UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

GREGÓRIO MARIANO DE AZEVEDO NETO

FACELOCK: FECHADURA INTELIGENTE COM RECONHECIMENTO FACIAL

QUIXADÁ

2022

GREGÓRIO MARIANO DE AZEVEDO NETO

FACELOCK: FECHADURA INTELIGENTE COM RECONHECIMENTO FACIAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Thiago Werlley Bandeira da Silva

QUIXADÁ

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A987f Azevedo Neto, Gregório Mariano de.
FaceLock: fechadura inteligente com reconhecimento facial / Gregório Mariano de Azevedo Neto. – 2022.
78 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Computação, Quixadá, 2022.
Orientação: Prof. Dr. Prof. Dr. Thiago Werlley Bandeira da Silva.

1. Reconhecimento facial. 2. Interface de programação de aplicações. 3. Internet das coisas. 4. Fiware. I. Título.

CDD 621.39

GREGÓRIO MARIANO DE AZEVEDO NETO

FACELOCK: FECHADURA INTELIGENTE COM RECONHECIMENTO FACIAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: ___/___/____

BANCA EXAMINADORA

Prof. Dr. Thiago Werlley Bandeira da Silva (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Antônio Joel Ramiro de Castro
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulo Armando Calvacante Aguilar
Universidade Federal do Ceará (UFC)

À minha família, por sempre acreditar em mim e me apoiar em todos os momentos nessa caminhada. Aos meus amigos e colegas que participaram e me apoiaram nessa trajetória.

AGRADECIMENTOS

Meu Deus, obrigado(a) pelos teus planos para minha vida, pois são sempre maiores que meus próprios sonhos.

Agradeço a minha mãe Maria de Jesus, heroína que me deu apoio, incentivo nas horas difíceis, de desânimo e cansaço e sempre batalhou por mim.

Ao meu pai Pedro Amaro, onde no céu essa estrela brilha e sempre está olhando e torcendo por mim.

Aos meus irmãos, Pedro Júnior, Maria Luiza e Pedro Victor, por toda a ajuda e acompanhamento na realização deste trabalho.

Ao meu orientador, professor Dr. Thiago Bandeira, por toda dedicação, disposição, auxílio e orientação realizados a mim durante toda a execução deste trabalho.

A todos os meus colegas e amigos da graduação, por todas as experiências vividas, aprendizados e amizades construídas ao longo destes anos e todos os momentos compartilhados. Em especial quero agradecer aos meus amigos do grupo Resistência, Alan Nascimento, David Tavares, Jorge Lucas, Michael Douglas, Natan Nobre, Paulo Miranda e Ruan Felipe que estiveram comigo desde o início desta jornada, proporcionando inúmeros momentos que irei recordar para sempre em minha vida.

Agradeço a Gabriela Araújo, uma pessoa especial que me fortaleceu sempre com energias positivas.

A todos os professores da Universidade Federal do Ceará, Campus Quixadá, que colaboraram no processo de minha formação profissional e pessoal, com muito empenho e profissionalismo, compartilhando seus conhecimentos e vivências.

A todos que fizeram parte desse ciclo vivido ao longo destes 5 anos e contribuíram de alguma forma na minha formação profissional e pessoal.

“Mas quem tem medo de sofrer não merece o melhor da vida.”

(Projota)

RESUMO

A *Internet of Things* (IoT) é uma área de conhecimento que vem tendo um impacto na tecnologia exponencialmente nos últimos anos em todo o mundo, podemos perceber com muitos dispositivos eletrônicos, que hoje em dia estão conectados a rede *Wireless Fidelity* (Wi-Fi) e conseguem interagir entre-si. Como em uma *Smart Home* ou Casa Inteligente, que nela, existem dispositivos automatizados e controlados que possam comunicar-se uns com os outros, como, por exemplo, as *smart locks* ou fechaduras inteligentes, que possibilitam uma tranca e destranca de uma porta ou outro utensílio por meio do acesso remoto, como aplicações mobile, *Web* ou outras tecnologias IoT. Em busca por maior segurança no dia a dia, as fechaduras são utilizadas como meio de prevenir situações delicadas em que pessoas não autorizadas possam aderir pertence privado. Fechaduras convencionais apresentam falhas por não reconhecer os usuários que tentam acessar, visto que em muitos casos não existem um sistema de reconhecimento. Então a autenticidade ao processo de identificação de indivíduos é um fator relevante na segurança e proporciona uma grande escala de aplicações, pois o mesmo consegue estabelecer identificações de usuários. O reconhecimento facial faz com que as pessoas autorizadas exerçam controle de dispositivo com acesso seguro. Aderindo nas *smart locks*, é possível auxiliar pessoas com deficiência em membros superiores ou inferiores na tranca e destranca de uma porta, como, por exemplo, um deficiente dos membros superiores, conseguiria abrir sua porta com apenas o reconhecimento facial, torando prático e eficiente. Além disso, algumas pessoas com mobilidade reduzida têm uma grande dificuldade em acessar portas que estão distantes, dificultando a destranca ou tranca de portas, por exemplo, caso tenha escadas em sua residência, ter que descer para poder apenas destravar uma porta é algo que pode ser evitado. Este trabalho, propõe uma solução para *smart locks* ativadas por reconhecimento facial e por aplicação, baseado em *Apllication Programming Interface* (API) de reconhecimento facial. O procedimento é feito em aplicação *Web* desenvolvida em *Ruby on Rails* (RoR), um sistema embarcado controlado por um microcontrolador Esp32CAM, para comunicação entre os dispositivos, é utilizado o *midleware* de segurança denominado *FIWARE*. Esse sistema completo é denominado FACELOCK, os resultados obtidos pelo sistema proposto tem potencial para uso prático e propõe uma solução para que as pessoas tenham melhor controle das *smart locks*, em especial os deficientes.

Palavras-chave: Reconhecimento Facial. Interface de programação de aplicações. Fiware. Internet das coisas.

ABSTRACT

The Internet of Things (IoT) is an area of knowledge that has had an impact on technology exponentially in recent years around the world, we can realize with many devices electronics, which today are connected to the Wireless Fidelity network (Wi-Fi) and can interact with each other. As in a SmartHouse, in which there are devices automated and controlled that can communicate with each other, such as, for example, smart locks, which make it possible to lock and unlock a door or another appliance through remote access, such as mobile applications, the Web or other technologies IoT. In search of greater security in everyday life, locks are used as a means of prevent sensitive situations in which unauthorized persons can join private property. Conventional locks fail because they do not recognize users trying to access, since in many cases there is no recognition system. So authenticity to the process of identifying individuals is a relevant factor in security and provides a large scale of applications, as it manages to establish user identifications. Facial recognition makes authorized people exercise device control with secure access. By adhering to smart locks, it is possible to help people with disabilities in upper or lower limbs when locking and unlocking a door, such as, for example, a deficient in the upper limbs, he would be able to open his door with only the recognition facial, making it practical and efficient. In addition, some people with reduced mobility have great difficulty in accessing doors that are far away, making it difficult to unlock or door locks, for example, if you have stairs in your home, having to go down to be able to just unlocking a door is something that can be avoided. This work proposes a solution for App-based and Face-Recognition-enabled smartlocks, based on Apllication Programming Interface (API). The procedure is made in a Web application developed in Ruby on Rails (RoR), a embedded system controlled by an Esp32CAM microcontroller, for communication between the devices, the security middleware called FIWARE is used. This complete system is called FaceLock, the results obtained by the proposed system have potential for use practical and proposes a solution for people to have greater flexibility in controlling of smartlocks, especially the disabled.

Keywords: Face Recognition. Apllication Programming Interface. Fiware. Internet of Things

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

IoT	*Internet of Things*
Wi-Fi	*Wireless Fidelity*
API	*Apllication Programming Interface*
RoR	*Ruby on Rails*
RFID	*Radio Frequency Identification*
NFC	*Near Field Communication*
IP	*Internet Protocol*
FPGA	*Field-Programmable Gate Array*
EF	Entidades Físicas
EV	Entidades Virtuais
RDF	*Resource Description Framework*
OWL	*Ontology Web Language*
EXI	*Efficient XML Interchange*
GE	*Enabler* Generic
IDE	*Integrated Development Environment*
RNA	*Artificial Neural Network*
CNN	*Convolutional Neural Network*
REST	*Representational State Transfer*
URL	*Uniform Resource Locator*
GEM	*Bibliotecas do Ruby on Rails*
MQTT	*Message Queue Telemetry Transport*
UX	*User Experience*
UI	*User Interface*

LISTA DE SÍMBOLOS

ms	Tempo em milissegundos
s	Tempo em segundos
kHz	Frequência
Hz	Frequência
$k\Omega$	Resistência elétrica
V	Tensão Elétrica
mb	Megabytes

SUMÁRIO

1 INTRODUÇÃO

Atualmente no século XXI, é comum a comunicação entre máquinas por meio da Internet, o que possibilitou um avanço em diversas áreas como sistemas embarcados, micro-eletrônica, comunicação, aprendizado de máquina (do inglês *machine learning*), entre outras. Devido ao avanço dessas diversas áreas, se formou um conceito de Internet das Coisas (IoT, do inglês, *Internet of Things*) (MAGRANI, 2018).

O princípio de IoT foi construído junto a difusão do conceito de Internet, que era esperado comunicar os equipamentos que usamos no dia a dia com a Internet (OLIVEIRA, 2017). A IoT está mudando rapidamente os cenários da economia e vida social com muitos sistemas automatizados, que, estão presentes em setores tais como agricultura, transporte de pessoas, saúde, indústrias e presente também nas residências dos indivíduos (MAGRANI, 2018).

Sistemas que proporcionam automação residencial utilizam muito o conceito de IoT. Esse conceito está relacionado principalmente aos objetos físicos controlados de forma digital por meio da Internet, em que pode ser aplicado desde uma lâmpada até uma fechadura (MAGRANI, 2018). Esses objetos controlados, permitem que os usuários consigam controlar antes mesmo de chegarem às suas casas, isso se dá pelas tecnologias atuais implementadas, como a tecnologia de comunicação sem fio (Wi-Fi, do inglês *Wireless Fidelity*). Essa tecnologia promove comunicação entre usuários por diversos dispositivos (MARTINS *et al.*, 2017).

Na automação residencial, há uma vasta possibilidade de controle de objetos físicos. Porém, há alguns desafios que ao integrar objetos a Internet, são relevantes como, por exemplo, comunicação, dispositivos eletrônicos, energia e segurança (MARTINS *et al.*, 2017). O termo segurança é sempre manchete de jornais, revistas ou por qualquer outro meio de comunicação atualmente, portanto, considerando esse assunto e trazendo para automação residencial, é comum a população sentir necessidade de equipamentos seguros, que sejam eficientes e possam substituir fechaduras tradicionais (GROSSMANN, 2018).

No entanto, apesar das pessoas perceberem a carência da segurança e da privacidade de seus pertences contidos em suas residências, são poucas pessoas que procuram investir em travas de boa qualidade hoje em dia. Considerando também que várias dessas pessoas passam maior parte do seu dia distante de casa, é comum que se preocupem com a tranca residencial, gerando uma grande preocupação pessoal (GROSSMANN, 2018).

Para assegurar que nossos pertences não sejam roubados, danificados ou tenham acesso indesejado, trancas tradicionais são uma peça fundamental, porém são mais vulneráveis

a obstruções por invasores (MISHRA *et al.*, 2020). Nas fechaduras tradicionais, existem algumas necessidades, como, trancar a porta corretamente e fazer cópias de chaves. Alguns problemas também são encontrados, como, pessoas estão sujeitas a perdas de chaves e também há dificuldades de identificar suas chaves, além desses, encontram-se outros fatores que acabam influenciando na segurança.

As fechaduras inteligentes (do inglês *smart lock*) são equipamentos eletromecânicos que permitem ao usuário um acesso rápido e fácil a uma porta, gaveta, baús, cofres ou qualquer outro mecanismo de tranca e destranca assim como uma fechadura tradicional. Contudo, essas fechaduras conseguem transmitir e receber mensagens, principalmente, pela comunicação Wi-Fi, por um ou mais dispositivos que tenha acesso autorizado (MARGALHO, 2019). A facilidade com que as fechaduras podem interagir com os dispositivos, vêm possibilitando diferentes funcionalidades aplicadas em fechaduras tradicionais, tais como: acelerar no processo de acessibilidade e evitar problemas contra a segurança. Essas são as principais ações executadas ao ter uma fechadura tradicional. Portanto, as *smart lock* proporcionam segurança e acessibilidade aos usuários com funcionalidades inteligentes.

A viabilidade de fechaduras inteligentes vai além da segurança, pode-se perceber que atua também com acessibilidade para pessoas com deficiências físicas e pessoas que possuem sua mobilidade reduzida, como gestantes e idosos. Por exemplo, permite uma comodidade quando se refere ao deslocamento para abrir uma porta durante alguma visita, seja para parentes, ou seja, para amigos. Com relação a pessoas com problemas em locomoção, mover-se de um local para o outro se torna uma grande dificuldade, assim como as pessoas em que há comprometimento com os membros superiores ou inferiores (ANURADHA *et al.*, 2016). Essas são uma das grandes dificuldades enfrentadas por deficientes físicos que tem os membros inferiores comprometidos, mas devido a esse motivo, há algoritmos e tecnologias qualificadas para fornecer ao usuário reconhecimento facial, possibilitando identificar o usuário e efetuar a destranca da fechadura (JAHNAVI; NANDINI, 2019).

Portanto, compreende-se que as *smarts locks* são uma solução para diversos problemas do cotidiano. Dada a demanda de muitas pessoas solicitarem segurança, conforto, acessibilidade e fácil manuseio de uma fechadura, este trabalho propõe o desenvolvimento de um protótipo constituída por uma plataforma de IoT denominada FIWARE, onde estabelece de forma segura a comunicação entre dispositivos o recebimento e armazenamento de dados coletados, tornando um cenário IoT. O protótipo desenvolvido visa proporcionar ao usuário

acesso rápido por uma aplicação Web desenvolvida em RoR. Além disso, consegue fazer o reconhecimento facial das pessoas para controlar a *smart lock* proposta, em que esse protótipo visa contribuir com a tecnologia assistiva com o foco de auxiliar os usuários com deficiência física e com mobilidade reduzida.

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho tem como objetivo geral desenvolver uma fechadura inteligente controlada por Wi-Fi e com auxílio de uma câmera para realizar o reconhecimento facial, cujo o sistema completo foi denominada FACELOCK.

1.1.2 Objetivos Específicos

1. Empregar técnicas de reconhecimento facial na aplicação Web.
2. Desenvolver uma aplicação Web capaz de gerenciar usuários.
3. Coletar, tratar e armazenar os dados provenientes das aplicações.
4. Desenvolver um protótipo de fechadura inteligente.
5. Definir uma interconexão entre aplicação Web e protótipo.

1.2 Organização do Trabalho

Esse trabalho está organizado da seguinte forma:

- Capítulo 2: trata a fundamentação teórica, com os conceitos fundamentais para a compreensão da proposta descrita.
- Capítulo 3: abordam os trabalhos relacionados, comparando os aspectos comuns ou divergentes entre eles e o trabalho aqui proposto.
- Capítulo 4: descreve a metodologia que deve ser abordada para o desenvolvimento deste trabalho. Além disso, define o cronograma e o planejamento de conclusão deste trabalho.
- Capítulo 5: é demonstrado quais materiais e métodos foram utilizados para a construção do sistema FACELOCK, desde a montagem do *hardware*, *middleware* até a implementação do *software*.
- Capítulo 6: os experimentos e resultados obtidos do sistema FACELOCK, verificando

alguns pontos importantes, como a precisão da ferramenta utilizada para o reconhecimento facial.

- Capítulo 7: descreve a conclusão deste trabalho e informa sobre como o sistema se comportou mediante aos resultados e ideias para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capitulo apresenta e fundamenta uma visão geral dos principais conceitos relacionados a este trabalho, trazendo a colaboração de cada conceito para o trabalho proposto. A Seção 2.1 mostra o conceito de Internet das Coisas, Casa Inteligente ou *Smart Home* e aborda o conceito sobre a plataforma utilizada neste trabalho. Na Seção 2.2 é apresentado o tipo de comunicação utilizado neste trabalho. Na Seção 2.3 é apresentado os principais conceitos de sistemas embarcados e qual o principal microcontrolador está sendo utilizado neste trabalho. Na Seção 2.4 retrata a ideia de reconhecimento facial, redes neurais e aborda o principal algoritmo utilizado neste trabalho. Na Seção 2.5 é abordado sobre a ferramenta *Ruby on Rails* ou RoR, com objetivo de explicar o potencial de desenvolvimento rápido de páginas WEBs robustas e de alto desempenho.

2.1 Internet das Coisas (IoT)

Com o passar dos anos, as tecnologias vêm ganhando cada vez mais espaço no nosso cotidiano, com isso, surgem novos conceitos com o objetivo de facilitar a vida humana cada vez mais. Ashton (2009) propôs o termo Internet das Coisas ou IoT e dez anos depois escreveu um artigo enfatizando o termo para tecnologia de Identificação por Radiofrequência ou *Radio Frequency Identification* (RFID) (ASHTON, 2009). De acordo com Ashton, as pessoas necessitam conectar-se com a Internet através de variadas formas, isso é ocasionado pela falta de tempo oferecida pela rotina diária de cada pessoa (MAGRANI, 2018).

A IoT traz consigo o conceito de objetos inteligentes que podem ser interconectados para auxiliar efetivamente na resolução de problemas reais das pessoas. Dessa maneira, há vários produtos que hoje estão integrados com a tecnologia da IoT e estão contidos em diversas áreas, estabelecendo muitas funções, como eletrodomésticos, meios de transporte, automação residencial e brinquedos (MAGRANI, 2018).

Interligar objetos diferentes à Internet, significa criar objetos para Internet das Coisas (SANTOS *et al.*, 2016). Na IoT, os objetos podem estabelecer comunicação entre usuários e dispositivos. Com isto, surgem várias aplicações, tais como coleta de dados de sensores, sistema de monitoramento de idosos, sensoriamento de ambientes de difícil acesso, automação e controle de residências, entre outras (SANTOS *et al.*, 2016). Dessa maneira, a IoT permite uma vasta gama de dispositivos serem conectados por uma rede, sendo esses dispositivos heterogêneos ou

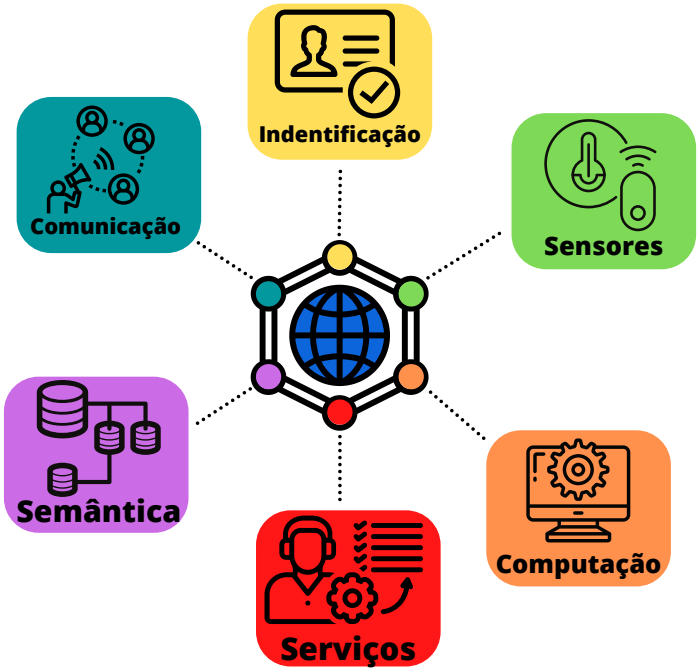
não.

