UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

JORGE LUCAS COLARES MARTINS

SMART LOCK: DESENVOLVIMENTO DE UMA FECHADURA ELÉTRICA ACIONADA POR VOZ

QUIXADÁ

2022

JORGE LUCAS COLARES MARTINS

SMART LOCK: DESENVOLVIMENTO DE UMA FECHADURA ELÉTRICA ACIONADA POR VOZ

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
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M343s Martins, Jorge Lucas Colares.
SMART LOCK: desenvolvimento de uma fechadura elétrica acionada por voz / Jorge Lucas Colares Martins. – 2022.
60 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. Thiago Werlley Bandeira da Silva.

1. Assistente virtual inteligente. 2. Fiware. 3. Internet das Coisas. I. Título.

CDD 621.39

JORGE LUCAS COLARES MARTINS

SMART LOCK: DESENVOLVIMENTO DE UMA FECHADURA ELÉTRICA ACIONADA POR VOZ

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

Prof. Me. Leonardo Torres Marque
Universidade Estadual do Ceará (UECE)

Me. Abdul-Hamid Matos Moreira
Universidade Federal do Ceará (UFC)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

AGRADECIMENTOS

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

Agradeço aos meus pais, principalmente a minha mãe Francisquinha Colares, mas conhecida como Kinha, por seu apoio, incentivo nas e cobranças, horas difíceis, de desânimo e cansaço e sempre batalhou por mim.

À minha irma Ana Ádila, 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 tudo, 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, Gregório Neto, 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.

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 deste ciclo vivido ao longo desses 5 anos e contribuíram de alguma forma na minha formação profissional e pessoal.

"O único lugar onde o sucesso vem antes do trabalho é no dicionário."

(Stubby Currence)

RESUMO

Propõe-se com este trabalho o desenvolvimento de um protótipo chamado *Smart Lock*, como uma solução de fechadura inteligente controlado por uma aplicação *mobile* e com controle por voz através da assistente virtual Alexa. O dispositivo desenvolvido trata-se de um microcontrolador ESP32CaM, para controle e comunicação com os demais dispositivos e as aplicações. Esse sistema contem uma aplicação *mobile*, capaz de realizar *login* de identificação, cadastro de usuários e fechaduras, e controle das fechaduras, desenvolvida em Flutter. Para realizar a interoperabilidade entre os dispositivos, foi utilizado um *middleware*, caracterizado pela segurança na transmissão de dados, denominado FIWARE. Após os testes realizados no sistema, os resultados obtidos foram satisfatórios, conseguindo controlar a fechadura pela aplicação *mobile* e pela voz. Isso evita que usuários não identificados tenham acesso, tanto pela aplicação *mobile*, como pela *skill* da assistente virtual Alexa. Após análise de acesso, observa-se que o dispositivo desenvolvido tem potencial prático para uso, propondo uma solução para que as pessoas tenham melhor controle das fechaduras, em especial as pessoas com mobilidade reduzida.

Palavras-chave: Assistente virtual inteligente; Fiware; Internet das Coisas.

ABSTRACT

This work proposes the development of a prototype called Smart Lock, as a smart lock solution controlled by a mobile application and with voice control through the virtual assistant Alexa. The developed device is an ESP32CaM microcontroller, for control and communication with other devices and applications. This system contains a mobile application, capable of performing identification login, user and lock registration, and lock control, developed in Flutter. To perform interoperability between devices, using middleware, characterized by security in data transmission, called FIWARE. After the tests carried out on the system, the results obtained were satisfactory, managing to control the lock through the mobile application and through voice. This prevents unidentified users from having access, both through the mobile application and through the Alexa virtual assistant skill. After access analysis, it is observed that the developed device has practical potential for use, proposing a solution for people to have better control of the locks, especially people with reduced mobility.

Keywords: Smart virtual assistant; Fiware; Internet of Things.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

IoT	*Internet of Things*
P.M.R	Pessoa com Mobilidade Reduzida
IBGE	Instituto Brasileiro de Geografia e Estatística
MQTT	*Message Queue Telemetry Transport*
Wi-Fi	*Wireless Fidelity*
GE	*Generic Enablers*
API	*Apllication Programming Interface*
NGSI	*Next Generation Service Interface*
CRUD	*Create Read Update Delete*
REST	*Representational State Transfer*
HTTPs	*Hyper Text Transfer Protocol Secure*
HTTP	*Hyper Text Transfer Protocol*
JSON	*JavaScript Object Notation*
UART	*Universal asynchronous receiver/transmitter*
I2C	*Inter-Integrated Circuit*
CAN	*Controller Area Network*
IDE	*Integrated Development Environment*
RFID	*Radio Frequency Identification*
IR	*Infrared radiation*
SMS	*Short Message Service*
ID	*Identity Document*

LISTA DE SÍMBOLOS

ms	Tempo em milissegundos
s	Tempo em segundos
kHz	Frequência em *Kilohertz*
Hz	Frequência equivale a 1 segundo
$k\Omega$	Resistência elétrica em *Kiloohms*
V	Tensão Elétrica
mb	*Megabytes*

SUMÁRIO

1 INTRODUÇÃO

A tecnologia com o passar dos anos se torna mais essencial para a humanidade, com melhorias como, por exemplo, aparelhos de comunicação, veículos, eletrodomésticos ou até mesmo proporcionado um serviço digital. Com essa evolução, a *internet* vem ganhando cada vez mais espaço e permitindo que "coisas" que antes não estariam conectadas possam se conectar (PANCHANATHAN *et al.*, 2019). Nesse sentido, o conceito de *Internet of Things* (IoT) vem ganhando mais espaço atualmente (MAGRANI, 2018).

Segundo Oliveira (2017), IoT não é somente ligar luzes utilizando o celular, não é meramente ligar ou desligar as "coisas" com o auxílio da *internet*, mas também transformá-las em inteligentes, possuindo a capacidade de processar e coletar informações adquiridas das redes ou ambientes em que estão implantados. Nesse contexto, utilizam-se aparelhos de forma inteligente e transformadora, facilitando o dia a dia das pessoas, em alguns casos inserem a IoT para auxiliar pessoas com deficiência ou que tenham sua mobilidade reduzida.

Pessoas com deficiência, são aquelas que apresentam limitações físicas, mentais, intelectuais ou sensoriais de longo prazo, o que pode impedi-los de participar plena e efetivamente da sociedade em igualdade de condições com os demais, podendo possuir uma ou mais limitações (BRASIL, 2015).

Pessoa com Mobilidade Reduzida (P.M.R), são pessoas que apresentam dificuldades de mobilidade permanente ou temporária por qualquer motivo, resultando em redução efetiva da mobilidade, da flexibilidade, da coordenação motora, ou da percepção sensorial. Nesse grupo estão inclusos idosos, gestantes, lactantes, lactentes e obesos (BRASIL, 2015).

Segundo o censo da população brasileira de 2010, realizada pelo Instituto Brasileiro de Geografia e Estatística (IBGE), indicam que cerca de 45,6 milhões de brasileiros declararam ter alguma deficiência. Esses dados se tratam de um percentual de 23,9% da população do Brasil. A deficiência visual foi a que mais se mostrou presente nos entrevistados e chegou a 35,7 milhões de pessoas. Outros dados mostram que o número de pessoas que possuem deficiência motora atingiu a marca de cerca de 13,1 milhões (OLIVEIRA *et al.*, 2012).

A deficiência visual é a perda ou redução da capacidade visual em ambos os olhos em caráter definitivo que não pode ser corrigida com lentes ou óculos e aumenta a dificuldade em realizar tarefas do seu cotidiano (WITTE, 2020). Pessoas com deficiência física enfrentam dificuldades cognitivas ou físicas ao realizar tarefas, desde tarefas simples como abrir uma porta, até utilizar um celular ou computador se tornam mais desafiadoras para essas pessoas.

A tecnologia assistiva fornece acessibilidade adicional, proporcionando a estes uma melhor condição de vida (DHANJAL; SINGH, 2019).

As tecnologias atuais vêm proporcionando uma maior acessibilidade para todos os indivíduos da sociedade. Algumas dessas tecnologias em ascensão são sistemas que possuem auxílio por comando por voz. Pessoas que possuem mobilidade reduzida ou pessoas com deficiência, conseguem desfrutar melhor dessa tecnologia, principalmente idosos ou pessoas que apresentam deficiência visual. À área de automação residencial controlada por voz vem se expandindo recentemente com as melhorias de assistentes virtuais (SOUSA, 2018).

Com o uso de microcontroladores, fez-se praticável a manipulação de sensores, motores e outros dispositivos eletrônicos (HOLLANDA; MIYANO, 2015). Além disso, também é possível sua integração com assistentes virtuais já estabelecidos no mercado, de modo a controlar dispositivos eletrônicos, assim possibilitando qualidade e praticidade de manuseio dos dispositivos.

Apesar da acessibilidade está melhorando cada vez mais, há a necessidade de dispositivos que auxiliem. É proposto com este trabalho o desenvolvimento de uma fechadura inteligente direcionado para pessoas com deficiência visual, mobilidade reduzida, ou ainda para aqueles que buscam mais praticidade para o gerenciamento e controle de suas fechaduras. O dispositivo proposto visa controlar e monitorar uma fechadura eletrônica que seja capaz de ser manuseada tanto com uma aplicação *mobile*, quanto por comando de voz utilizando um assistente eletrônico como intermediário. O dispositivo enviará instruções para o *middleware* denominado FIWARE, que, irá conduzir instruções para a fechadura, assim possibilitando trancá-la e destrancá-la e permitindo assim um controle à distância de sua fechadura.

1.1 Objetivos

O objetivo com este trabalho é desenvolver uma fechadura inteligente, que consiga ser controlada tanto por uma aplicação *mobile* quanto por um assistente eletrônico com detecção por voz, voltado ao auxílio de pessoas que possuem alguma deficiência ou dispõem de sua mobilidade reduzida.

Objetivos específicos:

a. realizar teste de controle por voz;
b. elaborar uma aplicação *mobile* para o manuseio e controle da fechadura;
c. testar segurança e a acessibilidade do usuário;

d. testar o tempo de comunicação do usuário e a perda de pacotes;

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte forma:

- Capítulo 2: apresenta-se a fundamentação teórica, com os conceitos fundamentais para a compreensão da proposta descrita.
- Capítulo 3: são apresentados os trabalhos relacionados, comparando os aspectos comuns ou divergentes entre eles e o trabalho aqui proposto.
- Capítulo 4: é apresentada a metodologia utilizada para o desenvolvimento deste trabalho. Além disso, define o cronograma e o planejamento de conclusão deste trabalho.
- Capítulo 5: são apresentados os materiais e métodos utilizados para o desenvolvimento do protótipo. Exibindo a esquematização do sistema, e desenvolvimento da aplicação e *skill* Alexa.
- Capítulo 6: é relatado os experimentos e resultados obtidos do sistema VOICELOCK, verificando alguns pontos importantes, como perda e velocidade de entrega de dados e o comando de voz.
- Capítulo 7: são apresentadas as conclusões sobre a execução deste trabalho, informa sobre o sistema se comportou mediante aos resultados e aborda ideais para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são descritos os conceitos importantes deste trabalho. Na Seção 2.1 é apresentado o conceito de automação residencial, listando suas características. Na Seção 2.2 é abordado o *middleware* FIWARE com seu funcionamento e suas aplicações. Na Seção 2.3 é tratado sobre o protocolo de comunicação *Message Queue Telemetry Transport (MQTT)*, explicando como é realizada a troca de mensagens com esse protocolo. Na Seção 2.4 é contextualizado os detalhes sobre o microcontrolador *ESP32*, tais como alguns de seus componentes e aplicações. Na Seção 2.5 são expostos os conceitos relacionados a área de IoT, relacionando-a com a área de automação residencial.

2.1 Automação Residencial

O termo automação foi introduzido em meados do século XX, desde então, o próprio conceito está passando por mudanças, revolucionado a indústria, através da introdução a máquinas que não precisam ser manuseadas pelas pessoas. O conceito de automação está diretamente relacionado a automação de quaisquer dispositivos para reduzir a carga de trabalho e seguir os requisitos especificados. O exemplo mais comum de automação é a automação residencial (DUTT *et al.*, 2020).

Automação residencial é uma área de pesquisa que busca a capacidade de programar eventos em uma casa e de tornar automático o funcionamento de diversos equipamentos. O sistema utilizado em automação residencial é composto por *hardware* com comunicação entre dispositivos e interfaces eletrônicas capazes de integrar dispositivos elétricos mutualmente. Desse modo, é possível realizar atividades domésticas até mesmo com um botão, controlando remotamente sistemas como TV, ar-condicionados, geladeiras, dentre outros (JAIN *et al.*, 2019).

Com a inclusão de aparelhos celulares, *tablets* e computadores na vida das pessoas, facilita-se o uso da automação residencial, sendo que em qualquer comodo de sua casa tem acesso à conexão. Em razão da disponibilidade de rede em toda residência, diversos aparelhos domésticos já estão sendo fabricados com o suporte para a comunicação *Wireless Fidelity* (Wi-Fi), sendo isto uma vantagem para a implementação das técnicas da automação residencial (VALOV; VALOVA, 2020).

Todavia, os desenvolvedores muita das vezes não disponibilizam os dados coletados de seus aparelhos, geralmente aparelhos são adquiridos de diversas marcas diferentes, isso pode

acarretar uma sobrecarga de aplicativos de controle (VALOV; VALOVA, 2020).

