INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA – CÂMPUS FLORIANÓPOLIS

DEPARTAMENTO ACADÊMICO DE METAL-MECÂNICA

CURSO DE GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

MANOEL VITOR DE SOUSA

SOFTWARE DE CONTROLE DE ACESSO VEICULAR POR IMAGENS

FLORIANÓPOLIS, 2024.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA – CÂMPUS FLORIANÓPOLIS

DEPARTAMENTO ACADÊMICO DE METAL-MECÂNICA

CURSO DE GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

MANOEL VITOR DE SOUSA

SOFTWARE DE CONTROLE DE ACESSO VEICULAR POR IMAGENS

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Engenheiro em 2024.

Orientador:
Prof. Valdir Noll, Dr. Eng.

FLORIANÓPOLIS, 2024.

Ficha de identificação da obra elaborada pelo autor.

Sousa, Manoel
Software de Controle de Acesso Veicular por Imagens / Manoel Sousa; orientação de Valdir Noll. – Florianópolis, SC, 2024.
51 p.

Trabalho de Conclusão de Curso (TCC) – Instituto Federal de Santa Catarina, Câmpus Florianópolis. Bacharelado em Engenharia Mecatrônica. Departamento Acadêmico de Metal Mecânica.
Inclui Referências.

1. Controle de acesso. 2. Reconhecimento de placas veiculares. 3. Segurança condominial. I. Noll, Valdir. II. Instituto Federal de Santa Catarina. III. Software de Controle de Acesso Veicular por Imagens.

SOFTWARE DE CONTROLE DE ACESSO VEICULAR POR IMAGENS

MANOEL VITOR DE SOUSA

Este trabalho foi julgado adequado para obtenção do título de Engenheiro em Mecatrônica e aprovado na sua forma final pela banca examinadora do Curso Engenharia Mecatrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 26 de agosto de 2024.

Banca Examinadora:

gov.br Documento assinado digitalmente
VALDIR NOLL
Data: 23/09/2024 20:48:24-0300
Verifique em https://validar.iti.gov.br

Valdir Noll, Dr.

gov.br Documento assinado digitalmente
MAURICIO EDGAR STIVANELLO
Data: 24/09/2024 07:01:44-0300
Verifique em https://validar.iti.gov.br

Maurício Edgar Stivanello, Dr.

gov.br Documento assinado digitalmente
DELCINO PICININ JUNIOR
Data: 24/09/2024 14:05:05-0300
Verifique em https://validar.iti.gov.br

Delcino Júnior Picinin, Dr.

RESUMO

O presente Trabalho de Conclusão de Curso, apresenta o desenvolvimento de um sistema de controle de acesso veicular integrado a câmeras IPs . O sistema acessa uma *stream* de vídeo das câmeras com uso do protocolo RTSP para realizar a captura das imagens e o processamento das mesmas. O objetivo é realizar o reconhecimento de veículos, placas veiculares brasileiras e a leitura dos caracteres, utilizando modelos de visão computacional pré-treinados.

Para acionamento remoto do portão veicular e leitura de sensores de estado (indicando abertura e fechamento do portão), utilizou-se uma ESP 32 com conexão Wi-Fi, comunicando-se com o software desenvolvido (VehiKey) através do protocolo MQTT. Além disso, é possível a integração com sistemas terceiros por meio de API, permitindo o cadastro de veículos, configurações de horários, bem como a conexão a *stream* de eventos, em tempo real.

O desenvolvimento do sistema seguiu as boas práticas e conceitos definidos pelos livros “Clean Architecture” e “Clean Code” escritos por Robert Cecil Martin, assegurando uma organização arquitetural conceituada e elevando as chances de escalabilidade do projeto serem mais assertivas.

Os testes abordados no final do capítulo demonstram a capacidade do software em identificar veículos, placas veiculares, caracteres e até cores, além de aplicar determinadas regras de negócios para liberar ou não o acesso do veículo, aumentando a segurança do condomínio e reduzindo a necessidade de intervenção humana. O projeto desenvolvido provou ser uma solução viável para o controle de acesso veicular, necessitando de alguns ajustes para tornar algo comerciável, oferecendo benefícios significativos em termos de segurança e automação para condomínios residenciais.

Palavras-chave: Controle de acesso. Reconhecimento de placas veiculares. Segurança condominial.

ABSTRACT

This final paper presents the development of a vehicle access control system integrated with IP cameras. The system accesses a video stream from the cameras using the RTSP protocol to capture and process images. The objective is to recognize vehicles, Brazilian vehicle license plates, and read characters using pre-trained computer vision models.

For remote gate operation and reading of state sensors (indicating the opening and closing of the gate), an ESP32 with Wi-Fi connection was used, communicating with the developed software (VehiKey) through the MQTT protocol. Additionally, integration with third-party systems is possible through an API, allowing the registration of vehicles, schedule settings, and connection to a real-time event stream.

The system's development followed the best practices and concepts defined by the books “Clean Architecture” and “Clean Code” written by Robert Cecil Martin, ensuring a well-conceived architectural organization and increasing the likelihood of the project's scalability being more assertive.

The tests addressed at the end of the chapter demonstrate the software's ability to identify vehicles, license plates, characters, and even colors, as well as apply certain business rules to grant or deny vehicle access, enhancing the condominium's security and reducing the need for human intervention. The developed project proved to be a viable solution for vehicle access control, requiring some adjustments to make it marketable, offering significant benefits in terms of security and automation for residential condominiums.

Keywords: Access control. Vehicle license plate recognition. Condominium security.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CRUD	*Create, Read, Update e Delete*
IFSC	Instituto Federal de Santa Catarina
IP	*Internet Protocol*
MQTT	*Message Queuing Telemetry Transport*
OCR	*Optical Character Recognition*
ORM	*Object-Relational Mapping*
RTSP	*Real Time Streaming Protocol*
SOLID	*Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion*
UI	*User Interface*
VEDA	*Vertical Edge Detection Algorithm*
Wi-Fi	*Wireless Fidelity*
YOLO	*You Only Live Once*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, a tecnologia tem desempenhado um papel crucial na transformação de diversos setores, incluindo o residencial, onde a segurança é uma prioridade inegável. Neste contexto, o controle de acesso veicular surge como uma necessidade premente, especialmente em condomínios, onde a gestão eficiente do fluxo de veículos é essencial para garantir a facilidade de acesso, segurança e a tranquilidade dos moradores. Este trabalho apresenta o desenvolvimento de um software com integração a hardwares comerciais, que visa automatizar o controle de acesso veicular em condomínios através da análise de *streams* de vídeo de câmeras IPs.

Utilizando o protocolo RTSP (Real Time Streaming Protocol), o sistema é capaz de identificar veículos através da detecção de placas veiculares em tempo real com uso de bibliotecas de softwares. Ao reconhecer uma placa, o software embarcado analisa se o veículo possui autorização de acesso. Em caso afirmativo, o sistema automaticamente comanda a abertura do portão, permitindo a entrada do veículo. Este processo não apenas agiliza o acesso, como também reduz a necessidade de intervenção humana, minimizando erros e aumentando a segurança do perímetro.

Além de realizar o controle de acesso, o sistema proposto oferece uma interface para integração com outros softwares de empresas parceiras. Essa funcionalidade amplia as possibilidades de uso do sistema, permitindo, por exemplo, o monitoramento do status do portão (aberto, fechado, etc.), o envio de comandos para ação remota (como a abertura do portão ou cadastro de novos veículos) e a geração de alertas para situações específicas, contribuindo para uma gestão mais eficaz do ambiente condominial.

Este projeto não apenas visa melhorar a segurança e a eficiência nos processos de controle de acesso veicular em condomínios, mas também propõe uma solução econômica e facilmente integrável a sistemas existentes, trazendo inovação e valor agregado aos gestores de condomínios e moradores.

1.1 Justificativa

Conforme analisado por Meirelles (2019, p. 8), praticamente todos os casos de invasões ocorrem pela entrada do condomínio, com os invasores em sua maioria, caracterizados de entregadores ou prestadores de serviço.

Este fato, muito se deve ao depender exclusivamente da verificação de acessos realizada por funcionários humanos, frequentemente sobrecarregados com múltiplas responsabilidades, é possível que, em certos cenários, não sejam exigidos todos os documentos comprobatórios necessários para a liberação de acesso, conforme as normas de segurança do condomínio. Falhas como essas, plausíveis e recorrentes em serviços humanos, demonstram a necessidade de implementar soluções automatizadas que garantam a conformidade e a eficácia dos procedimentos de segurança.

A substituição de funções metódicas de controle de acesso, como as descritas anteriormente, por sistemas automatizados, aumenta significativamente a confiabilidade no cumprimento das normas estabelecidas. Além disso, quando esses sistemas são integrados a tecnologias de alarme para casos de acessos não autorizados, a rapidez e a eficiência na resposta a tais incidentes podem ser consideravelmente melhoradas.

1.2 Definição do Problema

O problema identificado envolve a vulnerabilidade dos sistemas de controle de acesso em condomínios residenciais, que atualmente dependem largamente da intervenção humana. A análise de dados e estudos prévios indica que a maior parte das invasões a condomínios ocorre através das entradas principais, como relatado por Meirelles (2019, p. 8).

Godoy (2018, 5° edição) comenta sobre os principais modos de acessos usado por ladrões para cometer crime, dentre eles estão:

1. Entrar no “vácuo” deixado pela passagem de um outro morador pelo portão veicular;

2. Entrar a pé pelo portão veicular juntamente com o acesso de um veículo de um morador;
3. Buzinar em frente ao portão veicular, fingindo que esqueceu a chave, e forçando o porteiro local abrir manualmente o portão;

Essas falhas no processo de verificação não apenas facilitam acessos indevidos, mas também colocam em risco a segurança dos moradores e a integridade do patrimônio dentro do condomínio. Portanto, torna-se imperativo desenvolver um sistema de controle de acesso mais eficaz e confiável que minimize a dependência de intervenções humanas e aumente a segurança através da automação e integração de tecnologias avançadas.

