



MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO
TOCANTINS
CAMPUS PALMAS
BACHARELADO EM ENGENHARIA ELÉTRICA

BERNARDO SOUZA FERREIRA DA SILVA

PROTÓTIPO DE UMA PLATAFORMA ELETRÔNICA PARA ACESSO A
AMBIENTES FÍSICOS INTEGRADA A UM SISTEMA DE GERENCIAMENTO NA
NUVEM

Palmas – TO

2023

BERNARDO SOUZA FERREIRA DA SILVA

**PROTÓTIPO DE UMA PLATAFORMA ELETRÔNICA PARA ACESSO A
AMBIENTES FÍSICOS INTEGRADA A UM SISTEMA DE GERENCIAMENTO NA
NUVEM**

Trabalho de Conclusão de Curso apresentado
como requisito parcial para obtenção do Título de
Bacharel em Engenharia Elétrica do Curso de
Engenharia Elétrica do Instituto Federal de
Educação, Ciência e Tecnologia do Tocantins,
Campus Palmas.

Orientador: Prof. Márcio Augusto Tamashiro, Dr.

Banca Examinadora:

Prof. Msc. Mateus Lima Peduzzi

Prof. Dr. Marcus André Pereira Oliveira

Palmas – TO

2023

Dados Internacionais de Catalogação na Publicação (CIP)
Bibliotecas do Instituto Federal do Tocantins

S586p Silva, Bernardo Souza Ferreira da
PROTÓTIPO DE UMA PLATAFORMA ELETRÔNICA PARA
ACESSO A AMBIENTES FÍSICOS INTEGRADA A UM SISTEMA DE
GERENCIAMENTO NA NUVEM / Bernardo Souza Ferreira da Silva.
– Palmas, TO, 2023.
123 p. : il. color.

Trabalho de Conclusão de Curso (Bacharelado em Engenharia
Elétrica) – Instituto Federal de Educação, Ciência e Tecnologia do
Tocantins, Campus Palmas, Palmas, TO, 2023.

Orientador: Dr. Márcio Augusto Tamashiro

1. Acesso. 2. IoT. 3. Fechadura. I. Tamashiro, Márcio Augusto. II.
Título.

CDD 621

A reprodução total ou parcial, de qualquer forma ou por qualquer meio, deste documento é autorizada para fins
de estudo e pesquisa, desde que citada a fonte.

**Elaborado pelo sistema de geração automática de ficha catalográfica do IFTO com os dados fornecidos
pelo(a) autor(a).**

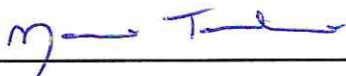
BERNARDO SOUZA FERREIRA DA SILVA

**PROTÓTIPO DE UMA PLATAFORMA ELETRÔNICA PARA ACESSO A
AMBIENTES FÍSICOS INTEGRADA A UM SISTEMA DE GERENCIAMENTO NA
NUVEM**

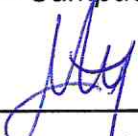
Trabalho de Conclusão de Curso apresentado à
Coordenação do Curso de Bacharelado em
Engenharia Elétrica do Instituto Federal de
Educação, Ciência e Tecnologia do Tocantins –
Campus Palmas, como exigência à obtenção do
grau de Bacharel em Engenharia Elétrica.

Aprovado em: 19/06/23

BANCA AVALIADORA



Doutor Márcio Augusto Tamashiro (Orientador)
IFTO – Campus Palmas



Mestre Mateus Lima Peduzzi
IFTO – Campus Palmas



Doutor Marcus André Pereira Pliveira
IFTO – Campus Palmas

Aos meus pais, amigos e irmãos.

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Márcio Augusto Tamashiro, por ser uma fonte de inspiração e motivação para o desenvolvimento deste trabalho.

Aos amigos Matheus Ávila, Jhones Bispo, Thiago Meneses, Isaías Caldeira, Bruno Sousa, Beatriz Santos, Ezequiel Almeida, Iago Souza, Janaína Moraes, Dónovan Jacobina e João Emanuel que repartiram o fardo comigo na jornada acadêmica, e compensaram as dificuldades com o companheirismo.

Aos meus pais, Paulo Roberto e Renilda, e aos meus irmãos Paulo Gabriel e Samuel, pelo incentivo e o sustento.

A Deus, por tudo o que fez.

Eis que estou à porta, e bato; se alguém ouvir a minha voz, e abrir a porta, entrarei em sua casa, e com ele cearei, e ele comigo.

Apocalipse 3:20

RESUMO

Este trabalho teve como foco o desenvolvimento de uma aplicação de Internet das Coisas (IoT) que consiste em um sistema de controle de acesso a ambientes físicos, composto por uma aplicação web integrada a uma fechadura eletrônica. A aplicação web tem a capacidade de gerenciar as permissões dos usuários para a fechadura eletrônica, e monitorar o histórico de acesso ao ambiente escolhido. Além disso, foi construído um protótipo de fechadura eletrônica utilizando o microprocessador ESP-WROOM-32 e integrando-o com um módulo leitor *RFID*, um leitor de impressões digitais, um teclado matricial e um display *LCD*. Os componentes do protótipo possibilitaram a autenticação do usuário por métodos variados, como cartões *RFID*, impressões digitais e senha numérica. A aplicação web controla a fechadura eletrônica, enviando comandos para abrir ou fechar de acordo com as permissões do usuário. O protótipo da fechadura pode ainda autenticar o usuário corretamente, mesmo em cenários de perda de conexão à internet.

Palavras-chave: Acesso. *IoT*. Fechadura.

ABSTRACT

This work focused on the development of an Internet of Things (IoT) application that consists of an access control system for physical environments, composed of a web application integrated with an electronic lock. The web application has the ability to manage user permissions for the electronic lock and monitor the access history to the chosen environment. In addition, a prototype of an electronic lock was built using the ESP-WROOM-32 microprocessor and integrating it with an RFID reader module, a fingerprint reader, a matrix keypad, and an LCD display. The components of the prototype enabled user authentication through various methods, such as RFID cards, fingerprints, and numerical passwords. The web application controls the electronic lock by sending commands to open or close it based on the user's permissions. Furthermore, the lock prototype can authenticate the user correctly even in scenarios of internet connection loss.

Keywords: Access. IoT. Lock.

LISTA DE FIGURAS

Figura 1 - Estrutura de um sistema RFID.	22
Figura 2 - Módulo leitor RFID.	23
Figura 3 - Tags RFID em diversos formatos.....	24
Figura 4 - Diagrama de blocos de um sistema biométrico da impressão digital.	27
Figura 5 - Minúcias da impressão digital e suas posições e angulações.	28
Figura 6 - Esquema de ligação de um teclado matricial 3x4.	29
Figura 7 - Pinos de ligação do ESP-WROOM-32.....	32
Figura 8 - Ilustração da comunicação serial.	33
Figura 9 - Ilustração da comunicação paralela.	33
Figura 10 - Estrutura do pacote de dados do protocolo UART.....	34
Figura 11 - Comunicação entre mestre e escravo no protocolo SPI.	36
Figura 12 - Formato de dados de transmissão do protocolo I2C.....	37
Figura 13 - Critério de início e fim de transmissão no protocolo I2C.	37
Figura 14 - Ilustração da estrutura cliente-servidor.	38
Figura 15 - Interação da API com múltiplos softwares.	42
Figura 16 - Diagrama simplificado do sistema proposto.....	45
Figura 17 - Diagrama completo do sistema proposto.....	46
Figura 18 - Esquemático de ligação do ESP-WROOM-32 para a fechadura eletrônica.	47
Figura 19 - Módulo RFID RC522 com pinagem sinalizada.....	49
Figura 20 - Esquemático de ligação do módulo leitor RFID para a fechadura eletrônica.	50
Figura 21 - Módulo leitor biométrico DY50.	51
Figura 22 - Esquemático de ligação do módulo leitor de impressões digitais para a fechadura eletrônica.....	52
Figura 23 – Teclado matricial rígido de doze teclas.	53
Figura 24 - Esquemático de ligação do teclado matricial para a fechadura eletrônica.	54
Figura 25 - Frente e verso do display LCD (16x2) com o módulo I2C acoplado.	55
Figura 26 - Pinagem do display LCD.....	56
Figura 27 - Fluxograma do algoritmo da fechadura eletrônica.	59
Figura 28 - Diagrama de operação da fechadura no modo offline.....	61

Figura 29 - Hardware do protótipo montado na protoboard.....	62
Figura 30 - Mola aérea para portas.	63
Figura 31 - Fechadura de sobrepor.	63
Figura 32 - Estrutura de Banco de Dados.	65
Figura 33 - Estrutura do site.	67
Figura 34 - Esboço da tela de Login.....	68
Figura 35 - Esboço da tela de controle da fechadura.....	69
Figura 36 - Tela de histórico de acessos.....	69
Figura 37 - Tela de controle de permissões dos usuários.....	70
Figura 38 - Tela de requisição de cadastro de “operador” via insomnia.....	71
Figura 39 - Tela de requisição de cadastro de ambiente via insomnia.....	72
Figura 40 - Requisição de cadastro de fechadura pelo insomnia.....	72
Figura 41 - Tela de Login da aplicação.	73
Figura 42 - Tela de controle de fechaduras da aplicação.....	73
Figura 43 - Tela de detalhe de fechaduras da aplicação.....	74
Figura 44 - Tela de controle de ambiente da aplicação.....	75
Figura 45 - Tela de gerenciamento de permissões da aplicação.	75
Figura 46 - Tela de detalhe da permissão da aplicação.....	76
Figura 47 - Tela de gerenciamento de permissões da aplicação.	77
Figura 48 - Display LCD exibindo as três fases do menu.....	78
Figura 49 - Teste do método de entrada RFID.....	79
Figura 50 - Display serial no processo de leitura da tag RFID.....	79
Figura 51 - Tela de histórico de fechadura.....	80
Figura 52 - Teste do método de entrada por impressão digital.	81
Figura 53 - Display serial no processo de leitura da impressão digital.....	81
Figura 54 - Teste do método de entrada por senha numérica.....	82
Figura 55 - Teste do LED indicador de abertura.....	83
Figura 56 - Tela da aplicação após realizar uma solicitação de entrada.....	84
Figura 57 - Tela de controle de fechadura indicando a abertura do dispositivo.....	84
Figura 58 - Display LCD após a entrada no modo offline.	85
Figura 59 - Display serial no modo offline de leitura do cartão RFID.....	86
Figura 60 - Display serial no modo offline de leitura da impressão digital.	86
Figura 61 - Display serial no processo offline de leitura da senha numérica.....	87

Figura 62 - Display serial no processo reconexão com a internet.88

LISTA DE QUADROS

Quadro 1 - Comparação de funcionalidades de hardware dos trabalhos relacionados.	
.....	19
Quadro 2 - Comparação das tecnologias de hardware dos trabalhos relacionados.	20
Quadro 3 - Comparação de funcionalidades de software dos trabalhos relacionados.	
.....	20
Quadro 4 - Comparação das tecnologias de software dos trabalhos relacionados.	21
Quadro 5 - Melhorias para trabalhos futuros.	91

LISTA DE TABELAS

Tabela 1 - Comparação entre o ESP 32 e o ATMEGA328P.....	31
Tabela 2 - Conexão dos pinos do microcontrolador com os demais elementos do protótipo.	48
Tabela 3 - Conexão dos pinos do microcontrolador com o módulo RC522.....	50
Tabela 4 - Conexão dos pinos do microcontrolador com o módulo DY50.	53
Tabela 5 - Conexão dos pinos do microcontrolador com o módulo de do teclado matricial (3x4).....	54
Tabela 6 - Conexão dos pinos do microcontrolador com o módulo LCD I2C.	56
Tabela 7 - Custo total do protótipo.....	89

SUMÁRIO

1 INTRODUÇÃO	14
1.1 PROBLEMATIZAÇÃO.....	15
1.2 JUSTIFICATIVA	15
1.3 OBJETIVOS.....	16
1.3.1 <i>Objetivos gerais</i>	16
1.3.2 <i>Objetivos específicos</i>	16
2 REVISÃO DA LITERATURA	17
2.1 TRABALHOS RELACIONADOS	18
3 CONCEITOS E ELEMENTOS FUNDAMENTAIS.....	22
3.1 <i>RFID</i>	22
3.2 IDENTIFICAÇÃO BIOMÉTRICA	24
3.2.1 <i>Biometria da impressão digital</i>	26
3.3 TECLADO MATRICIAL	29
3.4 MICROCONTROLADOR ESP-WROOM-32	30
3.5 COMUNICAÇÃO <i>SERIAL</i>	33
3.5.1 <i>Comunicação UART</i>	34
3.5.2 <i>Comunicação SPI</i>	35
3.5.3 <i>Comunicação I2C</i>	36
3.6 FERRAMENTAS DE DESENVOLVIMENTO <i>WEB</i>	38
3.6.1 <i>Framework e Bibliotecas</i>	39
3.6.2 <i>React.JS</i>	40
3.6.3 <i>API</i>	41
3.6.4 <i>Node.JS</i>	42
3.6.5 <i>Firebase</i>	43
3.6.6 <i>Firebase Realtime Database</i>	44
3.6.7 <i>Firebase Authentication</i>	44
4 DESENVOLVIMENTO DO PROTÓTIPO	45
4.1 DESENVOLVIMENTO DA FECHADURA ELETRÔNICA	47
4.1.1 <i>Componentes da Fechadura</i>	47

4.1.2 Algoritmo da fechadura eletrônica	57
4.1.3 Fechadura eletrônica	61
4.2 DESENVOLVIMENTO DA APLICAÇÃO WEB	64
4.2.1 Desenvolvimento da central de comunicação.....	64
4.2.2 Desenvolvimento da central de monitoramento e controle	67
5 RESULTADOS E DISCUSSÕES.....	71
5.1 CADASTRO E CONFIGURAÇÃO DA FECHADURA E PLATAFORMA.....	71
5.2 TESTES ONLINE E OFFLINE	77
5.2.1 Testes online	77
5.2.2 Testes offline	85
5.3 CUSTO DO <i>HARDWARE</i>	89
6 CONSIDERAÇÕES FINAIS	90
REFERÊNCIAS.....	92
APÊNDICE A – ESQUEMÁTICO DO CIRCUITO.....	99
APÊNDICE B – CÓDIGO-FONTE DO <i>HARDWARE</i> DESENVOLVIDO	100

1 INTRODUÇÃO

O surgimento dos microcontroladores teve grande relevância para o avanço de diversas tecnologias. O avanço tecnológico permitiu que tais ferramentas diminuíssem de tamanho e de preço, como afirma Oliveira (2017), “a evolução da microeletrônica permitiu a queda do custo dos microprocessadores 32-bit, que já podem ser encontrados a menos de um dólar em alguns casos”. E isso fomentou o desenvolvimento de diversas aplicações de Internet das Coisas, um conceito que visa integrar as informações e o controle de diversos objetos na internet.

Dentro do conceito de Internet das Coisas existem diversas aplicações, como por exemplo, os carros inteligentes da empresa Tesla ou então as casas inteligentes, com apurados sistemas de monitoramento e controle de dispositivos através da internet. De acordo com Pereira (2019):

Entre essas aplicações, o problema de controle de acesso tem se tornado cada vez mais relevante no cenário da internet das coisas e cidades inteligentes, tanto na segurança gerenciamento de casas e apartamentos como também em ambientes de trabalho onde se faz necessário controlar o acesso a determinados ambientes.

Indústrias, comércios, instituições educacionais e residências possuem restrições de acesso, isso se deve a diversos fatores. Nas indústrias é necessário controlar a entrada e saída das pessoas aos ambientes, tanto para proteger os funcionários quanto para proteger os maquinários ou informações restritas a algum núcleo de trabalho específico.

Nos comércios é necessário que o acesso às salas seja controlado para não haver furtos de produtos. Já nas instituições educacionais temos os laboratórios que possuem diversos instrumentos com elevados custos, e que necessitam de uma correta operação. Esses ambientes necessitam assim, de um controle e também restrição de acesso. Dentro de nossas residências isto é essencial, o controle de acesso de quem entra ou saí deste ambiente é necessário para a segurança dos residentes.

Sendo assim, este trabalho propõe apresentar uma solução para o problema de controle de acesso, por meio de um protótipo de fechadura eletrônica e um sistema de gerenciamento de acesso.

A fechadura eletrônica foi construída com base no microprocessador ESP-WROOM-32, que possui conexão com um sistema *online*, acessível por meio de uma aplicação *web* capaz de operar remotamente este protótipo, e ainda monitorar o acesso e as permissões dos indivíduos que frequentam um determinado ambiente.

1.1 Problematização

Diversos locais possuem restrição de acesso, um exemplo são as instituições educacionais, em que há diversas salas e laboratórios que necessitam de um monitoramento dos frequentadores do local. Geralmente as soluções encontradas para realizar tal controle de acesso são manuais, por meio de anotações em cadernos ou até mesmo através de uma pessoa que controle o fluxo de entrada e saída em um ambiente.

Portanto, o problema identificado é o controle de acesso aos ambientes, como de indústrias, empresas ou instituições educacionais que necessitam de um monitoramento e acesso a locais restritos. A dificuldade de adaptar tecnologias e supervisionar o histórico de entradas e saídas de usuários em locais que exigem uma segurança mais rigorosa, provoca a baixa confiabilidade no controle de acesso. De modo geral, os mecanismos encontrados para solucionar o problema são limitados a um controle físico de chaves para acessar tais localidades, tendo geralmente a necessidade de alocar uma pessoa naquele local para gerenciar o acesso ao ambiente.

Existe também a limitação que a própria chave cria, ao precisar deste artefato físico, o acesso ao local fica vinculado ao uso da chave, em casos de esquecimento ou perda, o local se torna inacessível.

1.2 Justificativa

O desenvolvimento de um sistema que possibilita uma variedade de métodos de acesso, e a visualização dos históricos de entrada em ambientes restritos, é uma

forma de ter controle e prover a segurança em ambientes que exigem uma restrição de entrada.

Nessa perspectiva, a elaboração do presente projeto composto pelo *site* e o protótipo se justifica por ser uma proposta para o controle de acesso de diversos ambientes, tanto o de residências como de empresas ou indústrias, além de prover novos métodos de acesso.

1.3 Objetivos

1.3.1 Objetivos gerais

Este trabalho tem como objetivo o desenvolvimento de um sistema distribuído de gerenciamento de acesso a ambientes. O sistema é constituído por um protótipo de uma fechadura eletrônica e um *software* de gerenciamento desse dispositivo. O projeto consiste em duas partes principais, um protótipo de fechadura eletrônica e um *software* gerenciador do dispositivo.

1.3.2 Objetivos específicos

- Gerenciar e identificar corretamente as permissões de acesso por método de entrada (*tag RFID*, senha numérica, impressão digital e abertura através do *software*);
- Controlar a abertura e monitorar o histórico acesso a fechadura;
- Armazenar permissões do banco de dados na memória flash do microcontrolador ESP-WROOM-32;
- Realizar validações de entrada *online* e *offline*.

2 REVISÃO DA LITERATURA

Por conta do avanço tecnológico, conceitos que anteriormente habitavam ambientes teóricos começaram a adentrar o cotidiano da sociedade, a internet das coisas ou *IoT* (*Internet of Things*) certamente é um destes conceitos. Segundo um estudo realizado pelo McKinsey Global Institute (junho de 2015), até 2025 a internet das coisas terá um impacto econômico entre 3,9 a 11,1 trilhões de dólares anuais.

Vinicius G. Oliveira et al (2017) definem a internet das coisas como:

“Objetos de uso cotidiano, pessoal ou industrial, dotados de computação embarcada e unicamente identificados, os quais trocam dados entre si e também com pessoas por meio da Internet, permitindo a criação de serviços e experiências de uso adaptados ao contexto.”

Com o crescente uso de smartphones, tablets, computadores e outros dispositivos conectados à internet, a população está cada vez mais capacitada a desfrutar e oferecer serviços online. Isso tem permitido a migração de diversas tarefas que antes eram realizadas de forma analógica para o ambiente digital.

Sérgio de Oliveira (2017) afirma que “para tornar Internet das Coisas uma realidade, foi preciso que o custo das soluções se tornasse compatível com as ‘coisas’ a serem conectadas”. Desta forma, a criação e a aquisição de novos dispositivos inteligentes estão se tornando cada vez mais acessíveis devido ao avanço e à popularização dos microcontroladores, propiciando o desenvolvimento de soluções de baixo custo nesta área da tecnologia.

As aplicações de *IoT* possuem um grande campo de atuação, desde o setor industrial, com o monitoramento remoto dos equipamentos envolvidos em uma linha de produção, ao setor agrícola com o controle automatizado de plantações.

Uma das aplicações que possuem mais influência no cotidiano da população é a automação residencial, por ser parte de um mercado mais abrangente, já que os dispositivos inteligentes podem ser utilizados na moradia de cada usuário. Como José Roberto Muratori e Paulo Henrique Dal Bó (2011) bem definem que a automação residencial é: “O conjunto de serviços proporcionados por sistemas tecnológicos integrados como o melhor meio de satisfazer as necessidades básicas de segurança, comunicação, gestão energética e conforto de uma habitação”.

Portanto, a automação residencial vem para proporcionar também mais segurança aos lares daqueles que utilizam os dispositivos inteligentes, sistemas de alarme de intrusão e sistemas de controle de acesso são sistemas que permitem controlar e monitorar o acesso a ambientes.

2.1 Trabalhos relacionados

Diversos artigos e trabalhos científicos já propuseram sistemas de controle de acesso. A abordagem de *Eike D. Santiago, Diogo F. Bento e Breno J. D. Costa* (2019) no trabalho *LetMeIn - Sistema de Controle de Acesso Para Ambientes Educacionais* é a de usar um sistema de controle de acesso gerenciado através de um aplicativo de celular, e ligado a um servidor em um microcomputador *Raspberry Py* e uma fechadura eletrônica. Esse sistema é interessante por oferecer uma solução de controle de acesso de baixo custo, porém possui a limitação de possibilitar acesso somente através do aplicativo de celular.

O trabalho de *Diogo Weber Silveira* (2021), *Desenvolvimento de uma fechadura eletrônica: um sistema de controle de acesso com registro em banco de dados e site de gerenciamento* apresenta uma abordagem diferente, adotando o uso de *tags RFID* e o microcontrolador arduino na construção da fechadura eletrônica, e o fornecimento do histórico de acessos através de uma plataforma *web* ligada a um banco de dados.

O sistema proposto por *Gabriel Niquini Lopez* (2015) na monografia intitulada *Sistema para monitoramento de acesso utilizando Arduino e leitura biométrica* se destaca pelo uso de um sensor de leitura de impressão digital, integrado a um sistema supervisor de autenticação. Porém, não armazena o histórico de acessos dentro da aplicação desenvolvida.

Já no trabalho *Sistema inteligente para controle de acesso e monitoramento de múltiplos ambientes (classcontrol)* de *Vitor V. Pereira, Samaherni M. Dias e Kurios I. P. M. de Queiroz* (2019), o histórico de acesso também está disponível, e a estrutura física conta com a fechadura, um servidor, e também um computador intermediário que realiza operações entre o servidor e a fechadura. Além disso, a fechadura ainda possui funcionalidades extras de controlar outros dispositivos da sala a qual está controlando o acesso.

Letícia da Silva Mota e Elloa B. Guédes (2019) em seu artigo *Controle de Acesso a Múltiplos Ambientes com Internet das Coisas* trabalham com um sistema semelhante ao anterior, com o uso de *tags RFID* como alternativa as chaves, porém com o diferencial de apresentarem mais uma interface para o usuário da fechadura, um *display LCD*. Mas não possuem o histórico de acessos dos usuários.

No trabalho *Desenvolvimento de um sistema de controle de acesso com armazenamento em nuvem* de Luis Otávio Manente e Pedro Moreno Crespo (2019), é proposto um sistema que combina várias funcionalidades. Isso inclui o controle da fechadura por meio do celular utilizando tecnologia *Bluetooth*, a disponibilidade de um histórico de acessos e a utilização de *tags RFID*. Além disso, como uma alternativa para o usuário da fechadura eletrônica, também é incorporado um teclado numérico.

A solução de Vinicius Rocha Gonçalves (2019) consegue juntar funcionalidades dos trabalhos anteriores, com o uso da *tag RFID*, do teclado numérico e do *display LCD* em uma só fechadura, além de disponibilizar o histórico de acessos através de uma plataforma *web*.

Dentro desse cenário que se encaixa a proposta deste trabalho, juntando algumas das funcionalidades citadas nos trabalhos anteriores e adicionando novas. Foram feitos quatro quadros comparativos para evidenciar as funcionalidades que este trabalho pretende agregar. Os dois primeiros quadros (Quadro 1 e Quadro 2) comparam as funcionalidades e tecnologias utilizadas na construção do *hardware*, já os dois últimos (Quadro 3 e Quadro 4) comparam as funcionalidades e tecnologias utilizadas na construção do *software* do sistema.

Quadro 1 - Comparação de funcionalidades de hardware dos trabalhos relacionados.

Comparação de funcionalidades dos trabalhos relacionados (Hardware)				
Trabalhos	<i>RFID</i>	Teclado Numérico	<i>Display LCD</i>	Sensor Biométrico
2019 - [EikeD. Santiago, Diogo F. Bento, BrenoJ. D. Costa]				
2019 - [Vitor V. Pereira, Samaherni M. Dias e Kurios I. P. M. de Queiroz]	X			
2021 - [Diogo Weber Silveira]	X			
2015 - [Gabriel Niquini Lopez]			X	X

2019 - [Letícia da Silva Mota, Elloa B. Guedes]	X		X	
2019 - [Luis Otávio Manente, Pedro Moreno Crespo]	X	X		
2019 - [Vinicius Rocha Gonçalves]	X	X	X	
Solução Proposta	X	X	X	X

Fonte: Acervo do autor (2022).

Quadro 2 - Comparação das tecnologias de hardware dos trabalhos relacionados.

Comparação das tecnologias dos trabalhos relacionados (<i>Hardware</i>)		
Trabalhos	Microcontrolador	Microcomputador
2019 - [EikeD. Santiago, Diogo F. Bento, BrenoJ. D. Costa]		RASPBERRY PY
2019 - [Vitor V. Pereira, Samaherni M. Dias e Kurios I. P. M. de Queiroz]	ESP8266	RASPBERRY PY
2021 - [Diogo Weber Silveira]	ARDUINO	
2015 - [Gabriel Niquini Lopez]	ARDUINO	
2019 - [Letícia da Silva Mota, Elloa B. Guedes]	ESP8266	
2019 - [Luis Otávio Manente, Pedro Moreno Crespo]	ESP 32	
2019 - [Vinicius Rocha Gonçalves]	ESP 32	
Solução Proposta	ESP 32	

Fonte: Acervo do autor (2022).

Quadro 3 - Comparação de funcionalidades de software dos trabalhos relacionados.

Comparação das funcionalidades dos trabalhos relacionados (<i>Software</i>)		
Trabalhos	Abertura via aplicação	Histórico de acessos
2019 - [EikeD. Santiago, Diogo F. Bento, BrenoJ. D. Costa]	X	
2019 - [Vitor V. Pereira, Samaherni M. Dias e Kurios I. P. M. de Queiroz]		X

2021 - [Diogo Weber Silveira]		X
2015 - [Gabriel Niquini Lopez]		
2019 - [Letícia da Silva Mota, Elloa B. Guedes]		
2019 - [Luis Otávio Manente, Pedro Moreno Crespo]	X	X
2019 - [Vinicius Rocha Gonçalves]		X
Solução Proposta	X	X

Fonte: Acervo do autor (2022).

Quadro 4 - Comparação das tecnologias de software dos trabalhos relacionados.

Comparação das tecnologias dos trabalhos relacionados (Software)			
TRABALHOS	APLICAÇÃO	PADRÃO DE COMUNICAÇÃO	BANCO DE DADOS
2019 - [EikeD. Santiago, Diogo F. Bento, Breno J. D. Costa]	NÃO INFORMADO	HTTP	NÃO INFORMADO
2019 - [Vitor V. Pereira, Samaherni M. Dias e Kurios I. P. M. de Queiroz]	NÃO INFORMADO	NÃO INFORMADO	NÃO INFORMADO
2021 - [Diogo Weber Silveira]	PHP	HTTP	MYSQL
2015 - [Gabriel Niquini Lopez]	C#	NÃO INFORMADO	NÃO POSSUI
2019 - [Letícia da Silva Mota, Elloa B. Guedes]	WEB2PY	MQTT	SQLITE
2019 - [Luis Otávio Manente, Pedro Moreno Crespo]	NÃO POSSUI	NÃO INFORMADO	GOOGLE PLANILHAS
2019 - [Vinicius Rocha Gonçalves]	ANGULAR	HTTP	MYSQL
Solução Proposta	REACT	HTTP	FIREBASE

Fonte: Acervo do autor (2022).

3 CONCEITOS E ELEMENTOS FUNDAMENTAIS

Apresentado o contexto no qual o trabalho proposto se encaixa, é necessário então apresentar os conceitos basilares da solução proposta. Esta seção é dedicada a apresentar tecnologias, definições e conceitos necessários para a realização da presente proposta.

3.1 *RFID*

RFID é uma sigla para *Radio Frequency Identification* (Identificação por rádio frequência). Essa tecnologia foi desenvolvida pouco antes da II Guerra Mundial, para fins de identificação das aeronaves britânicas durante conflitos. Porém, a partir dos anos 80 que o *MIT* (Instituto de Tecnologia de Massachusetts) e outros centros de pesquisa desenvolveram aplicações voltadas para o uso da radiofrequência para identificação, rastreamento e localização de produtos (OLIVEIRA e PEREIRA, 2006).

De acordo com Weinstein (2005):

A premissa básica por trás de um sistema *RFID* é marcar objetos com *tags*. Estas *tags* contêm transponders que emitem mensagens legíveis por leitores *RFID* especializados. A maioria das *tags RFID* armazenam algum número de identificação.

Portanto, comumente um sistema *RFID* geralmente é composto por três partes (como ilustrado na Figura 1): dispositivos de leitura, transponder (*Tag*) e central de processamento.

Figura 1 - Estrutura de um sistema *RFID*.



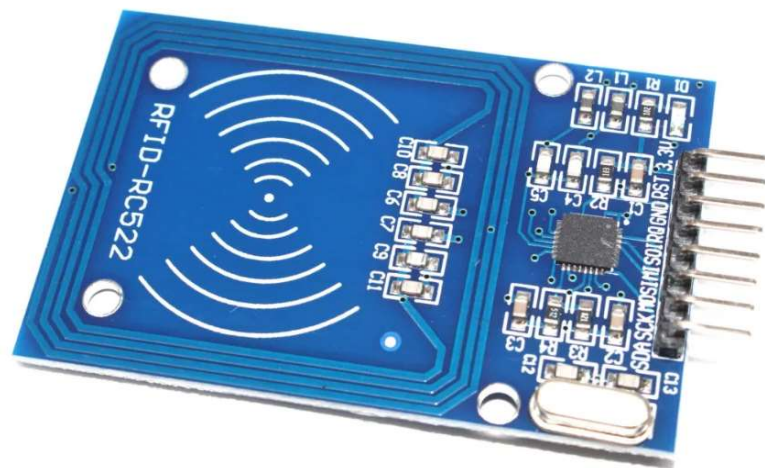
Fonte: Acervo do autor (2022).