O conceito de automação residencial é indispensável para este trabalho, pois o entendimento de controle e de programação de eletrônicos de uma casa está extensamente ligada a fechaduras que possam ser controladas a partir de um dispositivo *mobile.* Além de que, o propósito de controlar uma fechadura através de um dispositivo *mobile* e/ou assistente virtual, é bastante facilitado com o emprego de noções de IoT de automação residencial.

2.2 FIWARE

O FIWARE é um *middleware* que possui um conjunto de especificações de *Application Programming Interfaces* (*APIs*) públicas gratuitas agrupadas com implementações de referência de código aberto. A plataforma FIWARE está agregada em algumas partes principais, chamadas *Generic Enablers* (GE)s (SALHOFER *et al.*, 2019).

Uma *GE* representa um serviço do FIWARE fornecendo um ou mais componentes com as implementações de referência que suportam as *Apllication Programming Interface (API)s* especificadas, como, por exemplo, *IoT Agents* que possui componentes necessários para rotear dados de sensores e configurar rede de sensores para outras *GEs*. Além disso, o FIWARE possui uma grande compatibilidade com diversas linguagens de programação (SALHOFER *et al.*, 2019).

O FIWARE já foi utilizado para a elaboração de uma plataforma de dados, com o uso voltado para cidades inteligentes no Japão, devido a um dos seus recursos mais importantes, os modelos de dados, os quais são bastante padronizados, assim possibilitando que outras pessoas possam integrar com facilidade (PHAM *et al.*, 2020).

Alguns exemplos de cidades no Japão que utilizaram o FIWARE para construção de sua plataforma de dados, são: a cidade de Takamatsu na prefeitura de Kagawa, cidade de Kakogawa na prefeitura de Hyogoture, e a cidade de Kawasaki na prefeitura de Kanagawa (PHAM *et al.*, 2020).

Neste trabalho são aplicados a execução especialmente de duas *GEs*. Para o gerenciamento dos dados enviados pelo *Orion Context Broker*, cujo funcionamento e propriedades são explicadas na Subseção 2.2.1. Para realizar a integração do *middleware FIWARE* e o dispositivo desenvolvido *GE IoT Agent*, para designar tal função, seu funcionamento e propriedades são esclarecidas na Subseção 2.2.2.

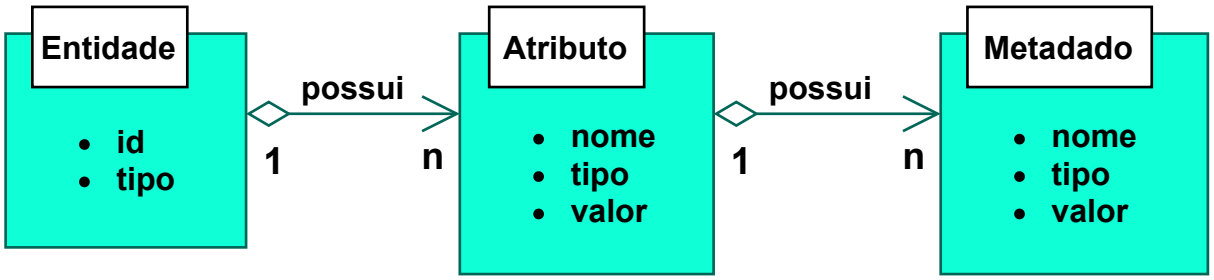
2.2.1 Orion Context Broker

O *Orion Context Broker* é uma *GE* para gerenciamento de dados, sendo a cabeça central do *middleware*. Ele gerencia e fornece acesso a informações de objetos além de seus recursos de contexto. Através da aplicação da *GE*, é possível criar entidades que possuem atributos com metadados, assim, como ilustrado na Figura 1. Além disso, conseguindo armazenar os dados de contexto atual, o banco de dados *MongoDB* é empregado para a persistência das informações (PHAM *et al.*, 2020).

Com a intenção de padronização de dados o Orion aplica o modelo de informação *Next Generation Service Interface* (NGSI)-v2, através da utilização deste modelo as entidades ficam definidas da seguinte forma (PHAM *et al.*, 2020):

- uma entidade possui um Identificador (ID) exclusivo e um tipo;
- uma entidade possui um ou muitos atributos;
- um atributo possui o nome do atributo, seu valor e a categoria de seu valor;
- um atributo possui um ou muitos metadados;
- o metadado está incluso nome, valor e tipo do valor.

Figura 1 – Modelo NGSI baseado no *Orion* do FIWARE



Fonte: Adaptado de (PHAM *et al.*, 2020)

O *Create Read Update Delete* (CRUD) das informações do Orion, podem ser realizadas através de *APIs Representational State Transfer* (REST), utilizando requisições *Hyper Text Transfer Protocol Secure* (HTTPs), ou manuseando uma aplicação denominada *Postman*, uma *API* que possibilita criar e salvar protocolos de requisição *Hyper Text Transfer Protocol* (HTTP) e *HTTPs* (PHAM *et al.*, 2020).

No contexto deste trabalho, o *Orion Context Broker* é utilizado para a criação das entidades que representam os dispositivos e os locais em que estão. Assim conseguindo armazenar os dados resultantes da última ação realizada na fechadura, possibilitando um monitoramento da última ação executada naquele dispositivo. Ainda sendo possível, a comunicação desta *GE* com

outro banco de dados, com o objetivo de armazenar ações ainda mais antigas sobre a fechadura, facilitando a análise dos dados.

2.2.2 IoT Agent

O *IoT Agent GE* é um conjunto de módulos de *software*, que possuem protocolos *IoT* sentido Sul e interações sentido Norte. Por meio da utilização desta *GE*, é possível trabalhar com dispositivos *IoT* que empregam os protocolos de comunicação *LW2M2M*, *JavaScript Object Notation* (JSON), *UltraLight* sobre *HTTP*, *MQTT*, *LoRaWAN*, dentre outros. Além disto, as unidades *IoT* usada nesta *GE* são representados dentro do *middleware* FIWARE, como entidades de formato *NGSI-v2* no *Orion* (CONDE *et al.*, 2021).

Na Tabela 1 é mostrado alguns agentes disponíveis nesta *GE*, além de suas semânticas e protocolos de transporte.

Tabela 1 – Agentes *IoT* do FIWARE

Agente	Semantica	Transporte
OMA M2M leve	Especificação *LWM2M* desenvolvida pela *Open Mobile Alliance*	*CoAP*
Agente *IoT JSON*	Dados codificados em *JSON*	*AMQP, HTTP, MQTT*
Agente IoT *Ultralight* 2.0	Protocolo *Ultralight* 2.0, protocolo baseado em texto desenvolvido em FIWARE	*AMQP, HTTP, MQTT*

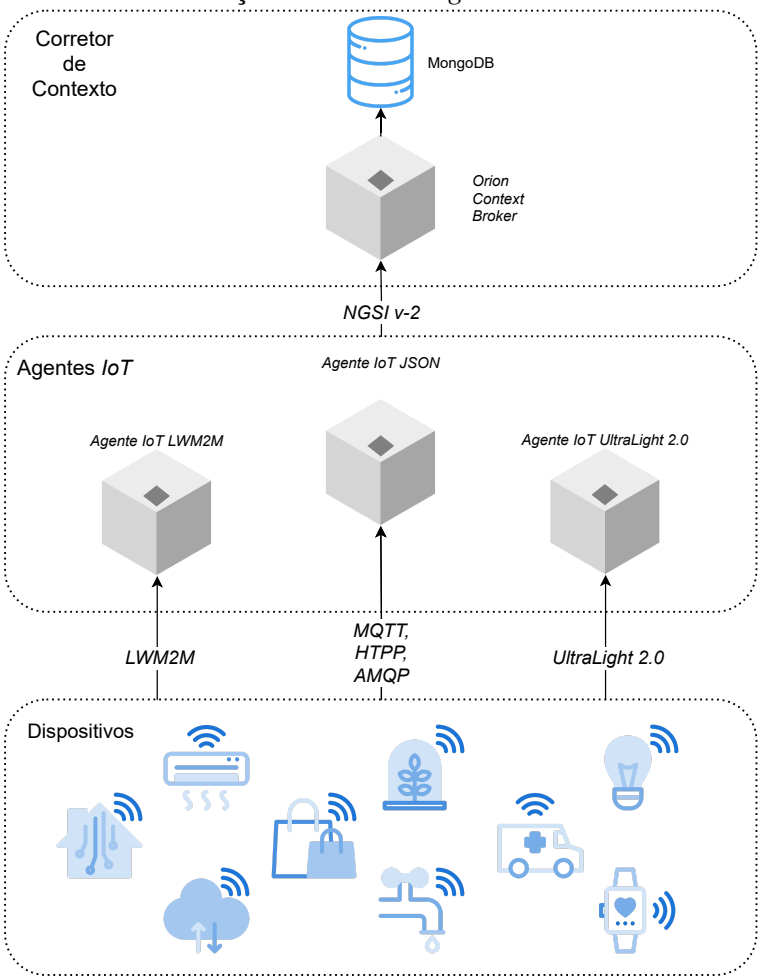
Fonte: Adaptado de (ARAUJO *et al.*, 2019).

A utilização desta *GE*, torna mais simples o manuseio e as criações de comandos para atuadores, pois apenas com a atualização de atributos específicos relacionados aos seus comandos em sua entidade *NGSI-v2* no *Orion Context Broker*, é possível enviar instruções para realizar certa ação (CONDE *et al.*, 2021).

É ilustrado na Figura 2, um exemplo de como é dada a comunicação de dispositivos do agente *IoT*. Conforme é possível perceber, os dispositivos enviam ou recebem dados e comandos dos agentes *IoT*, com uma variedade de protocolos de comunicação, logo após o agente transformar esses dados recebidos em formato *NGSI-v2*, em seguida, é enviado para o *Orion* onde é armazenado o contexto atual (ARAUJO *et al.*, 2019).

A *GE IoT Agent*, é utilizada para a criação e o manuseio das fechaduras, transformando-as em entidades *NGSI-v2* e colocando-as dentro do *middleware* FIWARE através do *Orion*. Ademais, são enviados comandos no formato *UltraLight*, utilizando o protocolo de comunicação

Figura 2 – Modelo de comunicação da *GE IoT Agent*



Fonte: Adaptado de (ARAUJO *et al.*, 2019)

MQTT, assim chegando no dispositivo no formato *NGSI-v2* e traduzidos para ações.

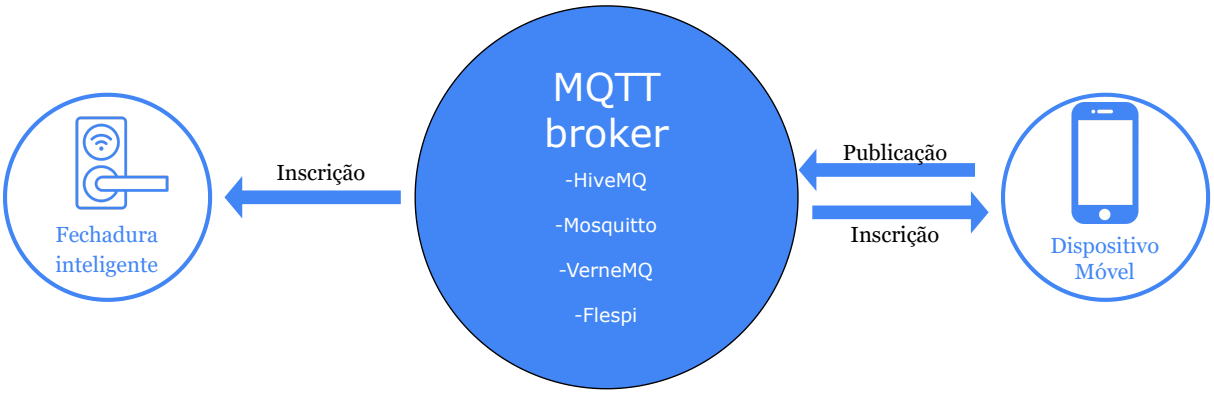
2.3 *Protocolo MQTT*

O termo *Message Queuing Telemetry Transport* (*MQTT*) é um protocolo de mensagens, muito utilizado em aplicações *IoT*, geralmente, bastante empregado com o uso de sensores e dispositivos *mobiles*. O funcionamento do protocolo é dado através de publicações e inscrições, divididos em três partes, que são *broker*, *publish* e o *subcrisbe* (SU *et al.*, 2019).

O *broker* é o responsável por gerenciar as mensagens, encaminhando cada mensagem para seu tópico específico. O *publish* é encarregado de publicar uma mensagem para um determinado tópico. O *subscribe* se inscreve em um tópico, assim recebendo todas as mensagens publicadas naquele tópico estabelecido. O tópico é uma área em que se pode inscrever ou publicar mensagens, cada tópico é único, assim somente inscritos deste tópico específico podem visualizar suas mensagens (SU *et al.*, 2019).

Na Figura 3 é mostrado o modelo de comunicação *MQTT* utilizando publicação e inscrição.

Figura 3 – Modelo de publicação e inscrição *MQTT*



Fonte: Adaptado de (MILEVA *et al.*, 2021)

Neste contexto, o protocolo *MQTT* é utilizado neste trabalho em busca da simplificação de troca de informações, devido a sua característica de ser um protocolo de mensagens leves, se torna ideal para o envio de dados de controle.