Além dos quesitos de segurança, manter no mínimo três porteiros para cobrir o ciclo completo de 24 horas representa um custo significativo para os condôminos. Adicionalmente, o uso de tecnologias antigas para abertura veicular, como receptores de RF, exige que cada morador possua pelo menos um controle RF, incrementando ainda mais os custos envolvidos. Controles genéricos podem ser facilmente clonados por criminosos, facilitando o acesso indevido às dependências do condomínio a qualquer momento, o que representa um ponto crítico de atenção.

Outro problema observado é o arrombamento de portões durante o período noturno, que muitas vezes pode passar despercebido, mesmo com a vigilância por câmeras. A visibilidade reduzida e a falta de alertas automáticos dificultam a detecção imediata desses incidentes por parte dos porteiros, comprometendo a segurança de todos os moradores.

1.3 Objetivo Geral

O objetivo geral do presente trabalho foi desenvolver um software de controle de acesso veicular automatizado, em conjunto com câmeras comerciais IP, que possuem o protocolo RTSP e microcontroladores para acionamento de portões e leitura de sensores.

Dado o reconhecimento do veículo através dos caracteres da placa e cor majoritária, é executado regras de verificação de acesso. Tudo isso, registrado em

um banco de dados e relatado através de eventos para os clientes conectados no software.

Além disto, o software permite configurações de diferentes maneiras, seja de forma direta, via interface WEB, ou através de API, em casos onde existe um software de monitoramento e deseja-se integrar com o software aqui desenvolvido.

1.4 Objetivos Específicos

Para alcançar o objetivo geral deste projeto, foram estabelecidos os seguintes objetivos específicos:

1. Implementar a Integração com Câmeras IP: Configurar o software para se conectar com câmeras IP comerciais que utilizam o protocolo RTSP, garantindo a captura de vídeo para análise.
2. Realizar reconhecimento de caracteres e cor: Criar e otimizar algoritmos para detectar veículos, identificar caracteres de placas e a cor predominante dos veículos utilizando bibliotecas prontas de visão computacional.
3. Estabelecer Sistema de Verificação de Acesso: Projetar um mecanismo que automaticamente verifique as informações capturadas contra um conjunto de regras de acesso predefinidas para determinar a autorização de entrada.
4. Gerenciar Dados e Eventos: Construir um sistema para o armazenamento seguro de dados em um banco de dados e para a geração e gerenciamento de eventos relacionados ao acesso veicular.
5. Desenvolver Interfaces de Usuário e API: Elaborar uma interface web intuitiva para a administração do sistema e uma API robusta para permitir a integração do software com outras plataformas.

Estes objetivos específicos visam assegurar que o sistema de controle de acesso veicular seja eficaz, seguro e flexível, atendendo às necessidades de modernização e automação de processos em condomínios.

1.5 Estrutura do Trabalho

Este trabalho de conclusão de curso está estruturado em cinco capítulos principais, cada um trazendo aspectos fundamentais do desenvolvimento e implementação do sistema de controle de acesso veicular:

1. Introdução

- Apresentação dos objetivos do projeto e justificativa para sua realização, destacando a importância da automação e segurança em condomínios residenciais.

2. Fundamentação Teórica

- Exploração de conceitos fundamentais como reconhecimento de imagem, OCR (Optical Character Recognition), e a aplicação da arquitetura Clean Architecture no desenvolvimento de software. Destaque para a biblioteca YOLO utilizada para detecção de veículos e placas veiculares.

3. Metodologia

- Definição detalhada dos requisitos do sistema, explicação da implementação da Clean Architecture no *back end*, uso do banco de dados MySQL, geração de eventos em tempo real e regras de controle de acesso. Entendimento da lógica sobre a integração dos hardware, como ESP 32 para automação do portão e monitoramento de sensores e câmera ao software.

4. Resultados

- Apresentação dos resultados obtidos durante os testes e validação do sistema, avaliando sua eficiência na identificação e autorização de veículos. Análise dos benefícios proporcionados em termos de segurança e automação para condomínios residenciais e viabilidade comercial do projeto.

5. Considerações finais

- Conclusões finais sobre o desenvolvimento do sistema, suas contribuições para a área de controle de acesso veicular e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica deste trabalho aborda conceitos, tecnologias e estudos prévios que formam a base necessária para o desenvolvimento do sistema de controle de acesso veicular automatizado com base em reconhecimento de placa veicular. Este capítulo é dividido em várias seções, cada uma dedicada a explicar detalhadamente os aspectos relevantes do projeto:

2.1 Estado da Arte

Atualmente, é possível encontrar diversos artigos relacionados a aplicações que utilizam o reconhecimento dos caracteres de placas veiculares.

Dentre eles, temos um exemplo do artigo de Mohd Amri, Mohamed Rehan Karim e Ahmad Abdullah Saifizul, intitulado "*Travel Time Measurement in Real-Time using Automatic Number Plate Recognition for Malaysian Environment*" (2009), onde utiliza-se de câmeras em rodovias para realizar o processamento de identificação da placa veicular e confrontar com dados já existentes para cálculo de tempo da viagem. Destaca-se que sua vantagem é dada pela não necessidade de dispositivos adicionais no veículo para calcular o tempo de viagem em determinados trajetos.

Outra aplicação, é dada pela identificação de origem e destino de veículos com base em reconhecimento da placa do veículo, conforme descrito no artigo "*Optimal Locations of License Plate Recognition to Enhance the Origin-Destination Matrix Estimation*" (2012) por Yu-Chiun Chiou, *et al*.

A lógica utilizada para fazer o processamento da imagem, segue-se em base os seguintes passos:

1. Adquirir imagem
2. Extrair a placa veicular
3. Isolar os caracteres
4. Identificar os caracteres

de Muhammad Sarfraz, *et al* (2003), na maior parte dos sistemas de reconhecimento de placa veicular atualmente.

Para a extração da placa veicular propriamente dita (passo 2), existem algumas técnicas utilizadas hoje em dia.

Métodos de detecção de borda para encontrar retângulos, como demonstrado no artigo feito por Bai Hongliang and Liu Changping (2004) o qual utilizaram equações matemáticas e também por Majid Ziaratban, *et al* (2007), utilizando a

máscara Sobel, onde ambos mostrou-se mais eficaz o uso de detector por borda vertical, pois consegue eliminar diversos traços ruidoso que estão presentes em um veículo.

Com base nestas técnicas, surgiu uma nova alternativa com algoritmos de detecção de borda vertical (VEDA), onde são realizados diversos pré-processamentos para tentar atingir uma melhor performance que os métodos descritos anteriormente. Abbas M. Al-Ghaili, *et al* (2008), demonstraram em seu artigo um VEDA com performance superior a máscara de Sobel utilizada até então na maioria dos casos, sendo de 7 a 9 vezes mais rápido.

Shan Du, *et al*, em seu artigo citam outras maneiras de localizar uma placa veicular, tais qual, extração de placa veicular usando informações globais da imagem, usando características de textura, usando características de cor e usando características de caracteres.

Um resumo das características, prós e contras podem ser observados na tabela abaixo.

Tabela 1 - Resumo de métodos de extração de placa veicular

Metódos	Justificativa	Prós	Contras
Usando recursos de limite	o limite da placa do carro é retangular	mais simples, rápido e direto	dificilmente pode ser aplicado a imagens complexas, pois são muito sensíveis a bordas indesejadas
Usando recursos de imagens globais	encontre um objeto conectado cuja dimensão seja semelhante a uma placa de veículo.	direto, independente da posição da placa de veículo	pode gerar objetos quebrados
Usando recursos de texturas	transição frequente de cor em placa de veículo.	capaz de detectar mesmo se o contorno estiver deformado	computacionalmente complexo quando há muitas bordas
Usando recursos de cores	cor específica na placa de veículo.	capaz de detectar placas de veículo inclinadas e deformadas	RGB é limitado às condições de iluminação, HLS é sensível ao ruído
Usando recursos de caracteres	devem haver caracteres na placa de veículo	robusto a rotação	demorado (processando todos os objetos binários), produz erros de detecção quando há outro texto na imagem
Usando dois ou mais recursos	a combinação de múltiplos recursos é mais efetivo	mais confiável	computacionalmente complexo

Fonte: Adaptado de Shan Du, *et al* (2013).

Para segmentação dos caracteres, passo 3, é comum realizar um pré processamento para retirar algumas características indesejadas repassadas pelo passo anterior, como inclinação e brilho não uniforme.

Por exemplo, a binarização da imagem é comumente utilizada como pré-processamento com objetivo de eliminar os ruídos gerados pela iluminação, como abordado por Xiangjian He, et al em seu artigo de 2008.

Na etapa de segmentar os caracteres temos alguns métodos comumente utilizados, um deles é a segmentação por conectividade de pixels. Onde, após a binarização da imagem, é feito diversas linhas na vertical onde todos os pixels presentes nessa linha, devem ser de uma determinada cor de interesse, no intuito de criar diversos “blocos”, onde cada bloco deve representar um caractere, tal método foi utilizado no trabalho feito por Xiangjian He, et al e também por diversos outros autores indicados por Shan Du, et al. Além de indicar diversos autores que utilizaram do método por conectividade, mostrou um segundo método muito utilizado, usando perfis de projeção.