O **dispositivo de leitura** é responsável por emitir um campo eletromagnético, que serve para energizar o transponder que o responde de acordo com o conteúdo de que está registrado na *tag*. Narciso (2008) afirma:

O leitor emite frequências de rádio que são dispersas em diversos sentidos no espaço, desde alguns centímetros até alguns metros ao emitir esse sinal, a etiqueta responde ao leitor com o conteúdo de sua memória. Por apresentar essa característica, o equipamento pode ler através de diversos materiais como papel, cimento, plástico, madeira, vidro, etc. Assim, a comunicação entre o leitor e a etiqueta é feita pela troca de sinais eletromagnéticos.

A Figura 2 apresenta o módulo leitor *RFID*.

Figura 2 - Módulo leitor *RFID*.



Fonte: Piscaled (2022).

O **transponder, tag ou etiqueta** é o dispositivo que guarda informações para identificar algo, cujo funcionamento se assemelha ao de uma carteira de identidade. Segundo Oliveira e Pereira (2006, p. 2), o transponder “faz uso de dispositivos eletrônicos (memória, processador, resistores, capacitores e indutores) para guardar uma informação ou gerar um sinal”.

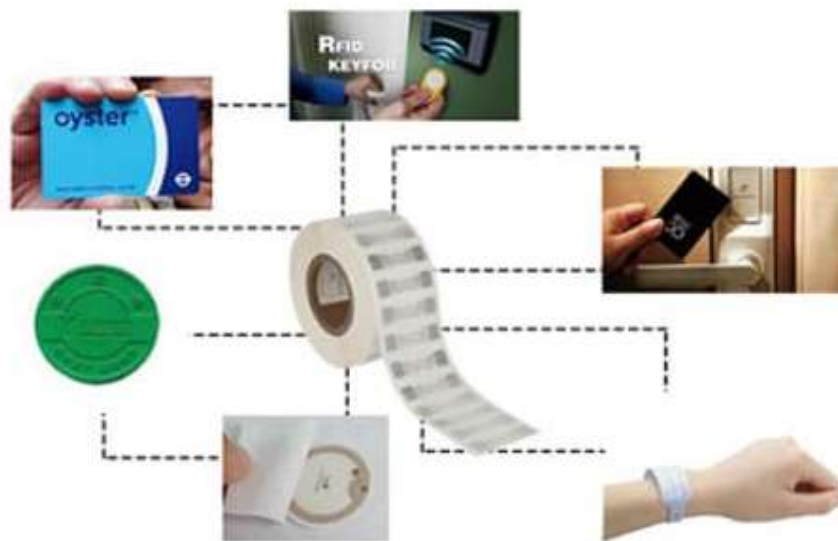
Ele é composto por um chip responsável por guardar informações e uma antena para realizar a comunicação com o dispositivo de leitura. A *tag* pode conter ou não uma fonte de alimentação, critério que é utilizado para categorizar as *tags RFID* como *tags* passivas ou ativas.

As *tags* ativas são aquelas que possuem fonte de alimentação no seu sistema, isso faz com que essas *tags* transmitam um sinal mais forte, que os leitores podem acessar de maiores distâncias. Leitores *RFID* podem se comunicar com *tags* ativas em distâncias de 20 a 100 metros, porém são maiores e mais caras se comparadas as *tags* passivas (WEINSTEIN, 2005).

As *tags* passivas não possuem alimentação e, portanto, dependem da energia fornecida pelo leitor, consequentemente o seu alcance é reduzido em comparação com as *tags* ativas.

Podemos observar na Figura 3 que o transponder pode ser encontrado em diversos formatos, como cartões e outros materiais de plástico, tudo isso devido a sua simplicidade estrutural.

Figura 3 - Tags RFID em diversos formatos.



Fonte: I3C soluções (2022).

A **central de processamento** geralmente é um computador responsável por tomar decisões a partir do dado que foi capturado pelo dispositivo de leitura. O processo de captação e tratamento dos dados é realizado através de um *Middleware*¹ *RFID* que serve para gerenciar os dados capturados pelos leitores e também tem a função de integrar o sistema que envolve a tecnologia *RFID* com sistemas externos (NARCISO, 2008).

3.2 Identificação Biométrica

Em sistemas de controle de acesso, um dos processos mais importantes é identificar aqueles que acessam os ambientes controlados, e o quanto mais preciso

¹ O middleware é um software que diferentes aplicações usam para se comunicar umas com as outras. Ele oferece funcionalidade para conectar aplicações de modo inteligente e eficiente (AMAZON, 2023).

for essa identificação do usuário melhor será efetuado o registro e o controle dos dados. Dentro deste cenário a identificação biométrica surge como uma alternativa que garante grande precisão na identificação do usuário e no controle de acesso dos ambientes. De acordo com Lourenço (2009, p. 12):

A expressão “biometria” advém da derivação de palavras (ditas “primitivas”) de origem grega, **bios** (vida) e **metros** (contagem ou medida). Assim, a biometria define-se como sendo a ciência que estuda estatisticamente as características, físicas, fisiológicas ou comportamentais dos seres vivos, mas atualmente tem sido associado a estudo das características físicas, fisiológicas ou comportamentais das pessoas como forma de identificá-las.

A partir do reconhecimento de indivíduos relacionando suas características físicas (estáticas), fisiológicas ou comportamentais (dinâmicas) podemos ter mais segurança para confirmar a identidade do ser. Porém, nem todos os atributos do ser humano podem ser utilizados para identificá-lo. De acordo com Clarke (1994) existem alguns critérios para identificação de um indivíduo:

- **Universalidade de cobertura:** todos os usuários devem possuir um identificador;
- **Unicidade:** cada usuário deve ter apenas um identificador; dois usuários não podem possuir o mesmo identificador;
- **Permanência:** o identificador não pode mudar, não pode ser mudado;
- **Indispensabilidade:** o identificador deve ser uma ou mais características naturais que cada pessoa tem. Caso seja artificial o identificador deve estar disponível em todos os momentos;
- **Capturabilidade:** O identificador deve ser capturável por qualquer um em qualquer ocasião;
- **Armazenabilidade:** O identificador deve ser armazenável em sistemas manuais ou automatizados;
- **Precisão:** Cada identificador deve ser suficientemente diferente de qualquer outro identificador para que os erros sejam improváveis;
- **Simplicidade:** o armazenamento e transmissão devem ser fáceis e não sujeitas a erros;
- **Custo:** medir e armazenar não deve ser excessivamente caro;

- **Conveniência:** Medir e armazenar o identificador não deve ser inconveniente ou demorado;
- **Aceitabilidade:** Seu uso deve estar de acordo com os padrões sociais contemporâneos.

De acordo com Costa *et al.* (2006), as características comportamentais são “aprendidas ou desenvolvidas ao longo da utilização constante, e que podem variar fortemente ao longo do tempo. Além disso, podem ser facilmente alteradas pela vontade ou estado do usuário”. Já os aspectos fisiológicos “podem variar com o avanço da idade da pessoa, ou devido a problemas de saúde” (LOURENÇO, 2009). Portanto, as características fisiológicas e comportamentais não podem ser consideradas adequadas para a identificação de um indivíduo.

Resta então para a identificação biométrica as características físicas, pois são atributos estáticos ligados diretamente às propriedades genéticas e as particularidades do corpo do indivíduo. Existem diversos atributos físicos utilizados para identificação sendo eles: íris, retina, traços faciais e a impressão digital. Dentre essas opções, a impressão digital é uma das mais populares e amplamente utilizadas na tecnologia biométrica.

3.2.1 Biometria da impressão digital

Desde 2008 o Brasil começou a implementar gradualmente a biometria da impressão digital durante as eleições visando, segundo o TSE, “tornar o processo eleitoral ainda mais seguro e evitar que uma pessoa votasse no lugar de outra”. Em 2008 este método de identificação foi testado no processo eleitoral somente em três municípios onde a urna eletrônica só era liberada para a votação após a impressão digital ser reconhecida no banco de dados da Justiça Eleitoral.

Já em 2010 a verificação da identidade do cidadão por meio da biometria da impressão digital foi utilizada em 60 municípios. Em 2020 devido à pandemia da SARS-CoV-2 não houve a identificação biométrica dos eleitores. Porém “espera-se que quase 100% do eleitorado esteja apto a votar com identificação biométrica até as eleições de 2026” (TSE).

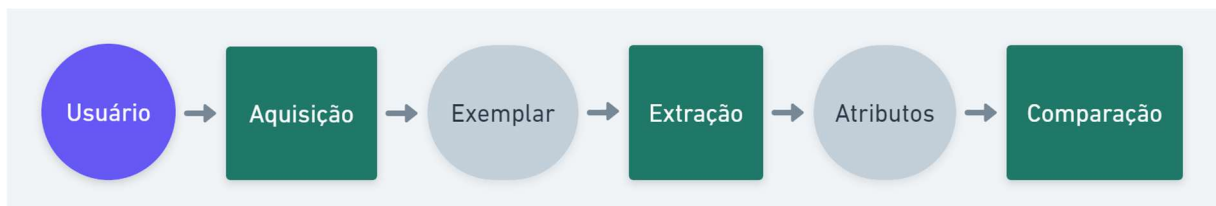
Formada ainda na gestação, a impressão digital de cada indivíduo possui marcas que constroem um desenho com padrões aleatórios e, portanto, único em

cada indivíduo, ainda que seja possível haver similaridades entre impressões digitais de diferentes pessoas, segundo Maltoni *et al* (2003):

Em estudos dermatológicos, a máxima diferença entre impressões digitais tem sido encontrada entre indivíduos de diferentes raças. Pessoas da mesma raça, porém sem grau de parentesco, possuem similaridade muito pequena nas digitais. Pai e filho possuem alguma similaridade, por compartilharem metade dos genes. Gêmeos monozigóticos (idênticos) possuem a máxima similaridade. Estima-se que 95% das características das digitais de gêmeos idênticos sejam iguais.

O processo para realizar a identificação através da impressão digital pode ser dividido em três etapas: **aquisição**, **extração** e **comparação**. A etapa da aquisição é a responsável por registrar a impressão digital do usuário e ao final de seu processo deve gerar os dados que passarão pelo processo de extração. O processo de extração é o responsável por identificar os atributos a partir dos dados fornecidos pelo processo de aquisição. Já a etapa de comparação é onde os atributos ou também chamados minúcias da digital são comparados com outro modelo para assim verificar a identidade do usuário analisado. Cada uma das três etapas possui maneiras diferentes de serem implementadas. A Figura 4 apresenta o diagrama de blocos de um sistema biométrico da impressão digital.

Figura 4 - Diagrama de blocos de um sistema biométrico da impressão digital.



Fonte: Adaptado de UFRJ (2009).

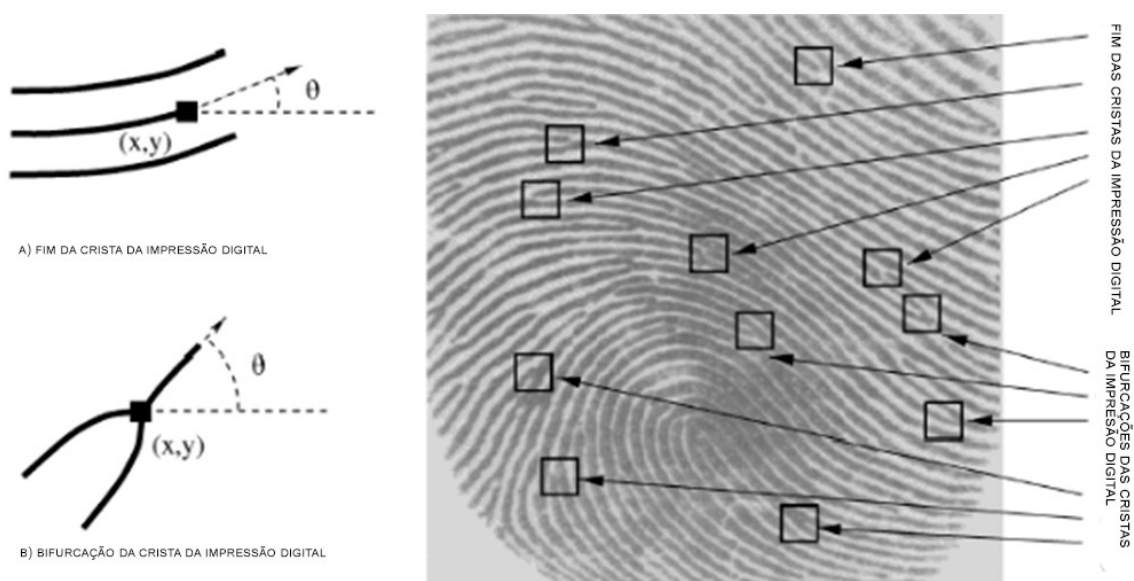
O processo de **aquisição** da imagem de uma digital é variado. Existem maneiras mais manuais, como a aquisição que é feita pressionando o dedo marcado de tinta em um papel que futuramente é escaneado para a identificação de um indivíduo. Mas existem também maneiras de obter a imagem da digital em “tempo real”, através de dispositivos eletrônicos que se utilizam das mais diversas tecnologias para tal atividade. As tecnologias utilizadas podem ser divididas em três: óptica, sensores e ultrassônica.

A **extração** das características da impressão digital varia e também serve para definir o projeto do sistema de identificação. Segundo Lourenço (2009) são três os tipos de abordagem para se obter tais individualidades, sendo elas: a abordagem global, abordagem local e a abordagem fina.

O processo de **comparação** em um sistema de identificação biométrico é a parte responsável por realizar a análise e comparação da impressão digital com um modelo previamente estabelecido no sistema. É nesta parte do sistema que podemos verificar a autenticidade de uma impressão digital e, portanto, identificar ou não o portador da digital. Podemos citar dois métodos de comparação que foram desenvolvidos por especialistas, o método baseado em imagens e o método baseado nas características.

O **método baseado nas imagens** é realizado por meio da superposição de duas imagens e então as comparações são computadas em diferentes posições. O **método baseado nas características** é o método com mais utilização e credibilidade. Se utiliza das minúcias presentes nas rugosidades (também chamadas de cristas) das impressões digitais para realizar uma comparação com o modelo registrado, realizando uma comparação da posição da *minúcia* (eixo vertical e horizontal) quanto da inclinação (ângulo de orientação) da mesma. A Figura 5 mostra as minúcias da impressão digital.

Figura 5 - Minúcias da impressão digital e suas posições e angulações.

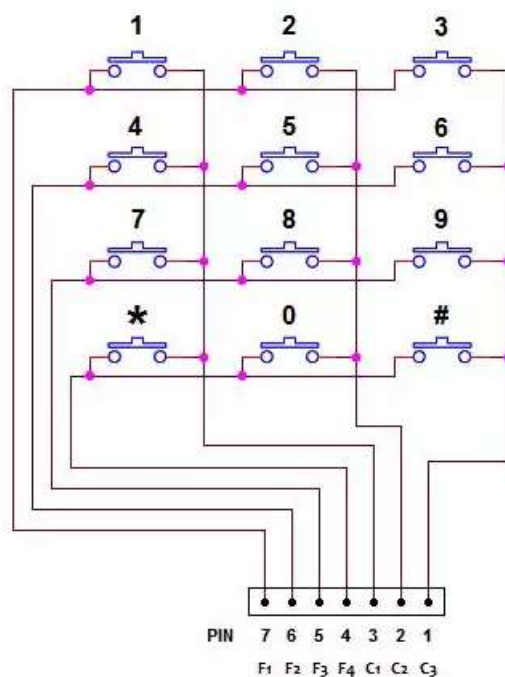


Fonte: Adaptado de UFRJ (2009).

3.3 Teclado Matricial

O teclado matricial é uma solução possível para qualquer aplicação que tenha a necessidade de operar um conjunto de botões. Para realizar a ligação de um *push-button* em um circuito comum são necessários dois terminais para de um botão. Este teclado surge como uma solução para o problema da quantidade de pinos utilizados de acordo com a quantidade de botões. A Figura 6 apresenta o esquema de ligação de um teclado matricial 3x4.

Figura 6 - Esquema de ligação de um teclado matricial 3x4.



Fonte: Adaptado de *Vistrónica* (2023).

A Figura 6 ilustra o arranjo dos botões dentro de um teclado matricial. O funcionamento do teclado matricial se dá através de vários *push-buttons* que são dispostos em linhas e colunas, formando uma matriz. Cada coluna possui um pino de ligação associado a quatro terminais de quatro botões e cada linha possui um pino de ligação associado a três terminais de três botões. A partir deste princípio estas trilhas são conectadas a pinos digitais dos microcontroladores, e existem diversas lógicas possíveis para a implementação deste teclado.

3.4 Microcontrolador ESP-WROOM-32

Desde 1970 iniciou-se o desenvolvimento de microcontroladores que se destacaram, e ainda se destacam pela sua importância e utilização no controle e automação. Tarefas antes feitas de forma manual puderam ser automatizadas devido ao uso de circuitos embarcados.

De acordo com Cardoso (2020), o microcontrolador “consiste em um único circuito integrado que reúne um núcleo de processador, memórias voláteis e não voláteis e diversos periféricos de entrada e de saída de dados.” Tudo isso em uma arquitetura compacta, e com uma performance eficaz para a realização de atividades que são comandadas pelo núcleo do processador e programáveis. Existem diferenças entre o microcontrolador e o microprocessador, pois, em concordância com Oliveira (2017):

Os microcontroladores têm interface de entrada e saída com dispositivos elétricos como botoeiras e relés, ao contrário dos microprocessadores de uso geral, normalmente dedicados a projetos de computadores com periféricos de entrada e saída como teclado, mouse e monitor.

Por conta do tamanho e preço destes dispositivos, eles possuem uma grande capacidade de integração com outras máquinas maiores. Isso foi de fundamental importância para a sua adoção dentro das aplicações de *IoT*, pois podem ser adicionados como periféricos de outros equipamentos, provendo “inteligência” ao dispositivo. Um dos microcontroladores mais populares atualmente dentro das aplicações de *IoT* é o ESP-WROOM-32.

O ESP 32 é um microcontrolador criado pela *Espressif Systems*. Este *chip* possui uma grande gama de funcionalidades que trazem a ele vantagens em relação a outros microcontroladores, como por exemplo o módulo *Wi-Fi* integrado. Isso permite a conexão deste microcontrolador com a internet sem a necessidade de incluir circuitos externos e por consequência ocupar um grupo de saídas e entradas digitais, além de possuir também um módulo *bluetooth* integrado.

O módulo *Wi-Fi* integrado ao ESP-WROOM-32, segundo os dados fornecidos pela *Espressif Systems* no *datasheet* do componente, tem capacidade de utilizar redes com o padrão *IEEE 802.11 b/g/n*, com possibilidade de transmissão de dados de até 150 Mbps. Porém ele só possui receptor e transmissor de 2.4 GHz, o que faz

com que não seja possível a comunicação com redes que operam na frequência de 5GHz, na sua configuração de fábrica.

Para fins de comparação, podemos observar na Tabela 1 as diferenças entre o ESP 32 e o ATMEGA328P (Arduino), um dos microcontroladores mais populares no mercado.

Tabela 1 - Comparação entre o ESP 32 e o ATMEGA328P.

Propriedades	ESP 32	ATMEGA328P
Núcleos	2	1
Arquitetura	32 bits	8 bits
<i>Clock</i> Máximo	240 MHz	16 MHz
<i>RAM</i>	512 KB	2 KB
<i>FLASH</i>	4 MB	32 KB
<i>GPIO</i>	36	23
Conversor A/D	18*	8
Interfaces	SPI, I2C, UART, CAN, SDIO	SPI, I2C, UART
Wi-Fi	Sim	Não
<i>Bluetooth</i>	Sim	Não

*Apenas 6 estão disponíveis caso o Wi-Fi seja utilizado

Fonte: Adaptado de DE SOUZA *et al.* (2021).

Como observado na Tabela 1, o ESP-WROOM-32 possui disponível uma memória *Flash* com capacidade de armazenamento de 4 MB, sendo 128 vezes a capacidade de armazenamento da memória *Flash* disponível no Atmega328P. Dentro das características da memória Flash, Codá (2013) destaca que:

A memória flash, embora seja uma EEPROM, é um tipo de memória com características muito semelhantes à memória RAM apenas com o diferencial

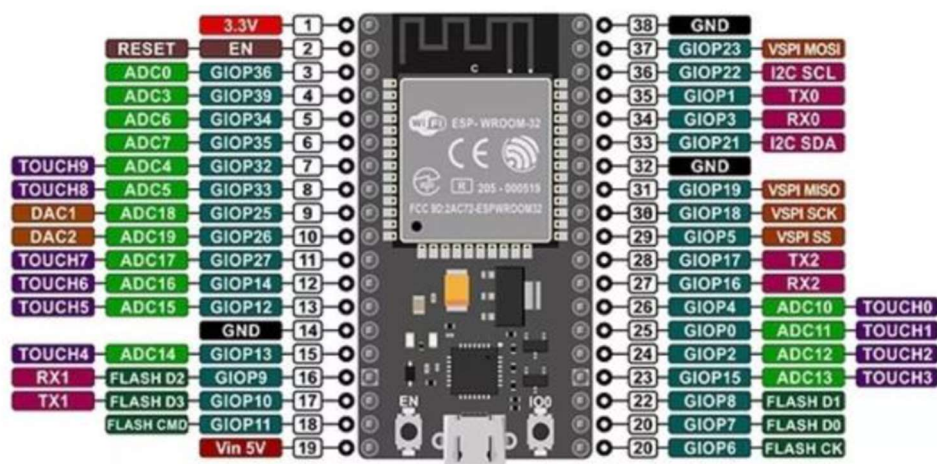
crucial de ser não volátil. Em outras palavras, isso significa que ela é capaz de preservar os dados armazenados por um longo tempo sem precisar de alimentação elétrica.

Para a operação correta dessa memória, o ESP-WROOM-32 precisa de um sistema de arquivos, que pode ser definido como a organização do conteúdo dos arquivos e diretórios dentro de um dispositivo físico (MAZIERO, 2019).

O sistema de arquivos que foi utilizado neste trabalho é o *LittleFS*, sistema sucessor do sistema de arquivos *SPiFFS*, que foi descontinuado e não recebe mais suporte da *Espressif Systems*. Para a implementação do *LittleFS*, foi utilizada a biblioteca *LittleFS.h*². O algoritmo implementado para o uso da memória foi completamente baseado nos exemplos disponibilizados na biblioteca.

Na Figura 7 temos todos os pinos disponíveis no ESP-WROOM-32, totalizando trinta e oito pinos, porém três pinos são reservados para o aterramento dos circuitos (GND), dois pinos fornecem tensão, sendo um com nível de 3,3V e outro com 5V. Os outros trinta e dois pinos são os responsáveis pelas entradas e saídas de sinais digitais e são chamados de *GPIO* (*General Purpose Input/Output*). Porém alguns pinos entre os *GPIO*'s podem desempenhar outras funções, possibilitando assim, múltiplas maneiras de comunicação com o ESP 32, como por exemplo as interfaces de comunicação *I2C*, *SPI* e *UART*.

Figura 7 - Pinos de ligação do ESP-WROOM-32.



Fonte: Extraído de BINKER et al. (2020).

² Disponível em: github.com/lorol/LITTLEFS.

3.5 Comunicação Serial

Dentro dos padrões de comunicação dos computadores um dos mais utilizados é a comunicação *serial*, que pode ser definida como o padrão de comunicação onde as informações são enviadas por um único meio de maneira sequencial, ou em série, daí o nome comunicação *serial* (Figura 8). De acordo com REIMBOLD, *et al* (s.d.) “a comunicação *serial* implica transferência não simultânea dos bits que compõem o byte, ou seja, um bit é transmitido após o outro”.

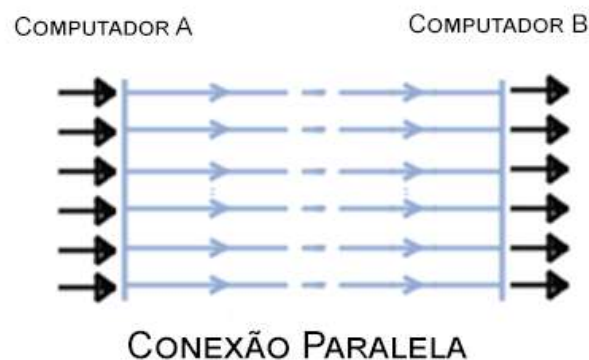
Figura 8 - Ilustração da comunicação *serial*.



Fonte: Adaptado de Porta Serial e Porta Paralela - CCM (2020).

Em contraste com esse tipo de comunicação existe também a comunicação em paralelo (Figura 9), onde os bits são transferidos em vias individuais. Em comparação com a comunicação paralela, a comunicação *Serial* exige menos vias de transmissão, porém isso faz com que o processo seja mais lento em relação ao padrão de comunicação paralelo.

Figura 9 - Ilustração da comunicação paralela.



Fonte: Adaptado de Porta Serial e Porta Paralela - CCM (2020).

Rabello (2009) afirma que “as comunicações seriais incluem a maioria dos dispositivos de rede, teclados, mouses, modems e terminais.” Além disso, é importante citar os microcontroladores, que muitas vezes são equipados com tecnologias que utilizam a comunicação *Serial*, por exemplo o ATMEGA328p (microcontrolador do Arduino) e o ESP-WROOM-32, que possuem as tecnologias *SPI*, *I2C* e *UART*. Essa são interfaces de comunicação que utilizam a comunicação serial.

3.5.1 Comunicação *UART*

O protocolo de comunicação *UART* (*Universal Asynchronous Receiver/Transmitter*), que significa Receptor/Transmissor Universal Assíncrono, é um padrão de comunicação de duas vias, uma de transmissão (TX) e outra de recepção (RX), responsáveis pela troca de informações na forma serial e assíncrona, ou seja, não existe um *clock* para sincronizar a transmissão de dados. GOLER (2004) sintetiza bem o protocolo *UART*: “*UART* consiste em um transmissor que transmite dados seriais através de um pino de transmissão e um receptor, que recebe dados seriais através de um pino receptor de dados”.

Pelo fato de não possuírem *clock* compartilhado, os dispositivos que implementam o protocolo *UART* precisam transmitir ao mesmo tempo e na mesma velocidade predefinida. Além disso, os dispositivos devem possuir a mesma estrutura de transmissão dos dados. Esse protocolo possui sua própria maneira de organizar os bits de transmissão que está demonstrada na Figura 10.

Figura 10 - Estrutura do pacote de dados do protocolo *UART*.



Fonte: Acervo do autor (2022).

Devido a assincronia do protocolo *UART*, é necessário a sinalização dos bits recebidos através do bit inicial, pois ele é uma transição do estado alto para o estado

baixo do sinal. Após o término dos bits de dados, o bit final indicará o fim dos dados do usuário, com isso o bit de parada será uma transição de retorno ou permanência do estado alto. Há a opção de configuração de um segundo bit final para a preparação do próximo *frame*, possibilitando mais tempo para o receptor.

Os *bits* de dados, também conhecidos como dados de usuário, são os bits úteis que seguem imediatamente após o *bit* inicial. Geralmente, podem variar de cinco a nove bits, sendo comum a utilização de sete ou oito bits. No protocolo UART, existem os bits de paridade, que são opcionais e têm a finalidade de detecção de erros. Esses bits são inseridos entre os bits de dados e o bit final. O valor do bit depende do tipo de paridade escolhido, podendo ser par ou ímpar.

3.5.2 Comunicação SPI

“A *Serial Peripheral Interface* (conhecida como *SPI*) foi desenvolvida pela Motorola para prover uma interface simples e de baixo custo para microcontroladores e *chips* periféricos” (Catsoulis, 2005).

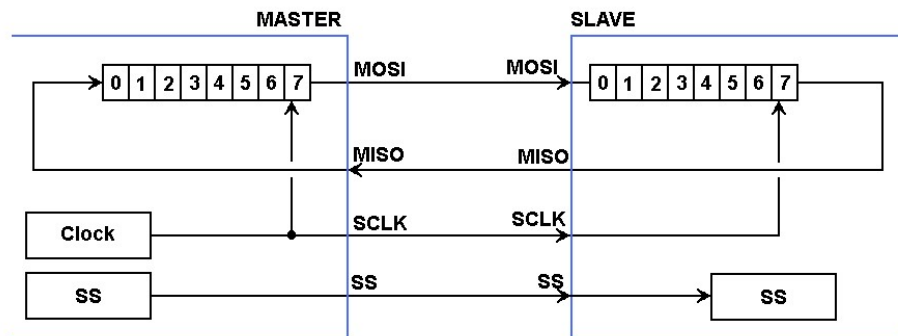
O protocolo de comunicação *SPI* é um protocolo síncrono que utiliza comunicação *serial*. Nele, a sincronia entre os dispositivos é estabelecida por meio de um pino de *clock*. Um dos dispositivos, conhecido como mestre, controla o *clock* de operação, enquanto o outro dispositivo, chamado de escravo, segue o ritmo desse *clock*. Dessa forma, o protocolo *SPI* segue o padrão de comunicação mestre-escravo. Como um exemplo disto, um microcontrolador pode atuar como mestre, enquanto um sensor ou um *display* pode ser um dispositivo escravo.

A Figura 11 ilustra o funcionamento dos pinos de comunicação na implementação do protocolo *SPI*, e para haver a comunicação através do padrão *SPI* são necessários no mínimo quatro conexões, que são chamadas:

- *MISO (Master Input Slave Output)*: Onde é realizada a entrada de dados do mestre e a saída de dados do escravo;
- *MOSI (Master Output Slave Input)*: Onde é realizada a saída de dados do mestre e a entrada de dados do escravo;
- *CLK (Clock)*: Onde é emitido o sinal que garante a sincronia de operação entre os dispositivos;

- **SS (Slave Select):** Pino necessário para ativar a operação de um escravo e selecionar qual dispositivo será comandado.

Figura 11 - Comunicação entre mestre e escravo no protocolo *SPI*.



Fonte: Acervo do autor (2022).

A troca de dados do protocolo *SPI* está bem ilustrada na Figura 11, onde temos os bits do mestre sendo enviados para o escravo na porta *MOSI* e os bits do escravo sendo enviados para o mestre a partir da porta *MISO*. Essa operação (a troca de bits) só é permitida de acordo com a atuação do *clock*.

