

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
Campus DIVINÓPOLIS
GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

Matheus Costa Barbosa

CARA CRACHÁ: PROTÓTIPO DE UM SISTEMA INTEGRADO POR UMA FECHADURA
INTELIGENTE E UM SISTEMA WEB PARA VALIDAÇÃO E CONTROLE DE ACESSO

Divinópolis
2019

Matheus Costa Barbosa

CARA CRACHÁ: PROTÓTIPO DE UM SISTEMA INTEGRADO POR UMA FECHADURA INTELIGENTE E UM SISTEMA WEB PARA VALIDAÇÃO E CONTROLE DE ACESSO

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.

Áreas de integração: Computação, Eletrônica e Controle.

Orientador: Prof. Me. Marlon Henrique Teixeira
Coorientador: Prof. Me. Luís Augusto Mattos Mendes



Divinópolis
2019

Matheus Costa Barbosa

CARA CRACHÁ: PROTÓTIPO DE UM SISTEMA INTEGRADO POR UMA FECHADURA INTELIGENTE E UM SISTEMA WEB PARA VALIDAÇÃO E CONTROLE DE ACESSO

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.

Áreas de integração: coloque aqui os nomes das áreas envolvidas.

Comissão Avaliadora:

Prof. Dr. Cláudio Gomes dos Santos
Engenharia Mecatrônica
CEFET/MG Divinópolis

Prof. Me. Daniel Morais dos Reis
Informática, Gestão e Design
CEFET/MG Divinópolis

Prof. Me. Luís Augusto Mattos Mendes
Engenharia de Computação
CEFET/MG Leopoldina

Prof. Me. Marlon Henrique Teixeira
Engenharia Mecatrônica
CEFET/MG Divinópolis

Divinópolis

2019

DEDICO ESSE TRABALHO A DEUS,
QUE SEMPRE TEM OLHADO POR
MIM, À MINHA FAMÍLIA, EM ESPE-
CIAL: MEUS PAIS REJANE E RO-
BERTO E IRMÃO MARCUS VINÍCIUS
E À TODOS OS MEUS AMIGOS QUE
ME ACOMPANHARAM NESSA JOR-
NADA.

Agradecimentos

Agradeço,

A Deus, que tem me dado saúde e me coberto com suas bênçãos.

A meus pais que sempre me apoiaram e se esforçaram incondicionalmente para que eu chegasse até aqui.

Ao meu irmão Marcus Vínicus que com sua amizade e inteligência sempre fez com que o caminho parecesse mais simples.

Ao CEFET-MG que permitiu que eu tivesse uma formação de qualidade e contribuiu diretamente para meu crescimento pessoal e profissional.

A todos os amigos que mesmo distantes foram refúgio para as dificuldades da rotina.

As pessoas que são loucas o suficiente para achar que podem mudar o mundo são as que, de fato, o mudam.

Steve Jobs

Resumo

O projeto proposto visa o desenvolvimento de um sistema inteligente para fechaduras, englobando três das grandes áreas da engenharia mecatrônica: computação, controle e eletrônica. Tem-se como objetivo promover maior segurança aos ambientes em que se instale o dispositivo, seja em meio residencial ou corporativo. A investigação de soluções que visam o aumento da segurança é algo pesquisado há anos, em todo o mundo. Entretanto as soluções geradas são pouco robustas. Os sistemas de segurança cotidianamente empregados envolvem a imposição de barreiras físicas como muros elevados e utilização de concertinas. Apesar do avanço tecnológico atual, poucos modelos de monitoramento fornecem informações precisas acerca do controle de acesso de maneira eficaz. O sistema será composto por leitor e *tag's RFID*, sensor de fim de curso, câmera, microprocessador e circuitos eletrônicos para alimentação adequada de todo o conjunto. A imagem adquirida será tratada utilizando técnicas advindas da teoria de processamento de sinais como, por exemplo, a aplicação de filtros, tendo como objetivo estabelecer o reconhecimento facial com precisão. Além disso, o protótipo apresentará um sistema de comunicação sem fio, o qual estará montado junto aos demais componentes, em uma estrutura compacta, projetada segundo as necessidades de resistência mecânica, que virá a ser acoplada nas portas. Na sequência, um software responsivo será desenvolvido para promover a interação do sistema com o administrador. Esta plataforma se comunicará e receberá os dados provenientes do sistema embarcado na porta, via comunicação *wireless* e fornecerá informações sobre acessos, intrusões e possibilitará ainda a inclusão de novos usuários. Espera-se, com o desenvolvimento desse projeto, contribuir para maior segurança nos ambientes em que forem instalados, promover maior possibilidade de investigação em caso de intrusões e promover uma experiência mais efetiva no que se refere a dispositivos de segurança tradicionalmente empregados. Além disso, busca-se a aplicação de conceitos envolvidos nas novas tendências da automação.

Palavras-chave: Automação; *IoT*; Segurança residencial; Fechadura inteligente.

Abstract

The proposed project intends the development of an intelligent door lock system, encompassing three of the mechatronic engineering big areas: computation, control and electronics. The goal is to promote more security on environments where the device is installed, should they be residential or corporate. Investigations on security improvement solutions has been researched for many year, all around the world. The solutions found so far could be more robust, though. The security systems usually employed are based on physical barriers such as tall walls and razor wires. Currently, very little monitoring systems provides accurate information about access control in an effective fashion. The system will be composed by RFID readers and tags, a micro switch, a camera, a microcomputer and an electronic circuit to properly supply energy to the whole set. The acquired image will be treated using techniques that come from signal processing theory, such as, filter application, with the goal of establishing facial recognition with precision. Besides that, the prototype will have a wireless communication system ensembled with the remaining components, in a compact structure, designed according to the mechanical resistance needs that will be coupled on the doors. As the next step a resposive software will be developed to promote the system interaction with its administrator. This platform will communicate and get the data from the embedded system on the door, via wireless communication and provide information about access, intrusions and will allow new users registration. It is expected to contribute for security improvement on environments where such a product will be installed and promote more effective investigations on intrusions cases and an enhanced experience when it comes to traditional security devices. Besides that, it is attempted to apply new automation's trends concepts.

Key-words: Automation; IoT; Residential security; Smart locker.

Sumário

Lista de Figuras	xiii
Lista de Tabelas	xiv
Lista de Acrônimos e Notação	xvi
1 Introdução	1
1.1 Definição do Problema	2
1.2 Motivação	2
1.3 Objetivo Geral	3
1.4 Objetivos Específicos	3
1.5 Metodologia	3
1.5.1 Recursos utilizados	4
1.6 Organização do Documento	5
2 Revisão da literatura	7
2.1 Sistemas embarcados	11
2.2 <i>Radio-Frequency Identification (RFID)</i>	12
2.3 <i>Unified Modeling Language - UML</i>	14
2.4 <i>Amazon Web Services (AWS)</i>	15
2.5 <i>Web Services</i>	15
2.6 Banco de Dados	16
2.7 Visão computacional	17
2.7.1 <i>OpenCV</i>	18
2.8 Tecnologias de front end	25
2.9 <i>Model-View-Controller</i>	26
2.9.1 <i>Model</i>	27
2.9.2 <i>View</i>	27

2.9.3	<i>Controller</i>	27
3	Desenvolvimento	28
3.1	Projeto Conceitual	28
3.1.1	Análise da Central de Processamento	29
3.1.2	Análise dos periféricos	32
3.1.3	Materiais e Orçamento	35
3.2	Transição do algoritmo para o sistema embarcado	36
3.3	Projeto do Software	37
3.4	Projeto do <i>Firmware</i>	41
4	Resultados	43
4.1	Construção do dispositivo	43
4.2	Reconhecimento facial	46
4.3	Implementação do firmware	47
4.4	Implementação do back end	48
4.5	Implementação do front end	52
5	Considerações Finais	62
5.1	Propostas de continuidade	63
	Referências	64

Lista de Figuras

2.1	Primeiras fechaduras egípcias [8].	7
2.2	<i>Conexis LI Smart Lock</i> [11].	9
2.3	Sistema de Biometria Facial[17].	10
2.4	Diagrama genérico de um sistema embarcado [18]. O	11
2.5	Foto <i>Raspberry PI 3 - Model B</i> [20].	12
2.6	Modelo básico de um sistema RFID. Adaptado [22].	13
2.7	Módulo RFID MFRC522 Mifare [24].	13
2.8	Esquema de uso de Web Services pela Apple [30].	16
2.9	a) Máscara genérica com a posição dos valores utilizados nas equações de <i>Prewitt</i> e <i>Sobel</i> . b) Máscaras dos operadores de <i>Prewitt</i> , c) Máscaras dos operadores de <i>Sobel</i> [41].	20
2.10	Segmentação de imagem utilizando a técnica de corte.a) Imagem em nível de cinza b) Histograma com o ponto de corte e c) Imagem binarizada [41].	21
2.11	Demonstração da atuação do <i>Background Subtraction</i> [40].	22
2.12	<i>Diferenças entre o processamento de imagens e a visão computacional</i> [41].	23
2.13	Diagrama do processamento de imagens.Adaptado [46].	24
2.14	Padrão <i>Model View Controller</i> [55].	27
3.1	Pinout obtido no terminal. Fonte: Elaborada pelo autor.	31
3.2	Pinout com descrição das comunicações[58].	31
3.3	Ligação com o Módulo RFID. Fonte: Elaborada pelo autor.	33
3.4	Ligação com o Módulo RFID. Fonte: Elaborada pelo autor.	34
3.5	Câmera utilizada na primeira fase de desenvolvimento. Fonte: Elaborada pelo autor.	35
3.6	Acesso remoto utilizando o <i>VNC Viewer</i> . Fonte: Elaborada pelo autor. . .	36
3.7	Modelagem UML de casos de uso. Fonte: Elaborada pelo autor.	38
3.8	Modelo DER do banco de dados. Fonte: Elaborada pelo autor.	39

3.9	Modelo lógico do banco de dados. Fonte: Elaborada pelo autor.	40
3.10	Diagrama de Implantação. Fonte: Elaborada pelo autor.	41
3.11	Diagrama de Máquina de Estados do firmware. Fonte: Elaborada pelo autor.	42
4.1	Fim de curso com haste e rolete. Fonte: Elaborada pelo autor.	43
4.2	Módulo relé utilizado. Fonte: Elaborada pelo autor.	44
4.3	Montagem final. Fonte: Elaborada pelo autor.	44
4.4	Parte inferior da <i>case</i> . Fonte: Elaborada pelo autor.	45
4.5	Tampa da <i>case</i> . Fonte: Elaborada pelo autor.	45
4.6	Vista frontal. Fonte: Elaborada pelo autor.	45
4.7	Parte interna da <i>case</i> . Fonte: Elaborada pelo autor.	46
4.8	<i>Case</i> concluída. Fonte: Elaborada pelo autor.	46
4.9	Detecção facial utilizando <i>HaarCascades</i> . Fonte: Elaborada pelo autor. . .	47
4.10	Resumo da implantação do banco de dados. Fonte: Elaborada pelo autor. .	49
4.11	Resumo da implantação do banco de dados. Fonte: Elaborada pelo autor. .	50
4.12	Máquina virtual provisionada para atuar como servidor. Fonte: Elaborada pelo autor.	51
4.13	Servidor configurado para rodar a aplicação. Fonte: Elaborada pelo autor.	52
4.14	Tela inicial. Fonte: Elaborada pelo autor.	53
4.15	Tela de registro. Fonte: Elaborada pelo autor.	53
4.16	Tela de login. Fonte: Elaborada pelo autor.	54
4.17	Dashboard após o login e menu lateral. Fonte: Elaborada pelo autor. . . .	54
4.18	Tela de perfil do usuário. Fonte: Elaborada pelo autor.	55
4.19	Popup para alteração de senhas. Fonte: Elaborada pelo autor.	55
4.20	Tela de cadastros: usuário. Fonte: Elaborada pelo autor.	55
4.21	Tela de cadastros: endereços. Fonte: Elaborada pelo autor.	56
4.22	Tela de cadastros: usuários. Fonte: Elaborada pelo autor.	56
4.23	Tela de listagem de endereços do usuário. Fonte: Elaborada pelo autor. . .	57
4.24	Tela para edição de endereços. Fonte: Elaborada pelo autor.	57
4.25	Confirmação da exclusão de um endereço. Fonte: Elaborada pelo autor. . .	57
4.26	Tela de listagem de fechaduras do usuário. Fonte: Elaborada pelo autor. .	58
4.27	Tela para adição de vínculo de usuário. Fonte: Elaborada pelo autor. . . .	58
4.28	Tela para exclusão de vínculo de usuário. Fonte: Elaborada pelo autor. . .	58
4.29	Tela para edição de fechadura. Fonte: Elaborada pelo autor.	59
4.30	Tela de confirmação de exclusão de fechadura. Fonte: Elaborada pelo autor.	59
4.31	Tela de listagem de acessos. Fonte: Elaborada pelo autor.	59
4.32	Tela de listagem de invasões. Fonte: Elaborada pelo autor.	60
4.33	<i>Date Picker</i> utilizado . Fonte: Elaborada pelo autor.	60

4.34	Tela de cadastro de solicitações de suporte. Fonte: Elaborada pelo autor. .	60
4.35	Alerta de confirmação de ação. Fonte: Elaborada pelo autor.	61

Lista de Tabelas

1.1	Descrição das tarefas	4
1.2	Cronograma de trabalho	5
3.1	Comparativo - <i>Raspberry PI 3</i> e <i>Beagle Black Bone</i>	30
3.2	Ligação módulo RFID e processador	32
3.3	Orçamento do projeto	35

Lista de Acrônimos e Notação

AWS	Amazon Web Services
AWS RDS	Amazon Relational Database Service (RDS)
AWS S3	Amazon Simple Storage Service
BD	Banco de dados
BS	Background Subtraction
CSS	Cascading Style Sheets
ER	Entidade-Relacionamento
GPIO	General Purpose Input/Output
HD	Hard Disk
HDMI	High-Definition Multimedia Interface
HTML	Hypertext Markup Language
HTTPS	Hyper Text Transfer Protocol Secure
I2C	Inter-Integrated Circuit
MCU	Microcontroller Unit
MPU	Microprocessor Unit
OpenCV	Open Source Computacional Vision
PCA	Análise de Componentes Principais
RFID	Radio-Frequency Identification
SD	Secure Digital Card
SGBD	Sistema de Gerência de Banco de Dados
SPI	Serial Peripheral Interface
SSL	Secure Sockets Layer
UART	Universal asynchronous receiver/transmitter
UID	Unique Identifier
UML	Unified Modeling Language
USB	Universal Serial Bus
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Introdução

A busca por segurança é um objeto de desejo do ser humano em diferentes âmbitos de compreensão. A definição mais genérica de segurança refere-se à um estado de percepção de proteção, de estar livre de riscos [1]. Um dos mecanismos mais utilizados afim de garantir proteção são as fechaduras.

A sensação de insegurança tem sido algo corriqueiro na vida do brasileiro. Segundo o IBGE [2], por meio do Suplemento de Vitimização e Justiça da Pesquisa Nacional por Amostra de Domicílios (Pnad), 11,9 milhões de brasileiros foram vítimas de roubo em 2009. Iniciativas como o site "Onde fui roubado?"¹ [3] tem crescido. De acordo com o próprio site mais de 1,3 milhões de brasileiros já utilizam o serviço. No que se refere às residências não tem sido diferente. Segundo o jornal O TEMPO [4] o arrombamento de residências cresceu 66% no ano de 2016 em Belo Horizonte.

Tendo todo o quadro acima em consideração, a motivação para este projeto parte da oportunidade de conciliação da tecnologia de comunicação e monitoramento com a internet das coisas, para uma solução prática. Propõe-se, desse modo, um dispositivo inteligente que seja responsável pelo controle de acesso. Esse dispositivo será projetado na perspectiva de substituir as fechaduras convencionais, sendo acoplados em portas e apropriar-se-á de um sistema *web*, que possibilitará o registro e a emissão de relatórios sobre os acessos. Na interface com os usuários serão empregadas as tecnologias de *RFID (Radio-Frequency Identification)* e reconhecimento facial, por meio de uma câmera. A interação entre os

¹Maiores informações em: <http://blog.ondefuirobado.com.br/>

usuários e a fechadura será conferida por meio de *software*. Este se comunicará via *wireless* com a fechadura inteligente, efetuando o cadastro de novos usuários e recebendo registros de acessos, além de notificações em caso de eventuais tentativas de intrusões.

A tecnologia supracitada já possui versões comerciais como a fornecida pela empresa *FIRS Technologies (Shenzhen)* [5]. A necessidade de importação eleva o custo do produto. Produtos similares de outras fabricantes podem ser encontrados em sites populares de varejo. Segundo o site Submarino [6], o custo para a aquisição de um dispositivo similar, em outubro de 2019, é de aproximadamente R\$3.990,00 (três mil novecentos e noventa reais). O presente trabalho visa o desenvolvimento de uma fechadura mais robusta, com foco na eficiência deste produto. Além disso, por se tratar de um sistema brasileiro, a interface homem-máquina² será mais amigável com pessoas que possuem o português como língua materna.

Sendo assim, o trabalho integrou três áreas da engenharia mecatrônica, sendo elas: eletrônica, no desenvolvimento do circuito de acionamento e alimentação da fechadura; computação, no desenvolvimento de toda a interface virtual entre o *software*, algoritmo embarcado e *hardware*; controle, no que tange ao processamento de imagens.

1.1 Definição do Problema

Os problemas abordados neste trabalho são as limitações atualmente existentes nas fechaduras convencionais, além da falta de informações acerca dos acessos e de imagens, em caso de intrusões.

1.2 Motivação

O tema do trabalho surge da percepção das demandas relacionadas à segurança residencial no Brasil. Um dos problemas verificados foi a dificuldade de acesso a equipamentos neste segmento, além da complexidade das opções oferecidas. Fechaduras sofisticadas, alarmes e câmeras de monitoramento tem elevado custo, além da necessidade de implan-

²Interação entre o usuário e o dispositivo

tação de um sistema auxiliar, para supervisão dos demais. Devido às especificidade do mercado a robustez é um requisito importante, sendo um pré requisito o acesso à internet.

1.3 Objetivo Geral

Desenvolver um protótipo de sistema inteligente de fechaduras, capaz de estabelecer o controle de acessos, monitoramento de tentativas de intrusões e registro de passagem utilizando reconhecimento facial e *RFID*, e, também, de ser acessado remotamente por seu administrador, utilizando comunicação por meio de um *software*.

1.4 Objetivos Específicos

- Projetar e construir uma estrutura para acondicionamento do *hardware*;
- Projetar e construir o circuito eletrônico para acionamento da fechadura;
- Elaborar o algoritmo para comunicação entre o sistema e o *software*.
- Implementar o algoritmo de reconhecimento facial;
- Modelar e implementar o banco de dados em um servidor *web* para registro;
- Desenvolver os sistemas de comunicação *wireless*;
- Desenvolver o sistema de reconhecimento de *RFID*;
- Desenvolver um *software web* com interface amigável para permitir o acesso aos registros e configurações por parte do usuário;

1.5 Metodologia

Visando a simplificação e organização, esta seção será dividida em subseções, uma vez que o desenvolvimento do projeto em questão integra diferentes áreas da engenharia mecatrônica e necessitou do desenvolvimento de diversos componentes. A descrição

das atividades que foram desenvolvidas é apresentada na Tabela 1.1 e o cronograma é apresentado na Tabela 1.2.

Inicialmente realizou-se o levantamento e o estudo bibliográfico acerca das tecnologias envolvidas. Foram utilizadas diversas ferramentas de conexão remota como *RFID* e *Wi-Fi*. É necessário que se avalie possíveis interferências entre elas, além dos esforços mecânicos envolvidos. Neste estágio do trabalho serão revisados artigos científicos e livros relacionados às técnicas que serão futuramente utilizadas.

Tabela 1.1: Descrição das tarefas

Item	Descrição
1	Revisão bibliográfica
2	Estabelecimento de requisitos e levantamento de componentes do projeto
3	Modelagem do Banco de Dados
4	Elaboração do Diagrama UML do sistema
5	Estudo das técnicas de reconhecimento facial
6	Implementação da comunicação da central com o módulo RFID
7	Elaboração do TCC1
8	Revisão do TCC1
9	Estudo das ferramentas de hospedagem disponíveis
10	Implantação do Banco de dados
11	Configuração da infraestrutura para hospedagem
12	Implementação do back end contendo as API's
13	Implementação da comunicação entre a fechadura e o back end
14	Implementação do front end consumindo as API's
15	Implementação de todo o código embarcado
16	Construção da estrutura física de acomodação do dispositivo
17	Elaboração do TCC2

1.5.1 Recursos utilizados

- IDE Back end: Visual Studio 2019 Community
- Back end: Web API Application - .NET Framework 4.5
- ORM (Object Relational Mapper): Entity Framework
- Gerenciador de pacotes do front end: Nodejs
- Framework de construção de interface de usuários: Reactjs

Tabela 1.2: Cronograma de trabalho

Etapas	2018				2019			
	Março	Abril	Maió	Junho	Julho	Agosto	Setembro	Outubro
1	x	x						
2		x	x					
3				x				
4				x				
5			x	x				
6			x					
7								
8					x			
9					x	x		
10					x			
11					x	x		
12						x		
13						x		
14							x	x
15							x	x
16								x
17							x	x

- Design System: Material Design
- Https Client: Axios
- Banco de dados: SQL Server
- Armazenamento de arquivos: Armazenamento de Blobs - Microsoft Azure
- Hospedagem da Aplicação Estática: netlify.com
- Hospedagem Banco de Dados: Microsoft Azure
- Hospedagem da aplicação dinâmica: Microsoft Azure (buscando alternativa gratuita)
- Prototipagem de Telas: Adobe XD

1.6 Organização do Documento

O presente documento está dividido em 5 capítulos, acrescido de referências. O primeiro capítulo refere-se à contextualização do projeto.