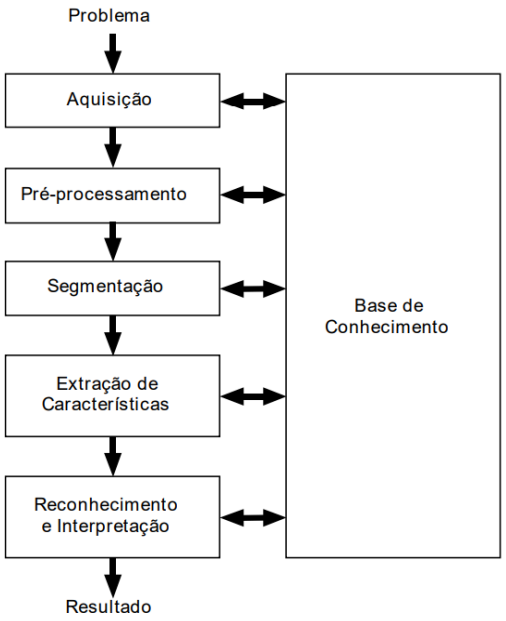
Finalmente, para a detecção dos caracteres, a técnica mais comum é a comparação de modelo, onde se mede a similaridade entre o caractere e o modelo. Algumas abordagens diferentes para medir essa similaridade incluem as distâncias de Mahalanobis, Hausdorff e Hamming, como mencionadas por Shan Du, *et al.*

2.2 Reconhecimento de Imagem

O reconhecimento de imagem está em constante evolução desde seus primórdios em 1964, conforme magnificamente descrito por Marques Filho e Vieira Neto (1999) em seu livro "Processamento Digital de Imagens", tal qual permeia hoje os mais diversos setores, impulsionando avanços em áreas como medicina, biologia, geociências, arqueologia, indústria, segurança e muito mais.

Os autores supracitados também comentam sobre uma estrutura primordial para um sistema de visão artificial (SVA), representada na Figura 1.

Figura 1 - Um sistema de visão artificial (SVA) e suas principais etapas



Fonte: Marques Filho e Vieira Neto (1999).

2.2.1 Aquisição de dados

A fase de aquisição de dados é fundamental no processo de reconhecimento de imagem, pois é neste momento que as imagens são coletadas para posterior análise. A qualidade dos dados adquiridos nesta etapa tem um impacto direto na eficácia do processamento subsequente. É crucial garantir que as imagens capturadas sejam de alta qualidade, contando com boa iluminação e nitidez adequada. Isso permite uma análise mais precisa e eficiente, reduzindo a possibilidade de erros na interpretação dos dados pelo sistema de visão computacional.

2.2.2 Pré-processamento de dados

Conforme dito por Ricardo Andrade (2016) em seu artigo, o ruído pode ser definido como qualquer elemento que interfira na capacidade de um sistema de reconhecimento de padrões de realizar sua função primordial. O pré-processamento é essencial para mitigar esses efeitos adversos, podendo também, aprimorar certas características importantes dos dados antes de serem processados pelo sistema de

reconhecimento. Este passo é crucial para assegurar que o sistema possa operar de maneira mais eficaz e precisa.

As operações realizadas nesse procedimento, são consideradas de baixo nível, dado o fato que trabalham diretamente com os valores de intensidade dos pixels e não possuem nenhuma contextualização sobre o que se trata as diversas sequências de bytes.

2.2.3 Segmentação de dados

A segmentação de dados, como o próprio nome se refere, tem como objetivo dividir a imagem recebida em unidades de interesse que a compõem.

Exemplificando em um cenário de reconhecimento de placa veicular em controle de acesso, no primeiro momento tenta-se localizar o veículo, para posteriormente encontrar um objeto retangular (placa), e por fim, trabalhando em cima dessa sub imagem gerada, é feito a segmentação de cada caracter da placa.

2.2.4 Extração de características

Nesta etapa, existe a transformação da imagem em um conjunto de dados relativos à entrada. O foco se encontra em extrair atributos das sub imagens geradas no passo anterior, com a utilização de descritores capazes de identificar de forma precisa cada caractere e diferenciar eficazmente dígitos similares, como '5' e '6', como citado por Marques Filho e Vieira Neto (1999).

2.2.5 Reconhecimento e Interpretação dos dados

Nesse último processamento das informações recebidas do tópico acima, a parte do reconhecimento, Marques Filho e Vieira Neto (1999) relatam que é responsável pela atribuição de um rótulo ao objeto baseado em suas características, traduzidas por seus descritores. Enquanto a parte de interpretação necessita de um contexto para realizar as validações.

Por exemplo, a placa veicular MERCOSUL é dada por uma sequência de 3 letras, 1 número, 1 letra e 2 números. Caso os dados recebidos não respeitem

essa sequência, é viável afirmar que os dados não fazem sentido e podem ser descartados para futuros processamentos.

2.3 Optical Character Recognition (OCR)

O Optical Character Recognition ou popularmente chamado de OCR, já existe há diversos anos, possuindo data de registros e pesquisas em 1950, quando David Shepard e Louis Tordella pesquisavam uma forma de automatizar dados da Agência de Segurança das Forças Armadas (Wanzeller, 2018).

Basicamente pode se dizer que OCR é uma tecnologia que possibilita a retirada de textos de imagens.

Atualmente existem diversas bibliotecas de programação que usufruem desta tecnologia para decodificar imagens em caracteres, como Tesseract, OpenCV, além de outras.

2.4 YOLO

A detecção de objetos é uma aplicação que está ficando cada vez mais popular nos dias atuais e existem diversos sistema que fazem isso para o usuário de forma automática e simples.

O sistema YOLO cujo nome´e a sigla em inglês de You Only Look Once (Você Só Olha Uma Vez, em tradução direta) é capaz de fazer detecção de objetos 1000x mais rápido que modelos mais antigos de R-CNN (ALMEIDA, 2019).

Seu nome é dado pelo motivo de que, modelos mais antigos o algoritmo analisa uma mesma imagem diversas vezes no intuito de reconhecer objetos e sinalizar sua existência. Já o YOLO, verifica a imagem uma única vez, e utiliza uma inteligência probabilística para detectar regiões que têm grandes probabilidades de ter um objeto conhecido, e partir dessas regiões faz uma análise mais aprofundada.

Além de esconder toda lógica complexa de reconhecimento de objetos e transformar em algo simples, YOLO fornece modelos treinados prontos, que aceleram muito o processo, pois não é necessário ficar coletando um conjunto de imagens para ensinar o modelo a reconhecer um objeto comum.

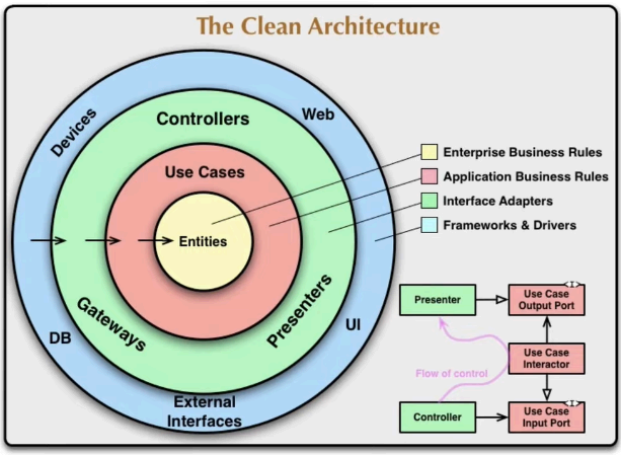
2.5 Arquitetura de Código

No intuito de estabelecer um código confiável, entendível e principalmente escalonável, é importante adotar uma ou mais de uma arquitetura de códigos. Abaixo temos a arquitetura mais popular entre a comunidade de desenvolvedores.

2.5.1 Clean Architecture

Uma das arquiteturas mais populares entre os desenvolvedores é a Clean Architecture, criada por Robert C. Martin e promovida em seu livro “Clean Architecture: A Craftsman 's Guide to Software Structure”. Ela tenta fornecer metodologias a serem utilizadas no intuito de facilitar o desenvolvimento, manutenção e diminuir as dependências de código, o que implica em um código menos complexo, demonstrado na Figura 2.

Figura 2 - Arquitetura limpa



Fonte: Martin (2017).

Resumidamente, podemos dizer que a camada externa em azul, é onde ficam conectadas todas as aplicações externas ao *back end*, como por exemplo uma interface WEB. Na camada mais ao centro, em verde, temos a camada que se comunica com o meio exterior, é nela que ficam presentes os *endpoints* e toda a lógica de receber uma requisição, processá-la e gerar uma resposta. Na camada vermelha, temos toda a regra de negócios da aplicação e todo processamento dos dados. E por último, na camada central, ficam mapeadas as

entidades que representam as tabelas do banco de dados, interfaces, enumerators, etc..

Importante ressaltar que a camada mais externa pode depender das camadas internas, porém o inverso não se aplica.

3 METODOLOGIA

A metodologia adotada neste trabalho segue uma abordagem sistemática e organizada para o desenvolvimento do sistema de controle de acesso veicular. A seguir, são descritos os principais passos e procedimentos seguidos durante a realização do projeto:

3.1 Definição dos requisitos

Inicialmente, foram identificados e detalhados os requisitos necessários para ser um software viável de comercialização e integração com demais sistemas, considerando as necessidades dos usuários e as especificidades do contexto de aplicação. São eles:

1. Possibilidade de executar em um computador no condomínio: Dado que a grande maioria dos condomínios possuem ao menos 1 computador, o software desenvolvido garante a possibilidade de rodar localmente. Possibilitar o cadastro e a execução de ações nos portões diretamente da guarita, sem a necessidade de empresas terceirizadas realizarem integração. Além deste ponto, o fato de conseguir rodar localmente e a possibilidade de ações diretas no software, facilitam a instalação e averiguação do funcionamento do sistema como um todo.