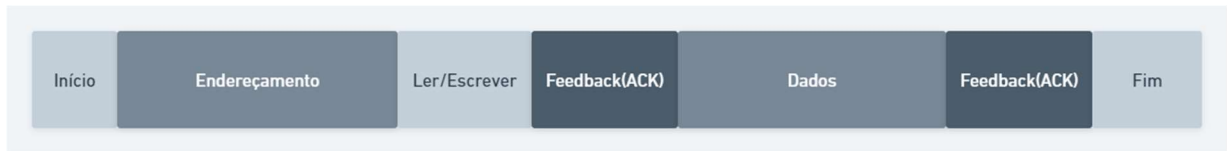
O modelo de comunicação *SPI* possibilita a comunicação entre um mestre e múltiplos escravos, porém, cada novo dispositivo escravo é necessário a inserção de um novo pino *SS* no mestre para realizar a seleção do dispositivo que se deseja operar.

3.5.3 Comunicação *I2C*

O *I2C* (*Inter-Integrated Circuit*) ou Circuito Inter-Integrado, também conhecido como *TWI* (*Two Wire Interface*), é um protocolo de comunicação serial síncrona desenvolvido pela *Phillips* na década de 1980. Ele permite a comunicação entre periféricos em um mesmo circuito, utilizando apenas duas saídas: *SCL* (*Serial Clock*) para controlar o sistema por meio de um sinal de *clock* gerado pelo dispositivo mestre, e *SDA* (*Serial Data*) para transmitir dados nos dispositivos escravos durante operações de escrita e leitura.

O pacote de dados transmitido pelo barramento *SDA* tem o formato ilustrado pela Figura 12.

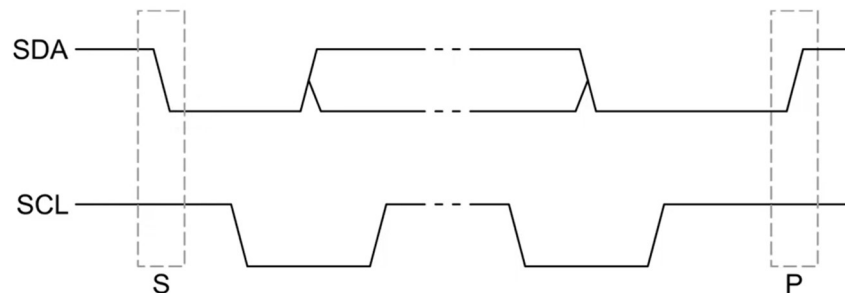
Figura 12 - Formato de dados de transmissão do protocolo I2C.



Fonte: Acervo do autor (2022).

Na transmissão dos dados, primeiramente é realizado um comando de início que acontece quando o pino *SDA* alterna do nível alto para o nível baixo enquanto o barramento *SCL* está com nível alto como mostrado na Figura 13 no trecho S. Após esta operação o pino de dados começa a enviar informações, porém esta troca de dados só deve ocorrer enquanto o *clock* estiver com nível baixo, pois fora disso, a comutação é interpretada como um comando de início ou de fim. Quando o *SDA* alterna para nível alto enquanto o barramento *SCL* está com o nível alto, isso é considerado um comando de fim, como mostrado na Figura 13 no trecho P.

Figura 13 - Critério de início e fim de transmissão no protocolo I2C.



Fonte: Mattioli (2020).

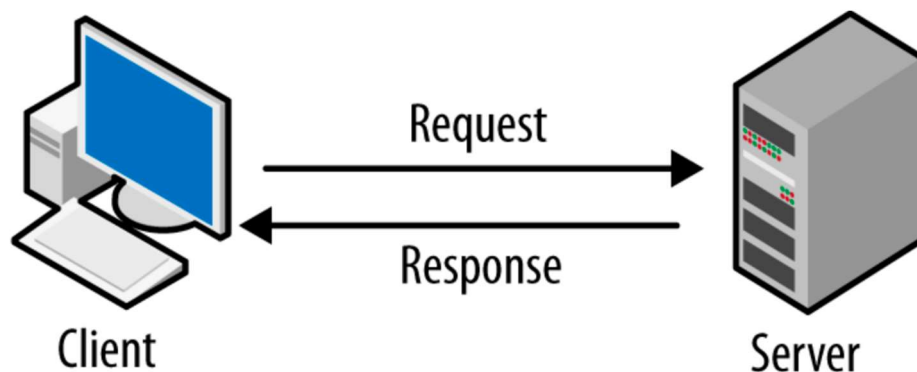
O fluxo de transmissão dos dados segue o esquema apresentado na Figura 13. Inicialmente, é enviado o endereço do dispositivo escravo com o qual a operação será realizada. Em seguida, um *bit* é transmitido para indicar se a operação será de escrita ou leitura de dados. Posteriormente, o dispositivo escravo envia um feedback, conhecido como *ACK (acknowledge)*, que serve para confirmar o reconhecimento dos dados. Em seguida, ocorre a transmissão dos dados propriamente ditos. Ao final de cada bloco de transmissão, o *bit* de *feedback* é novamente enviado, continuando até que o mestre ordene o sistema a encerrar o processo através de um comando de fim.

3.6 Ferramentas de desenvolvimento web

Com a popularização da internet, o desenvolvimento de aplicações *web* (*softwares* que são acessados via navegador) está cada vez mais popular, tendo possibilidade de acesso através de diversos dispositivos como celulares, computadores de mesa e tablets.

Para acessar estas aplicações *web* nós utilizamos o protocolo de comunicação *HTTP* que é definido pela documentação do Mozilla como: “um protocolo que permite a obtenção de recursos, como documentos *HTML*. É a base de qualquer troca de dados na *web* e um protocolo cliente-servidor”. A estrutura cliente-servidor (Figura 14) é essencial para entendermos o desenvolvimento de uma aplicação *web*.

Figura 14 - Ilustração da estrutura cliente-servidor.



Fonte: Darvish (2022)

Do lado do cliente (*client-side*), temos o *front-end* que é a aplicação responsável por prover uma interface amigável ao usuário para que o usuário interaja com a aplicação realizando requisições para o servidor. Do lado do servidor (*server-side*) temos o *back-end*, uma aplicação responsável por trabalhar com as requisições recebidas pelo lado do cliente, aplicar as regras de negócio da aplicação (lógica) e também gerenciar os dados da aplicação (comunicação com o banco de dados).

As tecnologias fundamentais para a criação de um *front-end* de um *site* são *HTML*, *CSS* e *Javascript*. Além destas tecnologias, existem diversos *frameworks* e bibliotecas que agilizam e auxiliam o processo de construção destas aplicações.

A documentação do navegador Mozilla define o *HTML* como “o bloco de construção mais básico da *web*. Define o significado e a estrutura do conteúdo da

web". Sendo esta a tecnologia mais básica do desenvolvimento *web*, necessária para a estruturação do conteúdo das páginas. Já o *CSS (Cascading Style Sheets)* é a ferramenta utilizada para a estilização dos elementos criados pelo *HTML*.

O *Javascript* é uma linguagem de programação de alto nível que foi criada voltada para ser executada em navegadores e, portanto, adicionar dinamicidade e inteligência às aplicações *web*. Roveda (2020) afirma que através do *Javascript* "é possível incluir, em uma página estática, elementos dinâmicos como mapas, formulários, operações numéricas, animações, infográficos interativos e muito mais".

As aplicações *front-end* de uma aplicação *web* são executadas dentro de um *browser*, mas esta limitação não se estende às aplicações *back-end* pois as mesmas são executadas dentro do servidor. Portanto o para desenvolvimento do *back-end* pode-se utilizar diversas tecnologias, por exemplo *Python*, *C#* e desde 2009 há também a possibilidade utilizar *Javascript* em aplicações *back-end* através do *software Node.js*.

3.6.1 *Framework* e Bibliotecas

Visando agilizar o processo de desenvolvimento de um *software*, existem diversas ferramentas que podem auxiliar nesta tarefa *frameworks* e bibliotecas possuem este foco, agilizar a produção de *softwares*, porém, tais instrumentos possuem suas diferenças.

Sommerville (2011) define *framework* como "uma estrutura genérica estendida para se criar uma aplicação ou subsistema mais específico". Sendo assim, o *framework* é a ferramenta que constrói para o desenvolvedor uma estrutura base para o projeto que irá ser desenvolvido e além disso tem controle sobre o fluxo das tarefas que serão realizadas, portanto, é criada uma "camada" entre o desenvolvedor e a aplicação, sendo assim, toda modificação feita na aplicação será através do *framework*.

Já a biblioteca pode é definida por Duarte (2015) como "um repositório de funcionalidades a que o desenvolvedor tem acesso para desenvolver a aplicação", ou seja, um conjunto de funções, que podem ser agregadas ao projeto de acordo com o critério do desenvolvedor.

Portanto a principal diferença entre estes dois conceitos segundo Wozniewicz (2019) é que:

Quando você usa uma biblioteca, você é responsável pelo fluxo da aplicação, você está escolhendo quando e como chamar a biblioteca. Quando você usa um *framework*, o *framework* está responsável pelo fluxo. Ele provê alguns lugares onde você pode encaixar seu código, mas chama este código somente quando é necessário.

No desenvolvimento *web* temos diversos *frameworks*, como por exemplo o *Angular*, criado em 2016 pela Google, ou também o Vue JS, outro *framework* popular no desenvolvimento *web*. Em relação às bibliotecas, se destacam o *JQuery*, criado em 2006 chegando a ser uma das bibliotecas mais utilizadas no desenvolvimento *web*, e mais recentemente temos o *React.JS*, uma biblioteca criada em 2013 pelo *Facebook* e tem se tornado uma das ferramentas mais populares utilizadas pelos desenvolvedores.

3.6.2 *React.JS*

A documentação do *React* o define como: “uma biblioteca *Javascript* para construção de interfaces de usuário”. Lançada em 2013 pelo time de desenvolvedores do Facebook, o *React* foi uma biblioteca criada com o objetivo de construir *SPA's* (*Single Page Applications*).

Single Page Applications são aplicações que possuem seu funcionamento centrado em uma página. Resumindo “*Single page application* quer dizer que o *browser* vai renderizar o *core* da aplicação apenas uma vez – todas as outras informações serão carregadas por demanda, de acordo com a necessidade do usuário” (BAILÃO, 2009).

Por conta de sua proposta o *React* é uma biblioteca baseada em componentes, que são basicamente, trechos de código (funções) que podem ser chamadas em várias partes da mesma página. As vantagens desta biblioteca são diversas, além da economia de código, a própria documentação do *React* aponta que os componentes “permitem você dividir a *UI* em partes independentes, reutilizáveis e pensar em cada parte isoladamente”. *UI* vem do inglês *User Interface*, ou seja, interface de usuário, e representa a interface que interage com o usuário da aplicação.

Porém, esta biblioteca também possui limitações, como sua proposta é focada na construção de *Single Page Applications* a aplicação necessita de outras bibliotecas para realizar a navegação entre páginas diferentes, e também para o compartilhamento de variáveis entre os componentes. Para navegação geralmente é utilizada a biblioteca *React Router*³, já para o compartilhamento de variáveis existem diversas bibliotecas como o *Redux*⁴ e a *Context API*⁵ uma solução criada pelos próprios desenvolvedores do *React*.

3.6.3 API

API é uma sigla para o termo em inglês *Application Programming Interface* (Interface de Programação de Aplicação). A documentação da *AWS (Amazon Web Services)* define *APIs* como: “mecanismos que permitem que dois componentes de *software* se comuniquem usando um conjunto de definições e protocolos.” As *APIs* desempenham o papel de criar uma interface entre os dados de uma aplicação e os consumidores desses dados. Isso permite que outro *software* possa interagir com esses dados, consumindo-os e modificando-os, sem precisar conhecer a organização e o processo de recuperação dos dados, pois essa responsabilidade é atribuída à *API*.

Dentro de uma aplicação *web* a *API* recebe as solicitações enviadas pelo cliente e de acordo com a solicitação enviada retorna um dado específico. A documentação do *Red Hat* (2017) explica que:

“As *APIs* funcionam como se fossem contratos, com documentações que representam um acordo entre as partes interessadas. Se uma dessas partes enviar uma solicitação remota estruturada de uma forma específica, isso determinará como a aplicação da outra parte responderá.”

A Figura 15 ilustra uma das principais vantagens do uso de uma *API* para o desenvolvimento de um *software*, que é a liberdade de poder interagir com a base de dados de múltiplas maneiras. Na figura citada temos o exemplo de três *softwares* diferentes: um *site*, um aplicativo de celular e um programa para computadores

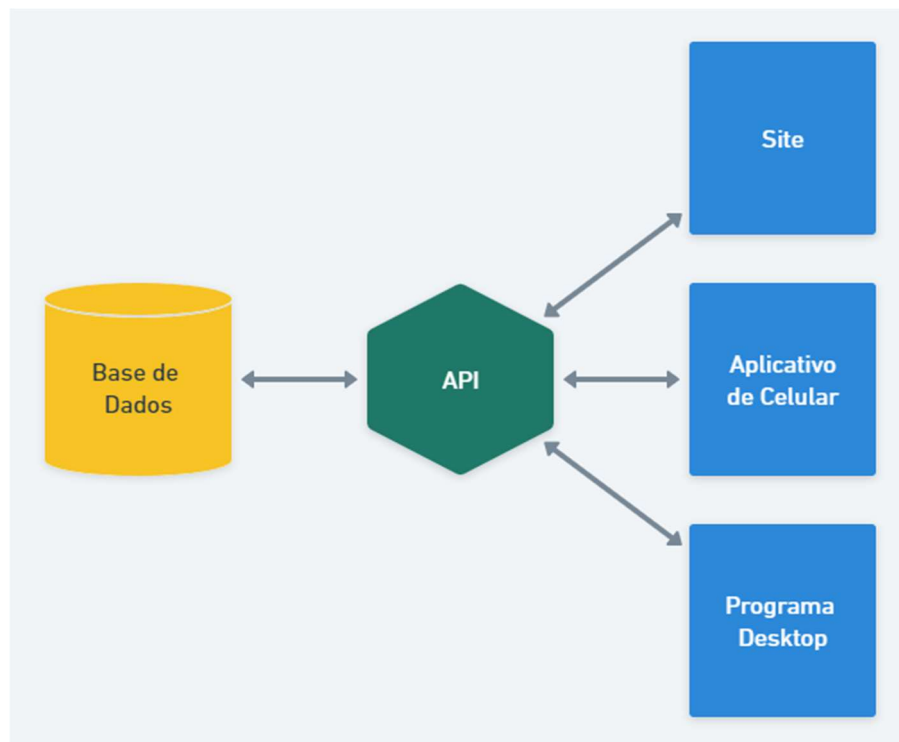
³ Disponível em: v5.reactrouter.com/;

⁴ Disponível em: redux.js.org/;

⁵ Disponível em: pt-br.reactjs.org/docs/context.html.

desktop. Todos os *softwares* exemplificados podem interagir com uma base de dados específica de maneira unificada e centralizada. Isso previne que cada aplicação que precise interagir com aquela base utilize uma regra de negócio unificada, evitando conflitos entre as funcionalidades das diversas plataformas.

Figura 15 - Interação da *API* com múltiplos *softwares*.



Fonte: Acervo do autor (2022).

Um dos padrões mais populares de implementação de uma *API* é chamado de *REST* (*Representational State Transfer*) que pode ser traduzido como Transferência Representacional de Estado. “*REST* define um conjunto de funções como *GET*, *PUT*, *DELETE* e assim por diante, que os clientes podem usar para acessar os dados do servidor. Clientes e servidores trocam dados usando *HTTP*.” (AWS, 2022). Além disso, o padrão de desenvolvimento de *APIs REST* compartilha seus dados no formato *JSON* (*Javascript Object Notation*), que pode ser interpretado em diversas linguagens de programação além do Javascript, por exemplo *Python*, *PHP* e *C#*.

3.6.4 Node.JS

Node.JS é um *software* de código aberto, que foi lançado no ano de 2009, e sua principal funcionalidade é executar a linguagem *Javascript* fora de um *browser*,

trazendo assim diversas possibilidades do uso do *Javascript*, como por exemplo, a construção de *APIs* e outras aplicações que anteriormente não eram possíveis por conta da limitação do *Browser*.

O funcionamento do *Node.JS* é descrito em sua documentação da seguinte maneira: “O *Node.JS* executa o motor V8 do *Javascript*, o núcleo do Google *Chrome*, fora do *browser*. Isso permite o *Node.JS* ser muito performático”.

Uma das principais vantagens do *Node.JS* é a variedade de bibliotecas que podem ser utilizadas. O *Node.JS* tem seu próprio gerenciador de pacotes o *NPM*⁶ (*Node Package Manager*) gerenciador de pacotes do Node, que permite a fácil instalação de diversas bibliotecas. Além disso, permite ao desenvolvedor construir uma aplicação *web* completa (*back-end* e *front-end*) utilizando a mesma linguagem, o *Javascript*.

3.6.5 Firebase

O *Firebase* é uma plataforma que fornece diversas ferramentas para a construção de aplicações, sendo criada em 2012 por uma empresa privada e adquirida pelo Google em 2014. O *Firebase* é classificado como um BaaS (Back-end as a Service), ou seja, um Back-end como serviço. Segundo a definição de Orlandi (2018), BaaS é:

[...] um serviço disponibilizado em que toda a estrutura do *backend* como: configuração de servidor, integração com banco de dados, sistema de *push notification* e outros serviços, que fazem parte do *backend*, estão completamente prontos para se integrar com o seu aplicativo.

Sendo assim o *Firebase* fornece uma gama de serviços que aceleram e auxiliam o desenvolvimento de aplicações. De acordo com o seu *site* seus serviços são divididos em três grupos:

- *Build* (Construir): Onde são oferecidos diversos serviços voltados para o desenvolvimento, como banco de dados em tempo real, sistema de autenticação e armazenamento em nuvem;

⁶ Disponível em: npmjs.com.

- Liberar e Monitorar: Onde são fornecidos serviços voltados para a análise da aplicação como por exemplo o Google *analytics* e o *Firebase Performance monitoring*;
- Engajamento: Onde são fornecidas ferramentas para analisar e aumentar o engajamento do usuário na plataforma, como por exemplo testes A/B.

3.6.6 *Firebase Realtime Database*

Firebase Realtime Database (Banco de dados em tempo real do *Firebase*), é um dos serviços oferecidos pela plataforma *Firebase* e fornece ao desenvolvedor um banco de dados *NoSQL* (não relacional) hospedado na nuvem e segundo sua própria documentação, “os dados são armazenados como *JSON* e sincronizados em tempo real para cada cliente conectado”.

Isso permite que um *software* seja desenvolvido sem a preocupação da implementação e hospedagem de um banco de dados. Assim o desenvolvedor tem acesso a uma base de dados disponível a todo momento, e deixando a cargo do *Firebase* o armazenamento dos dados da aplicação desenvolvida.

3.6.7 *Firebase Authentication*

Firebase Authentication (Autenticação do *Firebase*), é um serviço que provê um sistema de autenticação pronto para o desenvolvimento de aplicações, ou seja, possui todas as ferramentas necessárias para cadastrar, validar e gerenciar os usuários de uma aplicação.

Além disso, uma das grandes vantagens da utilização do *Firebase Authentication*, conforme a documentação, é que o serviço fornece “suporte à autenticação usando senhas, números de telefone, provedores de identidade federados conhecidos, como Google, Facebook e Twitter, entre outros”. Possuindo também um serviço de alta disponibilidade por ser controlado por um servidor robusto oferecido pelo Google.

4 DESENVOLVIMENTO DO PROTÓTIPO

No presente trabalho é proposto um sistema de controle de acesso a ambientes integrado com uma fechadura eletrônica que pode ser aberta de quatro maneiras diferentes: *tags RFID*, senhas numéricas, impressão digital e/ou comando remoto através do *site*. O controle de acesso é gerenciado por um usuário administrador através de uma plataforma *web*, que pode controlar quais convidados podem ter acesso àquele ambiente que adotou a fechadura eletrônica. Através do sistema o usuário operador poderá verificar o histórico de acessos e de tentativas de abertura, além de poder operar remotamente o dispositivo. Dessa maneira o usuário convidado poderá ter acesso ao ambiente, com diversas alternativas de autenticação no sistema.

Durante este capítulo, para fins didáticos, foi nomeado “operador” o usuário que interage com a aplicação *web*, que pode comandar a fechadura e verificar os dados da mesma. Já o usuário que interage diretamente com a fechadura eletrônica foi nomeado como “convidado”.

O sistema de controle de acesso proposto pode ser estruturado em três grandes partes, sendo essas: a central de controle e monitoramento, a central de comunicação e a fechadura eletrônica. Pode-se então dividir o projeto em duas partes: o desenvolvimento dos *softwares* (central de controle e monitoramento e a central de comunicação) e o desenvolvimento do *hardware* (fechadura eletrônica). Cada parte do sistema tem sua função específica e sua maneira de comunicação, a Figura 16 ilustra a relação entre as três partes do sistema proposto.

Figura 16 - Diagrama simplificado do sistema proposto.



Fonte: Acervo do autor (2022).

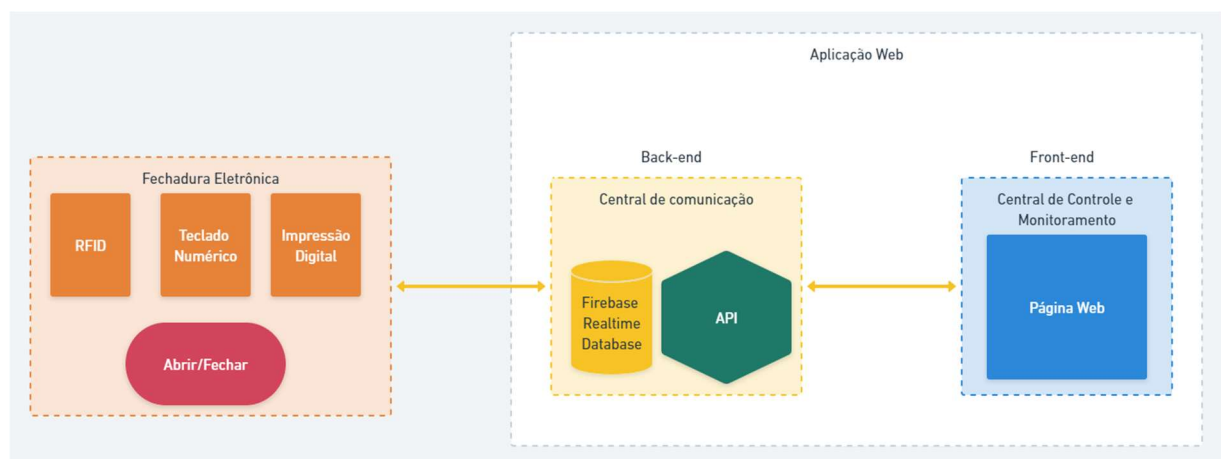
A **central de controle e monitoramento** é o *software* responsável por ser a interface entre o sistema e o usuário “operador”. É neste bloco onde o “operador” poderá consultar o histórico de operações de uma fechadura.

A **central de comunicação** é a parte responsável por coordenar as informações enviadas pelo usuário “operador” na central de controle e as informações enviadas pelo usuário “convidado”.

A **fechadura eletrônica** é o dispositivo eletrônico comandado por um microcontrolador responsável tanto por se comunicar com a central de comunicação quanto por controlar a abertura da própria fechadura através das senhas colocadas no teclado numérico, dos cartões *RFID*, das impressões digitais e dos comandos emitidos pela central de controle e monitoramento.

A aplicação *web* é composta por duas partes distintas: a central de controle e monitoramento, responsável pelo *front-end*, e a central de comunicação, responsável pelo *back-end*. A central de controle e monitoramento é uma página *web* desenvolvida utilizando a biblioteca *React.JS*, enquanto a central de comunicação é uma *API* construída com *Node.JS* que se conecta a um banco de dados *Firebase*. Após a compreensão detalhada do funcionamento destas três partes do sistema, o diagrama de blocos pode ser mais bem definido de acordo com a Figura 17.

Figura 17 - Diagrama completo do sistema proposto.



Fonte: Acervo do autor (2022).

Tabela 2 - Conexão dos pinos do microcontrolador com os demais elementos do protótipo.

Pinos do ESP WROOM 32	Função
3V3	Alimentação para os módulos RC522 e DY50
5V	Alimentação do <i>Display LCD I2C</i>
IO12, IO14, IO27, IO26, IO25, IO33 e IO32	Pinos de conexão do teclado matricial (3x4)
IO04	Pino de controle de abertura da porta
GND	Pino de aterramento
IO23, IO19, IO18, IO05 e IO13	Pinos de ligação do módulo RC522
IO22 e IO21	Pinos de ligação do <i>Display LCD (I2C)</i>
IO17 e IO16	Pinos de ligação do módulo DY50

Fonte: Acervo do autor (2022).

Como apontado na tabela 2 a *IO04*, será a saída responsável por controlar a abertura da fechadura eletrônica, podendo ela ser ligada a um relé, ou algum outro dispositivo que possa comandar a abertura e bloqueio de uma porta.

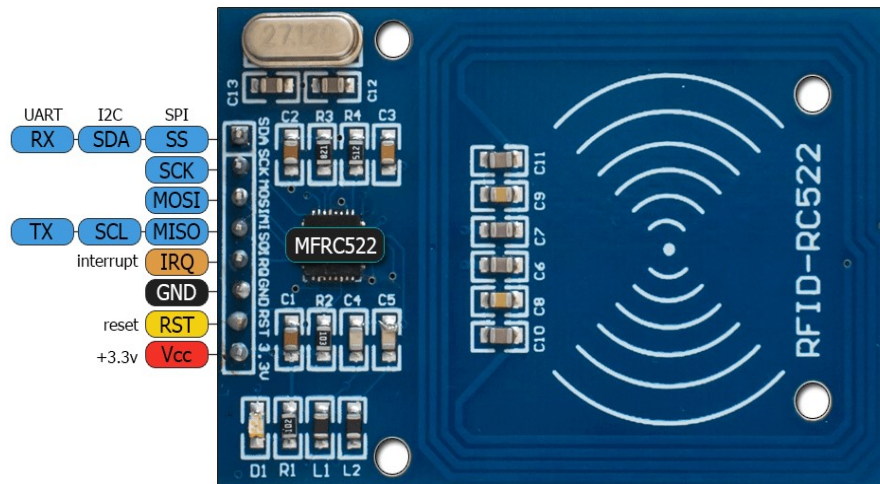
Uma das funcionalidades que também foi explorada no ESP-WROOM-32 é a sua memória *Flash* de 4MB. O uso desta memória será voltado para o funcionamento do protótipo em cenários onde a conexão com a internet é perdida e precisa-se armazenar as permissões e históricos *offline* para manter o funcionamento correto.

4.1.1.2 Módulo leitor *RFID*-RC522

Dentro do dispositivo o módulo leitor *RFID* - RC522 será o responsável pela leitura dos cartões, *tags* ou etiquetas *RFID*, que precisam ser previamente configurados para armazenar o código de acesso do “convidado”.

O módulo leitor *RFID* - RC522 apresentado na Figura 19 utiliza o chip MFRC522 da empresa *NXP* e é amplamente utilizado. Este leitor *RFID* opera em uma frequência de 13,56MHz, permitindo uma comunicação sem necessidade de contato entre o transponder e o leitor. Para mais, o módulo pode se comunicar com um microprocessador através do protocolo *SPI*.

Figura 19 - Módulo *RFID* RC522 com pinagem sinalizada.



Fonte: Grieger (2018).

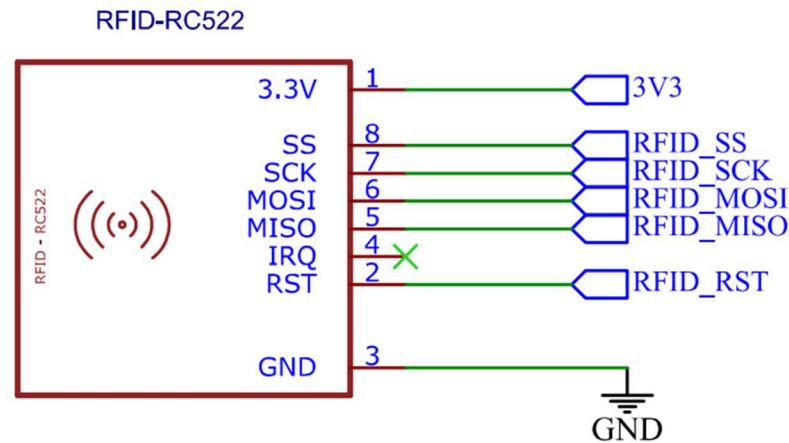
As especificações deste módulo são:

- Modelo: MFRC522;
- Tensão de operação: 3,3VDC;
- Corrente de operação: 13 mA a 26 mA;
- Tensão em modo inativo: 3,3VDC;
- Corrente em modo inativo: 10 mA a 13 mA;
- Frequência de operação: 13,56MHz;
- Interface: SPI;
- Taxa de transferência: 10Mbit/s;
- Alcance: 0 a 3 cm;
- Temperatura de operação: -20° a 80° celsius.

O módulo *RFID* RC522 possui oito pinos de ligação com o ESP 32, sendo dois pinos voltados para a alimentação do circuito que recebe 3,3V, e os outros seis pinos para realizar a operação do leitor. Este módulo irá se comunicar com o microcontrolador através do protocolo *SPI*, porém o pino *IRQ* não será utilizado para

este dispositivo. A disposição dos pinos do módulo está exibida na Figura 20 e suas conexões com o ESP 32 na Tabela 3.

Figura 20 - Esquemático de ligação do módulo leitor *RFID* para a fechadura eletrônica.



Fonte: Acervo do autor (2023).

Tabela 3 - Conexão dos pinos do microcontrolador com o módulo RC522.

Pinos do ESP WROOM 32	Pinos do módulo <i>RFID</i> RC522
IO05	SS
IO18	SCK
IO19	MISO
IO23	MOSI
IO13	RST
3V3	3V3
GND	GND

Fonte: Acervo do autor (2022).

Para a programação e uso deste módulo foram necessárias duas bibliotecas, a SPI.h⁷, responsável por implementar a interface de comunicação Serial SPI, e para as funcionalidades específicas do módulo foi necessária a biblioteca MFRC522.h⁸.

3.1.1.3 Módulo leitor biométrico DY50

Este módulo, mostrado na Figura 21, será responsável tanto pelo armazenamento quanto pela identificação da impressão digital do “convidado”. Como este módulo armazena a impressão digital do usuário localmente, é necessário então que o “convidado” que deseje utilizar a impressão digital para ter acesso aos ambientes cadastre previamente sua digital na fechadura que deseja ter acesso, e assim sua impressão pode ser associada a seu código de acesso, ou a algum outro identificador da digital apresentada.