O conceito de IoT torna-se um dos mais influenciadores neste trabalho, já que sua gama de aplicação e ferramentas incluídas ajuda a construir não só o conceito base do projeto, mas também define qual área deve atuar. Na subseção 2.1.1 é definido como a IoT é construída. Na subseção 2.1.2 é introduzido e definido qual área o projeto vai atuar. Na subseção 2.1.3 é apresentado a principal plataforma IoT que será utilizada neste projeto.

2.1.1 Blocos Fundamentais de Construção da IoT

A IoT tem como ponto de vista a junção de várias tecnologias, sendo assim, conseguem dar suporte para integração dos objetos no ambiente físico ao mundo virtual (SANTOS *et al.*, 2016). Na Figura 1 ilustram os blocos básicos de construção da IoT sendo eles (SANTOS *et al.*, 2016):

Figura 1 – Blocos básicos de IoT.



Fonte: Elaborado pelo próprio autor

Descrições sobre os blocos da figura 1.

- Identificação: este bloco é um dos mais essenciais, pois é necessário identificar os objetos exclusivos para conectá-los à Internet. As tecnologias como Led emissor/receptor infravermelho, RFID, *Near Field Communication* (NFC) e endereçamento *Internet Protocol* (IP)

são capazes de ser utilizado para identificar os objetos.

- Sensores/Atuadores: os sensores são responsáveis por coletarem informações sobre o contexto onde os objetos se localizam, após localizarem, conseguem armazenar/encaminhar esses dados para banco de dados, *clouds* ou centros de armazenamento. Os Atuadores conseguem controlar o ambiente ou executarem funcionalidades específicas segundo os dados lidos.
- Computação: é um bloco que contém a unidade de processamento, por exemplo, os micro-controladores, processadores e as *Field-Programmable Gate Array* (FPGA), encarregados de executar algoritmos nos objetos inteligentes e muitas vezes, responsáveis por integrar sensores.
- Comunicação: esse bloco é responsável pela relação de diversas técnicas empregadas para conectar objetos inteligentes. Algumas das tecnologias utilizadas são Wi-Fi, *Bluetooth* e RFID.
- Serviços: a IoT por ter uma vasta gama de aplicação, contém uma variedade de classes de serviços, e que contido nessas classes, evidenciam-se os Serviços de Identificação, onde são encarregados de mapear Entidades Físicas (EF) em Entidades Virtuais (EV) como, podemos exemplificar, a temperatura de um local físico em seu valor, coordenadas geográficas do sensor e instante da coleta. Além disso, exitem outros demais serviços, como, Serviços de Agregação, Serviços de Colaboração e Serviços de Ubiquidade.
- Semântica: remete a competência de obter conhecimento dos objetos na IoT e reutilizar, com o intuito de prover determinado serviço, como a técnica *Resource Description Framework* (RDF), *Ontology Web Language* (OWL) e *Efficient XML Interchange* (EXI).

2.1.2 Smart Home

Smart Home (ou para o português, Casa Inteligente) é uma área dentro da IoT que pretende impor conforto, convivência e a facilidade de integração entre os indivíduos e os dispositivos de suas residências, justamente pela forma a qual à IoT permite que os dispositivos estabelecidos com conexão a Internet possam ser controlados de qualquer lugar do mundo, mas nada impede que possam ser controlados por outros dispositivos sem acesso à Internet (GROSSMANN, 2018).

Sistemas de *Smart Home*, geralmente permitem que os usuários controlem remotamente ou programem uma série de dispositivos eletrônicos domésticos, como exemplo, te-

levisores, geladeiras, maquinas de lavar roupas, cafeteiras, fechaduras elétricas, entre outros eletrodomésticos automatizados que podem ser controlados digitando um único comando (ROBLES *et al.*, 2010).

Com a integração de dispositivos pessoais com a Internet, o impacto de outras tecnologias vem evoluindo exponencialmente, onde essa evolução permite afirmar que a implantação de tecnologias em residências passou a ter uma forte demanda (MTSHALI; KHUBISA, 2019). Como é notório, o principal fator que define uma instalação *Smart Home* é a junção entre os dispositivos combinado com à capacidade de executar funções e comandos mediante instruções programáveis (MURATORI; BÓ, 2011). Essa integração, remete ao principal ponto da *Smart Home*, as tecnologias sem fios, onde se destaca a comunicação *Wireless* (sem cabos). (GROSSMANN, 2018).

Assim, a *Smart Home* é composta por um emaranhamento de dispositivos e sistemas que podem fornecer aos usuários uma forma fácil de comunicar-se com suas residências e permita controlar remotamente todos os dispositivos através de um sistema pre-definido (MURATORI; BÓ, 2011).

2.1.3 Plataforma FIWARE

Um *middleware* IoT, conforme as leituras realizadas, é uma infraestrutura independente da aplicação que suporta o desenvolvimento de sistemas fundamentados em eventos, onde os *publishers* (ou, publicadores) notificam eventos para o *middleware* e *subscribers* (ou, subinscritos) se inscrevem com o *middleware* para obter notificações consideráveis (HERNANDES *et al.*, 2019).

FIWARE é um *middleware* que tem o intuito de criar um ecossistema aberto baseado em tecnologias da Internet. O objetivo dessa ferramenta é facilitar o custo e eficiência da criação e da despache de aplicações e serviços da Internet do futuro em diversas áreas, incluindo cidades inteligentes, transportes sustentáveis, automação residencial, monitoramento de idosos e entre outras (RAMPARANY *et al.*, 2014).

A plataforma FIWARE facilita o desenvolvimento de aplicativos inovadores, reduzindo os custos e a complexidade de auxiliar a um vasto número de usuários ao nível global e manipulando dados em grande escala. Além disso, o FIWARE busca oferecer um conjunto de documentos abertos, *open source* (código aberto), especificações que deixam aos desenvolvedores, provedores de serviços, empresas e outras organizações para construírem produtos que

satisfaça suas necessidades e seja inovador. Como mencionado anteriormente, o FIWARE é uma tecnologia de *open source*, suporta uma série de elementos chamados *Enabler* Generic (GE), que fornecem funções reutilizáveis e comumente compartilhadas pela comunidade servindo a várias áreas de uso em diversos setores. As GE's do FIWARE são partes desta tecnologia e abordam hospedagem em nuvem, de dados, gerenciamento, arquitetura de aplicativos, habilitação de serviços IoT, *interface* para redes e dispositivos, segurança e *interface* de usuário baseada na Web (PLAMANESCU *et al.*, 2019).

2.2 Comunicação

A comunicação estabelecida para dispositivos IoT é a parte que permite conectar os objetos, as "coisas" a rede, ou seja, é um dos grandes fundamentos da Internet das Coisas. Abaixo será descrito na subseção 2.2.1 o principal tipo de comunicação que o trabalho proposto utiliza, a comunicação Wi-Fi.

2.2.1 Wireless Fidelity (Wi-Fi)

A tecnologia Wi-Fi (do inglês, *Wireless Fidelity*) é um protocolo de comunicação que atualmente é bastante conhecido, onde traz uma solução de acesso à rede sem fio. Visto que, está presente nos mais variados lugares, estando em cômodos de casas, escritórios, indústrias, lojas comerciais e até espaços públicos das cidades, proporciona o acesso fácil para quem utiliza e eficiente para conexão de dispositivos. O nome original dado para a tecnologia Wi-Fi corresponde a IEEE 802.11 (LÓPEZ-PÉREZ *et al.*, 2019).

O Wi-Fi foi desenvolvido como uma possibilidade de substituir o padrão cabeado conhecido por *Ethernet*. Essa tecnologia permite ter poucas preocupações com dispositivos em que há consumo energético limitado, como podemos citar, as aplicações desenvolvidas para IoT. Assim, não necessariamente é esperado que alguns dispositivos utilizados em IoT adotem a tecnologia Wi-Fi como principal protocolo de comunicação. Entretanto, o Wi-Fi possui várias vantagens, como alcance de conexão e vazão, o que o torna apropriado para navegação na Internet em dispositivos móveis, como *smartphones*, tablets e notebooks. Tem como vantagem também o uso para conectar dispositivos heterogêneos, como conectar equipamentos eletrodomésticos em uma casa, como ar-condicionado através de um *smartphone* (SANTOS *et al.*, 2016).

2.3 Sistemas Embarcados

A capacidade de executar tarefas dentro de um circuito, equipamento ou sistema, integrado, é conhecido por sistemas embarcados (MARWEDEL, 2021). Este sistema é comparado com um computador de uso comum, pois possui memória, processador, dispositivos de armazenamento e outros periféricos (MARWEDEL, 2021). A principal diferença desse sistema, é que, ele tem como conceito realizar uma tarefa específica de maneira contínua e em muitas vezes, sem falhas, conseguindo estabelecer uma rede de comunicação com o exterior. Ainda mais nos últimos anos, motivados por um novo paradigma da computação denominado IoT, que auxilia na comunicação de qualquer coisa do mundo físico com o ambiente virtual, possibilitando que máquinas, dados e pessoas se comuniquem de forma adequada (MARWEDEL, 2021).

Os sistemas embarcados, comumente utilizam sensores para obter parâmetros e com eles controlar, ajustar e configurar aplicações ou dispositivos para atender as necessidades das pessoas. Além disso, sistemas embarcados estão cada vez mais baratos e acessíveis, estabelecendo menor consumo de energia, enxutos e robustos (MARWEDEL, 2021). Um grande exemplo de itens que usam sistemas embarcados são os smartphones, dispositivos frequentemente utilizados que desempenham funções específicas, e que contam com mecanismos mais limitados que os computadores.

Neste projeto, é utilizado o conceito de Sistemas Embarcados para construir a parte do *hardware* para controlar a fechadura, alguns requisitos são estabelecidos com ajuda das informações obtidas do conceito de Sistemas Embarcados, como, qual placa de desenvolvimento que mais se adéqua ao projeto, quais componentes e como implementar um código para efetuar a comunicação necessária dos dispositivos. Na subseção 2.3.1, foi escolhido a placa microcontroladora Esp32 CAM para auxiliar no desenvolvimento do projeto, tendo a função de enviar e coletar dados para fechadura, como opção também, capturar imagem do usuário pela câmera nela contida e entre outras funcionalidades.

2.3.1 Esp32 CAM

A placa de desenvolvimento ESP32 é um conjunto de funcionalidades com custo eficiente de energia em um microprocessador de *chip* com combinação da tecnologia Wi-Fi e Bluetooth. ESP32 é projetado e produzido pela empresa *Espressif Systems* e é construído pela TSMC. A ESP32 é um herdeiro do microprocessador ESP8266 (RAJ *et al.*, 2019). A

configuração é dada por meio da porta USB e para maior afinidade, pode-se utilizar a *Integrated Development Environment* (IDE) Arduino. Há uma grande facilidade em definir uma conexão Wi-Fi e entrada/saída, os pinos podem ser configurados por códigos simples. Por ter rede Wi-Fi integrada, faz da ESP32 ser bastante utilizado para se comunicar por meio da Internet (ROHITH, 2021).

O módulo ESP32 CAM é outra versão da placa de desenvolvimento baseada no módulo ESP32S Wi-Fi, sendo um módulo eletrônico altamente tecnológico implementado especificamente para conectar projetos robóticos ou de automação residencial com à Internet, propondo maior facilidade e tendo menor custo (SHAH *et al.*, 2021). O grande destaque desse modelo é a compatibilidade com uma câmera digital Ov2640, possibilitando ter mais funcionalidades, chegando a ter aplicabilidade de detecção e reconhecimento facial, registrar imagens e vídeos diretamente no microcontrolador (ROHITH, 2021). Além de que, apresenta um LED (do inglês, *Light Emitter Diode*) que pode ser usado como *flash*, com intuito de iluminar o local próximo à lente da câmera. Esta placa tem um grande potencial para ser utilizada em IoT. Outro diferencial exclusivo do ESP32 CAM é que, além de possuir conexão Wi-Fi nativa, também têm a tecnologia Bluetooth embutido e um microprocessador dual core 32-bit LX, tornando o projeto ainda mais prático, robusto e aumenta as possibilidades de uso (ROHITH, 2021). Na Figura 2 mostra a representação dessa placa de desenvolvimento.

Figura 2 – Placa Esp32 CAM.

Fonte: Elaborado pelo próprio autor com base (ROHITH, 2021)

2.4 Reconhecimento Facial

O reconhecimento facial pode ser conhecido como uma das funções mais essenciais dos seres humanos, pois, a partir do reconhecimento facial o ser humano consegue identificar uma vasta quantidade de faces/rostos e determinar aspectos psicológicos evidenciados por expressões (BIANCHINI, 2001).

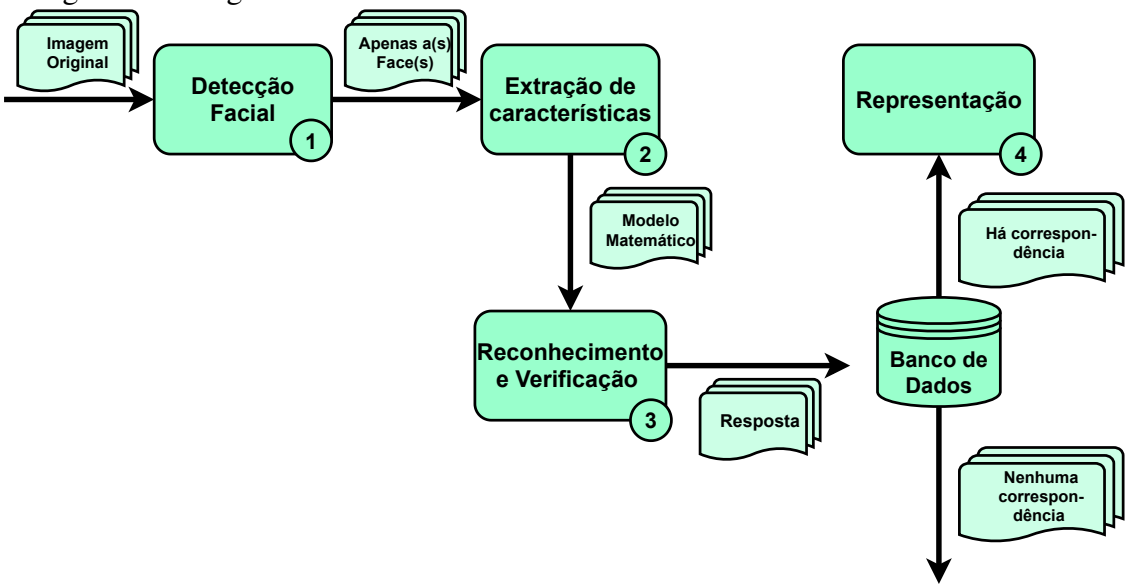
O surgimento dos primeiros trabalhos a respeito de reconhecimento facial se deu início em torno de 1960, porém só veio a ser implementado na década seguinte (BRAGA *et al.*, 2019). Desde então, diversas pesquisas foram feitas para prover sistemas de reconhecimento facial, já que teve uma enorme demanda de aplicações indispensáveis no mercado. A inserção de novas tecnologias no processo tornou cada vez mais robusto e eficaz a tecnologia de reconhecimento facial (BRAGA *et al.*, 2019). Um exemplo bastante promissor, trazendo para o século XXI, são os *smartphones*, pois conseguem desbloquear a tela através de um reconhecimento facial, isso só confirma o quão a tecnologia de reconhecimento facial está avançando atualmente.

O reconhecimento facial não pode ser observado de maneira independente, já que corresponde a uma técnica que usa várias tarefas que formam o processamento facial. Dependendo da área de aplicação, um sistema de reconhecimento facial pode ser constituído por algumas etapas que podem ser separadas nos seguintes grupos (BRAGA *et al.*, 2019) (BIANCHINI, 2001):

1. Detecção de faces: dada uma imagem qualquer, identifica o local onde aparece(m) a(s) face(s).
2. Extração de características e Representação da face: identifica as partes constituintes de uma face.
3. Reconhecimento e Verificação: é a categorização de uma face como familiar.
4. Representação: seleciona as informações da face que serão utilizadas para representá-la.

A Figura 3 ilustra um típico sistema de reconhecimento facial. Esse sistema consistem em uma imagem de entrada, podendo ser pré-processada para reduzir a taxa de erro do sistema e exibir ou não a imagem reconhecida pelo sistema de reconhecimento facial (BRAGA *et al.*, 2019) (BIANCHINI, 2001).

Figura 3 – Diagrama de um Sistema de Reconhecimento Facial.



Fonte: Elaborado pelo próprio autor

Para sistemas de reconhecimento facial, pode-se identificar pelo menos duas categorias (BIANCHINI, 2001):

1. sistemas onde o objetivo é reconhecer a face de uma pessoa dentro de uma base de imagens, não necessitam usualmente ser em tempo real, como no caso da área policial, que retorna várias imagens para reconhecimento de possíveis faces.
2. sistemas onde o objetivo é reconhecer a face de uma pessoa em individual e em tempo real, como os sistemas de segurança, ou permitir o acesso de algumas pessoas e negar outras, como no caso de acesso ao computador, *smartphones*, etc.

Independente de qual categoria está sendo utilizada, um dos grandes problemas dos modelos computacionais é tratar a dificuldade de encontrar padrões visuais. Apesar de que todas as faces sejam constituídas por padrões reconhecidos universalmente (boca, olhos e nariz), elas detêm de mudanças únicas. Dessa forma, para o reconhecimento facial faz-se de grande importância o uso destas variações para determinar algumas características relevantes (RUBACK *et al.*, 2021).

Existem algumas dificuldades que atrapalham no processo de reconhecimento facial mesmo após várias pesquisas executadas atualmente, dentre as principais dificuldades encontram-se (BIANCHINI, 2001):

1. iluminação.
2. ângulos e poses.
3. cosméticos e acessórios.

4. expressões.
5. extração da face do contexto ou do fundo.

Um bom exemplo seria, conforme o sistema que está sendo utilizado, a taxa de erro aumenta consideravelmente quando a iluminação do ambiente não favorece, ou quando a pessoa tem sua face reconhecida e capturada de perfil e não de forma frontal e também pode depender de algumas expressões faciais que acabam dificultando a detecção da face do indivíduo (BRAGA *et al.*, 2019).

Além disso, os algoritmos de detecção facial necessitam ser treinados exaustivamente antes de obterem um resultado satisfatório. Esses algoritmos precisam ser treinados com uma sequência de imagens de diversas faces diferentes e também com uma sequência de imagens de objetos que não são parecidos com faces. As redes neurais são uma solução que vem sendo bastante utilizada para aplicação de sistemas de reconhecimento facial, onde é aplicada diretamente sobre as imagens faciais, como uma grande ferramenta capaz de selecionar as informações mais relevantes da imagem (RUBACK *et al.*, 2021).

O conceito de Reconhecimento Facial é indispensável para este trabalho, já que neste projeto utiliza-se de reconhecimento facial para identificar indivíduos para efetuar a ação de destranca de fechaduras. Nesta mesma seção, exitem algumas subseções que contribuem para melhor enfatizar essa ideia. Na subseção 2.4.1 contribui com um dos principais conceitos de treinamento de algoritmo que pode ser é utilizado direto ou indiretamente neste projeto. Na subseção 2.4.2 aborda um pouco sobre o principal conceito mais utilizado para reconhecimento de faces, onde é ponto fundamental para o entendimento neste projeto. Na subsubseção 2.4.2.1 aborda o possível principal algoritmo utilizado neste projeto, pois ele tem uma grande precisão no reconhecimento de imagens.

2.4.1 Machine Learning

O aprendizado automatizado ou Aprendizado de Máquina (ML, do inglês *Machine Learning*), corresponde a programar computadores para eles poderem “aprender"com as informações transmitidas para eles. De certa forma, a ML é o processo de converter experiência em conhecimento. Um algoritmo de aprendizagem recebe uma entrada, essa entrada contém dados de treinamento, retratando experiência, e a saída desse algoritmo é alguma especialização, sendo normalmente estabelecido por outro software que pode executar algumas tarefas.