O segundo capítulo trata da Revisão da literatura. O desenvolvimento será abordado no capítulo 3. O capítulo 4 é composto pelos resultados, sendo seguido pelas considerações finais, no capítulo 5.

Revisão da literatura

Estima-se que a história das fechaduras de tambor de pinos tenha se iniciado no Egito, há 4000 anos. O primeiro registro histórico de um modelo precursor (Figura 2.1) foi retirado das ruínas do palácio sírio de *Khorasabad*, situado na cidade bíblica de Nínive e remonta a cerca de 2700 anos atrás [7]. Este modelo era construído em madeira e apresentava diversos problemas devido a isto: trata-se de um material muito susceptível a forças externas, além do elevado peso das chaves.

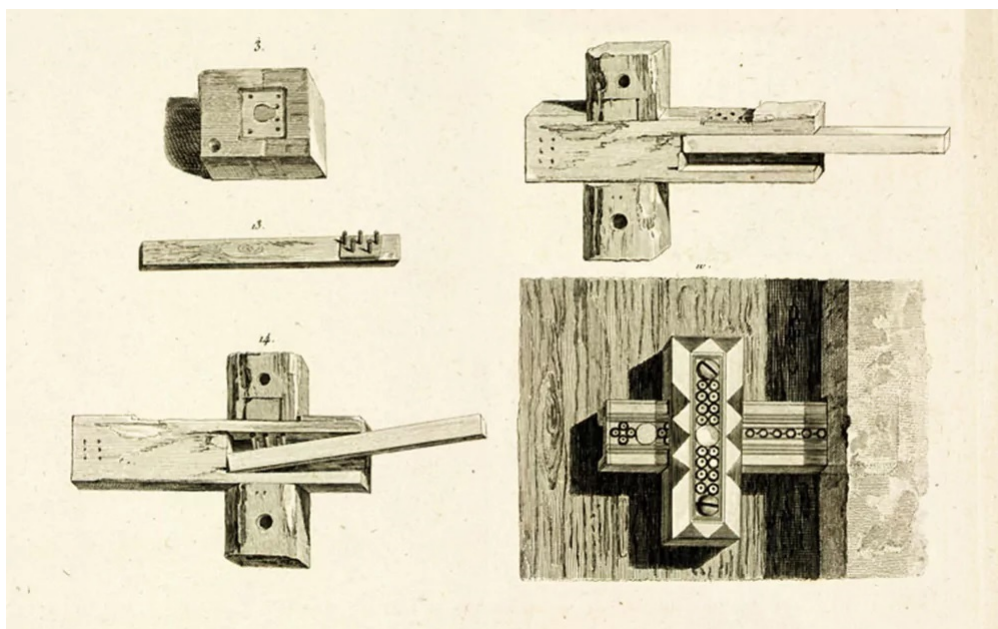


Figura 2.1: Primeiras fechaduras egípcias [8].

Os romanos desenvolveram e popularizaram o mecanismo, substituindo a madeira por ferro e criando chaves de bronze. Além disso o mecanismo foi melhorado, tendo as

projeções internas protegidas, sendo inacessível externamente. Essas mudanças conferiram ao sistema mais resistência e o tornaram menor, possibilitando que uma pessoa comum pudesse carregar as chaves [9].

O produto se manteve inalterado por vários séculos[9]. Um bom exemplo disso é que no século *XV*, momento de grande desenvolvimento dos países ibéricos, os chaveiros destas nações, responsáveis por desenvolver dispositivos mais seguros, concentravam seus esforços em produzir mais cadeados colocando um cofre menor dentro de outro maior e assim sucessivamente, tornando apenas o processo de furto e roubo mais cansativo.

Inovações neste ramo voltaram a ser notáveis somente em 1778, quando Robert Barron patenteou uma fechadura de tambor com dois cilindros. Invenção melhorada em 1818 por Jeremiah Chubb, que incorporou uma mola ao mecanismo [8].

Joseph Bramah patenteou, em 1784, um sistema que empregava uma chave cilíndrica. Esta versão ficou famosa pelo desafio proposto pelo inventor, no qual oferecia uma recompensa para quem conseguisse abri-la sem a chave: ninguém o fez por 50 anos [8].

No século *XIX* chegou-se ao modelo atual, patenteada por Linus Yale. A versão foi melhorada por ser filho, Linus Yale Jr. sendo registrada em 1865 [7].

Depois dessa inovação várias alterações foram empregadas, porém apenas como incremento às diversas combinações possíveis dos modelos supracitados. Dentre as tecnologias hoje empregadas destacam-se o uso de chaves magnéticas, chaves flexíveis¹ e também as chamadas *smart-locks*. Dentre estas destaca-se o modelo desenvolvido pela *Yale* em conjunto com a *Everest* (Figura 2.2) chamada de *Conexis LI SmartLock* [11], foi lançada em 2018 e todo o seu controle é feito via *smartphone* dispondo de chaves virtuais que podem ser, inclusive, recebidas e enviadas.

¹Patenteadas em 1992 por Yun-Tung Hsu, entretanto não se popularizaram [10]



Figura 2.2: *Conexis LI Smart Lock* [11].

Atualmente, diversas pesquisas estão sendo realizadas a respeito de sistemas de controle de acesso e biometria facial. As principais investigações se dão no aspecto de padronizar e otimizar sistemas de reconhecimento facial, além da produção de fechaduras inteligentes, possuindo várias maneiras de conexão wireless.

Em Timse et al.[12] é possível constatar o desenvolvimento de uma fechadura com reconhecimento facial desenvolvida em C#. Pode-se verificar a ausência da preocupação de embarcar a solução, o que torna necessário que, para o funcionamento do sistema, haja sempre conexão com a internet, limitando totalmente o desempenho em caso de falta de energia.

Já em Nascimento [13] é apresentada uma solução embarcada de um sistema dotado de reconhecimento facial. Os objetivos apresentados, no que tangem ao algoritmo de detecção e reconhecimento facial, são semelhantes. Entretanto não foram apresentadas aplicações práticas. As técnicas apresentadas, principalmente no que se refere à determinação do dispositivo empregado, sistema operacional e biblioteca utilizada são apresentadas e se encaixam no desenvolvimento do presente trabalho.

Tramontin [14] descreve um sistema semelhante ao anterior, diferindo na implementação, utilizando um microprocessador *Raspberry Pi*. Além disso, é descrito o método de Análise de Componentes Principais. Trata-se de um método estatístico e algébrico, com complexidade elevada em relação ao método tradicional de verificação, elevando o custo computacional do dispositivo.

A implementação do sistema utilizando o método PCA ² é contemplada por Gunawan et. al. [15], com o foco dedicado ao aprimoramento de um sistema de segurança de uma *Smart Home* (casa que possui produtos conectados em rede, podendo ser controlados remotamente). Além do reconhecimento facial são implementadas as funções de detecção de presença, dispondo de um sensor ultrassônico. Outro aspecto relevante observado neste trabalho é a notificação dos resultados das verificações por meio de *SMS*, estratégia não verificada nos artigos anteriores.

Analisando as opções disponíveis de mercado visualiza-se uma grande oferta de sistemas com características semelhantes às objetivadas neste trabalho. A empresa Fxbiometria [16] disponibiliza um sistema *plug and play* possuindo alimentação utilizando pilhas e câmera com visão noturna, porém sem registro em banco de dados. A empresa Madis [17] fornece o modelo apresentado na Figura 2.3. Este modelo é dotado de registro, oferecendo funcionalidades como relógio de ponto, porém com custo mais elevado.



Figura 2.3: Sistema de Biometria Facial[17].

Uma vez apresentado um breve histórico acerca dos recentes avanços na área serão apresentados abaixo os conceitos importantes no desenvolvimento do trabalho, iniciando-se com a parte computacional a ser discutida.

²Análise de Componentes Principais

2.1 Sistemas embarcados

Atualmente estima-se que mais de 90% dos microprocessadores fabricados não são direcionados a fabricação de dispositivos chamados de computadores. Dentre os aparelhos que dispõem desse tipo de mecanismo se destacam celulares, automóveis e eletrodomésticos em geral. A grande diferença destes eletrônicos para os chamados computadores se refere à especificidade do projeto. Define-se sistema embarcado um conjunto dedicado e especialista constituído por *Hardware*, *Software* e Periféricos [18].

Alguns elementos são necessários para a construção de um dispositivo embarcado, dentre eles destaca-se o "cérebro" do sistema, sendo responsável por todo o gerenciamento. As opções ideais para este tipo de tarefa são microcontroladores ou microprocessadores devido à alta capacidade de interagir com sinais externos, executar programas e gerenciar periféricos como atuadores e sensores. A Figura 2.4 apresenta um diagrama genérico básico de um sistema embarcado.

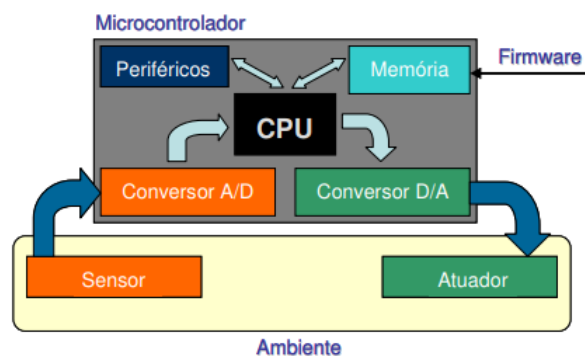


Figura 2.4: Diagrama genérico de um sistema embarcado [18]. O

Uma unidade central de processamento cada vez mais popular é o Raspberry PI 3 (Figura 2.5). Trata-se da terceira versão de um microcomputador desenvolvido pela *Raspberry PI Foundation* e tem como principal função tornar fácil o acesso ao desenvolvimento de maneira barata e robusta [19].



Figura 2.5: Foto *Raspberry PI 3 - Model B* [20].

Os objetivos são alcançados principalmente pela simplicidade do sistema, possuindo o tamanho de um cartão de crédito, conta com um chip de arquitetura ARM responsável pelo processamento, memória RAM e desempenho gráfico. O microprocessador é fabricado pela *Broadcom*[20], empresa responsável por fornecer o componente para diversas empresas de *smartphones* e *tablets*.

As especificações do *hardware* como a memória RAM, disponibilidade de portas e detalhes do conjunto de *GPIO's* serão apresentado mais a frente.

2.2 *Radio-Frequency Identification (RFID)*

O *RFID*, traduzindo ao português, Identificação por Radio-Frequência, é um novo método de investigação que vem sendo aplicado largamente em automações, de uma maneira geral [21].

A origem destas identificações remonta à segunda guerra mundial (1935) na qual o físico escocês, Robert Alexander Watson-Watt, formulou o sistema (IFF - *Identify Friend or Foe*). Na época, todos os aviões britânicos receberam um transmissor que ao receber sinais das estações de radar começavam a transmitir um sinal de resposta. O reconhecimento atual funciona pelo mesmo princípio: uma etiqueta eletrônica recebe um sinal e reflete o sinal de volta (passivo) ou transmite um novo sinal (ativos) [22].

A primeira aplicação comercial do *RFID* ocorreu em sistemas antifurto, identificando

itens ao passar por um portal e verificando se os produtos foram adquiridos ou resultados de roubos.

O modelo básico de funcionamento (Figura 2.6) é composto por dois componentes básicos. O primeiro é o *transponder*: popularmente conhecido como *tag* é a etiqueta que pode ser lida. O segundo é o leitor, que varia de acordo com a funcionalidade desejada.

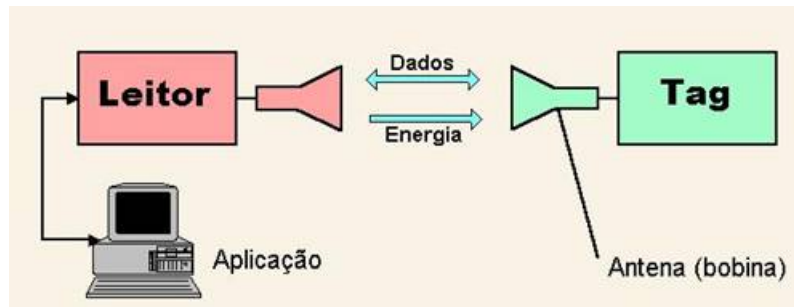


Figura 2.6: Modelo básico de um sistema RFID. Adaptado [22].

Ao realizar-se uma pesquisa acerca dos módulos de leitura disponíveis comercialmente fica claro que o mais difundido é o MFRC522. Ao analisar suas características verifica-se que o mesmo é adequado a maior parte das aplicações, seja ela comercial ou *do it yourself*.

O módulo possui alimentação de 3,3 V e suporte para leitura na frequência de 13,56MHz. Além disso possui vários tipos de comunicação como I2C, SPI e serial UART [23]. A Figura 2.7 apresenta uma versão comercial do produto, já com *tags* e barras de pinos inclusas, para uso facilitado.

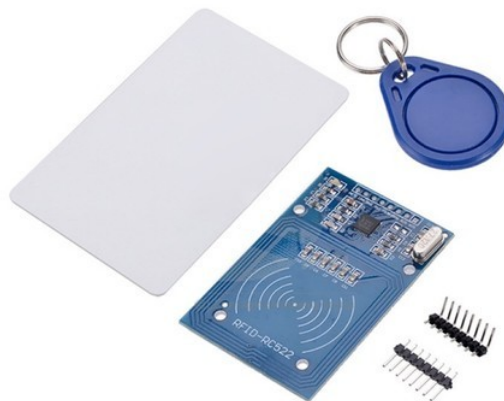


Figura 2.7: Módulo RFID MFRC522 Mifare [24].

2.3 *Unified Modeling Language - UML*

A linguagem de modelagem unificada é uma linguagem visual utilizada para a modelagem de *softwares*. É baseada no paradigma de orientação a objetos e tem sido largamente empregada nos últimos anos. A última versão de especificações (2.5.1) foi lançada em dezembro de 2017.

O papel do modelo obtido é auxiliar engenheiros de *software* a definir quais requisitos são necessários ao sistema, qual sua estrutura lógica, comportamentos e até necessidades físicas do *hardware* no qual será instalado[25].

Para o presente projeto serão utilizados os diagramas de Casos de Uso, Implantação e Máquina de Estados, detalhados abaixo.

O diagrama de casos de uso é o diagrama mais abstrato na linguagem UML, sendo utilizado no início da modelagem do sistema, nas fases de levantamento e análise de requisitos. Esta representação tem como objetivo apresentar uma visão externa de quais funcionalidades devem ser fornecidas aos usuários. É caracterizada pelo uso de atores (qualquer *hardware* especial ou usuário que vai interagir com o *software*) e casos de uso (tarefas à serem compreendidas no desenvolvimento). Outros elementos podem ser adicionados, como *includes*, *extends* e generalizações [25].

Já o diagrama de implantação é a representação mais física da UML, tendo total enfoque na organização da arquitetura material sobre a qual o sistema será implantado e executado em termos de hardware. Também são representadas as formas de conexão e os protocolos de comunicação entre os dispositivos envolvidos. Os elementos mais comuns em diagramas deste tipo são os nós, que representam máquinas e servidores empregados [25].

Por fim, o diagrama de máquina de estados tem como foco a representação do comportamento de um elemento por meio de um conjunto finito de transições. Se comparado à diagramas externos à UML pode ser correlacionado à um fluxograma. Esta exibição é composta por estados (a situação em que um elemento se encontra num determinado momento) e transições (eventos que resultam em mudanças de estado) [25].

2.4 Amazon Web Services (AWS)

A Amazon Web Services é uma plataforma de nuvem utilizada largamente em diversas aplicações com diferentes fins. Suas soluções são muito empregadas em grandes empresas como *Netflix*, *Twitch* e *LinkedIn* [26].

Na maioria das aplicações *web* o armazenamento das informações é realizado utilizando um banco de dados. O *Amazon Relational Database Service* é o serviço responsável por facilitar a operação de bancos de dados relacionais na nuvem. Este serviço será utilizado para a configuração do banco de dados da aplicação. Além disso todo o provisionamento de hardware, manutenção e backups são feitos automaticamente. É oferecido suporte para várias bases de dados diferentes como *MySQL*, *SQLServer*, *Oracle Database* e *MariaDB* [27].

A persistência de arquivos é realizada de maneira independente. No contexto do AWS o serviço correspondente é o S3 (*Simple Storage Service*). Como o nome deixa claro, é um serviço de armazenamento de arquivos e será utilizado para salvamento dos vídeos de invasão e as fotografias de usuários. Tratando-se de uma aplicação na nuvem se caracteriza pela alta disponibilidade e grande compatibilidade com diferentes casos de uso como sites, aplicativos para dispositivos móveis, dispositivos *IoT* e *Big Data* [28].

2.5 Web Services

A internet é um ambiente cada vez mais heterogêneo possuindo diversas tecnologias e equipamentos de diferentes fabricantes. Integrar tantas técnicas distintas se mostrou um grande desafio, solucionado atualmente por meio do emprego de *Web Services* (Figura 2.8)[29].

A grande vantagem deste tipo de implementação é a independência de plataforma. O uso de protocolos gerais é fundamental para agregar essa característica. O mesmo será empregado para a implementação do *backend*. Os principais padrões utilizados se referem ao protocolo de transferência de dados (HTTPS) e ao formato da mensagem (XML) [31].

A Linguagem Extensível de Marcação, mais conhecida como XML é uma linguagem

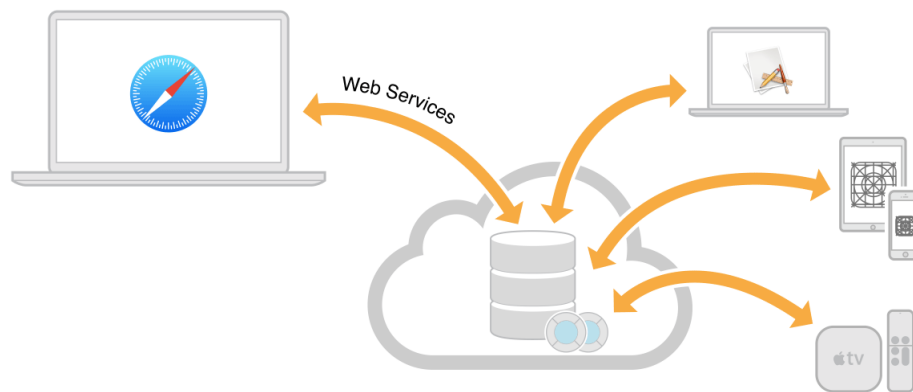


Figura 2.8: Esquema de uso de Web Services pela Apple [30].

designada à descrição e estruturação de informações. Similar ao HTML a mesma possui marcadores específicos para descrição dos dados. A grande diferença trata da ausência de pré-definição dos marcadores permitidos. A estrutura é uma recomendação da W3C (*World Wide Web Consortium*) desde 1998 [29].

Já o Protocolo Seguro de Transferência de Hipertexto é um aperfeiçoamento do protocolo de comunicação base da Internet, HTTP. O objetivo do protocolo é permitir uma transferência de arquivos, no formato HTML, entre um navegador (o cliente) e um servidor Web de maneira criptografada. A principal diferença é a melhoria da construção do mesmo, sendo construído sobre uma camada SSL (*Secure Sockets Layer*), permitindo que a transmissão seja criptografada e autenticada através de certificados digitais[31].

2.6 Banco de Dados

Os sistemas de gerência de banco de dados surgiram no início da década de 1970 com o objetivo de facilitar a programação de aplicações de bancos de dados[32]. Um SGBD é definido como o "*software* que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados"[32].

Ao se tratar da modelagem de BD relacional, inicialmente é necessário descrever a abordagem Entidade-Relacionamento. Técnica criada em 1976 por Peter Chen, podendo ser considerada o padrão de modelagem mais utilizado, dando subsídio aos tópicos relacionados abaixo[32].

Um modelo ER é representado graficamente através de um Diagrama Entidade-Relacionamento (DER).

Existem diversas soluções mercadológicas para a administração dos recursos. Neste trabalho será empregada o SQL Server uma das partes centrais da plataforma de dados da Microsoft, sendo líder do setor ODBMS (Sistemas de Gerenciamento de Bancos de Dados Operacionais) [33].

