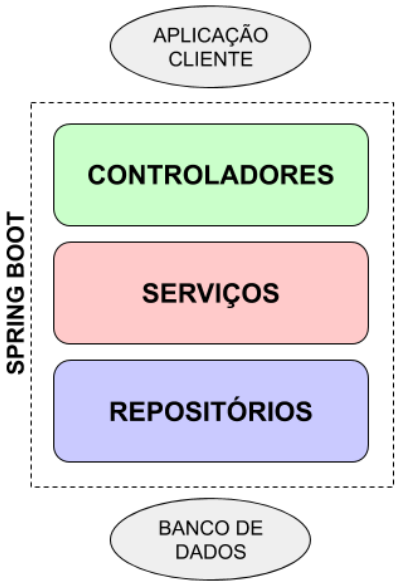
2.4.2 Plataforma Web

Para a plataforma Web, foi utilizado o *framework Spring Boot*, o qual trata de aplicações na linguagem Java. Um *framework* é uma plataforma que abstrai diversos processos necessários para o funcionamento de uma aplicação de modo a facilitar o seu desenvolvimento. Além disso, assim como um biblioteca, um *framework* disponibiliza coleções de módulos padrão que podem ser integrados à aplicação pelo desenvolvedor. Para este projeto, o *Spring Boot* simplifica a implementação de uma aplicação Web abstraindo procedimentos como a implantação de um servidor e a comunicação com o banco de dados.

A estrutura de desenvolvimento no *Spring Boot* se baseia em três elementos principais, os quais podem ser observados na Figura 26. Os controladores lidam com requisições feitas por uma outra aplicação (cliente); assim, este tipo de componente promove um mapeamento de cada caminho possível de ser acessado pela URL da aplicação em seus métodos, os quais devem comunicar com os demais componentes e devolver a resposta adequada. Por exemplo, no caso do cliente ser um navegador, o controlador responde com o arquivo de uma página Web.

Os repositórios são a representação do banco de dados na aplicação, fazendo a abstração das consultas (geralmente na linguagem SQL) ao banco; além disso, eles também são responsáveis por informar possíveis exceções que possam ter ocorrido no banco. Já os serviços processam a lógica necessária para operar os dados fornecidos pelos repositórios para entregá-los aos controladores, aplicando restrições e fazendo as

Figura 26 – Estrutura de componentes do *Spring Boot*



Fonte: própria

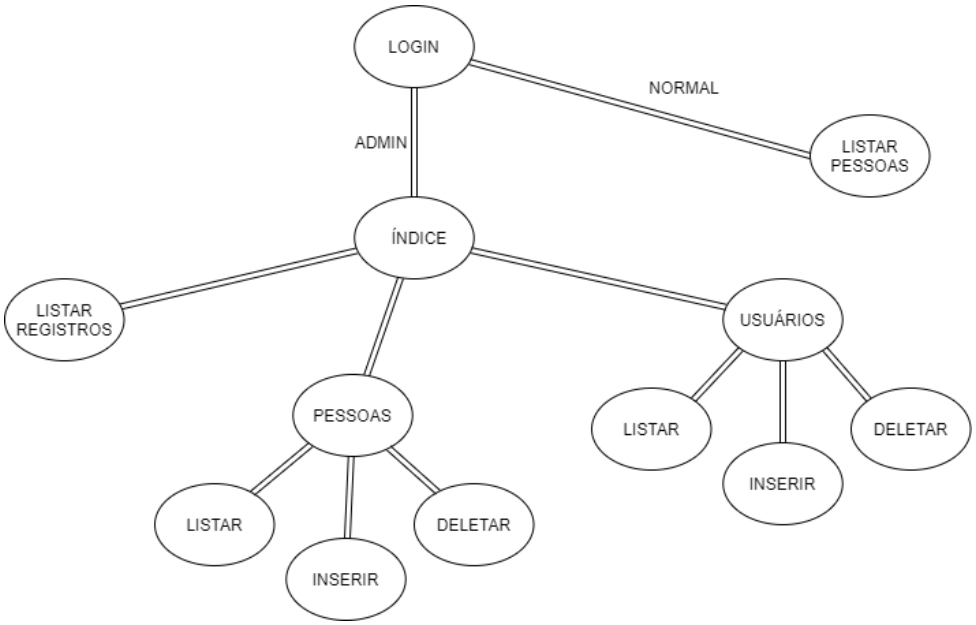
formatações necessárias.

Vale destacar também a presença de outros dois elementos: as entidades e os templates. Os primeiros são uma imagem das entidades do banco de dados, mas sendo representadas por objetos da linguagem Java; assim, quando é feita uma consulta ao banco, o repositório em questão formata os dados obtidos nesses objetos e os retorna à aplicação. Já os templates são modelos para as páginas Web, sendo parametrizados pelo estado atual do banco de dados; esses elementos estão associados que fornece essas funções de parametrização e os processa de modo a renderizar uma página que pode ser enviada ao cliente; no caso deste projeto, é utilizada a ferramenta padrão *Thymeleaf*.

Com relação à estrutura da aplicação Web, tem-se, na Figura 27, o esquema das páginas que buscou-se construir. A primeira é a de login, em que o usuário deve se identificar; caso os seus dados estejam cadastrados no sistema, ele é redirecionado de acordo com o seu tipo de acesso: normal ou administrador. O primeiro caso corresponde ao funcionário da portaria do prédio descrito na Introdução, que deve utilizar o sistema apenas para saber quais pessoas estão cadastradas; sendo assim, esse tipo de acesso é redirecionado para uma página de listar pessoas. Já o outro caso, correspondendo a um funcionário autorizado da empresa, tem-se acesso a todos os dados do sistema; para isso, é feito um redirecionamento para uma página índice, que elenca novas opções de navegação.

A partir do índice, pode-se acessar as pessoas cadastradas para reconhecimento ou os usuários da plataforma Web. Para ambos os casos, tem-se uma página de subíndice que

Figura 27 – Esquema de páginas da plataforma Web



Fonte: própria

permite listar, inserir ou deletar elementos do banco de dados. Vale destacar que, no caso em que se trata de pessoas, as páginas devem fazer uma distinção entre operações com funcionários ou com clientes. Além disso, o índice permite acessar os registros, redirecionando para uma página que faz a listagem dos mesmos; não deve haver disponibilidade para outras operações, uma vez que quem gera os registros é o próprio sistema a partir dos processos que ocorrem na parte embarcada.

2.5 Integração

2.5.1 Comunicação

Dadas as funcionalidades traçadas como objetivo para este projeto, temos que os subsistemas terão de interagir nas seguintes situações:

- na inicialização do sistema, o embarcado faz uma requisição para receber todas as faces cadastradas no momento na plataforma Web;
- a cada aproximação de um cliente com a câmera, o embarcado faz uma requisição para receber a mensagem de voz e o dados para notificação do funcionário responsável por este cliente;
- a cada aproximação de uma pessoa com a câmera, o embarcado faz uma requisição para enviar um registro contendo o resultado do reconhecimento e a imagem capturada para tal;
- quando há alteração nas pessoas cadastradas no banco de dados enquanto a Raspberry

está ligada, a plataforma Web faz uma requisição para enviar os dados dessa alteração para ocorrerem também nas estruturas de dados embarcadas.

Dado que ambos os subsistemas podem se conectar à rede local, optou-se por utilizar o protocolo HTTP como interface de comunicação entre os mesmos. Para isso, é preciso, em ambos os lados, desenvolver meios de enviar requisições HTTP e estabelecer um servidor para receber essas requisições. Nesse sentido, o subsistema embarcado utilizará os módulos `requests` e `flask` do *Python* e o Web apenas reservará um serviço para fazer requisições e um controlador para receber, uma vez que o *Spring Boot* já estabelece um servidor para a aplicação desenvolvida.