Existem diversas tarefas que nós seres humanos executamos rotineiramente, mas

nossa própria avaliação sobre como nós as fazemos, não é suficientemente bem preparada para fornecer informações que possam tornar um programa bem definido, como, por exemplo, tarefas que incluem direção, reconhecimento de voz, compreensão da imagem e reconhecimento de pessoa. Em todas essas tarefas, programas de ML atualizados, são programas que aprendem com seu treinamento, tem resultados bastantes satisfatórios, se tem um conjunto grande de exemplos para treinamento. Ainda assim, existem outras diversas tarefas que se beneficiam de técnicas de Aprendizado de Máquina, onde estão ligadas à análise de vários e complexos conjuntos de dados, como, dados astronômicos, transformando arquivos médicos em conhecimento médico, previsão do tempo, Web Motores de busca e comércio eletrônico. Dessa forma, ML é um ramo que está presente em várias áreas pela diversidade de algoritmos bem estruturados que o compõe (SHALEV-SHWARTZ; BEN-DAVID, 2014).

2.4.2 Redes Neurais

Os projetos baseados em Redes Neurais Artificiais (do inglês *Artificial Neural Network* (RNA)), ou comumente é apelidado “redes neurais”, é um grande parâmetro computacional que se baseia no processo de informações do comportamento do cérebro humano. O cérebro humano possui um funcionamento como um computador, com a capacidade de organizar estruturas capazes de fornecer e processar informações conhecida por neurônios, que realizam reconhecimento de padrões, percepções, coordenações motoras, etc. Em termos não formais, um neurônio está em constante “desenvolvimento”, onde o sistema nervoso em desenvolvimento consegue se adaptar ao meio ambiente, estando em contato com vários fatores que podem causar suas mudanças. Sendo o ser humano capaz de extrair todas as informações e adapta-la, assim obtendo “experiências” (LEITE, 2020).

De forma geral o termo rede neural é uma máquina construída para modelar como o cérebro executa uma atividade particular ou função de relevância a partir de suas aprendizagens (“experiências”). Essas redes neurais são geralmente implementadas usando componentes eletrônicos ou é simulada por programação em um computador sendo utilizadas principalmente na modelagem de relações complexas entre entradas e saídas, ou para encontrar padrões em dados. *Machine learning* (Aprendizagem de Máquina) e *Artificial Intelligence* (Inteligência Artificial) são os dois grandes ramos que mais utilizam esse conceito para estudos e construção de aplicações e uma das principais áreas onde as redes neurais têm sido bastante é na de processamento de imagem, onde abrange reconhecimento de faces, objetos, entre outros (LEITE,

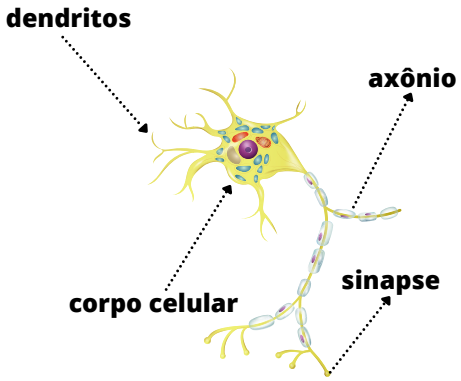
2020).

Tendo em mente que as RNA tiveram sua origem fundamentadas no modelo do sistema nervoso biológico, na Figura 4 podemos observar o modelo biológico do neurônio.

No modelo biológico representado pela Figura 4, podemos perceber que o neurônio é composto por alguns componentes, sendo eles (BIANCHINI, 2001):

- detritos: encarregados por adquirir impulsos provenientes de outros neurônios;
- corpo celular: capaz de processar os sinais recebidos;
- axônios: encarregados pela reprodução dos sinais;
- sinapse: é representado pela ligação entre o axônio de um neurônio e o dentrito de outro.

Figura 4 – Representação de um Neurônio Biológico.

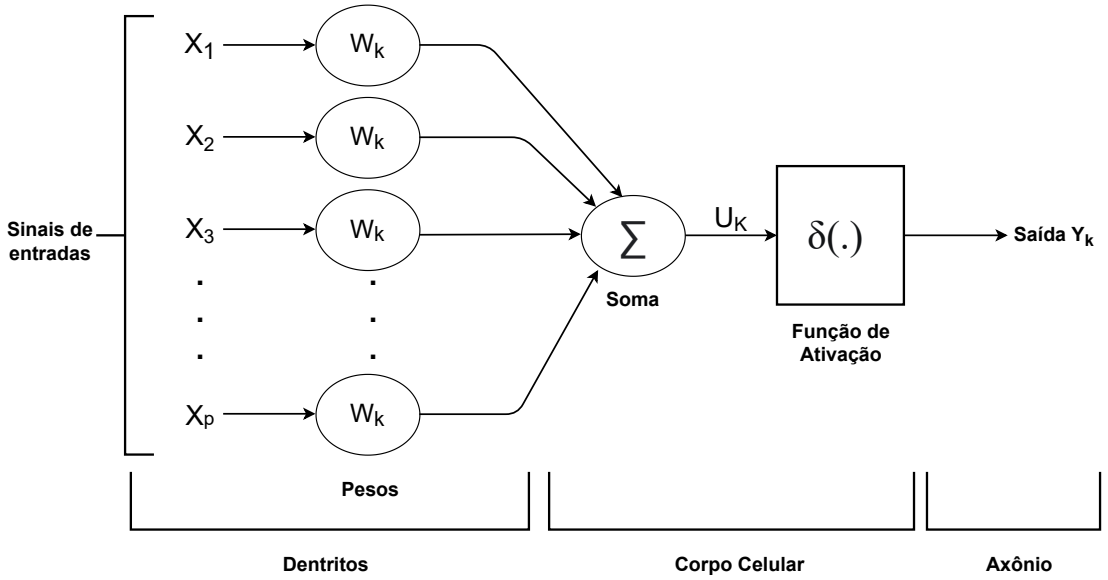


Fonte: Elaborado pelo próprio autor baseado em (BIANCHINI, 2001)

Na Figura 5 mostra-se a analogia do modelo computacional de um neurônio com um biológico, nessa analogia, tem-se o modelo computacional com características semelhantes ao do modelo biológico, tendo os seguintes elementos (BIANCHINI, 2001):

- sinais de entradas: podem ser comparados a um conjunto de sinapse, como especificado no modelo biológico, a sinapse permite a entrada dos sinais para o neurônio, no neurônio computacional, existe um peso associado a cada sinal de entrada, onde é executado uma operação de multiplicação. Dessa forma, cada neurônio pode verificar e ajustar o impulso recebido de maneira distinta;
- soma: é comparado com o corpo celular, pelo fato de estar realizando o processamento dos sinais. No modelo computacional executa o somatório de todos os resultados gerados da multiplicação dos sinais pelos pesos;
- axônios: encarregados pela reprodução dos sinais;
- função de ativação: encarregado por limitar a amplitude de saída de um neurônio.

Figura 5 – Comparação do modelo de um neurônio biológico e um artificial



Fonte: Elaborado pelo próprio autor baseado em (BIANCHINI, 2001)

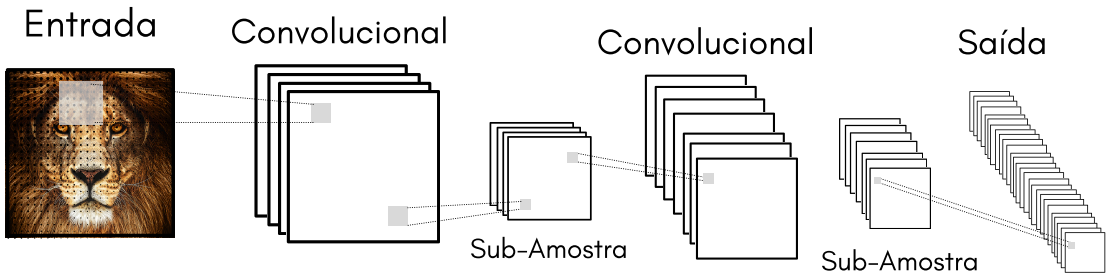
2.4.2.1 Algoritmo Rede Neural Convolucional (CNN)

Um algoritmo de aprendizado profundo que pega as imagens de entrada e os diferencia é conhecido como Rede Neural Convolucional (do inglês, *Convolutional Neural Network* (CNN)). Há uma grande semelhança do CNN com o RNA, para alguns aspectos, mas em CNN exige mais pré-processamento em comparação com qualquer outro algoritmo (RAHMAN *et al.*, 2020).

Para melhor visualização, um modelo comum de CNN pode ser representado pela Figura 6. Essa figura corresponde a um agrupamento de camadas incluindo um ou mais planos. Inicialmente é comum que a imagem seja apresentada para a primeira camada, determinada como camada de entrada (RAHMAN *et al.*, 2020). Em cada unidade em um plano obtém entrada de uma limitada vizinhança de um plano da camada preliminar. Os campos receptivos locais é uma pequena janela nos píxeis de entrada. Um ou mais plano possuem pesos determinados que são forçados a serem iguais, no intuito de obter as mesmas características (BIANCHINI, 2001). Cada plano pode ser classificado como um mapeamento de características, onde são informadas características já pré-estabelecidas. Dessa forma, múltiplos planos são comumente usados em cada camada, com finalidade que múltiplas características possam ser descobertas. Estas camadas são determinadas como camadas convolucionais. Após cada camada convolucional descobrem-se camadas que executam as operações de Sub-Amostragem, como demonstrado na Figura 6, encarregadas pelo fator de tolerância a distorções da rede, ou seja, essa camada reduz a

amostra da imagem e reduz a dimensionalidade, obtendo redução em esforços computacionais (VARGAS *et al.*, 2016).

Figura 6 – Exemplo de uma rede convolucional



Fonte: Elaborado pelo próprio autor baseado em (BIANCHINI, 2001)

De maneira geral, uma CNN é apta de aplicar filtros em dados visuais, permitindo à relação de vizinhança entre os píxeis da imagem durante todo o processamento da rede (RAHMAN *et al.*, 2020). Este tipo de rede vem sendo amplamente utilizado, principalmente nas aplicações de classificação, detecção e reconhecimento em imagens e vídeos. Tendo em mente que as CNN formam uma das arquiteturas de *Deep Learning* mais amplamente usada em tarefas de Visão Computacional e reconhecimento de imagens. Neste trabalho, é utilizado a abordagem CNN para reconhecimento de faces.

2.4.3 API SkyBiometry

Uma API é um agrupamento de padrões e rotinas definidos e arquivados para ser utilizada por uma por alguma aplicação (VALLADARES, 2019). Sem ter precisão que aplicação conheça realmente as particularidades da implementação do software, usando apenas seus métodos de modo a integrar dados entre diversas fontes e softwares ou aplicações.

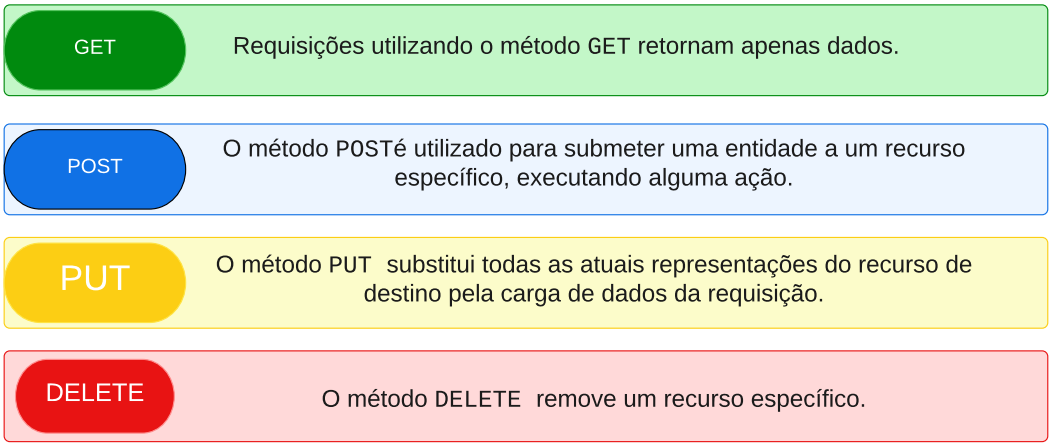
Uma API é conhecida como RESTful, quando atende aos princípios de *Representational State Transfer* (REST). De certa forma, REST é a abstração de uma arquitetura e comunicação usada no desenvolvimento Web, que possibilita o desenvolvimento de uma interface bem compacta, permitindo que as aplicações realizem uma comunicação de forma simples (VALLADARES, 2019).

A utilização do estilo de arquitetura REST para a construção de serviços Web tornou-se cada vez mais popular e com isso foram propostas e desenvolvidas diversas *frameworks* capazes de providenciar aos programadores uma forma mais simples e acessível de implementar

aplicações seguindo uma arquitetura REST. Para o estudo, já existente algumas soluções de APIs no mercado, tendo vários framework em diversas linguagens de programação destinadas à implementação de serviços REST, como nas linguagens Python, Java e Ruby (VALLADARES, 2019). Dessa forma, essa variedade de linguagem, permitem a conexão com APIs direcionada para reconhecimento facial.

Para APIs REST é utilizado o protocolo HTTP (Hypertext Transfer Protocol), o que proporciona o serviço de resposta às requisições solicitadas através dos navegadores de Internet (BARRETO *et al.*, 2015). Dessa forma a figura 7 abaixo, define um pouco sobre as principais requisições feitas, os GETs, POSTs, PUTs e DELETE utilizado em APIs.

Figura 7 – Requisições REST



Fonte: elaborado pelo autor.

Um exemplo de API para reconhecimento facial, e está sendo utilizada neste projeto, é a 2.4.3.1. Ela permite fazer todas as requisições REST para as suas *Uniform Resource Locator* (URL) específicas, sua implementação é sofisticada, tendo como confirmação vários algoritmos de aprendizagem de máquina e inteligência artificial (ROZANSKA *et al.*, 2018).

2.4.3.1 SkyBiometry

SkyBiometry é uma API de biometria que trabalha em nuvem, nela introduz-se o reconhecimento facial e análise de rosto. Essa ferramenta também consegue agrupar faces semelhantes sem a necessidade de realizar-se manualmente. Muito usada para reconhecimento de emoções através de expressões, essa ferramenta pode ser utilizada de forma gratuita com seu plano para uso pessoal (ROZANSKA *et al.*, 2018).

A ferramenta em questão funciona através de uma detecção de rosto, em seguida, é realizado o reconhecimento facial, posteriormente apos a realização da captação desses traços, ela consegue indicar o nome, de possivelmente quem pertence o rosto em questão. Devido a seu funcionamento em nuvem, pode ser integrada em diversos dispositivos que trabalham com *IoT* ou possuem algum acesso à *Internet*, assim possibilitando a integração reconhecimento fácil nesses dispositivos com um auxílio de uma câmera (MATIAS, 2016).

2.5 Ruby on Rails

Ruby on Rails ou RoR é um *framework* construído sobre a linguagem *Ruby*. O objetivo dessa ferramenta é fornecer aos desenvolvedores uma estrutura intuitiva para desenvolver rapidamente páginas da Web robustas e de alto desempenho. Esse *framework* possui princípios que norteiam o seu funcionamento. Através da abordagem utilizada é permitido a redução do tempo necessário para criação do aplicativo, no entanto, quando utilizado de forma inadequada pode causar conflitos e comportamentos imprevisíveis como resultado (HARTL, 2015).

Em *Rails* é possível ter uma estrutura de aplicação Web que inclui o padrão *Model-View-Controller* (MVC). Esse padrão divide a aplicação em três camadas: *Model*, *View* e *Controller*, cada uma desempenhando funcionalidades específicas (HARTL, 2015). Na camada *Model* é responsável por encapsular a lógica de negócio da aplicação. No *Rails*, as classes de modelo baseadas em banco de dados são derivadas de ActiveRecord::Base. Essa classe permite apresentar os dados das linhas do banco de dados como objetos, tornando a implementação mais simples. Através do *Controller* é possível lidar com solicitações HTTP. Normalmente, isso significa retornar HTML, mas os controladores *Rails* também podem gerar XML, JSON, PDFs, visualizações específicas para dispositivos móveis (VISWANATHAN, 2008).

A camada *View* é composta por *templates* que são responsáveis por fornecer representações apropriadas dos recursos de sua aplicação (VISWANATHAN, 2008). As visualizações geralmente são renderizadas para gerar uma resposta do controlador ou para gerar o corpo de um *e-mail*, por exemplo. No *Rails*, a geração de *Views* é tratada pela *Action View*. O *Ruby* também possui códigos executáveis chamados de *Gems* que auxiliam no desenvolvimento da aplicação (VISWANATHAN, 2008). Essas bibliotecas oferecem serviços importantes para a construção de sistemas *WEB* como o gerenciamento de banco de dados, autenticação de sistemas, compartilhamento de recursos, etc.

3 TRABALHOS RELACIONADOS

Nesta seção, será retratado e comparado uma revisão da literatura com relação aos tópicos *smart locks* e reconhecimento facial com o trabalho proposto.

3.1 Sistema de Gestão de Fechaduras Inteligentes Usando IoT para Aplicação em Cacifos de Universidade (MARGALHO, 2019)

Margalho (2019) propõe a ideia de desenvolver *smart locks* (fechaduras inteligentes) para cacifos públicos universitários utilizando a tecnologia de IoT. O sistema engloba uma componente de computação na *Cloud*, sendo responsável por todas as comunicações entre os módulos do sistema e pelo armazenamento de dados, bem como pela segurança dos acessos concedidos. Foi desenvolvido uma aplicação para *smartphone* com o intuito de gerir os acessos do usuário por um sistema de gestão de fechaduras inteligentes que utiliza comunicação *Wi-Fi* e também utiliza a tecnologia RFID como outro tipo mecanismo de destrava da fechadura.

No presente trabalho, serão fundamentados conceitos de *smart lock* voltados para portas residenciais, o que não foi feito no trabalho de Margalho (2019). Além disso, no presente trabalho, será desenvolvido uma aplicação Web que tenha acesso por multiplataformas, como *smartphones* ou computadores com interface de gerir várias fechaduras de diferentes cômodos.

3.2 *A Study on the IoT Based Smart Door Lock System* (JEONG, 2016)

O trabalho de Jeong (2016) propõe o plano de aprimoramento de segurança baseado no sistema de travamento inteligente de portas não tripuladas, como caixas eletrônicos, quiosques e máquinas de venda automática. Com base na tecnologia IoT, o sistema permite travar a porta ao conectar-se com um *smartphone* e utiliza meios de comunicação com e sem fio, como USB, canal de áudio e tecnologia *Bluetooth*.

Neste trabalho, é utilizado a comunicação sem fio *Wi-Fi* para ter acesso as fechaduras de qualquer lugar que tenha acesso à Internet, diferente do trabalho do Jeong (2016). Além disso, a proposta deste trabalho é feita para fechaduras de portas residenciais e permite que um sistema de gerenciamento estabeleça acesso às *smart locks* para vários usuários, o que no trabalho de Jeong (2016) não permite que usuários consigam conectar-se de forma simultânea.

3.3 *Remotely Accessible Smart Lock Security System with Essential Features* (PINJALA; GUPTA, 2019)

Em Pinjala e Gupta (2019) enfatiza a acessibilidade remota em tempo real para controle de fechaduras inteligentes. Os autores, realizam um estudo com uma microcontrolador Raspberry Pi que permite o controle de acesso da fechadura através do pressionamento do botão de uma campainha instalada na residência, que ainda tem uma câmera de monitoramento de visitas. Assim, quando um visitante pressiona o botão da campainha uma câmera é ligada e notifica no *smartphone* do proprietário que uma visita acabou de tocar a campainha e mostra imagens do visitante em tempo real. O proprietário, pode autorizar a entrada ou não do indivíduo através do aplicativo instalado em seu *smartphone*, facilitando efetuar a ação de abrir/fechar a porta remotamente. Além disso, permite que o proprietário por meio do aplicativo, encaminhar uma mensagem de voz para o visitante, alertando se está ou não presente em sua residência.