O uso do mesmo é atrativo principalmente devido ao acesso controlado e grande capacidade de processamento para atender aos requisitos de consumo de informações. Além disso é facilmente escalável [34], sustentando alta disponibilidade de fluxo.

Para mapear os objetos do banco de dados na programação será utilizado o Entity Framework (EF), um mapeador relacional de objetos desenvolvido para aplicações *.NET*. O uso de mapeadores tem como objetivo realizar uma interação direta entre os tipos de entidades do banco e os objetos tratados na programação, facilitando o tratamento dos dados. No caso do *Entity* a integração com a base é realizada de maneira automática, possuindo diversos provedores disponíveis para conexões com *SQL Server*, *Oracle*, *PostgreSQL*, entre outros.

2.7 Visão computacional

Visão computacional é um ramo da inteligência artificial dedicado ao processo de modelagem e replicação da visão humana por meio de *software* e *hardware*. Os estudos na área tem como objetivo estudar maneiras de reconstruir, interromper e compreender uma cena em 3D a partir de imagens 2D [35].

Como mencionado no subcapítulo anterior a visão computacional refere-se aos processos de alto nível, uma vez que trabalha com a cognição dos dados recebidos. Inicialmente estas técnicas eram aplicadas somente em tarefas repetitivas em linhas de montagem. Hoje já são desenvolvidos trabalhos mais complexos, como a modelagem tridimensional [36].

Existem diversos processos que já utilizam desta tecnologia, destacando-se detecção de

falhas em processos industriais [37], sistemas de segurança contra-terrorista de aeroportos, manipulação de braços robóticos, controle de qualidades em fábricas, identificação de incêndios, entre outros [38].

2.7.1 *OpenCV*

A *OpenCV* surgiu de uma iniciativa da *Intel Research* para avançar com aplicações intensivas em *CPU* em 1999. A *Open Source Computacional Vision* foi lançada com uma série de projetos incluindo *Ray Tracing*(algoritmo para a renderização de imagens tridimensionais), muito utilizado até os dias de hoje[39].

Os dois principais objetivos do projeto são fomentar o avanço da pesquisa no campo da visão computacional não somente com código aberto mas também com código otimizado, tornando-o facilmente lido e transferível e também gerar aplicações portáteis, fornecendo recursos gratuitos. A plataforma foi desenvolvida nas linguagens *C* e *C++* e conta com compatibilidade para diversos sistemas como *Android*, *BlackBerry*, *iOS*, *Linux*, *Mac OS X* e *Windows*. É possível realizar implementações em várias linguagens diferentes como *Python*, *Java*, *Matlab*, entre outras.

A biblioteca possui mais de 500 funções e cumprindo com um dos objetivos possui disponibilidade tanto no modo de código fonte e os executáveis (binários). Um programa que implementa a *OpenCV* ao ser executado invoca uma *DLL* (*Dynamic Linked Library*) que detecta o tipo de processador carregando a *DLL* otimizada de acordo com o resultado da primeira verificação. Junto ao pacote principal é oferecida uma biblioteca chamada *IPL* (*Image Processing Library*) que estabelece dependência da biblioteca principal, além de fornecer exemplos e compartilhar parte da documentação [41].

Segundo o mesmo autor a biblioteca pode ser dividida em cinco principais grupos de funções: Processamento de imagens; Reconhecimento de padrões e Calibração da Câmera; Análise Estrutural; Análise de movimento e rastreamento de objetos e reconstrução 3D.

A) Segmentação

Todo o sistema automático de processamento de imagens tem início com a segmentação da imagem [42]. Este processo consiste no particionamento do *frame* em regiões. As características que guiam esse procedimento são geralmente cor e proximidade. O nível da eficiência desta etapa depende de vários fatores como luminosidade e resolução (relação de *pixels* por unidade de tamanho) da imagem que se tem. Existem diversas técnicas que podem ser utilizadas, separando-se entre método de verificação de descontinuidades e por similaridades[41]. Abaixo são descritos dois métodos:

- Segmentação por detecção de borda: Uma borda pode ser definida como a variação abrupta na intensidade dos *pixels*. Ao realizar esse tipo de procedimento assume-se que se possui cores homogêneas, isso não é uma realidade, uma vez que existe um "borramento" devido ao processo de amostragem.

As variações podem ser observadas dispondo das derivadas primeira (operadores de gradiente) e/ou segunda. Na *OpenCV* utilizam-se dois operadores: o de *Prewitt* (equações 2.1 e 2.2) e *Sobel* (equações 2.3 e 2.4) a imagem considerada é apresentada na Figura 2.9.

$$g(x) = \frac{\partial f(x,y)}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \quad (2.1)$$

$$g(y) = \frac{\partial f(x,y)}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \quad (2.2)$$

$$g(x) = \frac{\partial f(x,y)}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \quad (2.3)$$

$$g(y) = \frac{\partial f(x,y)}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \quad (2.4)$$

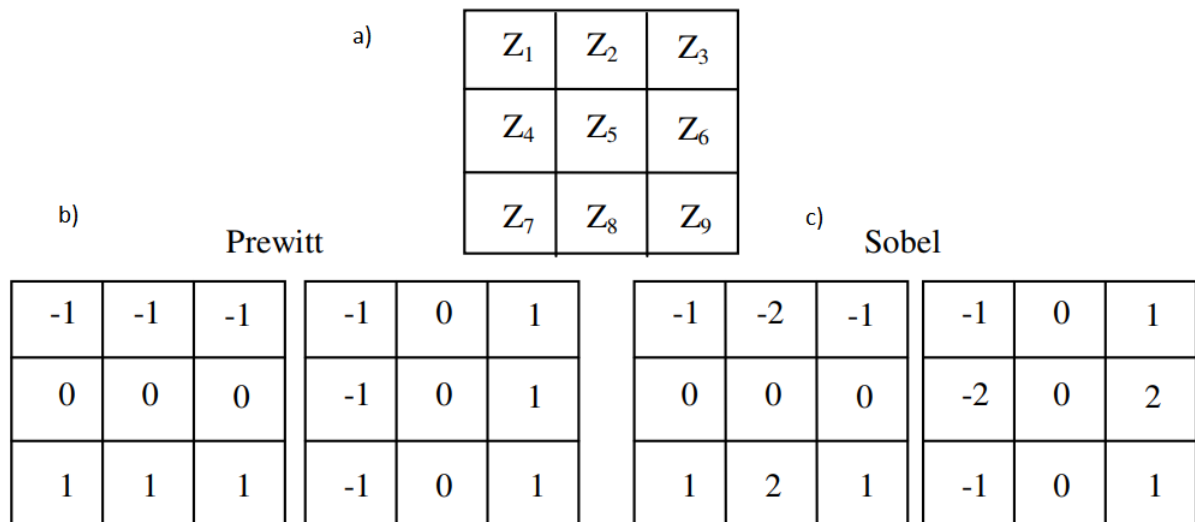


Figura 2.9: a) Máscara genérica com a posição dos valores utilizados nas equações de *Prewitt* e *Sobel*. b) Máscaras dos operadores de *Prewitt*, c) Máscaras dos operadores de *Sobel* [41].

As diferenças nos dois métodos são sutis, sendo verificadas apenas no segundo termo. Esta mudança tende a suavizar o resultado do operador, fazendo com que ruídos sejam atenuados.

- Segmentação por corte (Figura 2.10) Este método tem como principais vantagens a facilidade de ser implementada, a rapidez com que é processada e o fato de ser extremamente intuitiva. O particionamento se dá observando os valores de intensidade dos *pixels*. Uma vez gerado o histograma da imagem define-se o limiar. Utilizando este método é possível criar regiões segmentadas não somente binárias, ruídos podem ser tratados diretamente no histograma.

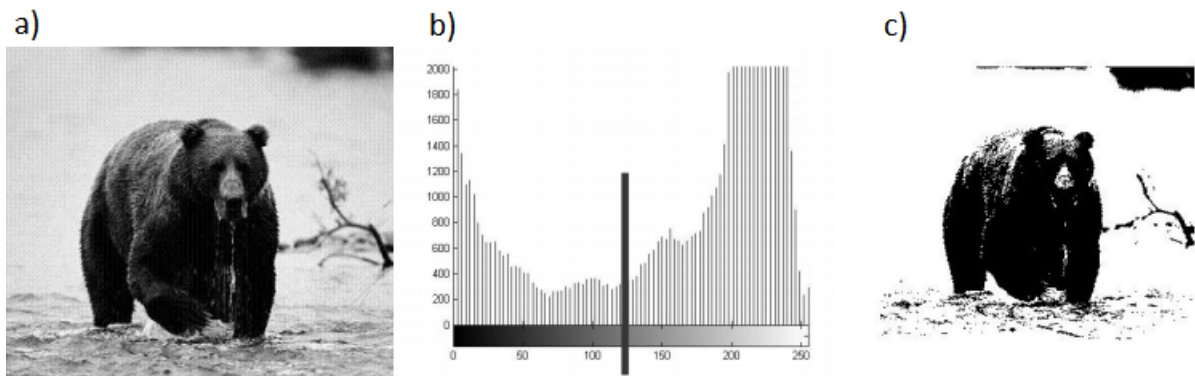


Figura 2.10: Segmentação de imagem utilizando a técnica de corte. a) Imagem em nível de cinza b) Histograma com o ponto de corte e c) Imagem binarizada [41].

B) Reconhecimento de padrões

O reconhecimento de padrões é uma etapa essencial em sistemas de análise de alto nível [43]. O reconhecimento de padrões como o algoritmo capaz de diferenciar objetos [37].

- Método *HaarCascades*: Este procedimento de reconhecimento de padrões foi proposto por Paul Viola e Michael Jones. O algoritmo de *machine learning* é reconhecidamente um dos mais efetivos na detecção de objetos [40].

O uso do classificador na detecção de faces é realizado após o treinamento com diversas imagens, tanto positivas quanto negativas. O método seleciona pontos característicos baseados em características como o fato da região dos olhos ser mais escura que a das bochechas e do nariz [37].

- Método *Background Subtraction (BS)*: Trata-se de um passo crucial em diversos sistemas de visão computacional, sua aplicação mais clara é a detecção de movimentos de um objeto em vídeo, possuindo conhecimento sobre o item a ser monitorado. Este campo vem sendo explorado desde 1990 principalmente em sistemas de segurança [44].

Como o nome sugere, o procedimento é feito subtraindo o fundo do *frame* atual. Inicialmente é gerada uma máscara dispondo apenas do fundo, ou seja, a parte estática da

cena, em seguida é feita a subtração propriamente dita. Uma demonstração da aplicação do método pode ser verificada na Figura 2.11.

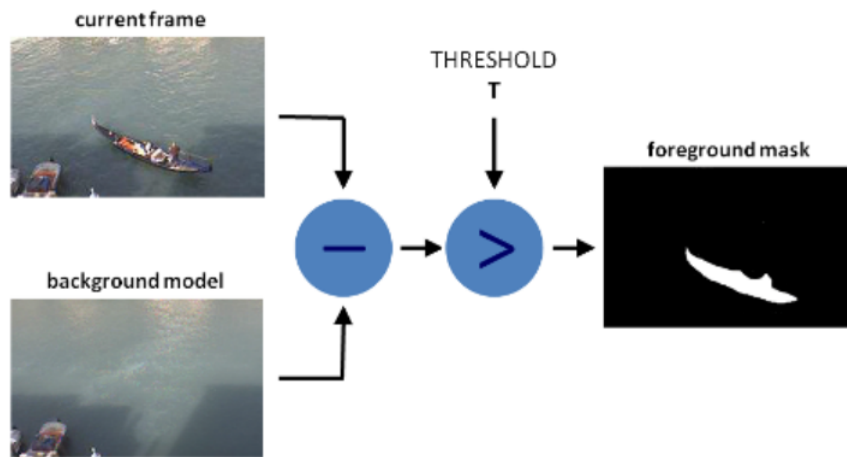


Figura 2.11: Demonstração da atuação do *Background Subtraction* [40].

Após a inicialização do fundo é necessário que se realize a atualização do fundo, no qual o modelo é atualizado para se adequar a possíveis alterações na cena.

C) Processamento de imagem

A fronteira entre o processamento de imagem e visão computacional não é delimitada claramente. Pode-se dizer que o processamento de imagens é o procedimento que antecede a visão computacional [41]. O primeiro processo tem como entrada uma imagem e a saída um conjunto de valores numéricos, podendo ou não compor uma outra imagem.

Dentre os procedimentos descritos no âmbito desta subseção os mesmos podem ser divididos em níveis. Os chamados procedimentos de baixo nível são exemplificados como o tratamento de ruídos e uma eventual melhora no contraste da imagem. A segmentação aparece como uma tarefa de nível médio. Os processos mais complexos são relacionados a cognição, associados diretamente com a visão humana [41].

A delimitação dos campos aqui descritos pode ser melhor compreendida na Figura 2.12. A imagem (a) mostra uma foto escura. A figura (b) é obtida após o processamento. É possível notar a melhora no contraste e consequentemente a possibilidade de leitura da placa. Leitura feita, por sua vez, utilizando visão computacional.

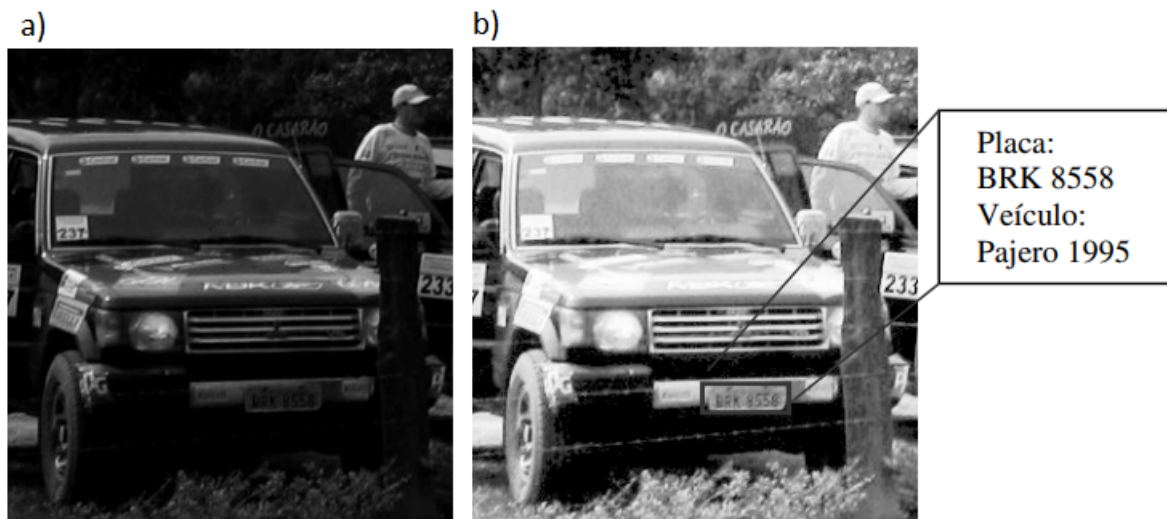


Figura 2.12: *Diferenças entre o processamento de imagens e a visão computacional* [41].

Outro aspecto a ser explorado quando se desenvolve sistemas embarcados é a possibilidade de compactação da imagem, visto que se trata de um procedimento computacionalmente caro [37].

O sistema de processamento de imagem consiste nas seguintes etapas [45]:

- **Aquisição:** Primeiro passo para a realização do processamento, tem como função a conversão da imagem em uma representação numérica adequada ao processamento digital seguinte. Neste processo, são dois os principais elementos: um dispositivo físico, capaz de captar energia em uma faixa específica no espectro eletromagnético produzindo na saída um sinal elétrico, e um digitalizador. Este segundo módulo toma o sinal analógico como entrada e o converte em informação digital, ou seja em *bits*. Deste modo, o sinal obtido é amostrado e em seguida discretizado em amplitude, processo conhecido como quantização, tornando-a ideal ao processamento computacional.
- **Armazenamento:** Uma vez adquirida é necessário armazenar a imagem. O armazenamento pode ser dividido em 3 principais grupos:

Curta duração: esse armazenamento é empregado quando a imagem é utilizada em diversas etapas do processo. Os principais requerimentos são quanto a velocidade de acesso, sendo assim muitas das vezes a imagem é armazenada na memória RAM

do computador principal.

Em massa: este método é utilizado quando se realiza operações de recuperação da imagem relativamente rápidas. Nesta modalidade são utilizados discos magnéticos (HD's) e caracterizam-se pela disponibilidade de muitas informações sem que haja perda considerável da velocidade de acesso. Neste caso o tipo de imagem armazenado é algo relevante e não possui um padrão único. Destacam-se os formatos BMP, PCX, TIFF, JPEG e GIF.

Arquivamento de imagens: Caracteriza-se por um volume muito grande de *bytes* contendo imagens que são acessadas de maneira esporádica. Nesta modalidade são utilizados HD's e eventualmente, dependendo da disponibilidade e qualidade da internet, armazenamento em nuvem.

- **Processamento:** Esta etapa compreende métodos algorítmicos implementados via *software*. A utilização de *hardware* dedicado ao processamento só é empregada quando o computador principal possui limitações intoleráveis ao processo como, por exemplo, a velocidade de transferência de dados. Nesta fase podem ser empregadas técnicas de tratamento de diversos aspectos como filtros Gaussianos, Passa-Alta entre outros [37].
- **Saída:** Executados os tratamentos é necessário que se realize a transmissão, seja para um monitor ou outro periférico cuja função seja de visão computacional. A grande dificuldade na transmissão se refere ao volume de dados a ser transferido. Este problema torna-se ainda maior caso envolva grandes distâncias.

O fluxograma abaixo (Figura 2.13) ilustra o procedimento acima descrito.



Figura 2.13: Diagrama do processamento de imagens. Adaptado [46].

2.8 Tecnologias de front end

Também conhecida como interface do usuário é a maneira como a aplicação se comunica com o mundo externo, recebendo e enviando dados para as camadas de informação.

A interface do usuário tem como principal requisito a interatividade: é necessário entender por quem o sistema será utilizado e quais suas necessidades [47]. De uma maneira geral, essa parte do *software* é construída utilizando HTML (Marcadores Gráficos), Javascript e CSS (folha de estilos). Além disso, existem diversas bibliotecas e *frameworks* dedicados à facilitar a implementação de *websites* cada vez mais interativos e otimizados. Um desses frameworks é o NodeJS. Em suma, define-se o mesmo como uma "plataforma para construir aplicações web escaláveis de alta performance usando JavaScript"[48].

A construção dessa plataforma foi baseada na *Engine V8* do Google Chrome. Esse fato leva à um comportamento diferente dos produtos similares ao mesmo no mercado: a linguagem pode ser utilizada na construção de servidores, não apenas pelo *Browser*. O gerenciamento de pacotes utilizados na aplicação é uma funcionalidade recorrentemente utilizada[48].

Além da importação e gerenciamento de pacotes, é comumente utilizado uma ferramenta para manipulação da interface visual com o usuário. Um bom exemplo é o React, biblioteca *Javascript*. Sua arquitetura é baseada na construção de componentes reutilizáveis, de maneira que cada um deles tenha um estado e sejam integrados facilmente ao término do desenvolvimento. O suporte ao uso do mesmo em dispositivos móveis é conferido através do *React Native*. Duas das propriedades presentes em todos os componentes dois merecem destaque: o *Estado* e a *props*, sendo a primeira privada e a segunda advinda de componentes que se comunicam[49].

A construção visual dos componentes pode ser facilitada pela utilização de um sistema de design. O *Material UI* é um conjunto de componentes React, criado pela comunidade, visando proporcionar à desenvolvedores a criação de UI's intuitivas e bonitas de maneira rápida. Trata-se de um *design system* suportado por código *open-source* tornando-o livre para qualquer uso[50].

As interações com a camada de informação são realizadas por meio de consultas à Web Services. A AxiosJS é um biblioteca destinada à tal função, consumindo HTTPS baseado em *promises* podendo ser utilizada tanto no *Browser* quanto no *NodeJS*. Dentre as funcionalidades destaca-se a grande flexibilidade no modo de fazer as requisições: é possível utilizar todos os verbos HTTP (*POST*, *DELETE*, *GET*, *PUT*). Além disso toda a informação trafegada é automaticamente transformada em JSON (JavaScript Object Notation), tornando-a integrada a UI [51].