2.5.2 Interação

Como foi exposto anteriormente, o sistema demanda meios de comunicação com o usuário, de modo a comunicar sobre a sua autorização para acesso e fazer uma saudação caso necessário. É importante frisar que este trabalho não tratou de interação com pessoas com deficiência, como auditiva ou visual; entretanto, cabe reconhecer a importância da acessibilidade para uma aplicação final do mesmo. Para comunicação visual, foram construídas 4 telas simples: "APROXIME-SE", estado inicial do sistema, "ANALISANDO", que aparece após um indivíduo entrar na região de reconhecimento, "DESCONHECIDO", caso o indivíduo não tenha sido identificado, e uma outra tela contendo nome e sobrenome caso contrário. Apesar da simplicidade, mostrou-se necessário utilizar uma biblioteca própria para desenvolvimento de interfaces gráficas, a `PyQt`, uma vez que bibliotecas simples não permitiam a execução concorrente da interface com o restante da aplicação.

Para a comunicação auditiva, tinha-se basicamente que transmitir uma mensagem de voz quando um cliente for reconhecido, saudando-o e informando que o funcionário responsável foi notificado. Para gerar essa mensagem, foi utilizado o serviço de sintetização de voz *Polly* da AWS, o qual será acionado a cada vez que um cliente for cadastrado, armazenando o áudio no banco de dados. Para reproduzir a mensagem pela *Raspberry*, será utilizado o *OMXPlayer*.

Com relação ao hardware necessário para suportar essas funcionalidades, foi utilizada uma tela de toque LCD de 7 polegadas, a qual recebe dados por um conector HDMI, e um par falante da marca *C3Tech*, o qual recebe sinais por um conector de áudio P2, ambos sendo alimentados por portas USB.

2.5.3 Notificação

Quando um cliente é identificado pela portaria, além de receber a saudação por voz, é preciso também notificar o funcionário responsável pelo mesmo, de modo que possa recepcioná-lo. Nesse caso, a notificação se deu por *e-mail*, enviando uma mensagem de texto simples informando qual cliente chegou, e pelo *Slack*, a plataforma de mensagens

utilizada pela empresa. O desenvolvimento desses meios de comunicação em *Python* se deu pelas bibliotecas `slack` e `smtplib`, respectivamente. O detalhe é que é preciso fazer um cadastro no sistema do *Slack* para autorizar a integração da conta com aplicações externas.

3 IMPLEMENTAÇÃO

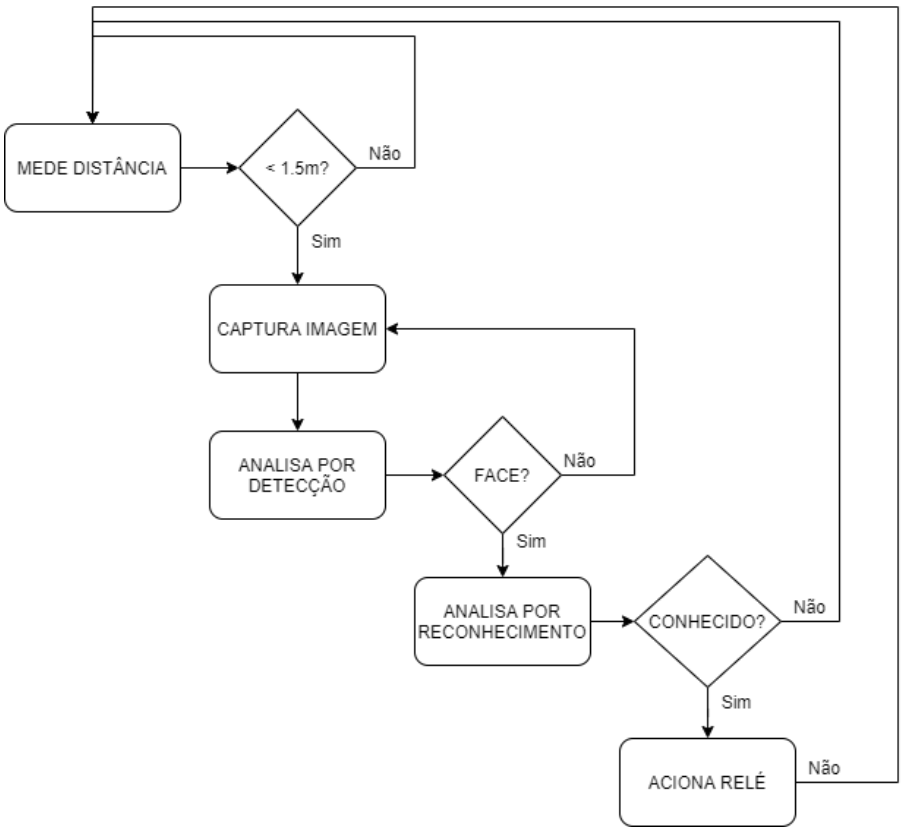
3.1 Embarcado

De modo a agilizar o teste de uma primeira versão, em que poderia-se observar o comportamento do sistema em uma forma mais primitiva, optou-se por fazer a implementação em duas etapas. Na primeira, o programa foi executado de modo totalmente sequencial; avaliado o desempenho do sistema nessa versão, partiu-se para o aprimoramento do mesmo, introduzindo elementos de programação concorrente. É preciso destacar que, como exposto anteriormente, a execução sequencial diminui o valor do FPS, o qual foi utilizado para escolher o fator de escala da detecção; no entanto, mesmo com essa alteração, o melhor desempenho ainda reside no fator $1,2$.

3.1.1 Sequencial

O fluxograma da implementação sequencial pode ser observado na Figura 28.

Figura 28 – Fluxograma da implementação sequencial

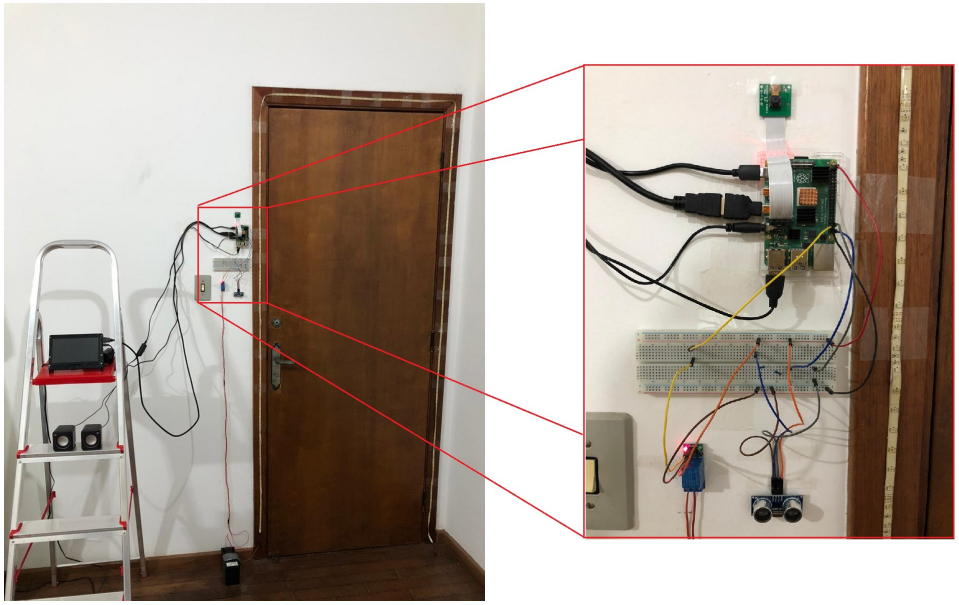


Fonte: própria

Primeiramente, o programa espera que uma pessoa se aproxime, medindo sucessivamente a distância até que seja menor que $1,5m$. Com a aproximação, é executado um laço até que se encontre uma face nas imagens; após a captura pela câmera, cada imagem é recortada horizontalmente para o terço central e convertida para a escala de cinza, já que este é um requisito de entrada dos classificadores do *OpenCV*. Com isso, a imagem é examinada pelo detector facial, gerando um vetor de faces. Caso haja mais de uma face, escolhe-se a face cuja caixa delimitadora tem maior área; a face escolhida passa pelo filtro de nitidez, e caso não esteja borrada, é tida como resultado da busca. Já caso o vetor esteja vazio ou a face escolhida não passou pelo filtro, continua-se com a execução do laço.