Neste trabalho, visa atender algumas multiplataformas, como *smartphones* quanto *computadores* para acesso remoto a fechadura. Além de que, neste trabalho é utilizado também uma interface Web que permita o usuário não só visualizar a visita através da câmera, mas também que possa reconhecer o rosto do visitante, assim, permitindo que a visita entre sem precisar do acesso do proprietário, isso se dá pelo fato de que um algoritmo de reconhecimento facial é atribuído ao sistema para indivíduos cadastrados no sistema.

3.4 Sistema de Telemonitoramento de Baixo Custo Usando IoT (LISLE; TEIXEIRA, 2020)

Os autores Lisle e Teixeira (2020) propõem um sistema de monitoramento por vídeo utilizando IoT com foco no monitoramento de crianças para detecção de movimento. Esse trabalho utiliza uma microcontroladora Raspberry Pi Zero W, uma câmera e a biblioteca OpenCV para monitorar, processar e enviar o vídeo e suas informações para outros dispositivos conectados na mesma rede *Wi-Fi*.

Neste trabalho utilizam-se vários algoritmos da biblioteca OpenCV para testar a precisão e incluir no projeto o que Lisle e Teixeira (2020) não fazem. Essa análise é feita conforme as amostras cadastradas pelo usuário para avaliar o comportamento dos algoritmos. Além disso, este trabalho irá reconhecer faces das pessoas e vincular a uma aplicação para fechaduras inteligentes ao contrário do Lisle e Teixeira (2020) que utiliza para monitoramento

de crianças.

3.5 *Convolutional Neural Network Based Smart Door Lock System* (MISHRA *et al.*, 2020)

Mishra *et al.* (2020) desenvolveu um sistema que pode conceder acesso a fechaduras eletrônicas simplesmente capturando a face do indivíduo. Um sistema de bloqueio de fechaduras baseado em CNN (Convolucional Rede Neural) é desenvolvido no microcontrolador Raspberry Pi. No sistema proposto, captura imagens da pessoa pela câmera Pi (própria do microcontrolador). Se alguma correspondência ao usuário que solicita acesso não for encontrada, o administrador do sistema é alertado por e-mail de um acesso indesejado.

Neste trabalho, o sistema além de fornecer alerta ao usuário administrador de um acesso indesejado, o sistema também será capaz de fornecer imagem do usuário que teve acesso negado. Além do reconhecimento de faces, neste trabalho, outras medidas serão implementadas para efetuar o desbloqueio da fechadura, o que não foi feito no trabalho de Mishra *et al.* (2020), como o gerenciamento de usuários, que a partir desse gerenciamento, os usuários podem desbloquear as fechaduras pela interface Web sem o reconhecimento facial.

Quadro 1 – Comparativo entre os trabalhos relacionados e o trabalho proposto

Trabalho	**Utiliza tecnologia *Wi-Fi***	**Plataforma de Aplicação é Web**	**Utiliza algoritmo de reconhecimento Facial**	**Dados armazenados em nuvem**	**Funciona sem internet**	**Permite acesso simultâneo de vários usuários**	**Utiliza Fiware**
(MARGALHO, 2019)	Sim	Não	Não	Sim	Sim	Sim	Não
(JEONG, 2016)	Não	Não	Não	Não	Sim	Não	Não
(PINJALA; GUPTA, 2019)	Sim	Não	Sim	Sim	Não	Não	Não
(LISLE; TEIXEIRA, 2020)	Sim	Sim	Não	Não	Não	Não	Não
(MISHRA *et al.*, 2020)	Sim	Não	Sim	Não	Sim	Não	Não
Trabalho Proposto	Sim	Sim	Sim	Sim	Não	Sim	Sim

Fonte: elaborado pelo autor.

3.6 Análise Comparativa

O Quadro 1 compara o presente trabalho com os trabalhos relacionados abordados anteriormente em alguns pontos, sendo estes pontos: (I) se o trabalho utilizada tecnologia *Wi-Fi*; (II) se utiliza uma plataforma de aplicação Web; (III) se os trabalhos utilizam algoritmo de reconhecimento Facial; (IV) se os dados do usuário são armazenados em nuvem; (V) se o sistema funciona sem Internet; (VI) se o sistema permite acesso simultâneo de vários usuários para controle de acesso da fechadura; (VII) se utiliza algum *middleware* para interferir na comunicação dos dispositivos, no caso o FIWARE.

O presente trabalho utiliza ou aborda todos os pontos mencionados. Também, neste trabalho é testado localmente, apesar de funcionar sem Internet para a tranca e destranca pelo aplicativo, o reconhecimento facial necessita de Internet para que o sistema funcione sem Internet, por isso não foi considerado que funcione sem Internet, utiliza o reconhecimento facial para identificar indivíduos, assim como o trabalho do Mishra *et al.* (2020).

Todos os trabalhos relacionados, exceto o de Jeong (2016), utilizam tecnologia *Wi-Fi*, o que também é fornecido por este trabalho proposto.

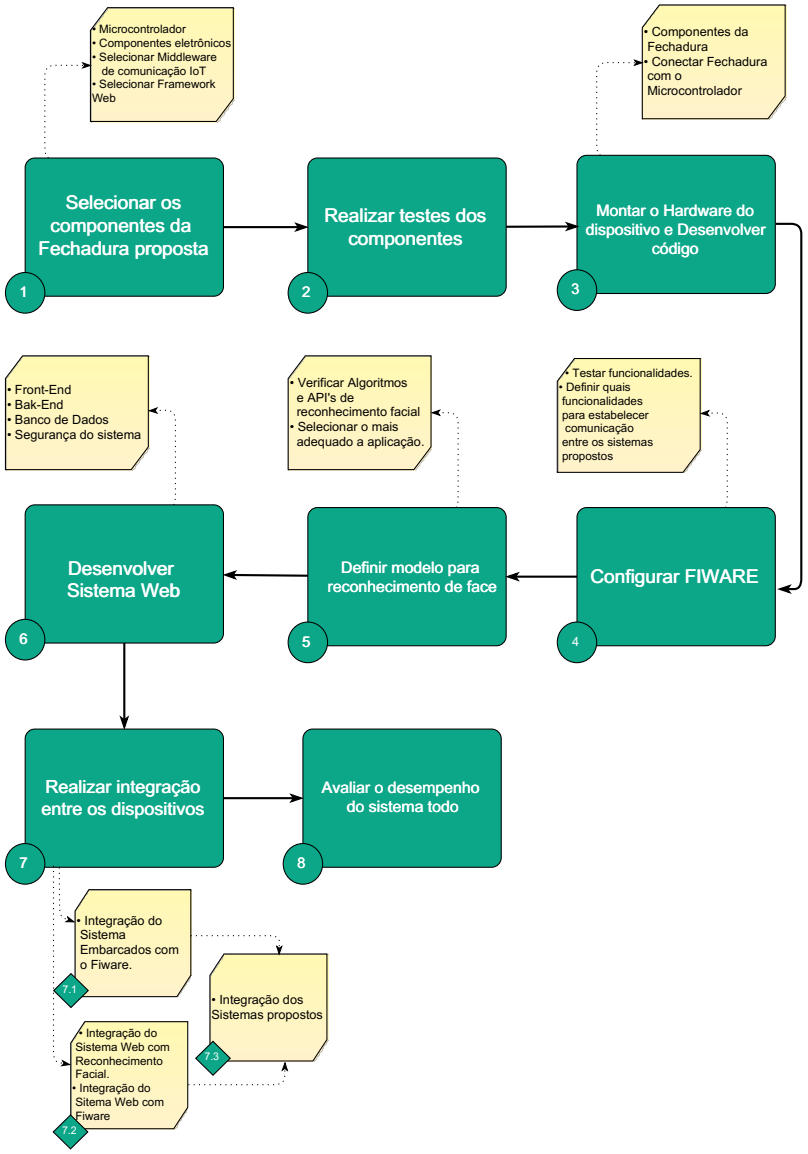
O trabalho de Margalho (2019) utiliza armazenamento em nuvem para controle de requisições e controle de acesso do usuário, baseado nesse conceito, neste trabalho também utiliza armazenamento em nuvem para estabelecer a consolidação dos dados armazenados, utilizando uma plataforma de serviços em nuvem para testes, como o *Heroku* ambiente de hospedagem de aplicações em nuvem.

Para controle de acesso sem Internet, Jeong (2016) utiliza a tecnologia *Bluetooth* para controle de acesso, Margalho (2019) utiliza tecnologia RFID e Mishra *et al.* (2020) usa o reconhecimento facial integrado no microcontrolador para desbloquear trancas, neste trabalho, é utilizado o sistema localmente, onde permite executar funções de desbloqueio da fechadura sem se conectar a Internet, porém, como o sistema disponibiliza mais funcionalidade, como banco em nuvens, é testado com banco localmente também, evitando problema sem Internet, mas o reconhecimento facial como é realizado por API, há a necessidade de conexão com Internet, então descartando a ideia de funcionar sem.

4 PROCEDIMENTOS METODOLÓGICOS

Neste Capítulo são apontados as etapas necessárias para a realização deste trabalho. Às sete subseções a seguir são representadas por etapas de desenvolvimento onde: a Seção 4.1 introduz-se a escolha dos componentes para a implementação da *smart lock* proposta, como microcontrolador, componentes eletrônicos (resistores, solenóide 12V, etc.), o *middleware* como plataforma de comunicação IoT e o *Framework* Web para criar a aplicação, proposta; a Seção 4.2 realizam-se os testes de funcionamento simples nos componentes selecionados, para verificar se está funcionando como o esperado para serem implementados; a Seção 4.3 trata-se a montagem do *hardware* da *smart lock* atribuído no microcontrolador, implementando o sistema embarcado; a Seção 4.4 configura-se e testa-se mais afundo a funcionalidade do *middleware* selecionado, FIWARE; a Seção 4.5 descrevem-se os testes para selecionar o algoritmo ou API ter mais precisão para reconhecimento facial; a Seção 4.6 aborda-se a ferramenta necessária para criar o sistema Web, para gerenciamento dos usuários e fechaduras, essa ferramenta foi escolhida, sendo ela o *Ruby on Rails* ou RoR; a Seção 4.7 realiza-se a integração dos dispositivos, nesta Seção, foi efetuado a comunicação entre os sistemas; a Seção 4.7.1 trata-se a integração do sistema embarcados com o FIWARE; a Seção 4.7.2 constitui-se a integração do Sistema Web com o reconhecimento facial e com o FIWARE; e por último a Seção 4.7.3 executa-se a integração do sistema todo, denominado FACELOCK; a Seção 4.8 analisa-se o desempenho do sistema e dos resultados dos experimentos. Na Figura 8 ilustram-se os passos da metodologia adotada.

Figura 8 – Fluxograma dos procedimentos.



Fonte: Elaborado pelo próprio autor

4.1 Selecionar os componentes para a *Smart Lock* proposta

Para construção da solução proposta, a princípio foi realizado um estudo sobre as principais tecnologias e componentes utilizados para o desenvolvimento de uma *smart lock*. Os trabalhos relacionados contido neste trabalhos, alguns constroem ou modelam o protótipo da *smart lock*, sendo assim, com base nesses trabalhos, foi determinado a montagem do *hardware* com alguns componentes fixos como, por exemplo, um solenoide de 12V, fonte de 12V, resistores, transistor, relé e um módulo ESP32 CAM, entre outros, para auxiliar nos requisitos do sistema e na construção do protótipo embarcado. Sendo assim, com alguns componentes *hardware* já determinado, a quantidade de cada componente é definida conforme a necessidade do projeto. Além disso, foi estabelecido como uma componente essencial para a solução de comunicação

do projeto, um *middleware* que funcione de forma segura e seja eficiente, foi definido então a utilização do FIWARE como plataforma IoT. Outro componente essencial para o projeto, também foi definido, o *framework* RoR como principal ferramenta para desenvolvimento do sistema Web.

4.2 Realizar testes nos componentes

Quando já escolhido os componentes aplicados na construção do protótipo, faz-se necessário os testes em cada componente para verificar se cada um desses está funcionando segundo o esperado. Nesta etapa, os componentes eletrônicos foram avaliados pela efetuação de testes de tensão, corrente e se há danificação do produto, esses pontos avaliados devem evitar falhas quando empregado no projeto. Além disso, foi testado também em conjunto com outros componentes para certificar-se que quando conectados funcionem de maneira correta. Também foi pensado na melhor maneira possível, com os recursos disponíveis do desenvolvedor, como acoplar o protótipo em uma porta real.

O *middleware* escolhido, FIWARE, foi instalado de maneira correta no ambiente de desenvolvimento escolhido. Teve seus microsserviços escolhidos e testados de modo a verificar se estão aptos para gerenciamento de dados e troca de informações para os componentes escolhidos no projeto. Foi executado testes de inicialização do servidor conforme a própria documentação do *middleware*.

Com o RoR, foi testado a criação de uma aplicação Web, se consegue suportar comunicação com o FIWARE, se o banco de dados funciona corretamente, testar seu modelo *MVC* como abordado sobre ele neste documento e as *Bibliotecas do Ruby on Rails* (GEM) que tem grande papel no funcionamento do *framework*, possibilitando fazer várias implementações desejadas, como autenticidade do usuário, manipulação de imagem, etc.

4.3 Montar o *Hadware* do dispositivo e Desenvolver o código

Após testar todos os dispositivos, se estão executando conforme o esperado, esta etapa condiz a junção dos componentes para montar a arquitetura da *smart lock*. A princípio, todos os componentes escolhidos estarão dispostos em uma *protoboard*, essa é uma placa de prototipagem, contida de furos e conexões internas para montagem de circuitos, sendo a base para o projeto funcionar. Nesta etapa também, além do circuito montado na *protoboard*, foi

estabelecido também, a conexão com o microcontrolador já funcionando.

O microcontrolador utilizado, foi o módulo ESP32 Cam, atendendo as funcionalidades especificadas, onde suporta a gravação de código estabelecidos pelo Arduino IDE, onde se torna a principal IDE estabelecida para desenvolver o código responsável pelo comportamento do microcontrolador.

4.4 Configurar Fiware

O FIWARE, é uma plataforma que oferece microserviços para o gerenciamento de dados de contexto das informações geradas por dispositivos. Os microserviços usados dessa plataforma configurados com os seguintes componentes: Orion Context Broker, componente encarregado por fazer o gerenciamento dos dados de contexto, tanto para enviar quanto receber mensagens e o IoT Agent que permite a comunicação direta com os dispositivos físicos com diferentes protocolos. Além disso, com o FIWARE, conseguimos definir e estabelecer a comunicação utilizando o protocolo *Message Queue Telemetry Transport* (MQTT), que permite a comunicação entre dispositivos, por publicações (mensagens enviadas) e assinaturas de tópicos (responsável por receber mensagens), que, de maneira geral, um dispositivo se inscreve em um tópico a qual quer receber e enviar mensagens.

Sendo assim, inicializando um servidor *broker* MQTT Mosquitto para permitir a comunicação via protocolo MQTT entre os dispositivos físicos e o componente *IoT Agent* que entrega e recebe as mensagens do *Orion Context Broker*.

4.5 Definir modelo para reconhecimento de face

Com a parte de *hardware* e do *middleware* já bem definida e construída, agora deve-se voltar atenção para a implementação do algoritmo mencionado por este trabalho. Nesta etapa, foi verificado a possibilidade de usar a biblioteca OpenCV, que, esta biblioteca é uma das principais *open source* para processamento de imagem, inteligência artificial e aprendizagem de máquina, que o projeto proposto precisa. Também foi viabilizado a utilização de API's que possam executar o reconhecimento facial e por requisições feitas, entreguem uma resposta como resultado.

A partir da OpenCV, foi testado alguns algoritmos de reconhecimento facial e optado por utilizar apenas o reconhecimento de rosto, e para o reconhecimento facial de imagens, foi

optado por utilizar API's já implementadas para tal funcionalidade, sendo a API definida, API SkyBiometry, como a responsável por reconhecer os rostos das pessoas. Além disso, para que essas imagens sejam enviadas para a API SkyBiometry, foi implementado um algoritmo utilizando a linguagem de programação *python*, que utiliza a biblioteca OpenCV, como mencionado no início do parágrafo e tem como funcionalidade reconhecer apenas o rosto do usuário através de uma câmera e então enviada para análise da imagem, esse código pode ser definido como uma API também, visto que ela será responsável diretamente pela captação de imagens da câmera do sistema, logo foi apelidado de API Cãmera.

4.6 Desenvolver Sistema Web

Com o *framework* já escolhido, RoR, foi implementado um sistema que consiga gerenciar usuários de forma segura, gerenciar fechaduras e o reconhecimento facial. O Sistema Web foi dividido em 4 preocupações, sendo elas: *Front-End*, *Back-End*, Banco de Dados e Segurança do Sistema.

- *Front-End*: a parte visual para o usuário, mostrando os principais recursos do sistema, seria o controle dos usuários, da fechadura e do reconhecimento facial. Tendo telas que podem ser simples e de fácil entendimento.
- *Back-End*: parte responsável por manter todo o controle do fluxo entre as diferentes integrações, como, FIWARE, API's, banco de dados, *front-end* e etc,
- Banco de Dados: parte responsável por armazenar todas as informações necessárias para gerenciamento do sistema, o PostgreSQL foi o banco de dados definido e utilizado.
- Segurança do sistema: utilização de gerenciamento de usuários com uma GEM específica do framework, que ela é responsável por fazer a autenticação dos usuários e das requisições que o sistema web realiza ou recebe, o nome é GEM *Devise*.

4.7 Realizar Integração entre os dispositivos

Esta é uma das etapas mais importantes do projeto, onde os sistemas construídos já foram escolhidos e construídos, assim, estarão disponíveis para serem integrados e efetuarem a comunicação entre si. Nessa etapa a plataforma FIWARE escolhida será responsável por efetuar a comunicação e integração dos dispositivos para construir o sistema apelidado de FACELOCK, onde será integrado o sistema Web com o *hardware* construído. Também é utilizado a integração

das API's com a aplicação Web, onde foi integrado conforme o desejado.

4.7.1 Integração do Sistema Embarcados

Essa integração foi feita, onde o sistema utiliza o servidor para está realizando publicações e inscrições em tópicos específicos e gerenciados pelo FIWARE.

4.7.2 Integrações do Sistema Web

A integração do sistema Web com as API's foram feitas, de forma que o framework permite utilizar de requisições REST, que facilitam a comunicação com a API SkyBiometry, visto que ela aceita essas requisições. Também com a API desenvolvida para o reconhecimento de rosto através da câmera, foi estabelecido com sucesso essa conexão. Também ocorreu a integração com o FIWARE, visto que o sistema Web precisa publicar e se inscrever em tópicos para poder enviar e receber mensagens específicas, para manter o gerenciamento e controle do sistema.

4.7.3 Integrações do Sistema proposto

Por fim, nesta etapa, com a integração já estabelecida do sistema embarcado, com a do sistema Web também já estabelecida, foi feito a integração dos dois sistemas e unificando no nome do projeto, o FACELOCK, assim verificando e testando a comunicação entre o sistema Web e o sistema embarcado por meio do *middleware*.

4.8 Analisar o desempenho do sistema todo

Por fim, nesta etapa, com a integração já realizada, o desempenho do sistema foi testado em todos os pontos definidos acima. Primeiro foi testado o algoritmo de reconhecimento de rostos utilizado no projeto e a API de reconhecimento facial por imagens. Após testar essas API's, a comunicação entre os sistemas é testada por tempo de resposta, verificando se não há nenhuma interferência nesse meio de comunicação entre a aplicação Web com o *hardware*, também foi analisado qual banco de dados é mais apropriado para o uso do sistema. Além disso, o sistema Web foi testado para evitar falhas de acesso ou qualquer *bug* identificado. Por fim, os resultados foram analisados por requisições executadas pelos sistemas, tabelas e comportamentos em diferentes, e representações gráficas, modelos de imagens e experimentos com usuários reais,

essas análises resultaram no quão o trabalho pode contribuir para sua vida cotidiana.