A capacidade máxima de armazenamento deste módulo é 162 impressões digitais, o que cria uma limitação para o dispositivo proposto de apenas 162 impressões digitais com a opção de acesso através da biometria.

Figura 21 - Módulo leitor biométrico DY50.



Fonte: Casa da Robótica (2022).

⁷ Disponível em: github.com/arduino/ArduinoCore-avr/tree/master/libraries/SPI;

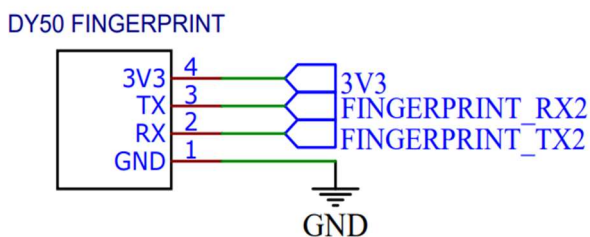
⁸ Disponível em: github.com/miguelbalboa/rfid.

As especificações deste módulo são:

- Modelo: DY50;
- Tensão de alimentação DC: 3,6 a 6V;
- Corrente de operação: <120 mA;
- Corrente de pico máx: 150 mA;
- Tempo de impressão digital de imagem: <1,0 segundo;
- Capacidade de armazenamento on-board: 162 digitais;
- Nível de segurança configurável: 1-baixo, 3-médio, 5-alto;
- Taxa de aceitação falsa: <0,001% (nível de segurança 3);
- Taxa de rejeição falsa: <1,0% (nível de segurança 3);
- Interface: *UART*;
- Taxa de transmissão: 9600, 19200, 28800, 38400, 57600 (o padrão é 57600);
- Temperatura de operação: -20°C a +50°C;
- Umidade de trabalho: <85%;
- Comprimento do cabo: 150 mm;
- Área da janela de leitura: 14x18 mm;
- Dimensões totais: 55x 21x21 mm;
- Peso: 20g.

A pinagem deste módulo é simples serão necessárias somente quatro conexões com o microcontrolador, sendo uma para alimentação e uma para aterramento do circuito. As outras conexões do módulo se comunicarão através do protocolo *UART* com os pinos TX e RX. Sua pinagem está ilustrada na Figura 22 e na Tabela 4 pode-se conferir a conexão que este módulo terá com o ESP-WROOM-32.

Figura 22 - Esquemático de ligação do módulo leitor de impressões digitais para a fechadura eletrônica.



Fonte: Acervo do autor (2022).

Tabela 4 - Conexão dos pinos do microcontrolador com o módulo DY50.

Pinos do ESP-WROOM-32	Pinos do módulo DY50
IO16	TX
IO17	RX
3V3	3V3
GND	GND

Fonte: Acervo do autor (2022).

A programação deste módulo foi feita com o auxílio da biblioteca chamada `Adafruit_Fingerprint.h`⁹, sendo a responsável pela implementação do protocolo de comunicação *UART* para a correta operação do módulo DY50.

4.1.1.4 Teclado matricial de doze teclas

Este componente (Figura 23) servirá como uma interface de interação do “convidado” com a fechadura, possibilitando a escolha do método de abertura da porta e além disso poderá ser utilizado também para que o “convidado” possa inserir sua senha numérica para solicitar o acesso ao ambiente.

Figura 23 – Teclado matricial rígido de doze teclas.

Fonte: USINAINFO (2022).

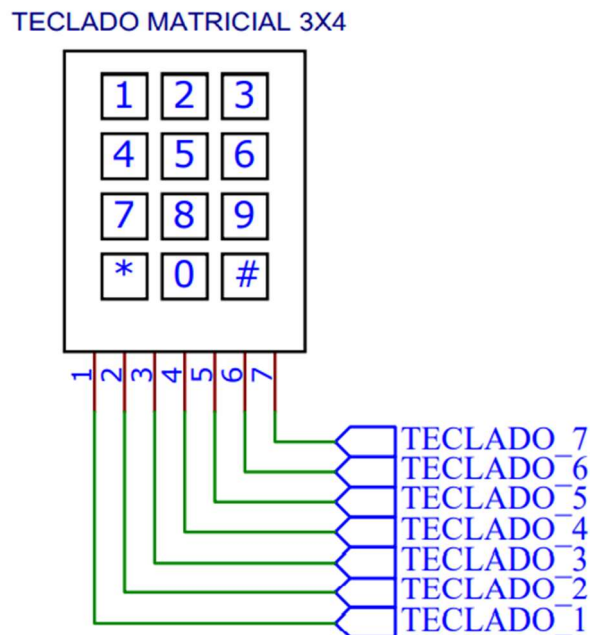
⁹ Disponível em: github.com/adafruit/Adafruit-Fingerprint-Sensor-Library.

As especificações deste teclado matricial são:

- Tensão máxima: 24V DC;
- Corrente de funcionamento: 20 mA;
- Temperatura: -20 a 60°C;
- Dimensões (CxLxE): 69x51x10 mm;
- Peso: 18g.

As conexões do teclado matricial são entradas e saídas digitais simples, como mostradas no esquemático da Figura 24. As conexões que serão realizadas com o microcontrolador estão organizadas na Tabela 5.

Figura 24 - Esquemático de ligação do teclado matricial para a fechadura eletrônica.



Fonte: Acervo do autor (2022).

Tabela 5 - Conexão dos pinos do microcontrolador com o módulo de do teclado matricial (3x4).

Pinos do ESP WROOM 32	Pinos do teclado matricial
IO32	1
IO33	2
IO25	3

IO26	4
IO27	5
IO14	6
IO12	7

Fonte: Acervo do autor (2022).

Para facilitar a implementação do algoritmo do teclado matricial, foi utilizada uma biblioteca chamada Keypad.h¹⁰, que faz todo o trabalho de configurar os pinos de entrada e saída dos *push-buttons* e associar tudo isso a uma matriz de valores que representam as teclas do teclado matricial.

4.1.1.5 Display LCD (16x2) com módulo I2C acoplado

O *display LCD* (Figura 25) usado neste dispositivo terá como objetivo estabelecer uma simples interface de interação com o “convidado” e juntamente com o teclado matricial permitir ao usuário realizar algumas operações na fechadura eletrônica. O *display* terá a função de aprimorar a experiência de usuário do “convidado”, para que ele possa ter um *feedback* das operações do aparelho.

Figura 25 - Frente e verso do *display LCD* (16x2) com o módulo I2C acoplado.



Fonte: FilipeFlop (2022).

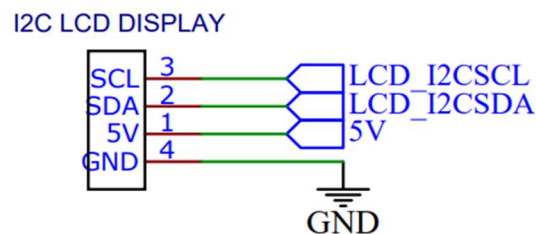
As especificações deste módulo são:

¹⁰ Disponível em: playground.arduino.cc/Code/Keypad/.

- Cor da Luz de fundo: azul;
- Cor dos caracteres: branco;
- Controlador: HD44780;
- Adaptador *display I2C* integrado;
- Potenciômetro para ajuste do contraste;
- Tensão de operação: 5V;
- Linhas: 2;
- Colunas: 16;
- Interface: I2C;
- Dimensões: 80 x 36 x 12 mm;
- Área visível: 64,5 x 16mm;

A pinagem deste componente está apresentada na Figura 26, pode-se notar que os dois pinos são responsáveis por transmitir as informações utilizando o protocolo *I2C*. As conexões dos pinos do microcontrolador com o módulo estão dispostas na Tabela 6.

Figura 26 - Pinagem do *display LCD*.



Fonte: Acervo do autor (2022).

Tabela 6 - Conexão dos pinos do microcontrolador com o módulo *LCD I2C*.

Pinos do ESP-WROOM-32	Pinos do módulo <i>LCD I2C</i>
IO22	SCL
IO21	SDA
GND	GND
5V	5V

Fonte: Acervo do autor (2022).

Para o uso deste periférico na programação do ESP-WROOM-32, foi utilizada a biblioteca `LiquidCrystal_I2C.h`¹¹ que já prevê o uso do *display LCD* acoplado ao driver *I2C* e tem suas funções voltadas para a operação do dispositivo utilizado neste trabalho.

4.1.2 Algoritmo da fechadura eletrônica

Para que seja possível a fechadura eletrônica realizar as operações necessárias dentro do sistema, foi elaborado um algoritmo da lógica de programação implementada no *hardware*. O microcontrolador ESP-WROOM-32 foi programado através da *IDE* do Arduino, já que este programa vem com suporte para este microcontrolador. Inicialmente é necessário realizar a instalação das bibliotecas necessárias, a configuração da conexão com a rede *Wi-Fi* e a configuração para a utilização de cada dispositivo.

Inicialmente é necessário a configuração inicial de alguns dispositivos periféricos do protótipo, as *tags* e cartões *RFID* que serão utilizadas pelos convidados e o sensor de leitura de impressões digitais. Para configurar as *tags* e cartões *RFID*, foi utilizado o código presente no artigo “*Identificação por impressão digital com Arduino*” de Koyanagi (2020), onde se pode escrever o código de permissão do usuário no cartão *RFID*.

Para realizar o cadastro das impressões digitais através do sensor é necessário entender que o sensor DY50 possui capacidade de armazenar 162 impressões digitais. Para elaborar o algoritmo foi presumido que cada “convidado” possa cadastrar todas as impressões digitais das mãos, portanto para realizar a correta identificação do usuário o algoritmo divide as posições do sensor em 16 grupos de dez impressões digitais. Cada conjunto de dez impressões digitais são associadas ao código de um “convidado”. Para o cadastro das impressões digitais foi utilizado o algoritmo presente no artigo “*ESP32 com RFID: Controle de Acesso*” de Koyanagi (2018).

Em seguida um algoritmo será executado em *loop*, ou seja, será realizado de maneira recorrente enquanto o dispositivo estiver operando. A Figura 27 apresenta o fluxograma do *loop*.

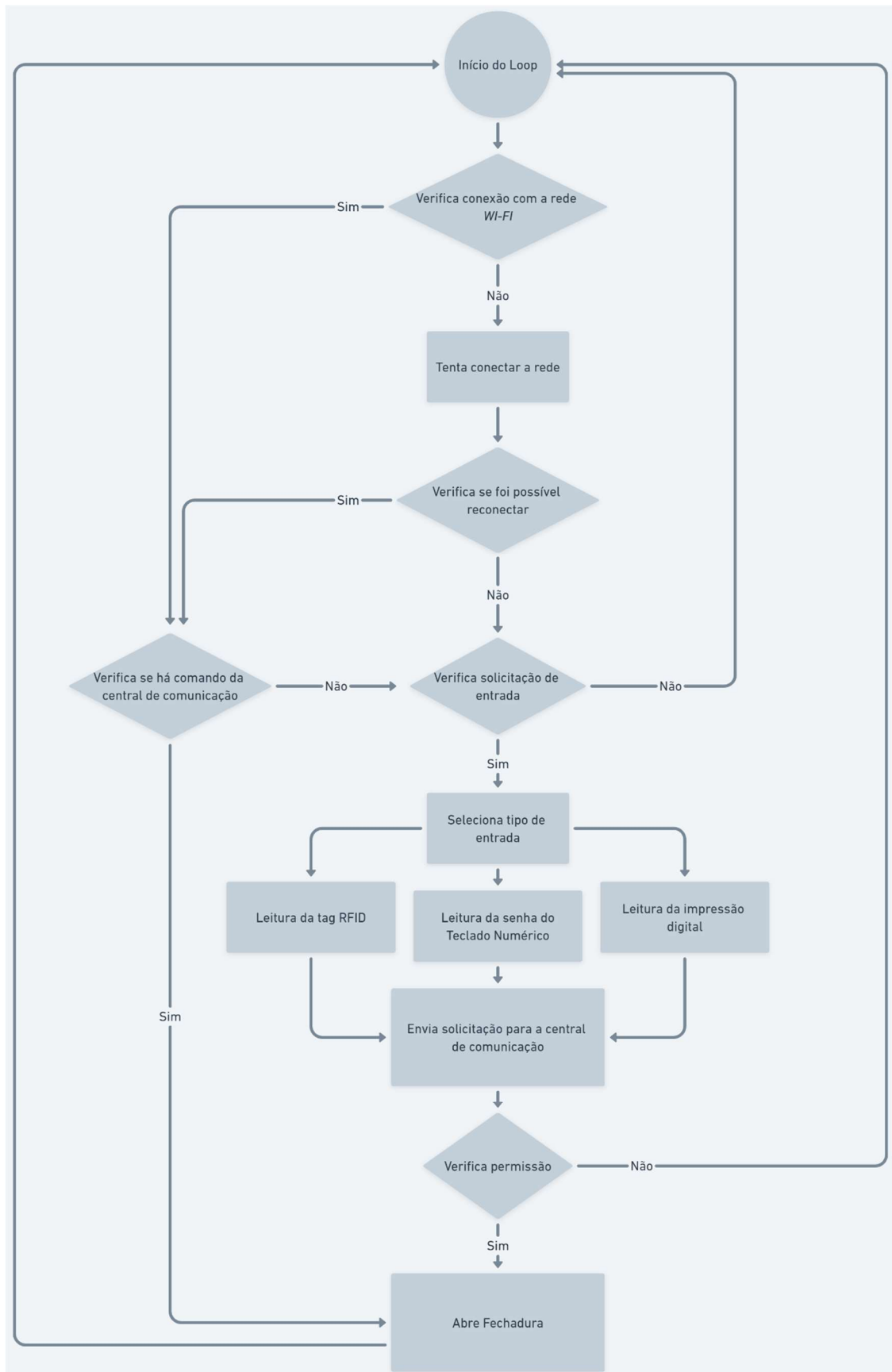
¹¹ Disponível em: github.com/johnrickman/LiquidCrystal_I2C.

Primeiramente o algoritmo verificará se o dispositivo possui conexão com a rede *Wi-Fi* previamente configurada. Caso não possua, é realizada uma tentativa de conexão à rede, em caso de falha, o programa começa a operar no modo offline. Caso a conexão seja bem-sucedida o programa consultará o banco de dados (*Firebase Realtime Database*) para verificar se foi emitido algum comando para abrir a porta. Caso a porta deva ser aberta, será executado o comando de desbloqueio da fechadura, caso não seja detectado nenhum comando o algoritmo segue para o próximo passo.

Em seguida, o algoritmo verifica se o “convidado” solicitou entrada, e então será oferecida ao usuário a opção de utilizar um dos meios de autenticação para solicitar o desbloqueio da fechadura. Ao escolher uma das opções, o código de acesso daquele “convidado”, é extraído. Após a extração do código daquele usuário, é enviada para a *API* uma solicitação de entrada, que contém o código de acesso e o código de identificação da fechadura. Se existir permissão que vincule aquele código de acesso do usuário aquela fechadura específica a solicitação de entrada do usuário é aceita.

Caso a solicitação do usuário tenha sido aceita, o estado da fechadura será alterado no banco de dados do *Firebase*, e após receber um retorno da *API* o algoritmo lerá o estado atual da fechadura e desbloqueará o acesso caso o usuário tenha tido permissão.

Figura 27 - Fluxograma do algoritmo da fechadura eletrônica.



Fonte: Acervo do autor (2023).

A implementação do algoritmo da fechadura foi desenvolvida na *ArduinoIDE*. Utilizando a linguagem de programação C++. Antes de começar o desenvolvimento foi necessário a formatação da memória *Flash* do ESP-WROOM-32 para o sistema de arquivos LittleFS, somente a partir dessa formatação é possível utilizar o sistema de arquivos que foi escolhido.

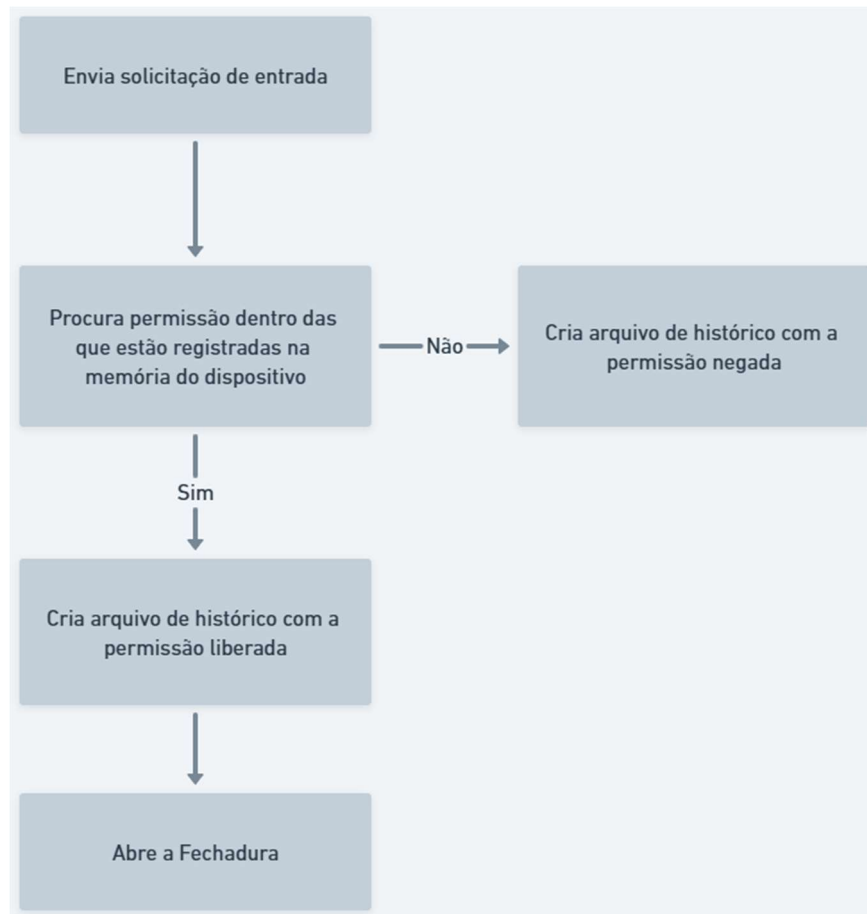
Para elucidar completamente o funcionamento do protótipo é necessário abordar o fluxo de validação de entrada quando a conexão do ESP-WROOM-32 com a internet é perdida.

Ao iniciar o programa, são criados dois diretórios dentro da memória flash do ESP-WROOM-32, uma pasta voltada para o armazenamento das permissões e outra voltada para armazenar os históricos de entrada que foram efetuados *offline*. Após a criação das pastas e a reconexão do microcontrolador com a internet, é realizada uma solicitação de recuperação da lista de permissões da fechadura para a *API*, dessa forma as informações são armazenadas em um arquivo de texto.

Em caso de perda de conexão com a internet, o fluxo ilustrado na Figura 27 permanece praticamente o mesmo, com exceção dos últimos passos após a extração da senha numérica, código RFID ou impressão digital. É montada uma solicitação contendo uma chave que combina o *Id da fechadura* com o código extraído e então em vez de enviar uma solicitação para a *API*, o programa envia essa solicitação para uma outra função, que será responsável por procurar a permissão daquele usuário dentro das permissões que foram cadastradas na memória.

Ao validar a permissão o programa armazena cada histórico de solicitação de entrada em um arquivo único que irá conter todos os dados necessários para depois ser enviado para a *API* que registrará os históricos de acesso que foram captados enquanto o dispositivo estava *offline*. Caso o “convidado” possua permissão de entrar ele efetua a ação de desbloquear a fechadura. Para melhor compreensão foi elaborado um diagrama que está presente na Figura 28.

Figura 28 - Diagrama de operação da fechadura no modo *offline*.



Fonte: Acervo do autor (2023).

Quando o programa está *offline*, em períodos de um minuto ele busca a reconexão com a rede, caso a conexão seja bem-sucedida ele verifica se existe algum arquivo de histórico. Havendo algum arquivo na pasta de históricos, cada arquivo tem seus dados extraídos, enviados para a *API*, e então deletados da pasta de histórico.

Para facilitar os testes de operação do dispositivo em modo *offline*, foi criada também uma função que altera a senha de conexão com a rede para o valor “0”. Dessa forma, quando o modo *offline* é ativado não há a necessidade de manipulação do código ou desligamento da rede para testar o modo de operação *offline* do dispositivo.

4.1.3 Fechadura eletrônica

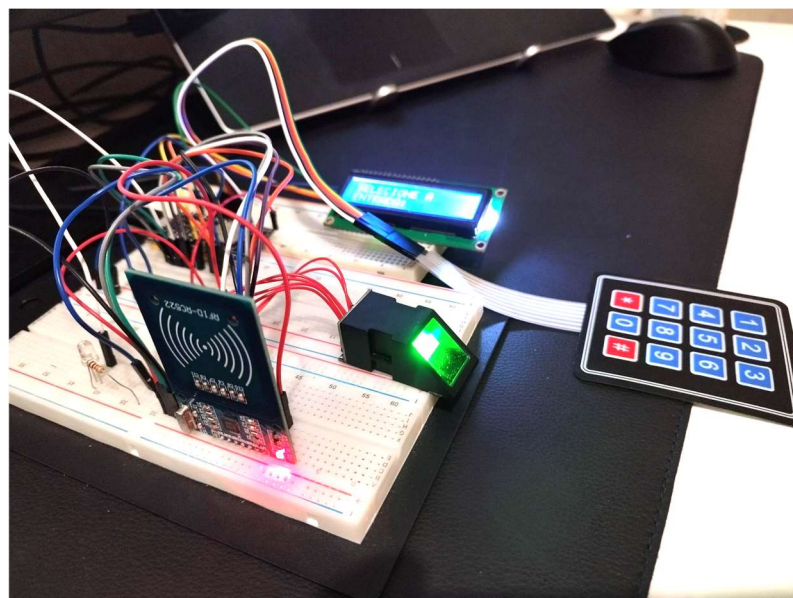
No apêndice A é apresentado o esquemático completo da fechadura eletrônica, onde podemos observar as ligações entre o ESP 32 e os demais componentes. Cada módulo possui a sua própria comunicação como foi apontado no tópico anterior.

O funcionamento da fechadura eletrônica proposta se dará de acordo com o algoritmo implementado. O dispositivo poderá ser operado remotamente pelo “operador” ou localmente pelo “convidado”.

O “operador”, poderá realizar o destravamento da fechadura por meio de um comando emitido pela central de controle e monitoramento. Já o “convidado” necessitará solicitar entrada e selecionar o método de identificação, que poderá ser por meio de um cartão *RFID*, de uma senha numérica, ou sua própria impressão digital. Para a senha numérica é necessário somente o cadastro através da própria plataforma do “operador”. Já o acesso através das tags *RFID* e da impressão digital será necessária a configuração prévia, no cartão *RFID* e no cadastro da impressão digital do usuário na fechadura que ele deseja utilizar este método de entrada.

Para a montagem do protótipo, foi utilizada uma *protoboard* e jumpers diversos para realizar as conexões com os diversos sensores. Para representar o sinal de controle da fechadura, foi utilizado um led. A Figura 29 mostra como ficou disposto o protótipo que foi utilizado para realizar os testes deste trabalho.

Figura 29 - Hardware do protótipo montado na *protoboard*.



Fonte: Acervo do autor (2023).

Vale ressaltar que o sistema composto pela fechadura e pelas aplicações *web* tem a finalidade de realizar somente o controle da entrada de “convidados” ao ambiente. O fechamento deverá ocorrer por meio de outro mecanismo atrelado à porta,

para realizar o fechamento automático da mesma. Um exemplo de mecanismo e a mola aérea para portas apresentada na Figura 30.

Figura 30 - Mola aérea para portas.



Fonte: Intelbras (2023).

Para a implantação do protótipo é importante que o controle de abertura seja associado a uma fechadura de sobrepor (Figura 31), que possui um mecanismo de abertura manual para os usuários que já estão dentro do ambiente.

Figura 31 - Fechadura de sobrepor.



Fonte: Gravia (2023).

Como alternativa para o método de saída do ambiente seria possível também utilizar uma botoeira conectada ao sistema, que seja acessível somente na parte interna do ambiente. Ao “convidado” acionar a botoeira o sistema desbloquearia a fechadura.

4.2 Desenvolvimento da aplicação web

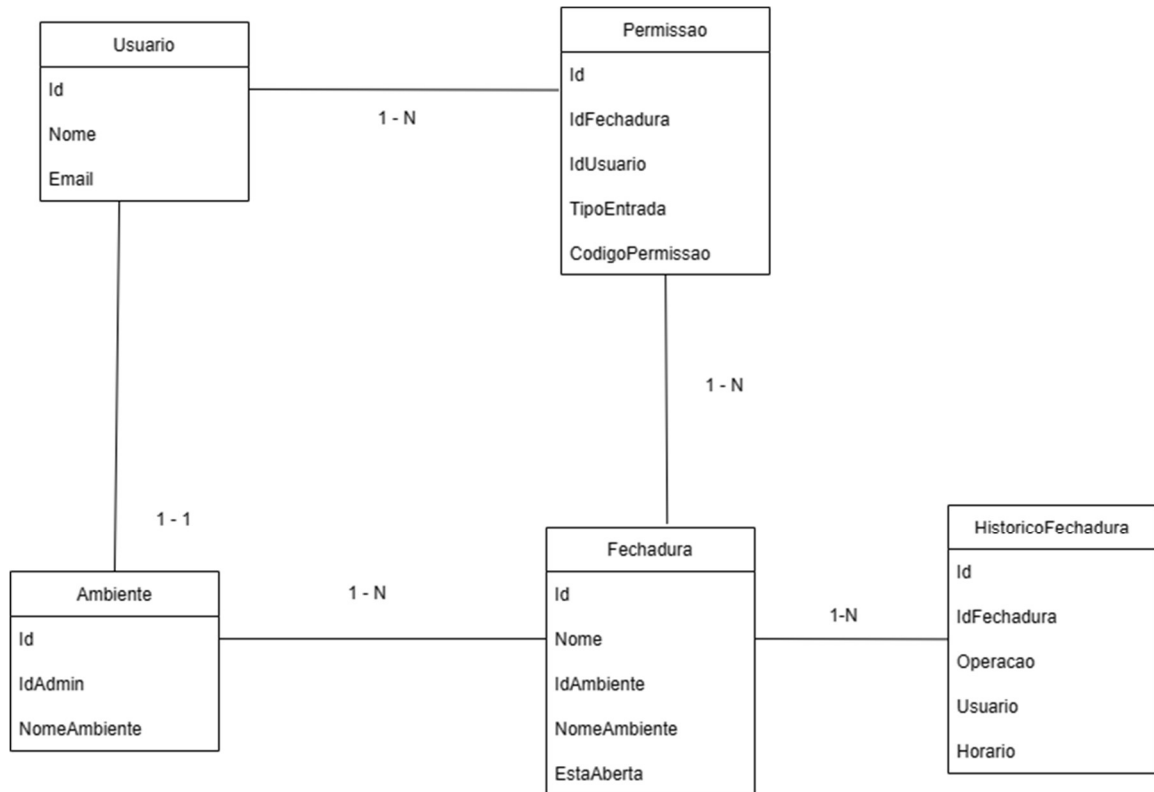
Como abordado no início deste capítulo, o desenvolvimento da aplicação *web* se dará em duas etapas: o desenvolvimento do *back-end*, uma *API* construída com o *Node.js* integrada ao banco de dados *Firebase*; o desenvolvimento do *front-end*, um *site* criado com a biblioteca *ReactJS*. O *site* se comunicará com a *API* através do protocolo *HTTP*.

Essa estrutura de aplicação tem diversas vantagens, como permitir a interação com o sistema através de outros *softwares*. Por conta da independência da central de comunicação, as interações com o sistema não estão limitadas ao *site*, podem também ser feitas através de aplicativos *mobile*, ou aplicativos *desktop* e diversas outras possibilidades. Outra vantagem é prevenir erros de acoplamento, quando uma aplicação está bem dividida em módulos, um *bug* na interface do *site* não necessariamente irá afetar o resto da aplicação.

4.2.1 Desenvolvimento da central de comunicação

Como mostrado na Figura 17, a central de comunicação pode ser dividida em duas partes, o banco de dados em tempo real *Firebase* e a *API*. Ambas desempenham papéis fundamentais neste sistema.

O **banco de dados** é responsável pelo armazenamento dos dados enviados a ele, e pela operação das fechaduras. O banco de dados em tempo real do *Firebase* tem um papel crucial, devido a sua velocidade em atualizar os dados, será através dele que será controlado a abertura da fechadura. Embora o banco de dados não seja *SQL*, foi elaborada uma estrutura base (Figura 32) para melhor organização dos registros no banco de dados *Firebase*.

Figura 32 - Estrutura de Banco de Dados.

Fonte: Acervo do autor (2023).

A estrutura do banco de dados possui cinco tabelas: “Usuario”, “Permissao”, “Ambiente”, “Fechadura” e “FechaduraHistorico”. Todas as tabelas e colunas foram nomeadas sem caracteres especiais, para evitar eventuais problemas na implementação das mesmas.

As responsabilidades de cada estrutura estão listadas abaixo:

- **Usuario:** será onde serão armazenados os dados para identificar o usuário “convidado”;
- **Ambiente:** será responsável por definir o *operador* de uma fechadura, e ela armazenará somente um identificador do administrador (*idAdmin*) daquele ambiente, além da identificação do próprio ambiente (*Id* e *Nome*);
- **Fechadura:** será a responsável por operar a fechadura eletrônica ao alternar sua propriedade “*EstaAberta*”, ela é recuperada pelo dispositivo para então realizar a abertura da porta;

- **HistoricoFechadura:** será responsável por armazenar cada tentativa de acesso realizada em uma fechadura. Dentro dela será armazenado os dados da tentativa de acesso, e o horário em que esta tentativa foi realizada.
- **Permissao:** será responsável pela configuração de permissões de acesso a uma fechadura, vinculando um usuário “convidado” a uma fechadura, através de um método de entrada específico.

Portanto as principais funcionalidades do **banco de dados** serão:

- Armazenar os dados de histórico da fechadura;
- Armazenar as permissões de acesso;
- Armazenar os dados dos usuários;
- Armazenar os dados da fechadura;
- Alternar o estado da fechadura (fechada ou aberta).

API é a responsável por coordenar todo o fluxo de dados enviados pelo “operador” e o “convidado” para operar de maneira correta a fechadura e assim realizar o objetivo do sistema, controlar e monitorar o acesso aos ambientes. As principais funcionalidades da **API** serão:

- Manipular as tabelas do banco de dados (Cadastrar, Ler, Editar e Deletar);
- Gerenciar as permissões de acesso das fechaduras;
- Receber solicitações de abertura do “convidado” e do “operador”;
- Cadastrar novos usuários no sistema;
- Retornar o Histórico de acesso da fechadura.