2.4 Microcontrolador *ESP32*

O *ESP32* é um microcontrolador que devido às suas especificações competitivas proporcionam baixo custo e alta velocidade (KAREEM; DUNAEV, 2021). Além de que é um microcontrolador com *Bluetooth* de duplo modo e *WI-FI* integrado, o que auxilia bem no trabalho em projetos voltados a *IoT*. Em sua arquitetura, vem integrado um microprocessador *Tensilica Xtensa LX6* nas variações *dual-core* como *single-core*, possibilitando variedade e potência de processamento (MACHESO *et al.*, 2021).

O microcontrolador *ESP32* suporta os protocolos de comunicação sem fio que placas Arduino suportam (*ISP, Universal asynchronous receiver/transmitter (UART)* e *Inter-Integrated Circuit (I2C)*). No entanto, a placa *ESP32* disponibiliza opções adicionais. Dentre eles estão, o barramento de rede de área de controle (*Controller Area Network (CAN)*) e barramento serial *I2S* de interface, proporcionando diferentes abordagens de trabalho (KAREEM; DUNAEV, 2021).

Logo, a placa *ESP32* pode ser adequada para sistemas tanto simples como para complexos, que requerem processamento e potência elevadas. Além disso, ela possui compatibilidade com tanto a linguagem de programação *C* para Arduino *Integrated Development Environment* (IDE), assim dispondo ao usuário variadas maneiras de trabalho (KAREEM; DUNAEV, 2021).

A placa *ESP32* foi escolhida para ser utilizada neste trabalho para realizar o controle do atuador e realizar a comunicação com a aplicação através do *middleware* FIWARE, devido a sua característica de possuir conexão *bluetooth* e *WI-FI* integrada, facilitando a comunicação e sem necessitar de módulos externos. Com um custo benefício satisfatório, proporcionando um nível de processamento adequado.

2.5 *Internet of Things*

Com o passar dos anos, com melhorias nos equipamentos de comunicação, veículos, eletrodomésticos e até na prestação de serviços digitais. A tecnologia tornou-se cada vez mais importante para a humanidade. Com essa evolução, a Internet ganhou cada vez mais espaço e entrou nas "coisas" que antes não estariam conectadas (PANCHANATHAN *et al.*, 2019).

Com o estudo de *IoT*, empregou-se o conceito de que "coisas" podem ser interconectadas, para poderem ajudar a solucionar os problemas das pessoas. Logo, a tecnologia atual da *IoT* é combinada com vários produtos, podendo cobrir diferentes áreas de aplicação. Assim, sendo utilizada para a solução de diversos problemas e ainda podendo proporcionar segurança, conforto e praticidade (MOHAMMED, 2021).

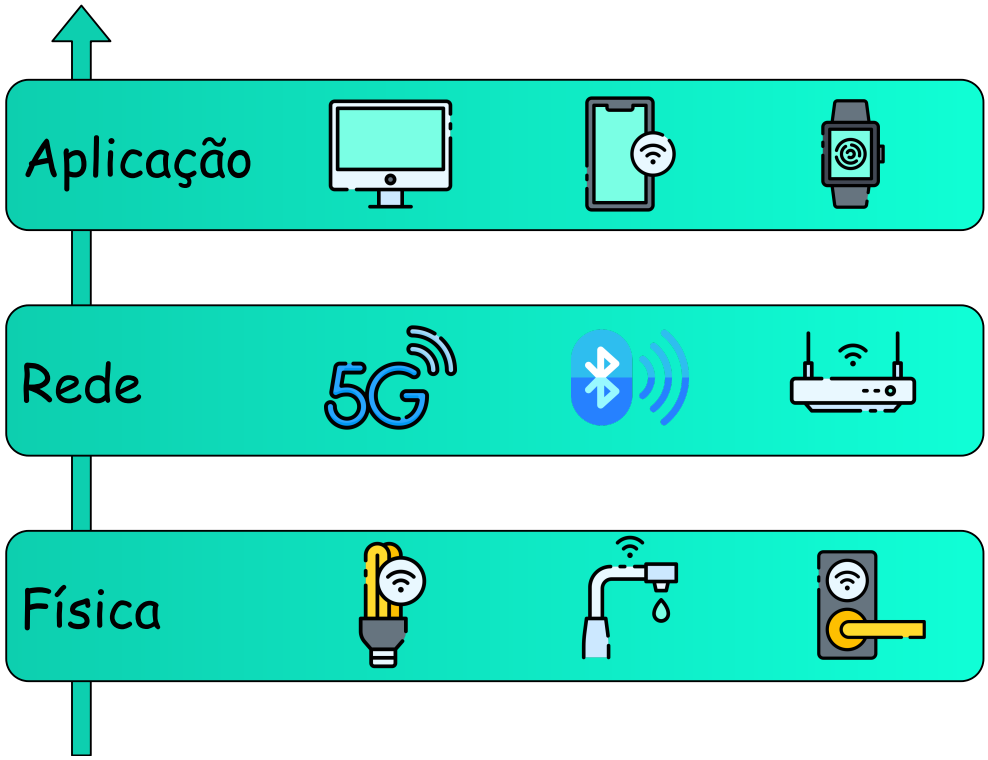
Dada a conexão dessas "coisas" à *internet*, significa criar objetos para a *IoT*. As "coisas" podem realizar comunicação entre usuários e dispositivos. Como resultado, diversas aplicações surgem, como coleta de dados de sensores, automação e controle residencial, sistemas de monitoramento para pessoas com idade avançada, dentre outras (LI *et al.*, 2021).

A tecnologia *IoT* pode ser dividida em diversas camadas, mas sendo focada em três camadas, denominadas camada física, camada de rede e camada de aplicação (GOLPÎRA *et al.*, 2021). As camadas ilustradas na Figura 4 são melhor abordadas nos tópicos a seguir.

1. **Física** – é uma camada de baixo nível, ela é responsável por gerenciar os dispositivos físicos envolvidos no projeto, geralmente essa categoria está atrelada com atuadores (GOLPÎRA *et al.*, 2021);
2. **Rede** – recebe a função de propagação de dados. Esta camada também analisa e transforma dados usando diversas tecnologias de comunicação como, *3G*, *Wi-fi*, *Bluetooh*, Infravermelho, *Zigbee*, etc. Dependendo do que os atuadores e sensores necessitarem (GOLPÎRA *et al.*, 2021);
3. **Aplicação** – A camada de aplicação detém a função de gerenciar, de receber dados provindos da camada de rede, realizando ações no dispositivo. Por exemplo, realizar uma

ação de desbloqueio de um dispositivo, e receber de um sensor de proximidade se uma porta ou janela está realmente fechada, assim podendo alertar e armazenar ações ocorridas no dispositivo (GOLPÎRA *et al.*, 2021).

Figura 4 – Camadas de *IoT*



Fonte: Elaborado pelo próprio autor

3 TRABALHOS RELACIONADOS

Neste capítulo são abordados trabalhos da literatura que este trabalho possui relação e que auxiliaram ao desenvolvimento da proposta. Desta forma, foram analisados trabalhos que possuem tecnologias similares e técnicas utilizadas em cada trabalho, com finalidade de analisar seus impactos positivos obtidos a cada um desses trabalhos pesquisados. Ao final deste capítulo, realiza-se uma comparação entre os trabalhos descritos neste capítulo e o trabalho proposto para determinar as principais semelhanças e diferenças relacionadas à tecnologia aplicada em cada estudo.

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

Em Pinjala e Gupta (2019), é elaborado um sistema de fechadura para garantir a segurança de uma casa ou prédio. Os autores utilizam uma placa de desenvolvimento denominada *Raspberry Pi 3* para o controle de seu dispositivo, *switch* para passagem de tensão para seu dispositivo, cervo motor para a ação de abrir e fechar da porta e uma câmera para visualização de quem está na porta.

O funcionamento do sistema é realizado através da ativação da campainha, assim o sistema inicia, em seguida, a câmera é ligada e o proprietário recebe uma mensagem transmitindo para os visitantes através de um dispositivo móvel com auxílio da câmera. Para a destranca da porta, o sistema utiliza-se de uma aplicação *mobile*, em que o proprietário efetua a ação de destrancar, de trancar a fechadura. Além disso, é possível transmitir uma mensagem de áudio para o visitante, digitando o texto em seu aplicativo *mobile*.

Os resultados obtidos dos autores, mostram que conseguiram trancar a porta remotamente assim como proposto, quando pressionada a campainha o sistema ligou e realizou uma *stream*, assim o proprietário pode constatar a resposta da câmera ao vivo em seu aplicativo *mobile*, conseguindo identificar a pessoa e determinando a ação de abrir ou fechar a porta. A mensagem de voz emitida pelo aparelho, funcionou como o planejado.

No presente trabalho foi proposto construir à aplicação *mobile* para controle da fechadura e o cenário apresentado se assemelha ao proposto por Pinjala e Gupta (2019). Contudo, outro objetivo do trabalho proposto, é conseguir proporcionar acessibilidade através do comando de voz com o auxílio de assistente virtual, que se diverge da proposta desses autores, cuja

aplicação é voltada para segurança.

3.2 *Sistema De Gestão de Fechaduras Inteligentes Usando IoT para Aplicação em Cacifos de Universidade.* (MARGALHO, 2019)

Em Margalho (2019), a autora desenvolve trancas inteligentes para armários universitários. Para a elaboração desse dispositivo, foram utilizados, um microcontrolador para a comunicação sem fio ou via *ethernet*, um leitor *Radio Frequency Identification (RFID)* para leitura de cartões, um *display* numérico, isso alocado em um cacifo. Na fechadura estão conectados componentes controlados pelo microcontrolador, sendo eles, um atuador de bloqueio de porta, um sensor para detecção da posição da porta e uma mola que faz a posição da porta por padrão se manter fechada.

O comportamento do sistema é baseado no microcontrolador, conectar-se a *web* e recebe pedidos dos usuários, realizando a ação e desbloqueio ou bloqueio do cacifo. Requisições são feitas através de dispositivos moveis ou do leitor *(RFID)*, que está alocado na fechadura, quando uma requisição é feita, é checada a autentificação na *cloud* que utiliza o sistema de segurança *OAuth 2.0*, quando autorizada a requisição é enviada para o microcontrolador.

Com os resultados obtidos pela autora, foi possível determinar que o sistema elaborado é seguro e confiável, fornece aos usuários do sistema um conjunto de funções práticas, consistentes e com a usabilidade real no mercado. Apesar disso as comunicações mantidas na plataforma *cloud* entre o *hardware* e a autenticação do sistema a serem protegidas pelo *OAuth* 2.0, ainda são veneráveis. Sendo assim, foi perceptível que o sistema precisa de uma melhoria na segurança de transferência de dados.

O presente trabalho é semelhante ao trabalho de Margalho (2019), que também se utiliza da comunicação *Wi-fi*, utilizando-se do protocolo *MQTT* e desenvolve uma aplicação *mobile*. Porém, em Margalho (2019), ela faz o uso de sensores *RFID* para o controle de acesso a fechaduras, já neste trabalho utiliza controle de voz com o auxílio de um assistente eletrônico.

3.3 *Smart Anti-Theft Door locking System* (JAHNAVI; NANDINI, 2019)

Em Jahnavi e Nandini (2019), propõem um sistema de segurança usando reconhecimento de rosto integrado a uma placa de desenvolvimento *Raspberry Pi* para controlar fechaduras. No desenvolvimento do sistema foi usada uma *Raspberry Pi* com câmera para o

controle, filmagem e para detecção de rostos. A linguagem escolhida para programação na placa foi a *Python*, para abertura ou trancamento da porta é utilizado um motor de passos, e para detecção de obstáculos é selecionado o sensor *IR* (Infravermelho). Para o processo de reconhecimento de rosto é feito usando a ferramenta *MATLAB*.

O sistema opera da seguinte maneira, inicialmente, a câmera *Raspberry Pi* fica ociosa até que o sensor *Infrared radiation (IR)* detecte um obstáculo, e dado que o obstáculo é detectado, ele aciona a câmera para capturar a imagem do objeto. A imagem capturada é processada para realizar a detecção de rosto usando o algoritmo de *Viola Jones*. Visto que a comparação é feita, um sinal de permissão ou não permissão é enviado para a *Raspberry Pi*. Ao receber o sinal de permissão, o motor gira e a fechadura será aberta para a pessoa autenticada.

Os resultados gerados através de testes e de simulação indicam que o sistema consegue identificar rosto, assim funcionando de maneira esperada segundo os autores. Para reconhecimento de rosto, o método de padrão binário local é usado para extrair as características importantes de imagens faciais. Finalmente, o sistema verifica identidade de pessoas para melhorar o sistema de segurança da porta.

No presente trabalho faz o controle dos componentes como o trabalho em Jahnavi e Nandini (2019). Neste contexto, é utilizado um microcontrolador para o gerenciamento geral dos componentes. Apesar disso, as condições para a abertura da porta se diferencia, sendo em Jahnavi e Nandini (2019) o uso do reconhecimento facial pode ser uma condição para a abertura, já no trabalho proposto se trata reconhecimento de voz.

3.4 *IoT Enhanced Smart Door Locking System* (SHANTHINI *et al.*, 2020)

No trabalho, de Shanthini *et al.* (2020), é aprimorado a segurança do sistema de travamento de porta, utilizando uma fechadura convencional para abrir e fechar a porta, um servo motor para controlar o bloqueio, uma campainha piezo para detecção de intrusão, um microcontrolador Arduíno Uno para o processamento de dados, e um módulo *bluetooth* (*HC-05*) para a comunicação. Para a elaboração do código, foi utilizado o Arduino *IDE*, para armazenamento de dados em que é empregado o banco de dados Firebase.