A face resultante da busca passa para o escopo do reconhecimento facial, em que os 68 pontos de referência faciais são gerados, a face é alinhada e redimensionada para 150×150 e é gerado o vetor descritor da face. A partir disso, pode-se comparar o descritor desta face com aqueles cadastrados no sistema, calculando a distância Euclidiana normalizada para cada uma delas. Então, escolhe-se a menor distância e verifica se é menor que o limiar de $0,5$; caso seja, conclui-se que o nome associado à essa menor distância é o da pessoa presente em frente à câmera, sendo conhecida; caso contrário, a pessoa presente é desconhecida. Por fim, se a pessoa for conhecida, o relé é acionado para abrir a porta e, antes de continuar a execução do laço, espera-se $2s$ para que a pessoa possa passar pela porta.

Figura 29 – Configuração do hardware

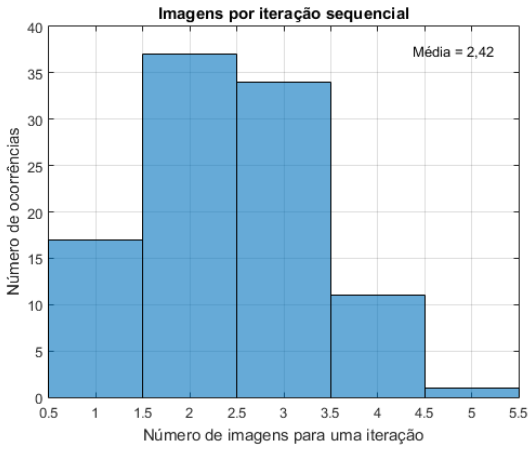


Fonte: própria

A configuração do hardware pode ser observada na Figura 29. Como não havia uma porta com eletroímã disponível para o desenvolvimento deste projeto, o mesmo foi

substituído, de forma equivalente, por uma fita de LEDs $12V$; para isso, bastou inverter a lógica de chaveamento do relé, mantendo a fita desligada para representar a porta fechada.

Figura 30 – Número de imagens utilizadas por iteração no teste sequencial



Fonte: própria

Com o que foi desenvolvido, foi feito um teste para avaliar o número de imagens necessárias para que ocorresse um reconhecimento, uma vez que algumas são descartas durante o fluxo de execução por não ter sido possível detectar a face ou por não passar pelo filtro de nitidez. Neste teste, o laço foi executado 100 vezes para o reconhecimento de uma pessoa de $1,64m$ de altura; os resultados podem ser vistos na Figura 30. Durante os testes com a câmera, notou-se que a primeira captura de imagem sempre durava $100ms$ a mais que as demais, o que ocorre devido à inicialização da função de captura. Somando isso à média de imagens obtidas, de $2,42$, tem-se que o tempo de detecção médio seria de $0,31s$ considerando o FPS máximo da câmera, fazendo com que o tempo de resposta médio do sistema fosse de $0,61s$.

O tempo adicional da primeira imagem pode ser removido desse resultado caso considere-se que a função de captura seria executada sem parar; porém, para isso, seria necessária uma outra instância de execução concorrente para fazer uma "limpeza" do *buffer* de imagens, já que a maioria delas não seria aproveitada pelo sistema. Dados os resultados obtidos e as considerações feitas, pôde-se, então, remodelar o sistema, agora utilizando elementos de programação concorrente.

3.1.2 Concorrente

Nessa nova versão, foi proposto fazer a captura de imagens e detecção de faces de forma contínua em um programa separado. A ideia é que, além de eliminar o problema de inicialização da função de captura, o sistema seja capaz de processar imagens do usuário antes que ele entre na região de reconhecimento, a $1,5m$ de distância da câmera. Com

isso, pode-se obter uma melhora adicional no tempo de resposta com relação à percepção do usuário, já que este só nota a ação do sistema quando a sua interface gráfica (a ser implementada) muda de estado, o que só ocorrerá dentro da região definida. Vale destacar também que essa mudança não resulta em abertura da porta para pessoas distantes, já que, da mesma maneira, o relé só poderá ser acionado com a aproximação de um indivíduo.

Antes de tratar desse programa separado, deve-se definir uma nova resolução para a imagem capturada, o que está relacionado com a ampliação do alcance da detecção de faces. Nesse sentido, capturou-se novamente uma imagem 1856×1392 de uma pessoa de $1,64m$, dessa vez a $2m$ da câmera, o que resultou em uma caixa delimitadora de 150×152 para a *Dlib*, o que é muito próximo do tamanho requerido pelo modelo de reconhecimento. Com isso, dado que o FPS para essa resolução é de $9,38$ e que uma pessoa caminha com velocidade média de $1,4m/s$ (CAREY, 2005), tem-se que seria possível analisar uma média de $3,34$ imagens antes que o indivíduo adentre a região de reconhecimento, o que é próximo da média de imagens utilizadas pelo sistema registrada no último teste.

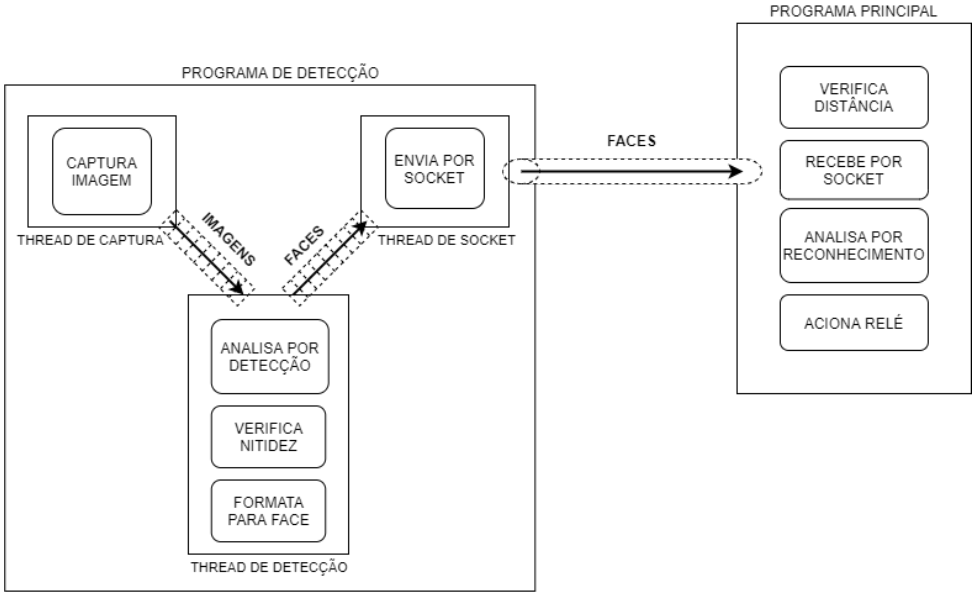
Foram capturadas também imagens a distâncias maiores; porém, julga-se que não é possível assumir que aumentar o alcance em maiores proporções resultaria em uma melhora no desempenho do sistema, uma vez que, para isso, estamos assumindo que a pessoa se deslocará frontalmente ao longo de toda essa distância, o que é razoável para $2m$ de distância mas que torna-se cada vez mais improvável para distâncias maiores. Portanto, definiu-se 1856×1392 como a nova resolução da captura; um detalhe é que essa mudança não altera o parâmetro de tamanho mínimo do objeto, já que o tamanho da face em *pixels* se mantém o mesmo.

Como foi exposto anteriormente, a documentação da *Pi Camera* sugere que, para obter um desempenho melhor em termos de FPS, deve-se fazer a captura utilizando a função `capture_sequence` sendo executada em uma *thread* separada; para isso, utilizou-se o módulo `threading`. Fazendo isso, precisamos também de uma estrutura de dados que faça o armazenamento de imagens de modo seguro, não permitindo interferência entre as *threads* para acesso aos dados; para isso, é utilizada uma fila do módulo `queue`, que é implementada de modo a bloquear *threads* competindo por acesso.

Além disso, é necessário haver um canal de comunicação entre o programa da detecção e o principal; isso é feito por meio de soquetes de comunicação entre processos. Nessa comunicação, o processo principal faz uma requisição de face ao processo de detecção quando uma pessoa se aproxima, o qual deve estar preparado para ser solicitado a qualquer momento. Assim, é introduzida uma nova *thread* no programa de detecção que suportará um servidor de soquete; da mesma maneira, utiliza-se uma outra fila para estabelecer o fluxo de imagens entre a *thread* principal e o servidor soquete. Com isso, temos o sistema esquematizado na Figura 31, o qual será descrito em mais detalhes a seguir.