Para o auxílio no desenvolvimento será utilizada o *software* *Insomnia*¹², *software* que permite ao usuário realizar requisições *HTTP* para um determinado *URL* e visualizar os dados retornados. Esta ferramenta ajudará no desenvolvimento pois os testes da *API* podem ser realizados sem necessidade de interação com o *site* ou com a fechadura, o que possibilita o desenvolvimento paralelo das partes do sistema.

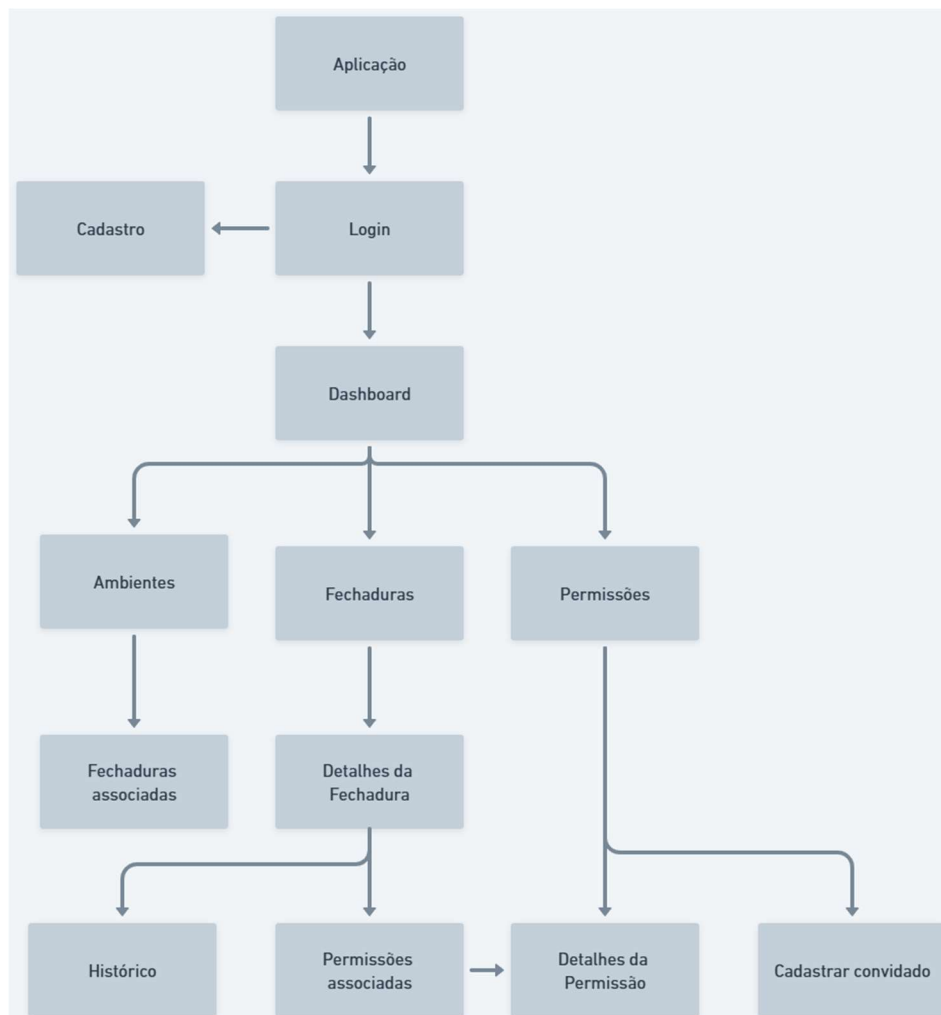
¹² Disponível em: insomnia.rest.

4.2.2 Desenvolvimento da central de monitoramento e controle

A central de monitoramento e controle é um *site*, uma plataforma *web* construída com a biblioteca *React.JS*, que tem como objetivo ser o meio pelo qual o “operador” irá gerenciar os acessos de uma fechadura eletrônica.

O desenvolvimento deste site se deu em três etapas. Primeiro é necessário elaborar a estrutura da aplicação, quais telas o “operador” terá acesso e quais funcionalidades serão desenvolvidas pela plataforma. Um diagrama da estrutura de navegação da plataforma foi construído e pode ser observado na Figura 33.

Figura 33 - Estrutura do site.



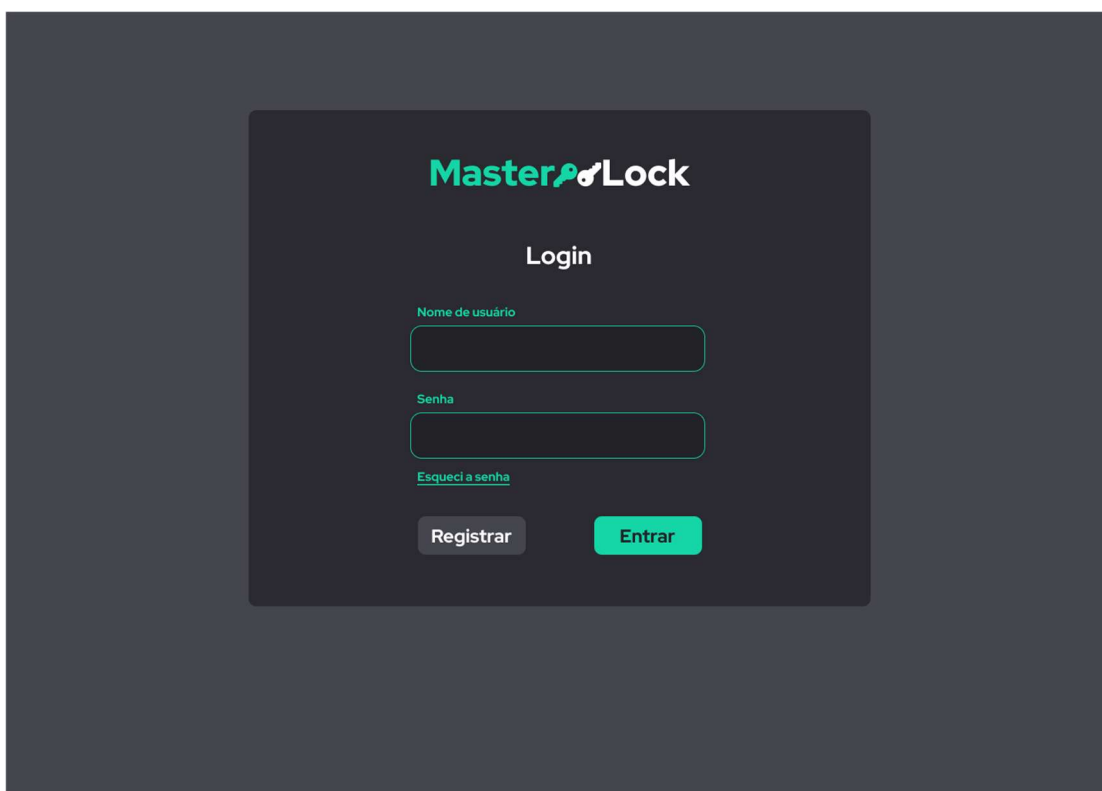
Fonte: Acervo do autor (2023).

A segunda etapa é construir o *design* das telas. Este processo é importante para aprimorar a experiência do usuário dentro da plataforma, facilitando assim o uso

da mesma. Nesta etapa temos também o planejamento dos componentes que serão utilizados dentro de nossa aplicação *React*, que por conta do princípio da componentização nos permite reutilizar diversos trechos de código.

Para o projeto das telas do *site* e da identidade visual do projeto foi utilizado o *site* Figma¹³. Para a melhor visualização da proposta da plataforma, foram elaborados os designs de quatro telas. Nesta etapa focou-se na criação da identidade visual do sistema e na usabilidade do mesmo. Foram construídas as telas de *Login* (Figura 34), onde o “operador” poderá acessar a plataforma, a tela de controle da fechadura eletrônica (figura 35), a tela onde será observado o histórico de operações realizadas na fechadura (figura 36) e a tela onde teremos o controle de permissões dos usuários cadastrados (figura 37). A aplicação *web* foi nomeada como *Master Lock*, que em tradução livre significa fechadura mestre.

Figura 34 - Esboço da tela de *Login*.



Fonte: Acervo do autor (2022).

¹³ Disponível em: figma.com.

Figura 35 - Esboço da tela de controle da fechadura.

Fonte: Acervo do autor (2022).

Figura 36 - Tela de histórico de acessos.

Fonte: Acervo do autor (2022).

Figura 37 - Tela de controle de permissões dos usuários.



Fonte: Acervo do autor (2022).

Para a elaboração dos diagramas estruturais, foram usados os *sites*: *Whimsical*¹⁴ e *Draw.io*¹⁵. No desenvolvimento da plataforma *web* foi utilizado o editor de texto desenvolvido pela *Microsoft*, *Visual Studio Code*, que permite um desenvolvimento ágil e dinâmico através de diversas extensões disponíveis no *software*.

¹⁴ Disponível em: whimsical.com;

¹⁵ Disponível em: drawio.com.

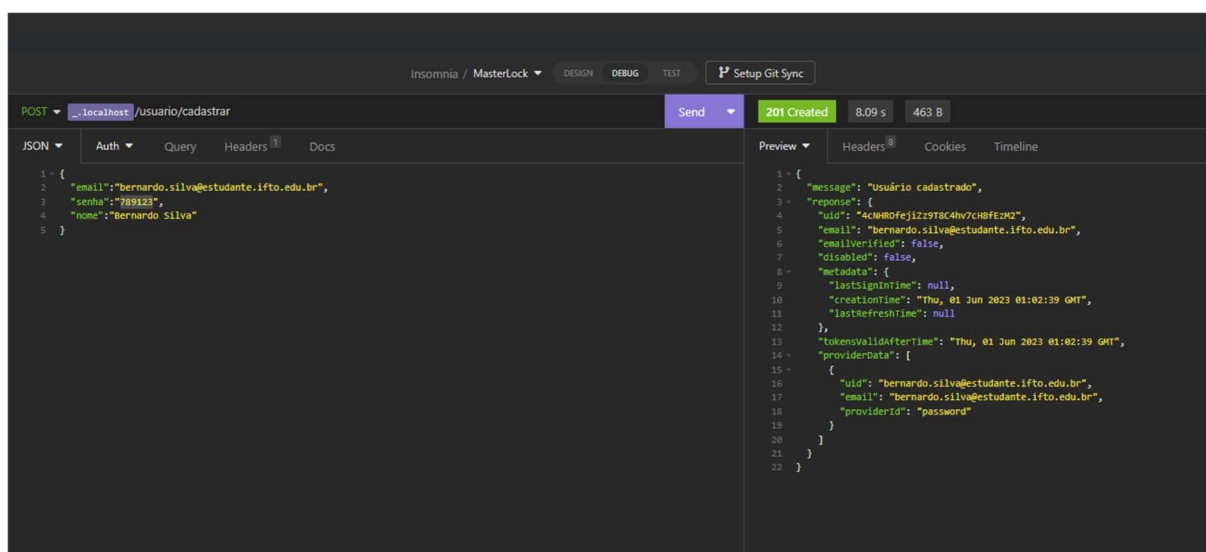
5 RESULTADOS E DISCUSSÕES

Este capítulo é destinado a explorar e demonstrar os resultados obtidos a partir do protótipo da fechadura eletrônica, e do sistema *web* que foi construído no presente trabalho.

5.1 Cadastro e configuração da fechadura e plataforma

Para que o controle da fechadura seja efetuado corretamente pelo sistema é necessário que sejam configurados previamente diversos parâmetros, sendo o primeiro deles o cadastro do usuário “operador”. A *API* desenvolvida foi utilizada para tal tarefa, e o *software insomnia* para enviar as solicitações *HTTP* (*Hypertext Transfer Protocol*). Neste passo o “operador” é cadastrado tanto no banco de dados *firebase* (*Firebase Realtime Database*) quanto no sistema de autenticação (*Firebase Authentication*). O resultado da solicitação de cadastro é mostrado na Figura 38.

Figura 38 - Tela de requisição de cadastro de “operador” via *insomnia*.

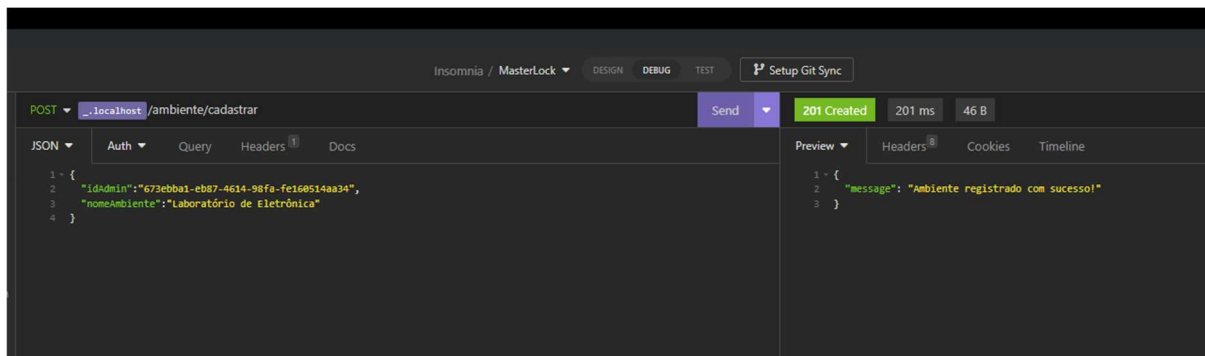


Fonte: Acervo do autor (2023).

Após o cadastro do usuário, é gerado automaticamente pela *API* um código de identificação do mesmo, sendo necessário consultá-lo para que se possa criar um ambiente do qual o usuário é o administrador. Para criação de um novo ambiente é feita uma requisição *HTTP* para a aplicação *back-end*. Como demonstrado na Figura

39, esta solicitação contém as informações do ambiente ao qual a fechadura será vinculada, como o nome do ambiente e o código identificador do “operador” (*idAdmin*).

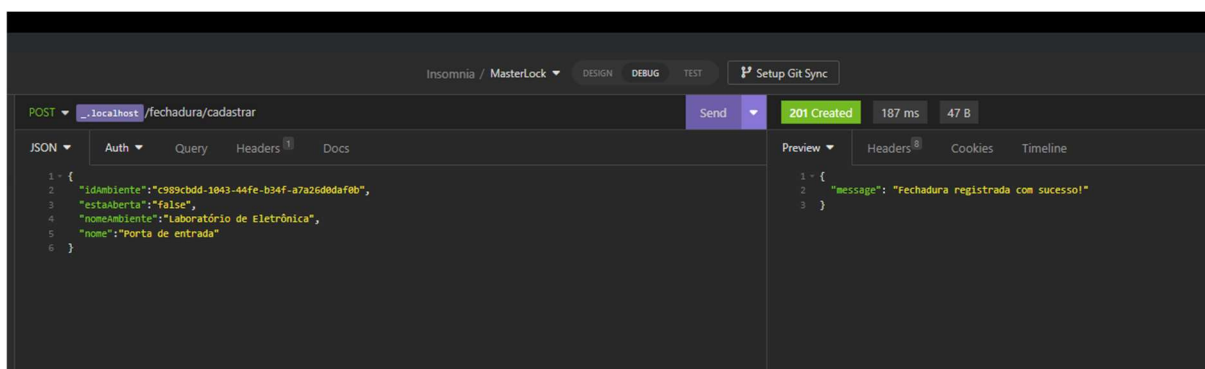
Figura 39 - Tela de requisição de cadastro de ambiente via *insomnia*.



Fonte: Acervo do autor (2023).

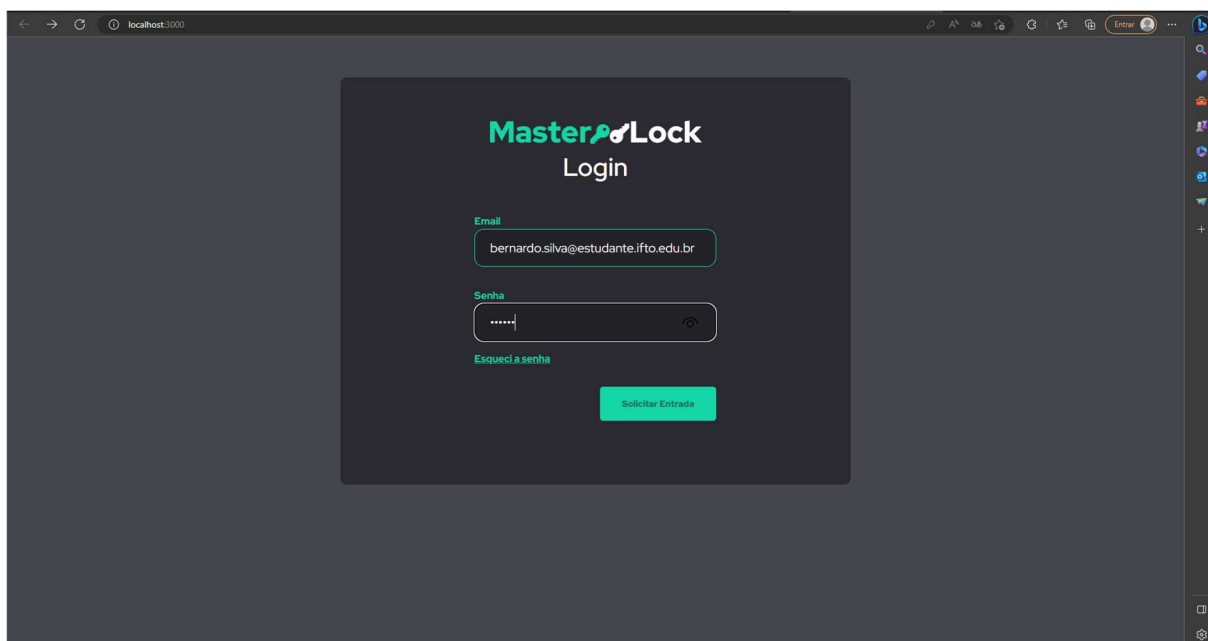
Com o registro do ambiente, podemos então cadastrar a primeira fechadura, vinculada a este ambiente. Um exemplo deste cadastro via *API*, é mostrado na Figura 40.

Figura 40 - Requisição de cadastro de fechadura pelo *insomnia*.



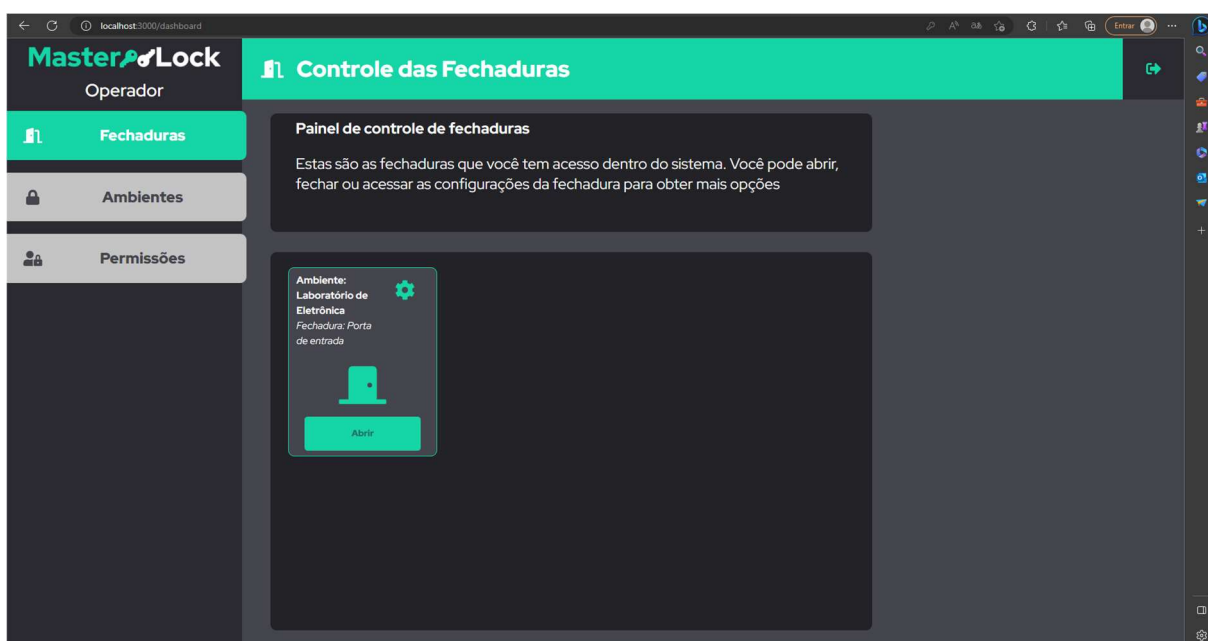
Fonte: Acervo do autor (2023).

A partir deste ponto, o “operador” pode então acessar a plataforma para visualizar e configurar os detalhes de permissão da fechadura. Ao acessar a plataforma *MasterLock* ele precisará primeiro realizar a autenticação (Figura 41) inserindo o e-mail e senha que foi cadastrado via *API* no primeiro passo.

Figura 41 - Tela de *Login* da aplicação.

Fonte: Acervo do autor (2023).

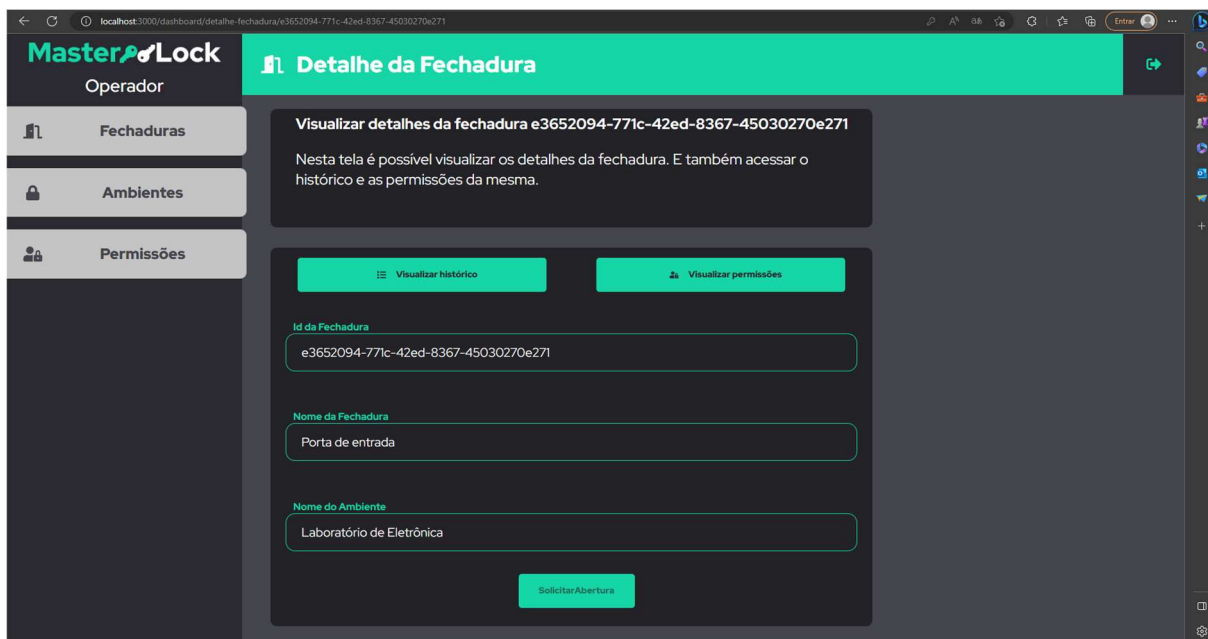
Após realizar a autenticação na plataforma o “operador” é direcionado à tela onde tem-se acesso a todas as fechaduras vinculadas aos ambientes que foram previamente associados. A Figura 42 mostra qual foi o resultado da implementação desta tela.

Figura 42 - Tela de controle de fechaduras da aplicação.

Fonte: Acervo do autor (2023).

Na tela mostrada na Figura 42, o “operador” é capaz de acessar os dados das fechaduras e ao clicar no ícone da engrenagem no canto superior direito o usuário tem acesso a tela mostrada na Figura 43. Neste formulário o usuário pode ter acesso ao código identificador da fechadura, que será utilizado para identificá-la, e também aos históricos de acesso e as permissões vinculadas àquela fechadura.

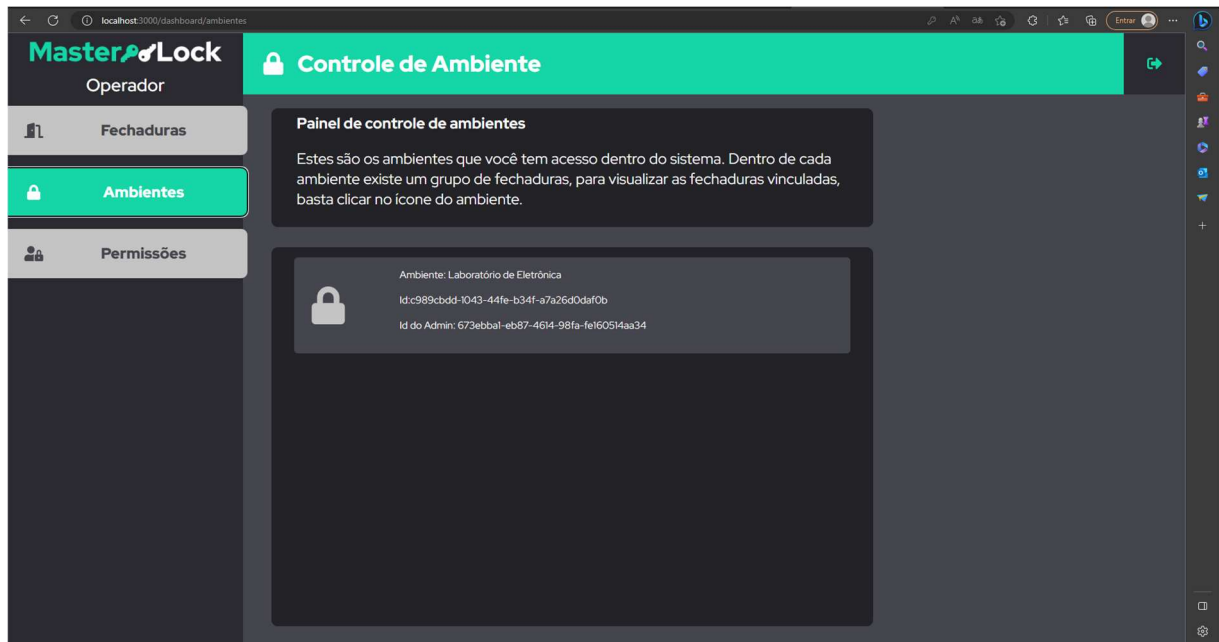
Figura 43 - Tela de detalhe de fechaduras da aplicação.



Fonte: Acervo do autor (2023).

A tela de ambientes, mostrada na Figura 44, permite que o operador acesse a lista de ambientes ao qual está vinculado, e clicando no item do ambiente será direcionado para uma tela que lista todas as fechaduras vinculadas àquele ambiente.

Figura 44 - Tela de controle de ambiente da aplicação.



Fonte: Acervo do autor (2023).

O terceiro menu disponível é onde ocorre o gerenciamento das permissões de acesso das fechaduras a tela desta funcionalidade é apresentada na Figura 45, que mostra o estado inicial sem nenhuma permissão configurada. Nela é possível realizar o cadastro de um novo convidado, pesquisar os convidados cadastrados e criar permissões.

Figura 45 - Tela de gerenciamento de permissões da aplicação.



Fonte: Acervo do autor (2023).

Para criar permissões o usuário precisa recuperar o identificador do convidado, isto é possível através da tela de pesquisa de usuário. Para isso foi utilizado a tela demonstrada na Figura 46, onde é vinculado um código de usuário a uma fechadura já cadastrada. E então, juntando estas duas informações é criada uma permissão no banco de dados.

Figura 46 - Tela de detalhe da permissão da aplicação.

MasterLock
Operador

Detalhe da Permissão

Criar nova permissão
Nesta tela é possível criar uma nova permissão associando um convidado a fechadura

Id da fechadura
e3652094-771c-42ed-8367-45030270e271

Id do usuário
dea6f289-f528-41fb-ba06-b3a23f08a749

Tipo de entrada
Cartão RFID

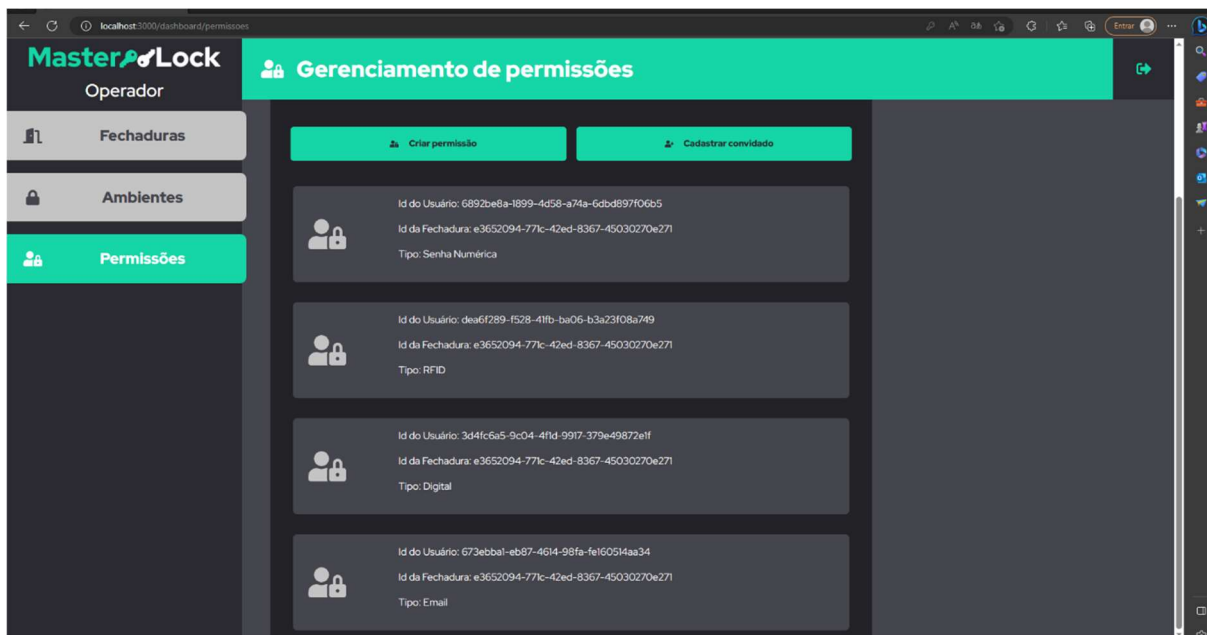
Código de Permissão
ABC5892

Criar permissão

Fonte: Acervo do autor (2023).