O sistema opera da seguinte maneira, é realizado um cadastro de usuário para o gerenciamento da porta, cada usuário terá um credencial de *login* exclusivo e armazenado no banco de dados Firebase, os usuários podem solicitar a abertura ou fechamento da porta, as ações são verificadas no banco de dados pela *internet*, usando um dispositivo *mobile* com o aplicativo

instalado, posteriormente o aparelho se comunicará com o microcontrolador enviando sinais, através da comunicação *bluetooth*. Se as credenciais são inválidas, a campainha toca e um alerta *Short Message Service (SMS)* é enviado para o proprietário da fechadura.

Mediante aos resultados gerados, o autor pode concluir que o sistema consegue realizar o controle da fechadura através da aplicação *mobile*, também é capaz de armazenar os dados no banco, distinguindo se o usuário possui permissão ou não para o gerenciamento da porta.

O presente trabalho é semelhando ao trabalho em Shanthini *et al.* (2020), no controle da fechadura pela aplicação *mobile* e no uso do banco de dados Firebase. Contudo, este trabalho utiliza a tecnologia *Wi-Fi* para comunicação com o microcontrolador. Em Shanthini *et al.* (2020), é utilizado o *bluetooth* que restringe o acesso à porta em uma área limitada.

3.5 Análise Comparativa

Nesta Seção é abordado uma comparação entre trabalhos encontrados na literatura e a solução proposta.

Demonstra-se no Quadro 1 uma análise comparativa entre os trabalhos relacionados e o trabalho proposto. Os seguintes aspectos de cada trabalho são observados: utiliza *middleware* FIWARE; faz o uso de *WI-FI* para comunicação; utiliza plataforma de aplicação *mobile*; possui controle por voz. Entre todos os trabalhos relacionados nenhum utiliza o *middleware* FIWARE para o gerenciamento de dispostos *IoT* e controle de contexto, diferente deste.

Ainda é importante destacar que, apenas o de Shanthini *et al.* (2020) não utiliza o *WI-FI* como meio de comunicação com o aparelho de controle, já que o autor não está focando em um acesso remoto da fechadura, assim acabando utilizando para comunicação do aparelho de controle com a fechadura a tecnologia *bluetooh*, essa, por sua vez, é uma tecnologia presente na maioria dos dispositivos móveis.

Já no quesito de desenvolvimento de aplicação mobile, todos os trabalhos relacionados desenvolvem uma aplicação *mobile* para realizar o controle da fechadura. Isto se dar devido à praticidade de utilizar dispositivos móveis, pois, estes dispositivos são portáteis, ainda assim proporcionam diversas ferramentas para a comunicação, como *WI-FI*, *Blueetooh*, dentre outras.

Em todos os trabalhos relacionados nenhum se utiliza do controle de voz para acionar a ação de seu dispositivo, somente neste trabalho proposto, é utilizado o controle por voz como um dos meio de realizar o manuseio da fechadura, visando maior acessibilidade de pessoas, com

Quadro 1 – Comparação entre os trabalhos.

Trabalho	**Utiliza *middleware* FIWARE**	**Faz o uso de Wi-FI para comunicação**	**Plataforma de aplicação *mobile***	**Possui controle por voz**
Pinjala e Gupta (2019)	Não	Sim	Sim	Não
Margalho (2019)	Não	Sim	Sim	Não
Jahnavi e Nandini (2019)	Não	Sim	Sim	Não
Shanthini *et al.* (2020)	Não	Não	Sim	Não
Trabalho Proposto	Sim	Sim	Sim	Sim

Fonte: elaborado pelo autor.

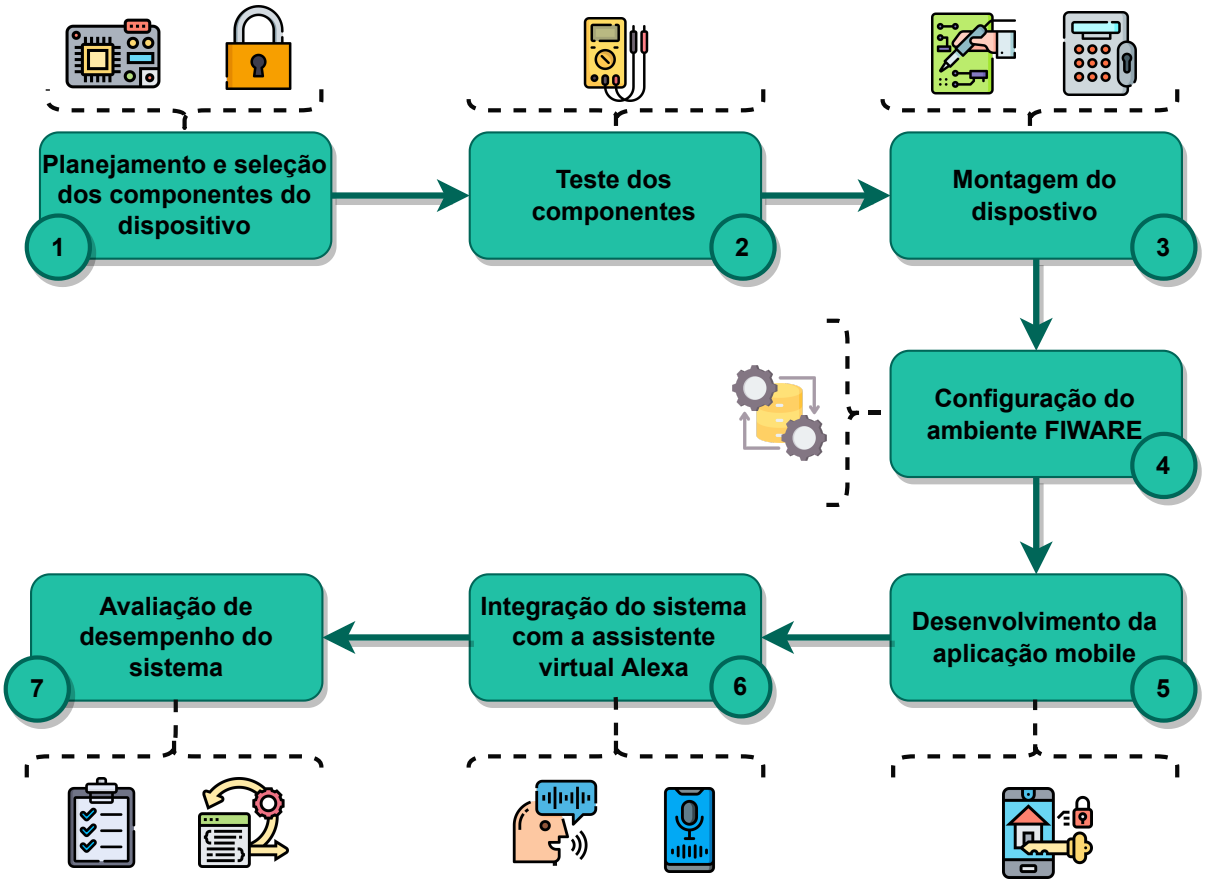
alguma dificuldade em utilizar um dispositivo móvel, além de proporcionar maior conforto para qualquer pessoa.

No quesito de utilizar o *middleware FIRWARE* não há nenhum trabalho utilizando, apenas o trabalho proposto, FIWARE é empregado neste trabalho para realizar a comunicação entre as partes do projeto e o gerenciamento dos dados de contexto. Montando uma arquitetura de camadas, sendo mais simples de identificar a relação daquela fechadura com o ambiente.

4 PROCEDIMENTOS METODOLÓGICOS

É descrito neste capítulo as etapas que foram executadas para desenvolver o equipamento proposto. A concepção deste projeto executará a sequência de cada tarefa, conforme mostrado na Figura 5.

Figura 5 – Fluxograma dos procedimentos.



Fonte: Elaborado pelo próprio autor

Os detalhes de cada etapa fluxograma, começando da Seção 4.1, explica a parte de seleção dos componentes do dispositivo, até a Seção 4.7, descreve como realizar a verificação do protótipo.

4.1 Planejamento e seleção dos componentes do dispositivo

A princípio é realizado um estudo e seleção das tecnologias para comunicação e, em seguida, o *hardware* necessário para o controle da fechadura inteligente. Para a transmissão de dados o protocolo *MQTT* se mostrou bastante útil, por se tratar de um protocolo leve e rápido ele é adequado para quando estamos tratando de aplicações *IoT*. Já para a comunicação o FIWARE

é um *middleware* altamente conveniente para o tratamento da interoperabilidade dos dispositivos *IoT* e gerenciamento das informações de contexto.

Além disso, é preciso definir os componentes de *hardware* que foram empregados nessa solução. Para a fechadura está sendo realizado um estudo para definir qual a melhor fechadura eletrônica que se encaixa com o projeto. Pontos que estão sendo considerados para analisar as fechaduras são: custo da fechadura, voltagem de alimentação, tempo de resposta, categoria de mecanismo e se é possível realizar a abertura sem o uso de energia e com uso de chave, por exemplo. Após o estudo da fechadura ser realizado, um relé adequado foi definido para a passagem de voltagem para a mesma. No controle, os microcontroladores da família *ESP* apresentam-se muito relevantes para o desenvolvimento deste projeto, pois já possuem comunicação *WI-FI*, facilitando e muito o trabalho de comunicação entre os dispositivos.

Ademais, uma análise de qual assistente virtual, melhor se encaixa para o intuito deste dispositivo foi realizada. Além disso, alguns pontos foram considerados, tais como: programabilidade, recursos disponíveis para uso e acessibilidade a esse assistente. Alexa, a assistente virtual desenvolvida pela Amazon, demonstra ser bastante vantajosa para ser utilizada neste trabalho, pois possui uma plataforma bastante amigável para o desenvolvimento de novas aplicações e apesar de não está disponível por padrão em alguns *smarthphones*, é possível integrá-la como sua principal assistente.

4.2 Testes dos componentes

Após selecionar os materiais usados para construir o protótipo, cada componente foi testado para verificar se estão funcionando corretamente. Nesta fase, os componentes elétricos precisam ser analisados para verificar se não estão danificados de alguma forma. Para fazer isso, é necessário verificar se o circuito está funcionando corretamente, mostrando a corrente e a tensão necessária em cada extremidade, e se as conexões elétricas presentes em cada dispositivo estão boas.

4.3 Montagem do Dispositivo

Após as verificações dos materiais usados para o dispositivo, a próxima etapa é montar o dispositivo de fechadura eletrônica que está sendo proposto. Para realizar a montagem é preciso selecionar onde foram alocados cada componente, para que todas as conexões necessárias

entre esses elementos possam ser realizadas e assim a fechadura funcione de forma satisfatória.

4.4 Configuração do ambiente FIREWARE

Em seguida a montagem do dispositivo foi a vez de realizar a configuração do ambiente FIWARE, que se trata o *middleware* utilizado na solução, sendo um intermediário de comunicação entre as aplicações de comando e a placa realiza a ação. Para a realização deste passo foi necessário utilizar a ferramenta denominada *Docker* que se trata de um conjunto de produtos de plataforma como serviço que usam virtualização de nível de sistema operacional para entregar software em pacotes chamados contêineres.

Através deste programa foi realizado o *download* das imagens *docker*, utilizadas para a configuração do ambiente, sendo elas: *mongoDB*, *mosquitto*, *Orion* e *Iot Agent*. Em seguida ao *downloads* das imagens, foi feita a configuração do ambiente realizando requisições *REST* para a configuração do *Broker*, e para a criação de entidades.

4.5 Desenvolvimento da aplicação *mobile*

Em seguida a montagem do dispositivo, é necessário desenvolver uma aplicação *mobile*. Está se comunicar com o *middleware* FIWARE e, posteriormente, com o dispositivo. Para o desenvolvimento da aplicação, foi utilizado o ambiente de desenvolvimento *Android Studio*, através da linguagem de programação *Flutter*. Durante o desenvolvimento do aplicativo *Android* foram implementados e testados os seguintes passos:

1. Interface de *login*: para que o usuário consiga entrar e efetuar o controle da fechadura;

2. Tela de controle de fechadura: onde mostra a fechadura e se pode controlá-la;

3. Tela de adicionar fechadura: nesta tela o usuário consegue adicionar novas fechaduras para que posteriormente consiga realizar o controle destas.

4.6 Integração do sistema com assistente virtual Alexa.

Depois do desenvolvimento da aplicação *mobile*, foi realizada a integração da assistente virtual Alexa com a aplicação *mobile*. A integração ocorre no desenvolvimento de novas

skills. Essas *skills* são usadas para criar habilidades ou integrações com outros aplicativos. No desenvolvimento da *skill* foi analisado meios de segurança, de modo que usuários indesejados não consigam ter acesso ao controle da fechadura.

4.7 Avaliação de desempenho do sistema.

Posteriormente ao desenvolvimento da aplicação *mobile* e a integração com a assistente virtual, foi realizada uma sequência de testes com o intuito de avaliar o desempenho do dispositivo, tanto no *hardware* como na aplicação e na integração. Algumas das caraterísticas observadas são: a fechadura está realizando corretamente os comandos destinados a ela; tempo de resposta do comando realizado até a ação ser devidamente executada; controlar portas em que não tem acesso; entre outros.