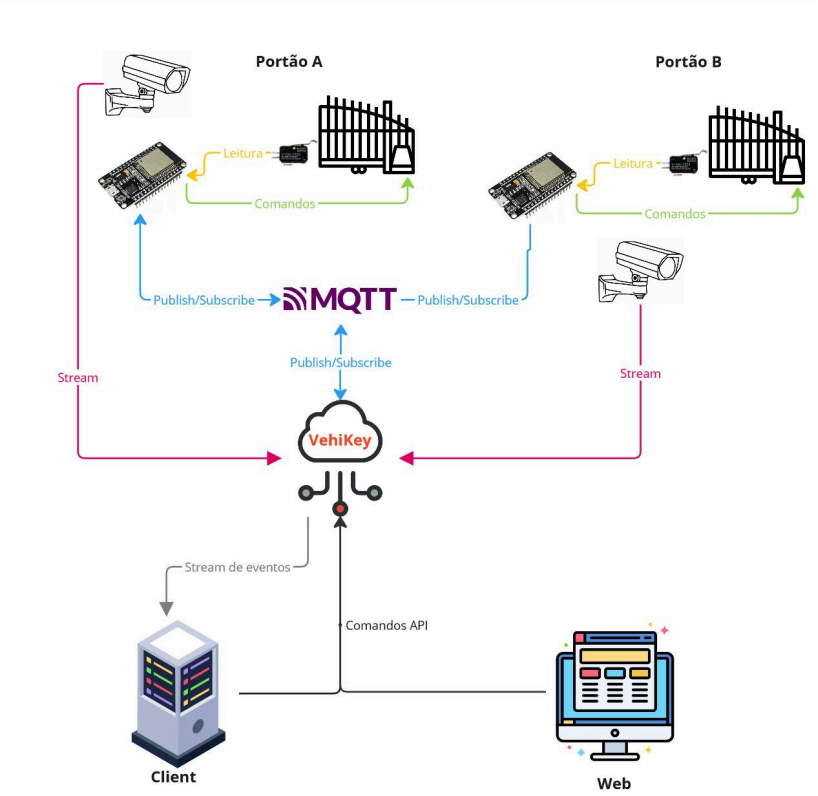
2. Detecção de placa veicular com uso de câmeras comuns: Utilização de câmeras IP com protocolo RTSP para capturas de imagem para posteriormente realizar o processamento e identificar os caracteres da placa.

3. Integração com softwares de terceiros: Para ser um produto integrável, é importante que possua uma API robusta, permitindo operações CRUD (Create, Update, Read e Delete) sobre diversas funcionalidades do software.

4. Geração de eventos: A geração de eventos em tempo real, como a indicação de um veículo com acesso permitido acessando o condomínio, status de porta e além de outros, é de extrema importância para que os serviços conectados na aplicação, saibam o que está acontecendo e tomem as devidas ações.

Para atingir os requisitos supracitados, foi elaborado um diagrama (Figura 3) mostrando uma visão macro de como as demais partes do sistemas irão realizar a comunicação entre si e como estarão dispostas no cenário. Seu funcionamento em mais detalhes serão explicados ao decorrer deste capítulo.

Figura 3 - Visão geral do funcionamento do sistema



Fonte: Elaboração própria (2024).

3.2 Back end

A robustez, eficiência e segurança do *back end* são vitais para a integridade de qualquer software. Um *back end* bem projetado pode escalar com o crescimento do número de equipamentos e com o aumento do volume de dados, sem comprometer o desempenho ou a disponibilidade do sistema.

Em suma, o *back end* é o motor da aplicação, funciona de forma ininterrupta, tem a responsabilidade de executar as regras de negócios de extrema relevância, processar requisições de endpoints, e no caso do presente trabalho, realizar a identificação de placas veiculares para abrir ou não o portão. Embora invisível aos olhos do usuário final, o mesmo tem interação constante com o *back end* de maneira indireta, contudo ele é fundamental para garantir que a aplicação no geral funcione da maneira mais eficaz possível, segura e escalável.

3.2.1 Fluxograma

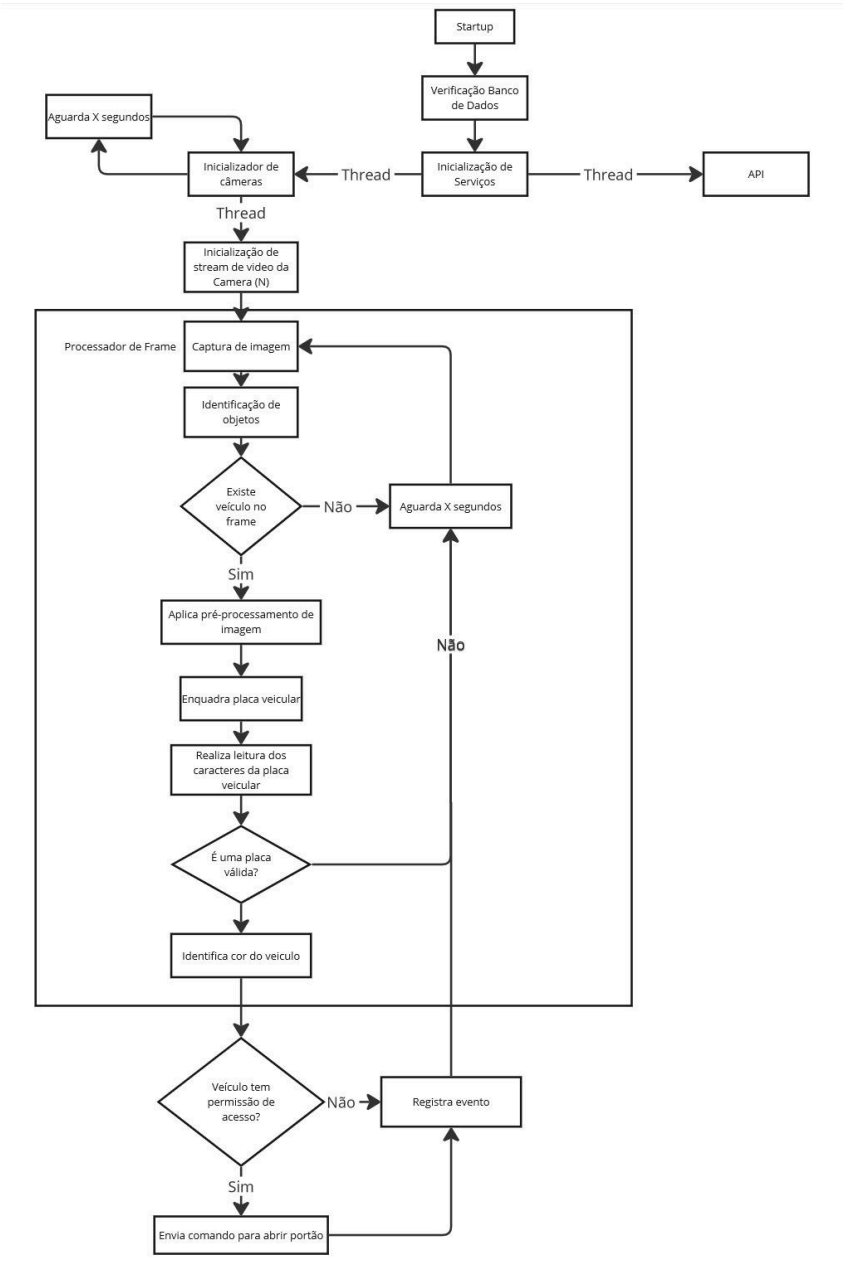
Para fornecer uma visão clara e estruturada do fluxo de operação do sistema de controle de acesso veicular, foi elaborado um fluxograma, mostrado na Figura 4. Este fluxograma apresenta cada etapa do processo a partir de uma visão macro do back end.

De primeiro momento ocorre uma checagem da existências de todas as tabelas e colunas no banco de dados, para que, caso haja necessidade, se faça a inserção dos itens faltantes.

Logo após, a inicialização em threads (processos executados de forma independente) separados de alguns serviços principais, como a API e o serviço responsável por manejar as conexões das câmeras.

No processador de frames temos a lógica mais relevante do projeto, onde é realizado as devidas ações para reconhecimento dos caracteres e cor do veículo, para então, acionar ou não a abertura do portão e registrar, em forma de evento, o acesso realizado.

Figura 4 - Fluxograma do *back end*



Fonte: Elaboração própria (2024).

3.2.2 Arquitetura

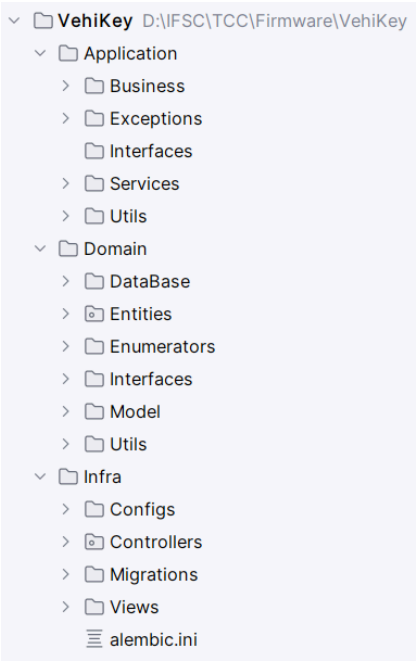
A arquitetura escolhida para desenvolver o back end foi a Clean Architecture, elaborada por Robert C. Martin.

Sua escolha foi dada pela as vantagens de ser testável, independente de interface gráfica (UI), independente de banco de dados ou de qualquer agente

externo. Além de que, é uma das arquiteturas mais populares atualmente, sendo de fácil compreensão para a maioria dos desenvolvedores *back-ends*.