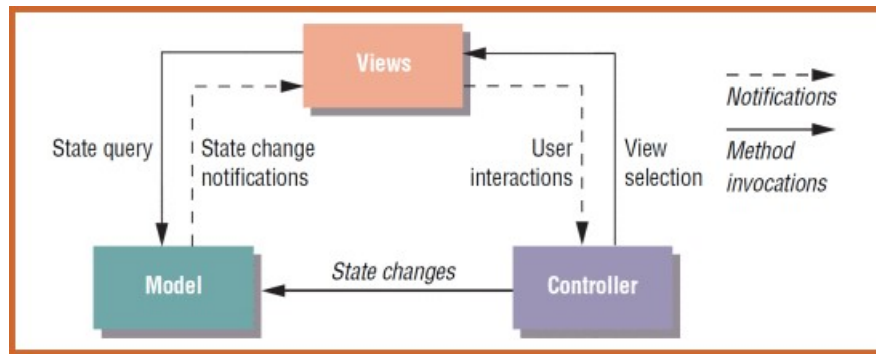
Após a interface construída é necessário publicar a mesma na Internet. O Netlify é um serviço de publicação de *websites* estáticos, possibilitando o acesso à interface online. O fato de ser gratuito e, principalmente, o *continuous deployment* são extremamente atrativos. A integração com repositórios pode ser feita com algumas plataformas como *GitHub*, *Gitlab* e *BitBucket*. Uma vez configurado o *build* a aplicação será publicada toda vez que o repositório remoto for atualizado, sem a necessidade de nenhum ajuste. A tecnologia é empregada por grandes corporações globais como *Facebook*, *Nike* e *Citrix* [52].

2.9 *Model-View-Controller*

O *MVC*, como é mais conhecido, é um padrão de projeto de *software* muito utilizado. A principal característica desse padrão é a separação das camadas do projeto de maneira muito bem definida em *Model*, *View* e *Controller* [53].

A definição das camadas pode ser melhor descrita através do isolamento das regras de negócio e da lógica de apresentação. Essa característica permite que a interação com o usuário seja modificada sem que haja nenhuma alteração nas regras de negócio, proporcionando flexibilidade e oportunidades de reuso das classes [54].

Uma representação gráfica do funcionamento deste padrão pode ser visualizada na figura 2.14.

Figura 2.14: Padrão *Model View Controller* [55].

2.9.1 *Model*

As classes pertencentes ao modelo se referem às entidades que são realmente modificadas e geralmente persistidas em uma base de dados. Além disso essa camada contém as regras de negócios, lógica e funções necessárias na aplicação. Essa camada não deve possuir interação direta com o usuário em nenhum momento [53].

2.9.2 *View*

As classes desse tipo não tem atividades ligadas à forma como a informação é construída. Sua função é fundamentalmente a exibição e inserção das informações, mostrando como referências. Em linhas gerais, pode-se dizer que se trata de uma fotografia do *Model* naquele exato momento, sendo a camada de interação com o Usuário, recebendo e enviando dados [55].

2.9.3 *Controller*

Conforme mencionado, a categoria do modelo interage diretamente com a informação e a de visualização apenas exibe e recebe dados. É clara a necessidade de uma interface entre elas, para garantir o fluxo perfeito de atualizações: essa é a função das classes do *Controller*, uma conexão entre as demais estruturas [54].

Desenvolvimento

Neste capítulo são descritas as etapas do projeto, desde o levantamento conceitual dos requisitos até os testes finais de integração de subsistemas. Inicialmente são descritas as premissas consideradas na concepção de todo o dispositivo e do sistema computacional. Baseando-se na discussão inicial e na fundamentação teórica são analisadas as ferramentas disponíveis, justificando-se as opções realizadas. O *hardware* adquirido foi verificado de maneira individual, atestando seu funcionamento.

Após descritos os testes físicos a descrição é ampliada para o aspecto computacional, apresentando modelagens e implementações. Ao término do capítulo é apresentada a lista de materiais e seu respectivo custo.

3.1 Projeto Conceitual

As primeiras definições acerca do trabalho foram questões relativas à comunicação e a retenção de dados. Na perspectiva de construir uma ferramenta *plug and play* é necessário que a necessidade de fiação seja a mínima possível, ou seja, inicialmente foram analisados microcontroladores e microprocessadores com baixo consumo energético e com comunicação *wireless*.

As condições de luminosidade influenciam diretamente no reconhecimento facial. Logo, a fechadura será instalada em um ambiente controlado, com condições de luminosidade conhecidas.

Além disso, decidiu-se que toda a interface de registro seria *web*.

3.1.1 Análise da Central de Processamento

A proposta inicial era que a central de processamento utilizada seria o *Arduino* Mega, devido a sua capacidade computacional e também seu baixo custo. Afim de confirmar esta escolha foram feitas pesquisas (tanto em trabalhos acadêmico quanto em comunidades *makers*). O resultado verificado foi a ausência de precedentes relevantes da utilização deste MCU¹ para o processamento de imagens.

Uma vez descartada a hipótese de utilização de um microcontrolador retornou-se à pesquisa bibliográfica. Alguns trabalhos semelhantes foram encontrados.

No primeiro trabalho analisado [13], foi utilizado um *Beaglebone Black*. As justificativas para o uso deste MPU foram o suporte da comunidade *open source*, além do baixo custo, estabilidade de desenvolvimento e a disponibilidade de aplicações e recursos *on-line*. Destacou-se também a possibilidade de se trabalhar com periféricos, como LCD's, sensores, antenas e câmeras.

Após o estudo da ferramenta discutida anteriormente retornou-se à pesquisa, buscando novas soluções afim de estabelecer um comparativo.

Dois trabalhos analisados [14] [37] utilizaram o *Raspberry PI 3*. As justificativas apresentadas foram a presença de periféricos de comunicação como *Wireless* e *Bluetooth*, além do baixo custo, disponibilidade de diversas interfaces com câmeras, grande suporte da comunidade *open source*.

Tendo base nos dois embarcados citados elaborou-se uma tabela (Tabela 3.1) comparativa, afim de facilitar a escolha da unidade de processamento.

¹*Microcontroller Unit*

Tabela 3.1: Comparativo - *Raspberry PI 3* e *Beagle Black Bone*

Componente	Raspberry PI 3	Beagle Black Bone
Fabricante:	<i>Raspberry PI Foundation</i>	<i>TI - Texas Instruments</i>
Custo: ²	R\$ 271,90	R\$534,90
<i>Clock</i> :	1,2 GHz	1 GHz
Processador:	<i>Broadcom</i> de 64 Bits	ARM <i>Cortex - A8 Core</i>
Memória RAM:	1GB	512 MB DDR3 RAM 800 MHz
<i>Bluetooth</i> :	4.1 BLE Integrado	4.1 BLE Integrado
<i>Wifi</i> :	802.11n integrado	802.11 b/g/n
Dimensões:	85 x 56 x 17mm	86,36 x 54,61 mm ³
Armazenamento:	<i>Slot</i> para cartão <i>MicroSD</i>	4GB <i>on-board</i> + <i>slot</i> para SD
<i>GPIO</i> :	40 pinos	2x 46 pinos
SO:	<i>Linux e Windows 10 IOT</i>	<i>Debian, Ubuntu e Android</i>
Alimentação:	5,1V/2,5A	5V/0,35A
Vídeo:	<i>HDMI</i>	<i>HDMI</i>
USB:	4 Portas 2.0	2 portas 2.0
Ethernet:	Conector RJ45	Conector RJ45
Áudio:	Estéreo de 4 polos	-
Diferenciais:	Interface DSI e CSI	Acelerador gráfico
Fornecedor:	FilipeFlop [56]	Mouser[57]

Os demais trabalhos encontrados utilizavam, em sua maioria, um computador propriamente dito para o processamento, condição incompatível com os objetivos do trabalho.

Após análise das informações verificou-se grandes semelhanças entre as duas plataformas. Os fatores que culminaram na escolha do *Raspberry PI* como central foram o custo e o tamanho de sua comunidade, possibilitando um suporte maior para o desenvolvimento.

O *pinout* da central processadora (Figura 3.1) pode ser consultado utilizando o comando "pinout" diretamente no processador. Esta informação tem bastante relevância para a análise e os testes a serem descritos abaixo.

²No Brasil, com frete incluso

³Dimensões obtidas no *site* do fornecedor brasileiro correspondente à placa sem conexão *wireless*

```

pi@raspberrypi: ~
┌───(raspberrypi)───┐
│ File Edit Tabs Help │
├───┤
│ Revision      : #52082 │
│ Soc           : BCM2837 │
│ RAM          : 1024Mb  │
│ Storage      : MicroSD │
│ USB ports    : 4 (excluding power) │
│ Ethernet ports : 1 │
│ Wi-fi       : True │
│ Bluetooth   : True │
│ Camera ports (CSI) : 1 │
│ Display ports (DSI) : 1 │
├───┤
│ JB: │
│ 3V3 (1) (2) 5V │
│ GPIO2 (3) (4) 5V │
│ GPIO3 (5) (6) GND │
│ GPIO4 (7) (8) GPIO14 │
│ GND (9) (10) GPIO15 │
│ GPIO17 (11) (12) GPIO18 │
│ GPIO27 (13) (14) GND │
│ GPIO22 (15) (16) GPIO23 │
│ 3V3 (17) (18) GPIO24 │
│ GPIO18 (19) (20) GND │
│ GPIO9 (21) (22) GPIO25 │
│ GPIO11 (23) (24) GPIO8 │
│ GND (25) (26) GPIO7 │
│ GPIO0 (27) (28) GPIO1 │
│ GPIO5 (29) (30) GND │
│ GPIO6 (31) (32) GPIO12 │
│ GPIO13 (33) (34) GND │
│ GPIO19 (35) (36) GPIO16 │
│ GPIO26 (37) (38) GPIO28 │
│ GND (39) (40) GPIO21 │
├───┤
│ For further information, please refer to https://pinout.xyz/ │
│ pi@raspberrypi:~ $

```

Figura 3.1: Pinout obtido no terminal. Fonte: Elaborada pelo autor.

A Figura 3.2 apresenta a pinagem disponibilizada no site indicado na documentação. A diferença para esta abordagem se deve ao fato da mesma possuir os pinos de comunicações explicitados.

Raspberry Pi GPIO BCM numbering

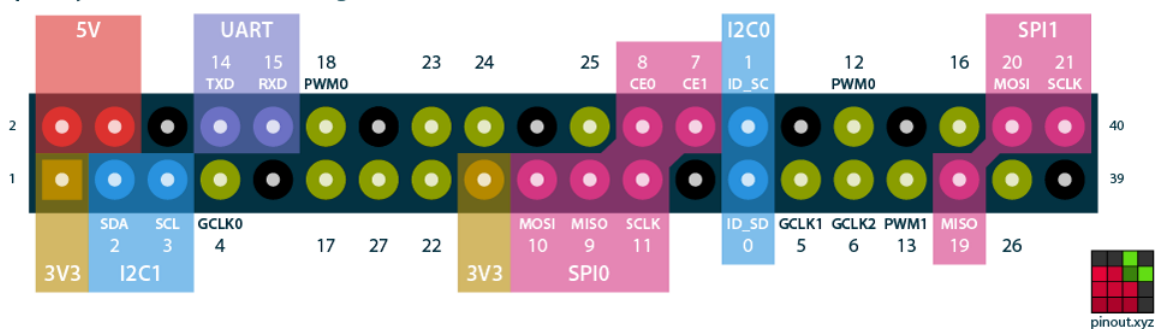


Figura 3.2: Pinout com descrição das comunicações[58].

Já o sistema operacional escolhido foi o *Raspbian Buster* em sua versão completa. A escolha se deve ao fato de que esse SO é desenvolvido e recomendado pela fabricante do microcomputador. Além disso é disponibilizado gratuitamente e constantemente atualizado (*release* de 26/09/2019)[20].

A instalação é feita de maneira simplificada fazendo o *download* diretamente no site da fabricante e, em seguida, gravando-o num cartão SD. O cartão de memória escolhido é fabricado pela *SanDisk* de classe 10 e possui 16GB de capacidade (o recomendado pela *Raspberry Pi Foundation* é de no mínimo de 8GB).

Para a gravação do sistema operacional foi utilizado o *software Win32 Disk Imager* por ser *opensource*, pequeno e de fácil uso, sendo recomendado pela comunidade de desenvolvimento.

3.1.2 Análise dos periféricos

Os periféricos nativos do processador se mostraram altamente eficientes. O *bluetooth* não foi explorado, pois não é relevante para a aplicação. O *wi-fi* se mostrou estável, sendo toda a instalação da biblioteca sendo baixada por esse meio. Foram realizados testes satisfatórios de reprodução de vídeos.

A comunicação do módulo RFID escolhido para o trabalho e a unidade de processamento se dá por meio de SPI. A comunicação, baseando-se no diagrama de pinos (Figura 3.2) foi realizada de acordo com a Tabela 3.2.

Tabela 3.2: Ligação módulo RFID e processador

Pino RC522	Pino PI	GPIO
SDA	24	GPIO8
SCK	23	GPIO11
MOSI	19	GPIO10
MISO	21	GPIO9
IRQ	-	-
GND	9	GND
RST	22	GPIO25
3.3V	1	3V3

A montagem física é apresentada na Figura 3.3.

A parte de *software* realizada inicia-se com a habilitação da comunicação SPI por parte do microprocessador. A mesma deve ser feita no caminho *"Menu>Preferences>Raspberry Pi Configuration"*. Após essa configuração deve-se reiniciar o sistema e utilizar o comando *"dmesg | grep spi"*. O retorno indicará se a função está habilitada.

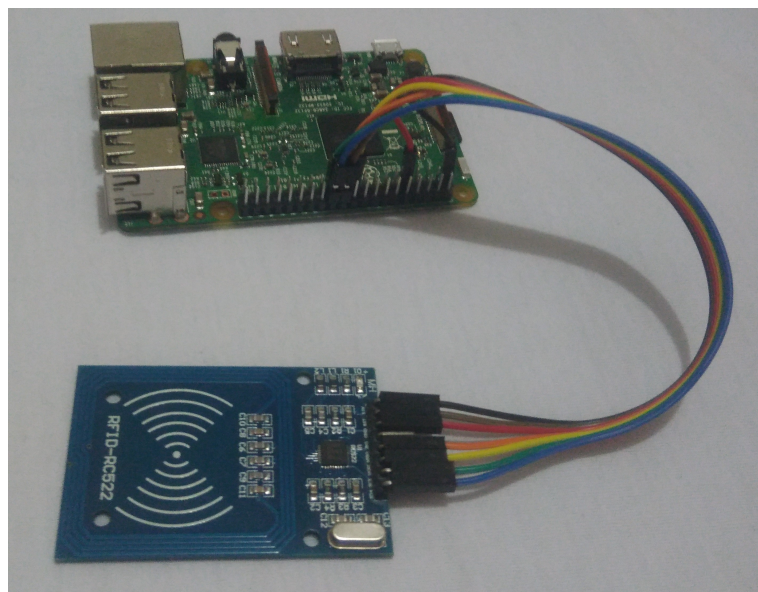
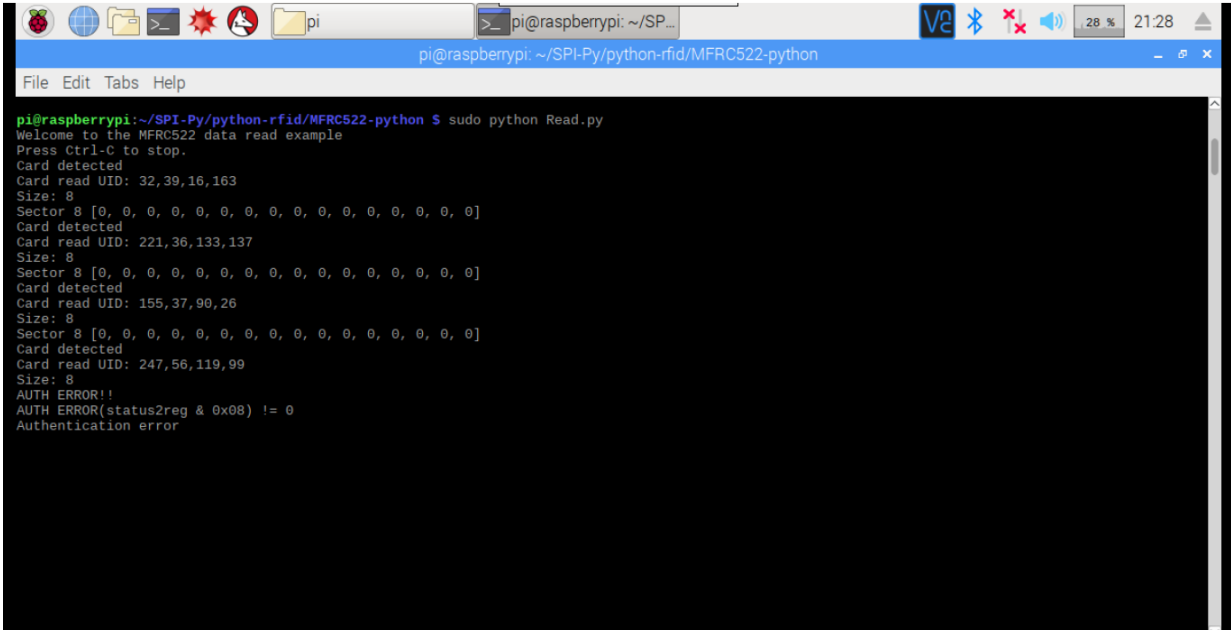


Figura 3.3: Ligação com o Módulo RFID. Fonte: Elaborada pelo autor.

A linguagem utilizada na documentação consultada [59] é a *Python*. A mesma é nativa do microcomputador, tem se expandido e possui grande comunidade. É necessário que se instale a interface entre a mesma e a comunicação. O pacote que faz essa interação é chamado de *SPI-Py* e está disponível no *GitHub*.

Após clonar e instalar o arquivo do repositório a comunicação está pronta para ser utilizada. Já existem implementações de exemplos utilizando essa combinação entre processador e periférico. A implementação utilizada para testes foi adquirida do *Github* e se chama *MFRC522-python*.

Utilizando o arquivo *Read.py* e aproximando-se cartões *RFID* padrões e também cartões já utilizados no dia-a-dia como o de identificação de alunos do CEFET e também do transporte coletivo da cidade foram obtidos os resultados apresentados na Figura 3.4.



```
pi@raspberrypi: ~/SPI-Py/python-rfid/MFRC522-python
File Edit Tabs Help
pi@raspberrypi:~/SPI-Py/python-rfid/MFRC522-python $ sudo python Read.py
Welcome to the MFRC522 data read example
Press Ctrl-C to stop.
Card detected
Card read UID: 32,39,16,163
Size: 8
Sector 8 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Card detected
Card read UID: 221,36,133,137
Size: 8
Sector 8 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Card detected
Card read UID: 155,37,90,26
Size: 8
Sector 8 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Card detected
Card read UID: 247,56,119,99
Size: 8
AUTH ERROR!!
AUTH ERROR(status2reg & 0x08) != 0
Authentication error
```

Figura 3.4: Ligação com o Módulo RFID. Fonte: Elaborada pelo autor.

Internamente a aplicação possui chaves chamadas de *smart card keys*. O erro de autenticação na imagem ocorreu ao se testar o cartão do transporte coletivo de Divinópolis. Este cartão necessita de maior segurança para que sejam evitadas fraudes. Portanto, o acesso a chave do mesmo é privada, de conhecimento apenas da administradora do sistema. O cartão de identificação estudantil provavelmente tem sua implementação baseada apenas no UID.

O código fonte é apresentado no repositório *smartlock-firmware* ⁴.

A câmera empregada no desenvolvimento (Figura 3.5) não foi discutida, devido ao fato de sua posse ser anterior ao desenvolvimento do trabalho e apresentar custo e desempenho condizente com o restante do projeto. Trata-se de uma *webcam* de baixa resolução com foco de 3.58mm, *zoom* digital de 10× e conexão USB.

⁴Disponível em: <https://github.com/matheus-mac/smarlock-firmware>



Figura 3.5: Câmera utilizada na primeira fase de desenvolvimento. Fonte: Elaborada pelo autor.

Dispositivos deste tipo são encontrados à custos menores do que a câmera de 5MP fabricadas pela Raspberry Pi Foundation. Como um dos focos do presente trabalho é o baixo custo, estas câmeras tornam-se soluções atrativas.

3.1.3 Materiais e Orçamento

A relação dos materiais escolhidos e adquiridos é apresentada na tabela 3.3.

Tabela 3.3: Orçamento do projeto

Item	Preço
Raspberry PI Kit ⁵	R\$205,00
Cooler 3x3cm ⁶	R\$9,49
Cartão de Memória	R\$27,49
Módulo RFID	R\$22,90
Câmera ⁷	R\$ 52,75
Chave Fim de Curso	R\$8,50
Módulo Relé de acionamento	R\$12,90
Insumos e Fretes	R\$35,62
Total:	R\$374,65

⁵Microprocessador e Fonte de alimentação

⁶Foi adquirido afim de aproveitar o frete. Eventualmente será incluído no protótipo final.

⁷Modelo similar ao utilizado

3.2 Transição do algoritmo para o sistema embarcado

A transição de algoritmos para o *Raspberry PI 3* pode ser efetuada de várias maneiras[37]. Duas das mais comuns são utilizando um monitor e periféricos como *mouse* e teclado conectados diretamente ao microcomputador. O segundo método, chamado de *headless* é feito sem a necessidade dos periféricos, precisando apenas de uma rede *wireless* e um outro computador de onde pode ser feito o acesso.