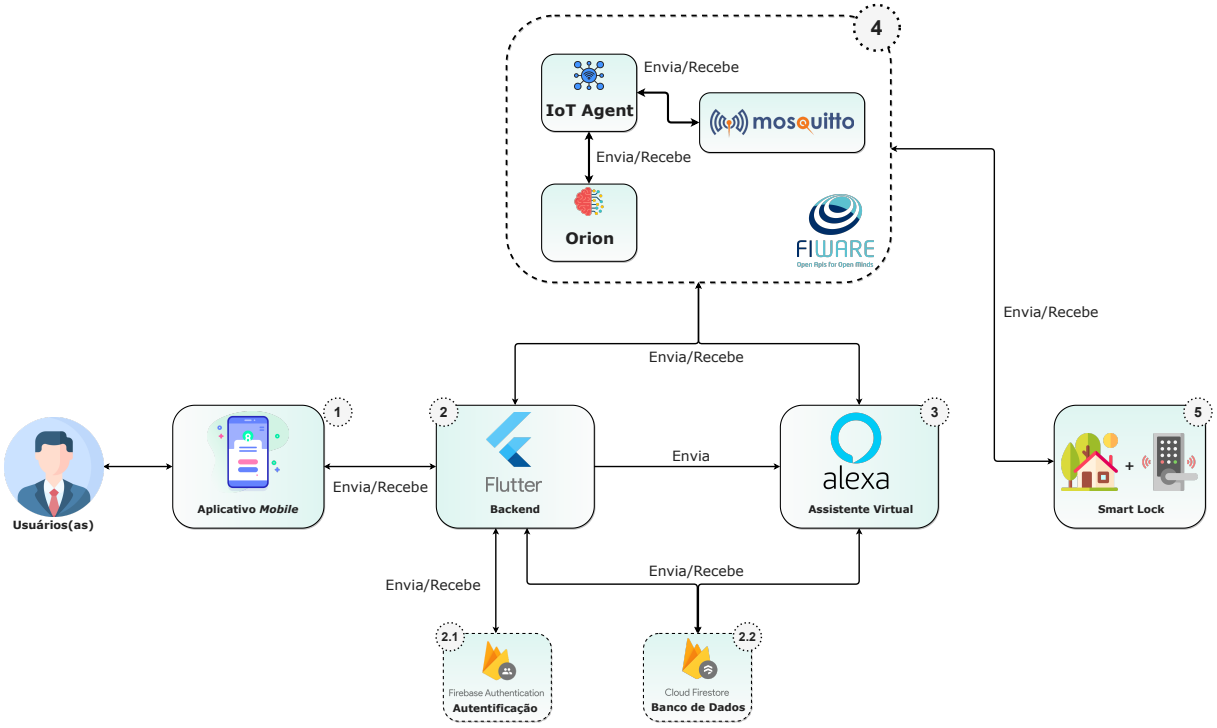
5 MATERIAIS E MÉTODOS

Neste capítulo, é especificado a solução proposta para o protótipo do sistema *Smart Lock*. As seções tratam os detalhes sobre os métodos, os componentes e as implementações, usufruídos para a integração e o funcionamento adequado do sistema proposto.

5.1 Modelo do sistema Smart Lock

Propões atraves deste trabalho uma solução para o gerenciamento e controle de fechaduras eletrônicas no cenário IoT, que, além disso, possibilita o controle da fechadura por voz, com o auxílio de assistentes eletrônicos. Na Figura 6 é ilustrado o fluxo de etapas que mostra o percurso dos dados.

Figura 6 – Fluxograma do Sistema Smart Lock.



Fonte: Elaborado pelo próprio autor

Realizando a análise da Figura 6, da esquerda para direita, pode-se analisar em (1), que o usuário tenta se conectar diretamente no aplicativo desenvolvido através de um aparelho móvel.

O (2) é o *backend* responsável por gerenciar as ações realizadas pelo usuário, fornecendo servidos de *login* e cadastro de usuário. Para a autenticação é empregado o (2.1), *Firebase auth* que cuida do gerenciamento de usuário, tanto para cadastrado e para *login*. Caso o usuário

consiga realizar o *login* ou o cadastro, ele tem acesso à tela de fechaduras, que estão associadas e armazenadas no (2.2), quando o usuário está logado em sua conta, ele pode vincular sua conta com um dispositivo, Alexa (3), que tendo se vinculado também tem acesso ao banco de dados (2.2), que assim possibilita o envio de comandos para as fechaduras corretas.

Na etapa (4), a solicitação é feita pela aplicação *mobile* no *backend* (2) ou Alexa (3), e o (4), representado pelo fluxo do *middleware* FIWARE, processa essa solicitação e encaminha para o sistema embarcado (5), onde tem uma *Smart Lock*. Após a fechadura receber essa requisição ela efetua a ação desejada, tranca ou destranca, e então encaminha (4) a sua ação para o *backend* da aplicação móvel (2), em seguida é obtido o *status* dessa fechadura, caso tenha êxito nessa etapa, ele registra os *status*.

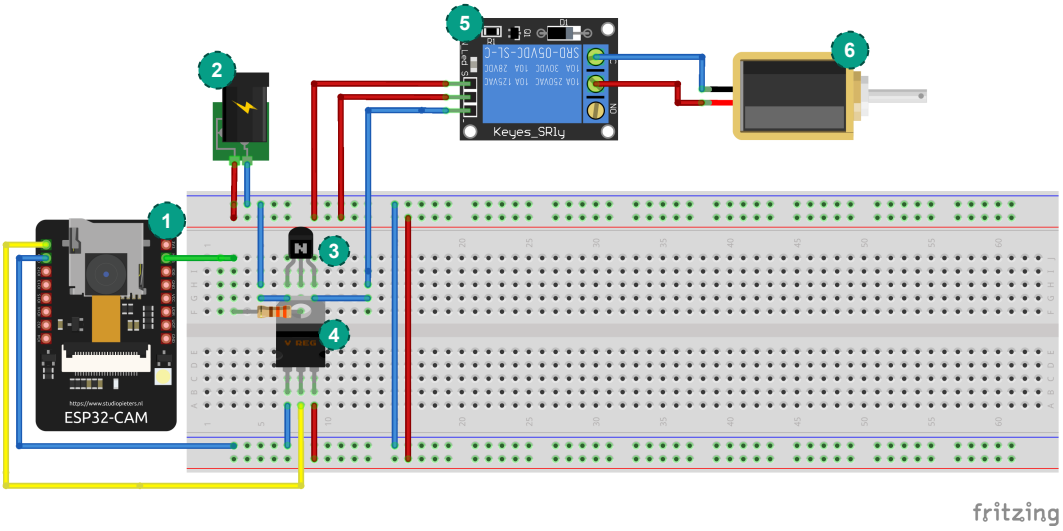
5.2 Protótipo embarcado

Para o desenvolvimento do protótipo da *Smart Lock*, inicialmente foi necessário definir e listar os componentes necessários para o real funcionamento da fechadura, além disso, foi elaborado um esquemático utilizando o programa *Fritzing*. Que se trata de um software de código aberto para *design* de *hardware* eletrônico, o programa possui alguns módulos que podem ser utilizados para o desenvolvimento do projeto. Na Figura 7 é exibido o circuito prototipado do da *Smart Lock*. Na figura tem vários componentes específicos que vão auxiliar no comportamento. do sistema, esses componentes estão listados como:

1 . **Módulo ESP32Cam:** é utilizado para realizar o controle e comunicação entre os componentes;
2 . **Fonte de alimentação 12v:** utilizada para passar energia para o circuito como um todo;
3 . **Transistor NPN BC548:** é o mediador, onde permitirá chavear o microcontrolador com o relé, permitindo que o sinal do pino do microcontrolador acione o relé quando fechado o circuito;
4 . **Regulador de tensão 7805:** é o 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**: é o atuador capaz de inibir ou permitir a passagem de corrente quando alimentado.
6 . **Fechadura Solenoide 12v:** é o atuador que haje como uma fechadura que se mantém fe-

chada quando não a alimentação e aberta (ou retraída), quando a passagem da alimentação adequada.

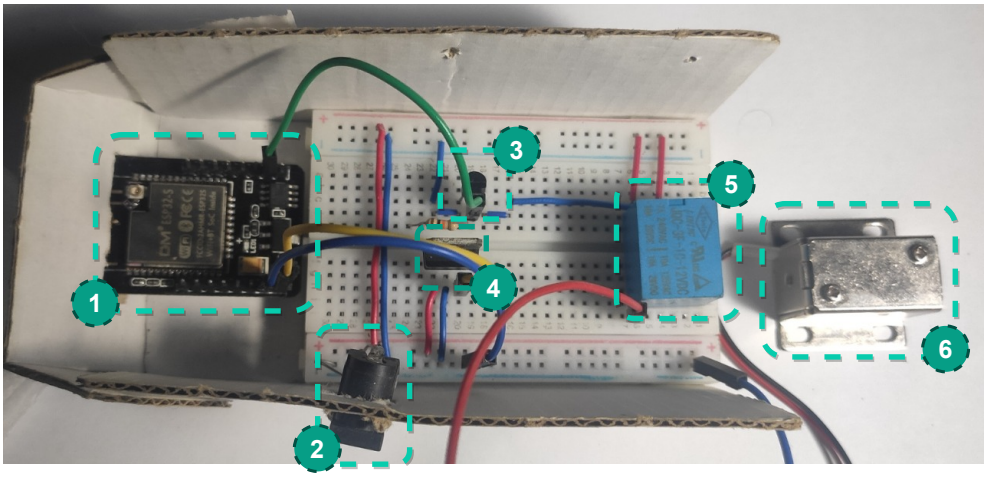
Figura 7 – Protótipo *Smart Lock* no *Fritzing*.



Fonte: Elaborado pelo próprio autor

Na Figura 8 é ilustrado o protótipo com todos os componentes mencionados nesta seção. Uma modificação no circuito *Fritzing* é necessária para que no protótipo construído seja adicionado um conector P4 (2), que se trata do padrão mais bem utilizado entre fontes de alimentação, para uma conexão mais segura e rápida. Na Figura 8 é mostrado os seguintes itens: Esp32Cam (1), regulador de tensão (4), transistor (3), relé (5) e, por fim, o solenoide 12V (6). Todos estão conectados por uma *protoboard* de 14x30, e por uma placa de ensaio na montagem de circuitos eletrônicos.

Figura 8 – Protótipo Smart Lock construído.



Fonte: Elaborado pelo próprio autor

A implementação do algoritmo deve permitir o bloqueio do pino responsável pelo sinal e, além disso, garantir que o microcontrolador se conecte à rede *Wi-Fi* e a FIWARE, intermediado por meio do *MQTT*. Dessa forma, o Código 1 é responsável pela conexão da *Wi-Fi* e o dispositivo criado no FIWARE.

Algoritmo 1: Código de configuração do Wi-Fi e Broker *MQTT*

```
#include <WiFi.h>

#include <PubSubClient.h>

#define pinFechadura1 16 //Pino da Fechadura

//WiFi
const char* SSID = "******";     //SSID/Nome da rede WiFi que
   deseja se conectar
const char* PASSWORD = "******";   //Senha da rede WiFi que
   deseja se conectar

//MQTT Server
const char* BROKER_MQTT = "localhost"; //URL do broker MQTT que
    se deseja utilizar
int BROKER_PORT = 1883;              // Porta do Broker MQTT

//Topico para mandar status da placa
#define TOPIC_PUBLISH "/(chave do Fiware)/(dispositivo criado
   no fiware)/attrs"

 //Topico que recebe comandos
#define TOPIC_SUBSCRIBE "(chave do Fiware)/(dispositivo criado
   no fiware)/attrs"
```

Fonte: elaborado pelo autor.

No FIWARE, a publicação realizada pelo MQTT segue um protocolo específico. O Ultralight, um protocolo leve baseado em texto destinado a controlar equipamento. Sua sintaxe é definida como "dispositivo | comando", conforme mostrado no Código 2, que exemplifica o bloqueio e desbloqueio de uma sala. A mensagem pode ser verificada na linha (2) do código, ou seja, no caso do comando "fechar"na fechadura da "sala", idêntico.

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

```
1
2       if (msg == "quarto|fechar") { //Checagem se o comando recebido
            coincide com um comando valido.
3
4           //Comando para desativar o solenoide/trancar a fechadura
5           digitalWrite(pinFechadura1, LOW);
6
7           //Mandado para o MQTT que a fechadura está trancada
8           MQTT.publish(TOPIC_PUBLISH, "quarto|trancada");
9       }
10
11      if (msg == "quarto|abrir") {
12          //Comando para ativar o solenoide/abrir a fechadura
13          digitalWrite(pinFechadura1, HIGH);
14
15          //Mandado para o MQTT que a fechadura está destrancada
16          MQTT.publish(TOPIC_PUBLISH, "quarto|destrancada");
17      } recebe comandos
```

Fonte: elaborado pelo autor.