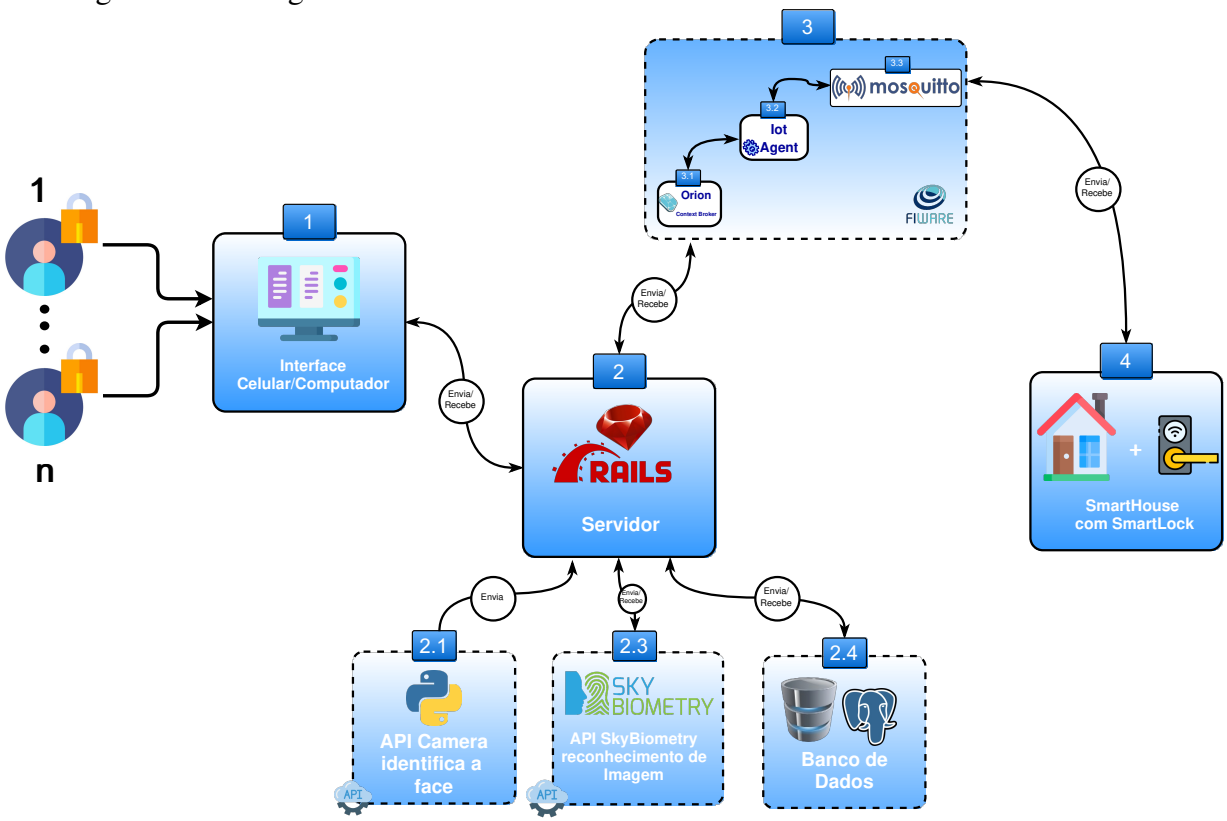
5 MATERIAIS E MÉTODOS

Nesta seção, será detalhado a solução proposta para o protótipo do sistema FACE-LOCK. Cada uma das subseções seguintes explica sobre os componentes, implementações e métodos utilizados para com que eles consigam interagir entre si e oferecer as funcionalidades propostas.

5.1 Modelo do sistema FaceLock

Este trabalho propõe uma solução para o gerenciamento e manipulação de fechaduras eletrônicas com base em cenários IoT, além disso, disponibiliza a funcionalidade de reconhecimento facial para destranca da fechadura. Na Figura 9 é apresentado a sequência de etapas que define o percurso dos dados.

Figura 9 – Fluxograma do Sistema FaceLock.



Fonte: Elaborado pelo próprio autor

Realizando à análise por ordem numérica e percorrendo o fluxo da esquerda para direita, temos a primeira análise em (1). Usuários que vão de *1* à *"n"*, uma quantidade não definida, mas que teja no limite do sistema, tentam conectar-se diretamente a *interface* do sistema

Web através de um celular ou computador com acesso à Internet.

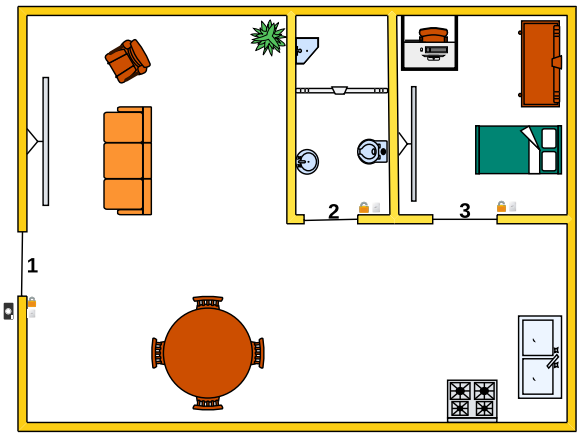
Em (2), há um *dashboard* do sistema Web capaz de fornecer o cadastro de usuários ou *login* de usuários já cadastrados. Caso a autenticação tenha êxito, o usuário consegue ter acesso as suas fechaduras cadastradas ou se não, cadastrar novas fechaduras, utilizando o (2.3) para fazer os registros dos dados. Além disso, existe uma API Câmera (2.1), responsável por enviar ao sistema Web imagens dos usuários sempre que um rosto na câmera for identificado. Continuando esse fluxo, o sistema Web permite que você cadastre uma ou três imagens no *dashboard* do sistema, porém, o cadastro depende da reposta vindo da API SkyBiometry (2.3), onde para poder acontecer o reconhecimento facial das imagens recebidas por (2.1). Então, quando o usuário tem cadastrado sua imagem na API (2.3), essa imagem é treinada pela própria API e no (2.4) armazenado os dados necessários para registro e consultas das informações de cadastro da pela aplicação. Após esse último processo, agora à aplicação Web consegue enviar a imagem obtida de (2.1) para (2.3) e por fim alterar os status da fechadura dependendo do passo (3).

Na etapa (3), a solicitação é feita pela aplicação Web, e o (3), representado pelo fluxo do *middleware* FIWARE, processa essa solicitação e encaminha para a etapa (4), o sistema embarcado, onde tem uma *smart lock*. Após a fechadura receber essa requisição ela efetua a ação desejada, tranca ou destranca, e então encaminha para o (3) a sua ação para que o sistema Web (2) obtenha os status dessa fechadura, caso tenha êxito nessa etapa, ele registra os status. Sendo assim, tornando um cenário IoT, onde tem usuários tentando conectar-se a sua *Smart Home* com uma *smart lock*. Podemos perceber também que há sempre comunicação nos dois sentidos, do FIWARE com os sistemas, isso refere-se a troca de informoçãos entre os dispositivos nele gerenciado.

5.2 Smart House

Na figura 10, apresenta uma imagem ilustrativa de um cenário com uma *Smart Home*, onde a casa tem 3 portas, identificada pelos números (1, 2, 3) e nessas portas é instalado uma FACELOCK, em que apenas na porta (1) é instalado com a câmera, essa porta terá acesso ao recurso de reconhecimento facial, enquanto os outros serão apenas controlados pelo sistema Web.

Figura 10 – Exemplo de Smart Home com SmartLock.



Fonte: Elaborado pelo próprio autor

5.3 Protótipo Embarcado

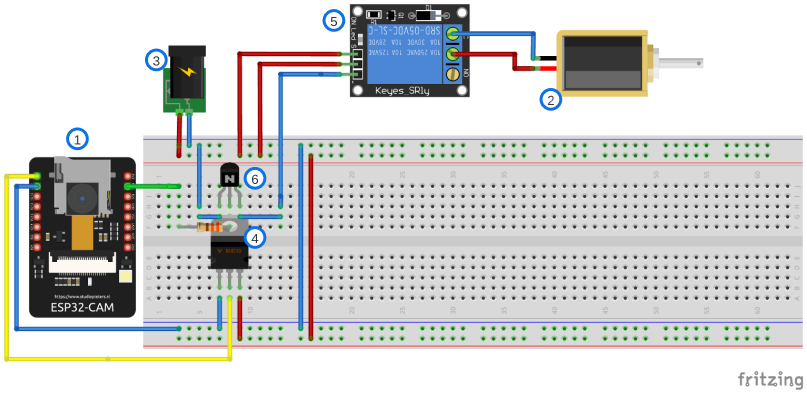
Para desenvolver o protótipo da *smart lock* proposta, foi necessário listar e definir quais componentes necessários para o funcionamento da fechadura, além disso, também foi elaborado um esquema de como poderia ficar organizado os componentes escolhidos, para isso foi utilizado a ferramenta *Fritzing*, um *software open source* para modelagem de *hardware*, lá tem alguns módulos que podemos utilizar e fazer conforme o projeto. Na figura 11, apresenta uma demonstração de como ficaria o protótipo.

Na figura 11 tem vários componentes específicos que vão auxiliar no comportamento do sistema, esses componentes estão listados como:

1 Módulo Esp32Cam: será o microcontrolador programado
2 Solenóid 12V: é o atuador eletrônico capaz de trancar e destrancar.
3 Fonte 12V: será a alimentação necessária para o microcontrolador e o solenóide.
4 Regulador de tensão 7805: dispositivo necessário para regular tensão de 12V para 5V, essa tensão de 5V é estabelecida para o microcontrolador, através do seu Pino de entrada que permite entrar 5V, isso permite que o microcontrolador não necessite da entra USB para alimentação.
5 Relé de 12v, capaz de chavear o circuito.
6 Transistor NPN BC548: será o mediador, onde permitirá chavear o microcontrolador com o relé, deixando assim o sinal do pino do microcontrolador acionar o relé quando fechar o circuito.

Conforme mostrado na figura 11, foi um circuito projetado em *software*, agora na Figura 12 representa o protótipo construído, onde tem todos os componentes listados anteriormente

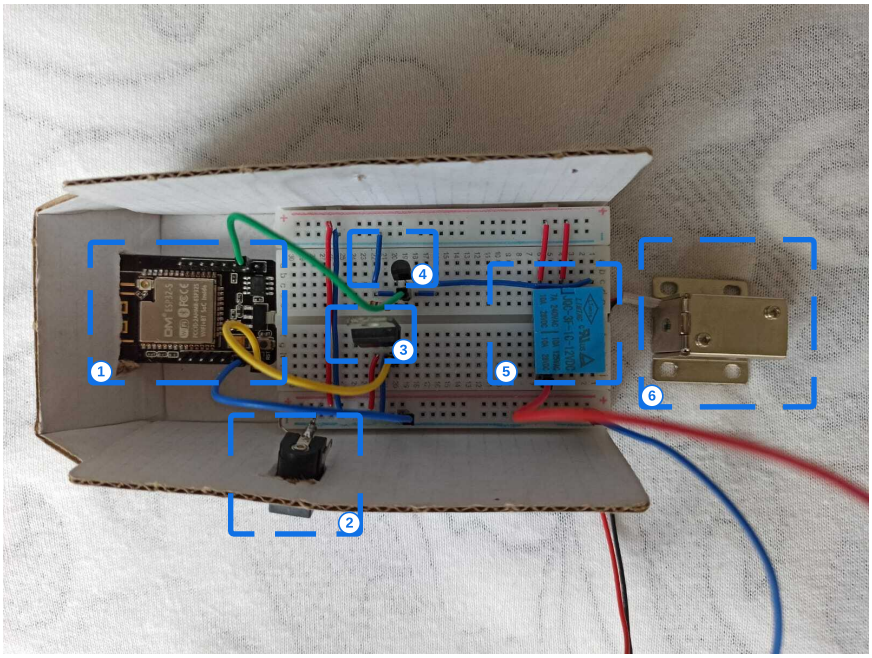
Figura 11 – Protótipo SmartLock no Fritzing.



Fonte: Elaborado pelo próprio autor

com acréscimo de mais um, seria o conector DC (direct current), representado por (2), onde será conectado a fonte de tensão de 12V. O (1) seria Esp32Cam, regulador de tensão (3), transistor (4), relé (5) e por fim o solenóide 12V (6). Todos estão conectados por uma *protoboard*, uma placa de ensaio na montagem de circuitos eletrônicos.

Figura 12 – Protótipo SmartLock construído.



Fonte: Elaborado pelo próprio autor

A implementação do algoritmo deve permitir com que o pino responsável pela emissão do sinal para fechadura ocorra, além disso, também é garantido com que o microcontrolador esteja conectado com a rede Wi-Fi e com o FIWARE, esse último, por meio do *broker* MQTT.

Sendo assim, podemos visualizar no código 1 a parte responsável por conectar ao Wi-Fi e ao dispositivo criado no FIWARE, onde será falado abordado na seção 5.4.2.

Algoritmo 1: Algoritmo para configuração do Wi-Fi e BrokerMQtt

```
1   #include <WiFi.h>
2   #include <PubSubClient.h>
3
4   #define pinFechadura1 16 //Pino da Fechadura
5
6   //WiFi
7   const char* SSID = "******";     // SSID / nome da rede WiFi que
        deseja se conectar
8   const char* PASSWORD = "******";    // Senha da rede WiFi que
       deseja se conectar
9
10  //MQTT Server
11  const char* BROKER_MQTT = "localhost"; //URL do broker MQTT que
        se deseja utilizar
12  int BROKER_PORT = 1883;              // Porta do Broker MQTT
13
14  #define TOPIC_PUBLISH "/(chave do Fiware)/(dispositivo criado
       no fiware)/attrs" //Topico para mandar status da placa
15  #define TOPIC_SUBSCRIBE "(chave do Fiware)/(dispositivo criado
       no fiware)/attrs" //Topico que recebe comandos
```

Fonte: elaborado pelo autor.

No FIWARE, as publicações executadas pelo MQTT, seguem um protocolo específico, sendo ele, o *Ultralight*, um protocolo leve, baseado em texto e destinado para controle de dispositivos. Sua sintaxe é definida como "dispositivo | comando", assim como no código 2, que exemplifica a tranca e destranca de um quarto. A mensagem pode ser verificada na linha (2) do código, que caso ocorra um comando de "fechar"na fechadura do "quarto", ocorre a tranca da mesma.

Algoritmo 2: Algoritmo no Arduino IDE para tranca e destranca da fechadura

```
1
2       if (msg == "quarto|fechar") { \\msg é a mensagem recebida no tó
           pico inscrito
3          digitalWrite(pinFechadura1, LOW); \\mandando sinal para
              desligar o pino da fechadura
4          MQTT.publish(TOPIC_PUBLISH, "quarto|trancada"); \\comando
              para indicar que esta trancada
5       }
6
7       if (msg == "quarto|abrir") {
8          digitalWrite(pinFechadura1, HIGH); \\mandando sinal para
              ligar o pino da fechadura
9          MQTT.publish(TOPIC_PUBLISH, "quarto|destrancada"); \\comando
               para indicar que esta destrancada
10      } recebe comandos
```

Fonte: elaborado pelo autor.

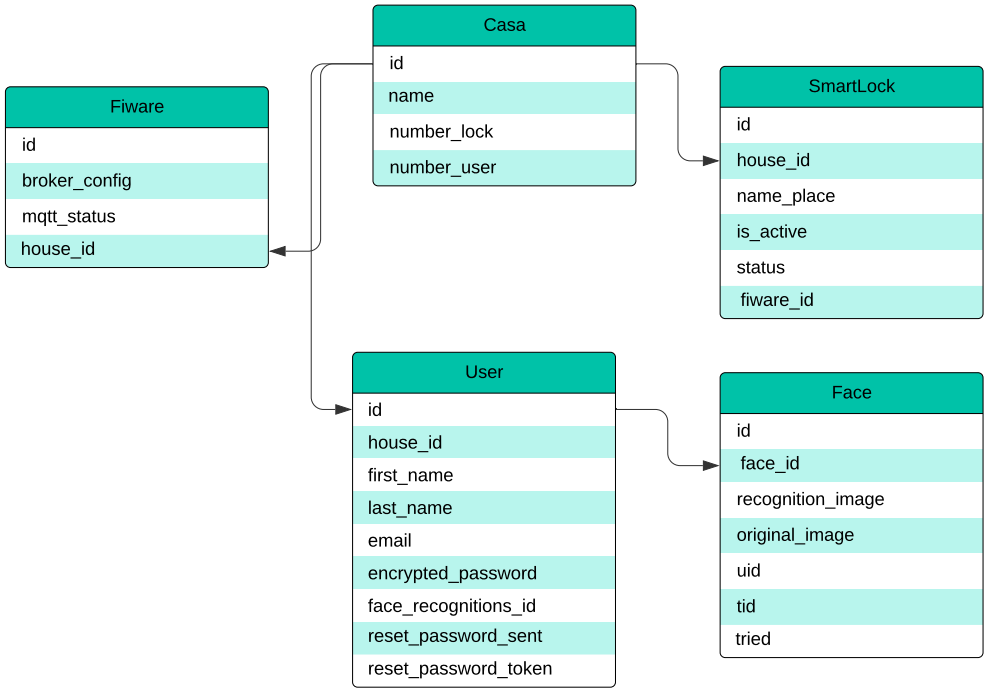
5.4 Protótipo Web

O protótipo desenvolvido tem impacto direto no contato com o usuário, permitindo assim, o gerenciamento de todo o fluxo da FACELOCK. O sistema Web possibilita o registro de usuários, casa, fechaduras e imagens utilizadas para o reconhecimento facial utilizando o banco de dados PostgreSql. A opção por usar esse banco se dá por ser opensource, afinidade com sintaxe SQL, apesar do framework facilitar as buscas de informações estabelecidas no banco de dados, além de que, tem uma facilidade na sua configuração, o código pode ser reaproveitável para outras arquiteturas, as relações estabelecidas entre tabelas podem ser replicadas, e a segurança de dados com a performance, torna-o essencial para o projeto.

Na Figura 13, é apresentado as tabelas criadas no banco de dados e suas relações. Cada tabela tem sua respectiva funcionalidade e importância para o projeto. Como, a tabela ***User*** será direcionada para registo de dados dos usuários, onde detém de uma alta segurança para constituído pela *Gem Devise*, uma biblitoca que fornece gerenciamento de usuários, com criptografia e autenticação contida, além de gerenciar acessos restritos e não autorizados, basta inserir no projeto e configurar ela conforme a documentação. A tabela ***Casa*** é responsável por registrar os usuários de uma casa, além disso, identificar quais fechaduras estão registradas na casa e quais fiwares estão sendo configurados na casa. A tabela ***SmartLock*** é onde se encontra o registro das fechaduras para determinada casa e o controle de status da mesma, sabendo se está trancada ou destrancada. A tabela ***Fiware*** é onde temos contato diretamente com os status do

servidor, para que sempre que quisermos alterar o status ou consultar o servidor *Broker* MQTT, seja registrado para o usuário saber quando ou não está ativo. Por último, a tabela ***Face***, que age diretamente com o registro das informações registradas da API SkyBiometry e da API Câmera, responsáveis pelo reconhecimento facial, além de que, mantém a relação direta com a tabela *User* para que um usuário tenha mais de uma face cadastrada.

Figura 13: Relação do Banco de dados.