Primeiramente, tem-se a thread de captura, que fica continuamente executando

Figura 31 – Arquitetura concorrente do subsistema embarcado



Fonte: própria

a função `capture_sequence`, a qual é reiniciada a cada vez que termina a captura de um número de imagens fixo; este número foi escolhido arbitrariamente como 100. A cada captura, é chamada uma função auxiliar para transferir os bytes do buffer de captura para a fila de imagens e limpar esse buffer. Na thread de detecção, cada imagem que é retirada da fila passa pelo mesmo procedimento da detecção descrito na implementação sequencial, resultando, ao fim, em uma caixa delimitadora que representa a face.

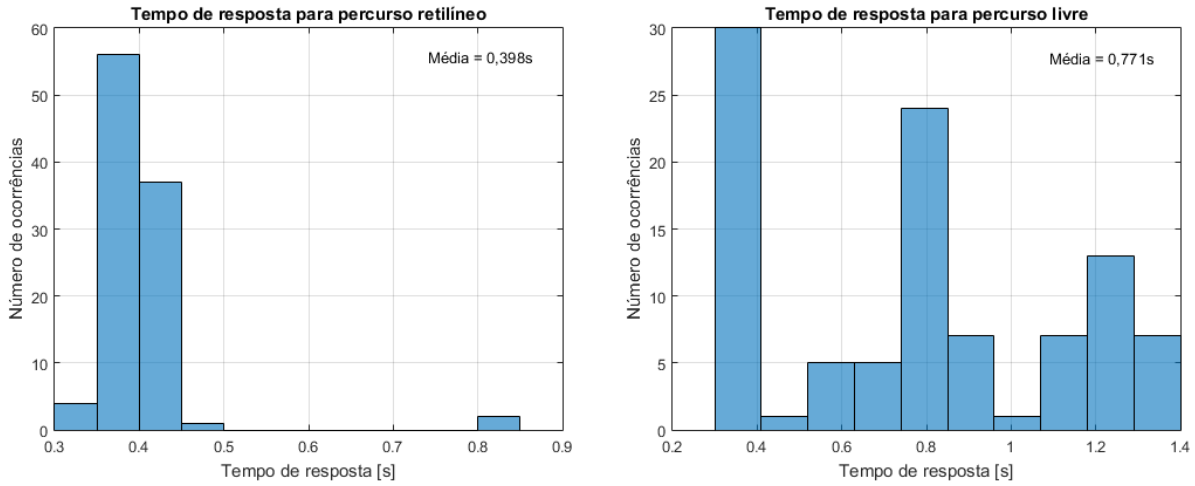
Para alimentar a segunda fila, esta thread deve gerar uma estrutura de face, composta pela caixa delimitadora, o instante de tempo em que ela foi gerada e um recorte da imagem original, contendo apenas a cabeça da pessoa na imagem; esse recorte visa diminuir o número de dados para a transmissão, tornando-a mais rápida. Já o registro de tempo serve para garantir que a face coletada para reconhecimento é recente, de modo que aquelas que não foram geradas no último segundo são descartadas. Além disso, a fila de faces é implementada com apenas 1 posição, garantindo que tem-se sempre o registro mais recente. Caso, não houvesse essa preocupação, seria possível que o reconhecimento da iteração atual se desse com base nas faces utilizadas na anterior, “enganando” o sistema.

No processo principal, são executados os demais procedimentos primários do sistema da mesma maneira que foi feito na implementação anterior. A diferença é que, após o sensor de distância constatar uma aproximação, faz-se uma requisição soquete ao processo de detecção. Este, por sua vez, recebe a sinalização pela *thread* do servidor soquete, que prontamente retira uma face da fila, realiza a sua serialização, ou seja, codifica-a como bytes para transmissão, e faz o envio. Do outro lado do canal, os bytes recebidos são

decodificados e seguem para os procedimentos usuais de reconhecimento.

A partir da implementação dessa versão do sistema, realizou-se o mesmo teste feito com a versão anterior, novamente com uma pessoa de $1,64m$ sendo analisada 100 vezes, agora registrando o tempo de resposta, ou seja, o tempo decorrido entre a aproximação e a resposta do reconhecimento. Neste caso, além do usual percurso retilíneo, ou seja, em que a pessoa caminha em linha reta em direção ao protótipo, fez-se também um teste com um percurso livre, em que a pessoa era posicionada no lado oposto do aposento e era solicitada a caminhar até o protótipo de forma natural, como ela faria em um dia comum.

Figura 32 – Tempo de resposta com a implementação concorrente para um percurso retilíneo e outro livre



Fonte: própria

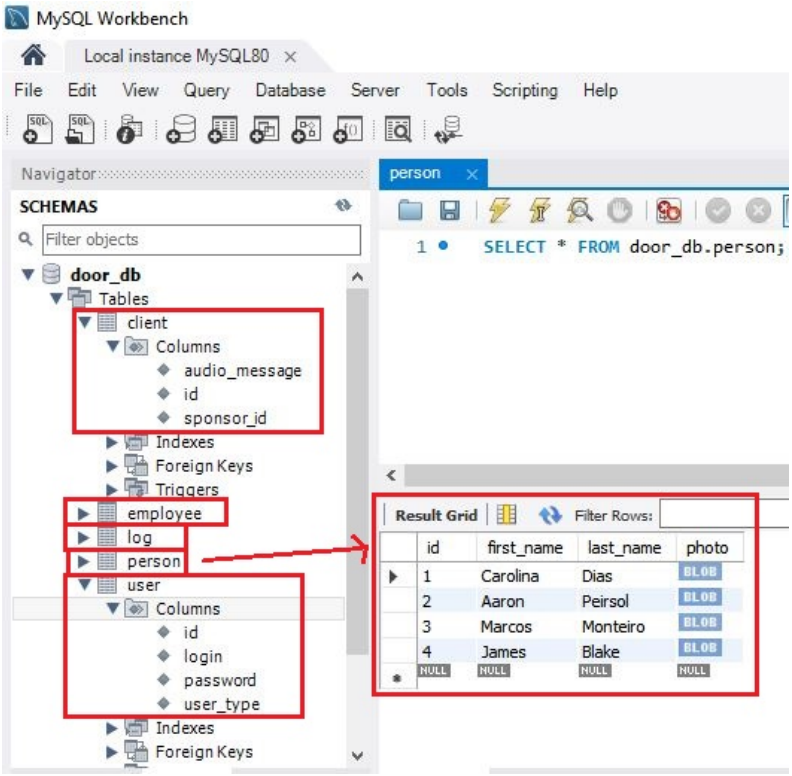
Os resultados são exibidos na Figura 32; para o percurso retilíneo, em que o tempo médio de resposta foi de $0,398s$; nota-se que a dispersão dos valores foi muito baixa, com a grande maioria ficando entre $0,3s$ e $0,5s$. A exceção fica por conta de dois tempos em torno de $0,8s$, que correspondem justamente a dois casos em que o reconhecimento falhou na sua primeira execução. Já para o percurso livre, o tempo de resposta cresce para $0,771s$, o que era esperado, já que a pessoa entra na área de análise com a face em semi-perfil, diminuindo a acurácia tanto da detecção quanto do reconhecimento. Entretanto, mesmo nessas condições, o sistema tem desempenho superior ao do equipamento atual (com biometria por impressão digital) em mais de 70% dos casos e, nos demais, não ultrapassa o tempo de $1,4s$.

3.2 Web

Dado o diagrama entidade-relacionamento construído no capítulo anterior, pôde-se implementar o banco de dados no MySQL. Na Figura 33, pode-se visualizar uma captura

do *MySQL Workbench*, em que são destacadas as tabelas correspondentes a cada entidade e, além disso, mostra-se o conteúdo de uma delas.