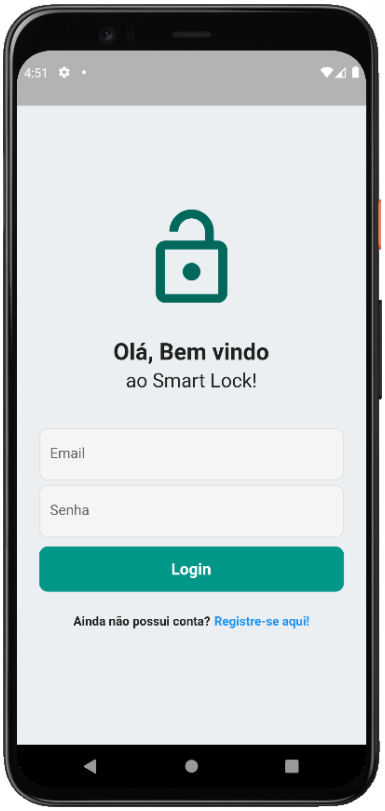
5.3 Protótipo aplicação *mobile*.

O protótipo desenvolvido tem a função de gerenciamento e controle de todo o fluxo da *Smart Lock*. Através da aplicação *mobile* é possível realizar o cadastro de usuários, *login*, cadastro de fechadura e vinculação da aplicação com uma conta Amazon Alexa. Para o controle de fechaduras e usuários cadastrados, é utilizado o banco de dados Firestore Database, que se trata de um banco de dados nosql, fornecido pela Firebase, que no que lhe concerne não há de uma formatação tão complexa para a criação de suas entidades e armazenamento de dados, assim facilitando o armazenamento e provendo fáceis modificações.

5.3.1 Login/Cadastro de usuários.

O aplicativo permite que o usuário logue na aplicação através de um *e-mail* e uma senha, conforme apresentado na Figura 9. Após a tentativa de *login*, os dados são verificados se há algum usuário que possua esses dados cadastrados na aplicação, se sim o usuário é redirecionado para a *dashboard* das fechaduras.

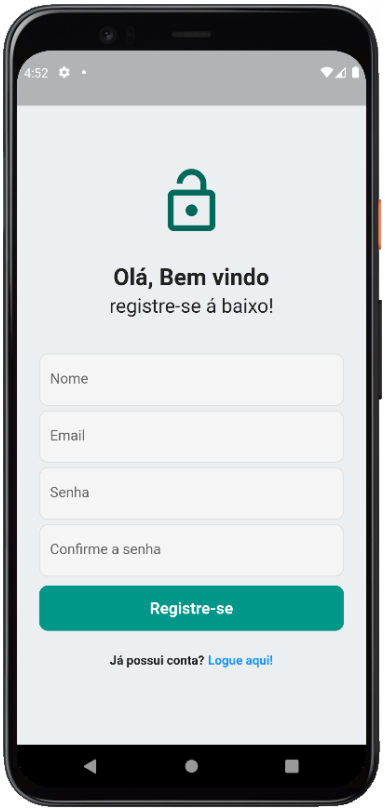
Figura 9: Tela de *login* da aplicação *Smart Lock*.



Fonte: Elaborado pelo próprio autor

Caso ainda o usuário em questão não possui cadastro, há possibilidade dele se registrar com nome, *e-mail* e senha como se mostra a Figura 10. Esse cadastro pode ser realizado por qualquer pessoa com acesso à aplicação, posteriormente é checado se o e-mail possui um formato valido, se não está cadastrado e se as senhas digitadas coincidem, caso não haja problema é realizado o cadastro de usuário, e o mesmo é redirecionado para a *dashboard* das fechaduras.

Figura 10: Tela de registro da aplicação *Smart Lock*.



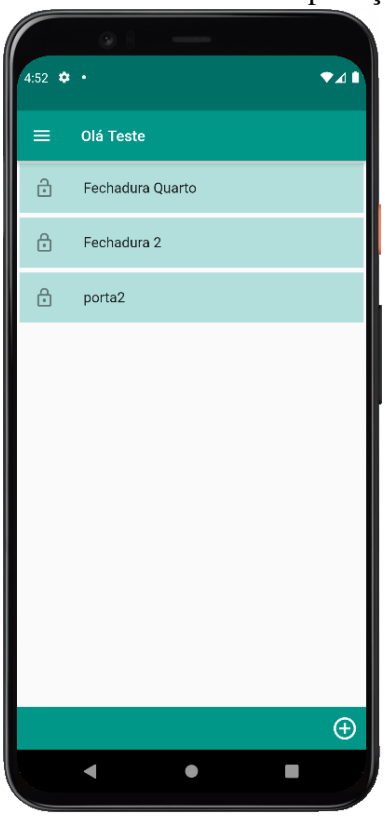
Fonte: Elaborado pelo próprio autor

5.3.2 Dashboard das fechaduras.

É ilustrado na 11, a tela de *dashboard* das fechaduras, oque exibe para o usuário as fechaduras que o mesmo tem acesso. Nessa tela é mostrado o nome da fechadura cadastrada e seus *status* atual, nessa tela há a possibilidade de realizar o *logout* da conta.

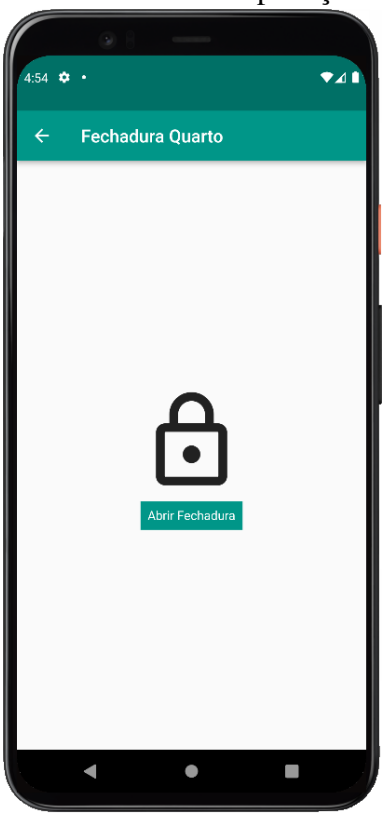
Quando é realizado um clique em cima de uma fechadura em questão, o usuário é redirecionado para a tela de controle de fechadura como é mostrado a Figura 12. Essa tela dispõe de um botão, com sua função de abrir/fechar a fechadura, e uma imagem de uma fechadura que indica o *status* atual da fechadura em questão.

Figura 11: Tela de *Dashboard* de fechaduras da aplicação *Smart Lock*.



Fonte: Elaborado pelo próprio autor

Figura 12: Tela de controle da fechadura da aplicação *Smart Lock*.



Fonte: Elaborado pelo próprio autor

5.3.3 Criação de fechadura

Dentro da tela de *dashboard* ilustrada na Figura 11, há a possibilidade da criação de fechadura através do botão (+), que se encontra no canto inferior direito. Quando pressionado o usuário é redirecionado para a página de cadastro de fechadura como se apresenta na Figura 13. Nessa página é possível realizar um cadastro de fechadura, informando o nome da fechadura, o local onde se encontra, e o *Identity Document* (ID) da fechadura em questão, caso não haja nenhuma fechadura com o mesmo ID já registrada, a fechadura é criada e o usuário é redirecionado para a *dashboard* de fechadura que é visto na Figura 11.

Figura 13: Tela de cadastro de fechaduras da aplicação Smart Lock.

Fonte: Elaborado pelo próprio autor

O FIWARE também é configurado nesta etapa pelo *backend* do sistema, inicializando os serviços necessários para estabelecer a comunicação entre os Orion *Context Brokers* e dispositivos conectados ao *Broker MQTT Mosquitto* via *IoTAgent*. Primeiro, é preciso que haja a configuração do *IoT Agent* para a viabilidade de inscrição em tópicos que os dispositivos poderiam se inscrever para realizar a comunicação. O *IoT Agent* atuará como um *middleware* entre esses dispositivos e agentes *IoT*, neste caso uma mensagem de uma aplicação *mobile*. Portanto, ele precisa criar a entidade dados de contexto com um ID. A configuração do *middleware*

é feita pela solicitação REST no *backend* que é exibido no Algoritmo 3, assim viabilizando uma comunicação com os dispositivos *IoT* com o *IoT Agent*.

Algoritmo 3: Código em *Dart* para executar configuração do *Broker* e *Iot Agent*.

```
Future<void> brokerConfig(String houseId) async {
  Map<String, String> headers = {
    'Content-Type': 'application/json',
    'fiware-service': 'openiot',
    'fiware-servicepath': '/'
  };
  var url = Uri.parse('http://$houseId:4041/iot/services');
  final data = {
    'apikey': apiKey,
    'cbroker': 'http://orion:1026',
    'entity_type': 'SmartLock',
    'resource': '/iot/json'
  };
  var response = await http.post(url, headers: headers, body: data);

  if (response.statusCode == 200) {
    print('Mensagem Recebida');
  } else {
    throw Exception('Falha ao enviar mensagem');
  }
}

```

Fonte: elaborado pelo autor.

Depois da configuração do ambiente é realizada a criação das entidades dos dispositivos, a criação da entidade funciona similar a uma classe em outras linguagens de programação, através delas é possível definir parâmetros e características que os dispositivos tem. Então quando uma fechadura é cadastrada, ele é cadastrada como uma entidade do tipo *Smart Lock*, como mostrada na linha 11 do Código 4, seu ID atribuído vira "lock#" concatenado com o id da fechadura que está registado no banco de dados, essa entidade tem a função de guardar o contexto e a troca de mensagem entre o dispositivo e a aplicação. Além disso, é atribuído um nome ao dispositivo, o nome registrado na fechadura em questão.

Algoritmo 4: Código em *Dart* para criação da entidade *Smart Lock* e do dispositivo no *IoT Agent*.

```
1   Future<void> entityConfig(String lockId, String houseId) async {
2     Map<String, String> headers = {
3       'Content-Type': 'application/json',
4       'fiware-service': 'openiot',
5       'fiware-servicepath': '/'
6     };
7     var url = Uri.parse('http://$houseId:4041/iot/services');
8     final data = {
9         "device_id": "Lock#$lockId",
10        "entity_name": "urn:ngsi-ld:Lock:$lockId",
11        "entity_type": "SmartLock",
12        "protocol": "PDI-IoTA-UltraLight",
13        "transport": "HTTP",
14        "endpoint": "http://iot-sensors:3001/iot/Lock$lockId",
15        "commands": [
16          {"name": "abrir","type": "command"},
17          {"name": "fechar","type": "command"}
18         ],
19         "attributes": [
20          {"object_id": "s", "name": "state", "type":"String"}
21         ],
22         "static_attributes": [
23           {"name":"refHouse", "type": "Relationship","value": "urn:
    ngsi-ld:House:$houseId"}
24        ]
25    };
26    var response = await http.post(url, headers: headers, body: data);
27    if (response.statusCode == 200) {
28      print('Mensagem Recebida');
29    } else {
30      throw Exception('Falha ao enviar mensagem');
31    }
32  }
33
```

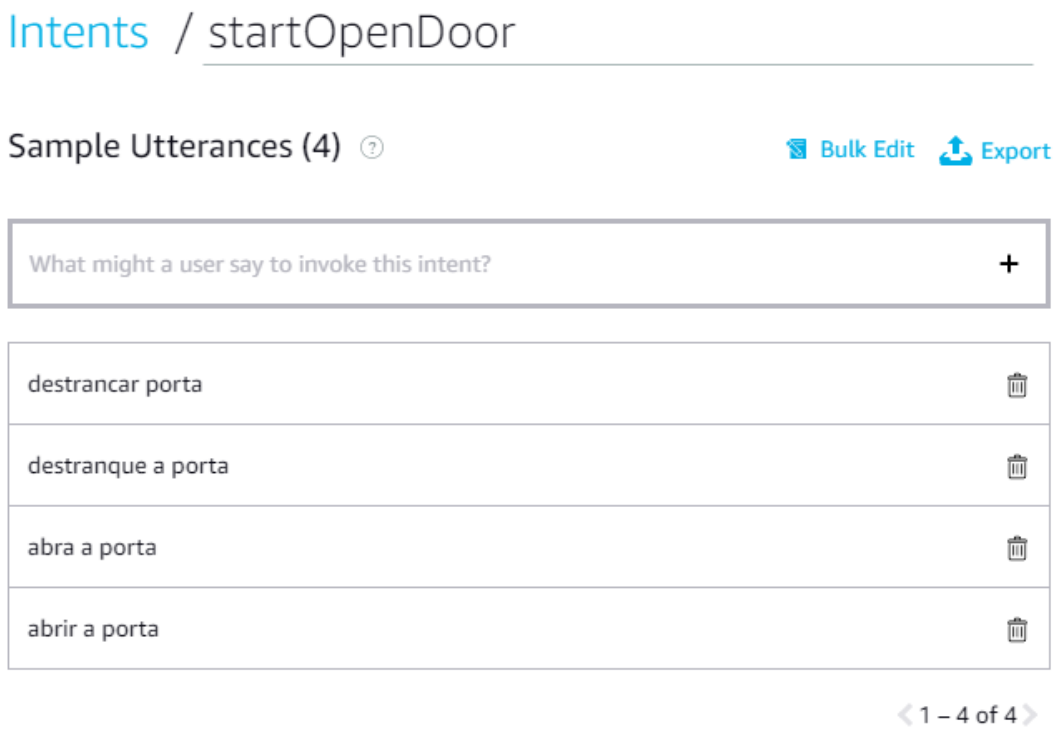
Fonte: elaborado pelo autor.

5.4 Integração com assistente virtual Alexa.

A assistente virtual Alexa empregada no projeto *Smart Lock*, tem a função de assimilar a voz de um usuário e traduzi-la em texto para poder ser utilizada para o controle da fechadura. Para essa assimilação é necessário desenvolver uma "*skill*" que é como é chamado aplicações na assistente, através da criação dessas *skills* se consegue usar a voz de um usuário para a ativação de funções.