Apesar de a *Clean Architecture* não se resumir apenas à organização de pastas, ela desempenha um papel crucial na estruturação e compreensão do projeto como um todo. Enquanto a *Clean Architecture* abrange princípios de design mais profundos, como a separação de preocupações e a definição clara de fronteiras entre os componentes do sistema, a organização das pastas reflete e facilita a implementação desses princípios, a qual podemos ver na Figura 5, a estruturação dos arquivos do projeto VehiKey.

Figura 5 - Estruturação de Arquivos



Fonte: Elaboração própria (2024).

Na camada Domain, conforme preconizado pela Clean Architecture, foram preservadas as interfaces para os *Casos de Uso* e *Repositórios*, bem como as entidades (que representam as tabelas do banco de dados), modelos (semelhantes às entidades) e enumeradores.

No módulo de Application, foram incorporadas regras de negócio para consultas no banco de dados, exceções personalizadas, e os serviços responsáveis por executar funções específicas, tais como o processamento de eventos e vídeos.

Enquanto na camada de infraestrutura, chamada de “Infra” no projeto, foram criados arquivos de configurações externas, views, migrations e principalmente os controllers, onde são registrados os endpoints.

3.2.3 Banco de Dados

Para a gestão dos dados do software de controle de acesso veicular, foi escolhido o MySQL como sistema de gerenciamento de banco de dados.

Para a criação das tabelas no banco de dados, utilizou-se o método de *migrations*, implementado através da combinação dos frameworks Alembic e SQLAlchemy. Essas ferramentas convertem os modelos de objetos definidos no código em comandos SQL executáveis. Cada nova *migration* gerada é registrada em um histórico controlado pelos frameworks, permitindo monitorar as modificações estruturais no banco de dados. Com isso, é possível gerenciar de forma simples, fácil e automática, alterações como adições, atualizações ou remoções de informações na base de dados, simplificando significativamente o processo de manutenção.

Na Figura 6, são apresentados os ORMs (*Object-Relational Mapping)*, os quais permitem configurar como cada coluna deve ser criada com base nos objetos e suas propriedades.

Figura 6 - Modelos

```python
class Camera(Base):
    __tablename__ = 'camera'
    id = Column( *args: Integer, primary_key=True)
    created_at = Column( *args: DateTime(timezone=True), default=datetime.now)
    ip = Column( *args: String(20), nullable=False)
    port = Column( *args: Integer, nullable=False)
    password = Column( *args: String(10), nullable=False)
    user = Column( *args: String(10), nullable=False)
    name = Column( *args: String(30), nullable=False)
    manufacturer = Column( *args: String(20), nullable=False)
    valid_time = Column( *args: String(50), nullable=True)
    controller = Column( *args: String(50), nullable=True)
```

```python
class Event(Base):
    __tablename__ = 'event'
    id = Column( *args: Integer, primary_key=True)
    created_at = Column( *args: DateTime(timezone=True), default=datetime.now)
    message = Column( *args: String(600), nullable=True)
    sent_to_server = Column( *args: Boolean, default=False, nullable=False)
    vehicle_id = Column( *args: Integer, ForeignKey('vehicle.id'))
    camera_id = Column( *args: Integer, ForeignKey('camera.id'))
class Vehicle(Base):
    __tablename__ = 'vehicle'
    id = Column( *args: Integer, primary_key=True)
    created_at = Column( *args: DateTime(timezone=True), default=datetime.now)
    color = Column( *args: String(10), nullable=True)
    plate = Column( *args: String(10), nullable=False)
    valid_time = Column( *args: String(50), nullable=True)
    access_time = Column( *args: String(500), nullable=True)
    number_access = Column( *args: Integer, nullable=True)
```

Fonte: Elaboração própria (2024).

3.2.4 Controle de Acesso

Para realizar o controle de acesso efetivo, inicialmente é necessário garantir que o banco de dados esteja devidamente populado, o que pode ser feito através da API de integração ou diretamente pela interface web do sistema.

Uma vez que os dados estão cadastrados, procede-se à conexão com a câmera IP utilizando o protocolo RTSP. Para que essa conexão seja bem-sucedida, é essencial que a câmera esteja acessível pelo software, o que pode ser conseguido se ambas estiverem na mesma rede ou através de métodos como VPN, DDNS, entre outros.

Após estabelecer a conexão com a câmera, o sistema realiza capturas de imagem em intervalos definidos, para as configurações da máquina. A cada intervalo, uma nova imagem é capturada para análise.

O primeiro passo na análise da imagem é verificar a presença de um veículo, utilizando um modelo de visão computacional pré-treinado base fornecido pela *Ultralyrics* em seu *GitHub*, com uso da tecnologia fornecida pelo YOLO V8. Identificada a presença do veículo, é recortada a área da imagem que corresponde ao veículo para uma análise mais detalhada.

Na etapa seguinte, esta sub figura passa por um processo de tratamento de imagem com auxílio da biblioteca OpenCV, adicionam-se filtros específicos para melhorar a clareza, mostrado na Figura 7. Otimizando a detecção dos caracteres da placa pelo modelo pré-treinado com base em mais de 20 mil imagens de placas veiculares, modelo fornecido por Muhammad Zeerak Khan em seu *GitHub*, para reconhecimento de placas, também baseado utilizando a biblioteca YOLO V8.

Figura 7 - Tratamento de imagem para reconhecimento de placa

Fonte: Elaboração própria (2024).

Após a detecção da placa na imagem com auxílio novamente do YOLO V8 pré treinado com imagens de placas veiculares brasileiras, procede-se ao recorte da área correspondente à placa em forma de retângulo. Em seguida, a imagem recortada é submetida a um tratamento específico: as cores são convertidas para a escala de cinza. Posteriormente, aplica-se uma técnica de limiarização (thresholding), na qual pixels que apresentam tonalidades abaixo de um certo valor são transformados em branco, enquanto os demais são convertidos em preto. Esse processamento resulta em uma imagem binária, facilitando a identificação dos caracteres, como pode ser observado na Figura 8.

Figura 8 - Tratamento de imagem para reconhecimento de caracteres

Fonte: Elaboração própria (2024).

Utilizando a tecnologia de OCR, neste caso, com auxílio da biblioteca easyOCR, escolhida dada sua facilidade de uso e com modelos pré-treinados, os caracteres da placa são lidos e analisados. Caracteres que podem ser facilmente confundidos, como o 'i' e o '1', ou o '0' e a letra 'O', são cuidadosamente formatados para evitar erros, baseando-se na posição típica dos caracteres em placas veiculares, sejam elas no padrão Mercosul ou nas tradicionais placas cinzas.

Finalmente, com a placa identificada, o sistema acessa o banco de dados em busca de informações da placa lida e executa as regras de negócio estabelecidas para determinar se o veículo tem permissão para entrar no condomínio. Esta decisão é baseada em critérios previamente configurados, como horários de acesso permitidos, e listas de autorizações, garantindo assim a segurança e a gestão eficiente do acesso veicular ao condomínio.

Ao realizar um acesso permitido, é publicado via MQTT uma mensagem formatada em JSON, contendo valores previamente definidos para os dispositivos que estão subscritos neste tópico, realizem a devida ação de abertura do portão.

3.2.5 Detecção de cor

Após isolar o veículo, recortando-o especificamente da imagem original, é realizada a computação da média das cores presentes na imagem utilizando bibliotecas externas, resultando na média de cores em formato RGB.

Em seguida, um script é utilizado para mapear aproximadamente 100 variações de cores no formato RGB. O script realiza cálculos matemáticos para determinar qual dessas variações se aproxima mais da média calculada anteriormente.