Ao finalizar o cadastro de todas as permissões, é possível visualizar os cadastros na página de permissões, como mostrado na Figura 47 e então utilizar o protótipo da fechadura eletrônica para acessar o ambiente configurado.

Figura 47 - Tela de gerenciamento de permissões da aplicação.



Fonte: Acervo do autor (2023).

5.2 Testes online e offline

Com todas as configurações de ambiente e de fechaduras que foram ajustadas, foi então testado o protótipo de fechadura eletrônica. O algoritmo do *hardware* foi previamente configurado para o armazenamento do código identificador gerado pela *API*, os dados de conexão de rede e de conexão com o *Firebase*.

Após realizar as devidas configurações, o código foi executado na fechadura. Para essa atividade, o algoritmo desenvolvido foi compilado e transferido para o ESP-WROOM-32 por meio de um cabo USB. Em seguida, foram realizados testes de operação do hardware em dois cenários distintos: um com conexão à internet e outro com a interrupção dessa conexão.

5.2.1 Testes *online*

Os testes realizados na fechadura foram consultados através do monitor serial, utilizando a *Arduino IDE*, e então extraídas as figuras deste tópico. Foram realizados testes individuais em cada método de entrada, sendo eles o *RFID*, a senha numérica e a impressão digital. E em cada teste foi inserido uma credencial válida, ou seja,

cadastrada previamente nas permissões da plataforma e uma credencial inválida, que não possui permissão cadastrada.

Ao ligar o dispositivo e ocorrer o estabelecimento da conexão com a internet é apresentada no *display LCD* uma mensagem de boas-vindas, como indicado na Figura 48. E logo em seguida é exibido um menu solicitando que o “convidado” selecione o método de entrada. Enquanto isso, o teclado matricial capta qualquer comando que o usuário digitar para realizar os procedimentos de cada opção escolhida.

Figura 48 - *Display LCD* exibindo as três fases do menu.

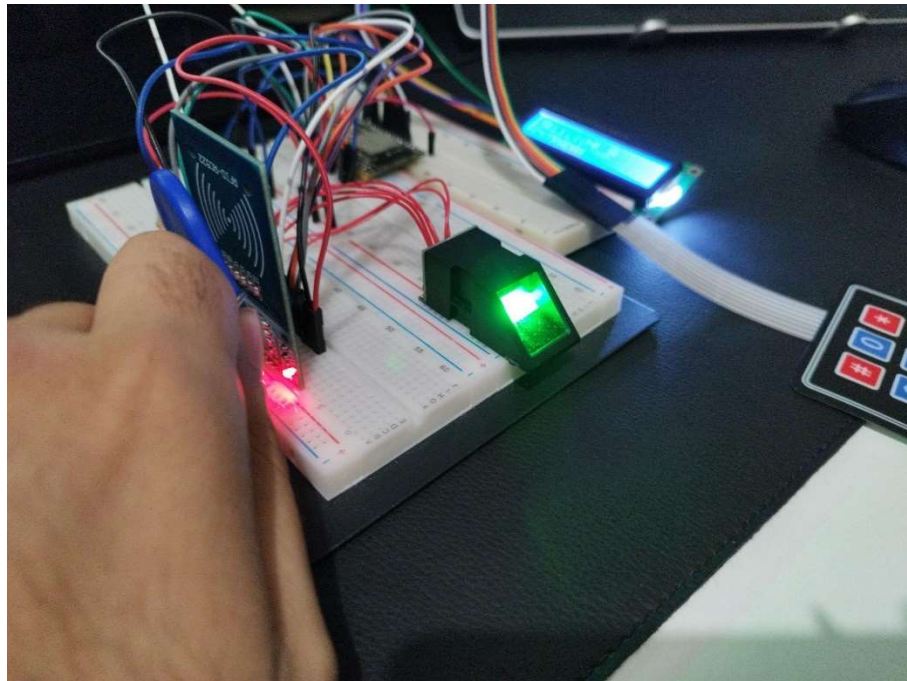


Fonte: Acervo do autor (2023).

5.2.1.1 Testes *online* da entrada por *tag RFID*

Para a realização destes testes, foi necessário a configuração prévia da *tag RFID*, inserindo nela um código de verificação. Ao pressionar a tecla número “1” no teclado matricial é então selecionado o método de entrada por *RFID* e o *display LCD* exibe uma mensagem para o “convidado” aproximar a tag ou cartão do dispositivo de leitura. Ao aproximar uma *tag* (Figura 49) com um código de verificação válido o sistema verificou se o usuário possuía uma permissão com aquele código e após isto sinalizou a abertura da fechadura, enviando o sinal para o pino do *LED* e informando através do *display LCD* que a fechadura estaria sendo bloqueada novamente em dez segundos.

Figura 49 - Teste do método de entrada *RFID*.



Fonte: Acervo do autor (2023).

Após o teste da aproximação de uma *tag* válida, testou-se também uma *tag* inválida com o mesmo processo realizado anteriormente. O comportamento da fechadura quando o usuário aproximou uma *tag* inválida foi informá-lo que o acesso tinha sido negado.

Como pode ser observado na Figura 50, o monitor *serial* registrou todo o processo de solicitação de entrada, registrando a captação do código do cartão e o resultado da solicitação de entrada.

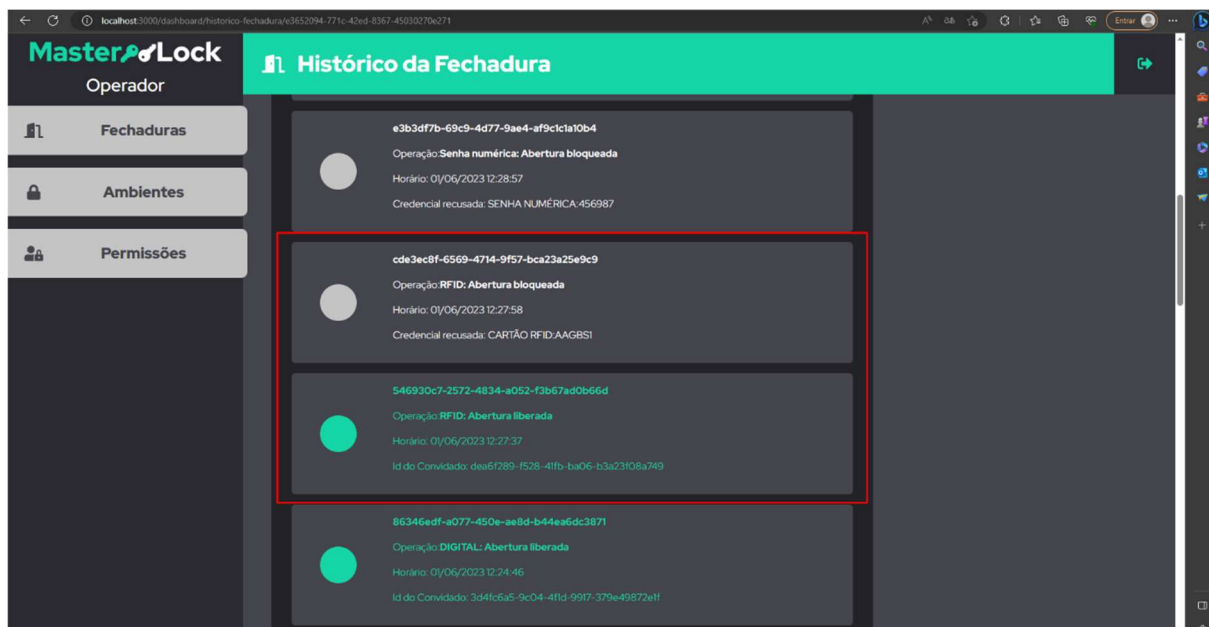
Figura 50 - *Display serial* no processo de leitura da *tag RFID*.

```
infoCartao :  
ABC5892  
Acesso liberado!  
Abrindo fechadura...  
Fechadura trancada!  
  
infoCartao :  
AAGBS1  
Acesso negado!
```

Fonte: Acervo do autor (2023).

Com as tentativas de acesso realizadas foi observado que os registros dos históricos foram feitos corretamente como mostrado na Figura 51.

Figura 51 - Tela de histórico de fechadura.



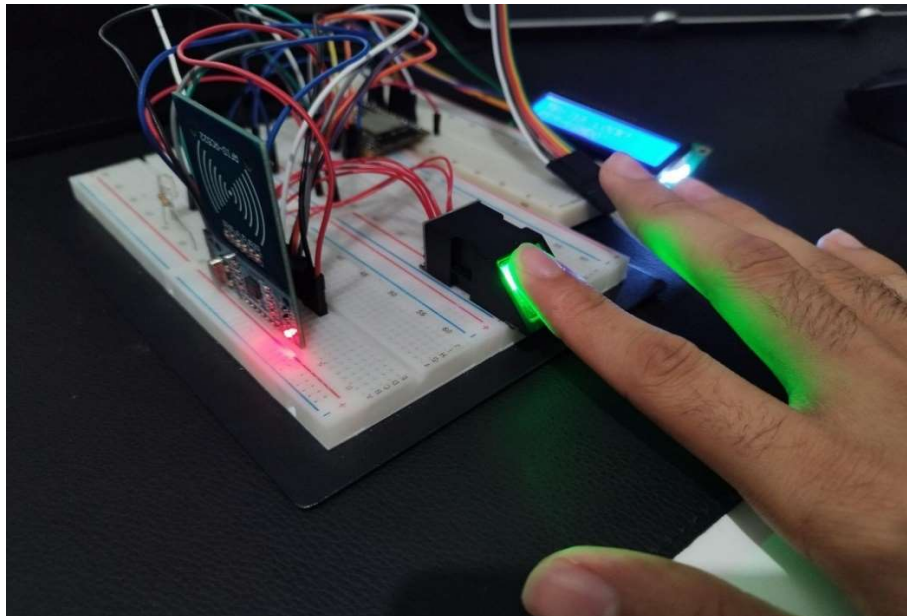
Fonte: Acervo do autor (2023).

5.2.1.2 Testes *online* da entrada por impressão digital

Após configurar previamente o dispositivo de leitura de impressão digital com algumas digitais cadastradas, foram realizados testes em três cenários distintos. No primeiro cenário, o usuário possuía uma digital cadastrada no dispositivo, porém não tinha permissão de entrada. No segundo cenário, o usuário tinha uma digital cadastrada e também possuía permissão de entrada. Já no último cenário, o usuário não tinha uma digital cadastrada no dispositivo.

A realização do teste foi simples, bastava selecionar o método de entrada, pressionando a tecla número “3” no teclado matricial e aguardar o *display LCD* sinalizar que o “convidado” aproximasse a digital do sensor, como mostrado na Figura 52, e então, após a aproximação foi realizada a validação.

Figura 52 - Teste do método de entrada por impressão digital.



Fonte: Acervo do autor (2023).

Dentro destes três cenários o sistema apresentou as validações corretas, rejeitando o acesso do “convidado” que tinha digital cadastrada, mas não tinha permissão e do “convidado” não tinha digital cadastrada. No caso do usuário que possuía a autorização e a digital cadastrada, a entrada foi permitida a sinalização de operação da fechadura feita corretamente. Na Figura 53, podemos ver o que foi exibido no monitor serial durante a validação da digital que possuía cadastro e permissão e da digital que não tinha sido cadastrada.

Figura 53 - *Display serial* no processo de leitura da impressão digital.

```
Digital encontrada com confiança de 312 na posição 11  
Código da Digital: MATHEUS  
Acesso liberado!  
Abrindo fechadura...  
Fechadura trancada!  
Digital não encontrada  
Erro ao validar digital
```

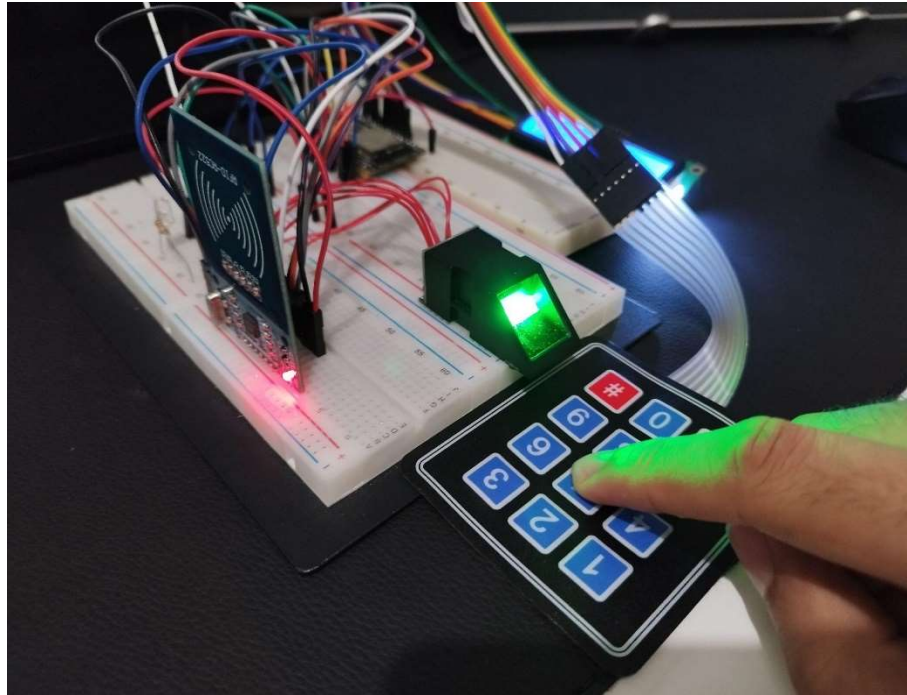
Fonte: Acervo do autor (2023).

5.2.1.3 Testes *online* da entrada por senha numérica

Para realizar os testes deste método de entrada é necessário somente que o usuário possua uma senha numérica de seis dígitos vinculada pela plataforma *web*

pelo “operador”. Após este cadastro o “convidado” pode então pressionar a tecla número “2” no teclado matricial, e então o dispositivo aguardará a inserção de seis dígitos pelo teclado matricial como mostrado na Figura 54.

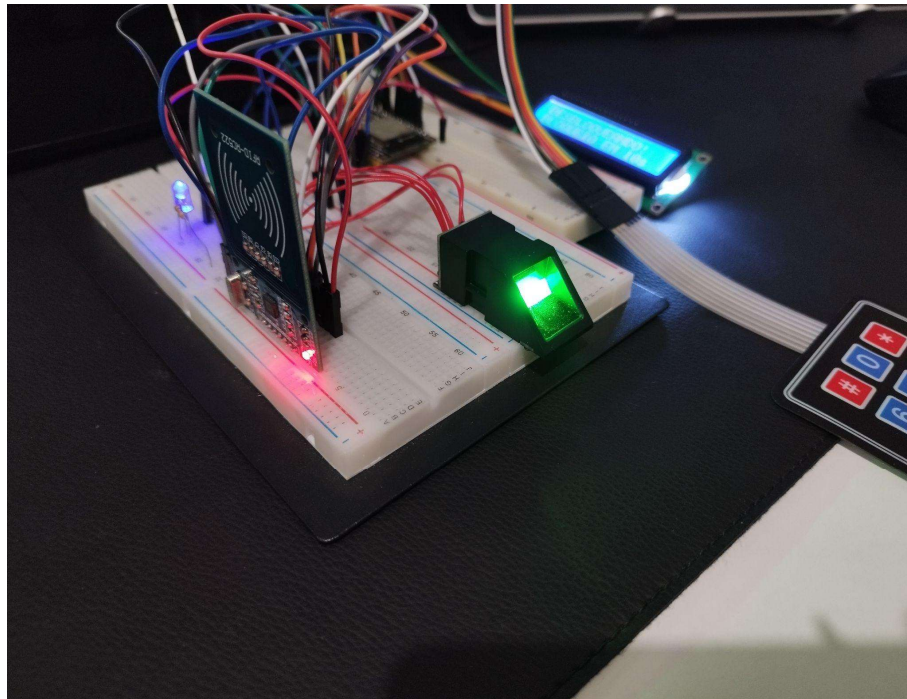
Figura 54 - Teste do método de entrada por senha numérica.



Fonte: Acervo do autor (2023).

Após a digitação de uma senha válida, o dispositivo operou corretamente enviando o sinal de abertura para o LED indicador como indicado na Figura 55. Ao realizar o teste com uma senha que não estivesse cadastrada no sistema de permissões, a entrada foi bloqueada.

Figura 55 - Teste do LED indicador de abertura.



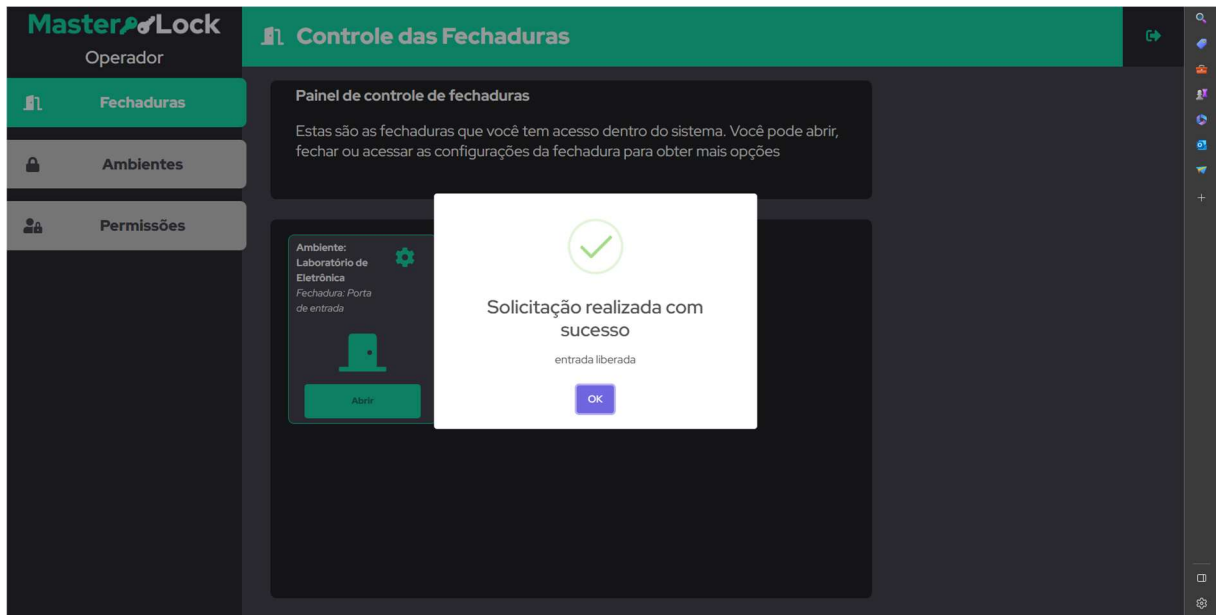
Fonte: Acervo do autor (2023).

5.2.1.4 Testes *online* da entrada via plataforma *Web*

O último teste *online* que pode ser realizado é o teste de acesso via plataforma web. Este teste só foi possível enquanto o dispositivo está conectado com a internet, pois ele precisava ler a propriedade que indica abertura ou bloqueio de acesso do banco de dados *Firebase*. Para isso é necessário que haja conexão disponível para o dispositivo. Foi necessário também criar uma permissão para o usuário “operador”, vinculando seu e-mail a fechadura, para que esta funcionalidade fosse habilitada.

Para realizar este teste foi necessário somente acessar a plataforma com as credenciais do usuário operador (Figura 42), clicar no botão “Abrir” da fechadura desejada e instantaneamente a solicitação foi enviada e o LED indicador foi operado pelo ESP-WROOM-32. Além disso, a plataforma também apresenta uma mensagem indicando que a porta foi operada com sucesso como mostrado na Figura 56.

Figura 56 - Tela da aplicação após realizar uma solicitação de entrada.



Fonte: Acervo do autor (2023).

Entre os dez segundos que a fechadura está aberta, é possível atualizar a página e visualizar que a operação foi realizada com sucesso pela própria plataforma, que indica no ícone que a fechadura está aberta. Como indicado na Figura 57.

Figura 57 - Tela de controle de fechadura indicando a abertura do dispositivo.

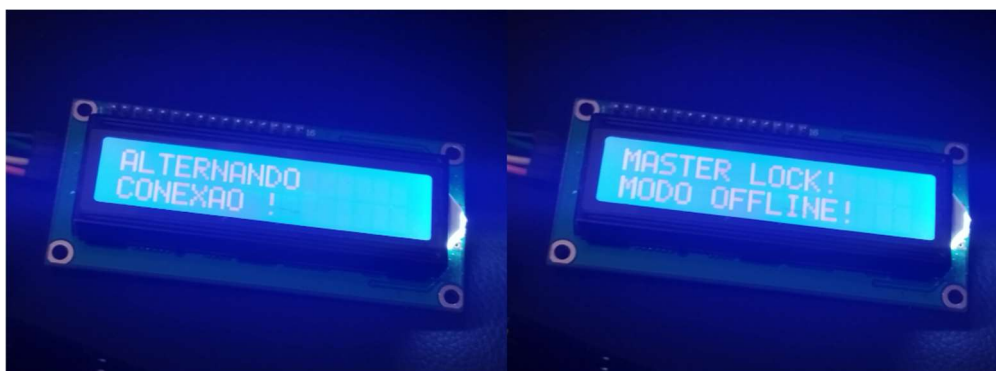


Fonte: Acervo do autor (2023).

5.2.2 Testes *offline*

Para a realização dos testes *offline* foi utilizada a funcionalidade desenvolvida de alterar a senha de conexão com a rede do dispositivo que pode ser acionada ao pressionar a tecla “*”. O *display LCD* então apresenta uma mensagem informando que a conexão está sendo alternada, como mostrado na Figura 58. Após aguardar um tempo, o *display* indica que foi ativado o modo *offline* em vez de apresentar a mensagem de boas-vindas ao usuário “convidado”, e então foi iniciado os testes do modo *offline* do dispositivo.

Figura 58 - *Display LCD* após a entrada no modo *offline*.



Fonte: Acervo do autor (2023).

Após iniciar o modo *offline* foram testados então os três métodos de entrada do "convidado" e alternada novamente a conexão para que fosse possível testar também como o dispositivo funcionaria após se reconectar com a rede.

5.2.2.1 Testes *offline* da entrada por *tag RFID*

O processo de teste dos métodos de entrada *offline* foi semelhante ao teste do método *online*. Após selecionar o método de entrada foi aproximada a tag ou cartão *RFID* do leitor, e então o resultado foi indicado pelo *display LCD*. A maior diferença pode ser vista no monitor *serial* (Figura 59) durante este processo, pois quando foi solicitada a entrada por este método, o monitor indicou que o arquivo de permissões havia sido consultado para que houvesse a autenticação do usuário. O processo transcorreu sem problemas, com o usuário autorizado sendo capaz de acessar tanto no modo online quanto no modo offline, enquanto o usuário não autorizado teve corretamente seu acesso bloqueado.

Figura 59 - Display serial no modo *offline* de leitura do cartão *RFID*.

```
infoCartao :
AAGBS1
e3652094-771c-42ed-8367-45030270e271|AAGBS1
LENDO ARQUIVO: /permissoes/permissoes.txt
- LENDO O ARQUIVO:
ESCREVENDO NO ARQUIVO: /historico/1.txt
- A ESCRITA FOI UM SUCESSO
LENDO ARQUIVO: /historico/1.txt
- LEU DO ARQUIVO:
{"permisaoOffline":"e3652094-771c-42ed-8367-45030270e271|AAGBS1 ", "operacao":"Abertura bloqueada", "horario":"01/06/2023 00:31:54"}

infoCartao :
ABC5892
e3652094-771c-42ed-8367-45030270e271|ABC5892
LENDO ARQUIVO: /permissoes/permissoes.txt
- LENDO O ARQUIVO:
ACHOU A PALAVRA:
e3652094-771c-42ed-8367-45030270e271|ABC5892
ESCREVENDO NO ARQUIVO: /historico/2.txt
- A ESCRITA FOI UM SUCESSO
LENDO ARQUIVO: /historico/2.txt
- LEU DO ARQUIVO:
{"permisaoOffline":"e3652094-771c-42ed-8367-45030270e271|ABC5892", "operacao":"Abertura liberada", "horario":"01/06/2023 00:32:13"}
```

Fonte: Acervo do autor (2023).

Nota-se também, pela leitura do monitor serial que os históricos foram registrados corretamente, cada um em seu determinado arquivo e todos na pasta de históricos, que é criada na primeira inicialização do dispositivo.

5.2.2.2 Testes *offline* da entrada por impressão digital

Para este método de entrada foram testados novamente os três cenários possíveis que foram indicados no tópico 5.2.1.2. A Figura 60 mostra o monitor *serial* destes testes indica que o dispositivo teve o comportamento adequado para as três situações, validando estes casos corretamente como registrando o histórico *offline*.

Figura 60 - Display serial no modo *offline* de leitura da impressão digital.

```
Digital encontrada com confiança de 263 na posição 11
Código da Digital: MATHEUS
e3652094-771c-42ed-8367-45030270e271|MATHEUS
LENDO ARQUIVO: /permissoes/permissoes.txt
- LENDO O ARQUIVO:
ACHOU A PALAVRA:
e3652094-771c-42ed-8367-45030270e271|MATHEUS
ESCREVENDO NO ARQUIVO: /historico/5.txt
- A ESCRITA FOI UM SUCESSO
LENDO ARQUIVO: /historico/5.txt
- LEU DO ARQUIVO:
{"permisaoOffline":"e3652094-771c-42ed-8367-45030270e271|MATHEUS", "operacao":"Abertura liberada", "horario":"01/06/2023 00:33:04"}

Digital encontrada com confiança de 335 na posição 1
Código da Digital: BERNARDO
e3652094-771c-42ed-8367-45030270e271|BERNARDO
LENDO ARQUIVO: /permissoes/permissoes.txt
- LENDO O ARQUIVO:
ESCREVENDO NO ARQUIVO: /historico/6.txt
- A ESCRITA FOI UM SUCESSO
LENDO ARQUIVO: /historico/6.txt
- LEU DO ARQUIVO:
{"permisaoOffline":"e3652094-771c-42ed-8367-45030270e271|BERNARDO", "operacao":"Abertura bloqueada", "horario":"01/06/2023 00:33:33"}

Digital não encontrada
ESCREVENDO NO ARQUIVO: /historico/7.txt
- A ESCRITA FOI UM SUCESSO
LENDO ARQUIVO: /historico/7.txt
- LEU DO ARQUIVO:
{"permisaoOffline":"e3652094-771c-42ed-8367-45030270e271|DIGITAL NAO CADASTRADA", "operacao":"Abertura bloqueada", "horario":"01/06/2023 00:33:44"}
Erro ao validar digital
```

Fonte: Acervo do autor (2023).

5.2.2.3 Testes *offline* da entrada por senha numérica

Semelhante ao teste *online*, após a seleção do método pela senha numérica, o dispositivo aguarda a entrada dos dados e indica no monitor serial cada tecla que o usuário pressiona no teclado. O monitor serial mostrou (Figura 61) novamente o processo de leitura do arquivo de permissões para que o usuário fosse validado corretamente e todo o processo de autenticação teve o comportamento esperado.

No teste indicado na Figura 61, foi testado primeiramente uma senha inválida e em seguida uma senha válida. Nota-se pela leitura do arquivo de histórico que é realizada no final de cada processo de autenticação que a primeira tentativa possuía uma senha que não estava cadastrada e a abertura foi bloqueada. Já na segunda tentativa o usuário tinha uma senha cadastrada e a abertura foi liberada.

Figura 61 - *Display serial* no processo *offline* de leitura da senha numérica.

```

7
8
0
5
4
6
e3652094-771c-42ed-8367-45030270e271|780546
LEND0 ARQUIVO: /permissoes/permissoes.txt
- LEND0 O ARQUIVO:
ESCREVENDO NO ARQUIVO: /historico/3.txt
- A ESCRITA FOI UM SUCESSO
LEND0 ARQUIVO: /historico/3.txt
- LEU DO ARQUIVO:
{"permissaoOffline":"e3652094-771c-42ed-8367-45030270e271|780546","operacao":"Abertura bloqueada","horario":"01/06/2023 00:32:33"}
4
0
0
2
8
9
e3652094-771c-42ed-8367-45030270e271|400289
LEND0 ARQUIVO: /permissoes/permissoes.txt
- LEND0 O ARQUIVO:
ACHOU A PALAVRA:
e3652094-771c-42ed-8367-45030270e271|400289
ESCREVENDO NO ARQUIVO: /historico/4.txt
- A ESCRITA FOI UM SUCESSO
LEND0 ARQUIVO: /historico/4.txt
- LEU DO ARQUIVO:
{"permissaoOffline":"e3652094-771c-42ed-8367-45030270e271|400289","operacao":"Abertura liberada","horario":"01/06/2023 00:32:46"}

```

Fonte: Acervo do autor (2023).

5.2.2.3 Testes de reconexão com a internet

Para realizar este teste foi utilizada novamente a funcionalidade de alterar a conexão do dispositivo ao pressionar a tecla “*”. Foi verificado então, como se pode notar na Figura 62, que todo o processo de reconexão foi realizado corretamente e

além disso, todos os históricos que tinham sido registrados *offline* foram enviados para registro via *API* e então deletados da pasta de histórico.

Vale ressaltar que em um cenário real, a reconexão com a rede *Wi-Fi* seria realizada pelo *loop*, que quando está sem conexão com a rede, periodicamente tenta realizar a conexão com a rede que foi previamente configurada, como explicado pelo fluxograma da Figura 27.

Figura 62 - *Display serial* no processo reconexão com a internet.

```
Conectando ao Wi-Fi.CONECTADO!

Conectado a rede com o IP: 192.168.63.114

.
Começando processo de upload de historicos registrados offline
Começando upload de:/historico/1.txt
Cod. de resposta HTTP: 201
{"message":"Historico registrado com sucesso!"}
DELETANDO O ARQUIVO: /historico/1.txt
- ARQUIVO DELETADO
Começando upload de:/historico/2.txt
Cod. de resposta HTTP: 201
{"message":"Historico registrado com sucesso!"}
DELETANDO O ARQUIVO: /historico/2.txt
- ARQUIVO DELETADO
Começando upload de:/historico/3.txt
Cod. de resposta HTTP: 201
{"message":"Historico registrado com sucesso!"}
DELETANDO O ARQUIVO: /historico/3.txt
- ARQUIVO DELETADO
Começando upload de:/historico/4.txt
Cod. de resposta HTTP: 201
{"message":"Historico registrado com sucesso!"}
DELETANDO O ARQUIVO: /historico/4.txt
- ARQUIVO DELETADO
Começando upload de:/historico/5.txt
Cod. de resposta HTTP: 201
{"message":"Historico registrado com sucesso!"}
DELETANDO O ARQUIVO: /historico/5.txt
- ARQUIVO DELETADO
Começando upload de:/historico/6.txt
Cod. de resposta HTTP: 201
{"message":"Historico registrado com sucesso!"}
DELETANDO O ARQUIVO: /historico/6.txt
- ARQUIVO DELETADO
Começando upload de:/historico/7.txt
Cod. de resposta HTTP: 201
{"message":"Historico registrado com sucesso!"}
DELETANDO O ARQUIVO: /historico/7.txt
- ARQUIVO DELETADO
```

Fonte: Acervo do autor (2023).