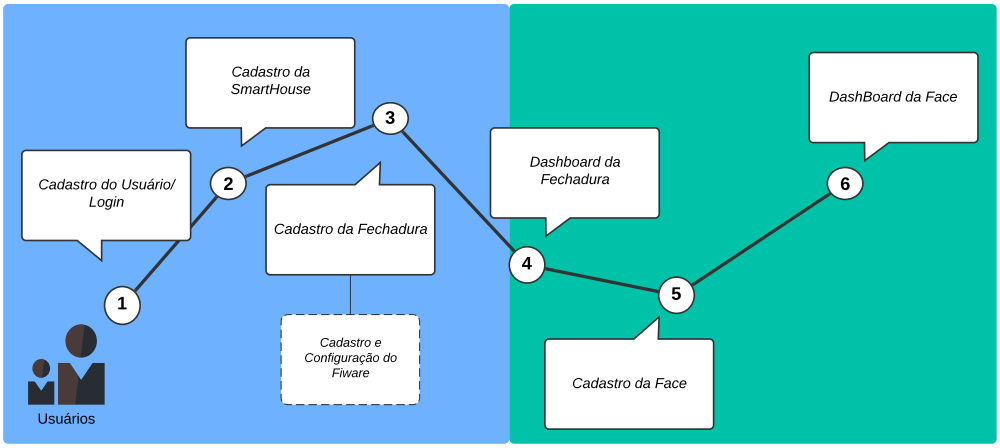


Fonte: Elaborado pelo próprio autor

Existe também algumas funcionalidades que possibilitam a comunicação com o FIWARE, essas ações são disponibilizadas pelo dashboard do sistema, para que os usuários consigam manipular suas fechaduras de forma remota. Analisando a Figura 14, podemos entender melhor o fluxo do Sistema Web e como o usuário passa por cada etapa, a seguir, será retratado cada etapa de como foi implementado esse fluxo.

As etapas apresentadas na Figura 14, mostra como o usuário pode interagir com a aplicação Web e nas subseções abaixo, abordará um pouco mais a fundo sobre a implementação de cada etapa.

Figura 14: Fluxo do usuário no Sistema Web



Fonte: Elaborado pelo próprio autor

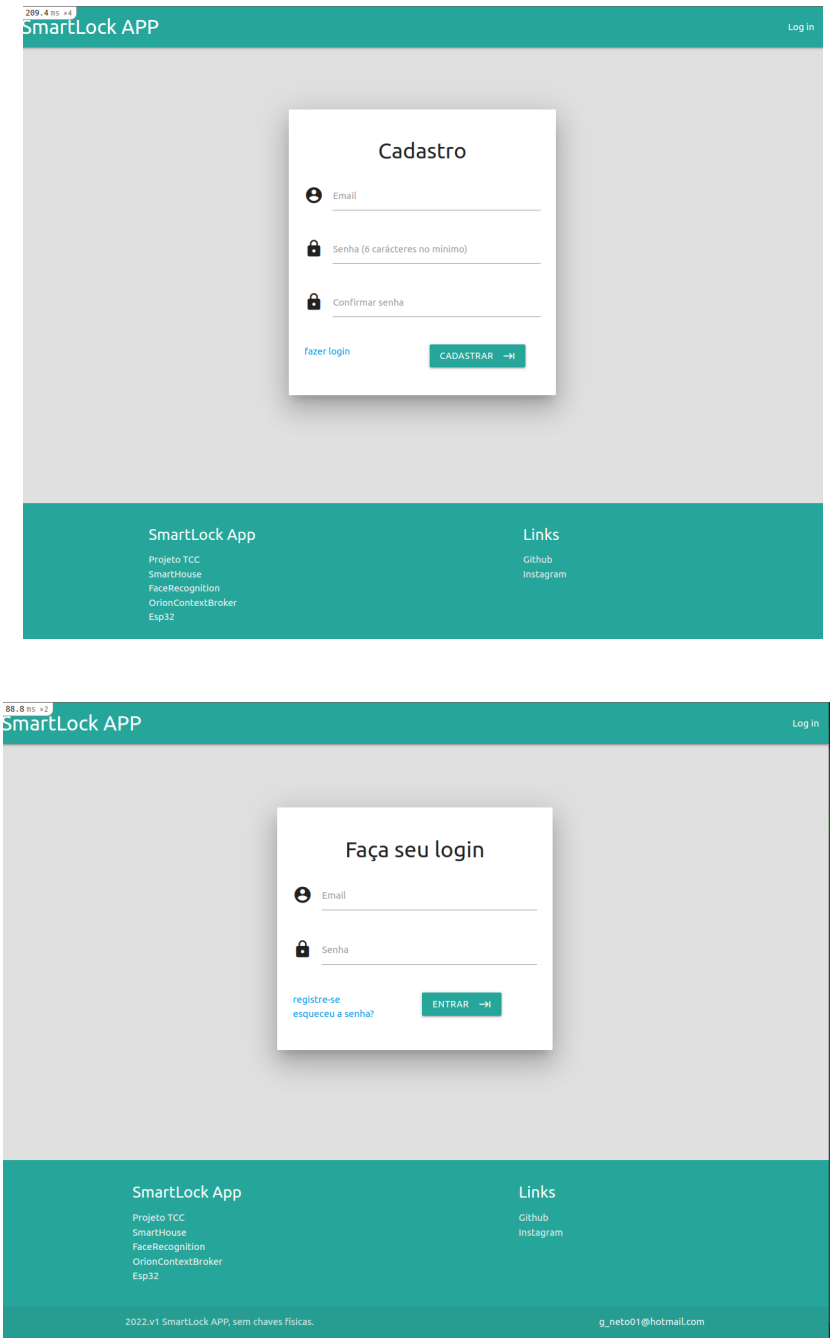
5.4.1 Cadastro de Usuário/Login

A aplicação permite com que o usuário registre-se por e-mail e senha. Esse cadastro pode ser feito por qualquer pessoa que deseja utilizar a aplicação e tenha uma fechadura instalada na sua casa, caso contrário, não conseguirá utilizar a aplicação de maneira adequada, no caso do projeto, está disponível apenas para *localhost*, ou seja, utilidades apenas no servidor localmente. O registro da aplicação pode ser feito após o usuário ter o primeiro contato, que seria a tela *home* ou principal, como demonstra a imagem 15, você pode efetuar a entrada do sistema ou pode registrar o usuário, para seguir com o registro de um novo usuário, você clicar em “começar agora”, às duas telas para cadastro do usuário e *login*, são representados pela imagem 16.

Figura 15: Tela Principal

Fonte: Elaborado pelo próprio autor

Figura 16: Cadastro e Login do usuário



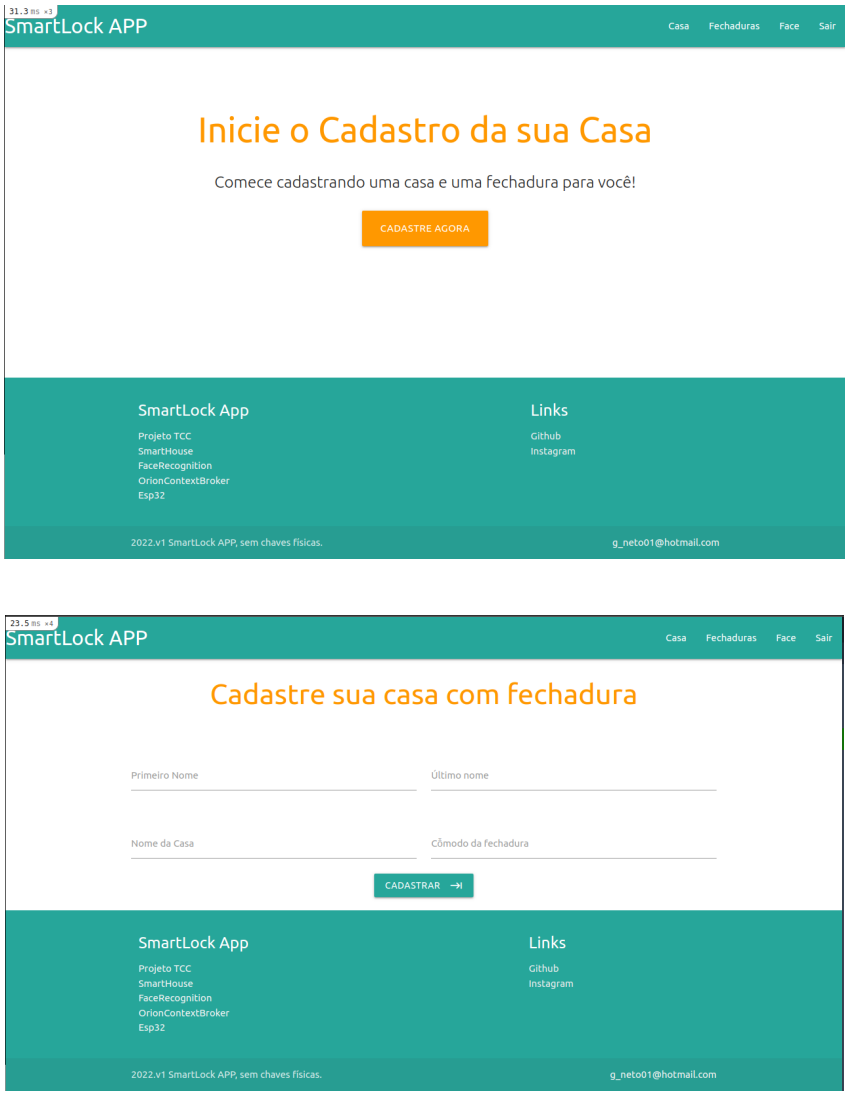
Fonte: Elaborado pelo próprio autor

5.4.2 Cadastro da Smart Home com uma fechadura

Como apresentado na figura 14, a etapa (2) inclui também o passo (3), visto que para o funcionamento da *Smart Home* é necessário ter pelo menos 1 fechadura cadastrada. Então é iniciado o cadastro no banco a partir dos dados fornecido pelo usuário. O usuário fornece os dados solicitados nas telas, você pode verificar na imagem 17 de cadastro da casa com

fechadura, assim, permitindo montar o cenário IoT, em que terá fechaduras, usuários e FIWARE relacionados, assim facilitando o gerenciamento do sistema.

Figura 17: Cadastro da Smart House



Fonte: Elaborado pelo próprio autor

O FIWARE, é configurado também nessa etapa pelo *Back-end* do sistema, onde é inicializado os serviços necessários para estabelecer a comunicação entre o *Orion Context Broker* e os dispositivos conectados ao *Broker* MQTT Mosquitto pelo *IoTAgent*. Para iniciar, é necessário configurar o IoTAgent para conseguir inscrever-se em um tópico em que os dispositivos possam se comunicar. O *IoTAgent* irá atuar como um *middleware* entre esses dispositivos IoT e o agente de contexto, no caso, as mensagens do sistema Web. Portanto, ele necessita criar entidades de dados de contexto com IDs (identificadores exclusivos). A configuração do *middleware* para troca de contexto é feita no código 3 por requisições REST, que possibilitam enviar para o

servidor do FIWARE executando localmente, essa requisição é feita através do método POST (linha 7) e verificada se foi um êxito por meio do método GET (linha 28).

Algoritmo 3: Algoritmo em RoR para executar configuração do *Brokker* e *Iot Agent*

```
1  module IotAgent
2    class BrokerConfig
3      @@url = IotAgent::Common.url + "/services" //url em que o
          servidor Fiware ta rodando
4      @@headers = IotAgent::Common.headers('application/json') //
          contexto da requisição feita em json
5      @@key = IotAgent::Common.key
6      def self.post
7        path = @@url
8        data = {
9          "services": [
10           {
11             "apikey":      @@key,               //tópico para
                  comunicação
12             "cbroker":     "http://orion:1026", //endpoint a qual
                  será passado o contexto
13             "entity_type": "Thing",
14             "resource":    "/iot/json"          //tipo de mensagem
15           }
16         ]
17        }
18       begin
19         response = RestClient.post(path, data.to_json, @@headers)
              //requisição para configurar o broker
20
21         Common.format_response(response)
22       rescue RestClient::ExceptionWithResponse => e
23         Common.format_error(e.response)
24       end
25     end
26
27     def self.get
28       path = @@url
29       begin
30         response = RestClient.get(path, @@headers)
31         Common.format_response(response)
32       rescue RestClient::ExceptionWithResponse => e
33         Common.format_error(e.response)
34       end
35     end
36   end
37 end
```

Fonte: elaborado pelo autor.

Após a configuração estabelecida, é necessário criar uma entidade que suporte vários

dispositivos, então quando é cadastrado a fechadura, ela é cadastrada em uma entidade criada com nome *smart lock* concatenado com o *id* da fechadura registrada no banco, essa entidade é responsável por guardar o contexto e as trocas de mensagens entre os dispositivos, podemos visualizar essa configuração no código 4 , além disso, também é criado um *devise* (dispositivo) para com o nome registrado para a fechadura, podemos perceber na (linha 20) do código.

Algoritmo 4: Algoritmo para criação da entidade smart lock e do dispositivo no *Iot Agent*

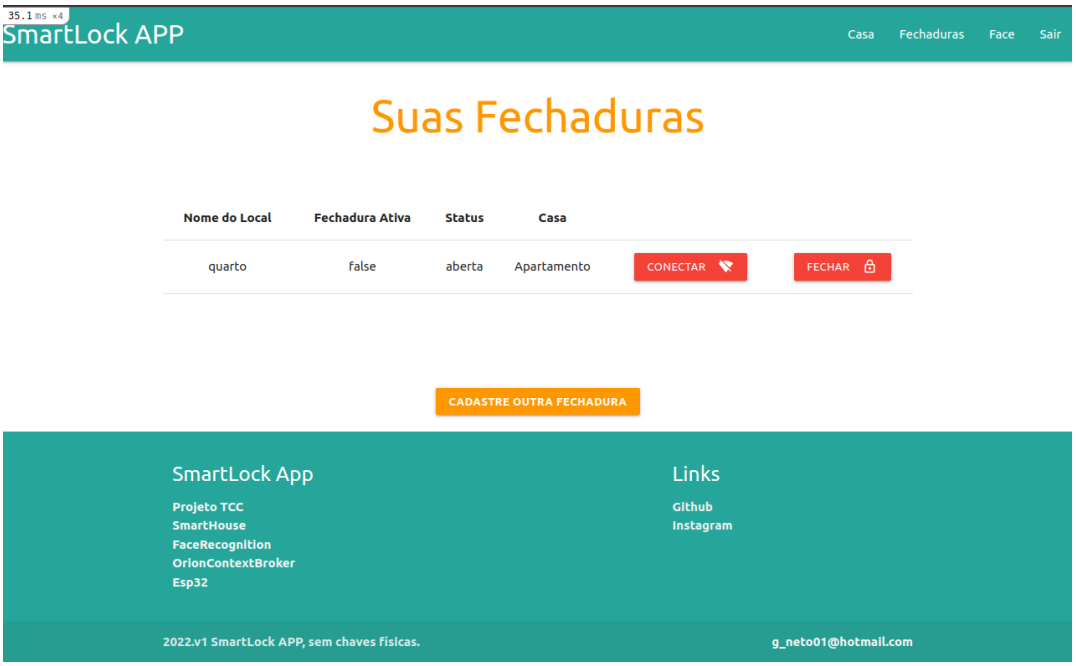
```
1
2  module IotAgent
3    class Entity
4      @@url = IotAgent::Common.url + "/devices"
5      @@header_txt = IotAgent::Common.headers('text/plain')
6      @@header_json = IotAgent::Common.headers('application/json')
7
8      def self.create_device(smart_loock)
9        path = @@url
10       data = {
11         "devices": [
12           {
13             "device_id":   "smartlock#{smart_loock.id}",
14             "entity_name": "urn:ngsi-ld:Ecg:001",
15             "entity_type": "Ecg_entidade",
16             "protocol":    "PDI-IoTA-UltraLight",
17             "transport":   "MQTT",
18             "timezone":    "America/Fortaleza",
19             "attributes": [
20               { "object_id": "Ecg", "name": "#{smart_loock.
                     name_place}", "type": "String"}
21             ]
22           }
23         ]
24        }
25       begin
26         response = RestClient.post(path, data.to_json,
                @@header_json)
27
28         IotAgent::Common.format_response(response)
29       rescue RestClient::ExceptionWithResponse => e
30         IotAgent::Common.format_error(e.response)
31       end
32     end
33   end
34 end
```

Fonte: elaborado pelo autor.

5.4.3 Dashboard das Fechaduras

A imagem 18 representa como as fechaduras podem ser controladas pelo sistema, após registrar uma fechadura ela iá mostrar alguns atributos e permitir que faça duas 4 ações. Os atributos exibidos é o nome da fechadura, o local onde se encontra, o status que inclui duas ações (aberta ou fechada) e a Casa onde foi registrada. Também tem a disponibilidade de dois botões, sendo o primeiro para conectar e desconectar com o *Broker* e o outro para destrancar ou trancar a fechadura.

Figura 18: Dashboard para gerenciar fechadura e conexão com o Broker



Fonte: Elaborado pelo próprio autor

5.4.4 Cadastro da Face

Etapa (5) e (6) do fluxo seguido pelo usuário na figura 14, andam bem atreladas. O cadastro do rosto ou face do usuário é feito por um upload de uma imagem do usuário, no sistema Web é permitido que possa ser cadastrado 3 imagens no máximo, sendo elas do mesmo usuário ou não, pois registradas para permitirem a destranca da fechadura quando reconhecidas. O cadastro é feito apenas com o upload da imagem.

Após ser enviado a imagem, o *Back-end* e a API SkyBiometry serão responsáveis por registrar esses dados, tanto na aplicação Web quanto na API. Para isso, o sistema envia uma

requisição autenticada para um *endpoint* específico para executar as seguintes etapas.

1 Cadastro na SkyBiometry: é preciso fazer o cadastro na API e criar uma aplicação, para poder ser executado todos os passos de reconhecimento facial através de imagens, além de que, deve criar um *namespace*, esse no que lhe concerne, é responsável por agrupar as imagens detectadas e treinadas.

2 Detectar a imagem: o upload da imagem é feito e limitado pela GEM Minimagick, uma biblioteca que manipula e faz o processamento de imagens, isso para não enviar para API um tamanho de imagem que não suporte, pois, na documentação exige no máximo 2mb. Quando essa imagem é detectada, a API retorna um status HTTP 200 e o **tid**, que indica que foi um sucesso e esse **tid** é o identificador da imagem detectada, que servirá para próxima etapa, caso de não funcionar, será retornado o erro para o sistema e mostrado para o usuário.

3 Criação de Tag: a criação de **tag** é uma das principais funcionalidades da SkyBiometry, por meio dessas tags ela consegue saber a qual imagem está fazendo referência, pois ela permite receber o **tid** detectado, além disso, ela armazena no seu banco para você conseguir ter acesso a essa tag. Outro parâmetro que ela necessita, é o **uid**, que você passa como identificador próprio, ele pode ser qualquer nome que você estabelecer, portanto, foi utilizado o nome do usuário concatenado com o id da face registrada no banco, isso para facilitar na hora da busca dos dados e registro no banco de dados.

4 Treinamento da imagem: após cadastrar a imagem, é feito o treinamento, a API SkyBiometry, utiliza de algoritmos de treinamentos sofisticados para fazer o treinamento, possibilitando com que você tenha mais sucesso ao tentar reconhecer um rosto.

5 Reconhecimento Facial: após feito todos os procedimentos anteriores, para ocorrer o reconhecimento dos rostos em uma imagem, a API SkyBiometry disponibiliza *endpoint* que reconhece o rosto do usuário treinado, basta passar a imagem a qual quer reconhecer e qual uid cadastrado você deseja comparar. Dessa forma, a imagem é passada pela API Câmera, responsável pelo capturamento de imagem obtida de uma *webcam* e envia diretamente para aplicação, a aplicação recebe essa imagem e o uid cadastrado no banco de dados, logo em seguida encaminha para API Skybiometry para fazer o reconhecimento facial e retorna a confiança, o principal parâmetro analisado, que estabelecemos 80 para a identificação da imagem reconhecida, porém retorna outros parâmetros, como localização do rosto, boca, nariz, etc.