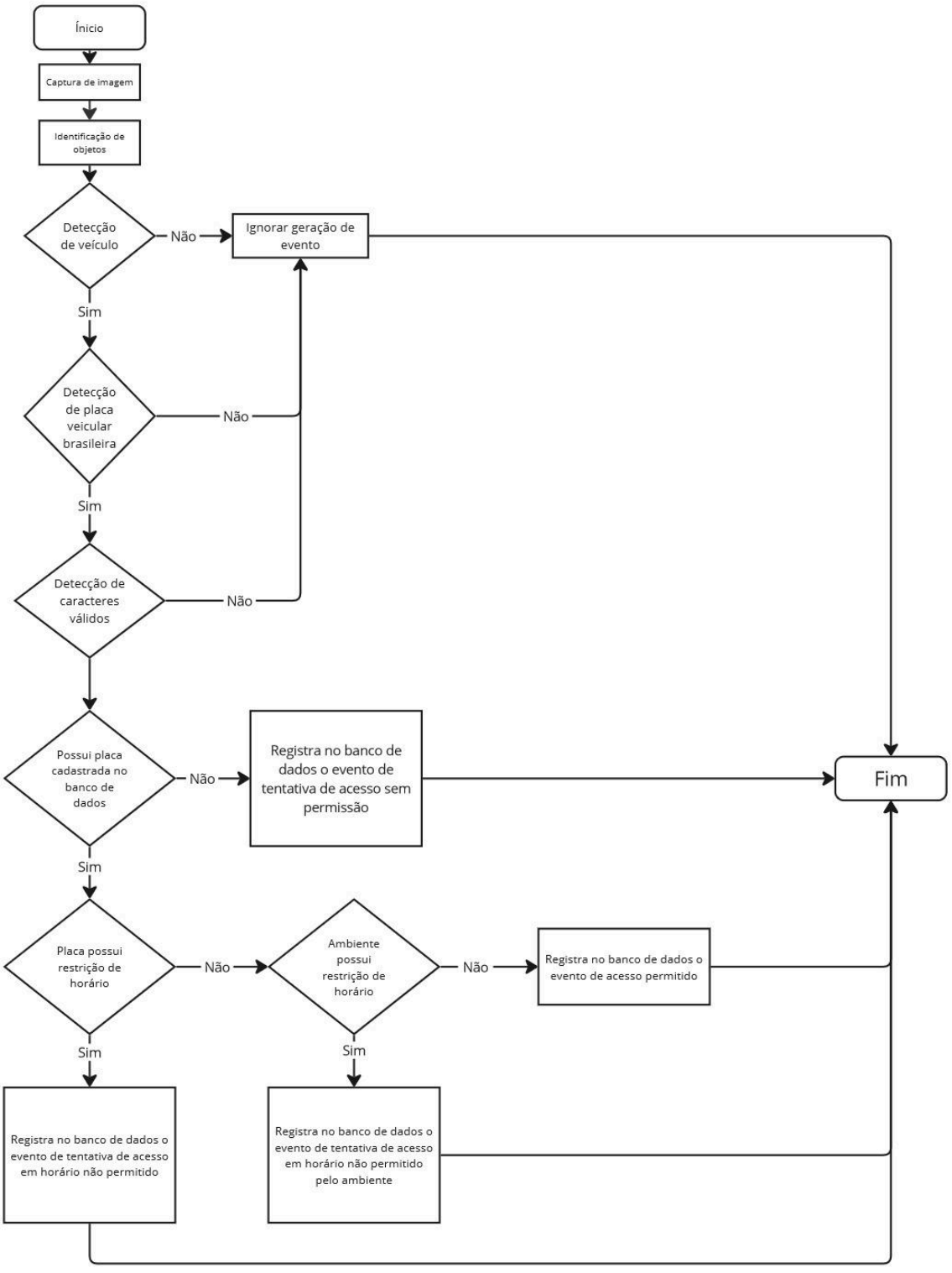
Após identificar a cor mais próxima, é verificado a qual cor primária essa variação pertence, permitindo assim a identificação precisa da cor predominante do veículo. Essa etapa é fundamental para complementar a identificação do veículo, garantindo maior exatidão e eficiência no processo de controle de acesso.

3.2.6 Eventos

Os eventos desempenham um papel crucial no sistema de controle de acesso veicular, pois fornece notificações e registros em tempo real sobre as atividades que estão ocorrendo no condomínio.

Os eventos relacionados aos veículos ocorrem no momento em que é detectado a presença de um veículo e a identificação de todos os caracteres da placa veicular. Caso os caracteres sejam identificados corretamente, é gerado um evento condizente ao do acesso, caso ele tenha permissão para entrar, será gerado um evento indicando o acesso com sucesso, caso não tenha permissão, seja pelo fato de não estar autorizado, ou estar tentando acessar em um horário não permitido previamente, será gerado um evento de acesso negado, com a devida identificação do motivo de acesso negado, conforme mostrado na Figura 9.

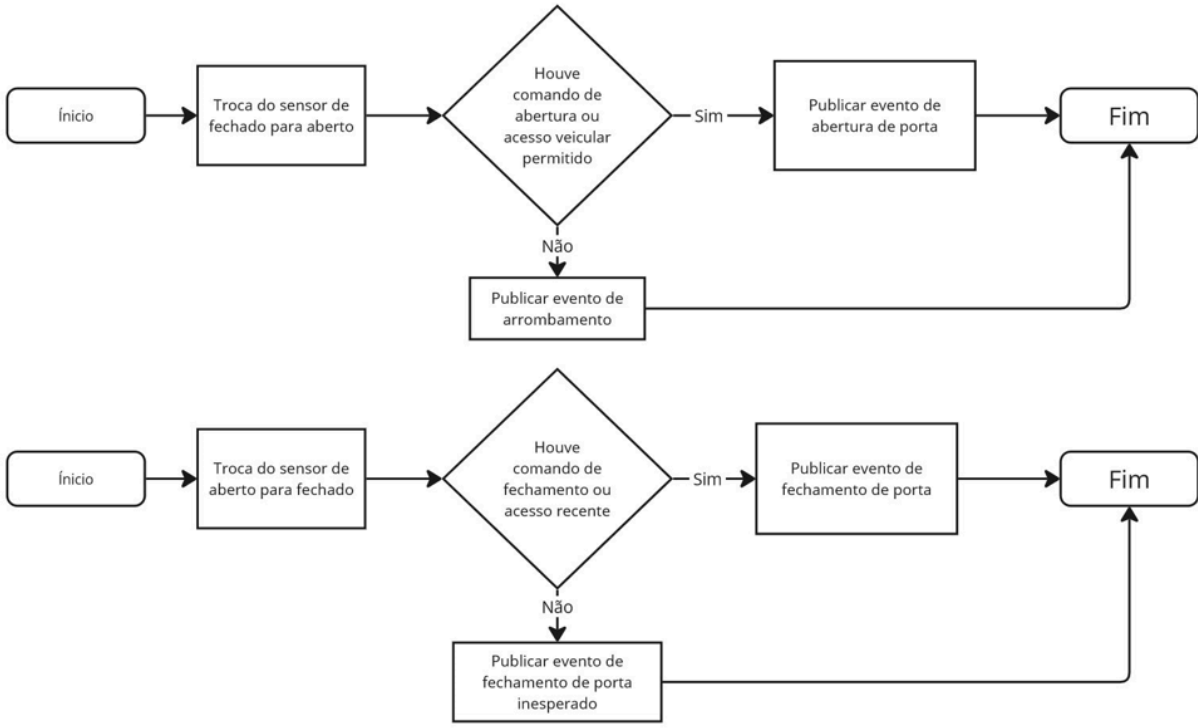
Figura 9 - Fluxograma geração de eventos de acesso



Fonte: Elaboração própria (2024).

Os eventos relacionados ao estado das portas, são gerados pelos hardwares vinculados ao portão (detalhado melhor no tópico seguinte), eles são os responsáveis por sinalizar quando houver uma troca de sinal do sensor do portão, indicando uma abertura ou fechamento do mesmo, mais detalhes do fluxo podem ser vistos na Figura 10.

Figura 10 - Fluxograma geração de eventos de status de porta



Fonte: Elaboração própria (2024).

Após os eventos serem devidamente registrados no banco de dados, caso não haja nenhum cliente conectado à aplicação, esses eventos são armazenados com uma flag indicando que não foram transmitidos aos ouvintes do MQTT (*subscribers*). Quando um cliente se conecta, todos os eventos não enviados até então são entregues de forma ordenada ao serviço recém-conectado. Isso garante que nenhuma informação importante seja perdida e que todos os eventos sejam processados e notificados assim que possível.

Em casos de conexão com um cliente, é enviada uma mensagem base a cada 30 segundos, conhecida como *keep-alive*. Essa mensagem indica que o

serviço está operante, permitindo ao cliente identificar rapidamente qualquer queda de conexão e assegurar que está efetivamente conectado ao servidor.

3.3 Hardware

Neste tópico, abordaremos a eletrônica do sistema de controle de acesso veicular, detalhando os componentes de hardware utilizados e suas respectivas funções.

Na Figura 3, temos uma visão do funcionamento do sistema de forma geral, onde a estrutura permite adicionar novos equipamentos de forma quase que plug-in-play, desde que previamente cadastrados no banco de dados.

Temos como foco principal e centralizado o software VehiKey, o responsável por realizar a interação entre todos os componentes.

A componente “câmera”, se comunica com o software por meio do uso do protocolo RTSP, passando a ele imagens em tempo real do que está acontecendo, que por sua vez, VehiKey usufrui para realizar o devido processamento já citado anteriormente.

Enquanto o microcontrolador ESP 32 tem interação direta com o portão (sensores e relés), sendo responsável por enviar os eventos gerados pelo o sensor do portão e atender as requisições de abertura ou fechamento do portão feitas pelo software, por meio da comunicação MQTT.

Os eventos recebidos da etapa anterior, são repassados para os clientes que estão conectados em VehiKey por meio de um *Long Polling HTTP*, mantendo-os informados em tempo real do que está ocorrendo no condomínio. Além de receber comandos via API da interface WEB ou demais softwares integrados ao sistema.

3.3.1 Microcontrolador ESP-32

Para a automação do portão, será utilizado um microcontrolador com acesso à Wi-Fi, o qual assumirá a responsabilidade de acionar um relé conectado ao motor do portão, para realizar as aberturas e fechamentos de forma automatizada. Além disso, irá realizar o monitoramento dos sinais dos sensores do portão,

indicando se ele está aberto ou fechado. A comunicação entre o software VehiKey e o hardware “Slave” será realizada através do protocolo MQTT, garantindo uma troca de informações eficiente e em tempo real.

O envio de mensagens no sentido software para hardware se encontra no tópico VehiKey/Subscribe/{id_referencia}, enquanto as mensagens no sentido hardware para software se encontram no tópico VehiKey/Publish/{id_referencia}. Onde o id_referencia é um valor único por microcontrolador, e registrado no banco de dados na tabela relacionada a câmeras, sendo possível ser definido através da API ou interface WEB, definindo o campo “serial_control”.

3.3.2 Câmeras

As câmeras IP serão utilizadas exclusivamente para capturar a stream de vídeo, que será analisada pelo VehiKey para a identificação das placas veiculares. A comunicação entre as câmeras e o software ocorre via protocolo RTSP, permitindo a transmissão contínua de vídeo para o processamento e reconhecimento das placas.

Devem ser instaladas de forma que facilite o reconhecimento dos caracteres da placa veicular e o ambiente possua iluminação adequada.