Quando criado uma *skill* é desenvolvido funções na Alexa através do uso de frases-chave e algoritmos escritos na linguagem de programação JavaScript. As frases-chave ou frases de ativação, são definidas como *Intents* que são variáveis que englobam várias chaves que ativam a mesma, como mostrado na Figura 14. Nesse contexto, foi criado o *intent* com o nome *startOpenDoor*, que armazena frases que aplicam o contexto de abrir a porta. Essas criações permitem adicionar diversas frases para essa ativação.

Figura 14: Criação de *Intentes* no console da assistente virtual Alexa.



Fonte: Elaborado pelo próprio autor

Apos a criação da *skill* e criação de alguns *intents*, foi prosseguindo para a criação do código que irá reconhecer a voz e realizar o comando de ativação da fechadura. Nos Códigos 5 e 6, apresentam-sew as funções referentes a aberta ou fechadura da porta, sucessivamente, no Código 5, pode-se ver na linha 4 que esse checa se á uma requisição realizada pelo *intnet*

denominado ‘*startOpenDoor*’, como mostrado na Figura 5.

Assim, isso mostra que essa função é ativada quando há uma invocação utilizando aquelas frases como “abrir porta” ou “abra porta” anteriores mostradas. Quando reconhecida como um comando válido o código vai para a linha 6, onde é realizado as funções a ele empregadas, no caso do Algoritmo 5, é guardado em uma variável uma mensagem que irá ser falada caso ocorra corretamente. Que se trata de um retorno sonoro dizendo que está realizando a ação de abrir a porta/fechadura concatenado com o nome da fechadura.

Em seguida é feito o envio do comando para a fechadura via MQTT, utilizando ao padrão do protocolo Ultralight. Como pode-se ver na linha 9, sendo **lockPath**, o destino da mensagem, onde tem seu único caminho, armazenado no banco de dados, e em seguida a mensagem em questão, se tratando comando de abrir porta.

Algoritmo 5: Função em JavaScript para abrir a porta usando por meio da voz utilizando assistente virtual Alexa

```
1     const OpenDoorIntentHandler = {
2       canHandle(handlerInput) {
3           return (handlerInput.requestEnvelope.request.type === '
    IntentRequest'
4           && handlerInput.requestEnvelope.request.intent.name === '
    startOpenDoor');
5       },
6       async handle(handlerInput) {
7         let outputSpeech = 'Certo, abrindo ' + lockName;
8
9         sendToMqtt(lockPath, "quarto|abrir");
10
11        return handlerInput.responseBuilder
12          .speak(outputSpeech)
13          .reprompt(outputSpeech)
14          .getResponse();
15      },
16    };
17
```

Fonte: elaborado pelo autor.

No Código 6, pode-se observar a função de trancar/fechar a porta, sendo bem semelhante ao algoritmo anterior, mas alterando na linha 4, onde é feita a requisição do *intenet startCloseDoor*, que possui frases para fechamento da porta, como, por exemplo: “fechar porta”, “feche porta”. Portanto, a mudança da mensagem retornada após o comando, na linha 7, e o comando para fechar a porta na linha 9.

Algoritmo 6: Função em *JavaScript* para fechar a porta usando por meio da voz utilizando assistente virtual Alexa.

```
1     const CloseDoorIntentHandler = {
2       canHandle(handlerInput) {
3           return (handlerInput.requestEnvelope.request.type === '
    IntentRequest'
4           && handlerInput.requestEnvelope.request.intent.name === '
    startCloseDoor');
5       },
6       async handle(handlerInput) {
7         let outputSpeech = 'Certo, fechando ' + lockName;
8
9         sendToMqtt(lockPath, "quarto|fechar");
10
11        return handlerInput.responseBuilder
12          .speak(outputSpeech)
13          .reprompt(outputSpeech)
14          .getResponse();
15      },
16    };
17
```

Fonte: elaborado pelo autor.

6 EXPERIMENTOS E RESULTADOS

Neste capítulo são relatados os experimentos realizados para verificar o comportamento do dispositivo desenvolvido em diferentes cenários. Os experimentos são tratados para testar o controle de acesso pela voz do usuário, reconhecimento de voz, e tempo de reposta entre cada dispositivo do sistema *Smart Lock*. Para isso, é considerada a arquitetura montada por todo o sistema e suas principais funções.

6.1 Configuração dos Experimentos

Descreve-se no Quadro 2, brevemente, os experimentos realizados para o sistema proposto. Para a geração dos resultados, a própria implementação é utilizada para simular ou apresentar dados reais obtidos do sistema. Os experimentos também consideram o fluxo e o comportamento do sistema, desde a configuração do aplicativo móvel até o sistema embarcado.

Quadro 2: Descrição dos Experimentos

Critérios	**Descrição**
Sistema	Infraestrutura baseada no *Flutter* e no sistema embarcado.
Métricas	Comando por voz e tempo de resposta.
Parâmetros	Requisições, confiança e tempo de resposta.
Fatores	Ambiente, oscilação na rede e respostas das APIs.
Projeto de Experimentos	**Experimento 1:** Verificação do tempo de resposta da comunicação dos dispositivos; **Experimento 2:** Comando de voz e acesso as fechaduras.

Fonte: elaborado pelo autor.

6.1.1 Configuração do Experimento I

Para o primeiro experimento foi utilizado a ferramenta *Open Source*, Postman, é uma plataforma de API para desenvolvedores projetar, construir, testar e iterar aplicações. Foi realizado um salvamento de requisições HTTP e HTTPs no PostMan, com intuito de enviar e receber respostas via *HTTP*. Com isso, foi utilizado a funcionalidade *Runner* mostrada na Figura 15, que executa uma série de requisições do número de vezes indicado em seu parâmetro. Com o uso dessa ferramenta há a facilidade de execução de um conjunto de requisições desejadas, além de possuir um registro de seus resultados após a execução dos mesmo, com a opção de*"save responsee"*. Possibilitando a comparação e análise entre um tempo de reposta e outro. Assim, com o emprego do Postman é possível realizar diversas e requisições e tanto *HTTP* e *HTTPs* e analisá-las de maneira mais coesa. Sendo assim, o Quadro 3 mostra a quantidades de requisições

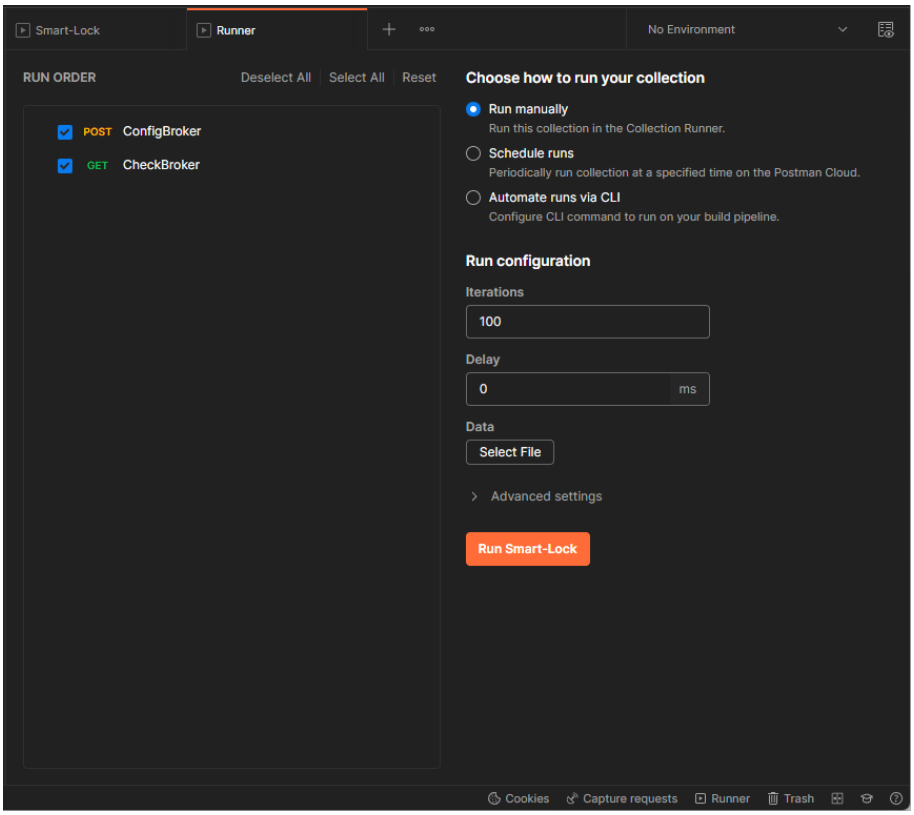
feitas para cada etapa da aplicação, Alexa e FIWARE.

Quadro 3: Configuração para experimento 2 para tempos de resposta.

	Número de requisições	**GET**	**POST**	**PUT**
Aplicação	100	0	100	0
Alexa	100	0	100	0
FIWARE	200	80	80	40

Fonte: elaborado pelo autor.

Figura 15: Ferramenta *Runner* do Postman.



Fonte: Elaborado pelo próprio autor

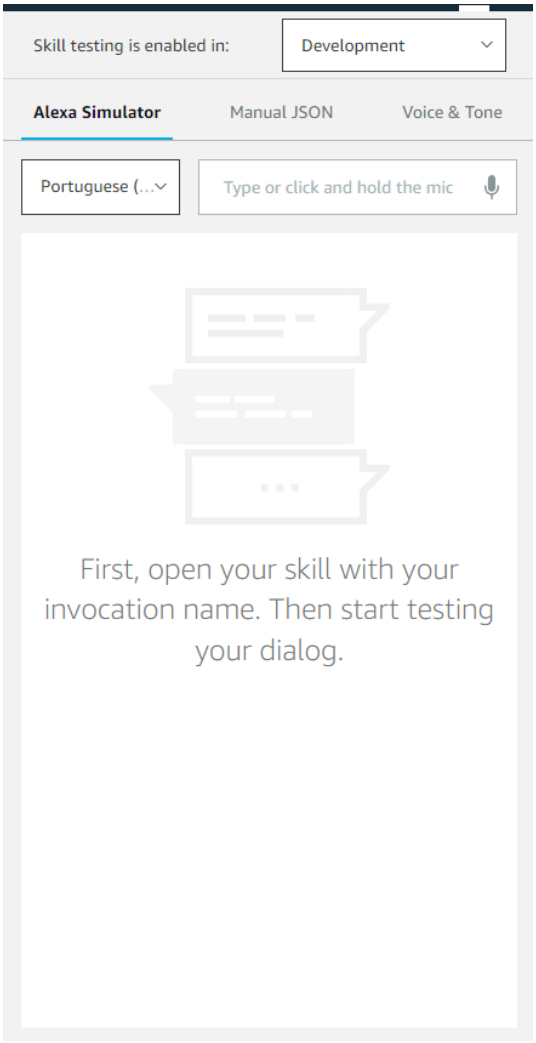
6.1.2 Configuração do Experimento II

Para o segundo cenário foi configurado dois usuários na aplicação *Smart Lock*, sendo um deles com cadastro de uma fechadura vinculado a sua conta, e outro sem ao acesso da mesma. Os dois usuários possuem acesso ao assistente virtual Alexa, via celular ou outro meio, os dois possuem acesso à aplicação *Smart Lock* em suas aplicações Alexa, e tentam ter acesso a uma fechadura em questão. Para a experimentação desse cenário foi usado o *console* da Alexa, que se trata de um *console* de desenvolvimento que prover ferramentas para testes de suas aplicações.

Dentro do *console* foi utilizado a função ***Test -> Alexa Simulator***, que se trata de

um simulador que transforma texto em fala para Alexa, como se mostra na Figura 16. Assim, para esse teste foi utilizada uma conta com acesso a uma fechadura em questão, para simular o usuário com acesso à fechadura, depois revogado o acesso da conta e da fechadura, para simular o usuário sem acesso, mas com acesso à aplicação, onde foram testando a tentativa de acesso à fechadura.

Figura 16: Ferramenta *Alexa Simulator* no *Console* de desenvolvimento Alexa.



Fonte: Elaborado pelo próprio autor

6.2 Resultados dos Experimentos

Nesta seção foram abordados os resultados dos experimentos e cenários abordados na seção 6.1. Nos experimentos foram geradas figuras e gráficos dos resultados, para melhor visualização.