Figura 33 – Banco de dados implementado no MySQL, com destaque para as tabelas associadas às entidades

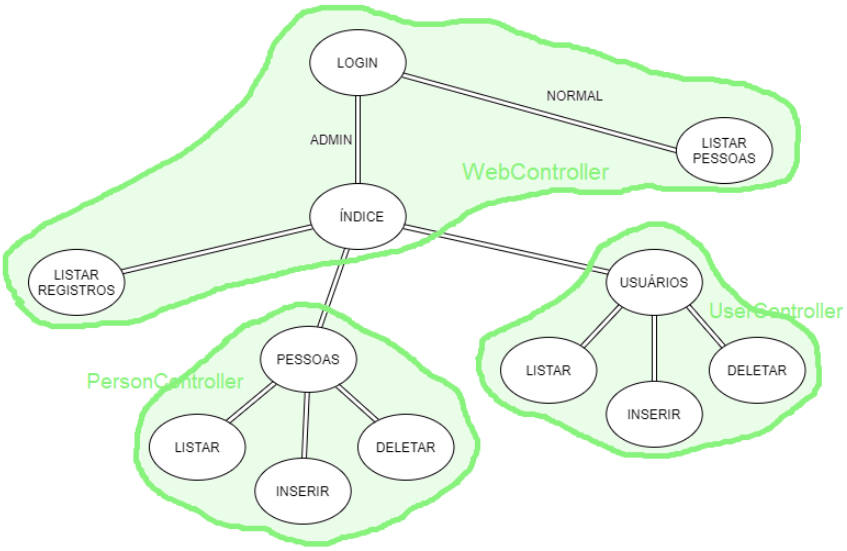


Fonte: própria

Dado o esquema de páginas construído no capítulo anterior, pode-se mapeá-las aos controladores que deverão tratar das requisições de acesso às mesmas. Neste caso, são utilizados três controladores: `WebController`, `PersonController` e `UserController`. Os dois últimos gerenciam os subíndices e as operações relativas a pessoas e usuários, respectivamente, enquanto o primeiro trata das demais páginas. Para completar a arquitetura Spring Boot, temos também os serviços `PersonService`, `UserService` e `LogService` e os repositórios `PersonRepository`, `UserRepository` e `LogRepository`, correspondendo aos dados de pessoas, usuários e registros, respectivamente. O mapeamento das páginas em função dos controladores é exibido na Figura 34.

Tratando primeiramente das páginas relativas a usuários, temos o esquema de ligações exposto na Figura 35. A exibição das páginas é mapeada pelos métodos `insertPage`, `listPage` e `deletePage`. Além destes, o controlador apresenta os métodos `insert` e `delete` para efetuar uma ação de inserir ou deletar caso o usuário interaja nesse sentido. Estes, por sua vez, acionam `saveCommand` e `remove` no serviço e `save` e `removeById` no repositório, respectivamente, para concretizar suas alterações no banco de dados. Tem-se

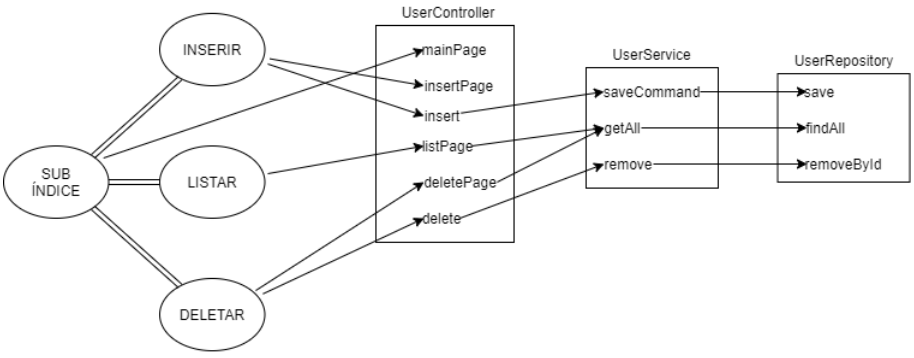
Figura 34 – Mapeamento das páginas Web para os controladores



Fonte: própria

ainda os métodos `getAll` e `findAll`, que retornam todos os elementos do tipo em questão, o que obviamente deve estar associado à pagina Listar mas também à Deletar, já que o usuário precisa saber quais são os elementos disponíveis para operação. Um detalhe é que, quando um usuário faz um inserção, a página correspondente gera um objeto que comporta os dados desta operação, chamado de comando, o qual deve ser convertido para um objeto do banco de dados pelo serviço.

Figura 35 – Relacionamento entre componentes Web para as páginas de usuário

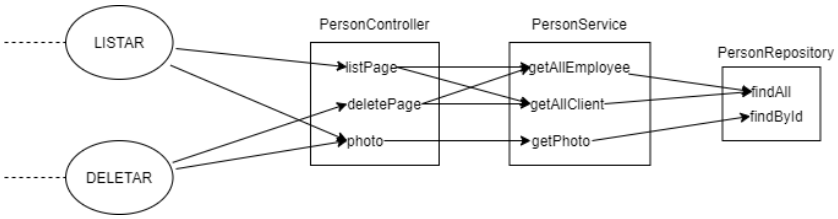


Fonte: própria

Para as operações relativas a pessoas, o esquema é muito semelhante ao de usuários, mas com algumas alterações, como pode ser visto na Figura 36. As páginas de pessoas fazem diferenciação entre funcionários e clientes; assim, elas chamam os métodos `getAllEmployee` e `getAllClient` para obter esses dois subconjuntos separadamente. Além disso, essas

páginas também exibem as fotos das pessoas, o que requer um método específico para fazer a formatação e permitir a renderização da imagem, o que ocorre com `photo`; este, por sua vez, deve ser chamado para cada foto, de modo que as obtém por meio de `getPhoto` e `findById`, que fazem a seleção pelo identificador da pessoa no banco.

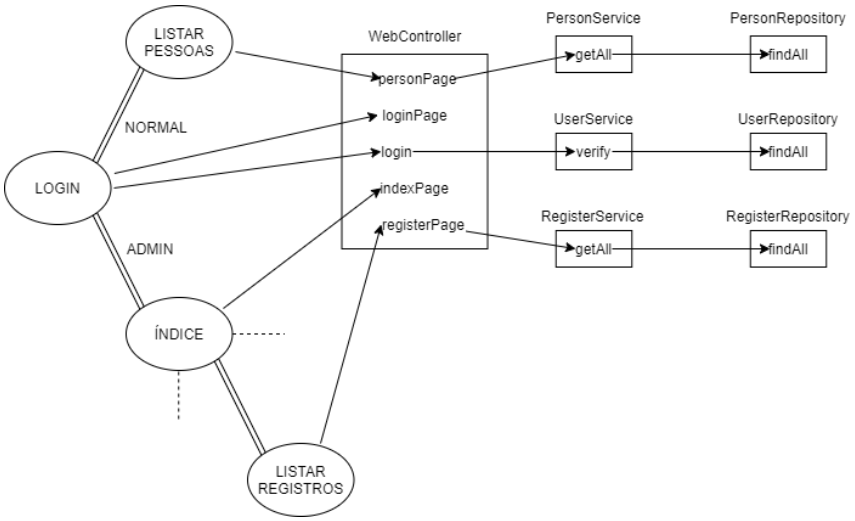
Figura 36 – Relacionamento entre componentes Web para as páginas de pessoas



Fonte: própria

Para as demais páginas, temos o diagrama da Figura 37. Login é exibida por `loginPage` e solicita verificação de usuário e senha inseridos com `login`; este, por sua vez, aciona `verify`, que pega todos os usuários presentes no banco de dados e verifica se algum deles corresponde ao que foi fornecido; caso haja uma correspondência, retorna-se o tipo do usuário em questão. Além disso, temos a exibição de Índice por `indexPage` e da listagem de pessoas e registros por `personPage` e `registerPage`, respectivamente, os quais chamam `getAll` e `findAll` para coletar todos os dados. Vale destacar que, nesse caso, não é preciso distinção entre pessoas, já que basta a sua foto e o seu nome para um usuário normal.