3.3.3 Comunicação entre Hardware e Software

A comunicação entre o hardware e o software VehiKey é fundamental para o funcionamento eficiente do sistema de controle de acesso veicular. Esta comunicação é estabelecida através do protocolo MQTT, que permite a troca de mensagens entre os microcontroladores e o software, para recebimentos e envios de comandos e eventos.

E também através do protocolo RTSP, para a stream de vídeo com a câmera.

Por sua vez, a Interface Web se comunica com o *back end* através da API desenvolvida, respeitando os verbos HTTPs.

Desta forma, o software VehiKey será projetado para suprir um condomínio sem se preocupar com a quantidade de portões que ele possui. Onde a

única limitação se dá na máquina hospedada, para realizar o processamento de imagem.

3.4 Interface Web

A interface web é uma alternativa para mexer no sistema de forma mais intuitiva, sem a necessidade de um software terceiro consumindo a API. Ela permite realizar as funções básicas de administração do acesso veicular, enquanto as funcionalidades mais complexas como restrição de horário, estão disponíveis somente via API.

As principais funcionalidades implementadas na interface web incluem:

3.4.1 Administração de Câmeras Via Web

Na página referente a câmera, ilustrado na Figura 11, podemos realizar a inserção de múltiplas câmeras ao mesmo tempo, para agilizar o processo. Na primeira coluna existe a opção de atribuir um nome à câmera, para fins de identificação. Logo depois, é atribuído um IP, porta, usuário e senha, informações cruciais para a aplicação conseguir se conectar a stream de vídeo fornecida pela câmera. E na última coluna, temos o “Serial Control”, com base nesse valor inserido, temos a relação entre a câmera e o microcontrolador.

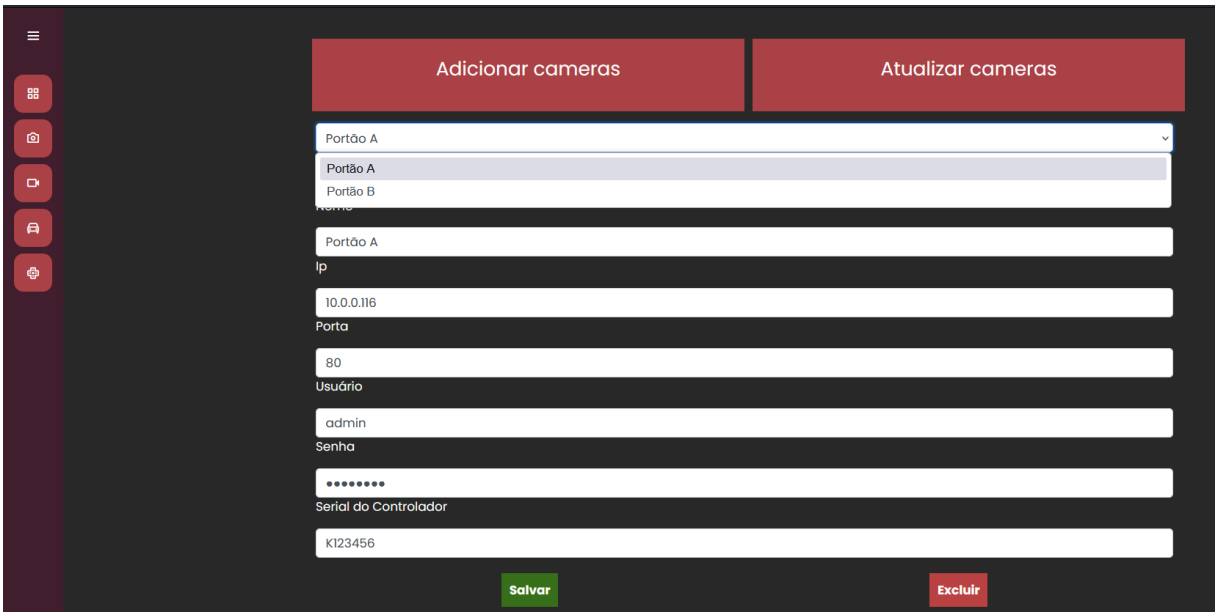
Como mencionado no tópico anterior, cada microcontrolador subscreve e publica num tópico único, cada tópico possui como sufixo o “Serial Control”, onde este, deve ser registrado corretamente nessa página.

Figura 11 - Página de adição de câmeras

Fonte: Elaboração própria (2024).

Na mesma página, porém na segunda aba “Atualizar câmeras”, temos a interface mostrada na Figura 12. Nessa aba, é possível editar as câmeras já cadastradas no sistema e também realizar a operação de exclusão.

Figura 12 - Página de edição de câmeras



Fonte: Elaboração própria (2024).

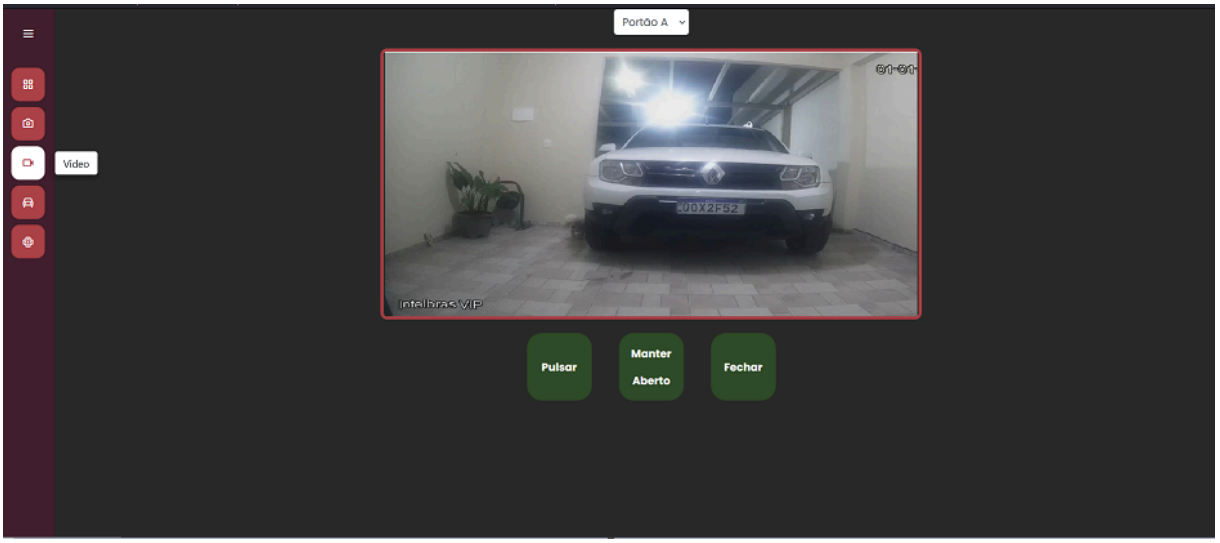
Cada operação realizada nesta página, chama um determinado endpoint fornecido pela a API do *back end*, seguindo os padrões dos verbos HTTPs (GET, POST, PUT, DELETE), o qual realiza as interações necessárias com o banco de dados e retorna o resultado, atualizando a página web com as informações corretas.

3.4.2 Visualização de vídeo

Na página seguinte (Figura 13), temos a possibilidade de ver em tempo real o vídeo da câmera escolhida. Ao selecionar a câmera, é estabelecido uma stream de conexão com o *back end* através da API, onde os dados transmitidos pelo *back end* são as imagens coletadas através do protocolo RTSP com a câmera.

Além da visualização, é possível enviar comandos, como abrir o portão, mantê-lo aberto ou realizar o fechamento.

Figura 13 - Página de vídeo

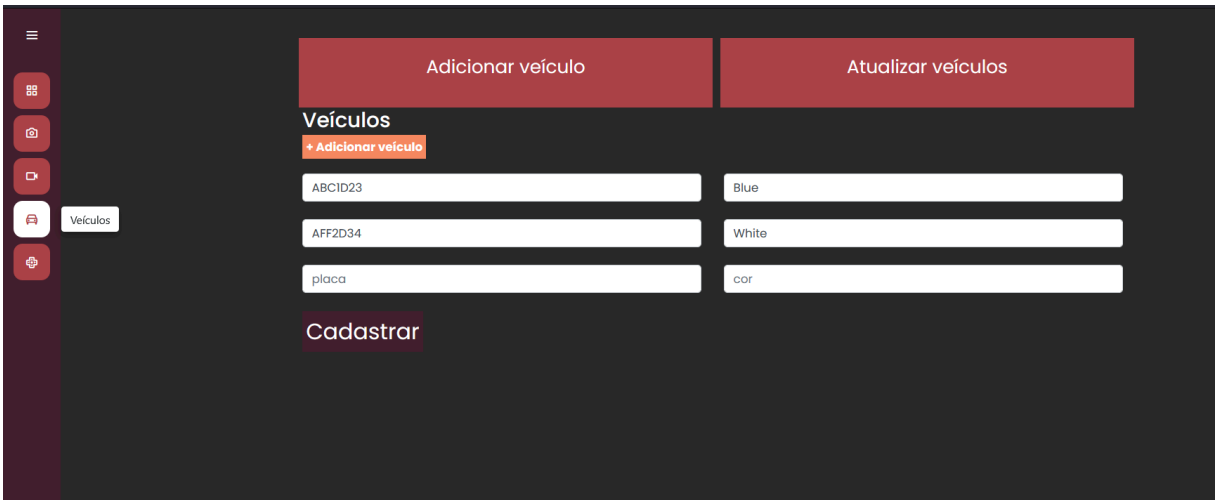


Fonte: Elaboração própria (2024).

3.4.3 Administração de Veiculos Via Web

De modo similar ao tópico 3.4.1, temos a possibilidade de adição de múltiplos veículos ao mesmo tempo, com basicamente as informações de placa e cor, representados na Figura 14.