5.3 Custo do *hardware*

Na Tabela 7, estão listados os componentes utilizados no desenvolvimento do protótipo, juntamente com seus respectivos preços. É importante observar que os preços mencionados foram consultados em junho de 2023. Cabe ressaltar que os custos da protoboard, dos jumpers e do frete não foram considerados na elaboração desta tabela.

Tabela 7 - Custo total do protótipo.

Componente	Preço
ESP-WROOM-32 Dev Kit 38 pinos	R\$ 17,28
Módulo DY50	R\$ 27,59
Módulo RC522	R\$ 24,90
Display LCD 16x2 Backlight Azul com Módulo I2C Integrado	R\$ 37,90
Teclado Matricial de Membrana 12 Teclas	R\$ 10,90
Total	R\$ 118,57

Fonte: Acervo do autor (2023).

6 CONSIDERAÇÕES FINAIS

Tendo em vista a proposta e o desenvolvimento do presente trabalho, este capítulo destina-se a descrever os problemas que foram encontrados durante o desenvolvimento, avaliar, comparar e analisar se os objetivos estabelecidos foram atingidos e propor possíveis expansões e melhorias para o presente trabalho.

Durante o desenvolvimento do sistema *web* e da *API*, aproveitou-se a vasta quantidade de recursos disponíveis na internet: sites, vídeos e artigos que utilizam as tecnologias *React.JS* e *Node.JS*. Portanto, a maior parte do trabalho para a elaboração das aplicações *web* (central de comunicação e central de controle e monitoramento) foi estruturar a aplicação de maneira correta e organizada para que o código pudesse ser lido e entendido, facilitando assim a manutenção do código.

Em relação aos objetivos do trabalho, constatou-se através dos testes realizados, que os objetivos do hardware foram alcançados. O dispositivo foi capaz de interagir com a *API* e com o banco de dados do *Firebase*, além de conseguir realizar a correta identificação dos usuários por diversos métodos de entrada tanto no modo *online* quanto *offline*.

Ao avaliar o *software*, os testes evidenciaram que a *API* foi capaz de lidar com as funcionalidades de cadastro. Além disso, foi possível gerenciar as permissões de entrada dos usuários e visualizar o histórico de acesso. Estas funcionalidades viabilizam a utilização do sistema para o controle de acesso aos ambientes.

Porém, ainda são possíveis diversas melhorias, que podem ser implementadas em trabalhos posteriores. Algumas melhorias foram listadas e podem ser consultadas no Quadro 5.

Quadro 5 - Melhorias para trabalhos futuros.

MELHORIAS PARA TRABALHOS FUTUROS	
CATEGORIA	MELHORIA
Site	Implementação de tecnologia em tempo real (<i>socket.io</i>).
	Desenvolver um design responsivo.
API	Implementação de tecnologia em tempo real (<i>socket.io</i>).
Software da fechadura	Cadastro de novas impressões digitais e novos cartões <i>RFID</i> pela própria fechadura.
	Desenvolvimento de uma biblioteca própria para o programa executado no ESP-WROOM-32.
Hardware	Cadastro de novas impressões digitais e novos cartões <i>RFID</i> pela própria fechadura.
	Uso de um sensor <i>reed-switch</i> para ter o controle mais preciso de abertura e fechamento.
	Um <i>push-button</i> para realizar a saída do ambiente.
Projeto	Analisar e comparar o uso do protocolo <i>MQTT</i> e do protocolo <i>HTTP</i> .
	Testes com a fechadura de sobrepor juntamente com a mola aérea para fechamento da porta.

Fonte: Acervo do autor (2023).

Todos os códigos-fonte das aplicações desenvolvidas neste projeto foram adicionados ao *github*, e podem ser consultados os códigos-fonte da *API*¹⁶, *site*¹⁷ e o *software* do dispositivo¹⁸.

A partir do que foi apresentado neste trabalho, pode-se afirmar que o sistema e o protótipo apresentado conseguem controlar e registrar os acessos de entrada em um ambiente, autenticando os usuários de maneira correta e tendo informações disponíveis e armazenadas para a visualização dos usuários.

¹⁶ Disponível em: github.com/BernardoS/MasterLock-API-Code;

¹⁷ Disponível em: github.com/BernardoS/MasterLock-Site-Code;

¹⁸ Disponível em: github.com/BernardoS/MasterLock-Code/.

REFERÊNCIAS

API REST. 2020. Elaborada por Red Hat. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>. Acesso em: 04 jun. 2022.

BAILÃO, Juliano. **Por que utilizamos *Single Page Applications*: spa?. SPA?.** 2018. Disponível em: <https://imasters.com.br/front-end/por-que-utilizamos-single-page-applications-spa#:~:text=Single%20page%20application%20quer%20dizer,com%20a%20necessidade%20do%20usu%C3%A1rio..> Acesso em: 04 jun. 2022.

BINKER, Carlos Alberto *et al.* **ACCESO REMOTO A DISPOSITIVOS IOT MEDIANTE TÉCNICAS DE MENSAJERÍA EMPLEANDO BOTS Y FREERTOS.** 2020. Disponível em: <https://webcache.googleusercontent.com/search?q=cache:Eu8ZQZvIFz4J:https://red di.unlam.edu.ar/index.php/ReDDi/article/download/117/243/+&cd=25&hl=ptBR&ct=cl nk&gl=br>. Acesso em: 05 jun. 2020.

BIOMETRIA: Métodos de Identificação. Métodos de Identificação. Disponível em: https://www.gta.ufrj.br/grad/09_1/versao-final/biometria/Mtodosdeldentificao.html. Acesso em: 30 maio 2022.

BRAGA, Newton C.. **Como funcionam as UARTs:** tel006. TEL006. Disponível em: <https://www.newtonbraga.com.br/index.php/telecom-artigos/1709->. Acesso em: 05 jun. 2022.

BRENNER, Gabriel Pitágoras Silva e; BIZARRIA, Walter. **Sistema de Controle de Acesso com Biometria da Digital.** Disponível em: <https://www.aedb.br/seget/arquivos/artigos11/44914520.pdf>. Acesso em: 03 jun. 2022.

CANTELON, Mike; HARTE, Marc; HOLOWAYCHUK, T.J.; RAJLICH, Nathan. **Node.js in Action.** 2014. Disponível em: [http://dl.ketabesabz.com/ebooks3/up/nodejs-in-action_\[www.ketabesabz.com\].pdf](http://dl.ketabesabz.com/ebooks3/up/nodejs-in-action_[www.ketabesabz.com].pdf). Acesso em: 04 jun. 2022.

CATSOULIS, John. **Designing Embedded Hardware.** 2. ed. Sebastopol: O'Reilly, 2005. 456 p.

CLARKE, Roger. **Human Identification in Information Systems:** Management Challenges and Public Policy Issues. *Information Technology & People*, v. 7, p. 6-37, 1994. Disponível em: <http://www.rogerclarke.com/DV/HumanID.html>. Acesso em: 30 maio 2022.

CODÁ, L. M. MEMÓRIA FLASH. 2013. Disponível em: https://edisciplinas.usp.br/pluginfile.php/4366748/mod_resource/content/1/MEM%C3%93RIA%20FLASH_2013.pdf#:~:text=A%20mem%C3%B3ria%20flash%2C%20embora%20seja,sem%20precisar%20de%20alimenta%C3%A7%C3%A3o%20el%C3%A9trica. acesso em: 31 de mai. de 2023

COMPONENTES e Props. Disponível em: <https://pt-br.reactjs.org/docs/components-and-props.html>. Acesso em: 04 jun. 2022.

COMPREENDER UART. Elaborada por Rohde & Schwarz. Disponível em: https://www.rohde-schwarz.com/br/produtos/teste-e-medicao/osciloscopios/educational-content/compreender-uart_254524.html#:~:text=UART%20significa%20Transmissor%20Receptor%20ass%C3%ADncrono,receber%20em%20ambas%20as%20dire%C3%A7%C3%B5es.. Acesso em: 05 jun. 2022.

COSTA, Luciano R.; OBELHEIRO, Rafael R.; FRAGA, Joni S. **Introdução à biometria**. Livro-texto de Minicursos-VI SBSeg, 2006. Disponível em: http://www.advancedsourcecode.com/minicurso_biometria.pdf. Acesso em: 30 maio 2022.

DA SILVA MOTA, Leticia; GUEDES, Elloá B. **Controle de Acesso a Múltiplos Ambientes com Internet das Coisas**. Disponível em: <http://www.elloaguedes.com/files/papers/elloa-encosis20191.pdf>. Acesso em: 06 jun. 2022.

DA SILVA, BERNARDO SOUZA FERREIRA et al. **MEDIDOR DA SATURAÇÃO DE OXIGÊNIO NO SANGUE, FREQUÊNCIA CARDÍACA E TEMPERATURA CORPORAL-COVID-OSTI**. Disponível em: <http://www.ifto.edu.br/ifto/reitoria/pro-reitorias/proae/ifto4change/relatorios-dos-projetos-da-2a-edicao/relatorio-equipe-4t.pdf>. Acesso em: 05 jun. 2022.

DARVISH, Ali. **Client-Server Architecture**. Disponível em: https://darvishdarab.github.io/cs421_f20/docs/readings/client_server/. Acesso em: 06 jun. 2022.

DE ALMEIDA, LUCAS CAVALCANTE. **APLICAÇÕES DA TECNOLOGIA DE IDENTIFICAÇÃO POR RÁDIO FREQUÊNCIA-RFID**. 2011. Disponível em: http://www.cgeti.ufc.br/monografias/LUCAS_CAVALCANTE_DE_ALMEIDA.pdf. Acesso em: 02 jun. 2022.

DE SOUZA, Marco Antonio. **A Biometria e suas Aplicações**. Revista Brasileira de Ciências Policiais, v. 11, n. 2, p. 79-102, 2020. Disponível em: <https://periodicos.pf.gov.br/index.php/RBCP/article/view/710>. Acesso em: 30 maio 2022.

DELGADO, Sergio Carlavilla. **Tutorial sobre Comunicações Seriais e UART**. Disponível em: <https://docs.freebsd.org/pt-br/articles/serial-uart/>. Acesso em: 05 jun. 2022.

DISPLAY LCD 16x2 I2C Backlight Azul. Elaborada por FilipeFlop. Disponível em: <https://www.filipeflop.com/produto/display-lcd-16x2-i2c-backlight-azul/>. Acesso em: 06 jun. 2022.

DUARTE, Nuno Filipe Brandão. **Frameworks e Bibliotecas Javascript**. 2015. Tese de Doutorado. Disponível em: https://recipp.ipp.pt/bitstream/10400.22/8223/1/DM_NunoDuarte_2015_MEI.pdf. Acesso em: 02 jun. 2022.

ESP32 com RFID: Controle de Acesso. [S. l.], 9 fev. 2018. Disponível em: <https://www.fernandok.com/2018/02/esp32-com-rfid-controle-de-acesso.html>. Acesso em: 1 jun. 2023.

FECHADURA Tetra Soprano Sobrepor - 235. Soprano. Disponível em: <https://www.gravia.com/fechadura-235-sobrepor-tetra-soprano/p>. Acesso em: 02 jun. 2023.

FIREBASE Realtime Database. Elaborada por Firebase. Disponível em: <https://firebase.google.com/docs/database?hl=pt-br>. Acesso em: 04 jun. 2022.

FIREBASE. Elaborada por Flutter. Disponível em: <https://docs.flutter.dev/development/data-and-backend/firebase>. Acesso em: 04 jun. 2022.

FRONT-END e back-end: entenda a diferença e tudo que precisa saber sobre o assunto. entenda a diferença e tudo que precisa saber sobre o assunto. Disponível em: <https://www.igti.com.br/blog/front-end-e-back-end>. Acesso em: 04 jun. 2022.

FRONT-END E BACK-END: ENTENDA ESSAS DUAS ÁREAS DO DESENVOLVIMENTO. ENTENDA ESSAS DUAS ÁREAS DO DESENVOLVIMENTO. 2021. Disponível em: <https://blog.somostera.com/carreiras-digitais/front-end-e-back-end#:~:text=O%20front%2Dend%20pode%20ser,funcionamento%20estrutural%20de%20uma%20p%C3%A1gina..> Acesso em: 04 jun. 2022. <https://portal.vidadesilicio.com.br/modulo-rfid-rc522-mifare/>. Acesso em: 30 maio

GBUR, Felipe. **Módulo RFID RC522 Mifare com Arduino**. 2017. Disponível em: 2022.

GHARSSELLAOUI, Bayrem. **Internet of Things prototyping with Firebase**: how to do more with less. how to do more with less. 2019. Disponível em: <https://www.freecodecamp.org/news/iot-prototyping-with-firebase-doing-more-with-less-2f5c746dac8b/>. Acesso em: 04 jun. 2022.

GOLER, Vernon. **Using the Universal Asynchronous Receiver Transmitter (UART) eTPU Function**. 2004. Disponível em: <https://www.nxp.com/docs/en/application-note/AN2853.pdf>. Acesso em: 05 jun. 2022.

GONÇALVES, Vinicius Rocha. **Sistema de controle de acesso utilizando autenticação por RFID e gerenciamento por meio de software WEB**. 2019. Disponível em: https://monografias.ufop.br/bitstream/35400000/2222/3/MONOGRAFIA_SistemaControleAcesso.pdf. Acesso em: 06 jun. 2022.

GRIEGER, Christian. **RFID und Arduino**. 2018. Disponível em: <https://elektro.turanis.de/html/prj102/>. Acesso em: 06 jun. 2022.

HTML: Linguagem de Marcação de Hipertexto. Linguagem de Marcação de Hipertexto. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. Acesso em: 03 jun. 2022.

IDENTIFICAÇÃO por impressão digital com Arduino. [S. l.], 30 jul. 2020. Disponível em: <https://www.fernandok.com/2020/07/identificacao-por-impressao-digital-com.html>. Acesso em: 1 jun. 2023.

JÁ ouviu falar em Single Page Applications? Disponível em: <https://www.devmedia.com.br/ja-ouviu-falar-em-single-page-applications/39009>. Acesso em: 04 jun. 2022.

L., Andrei. **O Que é React e Como Funciona?** 2021. Disponível em: <https://www.hostinger.com.br/tutoriais/o-que-e-react-javascript>. Acesso em: 04 jun. 2022.

LOPEZ, Gabriel Niquini. **Sistema para monitoramento de acesso utilizando Arduino e leitura biométrica.** 2015. Disponível em: https://monografias.ufop.br/bitstream/35400000/1200/1/MONOGRAFIA_SistemaMonitoramentoAcesso.pdf. Acesso em: 27 jun. 2023

LOURENÇO, Gonçalo Filipe da F. **Reforço da segurança das biométricas utilizando codificação de fonte distribuída.** 2009. Tese de Doutorado. Dissertação de Mestrado, Instituto Superior Técnico, Universidade Técnica de Lisboa. Disponível em: <https://fenix.tecnico.ulisboa.pt/thesis/2353642254448>. Acesso em: 03 jun. 2022.

MAKE your app the best it can be. Elaborado por Firebase. Disponível em: <https://firebase.google.com/>. Acesso em: 04 jun. 2022.

MANENTE, Luis Otávio; CRESPO, Pedro Moreno. **DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE DE ACESSO COM ARMAZENAMENTO DE DADOS EM NUVEM.** Disponível em: http://riut.utfpr.edu.br/jspui/bitstream/1/16241/1/PG_COELE_2019_1_07.pdf. Acesso em: 05 jun. 2022.

MAZIERO, Carlos A. Sistemas operacionais: conceitos e mecanismos. **Livro aberto**, 2014.

MENDONÇA, Hélio Sousa. **SPI e I2C.** Disponível em: <https://paginas.fe.up.pt/~hsm/misc/old/comp/spi-e-i2c/>. Acesso em: 05 jun. 2022.

MH102 A. Intelbras. Disponível em: <https://www.intelbras.com/pt-br/mola-aerea-para-portas-mh-102-a>. Acesso em: 02 jun. 2023.

MÓDULO LEITOR *RFID* RC522 MIFARE 13,56 MHZ. Disponível em: <https://www.piscaled.com.br/modulo-leitor-rfid-rc522-mifare-1356-mhz>. Acesso em: 06 jun. 2022.

NARCISO, Marcelo Gonçalves. **Aplicação da tecnologia de identificação por radiofrequência (RFID) para controle de bens patrimoniais pela web.** Embrapa Informática Agropecuária-Artigo em periódico indexado (ALICE), 2008. Disponível em: <https://www.alice.cnptia.embrapa.br/bitstream/doc/17793/1/rfid.PDF>. Acesso em: 02 jun. 2022.

NASSAR, Victor; VIEIRA, Milton Luiz Horn. **A aplicação de *RFID* na logística: um estudo de caso do Sistema de Infraestrutura e Monitoramento de Cargas do Estado de Santa Catarina.** Gestão & Produção, v. 21, p. 520-531, 2014. Disponível em: <https://www.scielo.br/j/gp/a/kj6hDnMnRVd56nCytKCW8rK/?format=html&lang=pt>. Acesso em: 02 jun. 2022.

NODE.JS Introduction. Elaborada por W3 schools. Disponível em: https://www.w3schools.com/nodejs/nodejs_intro.asp. Acesso em: 04 jun. 2022.

NODE.JS Introduction. Elaborada por W3 schools. Disponível em: https://www.w3schools.com/nodejs/nodejs_intro.asp. Acesso em: 04 jun. 2022. Elaborada por node.js. Disponível em: <https://nodejs.dev/learn>. Acesso em: 04 jun. 2022.

NODE.JS Introduction. Elaborada por W3 schools. Disponível em: https://www.w3schools.com/nodejs/nodejs_intro.asp. Acesso em: 04 jun. 2022. Elaborada por node.js. Disponível em: <https://nodejs.dev/learn>. Acesso em: 04 jun. 2022. Elaborada por node.js. Disponível em: <https://nodejs.dev/learn>. Acesso em: 04 jun. 2022.

O QUE é JavaScript? Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript. Acesso em: 03 jun. 2022.

O QUE é middleware? Explicação sobre software de middleware. [S. l.], 2023. Disponível em: <https://aws.amazon.com/pt/what-is/middleware/>. Acesso em: 4 jun. 2023.

O QUE é *RFID*? Disponível em: https://www.gta.ufrj.br/grad/07_1/rfid/RFID_arquivos/o%20que%20e.htm. Acesso em: 03 jun. 2022.

O QUE É Um Microcontrolador? 2020. Elaborada por IEEE RAS UFCG. Disponível em: <https://edu.ieee.org/br-ufcgras/o-que-e-um-microcontrolador/>. Acesso em: 05 jun. 2022.

O QUE é uma *API*? Disponível em: <https://aws.amazon.com/pt/what-is/api/>. Acesso em: 04 jun. 2022.

O QUE é uma *API*? Elaborada por Red Hat. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces>. Acesso em: 04 jun. 2022.

OLIVEIRA, Alessandro de Souza; PEREIRA, Milene Franco. **Estudo da tecnologia de identificação por radiofrequência-*RFID***. 2006. Disponível em: <https://bdm.unb.br/handle/10483/829>. Acesso em: 02 jun. 2022.

OLIVEIRA, Euler. **Como usar com Arduino:** kit rfid mfrc522. KIT RFID MFRC522. Disponível em: <https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-kit-rfid-mfrc522>. Acesso em: 30 jun. 2022.

OLIVEIRA, Euler. **Como usar com Arduino:** módulo adaptador i2c para display lcd (16x2 / 20x4). Módulo Adaptador I2C para Display LCD (16X2 / 20X4). Disponível em: <https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-modulo-adaptador-i2c-para-display-lcd-16x2-20x4>. Acesso em: 30 maio 2022.

OLIVEIRA, Vinícius G. de; NIIZU, Fábio Y.; BASSETO, Fernando. **Internet das coisas**. 2017. Disponível em: <https://www.osetoreletrico.com.br/wp-content/uploads/documentos/fasciculos/Ed-132-Fasc%c3%adculo-Capitulo-I-Internet-das-coisas.pdf>. Acesso em: 04 jun. 2022.

ORLANDI, Claudio. **Firestore**: serviços, vantagens, quando utilizar e integrações. serviços, vantagens, quando utilizar e integrações. 2018. Elaborada por Rocketseat. Disponível em: <https://blog.rocketseat.com.br/firebase/>. Acesso em: 04 jun. 2022.

PATTABIRAMAN, Krishna. **HOW TO SET UP A KEYPAD ON AN ARDUINO**. Disponível em: <https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/>. Acesso em: 06 jun. 2022.

PEDROSO, Marcelo Caldeira; ZWICKER, Ronaldo; SOUZA, Cesar Alexandre de. **Adoção de RFID no Brasil**: um estudo exploratório. RAM. Revista de Administração Mackenzie, v. 10, n. 1, p. 12-36, 2009. Disponível em: <https://www.scielo.br/j/ram/a/TwbxMBFfsjTTsDZTKdJBYRC/?format=pdf&lang=pt>. Acesso em: 02 jun. 2022.

PEREIRA, Vitor; DIAS, Samaherni; DE QUEIROZ, Kurios. Sistema inteligente para controle de acesso e monitoramento de múltiplos ambientes (*class control*). In: **Anais Estendidos do IX Simpósio Brasileiro de Engenharia de Sistemas Computacionais**. SBC, 2019. p. 25-30. Disponível em: https://sol.sbc.org.br/index.php/sbesc_estendido/article/view/8631/8532. Acesso em: 06 jun. 2022.

PORTA Serial e Porta Paralela. 2020. Disponível em: <https://br.ccm.net/contents/407-porta-serial-e-porta-paralela>. Acesso em: 05 jun. 2022.

RABELLO, Ladislau Marcelino. **Programa em linguagem JAVA para comunicação Serial**. 2009. Disponível em: <https://www.infoteca.cnptia.embrapa.br/bitstream/doc/663550/1/CT1092009.pdf>. Acesso em: 05 jun. 2022.

REACT: Introdução. Introdução. Disponível em: <https://pt-br.reactjs.org/docs/getting-started.html>. Acesso em: 04 jun. 2022.

REACT: Uma biblioteca *JavaScript* para criar interfaces de usuário. Uma biblioteca JavaScript para criar interfaces de usuário. Disponível em: <https://pt-br.reactjs.org/>. Acesso em: 04 jun. 2022.

REIMBOLD, Manuel M. P.; RASIA, Luiz A.; WEBER, Marlon. **ESTUDO E DESEMPENHO DAS PORTAS PARALELA E SERIAL DOS PCS COMO AMBIENTE DE APRENDIZAGEM**. Disponível em: <http://www.abenge.org.br/cobenge/legado/arquivos/19/artigos/177.pdf>. Acesso em: 05 jun. 2022.

RFID. Elaborado por I3C Soluções. Disponível em: <https://i3csolucoes.com.br/wp-content/uploads/2020/06/Etiquetas-RFID-capa.jpg>. Acesso em: 06 jun. 2022.

ROVEDA, Ugo. **JAVASCRIPT**: o que é, para que serve e como funciona o js?. O QUE É, PARA QUE SERVE E COMO FUNCIONA O JS?. 2020. Disponível em: <https://kenzie.com.br/blog/javascript/>. Acesso em: 03 jun. 2022.

ROVEDA, Ugo. **REACT**:: o que é, como funciona e porque usar e como aprender. O QUE É, COMO FUNCIONA E PORQUE USAR E COMO APRENDER. 2020. Disponível em: <https://kenzie.com.br/blog/react/#:~:text=O%20React%20%C3%A9%20uma%20bibli>

oteca%20criada%20para%20construir%20telas%20de, mobile%20ou%20qualquer%20outro%20dispositivo.. Acesso em: 04 jun. 2022.

SANTIAGO, Eike; BENTO, Diogo; COSTA, Breno. *LetMeIn*: sistema de controle de acesso para ambientes educacionais. In: **Anais da XIX Escola Regional de Computação Bahia, Alagoas e Sergipe**. SBC, 2019. p. 382-385. Disponível em: <https://sol.sbc.org.br/index.php/erbase/article/view/8999/8900>. Acesso em: 06 jun. 2022

SENSOR Leitor Biométrico Impressão Digital DY50. Disponível em: <https://www.casadarobotica.com/sensores-modulos/sensores/biometrico/sensor-leitor-biometrico-impressao-digital-dy50>. Acesso em: 06 jun. 2022.

SILVEIRA, Diogo Weber et al. **Desenvolvimento de uma fechadura eletrônica**: um sistema de controle de acesso com registro em banco de dados e site de gerenciamento. 2021. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/223836/tcc%20vers%c3%a3o%20final%20Diogo_assinado.pdf?sequence=1&isAllowed=y. Acesso em: 06 jun. 2022.

SISTEMAS Microprocessados - Aula 10 - Interface I2C. [S.L]: Leandro Mattioli, 2020. Son., color. Disponível em: <https://www.youtube.com/watch?v=mOSuHagLwpE>. Acesso em: 05 jun. 2022.

SOUTO, Mario. **O que é front-end e back-end?** 2019. Disponível em: <https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end>. Acesso em: 04 jun. 2022.

SOUZA, João André Noberto de. **UMA API REST PARA CONSUMO DE DADOS GEORREFERENCIADOS UTILIZANDO ARANGODB**. 2018. Disponível em: <https://di.uern.br/tccs2019/html/ltr/PDF/014003929.pdf>. Acesso em: 04 jun. 2022.

TECLADO Matricial 3x4 12 Teclas. Disponível em: https://www.usinainfo.com.br/teclado-matricial-e-buttons/teclado-matricial-3x4-12-teclas-4975.html?search_query=teclado&results=32. Acesso em: 06 jun. 2022.

TECLADO MATRICIAL 3X4 TIPO MEMBRANA. Vistrônica: Tienda Virtual de Electrónica. Disponível em: <https://www.vistronica.com/conectores-cables-y-switches/teclado-matricial-3x4-tipo-membrana-detail.html>. Acesso em: 27 mai. 2023.

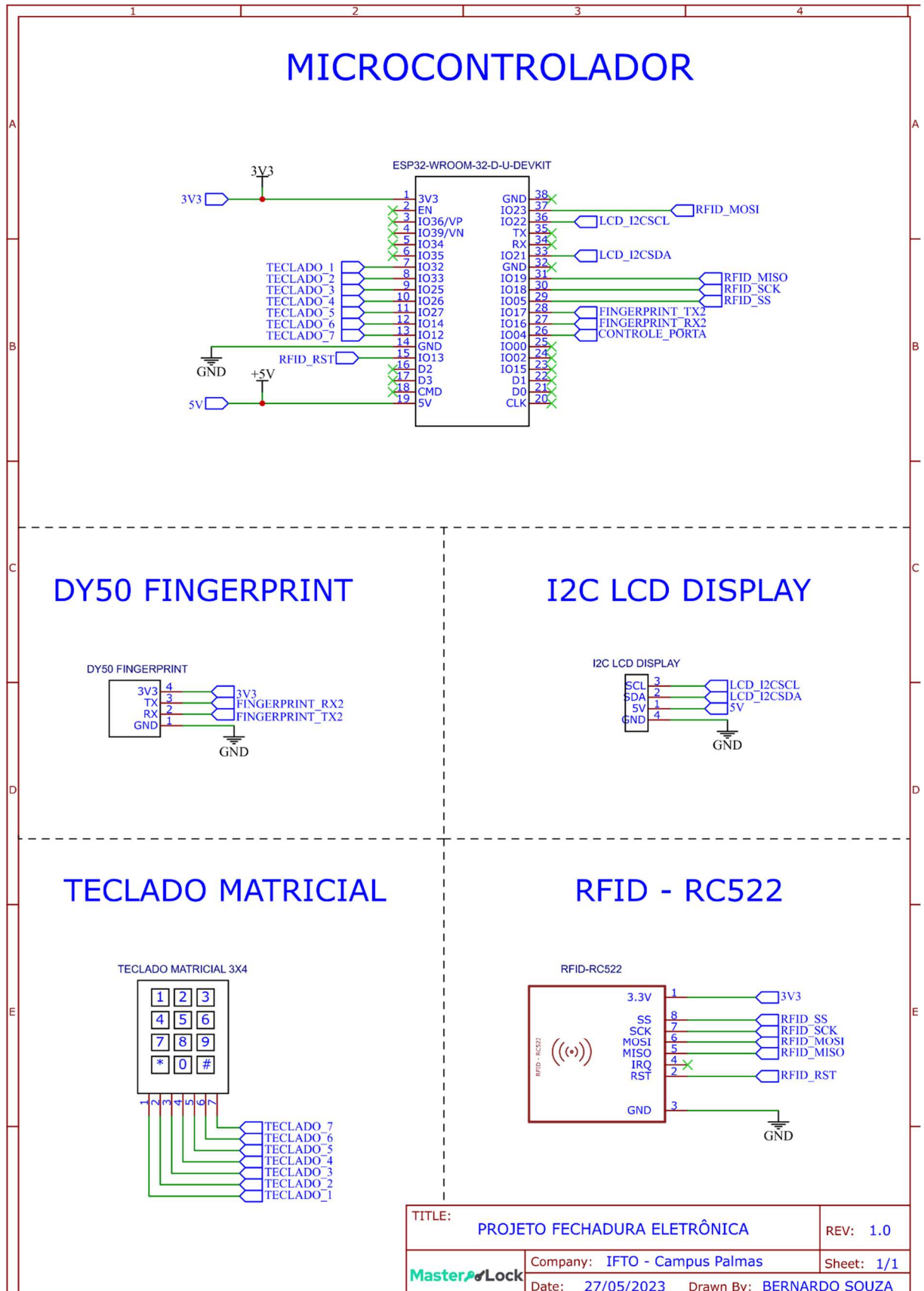
TECNOLOGIA Node.js. Elaborada por DEVMEDIA. Disponível em: <https://www.devmedia.com.br/guia/node-js/40312>. Acesso em: 04 jun. 2022.

UMA VISÃO geral do HTTP. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>. Acesso em: 04 jun. 2022.

WEINSTEIN, Ron. **RFID**: a technical overview and its application to the enterprise. IT professional, v. 7, n. 3, p. 27-33, 2005. Disponível em: https://web.archive.org/web/20131020194755id_/http://www.cs.sunysb.edu/~jgao/CS-E370-spring07/RFID.pdf. Acesso em: 02 jun. 2022.