No código 5 demonstra o código de configuração para requisições em REST para API SkyBiometry, onde faz diretamente para a identificação de imagem, onde utiliza o "/faces"como principal *endpoint*, já no código é referente ao salvamento da tag, que utiliza o *endpoint* "/tags".

Algoritmo 5: Algoritmo em ruby para comunicação no endpoint /face da API SkyBiometry

```ruby
module SkyBiometryApi
  class Faces
    @@url = SkyBiometryApi::Common.url + "/faces/"
    @@key = SkyBiometryApi::Common.keys

    def self.detect(files:)
      path = @@url + "detect.json?" + @@key.to_query
      data = {files: files.open}
      begin
        response = RestClient.post(path, data)
        SkyBiometryApi::Common.format_response(response)
      rescue RestClient::ExceptionWithResponse => e
        SkyBiometryApi::Common.format_error(e.response)
      end
    end

    def self.train(uids:)
      data = {uids: uids,}
      path = @@url + "train.json?" + @@key.to_query + "&#{data.
          to_query}"

      begin
        response = RestClient.post(path, data)
        SkyBiometryApi::Common.format_response(response)
      rescue RestClient::ExceptionWithResponse => e
        SkyBiometryApi::Common.format_error(e.response)
      end
    end

    def self.recognize(files:, uids:)
      path = @@url + "recognize.json?" + @@key.to_query
      data = {uids: uids,files: files.open,multipart: true}
      begin
        response = RestClient.post(path, data)
        SkyBiometryApi::Common.format_response(response)
      rescue RestClient::ExceptionWithResponse => e
        SkyBiometryApi::Common.format_error(e.response)
      end
    end
  end
end
```

Fonte: elaborado pelo autor.

Algoritmo 6: Algoritmo em ruby para comunicação no endpoint /tags da API SkyBiometry

```ruby

module SkyBiometryApi
  class Tags
    @@url = SkyBiometryApi::Common.url + "/tags/"
    @@key = SkyBiometryApi::Common.keys

    def self.save(tids:, uid:)
      data= {}
      tid = tids
      uids = uid
      path = @@url + "save.json?" + @@key.to_query + "&uid=#{uids}&tids=#{tid}"
      puts path
      begin
        response = RestClient.post(path, data)
        SkyBiometryApi::Common.format_response(response)
      rescue RestClient::ExceptionWithResponse => e
        SkyBiometryApi::Common.format_error(e.response)
      end
    end

    def self.get(uid:)
      data = {
        uids: uid
      }
      path = @@url + "get.json?" + @@key.to_query + "&#{data.to_query}"

      puts path
      begin
        response = RestClient.get(path)
        SkyBiometryApi::Common.format_response(response)
      rescue RestClient::ExceptionWithResponse => e
        SkyBiometryApi::Common.format_error(e.response)
      end
    end
  end
end
```

Fonte: elaborado pelo autor.

5.5 API Câmera

A API Câmera, foi criada com o intuito de simular uma câmera pré instalada na porta. O objetivo dessa API, é captar imagens do usuário em tempo real e enviar para a aplicação. O envio dessas imagens só é permitido quando ocorrer a detecção do rosto do usuário na câmera,

após reconhecer um rosto, ela faz a requisição para o servidor da aplicação. No projeto, foi utilizado webcam do notebook, ela tem melhor qualidade, e flexibilidade para testes, ao contrário da câmera do microcontrolador, então após ser detectada a imagem a aplicação segue o fluxo 5.4.4 para reconhecimento da imagem obtida. O código da API Câmera, pode ser observado no código 7 feito em *python* e utilizando a biblioteca OpenCV, ambos *open source* e permitem fazer processamento de imagem e requisições.

A função do código 7, é responsável por enviar o arquivo em formato jpg para a aplicação desenvolvida em Ruby on Rails, por métodos REST HTTP, importando uma biblioteca denominada 'requests', permitindo o envio para outros servidores por métodos GET's e POST's, além disso, a função definida, também deve indicar pelos parâmetros enviados, qual fechadura pelo id, qual o usuário com acesso à fechadura pelo e tem as imagens cadastradas. Na linha (18) é aberto a câmera e o restante do procedimento é utilizado a biblioteca OpenCV. A ideia por trás dessa API, é fazer com que ela interaja independente, fazendo com que cada fechadura possua uma API rodando com uma câmera e fazendo requisições para a aplicação.

Algoritmo 7: Algoritmo em python para enviar imagem à aplicação

```
1 import requests
2
3 def sendtoserver(frame): //servidor da aplicação que vai receber a
     imagem.
4   imencoded = cv2.imencode(".jpg", frame)[1]
5   file = {'file': ('image.jpg', imencoded.tostring(), 'image/jpeg',
        {'Expires': '0'})}
6   data = {"id" : "12", "user_id" : "3" }
7   response = requests.post("http://servidor_rails", files=file,
       data=data, timeout=5)
8
9   return response
```

Fonte: elaborado pelo autor.

Algoritmo 8: Algoritmo em python para capturar rostos da webcam

```
1
2  from urllib import request
3  import cv2
4  import requests
5  import time
6
7  if webcam.isOpened():
8    cam_is_active, frame = webcam.read()
9
10   while cam_is_active:
11     cam_is_active, frame = webcam.read()
12     gray                 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
13     faces                = face_cascade.detectMultiScale(gray, 1.1,
           4)
14     image_original       = frame.copy()
15
16     for (x, y, w, h) in faces:
17       cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
18
19     cv2.imshow("Video da Webcam", frame)
20     key = cv2.waitKey(5)
21
22     if key == 27: # ESC
23       break
24     len(faces): //detecção dos frame do rosto do usuário
25     sendtoserver(image_original) // envia a imagem para o servidor
26     time.sleep(10)
27
28 webcam.release()
29 cv2.destroyAllWindows()
```

Fonte: elaborado pelo autor.

6 EXPERIMENTOS E RESULTADOS

Neste trabalho foram realizados experimentos para verificar o comportamento da solução com diferentes cenários. Os experimentos foram conduzidos no sentido de testar a capacidade de detecção de rostos dos usuários, reconhecimento da imagem e o tempo de resposta entre cada dispositivo do sistema FACELOCK. Para isso, foi considerado a arquitetura montada por todo o sistema e suas principais funcionalidades.

6.1 Configuração dos Experimentos

O Quadro 2 apresenta de forma resumida os experimentos executados para o sistema proposto. Para a geração dos resultados, foi utilizado as próprias implementações para simular ou apresentar dados reais obtidos do sistema. O fluxo e comportamento do sistema também é considerado para os experimentos, desde a configuração do sistema Web até chegar no sistema Embarcados.

Quadro 2: Descrição dos Experimentos

Critérios	**Descrição**
Sistema	Infraestrutura baseada no *Framework Ruby on Rails* e no sistema embarcado
Métricas	Reconhecimento facial e imagem, tempo de resposta
Parâmetros	quantidade de imagens, requisições, confiança e tempo de resposta.
Fatores	Ambiente, oscilação na rede, respostas das APIs
Projeto de Experimentos	**Experimento 1:** reconhecimento de rostos por meio da API Câmera, com várias imagens simuladas e algumas reais. **Experimento 2:** reconhecimento facial por meio do sistema completo, através de dados registrados; **Experimento 3:** verificação do tempo de resposta da comunicação dos dispositivos.

Fonte: elaborado pelo autor.

6.1.1 Configuração do experimento 1

Os testes realizados em escala, foram com dados simulados, pois não há como obter várias faces de usuários para consolidação dos experimentos, então foram registrados para simulação, 100 imagens, dentre essas imagens, 50 imagens com iluminação boa, como em (a) da figura 19 e de rostos aleatórios, 25 imagens de rostos aleatórios com ambiente com iluminação parcial (b) e as outras 25 com iluminação baixa (c) para o primeiro experimento. Também, para o mesmo experimento, foi realizado uma amostra com cinco pessoas utilizando a webcam de um notebook para detecção de rosto em ambientes de iluminação diferentes, sendo eles, dia, tarde e noite. O primeiro experimento é configurado no quadro 3.

Figura 19: Dashboard para gerenciar fechadura e conexão com o Broker

(a) Iluminação boa **(b) Iluminação parcial** **(c) Iluminação baixa**

Fonte: Imagens ilustrativas do google

Quadro 3: Configuração experimento 1 Utilizando a API Câmera

Ambiente	**Quantidade de imagens**	**Quantidade de imagens pela web-Cam**
Iluminação boa/manhã	50	5
Iluminação parcia/Tarde	25	5
Iluminação baixa/Noite	25	5

Fonte: elaborado pelo autor.

6.1.2 Configuração para experimento 2

Para o experimento 2, foi registrado 3 fechaduras, sendo a primeira, uma fechadura real, a qual foi construída neste projeto, e outras duas, serão duas fictícias, apenas registradas para simulação. Além das fechaduras, foram registrados 3 usuários, e cada usuário tem uma quantidade de imagens registradas para análise do experimento. O usuário 1, tem a fechadura real com três imagens cadastradas para reconhecimento facial, o usuário 2 tem uma das fechaduras fictícias e 2 imagens registradas, o usuário 3 tem a última fechadura fictícia e apenas uma imagem registrada para o reconhecimento, podemos verificar esses dados no seguinte quadro 4.

Quadro 4: Configuração experimento 2 utilizando sistema Facelock

Usuários	**Fechaduras**	**Quantidade de imagens cadastradas**
1	fechadura real	3
2	fechadura fictícia 1	2
3	fechadura fictícia 2	1

Fonte: elaborado pelo autor.

6.1.3 Configuração experimento 3

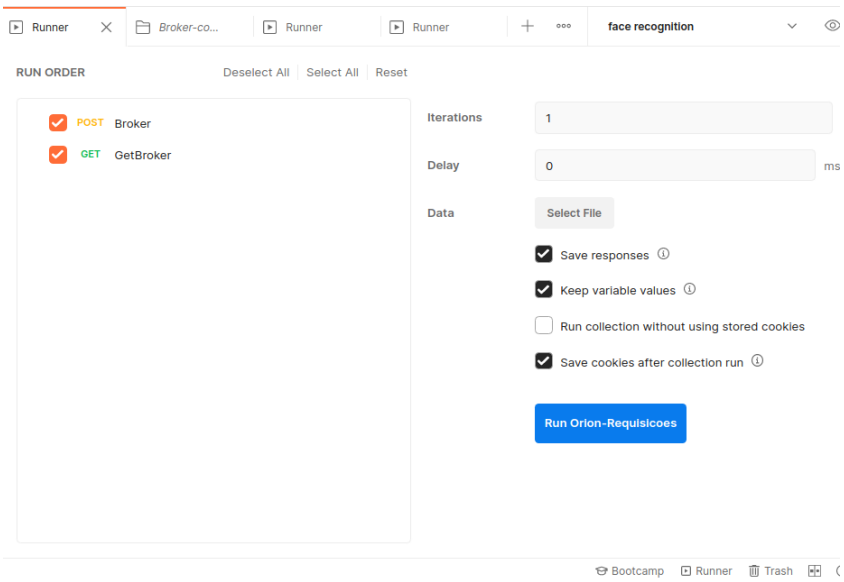
Para o experimento 3, foi utilizado uma ferramenta Open Source, Postman, é um API *Client* que ajuda na criação, compartilhamento, testes e documentar APIs. É feito, com salvamentos de requisições HTTP e HTTPs simples e complexas, para enviar e ler respostas, por meio dos métodos HTTP. Com isso, foi utilizado a ferramenta *Runner*, que agrupa uma série de requisições para fazer uma quantidade desejada como visto no exemplo da imagem 20, que você pode escolher as requisições feitas e setar o número de interações. O Runner permite que você execute conjuntos de solicitações em uma sequência especificada. Além de que, Runner registrará seus resultados de teste de solicitação, se desejar, como na imagem 20, tem a opção "save response". Com a utilização do Postman, é possível mapear o tempo de resposta de cada requisição feita pelas APIs. Dessa forma, foram mapeadas como no quadro 7, o quanto de requisições foram feitas para cada funcionalidade importante, API Câmera, API SkyBiometry e o FIWARE.

Quadro 5: Configuração para experimento 3 para tempos de respostas

	Número de requisições	**GET**	**POST**	**PUT**
API Câmera	100	0	100	0
API SkyBiometry	200	100	100	0
Fiware	100	20	40	40

Fonte: elaborado pelo autor.

Figura 20: Runner da ferramenta Postman



Fonte: elaborado pelo autor.

6.2 Resultados dos experimentos

Para observar os resultados obtidos pelo sistema FACELOCK, em cada experimento realizado foi gerado quadros e um gráfico com as análises feitas, descrevendo o real resultado gerado pelo sistema. Além disso, é apresentado algumas imagens dos resultados reais da aplicação, como a fechadura aderida na porta, o rosto reconhecido na imagem, o cadastro de uma imagem para reconhecimento facial e a detecção real da última pessoa que tentou o acesso à fechadura podemos averiguar na subseção 6.2.4.

6.2.1 Experimento 1

No experimento do quadro 6, foram armazenados um *Dataset* com 100 imagens aleatórias de pessoas, essas imagens foram adicionadas localmente, onde em 50 dessas imagens, foi aplicado um filtro que diminuísse o brilho das imagens em 50% e 25%, gerando assim imagens com dificuldade de reconhecimento do rosto para ambientes diferentes. Além de que, foi utilizado 5 usuários reais para interagirem dinamicamente com a *webcam* para identificar o seu rosto por meio da câmera em tempo real. Esses cenários, foram averiguados para diferentes cenários.

Ainda no qaudro 6, podemos identificar da esquerda para direita, que, na coluna (1), o ambiente em que se encontra as amostras, essas amostras é dado pelas imagens em análise ou os usuários que participaram do experimento com a câmera em tempo real, simbolizadas respectivamente pelas colunas (2) e (3). Já na coluna (4) e (5), são os resultados obtidos por cada face analisada, como, por exemplo, para uma boa iluminação, em 50 imagens simuladas e 5 usuários utilizando durante o período da manhã, foi identificada 100% dos rostos do usuário. Porém, pelo período da tarde, para imagens simuladas, um dos rostos ficou bem mais escuro do que devia e acabou não sendo reconhecido, porém, os usuários reais ainda sim, foram reconhecidos. No período da noite, foi onde teve mais rostos não identificados, sendo no total de 17 imagens não simuladas e 4 usuários reais, isso ocorreu, devido à falta de iluminação, o algoritmo utilizado para reconhecer os rostos, não aplicam filtro para estes casos, tornando-o mais dificultoso, além de que um ambiente mal iluminado realmente é difícil de identificar faces.

Quadro 6: Resultado Experimento 1

Ambiente	**Quantidade de imagens**	**Quantidade de imagens pela webCam**	**Face Indentificada**	**Face não Indentificada**
Iluminação boa/manhã	50	5	50 e 5	0 e 0
Iluminação parcia/-Tarde	25	5	24 e 5	1 e 0
Iluminação baixa/-Noite	25	5	8 e 1	17 e 4

Fonte: elaborado pelo autor.

6.2.2 Experimento 2

No experimento 2, já envolve o sistema FACELOCK completo, desde os usuários cadastrados até as faces desses usuários cadastradas e treinadas pela API SkyBiometry. Dessa forma, no quadro 4 apresenta os dados correlacionados e analisados pelos resultados obtidos. O seguinte experimento, mostra os resultados das tentativas de utilizar o reconhecimento facial para abrir a fechadura de uma porta. Como podemos perceber, é estabelecido algumas colunas para análise.

coluna 1: referência ao usuário cadastrado no sistema com o tipo de fechadura, no caso do usuário 1, ele utiliza a fechadura do protótipo e os outros as fechaduras fictícias.

coluna 2: referir-se a quantidade de imagens cadastradas no sistema para reconhecimento facial da imagem, nesse caso, cada usuário tem uma quantidade, o usuário 1 tem três faces cadastradas, todas distintas, possibilitando 3 pessoas diferentes abrir a porta, o usuário 2 tem duas imagens do mesmo usuário faces e o usuário 3 tem apenas uma.

coluna 3: é o número de tentativas para reconhecimento facial. Para o usuário 1, 30 são tentativas com usuário reais para o protótipo e as outras 10 são sempre com fotos aleatórias. Para o usuário 1 e 2, também segue o mesmo pensamento, sendo 20 e 10 respectivamente para tentativas com rostos registrados e 10 e 10 para rostos aleatórios da internet.

coluna 4: é a média das respostas das confianças de cada tentativa reconhecida pela API. A confiança maior ou igual a 80, indica que a imagem foi realmente reconhecida.

coluna 5: quantidade de rostos não reconhecidos pela API.

coluna 6: são os falsos positivos do reconhecimento pela API.

coluna 7: resposta da fechadura com relação ao seu status, caso seja aberta, a resposta foi um sucesso, se não abriu, ocorreu uma falha.

Com essas colunas definidas, podemos perceber que para o caso 1, do usuário 1, as tentativas todas foram um sucesso, onde reconheceu diferentes pessoas para a fechadura e

ainda conseguiu abrir corretamente. Para o caso 2, com o usuário 2, teve o mesmo resultado com uma confiança ainda maior por ter duas imagens da mesma pessoa cadastrada, onde teve como média 92. Já para o caso 3, do usuário 3, foi testado com uma imagem cadastrada de um gêmeo, essa imagem possibilitaria o reconhecimento de falso-reconhecimento, o intuito desse teste é averiguar como a API se comporta, sendo assim, ela se comportou reconhecendo os gêmeos da mesma forma, como podemos perceber, teve reconhecimento erroneamente, 2 deles, onde foram 2 imagens enviadas com o gêmeo, das 9 tentativas aleatórias, 2 foram com o gêmeo, assim possibilitando a abertura da fechadura e ocasionando falha de um rosto não autorizado.

Quadro 7: Resultado Experimento 2

Usuário e fechadura	**Faces cadastradas**	**Número de tentativas**	**Faces reconhecidas**	**Média das confianças reconhecida**	**Face não reconhecida**	**Face reconhecidas erroneamente**	**Resposta da fechadura por tentativa**
Usuário 1/fechadura real	3	40	30	88	10	0	sucesso(30) e falha(0)
Usuário 2/fechadura fictícia	2	30	20	92	10	0	sucesso(20) e falha(0)
Usuário 3/fechadura fictícia	1	20	10	83	8	2	sucesso(10) e falha(2)

Fonte: elaborado pelo autor.