Inicialmente utilizou-se a primeira abordagem devido a simplicidade e semelhança com a operação de um computador normal. Devido à necessidade de se trabalhar em ambientes como o próprio CEFET onde nem sempre há um monitor e periféricos disponíveis para tais aplicações migrou-se para o *headless*.

A maioria das abordagens utiliza-se de programas como o *PUTTY* ou o *Raspberry PI Finder* para determinar o IP do microprocessador na rede. A fim de simplificar o uso do programa foi definido um *IP* estático e criou-se uma rede particular do microcomputador.

No procedimento atual conecta-se um cabo de rede e em seguida a alimentação ao *Raspberry*. A rede *wireless* é criada automaticamente. Feito isso conecta-se o computador à rede e utilizando o *software VNC Viewer* realiza-se o acesso mediante autenticação por senha. O resultado do acesso é apresentado na Figura 3.6.

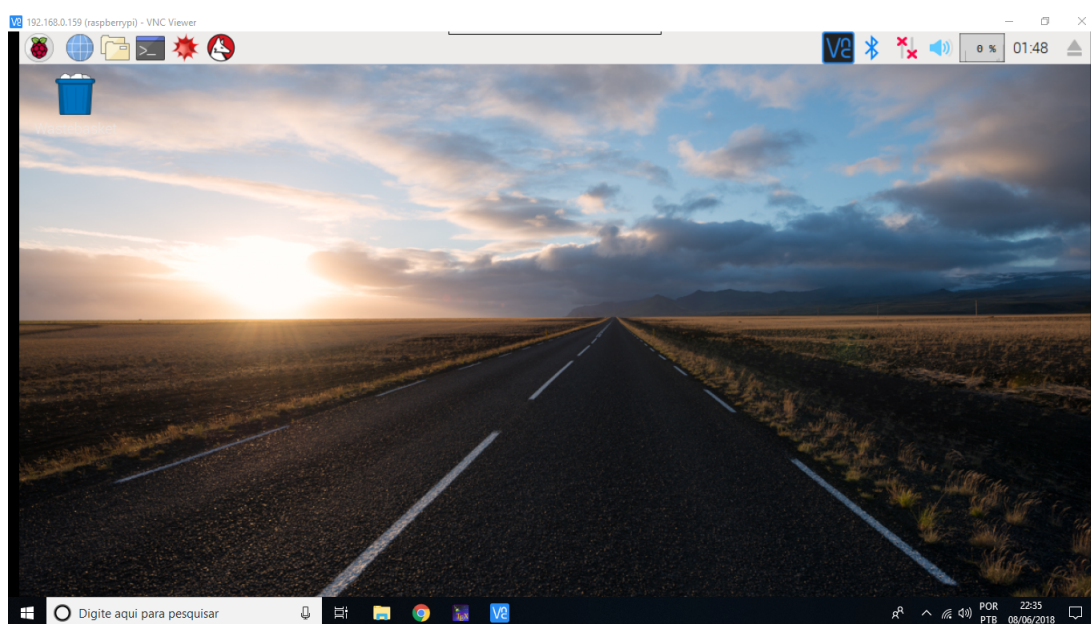
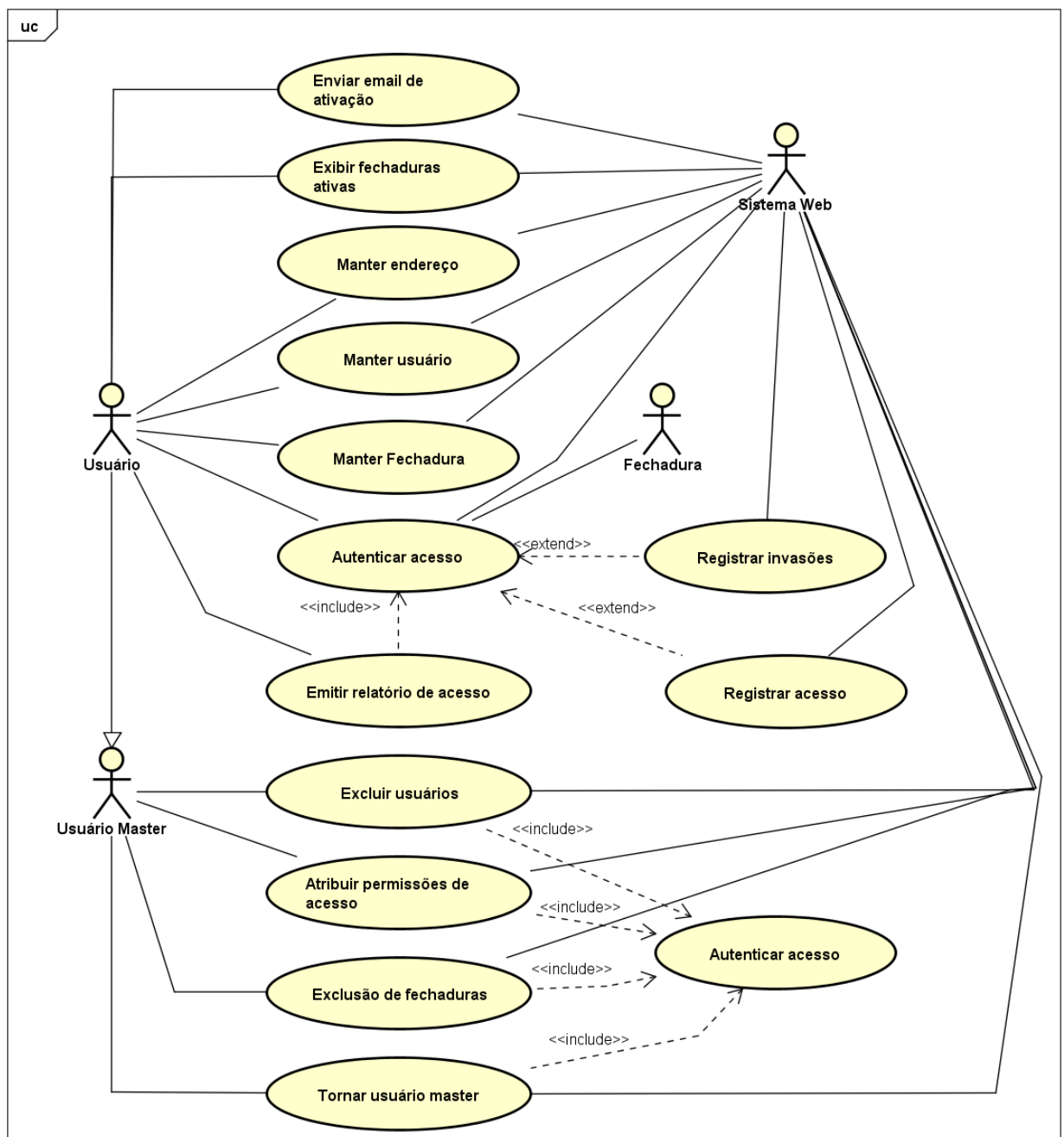


Figura 3.6: Acesso remoto utilizando o *VNC Viewer*. Fonte: Elaborada pelo autor.

3.3 Projeto do Software

O primeiro passo foi a definição de um padrão de projeto, o selecionado foi o *Model View Controller*. Em seguida, elaborou-se um diagrama de casos de uso (Figura 3.7). Foram definidas quais funções e atores essenciais deveriam ser envolvidos. O *software* utilizado para o desenvolvimento deste primeiro diagrama foi o *Astah Community*. A aplicação é fornecida gratuitamente para usos não comerciais. desenvolvedora.



powered by Astah

Figura 3.7: Modelagem UML de casos de uso. Fonte: Elaborada pelo autor.

O projeto do banco de dados foi desenvolvido após a definição das funções necessárias ao sistema, retiradas do diagrama de casos de uso. O software utilizado foi extraído de uma recomendação da bibliografia[32]. O BrModelo 3.0, disponível gratuitamente no site *sourceforge*.

O DER apresentado na Figura 3.8 define o banco de dados do projeto.

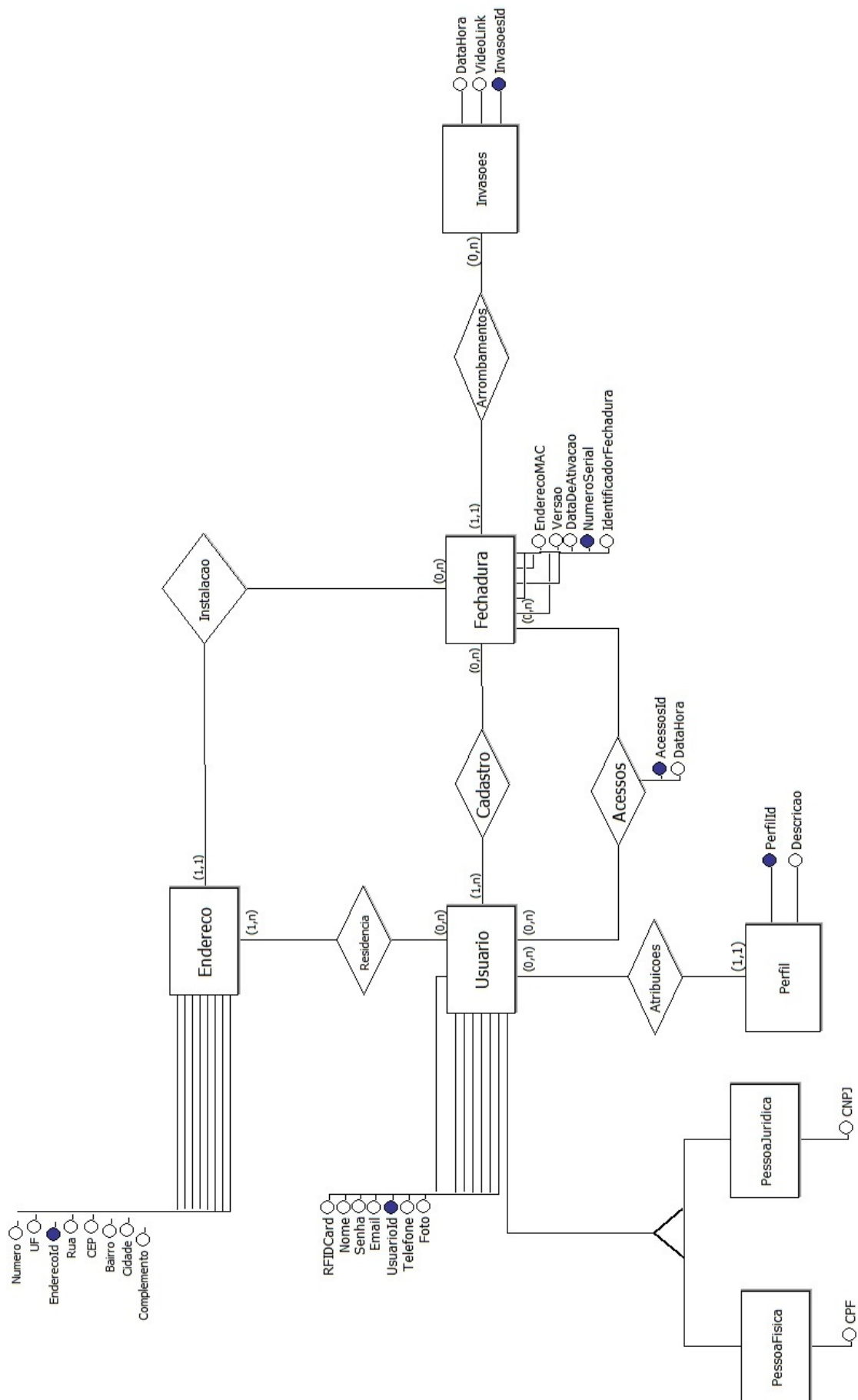


Figura 3.8: Modelo DER do banco de dados. Fonte: Elaborada pelo autor.

Com base no modelo conceitual (Diagrama Entidade-Relacionamento) gerou-se então o modelo lógico (Diagrama de Tabelas Relacionais) (Figura 3.9). Este modelo apresenta as informações no nível visto pelo usuário do SGBD. Portanto, para a construção de um modelo devem ser registradas quais tabelas o banco contém e quais colunas cada tabela possui [32].

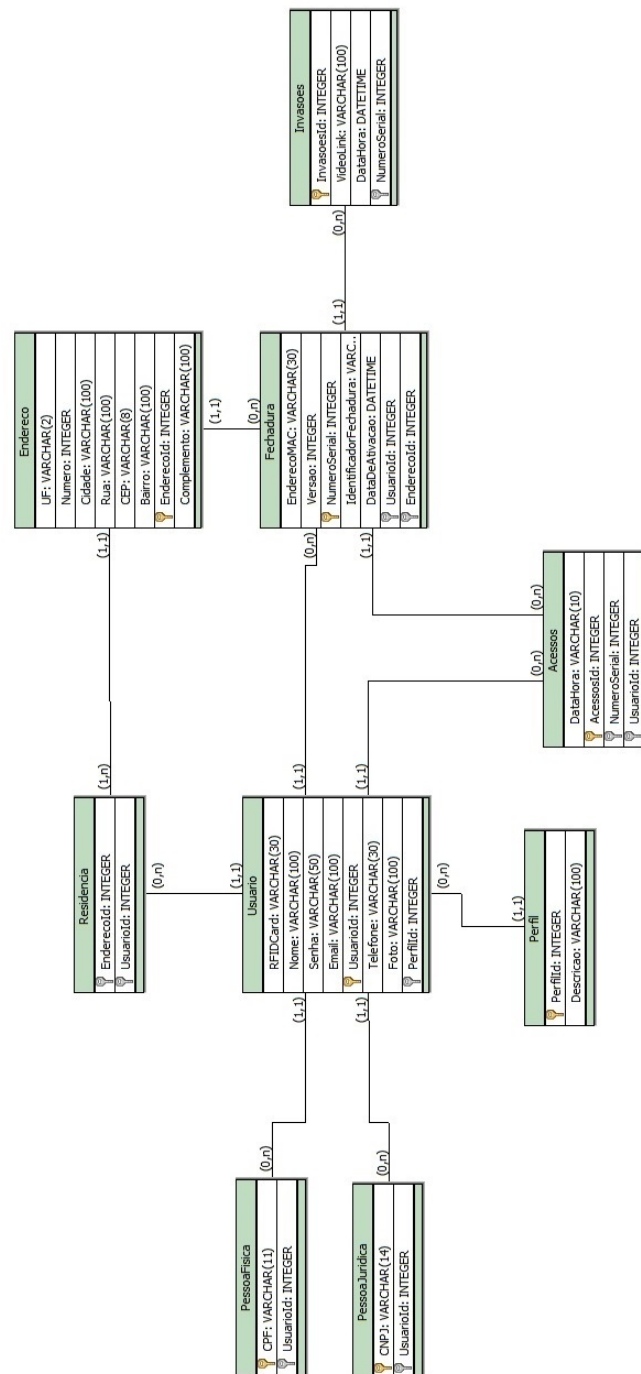


Figura 3.9: Modelo lógico do banco de dados. Fonte: Elaborada pelo autor.

Foi elaborado também o Diagrama de implantação (Figura 3.10).

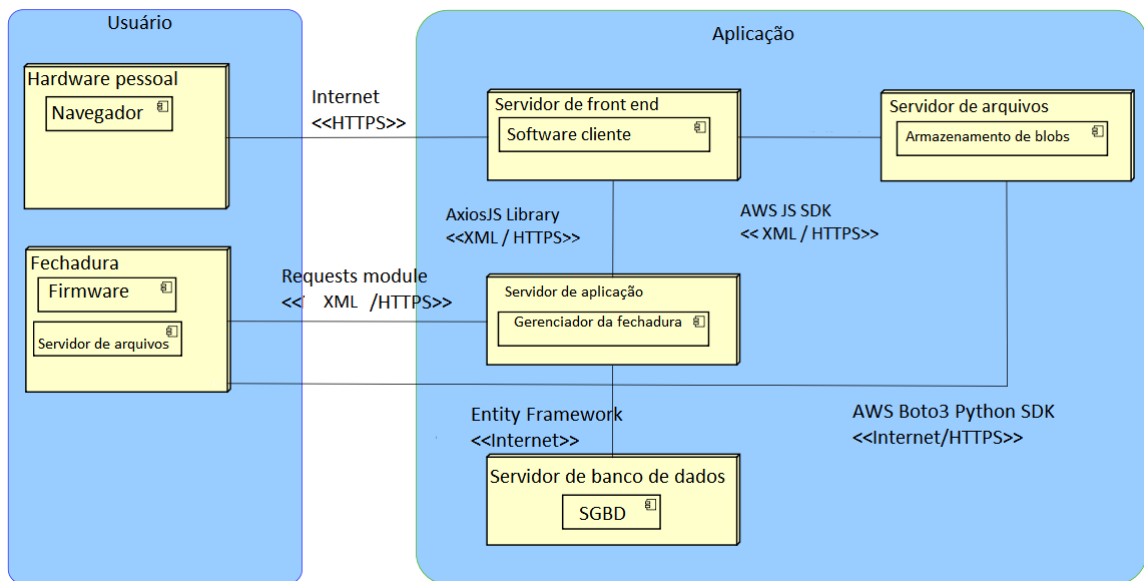


Figura 3.10: Diagrama de Implantação. Fonte: Elaborada pelo autor.

3.4 Projeto do *Firmware*

Uma vez levantados os requisitos da fechadura, com todas as funcionalidades desejadas, elaborou-se o diagrama de máquina de estados da mesma. O objetivo do desenvolvimento dessa representação tem foco na facilitação da implementação e principalmente na definição do escopo compreendido pelo produto. A Figura 3.11 apresenta o diagrama final obtido.

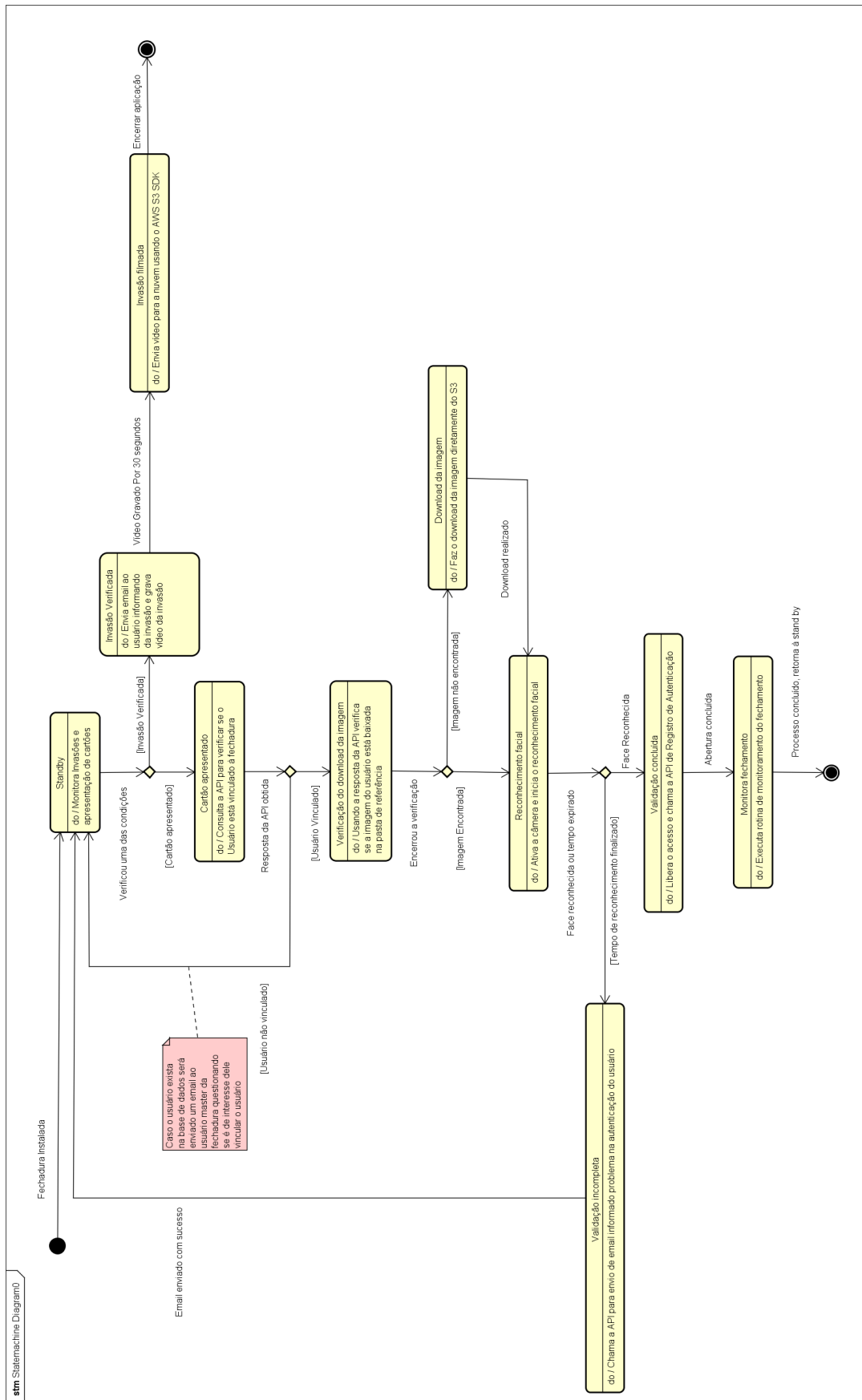


Figura 3.11: Diagrama de Máquina de Estados do firmware. Fonte: Elaborada pelo autor.