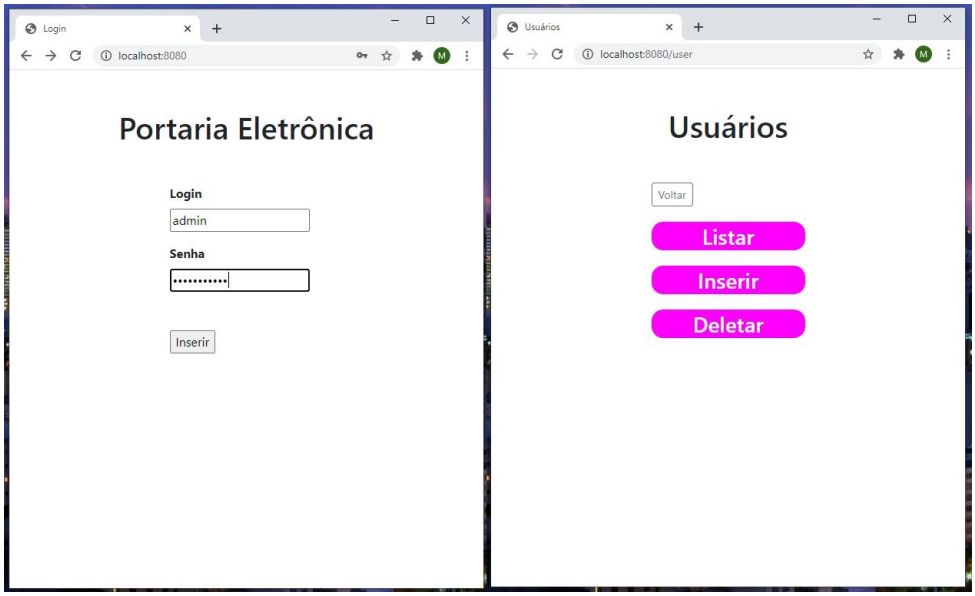
Figura 37 – Relacionamento entre componentes Web para as demais páginas



Fonte: própria

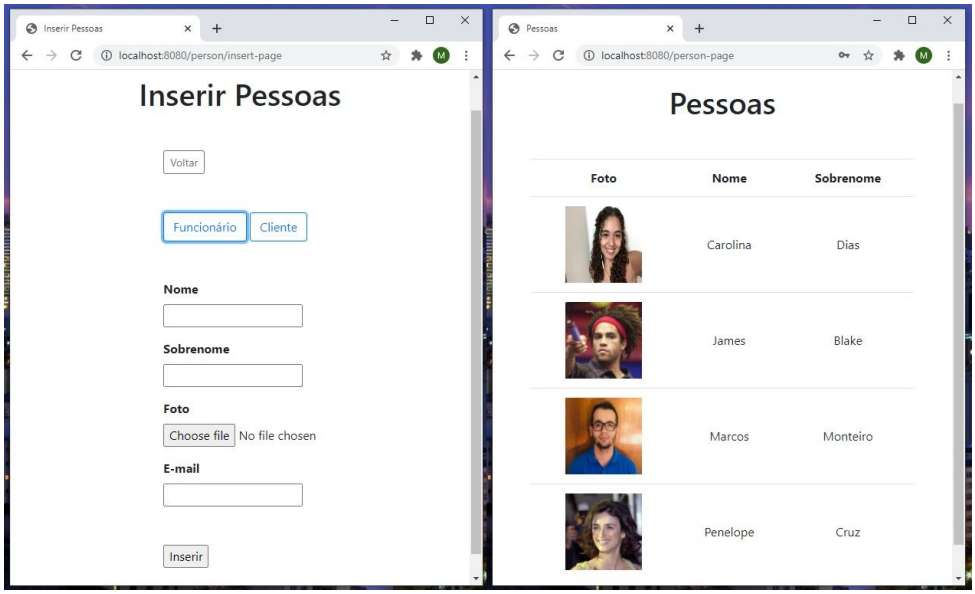
Nas Figuras 38 e 39, podemos observar alguns exemplos de páginas construídas por meio da implementação dos relacionamentos expostos ao longo desta seção.

Figura 38 – Exemplos das páginas Web construídas (1)



Fonte: própria

Figura 39 – Exemplos das páginas Web construídas (2)

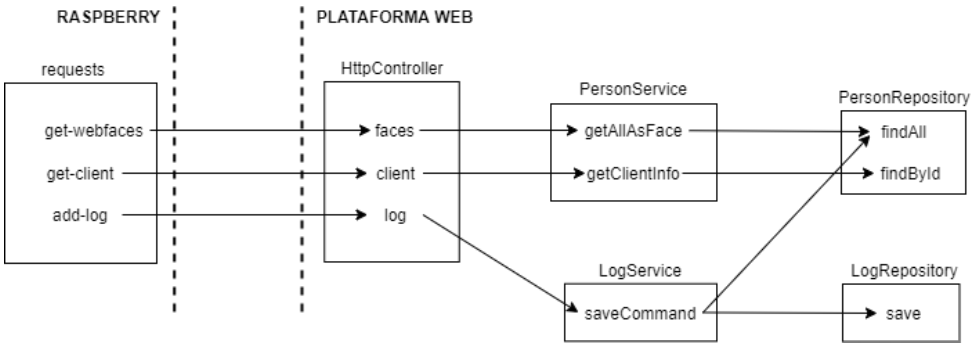


Fonte: própria

3.3 Integração

Como foi exposto na seção anterior, as requisições HTTP necessárias para comunicação entre os subsistemas requerem a implantação de servidores em ambos os lados. Com relação àquele presente na plataforma Web, tem-se que ele corresponde a mais um controlador, o `HttpController`; os relacionamentos para o mesmo estão expostos na Figura 40. Quando a *Raspberry* quer receber as faces presentes no banco durante a sua inicialização, faz uma requisição `get-web-faces`, que por sua vez chama `faces` e os métodos associados. Nesse procedimento, o serviço deve converter um objeto Pessoa em um novo objeto, chamado Face, que contém apenas informações relevantes para a *Raspberry*, diminuindo o tamanho da mensagem a ser transmitida.

Figura 40 – Relacionamento para requisições ao servidor da página Web



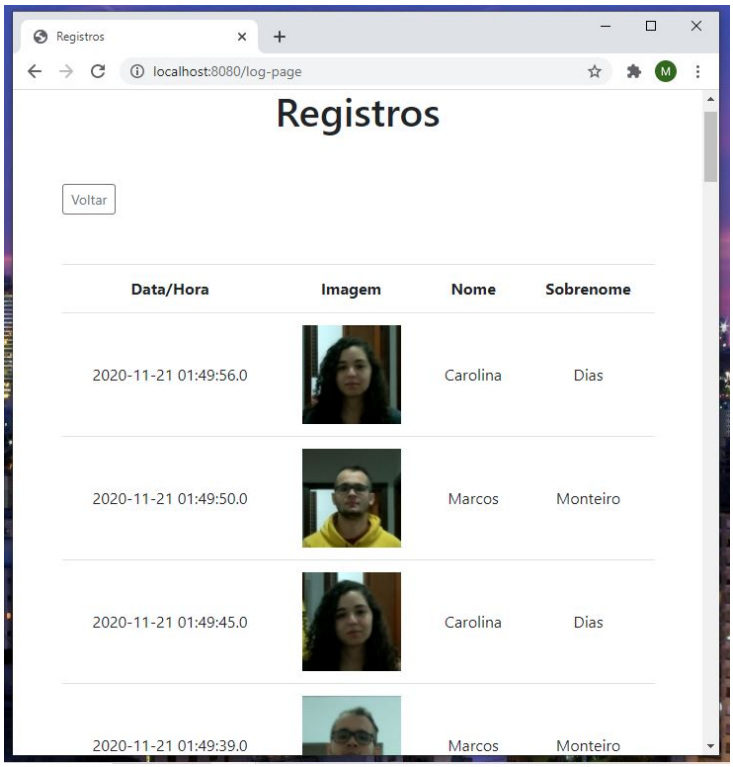
Fonte: própria

Quando um cliente obtém acesso do sistema, faz-se uma requisição `get-client`, que por sua vez chama `client` e os métodos associados. Nesse caso, o serviço deve coletar os objetos do cliente e de seu funcionário responsável e, a partir disso, gerar um novo objeto chamado ClienteInfo, que deve conter a mensagem de voz e os contatos do responsável para notificação. Já para adicionar um novo registro ao banco após uma movimentação, a *Raspberry* faz uma requisição `add-log`, que por sua vez chama `log` e os métodos associados. Aqui é necessário que o serviço receba a mensagem com o registro, formate a imagem para ser aceita pelo banco e faça uma busca para encontrar a pessoa associada a partir do seu nome. Um exemplo da página de registros construída pode ser observado na Figura 41.