WOZNIEWICZ, Brandon. **The Difference Between a Framework and a Library**. 2019. Disponível em: <https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/>. Acesso em: 30 mai. 2022.

APÊNDICE A – ESQUEMÁTICO DO CIRCUITO.



APÊNDICE B – CÓDIGO-FONTE DO *HARDWARE* DESENVOLVIDO

```
#pragma region Configurações das bibliotecas
```

```
//Biblioteca do módulo de leitura da digital
#include <Adafruit_Fingerprint.h>
//Biblioteca do Firebase
#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"
//Bibliotecas para interação com a API
#include <WiFi.h>
#include <HTTPClient.h>
#include <Arduino_JSON.h>
//Bibliotecas do módulo RFID
#include <SPI.h>
#include <MFRC522.h>
//Biblioteca do teclado matricial
#include <Keypad.h>
//Biblioteca do Display LCD
#include <LiquidCrystal_I2C.h>
//Bibliotecas do sistema de arquivos
#include "FS.h"
#include <LittleFS.h>
//Biblioteca para recuperação de tempo
#include "time.h"
```

```
#pragma endregion Configurações das bibliotecas
```

```
#pragma region Configurações da fechadura e configurações de rede
```

```
//Id da fechadura cadastrado no Firebase
String IdFechadura = "ID_DA_FECHADURA";
```

```
//Configurações da rede e do Firebase
String WIFI_PASSWORD = "SENHA";
#define WIFI_SSID "REDE"
#define CHAVE_API "CHAVE_API"
#define URL_DO_BANCO "URL_DO_BANCO"
```

```
unsigned long controleReconexaoWifi = 0;
```

```
//Configurações do Firebase
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
bool signupOK = false;
```

```
#pragma endregion Configurações da fechadura e configurações de rede
```

```
#pragma region Configurações dos periféricos
```

```
//Configurações da recuperação de data e hora
```

```
String DiaEHora;
```

```
const char* ntpServer = "0.br.pool.ntp.org";
```

```
const long  gmtOffset_sec = -70800;
```

```
const int  daylightOffset_sec = 60000;
```

```
//Configurações do RFID
```

```
#define SS_PIN 5
```

```
#define RST_PIN 27
```

```
#define SIZE_BUFFER 18
```

```
#define MAX_SIZE_BLOCK 16
```

```
MFRC522 rfid(SS_PIN, RST_PIN);
```

```
MFRC522::MIFARE_Key key;
```

```
MFRC522::StatusCode status;
```

```
//Configurações do Teclado
```

```
#define NUM_LINHAS 4
```

```
#define NUM_COLUNAS 3
```

```
char teclas[NUM_LINHAS][NUM_COLUNAS] = {
```

```
{'1', '2', '3'},
```

```
{'4', '5', '6'},
```

```
{'7', '8', '9'},
```

```
{'*', '0', '#'}
};
```

```
byte pinos_linha[NUM_LINHAS] = {12, 14, 27, 26}; // GIOP18, GIOP5, GIOP17,
GIOP16 - pinos de conexão das linhas do teclado
```

```
byte pinos_colunas[NUM_COLUNAS] = {25, 33, 32}; // GIOP4, GIOP0, GIOP2 - pinos
de conexão das colunas do teclado
```

```
Keypad keypad = Keypad( makeKeymap(teclas), pinos_linha, pinos_colunas,
NUM_LINHAS, NUM_COLUNAS );
```

```
//Confogurações do displayLCD
```

```
int colunasLCD = 16;
```

```
int linhasLCD = 2;
```

```
LiquidCrystal_I2C lcd(0x27, colunasLCD, linhasLCD);
```

```
//Configurações do sensor de digital
```

```
const uint32_t password = 0x0;
```

```
String idsDigitais[16] =
```

```
{"BERNARDO", "MATHEUS", "DONOVAN", "JOAO", "TAMASHIRO", "IAGO", "ISAIAS", "
THIAGO", "JHONES", "BEATRIZ", "EZEQUIEL", "BRUNO", "MARCUS", "JANAINA", "PA
ULO", "SAMUEL"};
```

```
Adafruit_Fingerprint fingerprintSensor = Adafruit_Fingerprint(&Serial2, password);
```

```
#pragma endregion Configurações dos periféricos
```

```
#pragma region Configurações genéricas
```

```
//Configurações genéricas
```

```
unsigned long controleLeituraPorta = 0;
unsigned long controleAtualizacaoPermissao = 0;
unsigned long controleUploadHistorico = 0;
String statusFechadura = "false";
int pinoFechadura = 4;
bool estaOnline = false;
```

```
//Configurações do menu
```

```
int menuStep = 0;
unsigned long controleExibicaoMenu = 0;
```

```
//Configurações para o armazenamento offline
```

```
#define FORMATAR_LITTLEFS_CASO_FALHE false
```

```
#pragma endregion Configurações genéricas
```

```
#pragma region Funcionalidades do sistema de arquivos
```

```
void deletarArquivo(fs::FS &fs, const char * path){
    Serial.printf("DELETANDO O ARQUIVO: %s\r\n", path);
    if(fs.remove(path)){
        Serial.println("- ARQUIVO DELETADO");
    } else {
        Serial.println("- ERRO AO DELETAR");
    }
}
```

```
void escreverArquivo(fs::FS &fs, const char * path, const char * message){
    Serial.printf("ESCREVENDO NO ARQUIVO: %s\r\n", path);
```

```
    File file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("- ERRO AO ABRIR ARQUIVO PARA ESCREVER");
        return;
    }
    if(file.print(message)){
        Serial.println("- A ESCRITA FOI UM SUCESSO");
    } else {
        Serial.println("- A ESCRITA FALHO");
    }
    file.close();
}
```

```
bool achouTextoNoArquivo(fs::FS &fs, const char * path, const char * textoDeBusca){
```

```

bool found = false;
Serial.printf("LENDO ARQUIVO: %s\r\n", path);

File file = fs.open(path);
if(!file || file.isDirectory()){
    Serial.println("- ERRO AO ABRIR ARQUIVO PARA REALIZAR LEITURA");
    return found;
}

Serial.println("- LENDO O ARQUIVO:");
while(file.available()){
    if(file.find(textoDeBusca)){
        Serial.println("ACHOU A PALAVRA:");
        Serial.println(textoDeBusca);
        found = true;
    }
}
file.close();

return found;
}

void listarPastas(fs::FS &fs, const char * dirname, uint8_t levels){
    Serial.printf("Listando pastas : %s\r\n", dirname);

    File root = fs.open(dirname);
    if(!root){
        Serial.println("- erro ao abrir pasta");
        return;
    }
    if(!root.isDirectory()){
        Serial.println(" - não é uma pasta");
        return;
    }

    File file = root.openNextFile();
    while(file){
        if(file.isDirectory()){
            Serial.print(" PASTA : ");
            Serial.println(file.name());
            if(levels){
                listarPastas(fs, file.name(), levels - 1);
            }
        } else {
            Serial.print(" ARQUIVO: ");
            Serial.print(file.name());
            Serial.print("\tTAMANHO: ");
            Serial.println(file.size());
        }
    }
}

```

```

    }
    file = root.openNextFile();
  }
}

```

```
int contarPastas(fs::FS &fs, const char * dirname, uint8_t levels){
```

```

    int countDirectories = 0;

    File root = fs.open(dirname);
    if(!root){
        return countDirectories;
    }
    if(!root.isDirectory()){
        return countDirectories;
    }

    File file = root.openNextFile();
    while(file){
        if(file.isDirectory()){
            if(levels){
                listarPastas(fs, file.name(), levels -1);
            }
            countDirectories++;
        }
        file = root.openNextFile();
    }
    return countDirectories;
}

```

```
int contarArquivosNaPasta(fs::FS &fs, const char * dirname){
    int countFiles = 0;
```

```

    File root = fs.open(dirname);
    if(!root){
        Serial.println("- ERRO AO ABRIR PASTA");
        return countFiles;
    }
    if(!root.isDirectory()){
        Serial.println(" - NÃO É UMA PASTA");
        return countFiles;
    }

```

```
    File file = root.openNextFile();
```

```

    while(file){
        if(file.isDirectory() == false){
            countFiles++;
        }
    }

```

```

        file = root.openNextFile();
    }
    return countFiles;
}

void criarPasta(fs::FS &fs, const char * path){
    Serial.printf("Criando pasta: %s\n", path);
    if(fs.mkdir(path)){
        Serial.println("Pasta criada");
    } else {
        Serial.println("Erro ao criar pasta");
    }
}

void lerArquivo(fs::FS &fs, const char * path){
    Serial.printf("LENDO ARQUIVO: %s\r\n", path);

    File file = fs.open(path);
    if(!file || file.isDirectory()){
        Serial.println("- ERRO AO ABRIR ARQUIVO PARA LEITURA");
        return;
    }

    Serial.println("- LEU DO ARQUIVO:");
    while(file.available()){
        Serial.write(file.read());
    }
    Serial.println("");
    file.close();
}

String extrairTextoDoArquivo(const char* filename){
    auto file = LittleFS.open(filename, "r");
    size_t filesize = file.size();
    String data = file.readString();
    file.close();
    return data;
}

#pragma endregion Funcionalidades do sistema de arquivos

#pragma region Configurações iniciais do Setup

void configurarDiretorios(){

    if(!LittleFS.begin(FORMATAR_LITTLEFS_CASO_FALHE)){
        Serial.println("Sistema de arquivo LittleFS montado com sucesso");
        return;
    }
}

```

```

    }

    listarPastas(LittleFS, "/", 0);
    int qtdDirectories = contarPastas(LittleFS, "/", 0);
    Serial.print("Quantidade de diretórios: ");
    Serial.println(qtdDirectories);
    if(qtdDirectories == 0){
        criarPasta(LittleFS, "/permissoes");
        criarPasta(LittleFS, "/historico");
        listarPastas(LittleFS, "/", 0);
    }
}

void configurarSensorDeDigital(){
    fingerprintSensor.begin(57600);
    if(!fingerprintSensor.verifyPassword())
    {
        Serial.println(F("Não foi possível conectar ao sensor. Verifique a senha ou a
conexão"));
        while(true);
    }
}

#pragma endregion Configurações iniciais do Setup

#pragma region Configurações de rede e horário

String retornaDataEHoraAtual(){
    struct tm tempoEHora;
    char dataFormatada[64];
    String dataFormatadaEmString;

    if(!getLocalTime(&tempoEHora)){
        Serial.println("Erro ao recuperar data e hora.");
        return "Erro ao recuperar data";
    }

    strftime(dataFormatada, 64, "%d/%m/%Y %H:%M:%S", &tempoEHora);

    dataFormatadaEmString = dataFormatada;

    return dataFormatadaEmString;
}

void tentarConexaoWifi(){
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

```

```

Serial.print("Conectando ao Wi-Fi");

int tentativasMaximasDeConexao = 20;
int tentativas = 0;
while (tentativas < 20) {
  if(WiFi.status() == WL_CONNECTED){
    Serial.println("CONECTADO!");
    tentativas = 20;
    Serial.println();
    Serial.print("Conectado a rede com o IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();
    estaOnline = true;
  }
  Serial.print(".");
  delay(300);
  tentativas++;
}
Serial.println("");
configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
}

void tentarConexaoFirebase(){
  config.api_key = CHAVE_API;
  config.database_url = URL_DO_BANCO;

  if (Firebase.signUp(&config, &auth, "", "")) {
    Serial.println("ok");
    signupOK = true;
  }
  else {
    Serial.printf("%s\n", config.signer.signupError.message.c_str());
  }

  config.token_status_callback = tokenStatusCallback;
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
}

#pragma endregion Configurações de rede e horário

void setup() {
  pinMode(pinoFechadura, OUTPUT);
  Serial.begin(115200);

  tentarConexaoWifi();

  tentarConexaoFirebase();
}

```

```

DiaEHora = retornaDataEHoraAtual();
Serial.print("Iniciando o programa dia:");
Serial.println(DiaEHora);

SPI.begin();
rfid.PCD_Init();

lcd.init();
lcd.backlight();

configurarSensorDeDigital();

configurarDiretorios();
}

#pragma region Funções do modo offline

void registrarHistoricoOffline(String permissaoOffline, String operacao){
    String horario = retornaDataEHoraAtual();
    String historico = "{"permissaoOffline\":"\""+ permissaoOffline +"\", \"operacao\":"\""+
operacao +"\", \"horario\":"\""+ horario +"\"}";
    int qtdHistoricos = contarArquivosNaPasta(LittleFS, "/historico");
    qtdHistoricos++;
    String nomeArquivo = String(qtdHistoricos);
    String caminho = "/historico/"+nomeArquivo+".txt";

    const char * caminhoFormatado = caminho.c_str();
    const char * historicoFormatado = historico.c_str();
    escreverArquivo(LittleFS, caminhoFormatado, historicoFormatado);
    lerArquivo(LittleFS, caminhoFormatado);
}

void fazerUploadDosHistoricosOffline(){

    int qtdHistoricos = contarArquivosNaPasta(LittleFS, "/historico");
    if(qtdHistoricos > 0){
        Serial.println("Começando processo de upload de historicos registrados offline");
        WiFiClient client;
        HTTPClient http;

        for(int historicoAtual = 1; historicoAtual <= qtdHistoricos; historicoAtual++){

            String nomeDoArquivo = "/historico/" + String(historicoAtual) + ".txt";
            Serial.println("Começando upload de:" + nomeDoArquivo);
            String dadosHistorico = extrairTextoDoArquivo(nomeDoArquivo.c_str());

            http.begin(client,      "http://192.168.63.104:8080/historico-fechadura/cadastrar-
offline");
            http.addHeader("Content-Type", "application/json");

```

```

int httpResponseCode = http.POST(dadosHistorico.c_str());

String payload = "{}";

if (httpResponseCode>0) {
    payload = http.getString();
    Serial.print("Cod. de resposta HTTP: ");
    Serial.println(httpResponseCode);
    Serial.println(payload);
    if(httpResponseCode == 404){
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("ACESSO NEGADO!");
        delay(2000);
    }else{
        deletarArquivo(LittleFS,nomeDoArquivo.c_str());
    }
}
else {
    Serial.print("Houve um erro na solicitação http: ");
    Serial.println(httpResponseCode);
}
delay(200);
}
}
else{
    Serial.println("Não foi encontrado arquivos de histórico pendentes");
}
}

void cadastrarPermissoesOffline(){
    WiFiClient client;
    HTTPClient http;
    http.begin(client, "http://192.168.63.104:8080/permissoes/fechadura-lista");
    http.addHeader("Content-Type", "application/json");
    int httpResponseCode = http.POST("{\"idFechadura\":\""+ IdFechadura +"\"}");

    String payload = "{}";

    if (httpResponseCode>0) {
        payload = http.getString();
        const char* textMessage = payload.c_str();
        Serial.print("Cod. de resposta HTTP: ");
        Serial.println(httpResponseCode);
        escreverArquivo(LittleFS,"/permissoes/permissoes.txt",textMessage);
    }else{
        Serial.print("Houve um erro na solicitação http: ");
        Serial.println(httpResponseCode);
    }
}
}

```

```

void alternarConexao(){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ALTERNANDO");
    lcd.setCursor(0,1);
    lcd.print("CONEXAO !");
    delay(500);
    if(estaOnline){
        WIFI_PASSWORD = "0";
    }else{
        WIFI_PASSWORD = "SENHA";
    }
    tentarConexaoWifi();
}

```

#pragma endregion Funções do modo offline

#pragma region Funções da fechadura

```

String lerStatusFechadura(){
    if (Firebase.RTDB.getString(&fbdo, "/fechadura/"+IdFechadura+"/EstaAberta")) {
        if (fbdo.dataType() == "string") {
            statusFechadura = fbdo.stringData();
            return statusFechadura;
        }
    }
    else {
        Serial.println(fbdo.errorReason());
        return "false";
    }
}

```

```

void abrirFechadura(){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("DESBLOQUEANDO!");
    lcd.setCursor(0,1);
    lcd.print("BLOQUEIO EM 10s");
    digitalWrite(pinoFechadura, HIGH);
    delay(9000);
}

```

```

void trancarFechadura(){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("FECHANDO A");
    lcd.setCursor(0,1);
    lcd.print("FECHADURA");
    delay(1000);
}

```

```

digitalWrite(pinoFechadura, LOW);
if(estaOnline == true){
    if (Firebase.RTDB.setString(&fbdo,    "/fechadura/"+IdFechadura+"/EstaAberta",
    "false")) {
        Serial.println("Fechadura trancada!");
    }
    else {
        Serial.println("Falha ao trancar fechadura!");
        Serial.println("MOTIVO: " + fbdo.errorReason());
        trancarFechadura();
    }
}
}
}

```

#pragma endregion Funções da fechadura

#pragma region Solicitações de abertura

```

void solicitarAberturaPorSenha(){
    String senhaNumerica = lerSenhaDoTeclado();
    if(senhaNumerica == "000000"){
        Serial.println("Erro ao coletar senha.");
    }else{
        if(estaOnline == true){
            Serial.print("Senha numérica: ");
            Serial.println(senhaNumerica);
            WiFiClient client;
            HTTPClient http;
            http.begin(client,          "http://192.168.63.104:8080/permissoes/solicitar-
            entrada/senha");
            http.addHeader("Content-Type", "application/json");
            int  httpResponseCode  = http.POST("{\"idFechadura\":\""+ IdFechadura
            +"\", \"senhaNumerica\":\""+ senhaNumerica +"\"}");

```

```

    String payload = "{}";

```

```

    if (httpResponseCode>0) {
        payload = http.getString();

```

```

        if(httpResponseCode == 404){
            Serial.println("Acesso negado!");
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("ACESSO NEGADO!");
            delay(2000);
        }

```

```

        if(httpResponseCode == 200){
            Serial.println("Acesso liberado!");

```

```

        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("ACESSO LIBERADO!");
        delay(2000);
    }

    }else {
        Serial.print("Houve um erro na solicitação http: ");
        Serial.println(httpResponseCode);
    }
}

else{
    String textoDeBusca = IdFechadura + "|" + senhaNumerica;
    const char * textoDeBuscaFormatado = textoDeBusca.c_str();
    Serial.println(textoDeBuscaFormatado);
    bool acessoLiberado =
achouTextoNoArquivo(LittleFS, "/permissoes/permissoes.txt", textoDeBuscaFormatado);
    if(acessoLiberado == true){
        registrarHistoricoOffline(textoDeBusca, "Abertura liberada");
        abrirFechadura();
        trancarFechadura();
    }else{
        registrarHistoricoOffline(textoDeBusca, "Abertura bloqueada");
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("ACESSO NEGADO!");
        delay(2000);
    }
}
}
}

```

```

    }
  }
  return senhaNumerica;
}

void solicitarAberturaPorRFID(){
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("APROXIME O");
  lcd.setCursor(0, 1);
  lcd.print("CARTAO/TAG");
  delay(2000);
  if (rfid.PICC_IsNewCardPresent()) { // new tag is available
    if (rfid.PICC_ReadCardSerial()) { // NUID has been readed
      String infoCartao = lerDadosDoCartao();
      if(infoCartao != "ERROR"){
        if(estaOnline == true){
          WiFiClient client;
          HTTPClient http;
          http.begin(client, "http://192.168.63.104:8080/permissoes/solicitar-
entrada/rfid");
          http.addHeader("Content-Type", "application/json");
          int httpResponseCode =
http.POST("{\"idFechadura\":\""+IdFechadura+"\", \"infoCartao\":\""+ infoCartao + "\"}");

          String payload = "{}";

          if (httpResponseCode>0) {
            payload = http.getString();
            if(httpResponseCode == 404){
              Serial.println("Acesso negado!");
              lcd.clear();
              lcd.setCursor(0, 0);
              lcd.print("ACESSO NEGADO!");
              delay(2000);
            }
            if(httpResponseCode == 200){
              Serial.println("Acesso liberado!");
              lcd.clear();
              lcd.setCursor(0, 0);
              lcd.print("ACESSO LIBERADO!");
              delay(2000);
            }
          }
        }
        else {
          Serial.print("Houve um erro na solicitação http: ");
          Serial.println(httpResponseCode);
        }
      }
    }
  }
}

```

```

    }else{
        String textoDeBusca = IdFechadura + "|" + infoCartao;
        const char * textoDeBuscaFormatado = textoDeBusca.c_str();
        Serial.println(textoDeBuscaFormatado);
        bool                acessoLiberado                =
achouTextoNoArquivo(LittleFS, "/permissoes/permissoes.txt", textoDeBuscaFormatad
o);
        if(acessoLiberado == true){
            registrarHistoricoOffline(textoDeBusca, "Abertura liberada");
            abrirFechadura();
            trancarFechadura();
        }else{
            registrarHistoricoOffline(textoDeBusca, "Abertura bloqueada");
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("ACESSO NEGADO!");
            delay(2000);
        }
    }

    }else{
        Serial.println("Erro ao ler cartão RFID");
    }
}

//Encerra uma leitura - previne o travamento após a primeira leitura
rfid.PICC_HaltA();
rfid.PCD_StopCrypto1();
}

String lerDadosDoCartao(){
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
    byte buffer[SIZE_BUFFER] = {0};
    byte bloco = 1;
    byte tamanho = SIZE_BUFFER;
    status = rfid.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, bloco,
&key, &(rfid.uid));
    if (status != MFRC522::STATUS_OK) {
        delay(1000);
        return "ERROR";
    }
    status = rfid.MIFARE_Read(bloco, buffer, &tamanho);
    if (status != MFRC522::STATUS_OK) {
        delay(1000);
        return "ERROR";
    }
    else{
        delay(1000);

```

```

}

String infoCartao = "AAA0000";
for (int i = 0; i < MAX_SIZE_BLOCK; i++)
{
    if(i >= 1 && i < 8){
        infoCartao[i-1] = buffer[i];
    }
}
Serial.println(" ");
Serial.println("infoCartao :");
Serial.println(infoCartao);
return infoCartao;
}

void solicitarAberturaPorDigital(){
    int posicaoDigital = verificarDigital();
    if(posicaoDigital>15){
        if(posicaoDigital == 30){
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("DIGITAL NAO");
            lcd.setCursor(0, 1);
            lcd.print("CADASTRADA!");
            delay(2000);
            if(estaOnline == true){
                WiFiClient client;
                HTTPClient http;
                http.begin(client, "http://192.168.63.104:8080/permissoes/solicitar-
entrada/digital");
                http.addHeader("Content-Type", "application/json");
                int httpResponseCode = http.POST("{\"idFechadura\":\""+ IdFechadura
+"\", \"idDigital\":\""+ "DIGITAL NAO CADASTRADA" +"\"}");
                String payload = "{}";

                if (httpResponseCode>0) {
                    payload = http.getString();
                }else{
                    Serial.print("Houve um erro na solicitação http:");
                    Serial.println(httpResponseCode);
                }
            }else{
                String textoDeBuscaErro = IdFechadura +"|"+"DIGITAL NAO CADASTRADA";
                registrarHistoricoOffline(textoDeBuscaErro,"Abertura bloqueada");
            }
        }
        Serial.println("Erro ao validar digital");
    }else{
        String codigoDigital = idsDigitais[posicaoDigital];
    }
}

```

```

Serial.print("Código da Digital: ");
Serial.println(codigoDigital);
if(estaOnline == true){
    WiFiClient client;
    HTTPClient http;
    http.begin(client, "http://192.168.63.104:8080/permissoes/solicitar-
entrada/digital");
    http.addHeader("Content-Type", "application/json");
    int httpResponseCode = http.POST("{\"idFechadura\":\""+ IdFechadura
+"\", \"idDigital\":\""+ codigoDigital + "\"}");

    String payload = "{}";

    if (httpResponseCode>0) {
        payload = http.getString();

        if(httpResponseCode == 404){
            Serial.println("Acesso negado!");
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("ACESSO NEGADO!");
            delay(2000);
        }

        if(httpResponseCode == 200){
            Serial.println("Acesso liberado!");
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("ACESSO LIBERADO!");
            delay(2000);
        }
    }
    else {
        Serial.print("Houve um erro na solicitação http: ");
        Serial.println(httpResponseCode);
    }
}
else{
    String textoDeBusca = IdFechadura + "|" +codigoDigital;
    const char * textoDeBuscaFormatado = textoDeBusca.c_str();
    Serial.println(textoDeBuscaFormatado);
    bool acessoLiberado =
achouTextoNoArquivo(LittleFS, "/permissoes/permissoes.txt", textoDeBuscaFormatad
o);
    if(acessoLiberado == true){
        registrarHistoricoOffline(textoDeBusca, "Abertura liberada");
        abrirFechadura();
        trancarFechadura();
    }
    else{
        registrarHistoricoOffline(textoDeBusca, "Abertura bloqueada");
    }
}

```

```

        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("ACESSO NEGADO!");
        delay(2000);
    }

}

}

}

int verificarDigital(){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("APROXIME A");
    lcd.setCursor(0, 1);
    lcd.print("DIGITAL");

    //Espera até pegar uma imagem válida da digital
    while (fingerprintSensor.getImage() != FINGERPRINT_OK);

    //Converte a imagem para o padrão que será utilizado para verificar com o banco de
    digitais
    if (fingerprintSensor.image2Tz() != FINGERPRINT_OK)
    {
        //Se chegou aqui deu erro, então abortamos os próximos passos
        Serial.println(F("Erro image2Tz"));
        return 20;
    }

    //Procura por este padrão no banco de digitais
    if (fingerprintSensor.fingerFastSearch() != FINGERPRINT_OK)
    {
        //Se chegou aqui significa que a digital não foi encontrada
        Serial.println(F("Digital não encontrada"));
        return 30;
    }

    //Se chegou aqui a digital foi encontrada
    //Mostramos a posição onde a digital estava salva e a confiança
    //Quanto mais alta a confiança melhor
    Serial.print(F("Digital encontrada com confiança de "));
    Serial.print(fingerprintSensor.confidence);
    Serial.print(F(" na posição "));
    Serial.println(fingerprintSensor.fingerID);
    int posicaoSensor = fingerprintSensor.fingerID;
    if(posicaoSensor > 0 && posicaoSensor <= 10){
        return 0;
    }
}

```

```

}else if(posicaoSensor > 10 && posicaoSensor <= 20){
    return 1;
}else if(posicaoSensor > 20 && posicaoSensor <= 30){
    return 2;
}else if(posicaoSensor > 30 && posicaoSensor <= 40){
    return 3;
}else if(posicaoSensor > 40 && posicaoSensor <= 50){
    return 4;
}else if(posicaoSensor > 50 && posicaoSensor <= 60){
    return 5;
}else if(posicaoSensor > 60 && posicaoSensor <= 70){
    return 6;
}else if(posicaoSensor > 70 && posicaoSensor <= 80){
    return 7;
}else if(posicaoSensor > 80 && posicaoSensor <= 90){
    return 8;
}else if(posicaoSensor > 90 && posicaoSensor <= 100){
    return 9;
}else if(posicaoSensor > 100 && posicaoSensor <= 110){
    return 10;
}else if(posicaoSensor > 110 && posicaoSensor <= 120){
    return 11;
}else if(posicaoSensor > 120 && posicaoSensor <= 130){
    return 12;
}else if(posicaoSensor > 130 && posicaoSensor <= 140){
    return 13;
}else if(posicaoSensor > 140 && posicaoSensor <= 150){
    return 14;
}else if(posicaoSensor > 150 && posicaoSensor <= 160){
    return 15;
}
}
}

```

#pragma endregion Solicitações de abertura

#pragma region Funções do Menu

```

void exibirMenu(){
    if((millis() - controleExibicaoMenu > 2000 || controleExibicaoMenu == 0)){
        controleExibicaoMenu = millis();
        switch (menuStep) {
            case 0:
                lcd.clear();
                lcd.setCursor(0, 0);
                lcd.print("MASTER LOCK!");
                lcd.setCursor(0,1);
                if(estaOnline == false){
                    lcd.print("MODULO OFFLINE!");
                }else{
                    lcd.print("BEM-VINDO!");
                }
            }
        }
    }
}

```

```

    }
    menuStep++;
    break;
case 1:
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("SELECIONE A");
    lcd.setCursor(0,1);
    lcd.print("ENTRADA:");
    menuStep++;
    break;
case 2:
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("1-RFID| 2-SENHA");
    lcd.setCursor(0,1);
    lcd.print("3-DIGITAL");
    menuStep = 0;
    break;
default:
    lcd.clear();
    break;
}
}
}

```

```

void fazerLeituraDoMenu(){
    exibirMenu();
    char key = keypad.getKey();
    if (key) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("ENTRADA");
        lcd.setCursor(0,1);
        lcd.print("SOLICITADA !");
        delay(100);
        switch (key){
            case '1':
                solicitarAberturaPorRFID();
                break;
            case '2':
                solicitarAberturaPorSenha();
                break;
            case '3':
                solicitarAberturaPorDigital();
                break;
            case '#':
                cadastrarPermissoesOffline();
                break;
        }
    }
}

```

```

    case '*':
        alternarConexao();
        break;
    default:
        Serial.println(F("Opção inválida"));
        break;
}
delay(1000);
}
}

```

#pragma endregion Funções do Menu

```

void loop() {
    if(WiFi.status() == WL_CONNECTED){
        estaOnline = true;
        if(millis() - controleUploadHistorico > 60000 || controleUploadHistorico == 0){
            controleUploadHistorico = millis();
            int qtdHistoricosPendentes = contarArquivosNaPasta(LittleFS, "/historico");
            if(qtdHistoricosPendentes > 0){
                fazerUploadDosHistoricosOffline();
            }
        }
    }else{
        estaOnline = false;
    }

    if(estaOnline == true){
        //checando se a conexão com o Firebase está disponível
        if (Firebase.ready() && signupOK ) {
            //checando o status da fechadura no banco, a cada dois segundos
            if((millis() - controleLeituraPorta > 500 || controleLeituraPorta == 0)){
                controleLeituraPorta = millis();
                if(lerStatusFechadura() == "true"){
                    Serial.println("Abrindo fechadura...");
                    abrirFechadura();
                    trancarFechadura();
                }
            }
            if(millis() - controleAtualizacaoPermissao > 3600000 ||
            controleAtualizacaoPermissao == 0){
                controleAtualizacaoPermissao = millis();
                cadastrarPermissoesOffline();
            }
        }
    }else{
        if(millis() - controleReconexaoWifi > 60000 || controleReconexaoWifi == 0){
            controleReconexaoWifi = millis();
            tentarConexaoWifi();
        }
    }
}

```

```
        if(WiFi.status() == WL_CONNECTED){  
            tentarConexaoFirebase();  
        }  
    }  
}  
fazerLeituraDoMenu();  
}
```