Resultados

4.1 Construção do dispositivo

O dispositivo construído é composto por todos os componentes mencionados na especificação de materiais. O leitor *RFID* que confere a capacidade de interação com as *tags* passivas utiliza comunicação SPI, sendo sua ligação definida por essa restrição. Os pinos utilizados pela comunicação são o 19 (MOSI), 21 (MISO), 23 (CLK), 24 (CE0), 26 (CE1). Além deles são alocadas duas portas para alimentação sendo a 01 para alimentação de 3,3V e 09 GND.

Os demais componentes são de acionamentos simples. O *cooler* é conectado nas portas 04 (5V) e 06 (GND), mantendo a alimentação constante. O fim de curso (Figura 4.1) é conectado à porta 17 (3,3V) e à porta 29 (GPIO05), sendo a segunda configurada como uma entrada para verificação de invasões. Por fim a conexão com o relé (Figura 4.2) é realizada utilizando os pino 02 (5V) e 34 (GND) para alimentação e o pino 36 (GPIO16) conectado ao pino IN1 do relé para acionamento, que ocorre quando o nível lógico da porta é baixo.



Figura 4.1: Fim de curso com haste e rolete. Fonte: Elaborada pelo autor.

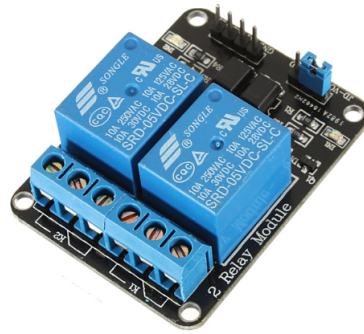


Figura 4.2: Módulo relé utilizado. Fonte: Elaborada pelo autor.

A montagem final, com todos os componentes conectados, é apresentada na Figura 4.3

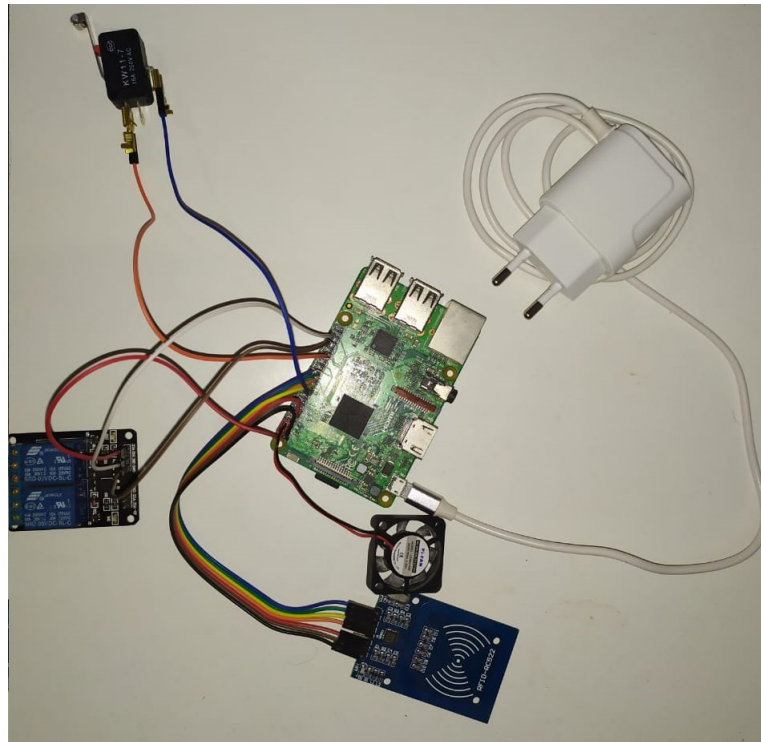


Figura 4.3: Montagem final. Fonte: Elaborada pelo autor.

A fim de acomodar todo o dispositivo foi projetada uma estrutura mecânica utilizando o *software Solid Works*. A mesma é composta pela parte inferior onde serão acoplados a maior parte dos componentes (Figura ?? e a tampa (Figura ??), seguida da montagem final (Figura 4.6).

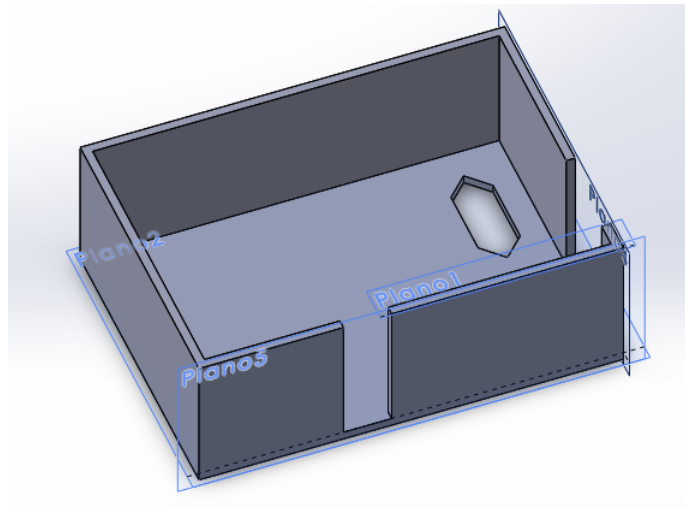


Figura 4.4: Parte inferior da *case*. Fonte: Elaborada pelo autor.

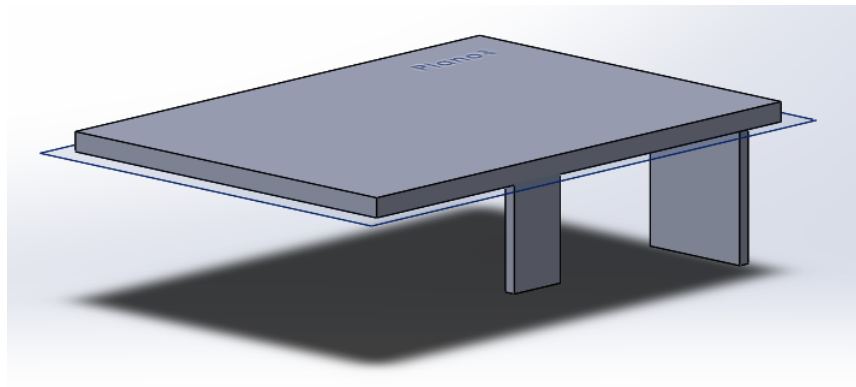


Figura 4.5: Tampa da *case*. Fonte: Elaborada pelo autor.

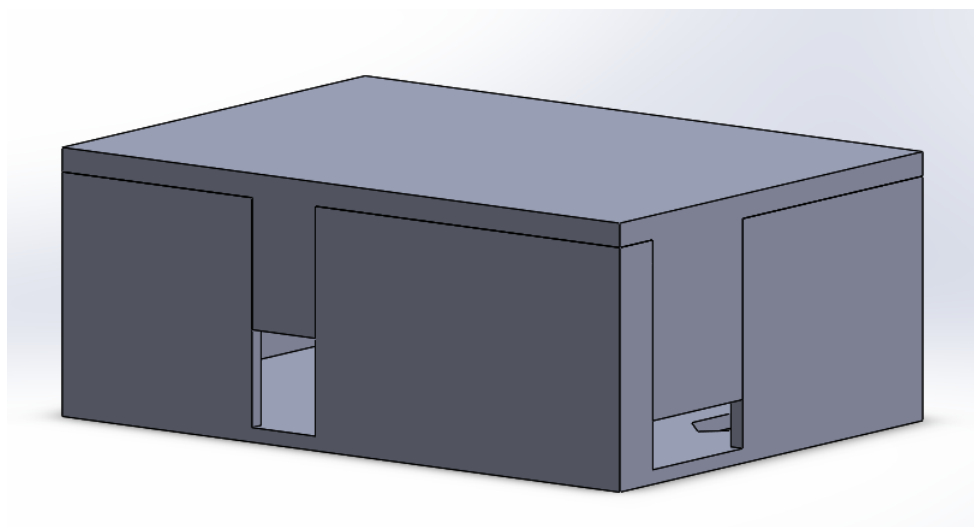


Figura 4.6: Vista frontal. Fonte: Elaborada pelo autor.

Após a impressão em ABS e acoplamentos realizados foi obtida a montagem conforme

visualizado nas Figuras 4.7 e 4.8.

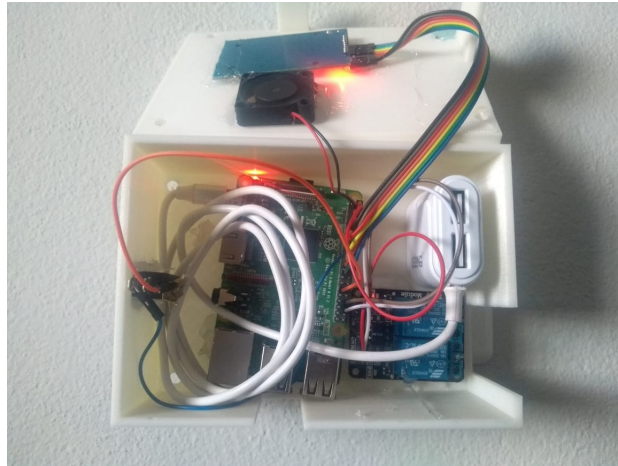


Figura 4.7: Parte interna da *case*. Fonte: Elaborada pelo autor.



Figura 4.8: *Case* concluída. Fonte: Elaborada pelo autor.

4.2 Reconhecimento facial

Para iniciar a implementação do reconhecimento facial foram elaboradas duas aplicações.

A primeira dispõe do método *HaarCascades* e do classificador *xml* dedicado a detectar faces de maneira frontal. O algoritmo captura a imagem, realiza o pré tratamento da mesma, transformando-a em uma matriz e realiza a detecção facial, desenhando quadrados sobre os olhos e faces (foi verificado o funcionamento para até 3 faces de maneira simultânea). O resultado obtido é apresentado na Figura 4.9. Observa-se que o processamento consumido foi de cerca de 56% no momento da detecção.

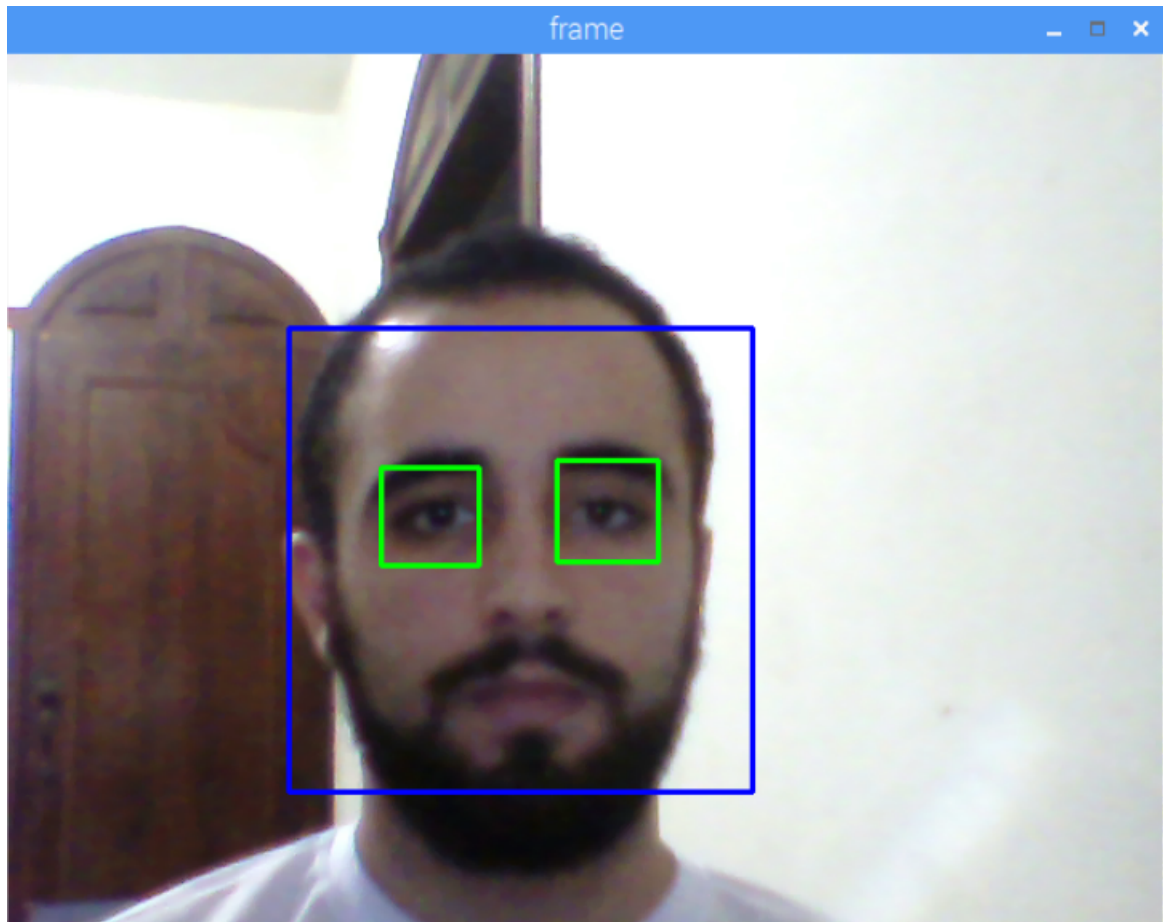


Figura 4.9: Detecção facial utilizando *HaarCascades*. Fonte: Elaborada pelo autor.

A segunda aplicação desenvolvida faz a captura de vídeo desde a sua execução até o momento em que a tecla "q" é pressionada. O resultado foi atestado com uma contagem em um pequeno vídeo de 5 segundos. A saída é um arquivo *.avi* de acordo com o *codec* definido no corpo da implementação.

4.3 Implementação do firmware

A partir do diagrama de implantação (Figura 3.10) foi implementado o fluxo no *firmware* utilizando *Python*. A estrutura indicada pela documentação da linguagem é baseada na construção de módulos, a fim de permitir que o código seja mais fluido, intuitivo e principalmente reutilizável.

O primeiro módulo implementado foi o *VerificaInvasao.py*. A mesma possui uma única função que tem retorno booleano chamada *portaoAberto*. A função verifica a posição da

chave de fim de curso retornando o status da abertura do portão.

As interações com o *back end* são realizadas por meio de requisições HTTPS. O módulo *requisicoesHTTP.py* é responsável pela conexão entre as duas plataformas. O mesmo utiliza a biblioteca *Requests* para lidar com as requisições.

O tratamento de imagens e gravação de vídeos em caso de invasões utilizando a *OpenCV* foi componentizado na *opencvFunctions.py*. Esse componente possui o tratamento de toda a cadeia de imagens obtidas, portanto o *upload* dos vídeos realizados pelo mesmo é realizado diretamente. As funções ligadas ao *upload* e *download* de arquivos foram implementadas no componente *AWSFunctions.py*, utilizando a biblioteca da *AWS boto3*.

A leitura e verificação da validade das *tags RFID* é realizada no módulo *leitorRFID.py*.

Por fim as tarefas ligadas ao reconhecimento facial foram implementadas no componente *FacialRecognition.py*. O mesmo importa a biblioteca *face_recognition* que lida com a decodificação de imagens em matrizes tornando-as comparáveis para realizar o reconhecimento.

O arquivo principal é chamado de *LockFirmware.py*, se comunicando com os módulos por meio do comando *import*. Todos os códigos mencionados podem ser encontrados no repositório *smartlock-firmware*¹.

4.4 Implementação do back end

O primeiro passo para a criação do *back end* foi a disponibilização do banco de dados de maneira online. A configuração foi realizada via AWS console, um resumo das configurações pode ser visualizado na Figura 4.10.

¹Disponível em: <https://github.com/matheus-mac/smarlock-firmware>

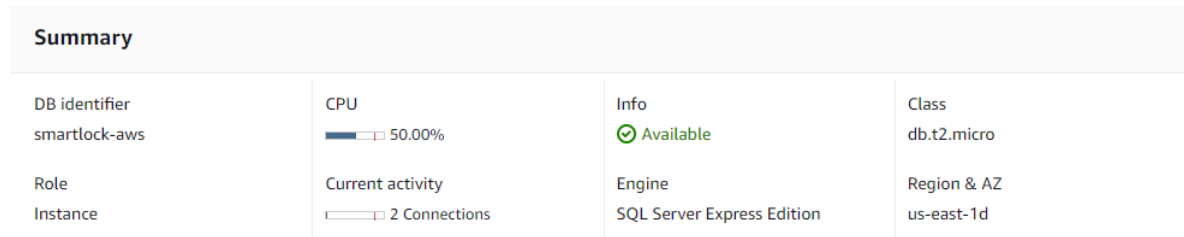


Figura 4.10: Resumo da implantação do banco de dados. Fonte: Elaborada pelo autor.

O link para acesso, criado automaticamente baseado na região de hospedagem, é o `smartlock-aws.clch9fh6mkji.us-east-1.rds.amazonaws.com`. A porta para acesso liberada é a 1433, padrão para bases de dados deste tipo.

A implementação propriamente dita iniciou-se com a criação do projeto *Web API* utilizando a IDE *Visual Studio*. As ações só podem ser implementadas a partir da conexão com o banco. Realizou-se então a configuração do *Entity Framework* após a instalação utilizando o *NuGET package Manager*.

Utilizando os parâmetros configurados na criação do BD é realizada a abstração de maneira automática, criando o modelo exibido na figura 4.11.

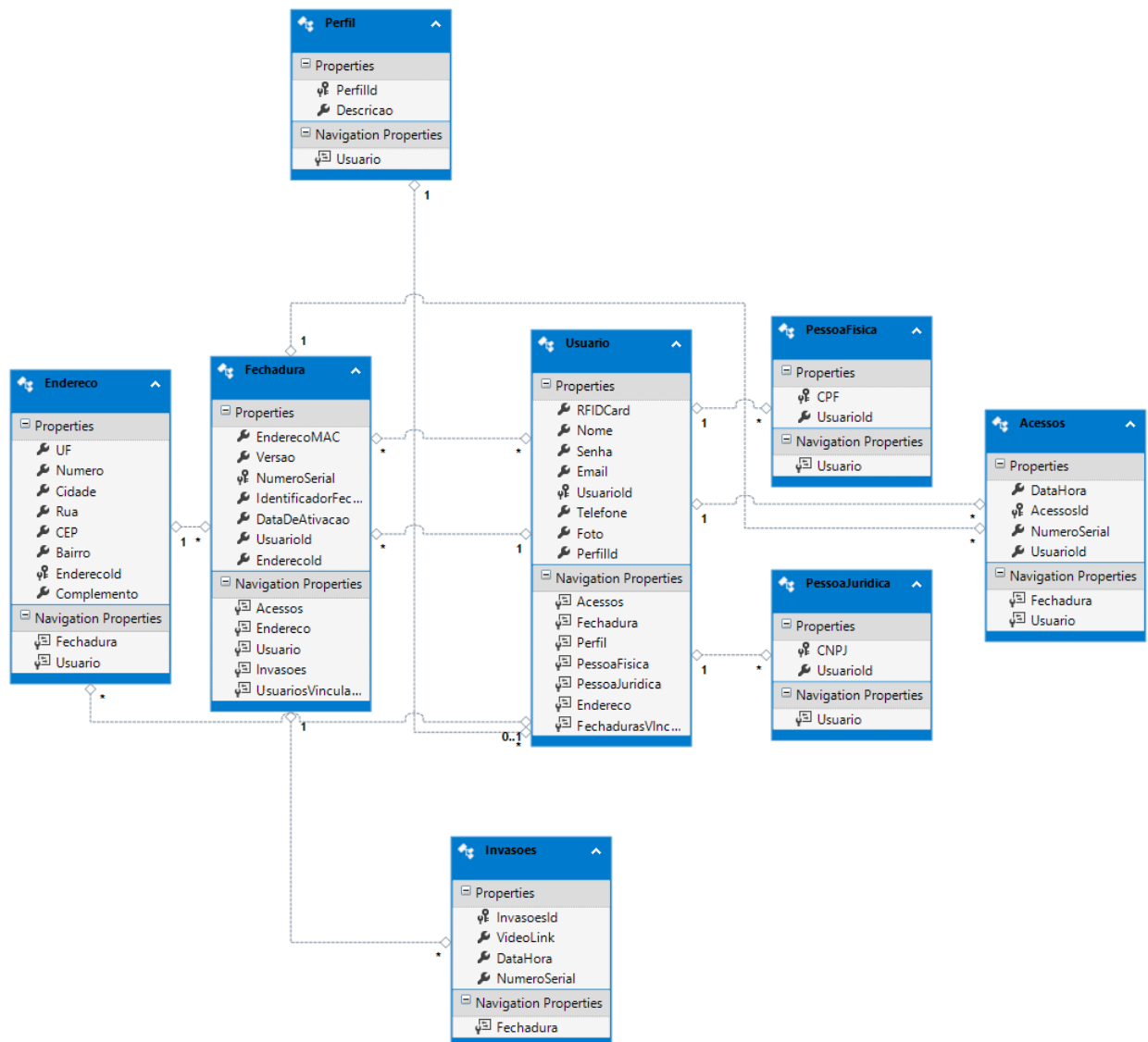


Figura 4.11: Resumo da implantação do banco de dados. Fonte: Elaborada pelo autor.