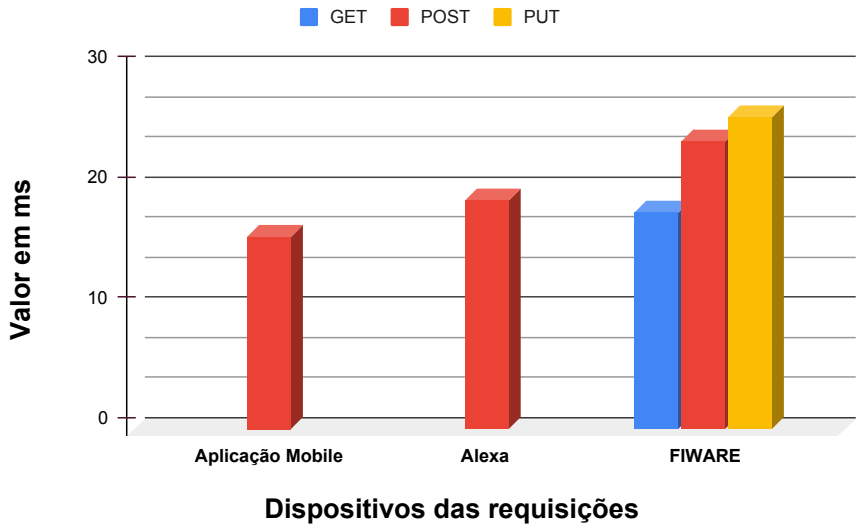
6.2.1 Resultados do Experimento I

Para os resultados do primeiro experimento ilustrado na Figura 17, um gráfico dos resultados das operações realizadas. No eixo y representa uma escala de tempo em milissegundos (ms), e no eixo x cada ferramenta que realizou certa requisição, onde é mostrado no gráfico o tempo médio da reposta, assim é possível perceber que as requisições tiveram tempos de repostas similares, observando a aplicação e a Alexa. Dessa forma, é notado uma diferença muito baixa se tratando da mesma ação, no caso o *POST*, isso foi possível, pois os dois executam quase a mesma função, só alterando qual linguagem e biblioteca foi utilizada para realizar essa requisição *REST*; por isso, há essa baixa diferença.

Agora partindo para os resultados do FIWARE é obtido os maiores resultados da latência do tempo de reposta. . Isso é devido ao tipo de instrução que ele realizou, observando-se à função *GET* se tratar de uma requisição mais simples teve o menor tempo do FIWARE, sendo até menor que a requisição *POST* da Alexa. Já a função *POST* do FIWARE se mostrou maior, tratando-se de uma função de configuração, assim sendo um pouco mais complexos que dos outros dispositivos.

Por último e com o maior tempo de latência, tem-se a requisição *PUT* seu tempo de resposta é maior, visto que você está enviando contexto diretamente para o fluxo do FIWARE e entre dispositivos. Assim, após a análise de resultados feita, é possível concluir que o sistema, em quesito de tempo de latência, tem sua reposta otimizada tendo em vista que os resultados são mostrados em ms, logo não prejudicando a experiência nem desempenho da fechadura em questão.

Figura 17: Gráfico dos resultado do tempo de resposta das requisições feitas.



Fonte: elaborado pelo autor.

6.2.2 Resultados do Experimento II

Nos resultados do segundo experimento é ilustrado nas Figuras 18 e 19, representando o cenário do usuário com acesso à fechadura e do usuário sem acesso à fechadura, respectivamente. Como mostra nas figuras, a primeira ação realizada se trata do "abrir smart lock", que se trata do comando para a Alexa entrar na *skill* da aplicação Smart Lock, e assim aceitar os seguintes comandos que foram executados. No cenário onde o usuário possui o acesso, em seguida ele realiza o comando de abrir a fechadura do quarto, onde é reconhecida na *skill*, e em seguida a uma reposta sonora para indicar que o comando foi entendido, no caso "Certo, abrindo fechadura quarto". Logo após é realizado o comando de fechar a porta, que também é reconhecido e executado, realizando uma mensagem de retorno, e por fim é realizado a saída da *skill* com o comando "adeus".

No cenário que o usuário não possui acesso à fechadura, são realizados os mesmos comandos efetuados no primeiro teste. A resposta do comando de "abrir fechadura quarto" é retornado a mensagem "Foi mal, não consegui te entender. Por favor, você poderia repetir?", que se trata de uma mensagem padrão para quando um comando é executado não é encontrado, essa mensagem também ocorrera na tentativa de realiza ao trancamento da fechadura.

Isso ocorre, pois o usuário em questão não possui acesso à fechadura, logo o comando para abrir e fechar uma fechadura não está disponível, para o mesmo. Assim, depois da análise dos testes realizados, foi possível constatar que o usuário quando possui acesso à fechadura e possui a *skill* da aplicação Smart Lock em sua aplicação Alexa, consegue realizar comando através

da fala. Já quem não possui acesso à fechadura, consegue usar *skill*, mas fica impossibilitado de realizar alguma ação em fechadura.

Figura 18: Teste em conta com acesso a fechadura em *Alexa Simulator* no *Console* de desenvolvimento Alexa.

Fonte: Elaborado pelo próprio autor

Figura 19: Teste em conta sem acesso a fechadura em *Alexa Simulator* no *Console* de desenvolvimento Alexa.

Fonte: Elaborado pelo próprio autor

7 CONCLUSÕES E TRABALHOS FUTUROS

Ao contrário das fechaduras tradicionais, as fechaduras inteligentes conseguem tomar decisões de que as tradicionais não conseguem, autorizando operações na fechadura de acordo com alguma implementação programada e executar várias outras ações, desde o trancamento da porta até a abertura. Muito utilizado em ambientes que requerem controle e automação, como residências, hotéis, hospitais, etc. A comodidade e eficiência que a abertura de porta traz para o usuário é enorme, pois auxilia em inúmeras situações, desde a perda de chaves até o atendimento a deficientes. A evolução tecnológica tornou este projeto possível, possibilitando a construção de uma fechadura inteligente, uma fechadura que se comunica por meio de comandos de voz, aplicativos móveis, sistemas embarcados e *middleware*.

A proposta deste trabalho foi desenvolver um sistema denominado *Smart Lock* capaz de gerenciar usuários e fechaduras através de um aplicativo *mobile*, construir um protótipo embarcado utilizando um microcontrolador ESP32Cam, utilizar *middleware* FIWARE para comunicação entre dispositivos, e utilizar um assistente virtual Alexa utiliza voz comandos.

A *Smart Lock* construída é um sistema embarcado funcionando corretamente, onde os componentes eletrônicos citados neste documento são bem sucedidos em sua funcionalidade, permitindo o correto comportamento da fechadura eletrônica controlada. O aplicativo móvel é construído com Flutter, gerenciando todos os recursos sugeridos. A configuração FIWARE foi construída e utilizada corretamente e o dispositivo consegue estabelecer comunicação eficiente e segura durante todo o seu ciclo de vida.

Para o comando de voz o uso da assistente virtual Alexa, se comportou de maneira adequada, conseguindo identificar falas, de pessoas, independentes de sotaque ou dificuldades de falas, assim permitindo que pessoas que possuam alguma dificuldade na fala ou sotaques fortes e característicos, consigam aproveitar o uso da ferramenta.

Com base nos experimentos e nos resultados obtidos, as etapas de prática e simulação podem ser seguidas. Esses experimentos estão atrelados à funcionalidade oferecida ao usuário pelo sistema Smart Lock, que permite a troca precisa e eficiente de informações entre a fechadura e o aplicativo, permitindo que o usuário opere sua fechadura consistentemente.

O sistema atende de maneira geral aos requisitos declarados, e além de toda a gestão do sistema e troca de informações com o sistema embarcado, também apresenta bom desempenho em termos de comandos de voz. O sistema torna-se viável e disponível para usuários que sentem necessidade de automação em fechaduras, principalmente para quem tem dificuldade

de locomoção.

7.1 Trabalhos Futuros

Como trabalhos futuros pretende-se aprimorar a conectividade da aplicação com a Alexa possibilitando uma vinculação de aplicativos, assim o usuário não precisaria manualmente definir qual fechadura ele possui o acesso. Também incluir outros assistentes virtuais para o controle e vinculação com a aplicação, como Google Assistente, não se limitando somente á assistente Alexa.

Além de realizar um aprimoramento tanto na aplicação *mobile* quanto também tornar possível ser acessada via navegador, além de organizar *layouts* e funcionalidades das páginas para mais usabilidade para o usuário.

Utilizar uma fechadura que consiga realizar o destranque manual através de uma maçaneta e uma chave, para casos que o usuário não possua acesso à internet e/ou energia no local que a fechadura se encontra

Além de tudo realizar um estudo de encomia de energia, para realizar o emprego de pilhas como a fonte de alimentação, assim não dependendo de tomadas disponíveis no local da instalação.

REFERÊNCIAS

ARAUJO, V.; MITRA, K.; SAGUNA, S.; ÅHLUND, C. Performance evaluation of fiware: A cloud-based iot platform for smart cities. **Journal of Parallel and Distributed Computing**, Elsevier, v. 132, p. 250–261, 2019.

BRASIL. Lei nº 13.146, de 6 de julho 2015. institui a lei brasileira de inclusão da pessoa com deficiência (estatuto da pessoa com deficiência). **Diário Oficial da República Federativa do Brasil**, Brasília, DF, 2015.

CONDE, J.; MUNOZ-ARCENTALES, A.; ALONSO, A.; LOPEZ-PERNAS, S.; SALVACHUA, J. Modeling digital twin data and architecture: A building guide with fiware as enabling technology. **IEEE Internet Computing**, IEEE, 2021.

DHANJAL, A. S.; SINGH, W. Tools and techniques of assistive technology for hearing impaired people. In: IEEE. **2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)**. [*S. l.*], 2019. p. 205–210.

DUTT, E.; MADURI, P. K.; GUPTA, A.; RATHOUR, S. *et al.* Touch-less home automation system with voice and gesture control. In: IEEE. **2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)**. [*S. l.*], 2020. p. 781–785.

GOLPÎRA, H.; KHAN, S. A. R.; SAFAEIPOUR, S. A review of logistics internet-of-things: Current trends and scope for future research. **Journal of Industrial Information Integration**, Elsevier, p. 100194, 2021.

HOLLANDA, A.; MIYANO, S. S. **Acessibilidade com Arduino**. Monografia – Faculdade Adventista de Hortolandia, 2015.

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.

JAIN, A.; TANWAR, P.; MEHRA, S. Home automation system using internet of things (iot). In: IEEE. **2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)**. [*S. l.*], 2019. p. 300–305.

KAREEM, H.; DUNAEV, D. The working principles of esp32 and analytical comparison of using low-cost microcontroller modules in embedded systems design. In: IEEE. **2021 4th International Conference on Circuits, Systems and Simulation (ICCSS)**. [*S. l.*], 2021. p. 130–135.

LI, Y.; ZHUANG, Y.; HU, X.; GAO, Z.; HU, J.; CHEN, L.; HE, Z.; PEI, L.; CHEN, K.; WANG, M.; NIU, X.; CHEN, R.; THOMPSON, J.; GHANNOUCHI, F. M.; EL-SHEIMY, N. Toward location-enabled iot (le-iot): Iot positioning techniques, error sources, and error mitigation. **IEEE Internet of Things Journal**, v. 8, n. 6, p. 4035–4062, 2021.

MACHESO, P.; CHISALE, S.; DAKA, C.; DZUPIRE, N.; MLATHO, J.; MUKANYIRIGIRA, D. Design of standalone asynchronous esp32 web-server for temperature and humidity monitoring. In: IEEE. **2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)**. [*S. l.*], 2021. v. 1, p. 635–638.

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) – Universidade de Coimbra, 2019.

MILEVA, A.; VELINOV, A.; HARTMANN, L.; WENDZEL, S.; MAZURCZYK, W. Comprehensive analysis of mqtt 5.0 susceptibility to network covert channels. **computers & security**, Elsevier, v. 104, p. 102207, 2021.

MOHAMMED, M. H. S. A hybrid framework for securing data transmission in internet of things (iots) environment using blockchain approach. In: IEEE. **2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)**. [*S. l.*], 2021. p. 1–10.

OLIVEIRA, L. M. B. *et al.* Cartilha do censo 2010: pessoas com deficiência. Secretaria de Direitos Humanos da Presidência da República (SDH-PR)., 2012.

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

PANCHANATHAN, S.; MCDANIEL, T.; TADAYON, R.; RUKKILA, A.; VENKATESWARA, H. Smart stadia as testbeds for smart cities: Enriching fan experiences and improving accessibility. In: IEEE. **2019 International Conference on Computing, Networking and Communications (ICNC)**. [*S. l.*], 2019. p. 542–546.

PHAM, V. C.; MAKINO, Y.; TAN, Y. A fiware iot agent for echonet lite protocol. In: IEEE. **2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)**. [*S. l.*], 2020. p. 235–239.

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.

SALHOFER, P.; BUCHSBAUM, J.; JANUSCH, M. Building a fiware smart city platform. In: IEEE. **Proceedings Of The 52nd Hawaii International Conference On System Sciences**. [*S. l.*], 2019.

SHANTHINI, M.; VIDYA, G.; ARUN, R. Iot enhanced smart door locking system. In: IEEE. **2020 Third International Conference On Smart Systems And Inventive Technology (ICSSIT)**. [*S. l.*], 2020. p. 92–96.

SOUSA, N. d. V. **Automação residencial por comandos de voz para pessoas com mobilidade reduzida**. Dissertação (Mestrado) – Instituto Federal da Paraiba, 2018.

SU, W.-T.; CHEN, W.-C.; CHEN, C.-C. An extensible and transparent thing-to-thing security enhancement for mqtt protocol in iot environment. In: IEEE. **2019 Global Iot Summit (GIOTS)**. [*S. l.*], 2019. p. 1–4.

VALOV, N.; VALOVA, I. Home automation system with raspberry pi. In: IEEE. **2020 7th International Conference on Energy Efficiency and Agricultural Engineering (EE&AE)**. [*S. l.*], 2020. p. 1–5.

WITTE, L. de. Assistive technology provision in low resource settings: Challenges and opportunities. In: IEEE. **2020 International Conference On Assistive And Rehabilitation Technologies (ICARETECH)**. [*S. l.*], 2020. p. 16–18.