Com relação ao servidor presente na *Raspberry*, implantado com o módulo `flask`, tem-se que ele recebe requisições da plataforma Web informando alterações de pessoas no banco, de modo que a *Raspberry* as faça também no seu armazenamento local. Com isso, temos os relacionamentos da Figura 42, que partem das páginas Web e chegam até a *Raspberry*, que, nesse caso, se coloca no lugar que usualmente é dos repositórios.

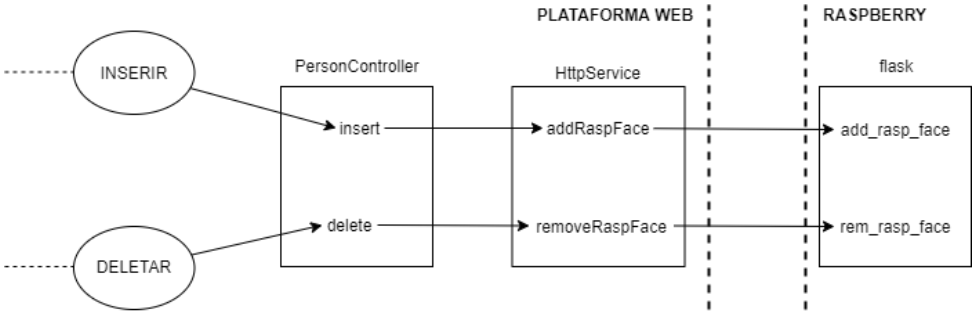
Por fim, tratemos da implementação dos elementos de interação e notificação.

Figura 41 – Exemplo da página Web de registros



Fonte: própria

Figura 42 – Relacionamento para requisições ao servidor da *Raspberry*



Fonte: própria

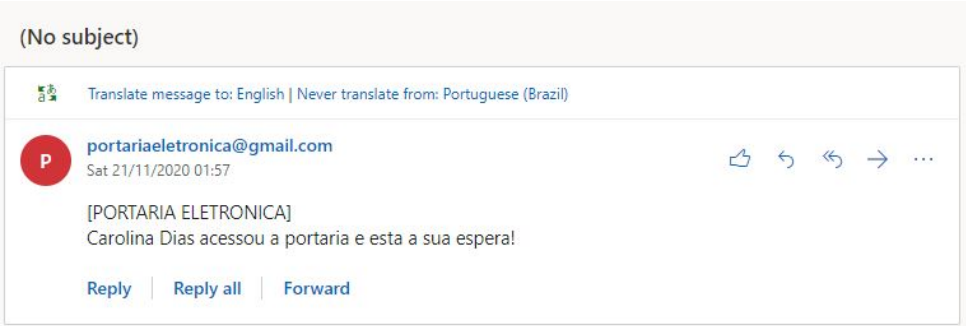
Para a interface gráfica, foi preciso adicionar duas *threads* no programa principal, pois a biblioteca `PyQt` exige que uma *thread* contenha o seu gerenciador, o qual permite que se exiba a interface ao mesmo tempo em que se executa o restante do programa. Além disso, renderizar a tela demanda um tempo considerável, o que prejudicaria o tempo de resposta caso fosse esse procedimento estivesse junto da *thread* de reconhecimento; assim, precisa de uma *thread* separada. Um exemplo tela exibida na interface gráfica pode ser vista na Figura 43.

Figura 43 – Exemplo de tela exibida na interface gráfica

Fonte: própria

Para a comunicação auditiva, foi preciso criar um novo serviço na plataforma Web que fica responsável por fazer uma requisição de sintetização de voz ao *AWS Polly* a cada vez que um cliente for adicionado. Em se tratando da reprodução da mensagem de voz, basta executar o *OMXPlayer* na *Raspberry* por meio de uma chamada ao sistema operacional utilizando a biblioteca `os`.

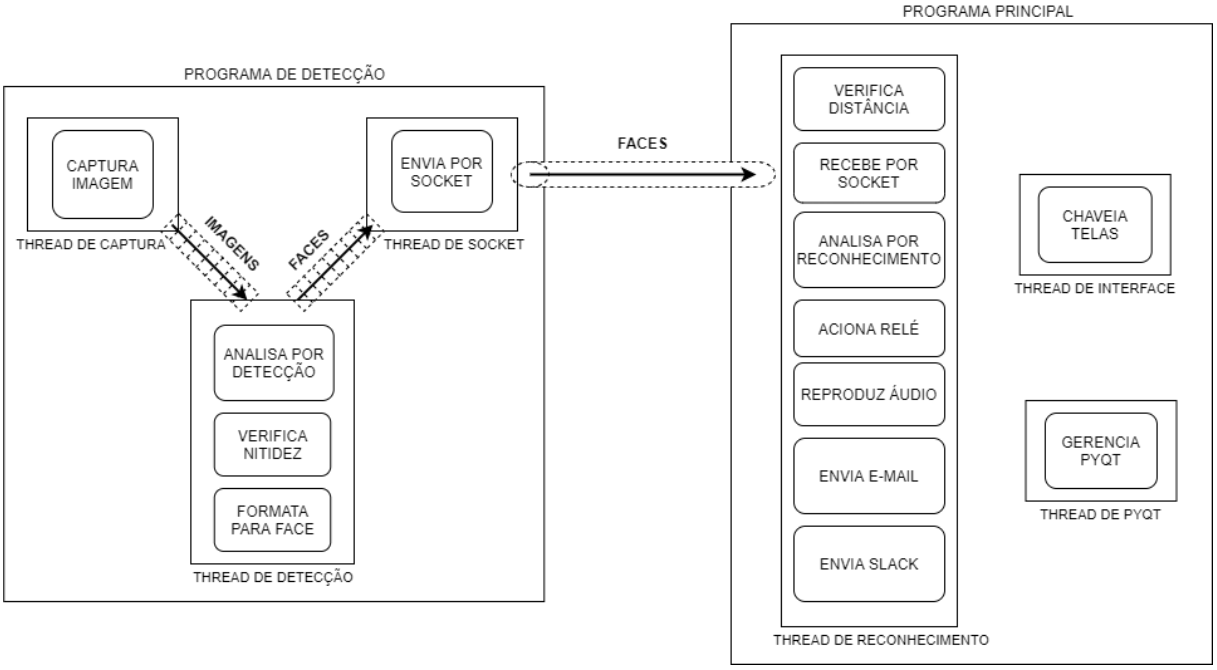
Figura 44 – Exemplo de e-mail enviado pela portaria



Fonte: própria

Como a notificação se trata apenas do envio de uma mensagem de texto simples, basta que se utilize as bibliotecas `smtplib` e `slack` de modo a fazer o login com as contas criadas para a portaria e enviar uma mensagem para os contatos obtidos pela requisição ao banco de dados; um exemplo de e-mail enviado pode ser observado na Figura 44. A partir do que foi exposto, tem-se o diagrama final dos processos executados na *Raspberry* na Figura 45.

Figura 45 – Arquitetura final do subsistema embarcado



Fonte: própria

4 CONCLUSÃO

Neste trabalho, foi proposto um sistema para realizar o controle de acesso a um estabelecimento empresarial por meio do reconhecimento facial. A utilização desse tipo de biometria traria benefícios como uma maior praticidade de acesso ao usuário e a possibilidade de cadastro de uma pessoa no sistema sem que fosse a necessária a sua presença física. Além disso, pretendia-se lidar com problemas específicos do ambiente do qual se trata a aplicação, facilitando a entrada de indivíduos que não frequentam o estabelecimento de forma recorrente, em especial, clientes da empresa.

Em se tratando do objetivo principal, obteve-se uma acurácia positiva no intervalo entre 76% e 100%, uma acurácia negativa acima de 95% e um tempo de resposta médio no intervalo entre $0,40s$ e $0,77s$. Esse último, por sinal, sendo menor que o tempo de resposta em torno de $1s$ para equipamento de controle de acesso atual do estabelecimento, que usa biometria por impressão digital. Além disso, pôde-se manter a dispersão desse tempo em um faixa aceitável em termos de não prejudicar a experiência do usuário, com o maior valor registrado sendo de $1,4s$ no pior caso.