Cada quadro em azul é traduzido como uma classe, com os devidos atributos. Dentro das seções *Navigation Properties* são exibidas as *foreign keys*. De maneira a facilitar a implementação as relações são convertidas em objetos, permitindo que um objeto do tipo *Fechadura* acesse todos os usuários vinculados por meio de um atributo de vínculo.

O próximo passo foi a criação dos *controllers*. Na metodologia proposta pela *Microsoft* esse é o nome dado às classes que contém as APIs e endereços desenvolvidos.

Nesse ponto são criados vários endereços de acordo com as necessidades do sistema, como por exemplo, *registrarUsuario*. Além disso, são definidos quais parâmetros devem

ser fornecidos pelo cliente para que as requisições sejam realizadas.

O tipo de objeto retornado por essas funções é o *HTTPResponse*, logo é necessário que as consultas e retornos de ações sejam convertidas para objetos serializáveis. Na versão atual do *.NET Framework* não é possível serializar diretamente as classes criadas por meio da integração com o *Entity framework* fazendo-se necessária a criação de DTO's.

Os DTO's, do inglês Data Transfer Object, são classes criadas apenas para o tráfego de informações nas requisições. A grande vantagem desta abordagem é a possibilidade de criação de retornos particulares, fazendo com que os pacotes XML's transmitidos contendam apenas a informação estritamente necessária para o tratamento no cliente.

Uma vez finalizada a implementação a mesma foi implantada em uma máquina virtual da Amazon. Para tanto foi necessário realizar a configuração da máquina.

Inicialmente realizou-se o provisionamento da máquina via *AWS console*. O sistema operacional escolhido foi o *Windows Server 2012 R2* devido à disponibilidade gratuita do mesmo. O endereço público disponibilizado foi o *ec2-52-54-145-159.compute-1.amazonaws.com*.

Utilizando a ferramenta *Remote Desktop Protocol*, nativa do *Windows* foi possível acessar a máquina e iniciar a configuração necessária (Figura 4.12).

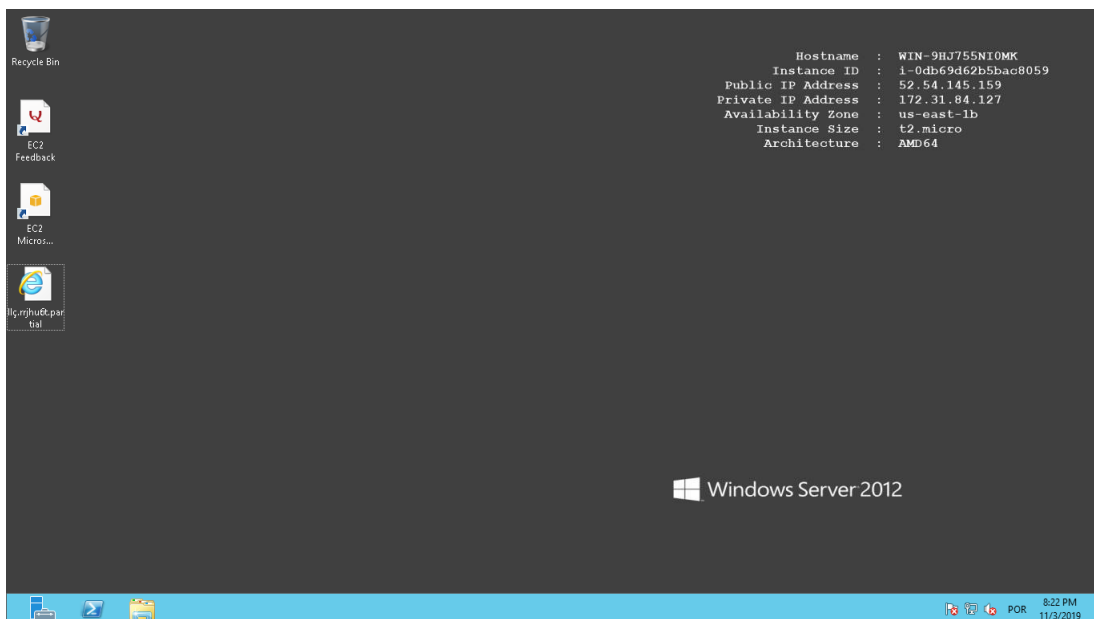


Figura 4.12: Máquina virtual provisionada para atuar como servidor. Fonte: Elaborada pelo autor.

Realizou-se a instalação de duas versões do *Internet Information Services*. A primeira delas (versão 6) foi utilizada para configuração do servidor de envio de e-mails. Já a versão 8 foi configurada para executar a aplicação. O resultado final é exibido na Figura 4.13.

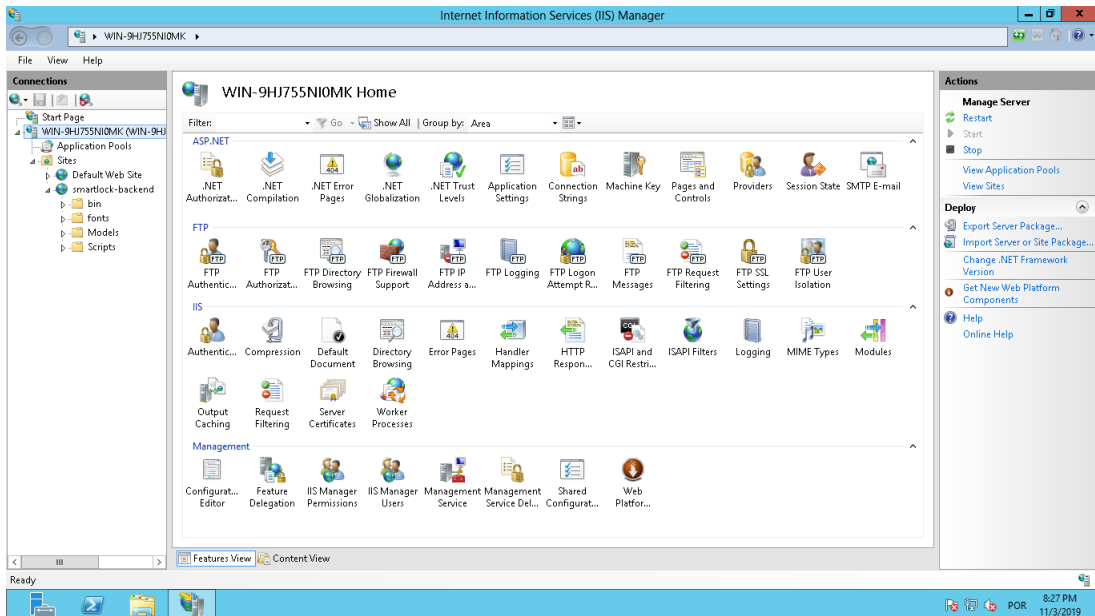


Figura 4.13: Servidor configurado para rodar a aplicação. Fonte: Elaborada pelo autor.

Uma vez configurado o servidor realizou-se a publicação no *Visual Studio* e em seguida a transferência de arquivos para a máquina virtual. Todo o código mencionado pode ser encontrado no repositório *smartlock-backend*².

4.5 Implementação do front end

A construção do front end se iniciou por meio da pesquisa de exemplos de aplicações semelhantes à ser desenvolvida. Um dos exemplos consultados foi o *dashboard* criado pela empresa *creative-tim*.

A partir da mesma foram desenvolvidas as telas de acordo com a realidade do presente sistema.

A *landingpage*, página externa do sistema, é apresentada na Figura 4.14.

²Disponível em: <https://github.com/matheus-mac/smarlock-backend>

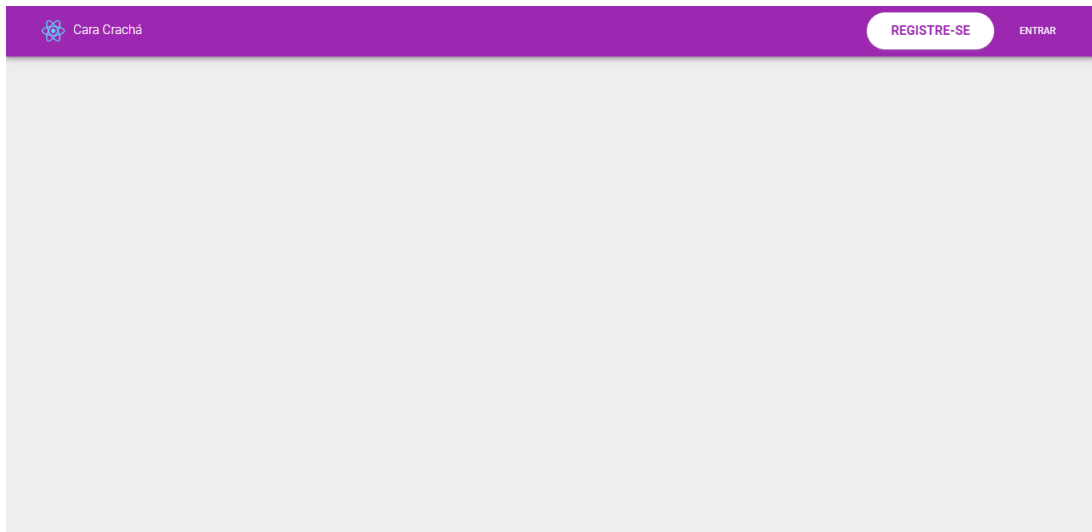


Figura 4.14: Tela inicial. Fonte: Elaborada pelo autor.

As opções de login e registro são exibidas nas figuras 4.16 e 4.15. Na tela de registro são solicitadas apenas informações essenciais e de contato afim de estimular a inserção de poucos dados. Após a confirmação da conta, via e-mail, ocorre o redirecionamento para a página de perfil, na qual os dados são completados.

Figura 4.15: Tela de registro. Fonte: Elaborada pelo autor.

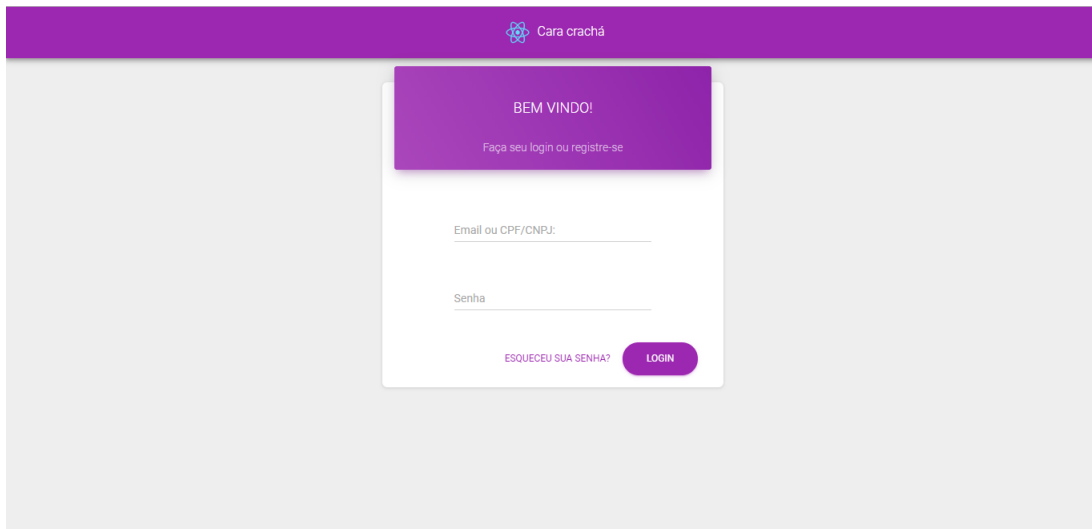


Figura 4.16: Tela de login. Fonte: Elaborada pelo autor.

Uma vez efetuado o *login* o usuário é redirecionado para a página principal, onde um *dashboard* com alguns gráficos e informações acerca das fechaduras que possui são apresentadas. Neste tela (Figura 4.17) é apresentado também o menu lateral utilizado em toda a plataforma. Nas demais imagens o mesmo será suprimido para evitar repetições.

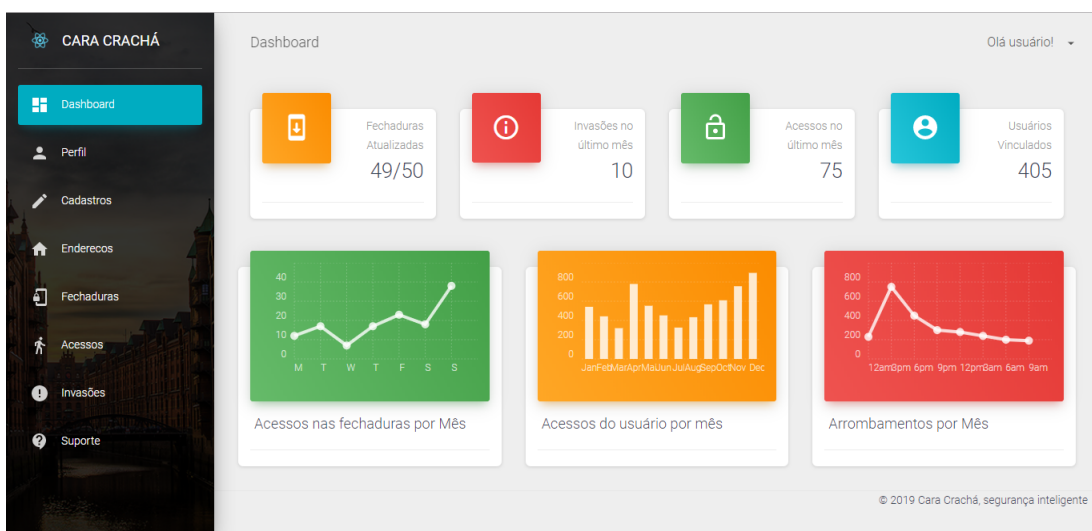


Figura 4.17: Dashboard após o login e menu lateral. Fonte: Elaborada pelo autor.

A segunda tela exibida pelo menu é a do usuário (Figura 4.18). Nela o usuário logado pode ver suas informações e modificá-las. A alteração de senhas a partir de um *pop up* nessa tela, mediante apresentação da senha atual e confirmação da nova senha informada (Figura 4.19).

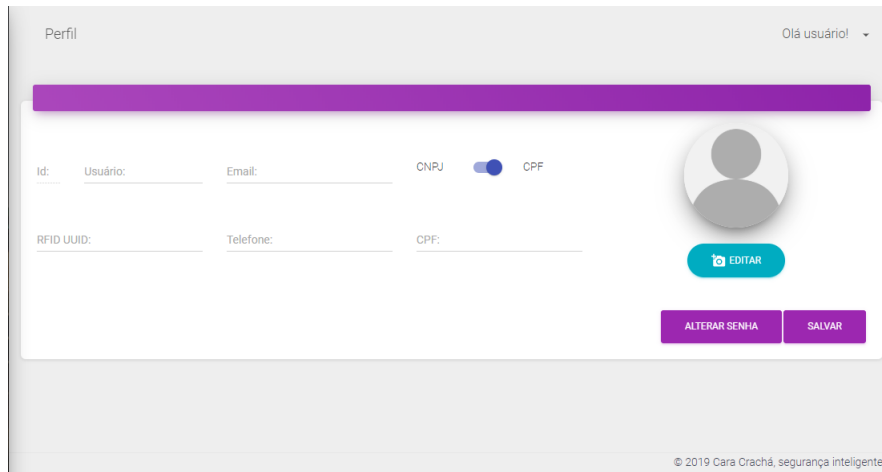


Figura 4.18: Tela de perfil do usuário. Fonte: Elaborada pelo autor.

ALTERAR SENHA

Insira sua senha antiga e confirme a nova:

Senha atual: _____

Nova senha: _____

Confirme a senha: _____

CANCELAR **ENVIAR**

Figura 4.19: Popup para alteração de senhas. Fonte: Elaborada pelo autor.

A tela de cadastro foi dividida em guias, de maneira a concentrar todos os novos cadastros em única opção. As figuras 4.20, 4.21 e 4.22 apresentam cada uma das abas.

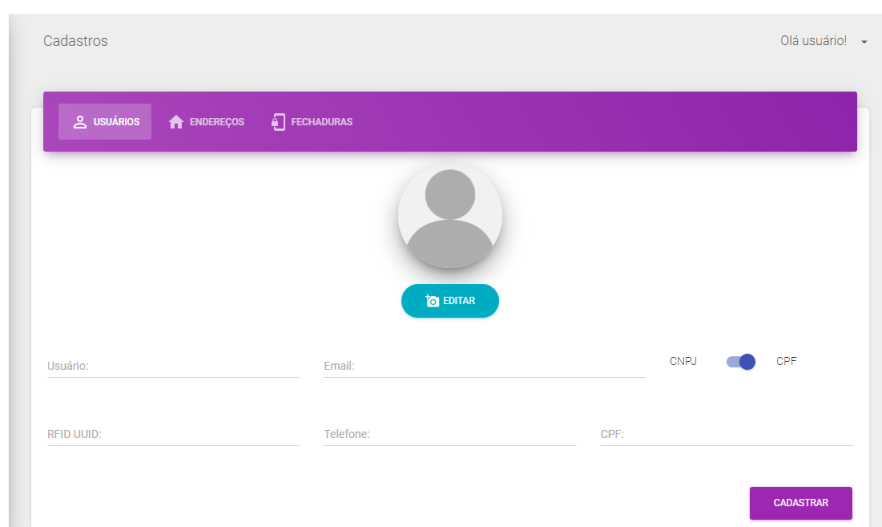


Figura 4.20: Tela de cadastros: usuário. Fonte: Elaborada pelo autor.

Cadastros Olá usuário!

USUÁRIOS ENDEREÇOS FECHADURAS

CEP: _____ Cidade: _____ Estado: _____

Rua: _____ Número: _____ Bairro: _____ Complemento: _____

CADASTRAR

© 2019 Cara Crachá, segurança inteligente

Figura 4.21: Tela de cadastros: endereços. Fonte: Elaborada pelo autor.

Cadastros Olá usuário!

USUÁRIOS ENDEREÇOS FECHADURAS

EDITAR

Usuário: _____ Email: _____ CNPJ CPF

RFID UUID: _____ Telefone: _____ CPF: _____

CADASTRAR

Figura 4.22: Tela de cadastros: usuários. Fonte: Elaborada pelo autor.

As listagens se iniciam pela apresentação dos endereços 4.23. Ao lado de cada uma das entidades é exibido um pequeno *toolbar*, permitindo modificações e exclusões, por meio de *pop up's* (Figuras 4.24 e 4.25).

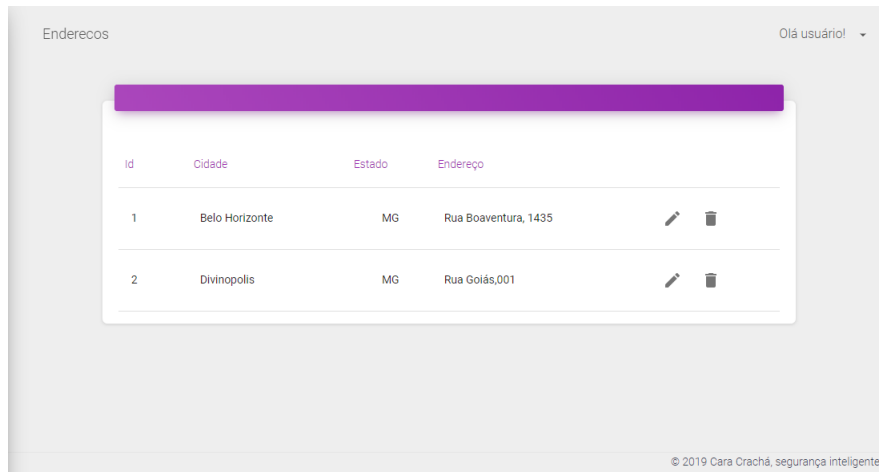


Figura 4.23: Tela de listagem de endereços do usuário. Fonte: Elaborada pelo autor.

The form is titled 'EDITAR ENDEREÇO'. It contains the following fields:

- CEP:
- Cidade:
- Estado:
- Rua:
- Número:
- Bairro:
- Complemento:

At the bottom right, there are two buttons: 'CANCELAR' and 'SALVAR'.

Figura 4.24: Tela para edição de endereços. Fonte: Elaborada pelo autor.

The dialog is titled 'DELETAR ENDEREÇO'. It contains the following text:

Esta ação não poderá ser desfeita. Caso o endereço possua alguma fechadura vinculada o mesmo não poderá ser excluído.