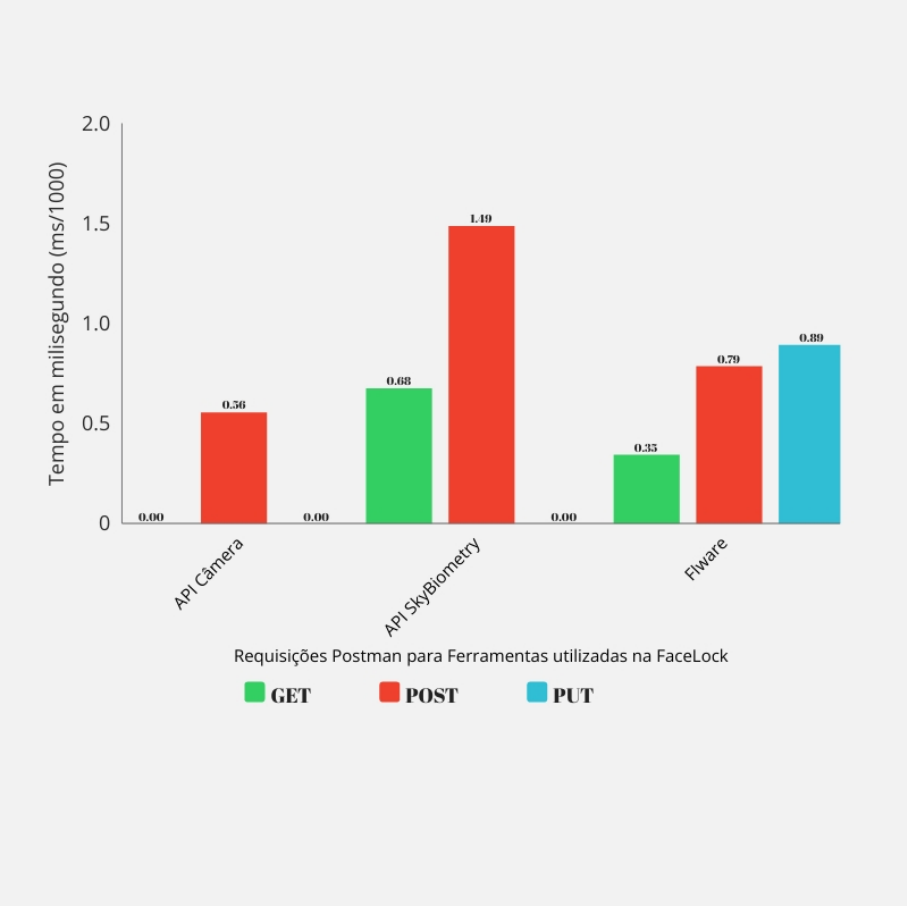
6.2.3 Experimento 3

O experimento 3, configurado em , é demonstrado por meio do gráfico 21, como as requisições se comportaram em tempo médio delas, dependendo do tempo obtido, tem-se uma conclusão sobre o quão é otimizada o suficiente para entregar ao usuário uma boa experiência, conseguindo executar as funcionalidades da FACELOCK sem precisar se preocupar com a demora das respostas.

No gráfico 21, o eixo y representa uma escala de tempo em (ms/1000) e no eixo y são as requisições de cada ferramenta, sendo assim, podemos perceber que apenas uma das funcionalidades do sistema atingiu mais que 1,4kms, ou seja, teve uma média próxima de 1,5s de resposta. O tempo de resposta informa o quão o serviço é rápido o suficiente para realizar as atividades propostas, sendo assim, esse tempo de resposta é um tempo bem rápido comparado a sua funcionalidade, o método post da API SkyBiometry, permite lançar um arquivo em imagem de no máximo 2mb para fazer o processamento de reconhecimento, e pelo tempo de resposta, levou apenas 1,49kms em média para fazer o reconhecimento facial da imagem, torna-o

eficiente para aplicação. Para API Câmera, podemos perceber que os envios de imagens para aplicação Web, são bem rápidas, durando certa de 0,36kms no método post, como ela só tem a funcionalidade de enviar imagens pela webcam, acaba que os métodos GET e PUT não são utilizados. Para o Fiware, podemos ver que os GET para identificar o contexto a receber é bem rápido, e para atualizar com o método PUT, já tem um tempo de resposta maior, visto que você está enviando contexto diretamente para o fluxo do FIWARE e entre dispositivos, já o método post para criar dispositivos é bem mais rápido, com 0,79kms.

Figura 21: Gráfico dos resultado do tempo de resposta das requisições feitas.



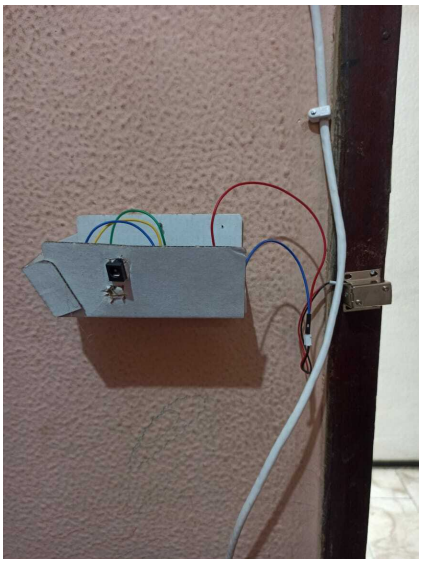
Fonte: elaborado pelo autor.

6.2.4 Imagens do protótipo e do dashboard de face

Nessa subseção tem duas imagens importantes consideradas em alguns testes, a imagem 22 tem a fechadura instalada em uma porta real, onde foi utilizada para os experimentos, e funcionou corretamente, já na imagem 23, temos a tela de cadastro de imagens para reconhecimento facial, onde na imagem mostra “imagem original cadastrada”, e a outra imagem que tem

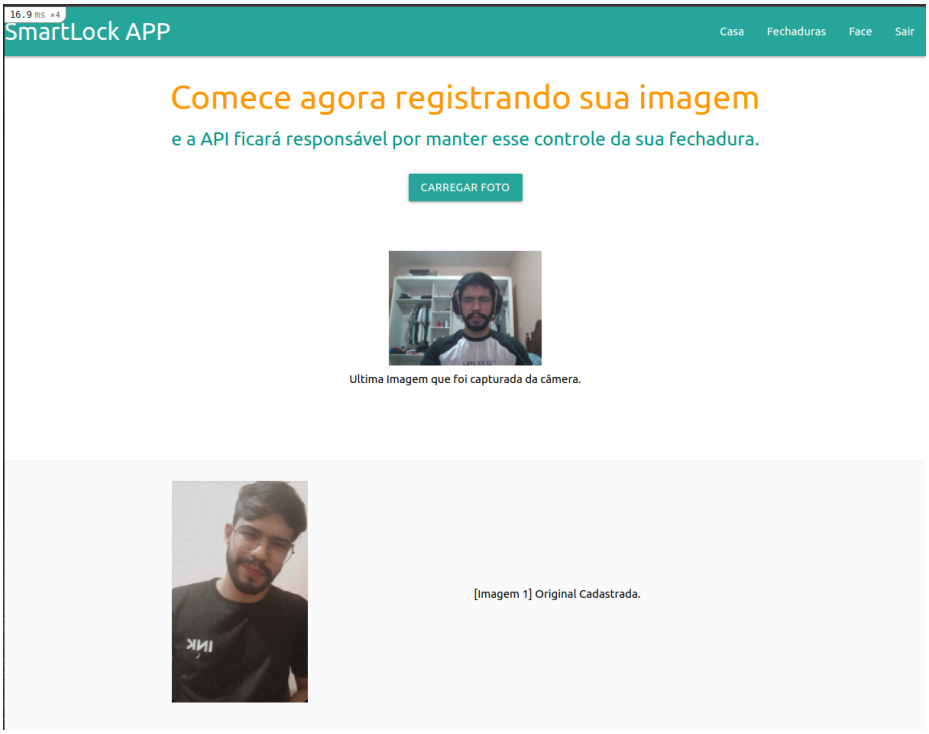
como descrição “ultima imagem capturada da câmera”, refere-se a imagem capturada para pela API Câmera e obtida pela aplicação, essa imagem é a utilizada para fazer o reconhecimento.

Figura 22: Protótipo contruído e utilizado para testes



Fonte: elaborado pelo autor.

Figura 23: Imagem do Dashboard Face



Fonte: elaborado pelo autor.

7 CONCLUSÕES E TRABALHOS FUTUROS

As *smart locks*, diferentemente das fechaduras convencionais, consegue tomar decisões, ou seja, que a fechadura consegue autorizar a manipulação da tranca, baseado em alguma implementação programada e realizar outras várias ações a partir do destravamento da porta. Sendo muito utilizado em ambientes que necessitam de controle e automação, como casas, hotéis, hospitais, etc. A facilidade e a eficiência que o destravamento de portas traz para um usuário é imensa, pois ajuda em inúmeros casos, desde a perca de chaves até no auxílio as pessoas deficientes. A evolução da tecnologia, permitiu que esse projeto seja feito, sendo assim, tornando viável a construção da *FaceLock*, uma fechadura que utiliza reconhecimento facial, aplicação Web, sistemas embarcados e um *middleware* para comunicação.

A proposta deste trabalho, foi desenvolver um sistema denominado *FaceLock*, capaz de gerenciar usuários e fechaduras por uma aplicação Web, construir um protótipo embarcado utilizando o microcontrolador ESP32 Cam, utilizar o *middleware FIWARE* para comunicação entre os dispositivos e também utilizar de API para o reconhecimento facial de imagens.

A FaceLock teve o seu sistema embarcado construído e o seu funcionamento de forma correta, onde os componentes eletrônicos mencionados neste documento tiveram êxito em suas funcionalidades, permitindo o comportamento certo da fechadura eletrônica controlada. A aplicação Web foi contruída em RoR, gerenciando todas as funcionalidades propostas. A configuração do *FIWARE*, foi contruída e utilizada de maneira correta, onde os dispositivos conseguiram estabelece a uma comunicação válida e segura, em todo o seu ciclo.

Para o reconhecimento facial, a API Câmera que implementada e utilizada pela webcam do notebook, se comportou exatamente como esperado, identificando rostos das pessoas e enviando para aplicação RoR desenvolvida, que realiza desta forma a identificação facial pela imagem obtida, onde solicita para API SkyBiometry fazer essa verificação e retornar as informações necessárias para o reconhecimento do usuário e assim possibilitando o destravamento da fechadura.

Conforme os experimentos e resultados obtidos, foi possível seguir os passos executando na prática e de forma simulada. Os experimentos trouxe relevância para o que o sistema *FaceLock* oferece para o usuário, onde possibilita de forma precisa e eficiente a troca de informações entre a fechadura e a aplicação, permitindo assim com que o usuário manipule sua fechadura de forma coesa. Porém, em alguns dos resultados, em específico no reconhecimento facial, apresentou um pequeno erro na verificação de pessoas parecidas, como irmãos gêmeos,

caso haja problema com esse tipo de verificação, deve ser possível que outras APIs possam fazer esse tratamento ou implementar algoritmos para treinar cada imagem em específica, evitando essas identificações errôneas.

O sistema de forma geral, cumpriu com o que foi estabelecido, atuando bem na identificação dos usuários pelo reconhecimento facial, além de todo o gerenciamento do sistema e troca de informações com o sistema embarcado. O sistema torna-se viável de ser utilizado para usuários que sentem a necessidade de automação de fechaduras, principalmente para quem tem dificuldade em se locomover.

7.1 Trabalhos Futuros

Como trabalhos futuros, pretende-se realizar o aprimoramento do sistema web, permitindo com que seja implantado em algum servidor em nuvem, visto que foi testado localmente e tornado IP público apenas para testes, além disso, organizar o layout e funcionalidades da página, para ficar mais próximo da usabilidade para usuário, trabalhando em *User Experience* (UX) e *User Interface* (UI).

Em relação à API Câmera, utilizada para captar imagens da webcam de um notebook e enviar para a aplicação, faz-se necessário adaptar para uma câmera externa acoplada a placa embarcados que ajude a identificar o usuário, possibilitando a instalação da câmera em portas, ou então, outra alternativa seria usar a câmera também do dispositivo para fazer o reconhecimento.

Além disso, outra possibilidade, seria implementar a funcionalidade de sensor de presença, que ao invés de está mandando requisições para a aplicação sempre que identificar um rosto pela câmera, enviar somente quando o usuário estiver a certa distância da câmera, assim, possibilitando ligar a câmera só quando necessário, que seria quando usuário estivesse próximo e também evitaria esta sempre mandando requisições para aplicação sem necessidade.

Utilizar uma fechadura que suporte abertura manualmente, seria uma opção para evitar quando a casa do usuário estiver sem energia fique incapacitado de abrir sua porta.

Realizar segurança de comunicação entre a API Câmera e a aplicação em RoR, pois a comunicação entre eles é livre, não existe segurança, poderia ser implementado criptografia assimétrica, para o compartilhamento de chaves privadas e públicas entre eles, para ter mais segurança nas informalções.

REFERÊNCIAS

ANURADHA, R.; BHARATHI, R.; KARTHIKA, K.; KRITHIKA, S.; VENKATASUBRAMANIAN, S. Optimized door locking and unlocking using iot for physically challenged people. **International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization)**, [*S. l.*], v. 4, n. 3, p. 3397–3401, 2016.

ASHTON, K. That 'internet of things' thing. **RFID journal**, [*S. l.*], v. 22, n. 7, p. 97–114, 2009.

BARRETO, M.; BEZERRA, F. J.; BARRA, W. F.; NOGUEIRA, F. Controle por realimentação para melhoria do desempenho dinâmico de requisições http em máquinas virtuias. In: **Simposio Brasileio de Automação Inteligente–SBAI**. [*S. l.*: *s. n.*], 2015.

BIANCHINI, Â. R. **Arquitetura de redes neurais para o reconhecimento facial baseado no neocognitron.** São Carlos: Universidade Federal de São Carlos, 2001.

BRAGA, L. F. Z. *et al.* **Sistemas de Reconhecimento Facial**. Tese (Trabalho de Conclusão de Curso) – Escola de Engenharia de São Carlos, UNIVERSIDADE DE SÃO PAULO, São Paulo, 2019.

GROSSMANN, G. H. **IoT Smart Lock (ISL)**: sistema de fechadura inteligente, utilizando protocolos de internet das coisas. Curitiba: Universidade Tecnológica Federal do Paraná, 2018.

HARTL, M. **Ruby on rails tutorial**: learn web development with rails. [*S. l.*]: Addison-Wesley Professional, 2015.

HERNANDES, S. C. L.; PELLENZ, M. E.; CALSAVARA, A. A study on publish-subscribe middlewares for selective notification delivery in smart cities. In: IEEE. **2019 XLV Latin American Computing Conference (CLEI)**. [*S. l.*], 2019. p. 1–10.

JAHNAVI, S.; NANDINI, C. Smart anti-theft door locking system. In: IEEE. **2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE)**. [*S. l.*], 2019. p. 205–208.

JEONG, J.-i. A study on the iot based smart door lock system. In: **Information Science and Applications (ICISA) 2016**. [*S. l.*]: Springer, 2016. p. 1307–1318.

LEITE, D. R. **PREVISÃO DE COTAÇÕES HISTÓRICAS DE AÇÕES UTILIZANDO REDES NEURAIS ARTIFICIAIS**. [*S. l.: s.n.*], 2020.

LISLE, B.; TEIXEIRA, J. M. X. N. Sistema de telemonitoramento de baixo custo usando iot. In: SBC. **Anais Estendidos do XXII Simpósio de Realidade Virtual e Aumentada**. [*S. l.*], 2020. p. 54–58.

LÓPEZ-PÉREZ, D.; GARCIA-RODRIGUEZ, A.; GALATI-GIORDANO, L.; KASSLIN, M.; DOPPLER, K. Ieee 802.11 be extremely high throughput: The next generation of wi-fi technology beyond 802.11 ax. **IEEE Communications Magazine**, IEEE, v. 57, n. 9, p. 113–119, 2019.

MAGRANI, E. **A internet das coisas**. [*S. l.*]: Editora FGV, 2018.

MARGALHO, R. C. **Sistema De Gestão de Fechaduras Inteligentes Usando IoT para Aplicação em Cacifos de Universidade**. Tese (Doutorado) – Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Ciências e Tecnologias, Universidade de Coimbra, Coimbra, 2019.

MARTINS, H. P.; FERREIRA, A. N.; FERREIRA, R. A.; TOKUMITSU, T. Teste comparativo de uma rede local com wi-fi e a tecnologia power line communication. **Caderno de Estudos Tecnológicos**, [*S. l.*], v. 5, n. 1, 2017.

MARWEDEL, P. **Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things**. [*S. l.*]: Springer Nature, 2021.

MATIAS, A. F. **Letramento digital de pessoas com deficiência intelectual durante a ação de blogagem**: uma análise das ações e das emoções. [*S. l.: s.n*], 2016.

MISHRA, R.; RANSINGH, A.; BEHERA, M. K.; CHAKRAVARTY, S. Convolutional neural network based smart door lock system. In: IEEE. **2020 IEEE India Council International Subsections Conference (INDISCON)**. [*S. l.*], 2020. p. 151–156.

MTSHALI, P.; KHUBISA, F. A smart home appliance control system for physically disabled people. In: IEEE. **2019 Conference on Information Communications Technology and Society (ICTAS)**. [*S. l.*], 2019. p. 1–5.

MURATORI, J. R.; BÓ, P. H. D. Capítulo i automação residencial: histórico, definições e conceitos. **O Setor elétrico**, [*S. l.*], p. 70–77, 2011.

OLIVEIRA, S. de. **Internet das coisas com ESP8266, Arduino e Raspberry PI**. [*S. l.*]: Novatec Editora, 2017.

PINJALA, S. R.; GUPTA, S. Remotely accessible smart lock security system with essential features. In: IEEE. **2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)**. [*S. l.*], 2019. p. 44–47.

PLAMANESCU, R.; GHEORGHE, S.; SARB, M.; ISTENES, A.; ALBU, M. A synchronized measurements fiware platform for smart grid applications. In: IEEE. **2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE)**. [*S. l.*], 2019. p. 1–5.

RAHMAN, T.; CHOWDHURY, M. E.; KHANDAKAR, A.; ISLAM, K. R.; ISLAM, K. F.; MAHBUB, Z. B.; KADIR, M. A.; KASHEM, S. Transfer learning with deep convolutional neural network (cnn) for pneumonia detection using chest x-ray. **Applied Sciences**, **Multidisciplinary Digital Publishing Institute** [*S.l.*], v. 10, n. 9, p. 3233, 2020.

RAJ, V.; CHANDRAN, A.; RS, A. P. Iot based smart home using multiple language voice commands. In: IEEE. **2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)**. [*S. l.*], 2019. v. 1, p. 1595–1599.

RAMPARANY, F.; MARQUEZ, F. G.; SORIANO, J.; ELSALEH, T. Handling smart environment devices, data and services at the semantic level with the fi-ware core platform. In: IEEE. **2014 IEEE International Conference on Big Data (Big Data)**. [*S. l.*], 2014. p. 14–20.

ROBLES, R. J.; KIM, T.-h.; COOK, D.; DAS, S. A review on security in smart home development. **International Journal of Advanced Science and Technology**, Citeseer, [*S. l.*], v. 15, 2010.

ROHITH, B. Computer vision and iot enabled bot for surveillance and monitoring of forest and large farms. In: IEEE. **2021 2nd International Conference for Emerging Technology (INCET)**. [*S. l.*], 2021. p. 1–8.

ROZANSKA, A.; RACHWANIEC-SZCZECINSKA, Z.; KAWALA-JANIK, A.; PODPORA, M. Internet of things embedded system for emotion recognition. In: **2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)**. [*S. l.*: *s. n.*], 2018. p. 1–5.

RUBACK, L.; AVILA, S.; CANTERO, L. Vieses no aprendizado de máquina e suas implicações sociais: Um estudo de caso no reconhecimento facial. In: SBC. **Anais do II Workshop sobre as Implicações da Computação na Sociedade**. [*S. l.*], 2021. p. 90–101.

SANTOS, B. P.; SILVA, L.; CELES, C.; BORGES, J. B.; PERES, B. S. N.; VIEIRA, M. A. M.; VIEIRA, L. F. M.; GOUSSEVSKAIA, O. N.; LOUREIRO, A. **Internet das coisas**, subtitle= da teoria à prática. **Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos**, [*S. l.*], v. 31, p. 16, 2016.

SHAH, D.; JHA, P.; AWASTHI, V.; MAU, P.; KOTHARI, B.; MARU, I. Enhanced pyrometric device with long range for mass screening based on mlx90614. In: IEEE. **2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE)**. [*S. l.*], 2021. p. 1–4.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding machine learning**: From theory to algorithms. [*S. l.*]: Cambridge university press, 2014.

VALLADARES, L. N. W. **UTFACE**: uma api restful para gerenciamento de presença com reconhecimento facial 2019. 27 f. Tese (Trabalho de Conclusão de Curso) – Universidade Tecnológica Federal do Paraná, Guarapuava, 2019.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: SN. **Proceedings of the xxix conference on graphics, patterns and images**. [*S. l.*], 2016. v. 1, n. 4.

VISWANATHAN, V. Rapid web application development: a ruby on rails tutorial. **IEEE software**, IEEE, [*S.l*], v. 25, n. 6, p. 98–106, 2008.