Em se tratando dos demais objetivos, foi possível desenvolver uma plataforma Web que gerencia um banco de dados, permitindo cadastro, alteração e visualização de dados. Além disso, foram implementados procedimentos simples de interação e notificação que funcionam de forma integrada com o sistema. Entretanto, o aspecto de confiabilidade do sistema não foi abordado em um nível suficiente para o que uma aplicação como esta requer, havendo carência de restrições para alterações nos dados, de mecanismos de autoteste e de camadas de segurança no protocolo de comunicação.

Por fim, por ter sido realizado, em sua maioria, durante a pandemia de COVID-19, não foi possível fazer testes no estabelecimento tomado para aplicação, o que poderia ser de muito proveito no sentido de verificar se o sistema de fato atende às necessidades de todos os envolvidos e promovendo aprimoramentos conforme fossem ocorrendo conflitos. Vale destacar também que não foi encontrado um banco de imagens que representasse da melhor maneira possível o caso de uso deste projeto, o que provocou uma incerteza com relação à real acurácia do reconhecimento facial. Esses aspectos que eram previstos e não foram abordados satisfatoriamente ficam como sugestões para serem tratados por trabalhos futuros. Além destes, deve-se atentar à questão da acessibilidade, buscando adicionar elementos que permitam o uso do sistema por pessoas com deficiência, e pode-se aprimorar a plataforma Web de modo a torná-la mais prática para os funcionários que a operam, como se utilizando de imagens da rede social profissional Linkedin e desenvolvendo um aplicativo *mobile* para acesso à plataforma.

REFERÊNCIAS

AHONEN, T.; HADID, A.; 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.

ALSAADI, I. M. Physiological biometric authentication systems, advantages, disadvantages and future development : A review. **International Journal of Scientific and Technology Research**, v. 1, 2012.

AMAZON WEB SERVICES. **SearchFacesByImage - Reference**. 2020. Disponível em: <https://docs.aws.amazon.com/rekognition/latest/dg/API_SearchFacesByImage.html>. Acesso em: 14 dez. 2020.

ASSISTA INFOCOMM. **Face Recognition Biometrics Door Access System from Assista Singapore**. 2020. Disponível em: <http://www.assista.com.sg/dooraccess/face_recognition_fingerprint_door_access_control_system_singapore.htm>. Acesso em: 02 nov. 2020.

CAREY, N. **Establishing Pedestrian Walking Speeds**. 2005. Disponível em: <https://www.westernite.org/datacollectionfund/2005/psu_ped_summary.pdf>. Acesso em: 16 oct. 2020.

DIVYA, R. S.; MATHEW, M. Survey on various door lock access control mechanisms. **2017 International Conference on Circuits Power and Computing Technologies**, p. 1–3, 2017.

DLIB. **Dlib Face Landmark Detection**. 2020. Disponível em: <http://dlib.net/face_landmark_detection_ex.cpp.html>. Acesso em: 03 nov. 2020.

_____. **Dlib Face Recognition**. 2020. Disponível em: <http://dlib.net/dnn_face_recognition_ex.cpp.html>. Acesso em: 03 nov. 2020.

GEORGIA INSTITUTE OF TECHNOLOGY. **The Database of Faces (AT&T)**. 2020. Disponível em: <https://git-disl.github.io/GTDLBench/datasets/att_face_dataset/>. Acesso em: 27 sep. 2020.

HE, K. e. a. Deep residual learning for image recognition. **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 770–778, 2016.

HUANG, G. B. e. a. **Labeled Faces in the Wild: A Database for Study Face Recognition in Unconstrained Environments**. [S.l.], 2007.

INTELBRAS. **KT 740 - Ficha Técnica**. 2020. Disponível em: <https://backend.intelbras.com/sites/default/files/2019-06/Datasheet\%20KT\%20740\%20Prata.pdf>. Acesso em: 14 dez. 2020.

JAIN, V.; LEARNED-MILLER, E. **FDDB: A Benchmark for Face Detection in Unconstrained Settings**. [S.l.], 2010.

JONES, D. **Rapid Capture and Processing - Picamera Documentation**. 2016. Disponível em: <https://picamera.readthedocs.io/en/release-1.13/recipes2.html#rapid-capture-and-processing>. Acesso em: 14 dez. 2020.

_____. **Sensor Modes - Picamera Documentation**. 2016. Disponível em: <https://picamera.readthedocs.io/en/release-1.13/fov.html#sensor-modes>. Acesso em: 14 dez. 2020.

KADIR, K. e. a. A comparative study between lbp and haar-like features for face detection using opencv. **2014 4th International Conference on Engineering Technology and Technopreneuship (ICE2T)**, p. 335–339, 2014.

MATHWORKS. **vision.CascadeObjectDetector - Documentation**. 2020. Disponível em: <https://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-system-object.html>. Acesso em: 20 nov. 2020.

OPENCV. **How does the parameter scaleFactor in detectMultiScale affect face detection?** 2013. Disponível em: <https://answers.opencv.org/question/10654/how-does-the-parameter-scalefactor-in-detectmultiscale-affect-face-detection/>. Acesso em: 20 nov. 2020.

_____. **CascadeClassifier::detectMultiScale alogical influence of minSize parameter**. 2016. Disponível em: <https://answers.opencv.org/question/87371/cascadeclassifierdetectmultiscale-alogical-influence-of-minsize-parameter/>. Acesso em: 20 nov. 2020.

_____. **Cascade Classifier**. 2020. Disponível em: <https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html>. Acesso em: 03 nov. 2020.

_____. **Face Recognition with OpenCV**. 2020. Disponível em: <https://docs.opencv.org/3.4/da/d60/tutorial_face_main.html>. Acesso em: 03 nov. 2020.

OUR WORLD IN DATA. **Distribution of adult heights**. 2019. Disponível em: <https://ourworldindata.org/human-height#:~:text=Globally\%2C\%20the\%20mean\%20height\%20of,present\%20everywhere\%20in\%20the\%20world.> Acesso em: 14 dez. 2020.

PAIVA, R. P. **Machine Learning: Applications, Process and Techniques**. 2013. Disponível em: <https://eden.dei.uc.pt/~ruipedro/publications/Tutorials/slidesML.pdf>. Acesso em: 20 nov. 2020.

PERTUZ, S.; PUIG, D.; GARCIA, M. A. Analysis of focus measure operators for shape-from-focus. **Pattern Recognition**, Elsevier, v. 46, p. 1415–1432, 2013.

PUTTEMANS, S.; ERGUN, C.; GOEDEMÉ, T. Improving open source face detection by combining an adapted cascade classification pipeline and active learning. **Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)**, p. 396–404, 2017.

RUBIOLO, M.; STEGMAYER, G.; MILONE, D. Compressing arrays of classifiers using volterra-neural network: Application to face recognition. **Neural Computing and Applications**, v. 6, n. 23, p. 1687–1701, 2013.

SEGURANÇAJATO. **Kit KT753 Automatiza p/ Porta Deslizante usado em Fechadura FS150**. 2018. Disponível em: <https://www.segurancajato.com.br/kit-fechadura-eletroim-automatiza-fs150-kt753-para-porta-deslizante>. Acesso em: 02 nov. 2020.

STACKOVERFLOW. **OpenCV detectMultiScale() minNeighbors parameter**. 2014. Disponível em: <https://stackoverflow.com/questions/22249579/opencv-detectmultiscale-minneighbors-parameter>. Acesso em: 20 nov. 2020.

TAN, X. e. a. Face recognition from a single image per person: A survey. **Pattern Recognition**, Elsevier, v. 39, p. 1725–1745, 2006.

TOM'S HARDWARE. **Raspberry Pi 4: Review, Buying Guide and How to Use**. 2020. Disponível em: <https://www.tomshardware.com/reviews/raspberry-pi-4>. Acesso em: 07 oct. 2020.

UNIVERSITY OF MASSACHUSETTS. **Labeled Faceds in the Wild - Results**. 2020. Disponível em: <http://vis-www.cs.umass.edu/lfw/results.html>. Acesso em: 03 nov. 2020.