At the bottom, there are two buttons: 'DISCORDO' and 'CONCORDO'.

Figura 4.25: Confirmação da exclusão de um endereço. Fonte: Elaborada pelo autor.

A tela de fechaduras (Figura 4.26) exibe as informações principais sobre cada dispositivo vinculado ao usuário e oferece opções para alteração, a exemplo dos endereços. Cada um das ações é expandida entre as figuras 4.27 e 4.30.

Número Serial	Identificador Fechadura	Versão	
1	Porta da sala	1	+ 👤 ✎ 🗑
13	Porta principal	2	+ 👤 ✎ 🗑
47	Escritório	2	+ 👤 ✎ 🗑

Figura 4.26: Tela de listagem de fechaduras do usuário. Fonte: Elaborada pelo autor.

VINCULAR USUÁRIO

Insira o email ou CPF/CNPJ do usuário para vincular:

CANCELAR **VINCULAR**

Figura 4.27: Tela para adição de vínculo de usuário. Fonte: Elaborada pelo autor.

DES-VINCULAR USUÁRIO

Desvincule os usuários (a ação não pode ser desfeita)

Dakota Rice	X
Minerva Hooper	X
Sage Rodriguez	X

CANCELAR

Figura 4.28: Tela para exclusão de vínculo de usuário. Fonte: Elaborada pelo autor.

EDITAR FECHADURA

Endereço MAC: _____

Versão: _____

Data de ativação:
03/11/2019

Identificador da fechadura: _____
Esse será o nome da fechadura, selecione nomes intuitivos como: Porta da frente

Endereço: _____

CANCELAR **SALVAR**

Figura 4.29: Tela para edição de fechadura. Fonte: Elaborada pelo autor.

DELETAR FECHADURA

Esta ação não poderá ser desfeita. Todo o registro relativo à mesma será apagado.

DISCORDO **CONCORDO**

Figura 4.30: Tela de confirmação de exclusão de fechadura. Fonte: Elaborada pelo autor.

As telas de exibição de acessos (Figura 4.31) e invasões (Figura 4.32) possuem temáticas bastante semelhantes. A diferença básica entre as mesmas é a disponibilização de um filtro para listar os acessos por um usuário específico. Os campos de datas podem ser preenchidos utilizando um *date picker* disponibilizados na *material-ui* (Figura 4.33).

Acessos Olá usuário!

Filtros disponíveis:

Usuário: Fechadura: Data de início: 03/11/2019 Data de fim: 03/11/2019 **FILTRAR**

Id	Fechadura	Usuário	Data	Horário
1	Quarto	Matheus	27/10/2019	11:00:00
2	Porta da frente	Marcus Vinicius	25/10/2019	10:36:05
3	Quarto de hóspedes	Rejane	21/10/2019	9:05:00

© 2019 Cara Crachá, segurança inteligente

Figura 4.31: Tela de listagem de acessos. Fonte: Elaborada pelo autor.

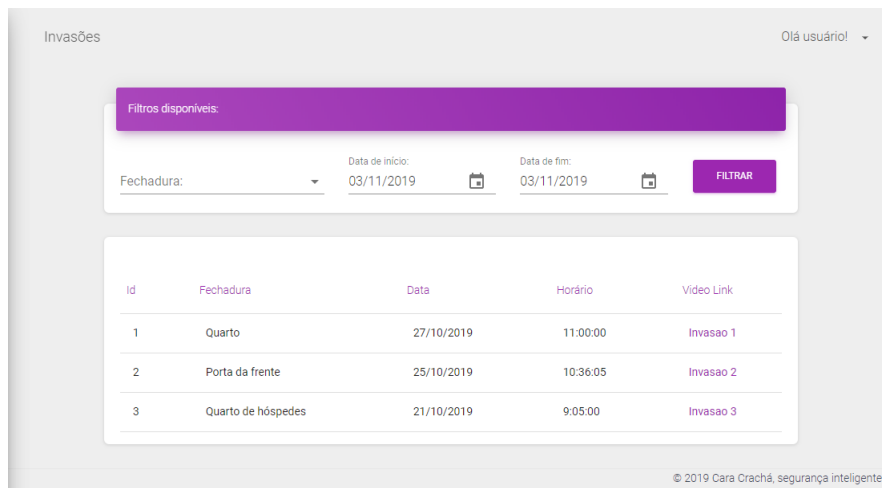


Figura 4.32: Tela de listagem de invasões. Fonte: Elaborada pelo autor.

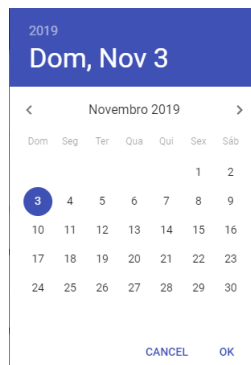


Figura 4.33: *Date Picker* utilizado . Fonte: Elaborada pelo autor.

Por fim foi adicionada uma tela de solicitações de suporte (Figura 4.34). As mesmas são enviadas via e-mail. Não sendo persistidas por meio do banco de dados.

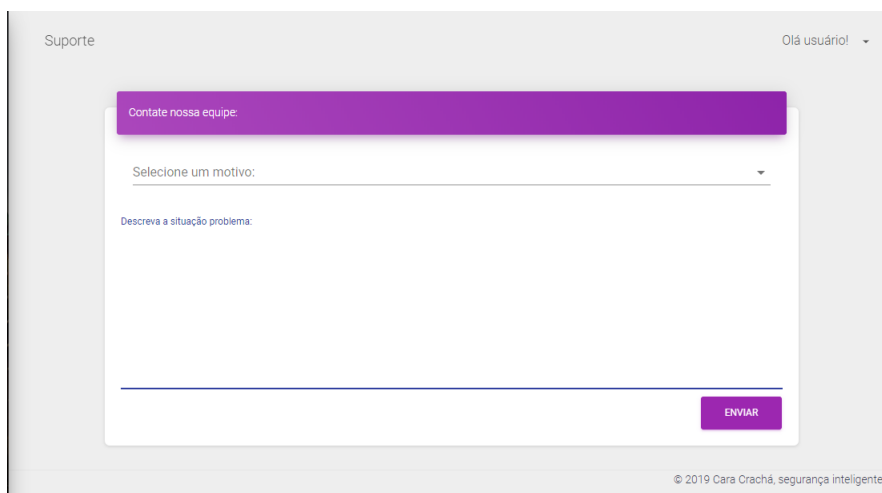


Figura 4.34: Tela de cadastro de solicitações de suporte. Fonte: Elaborada pelo autor.

Em cada ação em que é necessária a confirmação do usuário ou em casos de edição são exibidos pequenos alertas de confirmação ou erro. A aparência dos mesmos é exibida na Figura 4.35.



Figura 4.35: Alerta de confirmação de ação. Fonte: Elaborada pelo autor.

O código construído pode ser visualizado no repositório *smartlock-frontend*³. A hospedagem online do *software* foi realizada no aplicativo *netlify*. O mesmo possui comunicação direta com o *github*, realizando-o a implantação de maneira automática quando ocorrem alterações no repositório remoto. A aplicação pode ser acessada por meio do site: <https://cara-cracha.netlify.com>.

Terminada a construção foram realizados testes do sistema em funcionamento. Inicialmente foram simuladas 50 invasões: houve sucesso no registro 98% das vezes. Em uma delas foi apontado um erro, ligado à *upload*, possivelmente um momento de instabilidade.

Além destes testes desenvolveu-se 50 vezes o fluxo de liberação de acessos. Nesse modelo foram utilizados três usuários, sendo dois deles irmãos, possuidores de semelhanças físicas. Houve sucesso em 96% dos casos, sendo enviados 2 e-mails ao dono da fechadura informando que o usuário tentou o reconhecimento e não foi possível finalizá-lo. Tal resultado pode ser descrito pela incerteza do modelo e alterações na iluminação que implicam diretamente na leitura feita pela câmera.

³Disponível em: <https://github.com/matheus-mac/smarlock-frontend>

Considerações Finais

O desenvolvimento do dispositivo se mostrou extremamente facilitado pela definição precisa dos componentes. O *Raspberry Pi 3* se mostrou extremamente versátil, possuindo grande suporte. O trabalho iniciou-se com uma versão do sistema operacional e foi finalizado em outra, sendo visível a diferença de desempenho. Os periféricos interagiram de maneira satisfatória, permitindo a execução das rotinas. O *Python*, pouco conhecida pelo desenvolvedor no início da execução foi de rápido aprendizado, impressionando pelo tamanho da comunidade desenvolvedora.

Os conceitos envolvidos no desenvolvimento do *sistema* são muito específicos do contexto da Engenharia de *Software*, principalmente no que se refere à modelagem e projeto de banco de dados, não sendo apresentados sequer de maneira superficial ao longo do curso de Engenharia Mecatrônica. Foi um grande aprendizado, mesmo que ainda seja necessária mais prática para fixar os conceitos.

A disponibilização das aplicações na nuvem e todo o tratamento necessário de infraestrutura para tornar a aplicação realmente online são atividades do contexto *DevOps* e consumiram um grande tempo do aprendizado porém agregaram muito como conhecimento, gerando várias dúvidas e necessidades de suporte. Foram testadas as duas principais companhias de *cloud*: *AWS* e *Azure*, sendo a primeira definida como ferramenta final, devido ao melhor suporte.

A abertura de maneira remota foi uma *feature* levantada ao término do primeiro semestre de execução. Ao avaliar tecnicamente encontrou-se soluções parecidas, utilizando

servidores embarcados como *Apache* e o *Flask*, sendo realizados inclusive testes com a segunda tecnologia. Entretanto, após consultas com empresas que utilizam a ferramenta viu-se que a mesma é utilizada para construção de *web services*, não atendendo a demanda. O tópico será incluído na proposição de trabalhos futuros.

Ao findar o trabalho iniciou-se o contato com empresas do ramo de segurança para avaliação do desenvolvimento do protótipo para um lançamento do produto. As análises ainda são preliminares, porém promissoras acerca do mesmo.

5.1 Propostas de continuidade

- Adicionar bateria recarregável para minimizar problema de falta de energia
- Implementação da abertura remota por meio do sistema;
- Inclusão de flag indicando se a porta está aberta ou fechada no banco de dados;
- Adição de alertas via SMS;
- Permitir a customização do dashboard inicial de acordo com o usuário;
- Incorporar o registro de suportes no banco de dados, estabelecendo um *helpdesk* mais efetivo.

Referências

- [1] R. F. Zampolo A. T. A. Gusmão, A. L. S. Castro. Fechadura baseada em reconhecimento facial via dispositivos móveis android. *SBT*, 2016.
- [2] F. Werneck L. N. Leal. Ibge: 11,9 milhões foram vítimas de roubo em 1 ano. Online, Março 2018. Disponível em: <http://www.estadao.com.br/noticias/geral,ibge-11-9-milhoes-foram-vitimas-deroubo-em-1-ano,653922> Acessado em 05 Mar 2018.
- [3] Mais de 1.3 milhão de brasileiros engajados na construção de um país mais seguro. Online, abril 2018. Disponível em <<http://blog.ondefuiroubado.com.br/o-projeto/>> Acessado em: 04 Mar 2018.
- [4] B. Miranda N. Oliveira. Arrombamentos de residências aumentam em 66 Online, março 2018. Disponível em: <http://www.otempo.com.br/cidades/arrombamentos-de-resid%C3%Aanciasaumentam-66-em-bh-1.1304725> Acessado em 05 Mar 2018.
- [5] Fechadura de porta com reconhecimento facial. Online, abril 2018. Disponível em <<http://www.recognition-systems.com.br/1-7-facialrecognition-door-lock.html>> Acessado em: 02 Mar 2018.
- [6] Submarino. Fechadura biometria facial - reconhecimento facial - fl400 - inttellig, março 2018. Disponível em: <https://goo.gl/XCPu1J> Acessado em: 03 Mar 2018.

-
- [7] YALE Brasil. As origens da fechadura com cilindro. online, Outubro 2017. Disponível em: <https://www.yale.com.br/pt-br/yale/home/sobre-a-yale/nossa-marca/a-fechadura-de-cilindro/> Acessado em: 01 Maio 2018.
- [8] Brendan Quinn. The history of locks - from ancient egypt to the present day. online, Dezembro 2017. Disponível em: <https://www.telegraph.co.uk/property/home-improvement-tips/history-of-locks/> Acessado em: 04 Maio 2018.
- [9] Casa das Chaves. História das chaves e fechaduras. online, Junho 2016. Disponível em: <https://chavesefechaduras.com.br/historia-das-chaves-e-fechaduras/> Acessado em: 05 Maio 2018.
- [10] Floor 2 Alley 2 Lane 437 Nei-Hu Rd. Sec. 1 Nei-Hu Dist. Taipei TW) Hsu, Yuntung (No. 9. Flexible key and lock assembly, February 1992. Patente US5086632. Disponível em: <https://patents.google.com/patent/US5086632/en> Acesso em: 08 Maio 2018.
- [11] YALE United Kingdom. Conexis l1 smart door lock. online, Julho 2018. Disponível em: <https://www.yale.co.uk/en/yale/couk/products/smart-living/smart-door-locks/conexis-l1-smart-door-lock/> Acessado em: 01 Maio 2018.
- [12] P. Timse et al. Face recognition based door lock system using opencv and c# with remote access and security features. *Journal Of Engineering Research And Applications*, 2014.
- [13] V. Nascimento. Implementação de um sistema de identificação facial utilizando linux embarcado. *USP*, 2015.
- [14] E. D. Tramontin. Análise e aplicação de reconhecimento facial em sistema embarcado. *UFSC*, 2016.
- [15] T. S. Gunawan et. al. Development of face recognition on raspberry pi for security enhancement of smart home system. *ijeei*, 2017.

-
- [16] Fechadura biométrica com reconhecimento facial - face lock - com *Wifi*. Online, abril 2018. Disponível em <<https://www.fxbiometria.com.br/fechadura-biometrica-com-reconhecimentofacial-face-lock-com-wifi.html>> Acessado em: 22 Abr 2018.
- [17] MADIS. Biometria reconhecimento pessoal. Online, junho 2007. Disponível em: <http://www.madis.com.br/produtos/biometria-reconhecimento-facial> Acessado em 20 Abr 2018.
- [18] Otavio Chase and FJ Almeida. Sistemas embarcados. *Mídia Eletrônica. Página na internet*:< www.sabajovem.org/chase>, capturado em, 10(11), 2007.
- [19] P. Cipoli. Saiba tudo sobre o raspberry pi 3 e o que ele representa para o mercado, Março 2016. Disponível em: <https://canaltech.com.br/hardware/saiba-tudo-sobre-o-raspberry-pi-3-59065/> Acessado em 28 Maio 2018.
- [20] Raspberry Pi Foundation. Raspberry pi documentation, 2018. Disponível em: <https://www.raspberrypi.org/documentation/> Acessado em 01 Junho 2018.
- [21] G. C. Lopes C. B. Silveira. O que é rfid?, 2018. Disponível em: <https://www.citisystems.com.br/rfid/> Acessado em 05 Abril 2018.
- [22] Klaus Finkenzerler. Rfid handbook: Fundamentals and applications in contactless smart cards and identification, 2nd. *New York: John Wiley & Sons Ltd*, 2003.
- [23] NXP Semiconductors. Mfrc522 standard performance mifare and ntag frontend, Abril 2016. Disponível em: <https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf> Acessado em 03 Junho 2018.
- [24] FILIPEFLOP. Kit módulo leitor rfid mfrc522 mifare, Junho 2017. Disponível em: <https://www.filipeflop.com/produto/kit-modulo-leitor-rfid-mfrc522-mifare/> Acessado em 03 Junho 2018.
- [25] Gilleanes TA Guedes. *UML 2-Uma abordagem prática*. Novatec Editora, 2018.

-
- [26] Amazon Web Services. What is aws? Disponível em: <https://aws.amazon.com/pt/what-is-aws/> Acessado em 01 Outubro 2019.
- [27] Amazon Web Services. Amazon relational database service (rds). Disponível em: <https://aws.amazon.com/pt/rds/> Acessado em 01 Outubro 2019.
- [28] Amazon Web Services. Amazon s3. Disponível em: <https://aws.amazon.com/pt/s3/> Acessado em 01 Outubro 2019.
- [29] Aleksader Knabben Becker, Daniela Barreiro Claro, and João Bosco Sobral. Web services e xml um novo paradigma da computação distribuída. *Objetos Distribuídos*, pages 51–66, 2001.
- [30] Apple Inc. About cloudkit web services, 2019.
- [31] Claudimir Zavalik, Guilherme Lacerda, and José Palazzo M de Oliveira. Implementando web services com software livre. *Prefácio 5º WSL*, page 35, 2004.
- [32] Carlos Alberto Heuser. *Projeto de banco de dados: Volume 4 da Série Livros didáticos informática UFRGS*. Bookman Editora, 2009.
- [33] Microsoft. Documentação do sql server, Agosto 2019. Disponível em: <https://docs.microsoft.com/pt-br/sql/sql-server/sql-server-technical-documentation?view=sql-server-2017> Acessado em 30 Agosto 2019.
- [34] Amazon Web Services. Amazon rds for sql server, 2019. Disponível em: <https://aws.amazon.com/pt/rds/sqlserver/> Acessado em 31 Agosto 2019.
- [35] *Data Science Academy*. O que é visão computacional? Online, Janeiro 2017. Disponível em: <http://datascienceacademy.com.br/blog/o-que-e-visao-computacional/>. Acessado em: 15 Maio 2018.
- [36] Erik Hjelmås and Boon Kee Low. Face detection: A survey. *Computer vision and image understanding*, 83(3):236–274, 2001.

-
- [37] M. L. Pereira. Projeto e construção de um sistema embarcado para monitoramento de processos industrial utilizando visão computacional. *CEFET-MG*, 2017.
- [38] Danilo MILANO and Luciano Barrozo HONORATO. Visão computacional. *Universidade Estadual de Campinas-(Unicamp), Faculdade de Tecnologia*, 2010.
- [39] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. "O'Reilly Media, Inc.", 2008.
- [40] OpenCV. Opencv documentation. online, Dezembro 2016. Disponível em: <https://opencv.org/>. Acessado em 20 Maio 2018.
- [41] Maurício Marengoni and Stringhini Stringhini. Tutorial: Introdução à visão computacional usando opencv. *Revista de Informática Teórica e Aplicada*, 16(1):125–160, 2009.
- [42] André Luiz Barbosa Nunes da Cunha. *Sistema automático para obtenção de parâmetros do tráfego veicular a partir de imagens de vídeo usando OpenCV*. PhD thesis, Universidade de São Paulo, 2013.
- [43] Luis Renato Cruz Erpen. Reconhecimento de padrões em imagens por descritores de forma. 2004.
- [44] Andrews Sobral and Antoine Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122:4–21, 2014.
- [45] Ogê Marques Filho and Hugo Vieira Neto. *Processamento digital de imagens*. Brasport, 1999.
- [46] Antonio Escaño Scuri. Fundamentos da imagem digital. *Pontifícia Universidade Católica do Rio de Janeiro*, 1999.
- [47] F Mário Martins. Métodos formais na concepção e desenvolvimento de sistemas interactivos. 1995.

-
- [48] NodeJS. About nodejs. Disponível em: <https://nodejs.org/en/about/> Acessado em 01 Outubro 2019.
- [49] ReactJS. Reactjs. Disponível em: <https://pt-br.reactjs.org> Acessado em 01 Outubro 2019.
- [50] Material UI Team. Material ui. Disponível em: <https://material-ui.com/> Acessado em 01 Outubro 2019.
- [51] AxiosJS. Promise based http client for the browser and node.js. Disponível em: <https://github.com/axios/axios> Acessado em 01 Outubro 2019.
- [52] Netlify. Netlify. Disponível em: <https://www.netlify.com/> Acessado em 01 Outubro 2019.
- [53] John Deacon. Model-view-controller (mvc) architecture. *Online* [Citado em: 10 de março de 2006.] <http://www.jdl.co.uk/briefings/MVC.pdf>, 2009.
- [54] Glenn E Krasner, Stephen T Pope, et al. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3):26–49, 1988.
- [55] Edward Curry and Paul Grace. Flexible self-management using the model-view-controller pattern. *IEEE software*, 25(3):84–90, 2008.
- [56] FILIPEFLOP. Raspberry pi 3 - model b, Junho 2017. Disponível em: <https://www.filipeflop.com/produto/raspberry-pi-3-model-b/> Acessado em 08 Maio 2018.
- [57] Mouser. Beagle black bone, Junho 2016. Disponível em: <https://br.mouser.com/new/beagleboardorg/beaglebone-blackwireless/> Acessado em 08 Maio 2018.
- [58] Raspberry PI Foundation. Raspberry pi gpio pinout, 2016. Disponível em: <https://pinout.xyz/> Acessado em 30 Maio 2018.

- [59] Cleiton Bueno. Rfid com raspberry e python, Novembro 2014. Disponível em: <https://www.embarcados.com.br/rfid-raspberry-pi-python/> Acessado em 25 Maio 2018.