Figura 14 - Página de adição de veículos



Fonte: Elaboração própria (2024).

Na aba ao lado, representada pela Figura 15, é possível realizar a edição de um veículo selecionado ou a sua exclusão.

De modo análogo ao tópico supracitado, as alterações realizadas são feitas através de endpoints previamente mapeados em ambos serviços.

Figura 15 - Página de edição de veículos

Fonte: Elaboração própria (2024).

Com todas as operações fornecidas via Web, é possível realizar a implementação completa em um condomínio, testes e até mantê-lo operacional somente através da interface.

4 APRESENTAÇÃO DOS RESULTADOS

Nesta seção, serão apresentados os resultados obtidos a partir do desenvolvimento e testes do sistema de controle de acesso veicular automatizado. A análise dos resultados visa verificar a eficácia do sistema e avaliar o desempenho das funcionalidades implementadas e sua viabilidade comercial.

Primeiramente, serão expostos os resultados referentes ao reconhecimento de veículos e placas veiculares, detalhando a exatidão e a taxa de sucesso das detecções realizadas pelo software. Em seguida, será abordada a geração de eventos enviados para um cliente.

A câmera utilizada nos testes a seguir foi o modelo VIP 1020 B G2 da Intelbras, com qualidade de vídeo de 720p, sendo obtido o vídeo através do protocolo RTSP.

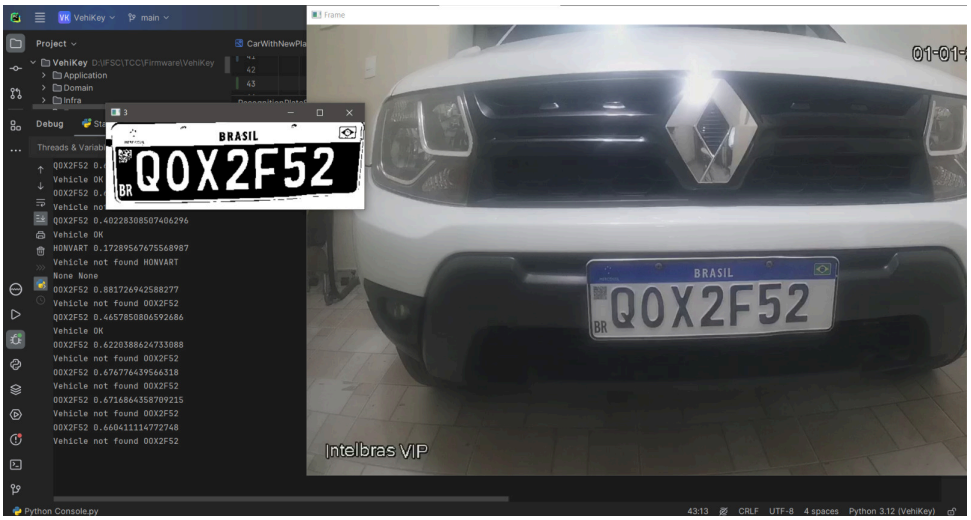
Como primeiro teste, foi posicionado o carro a aproximadamente 1 metro de distância da câmera que se encontra a 1 metro de altura em relação ao solo e perpendicular à dianteira do veículo, como mostrado na Figura 16.

Figura 16 - Configuração do ambiente do teste de 1 metro

Fonte: Elaboração própria (2024).

A captura e o processamento final da placa diante destas condições, pode ser observado na Figura 17.

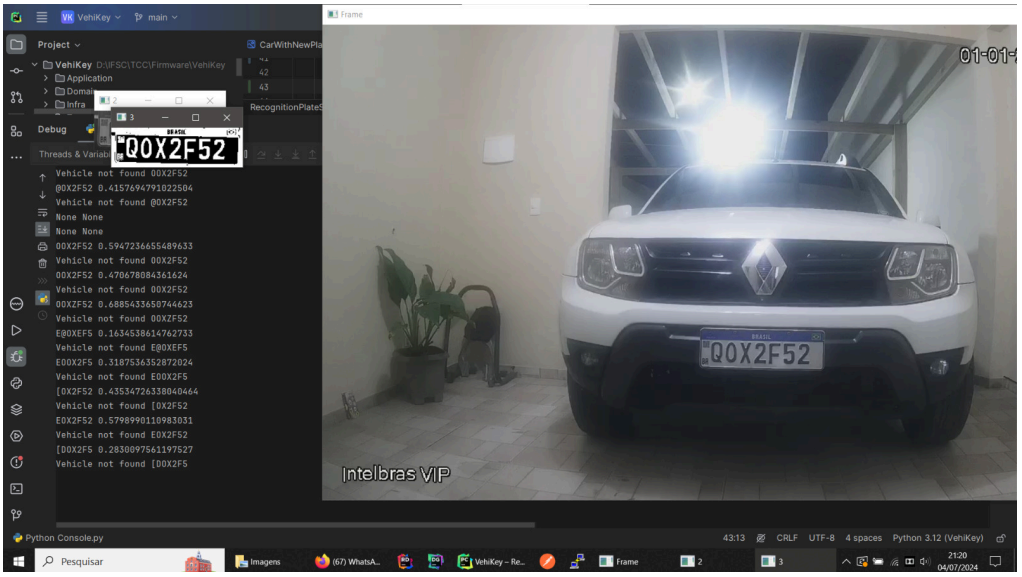
Figura 17 - Captura do teste de 1 metro



Fonte: Elaboração própria (2024).

No segundo teste, foi feito com uma distância de aproximadamente 2 metros, mantendo a mesma altura da câmera e ângulo em relação a dianteira do veículo, como pode ser observado na Figura 18.

Figura 18 - Configuração do ambiente do segundo teste de 2 metros



Fonte: Elaboração própria (2024).

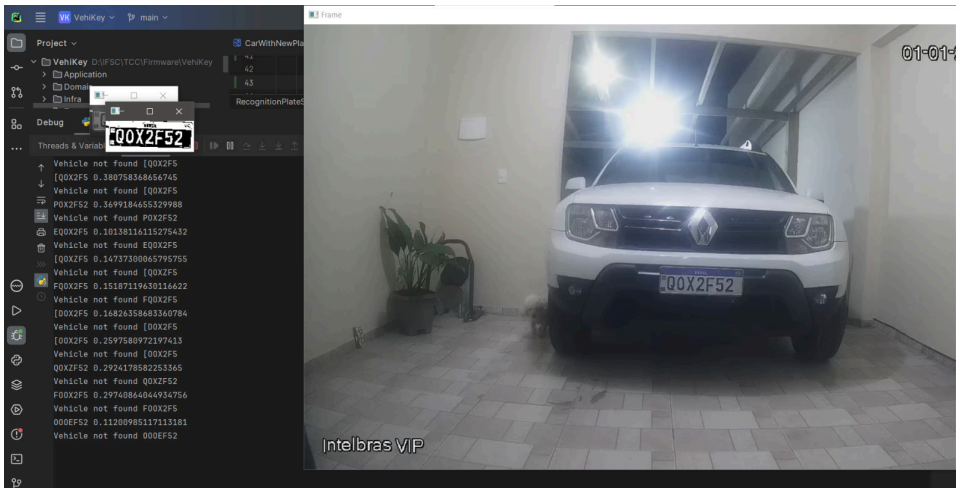
Em um dos testes realizados a uma distância de 2 metros, foi movido a câmera lateralmente, para que realizasse a leitura da placa com um determinado ângulo, como pode ser visto na Figura 19.

Figura 19 - Configuração do ambiente do segundo teste de 2 metros

Fonte: Elaboração própria (2024).

O último teste real, foi feito com uma distância de 3 metros, registrado na Figura 20.

Figura 20 - Configuração do ambiente do segundo teste de 3 metros



Fonte: Elaboração própria (2024).

Demais testes foram feitos utilizando imagens coletadas de veículos na internet, como o da Figura 21, com a representação da leitura final da placa veicular.

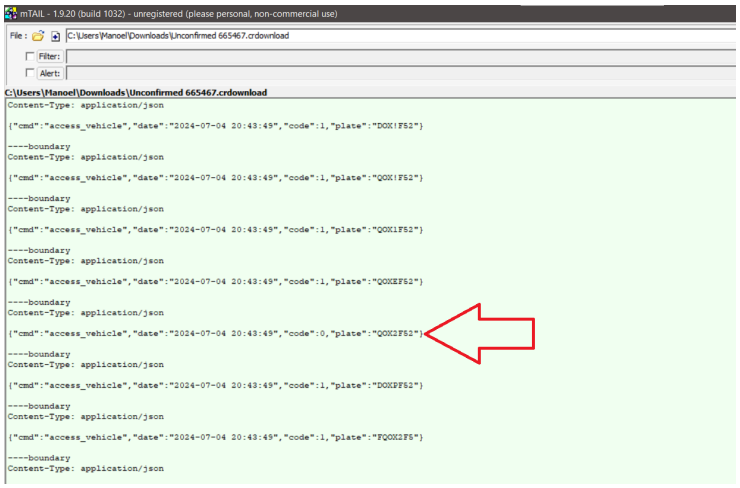
Figura 21 - Imagem de veículo

Fonte: Elaboração própria (2024).

Na figura 22, é possível ver os eventos serem enviados do servidor para o cliente. Para realizar a simulação da conexão com o servidor por meio de stream, foi acessado o endpoint dos eventos através do navegador Google Chrome e realizado a leitura do arquivo em download através do software mtail.

Destacado na imagem abaixo, podemos ver um acesso com sucesso, indicado pelo “code” igual a 0, enquanto os demais são de acessos não permitido.

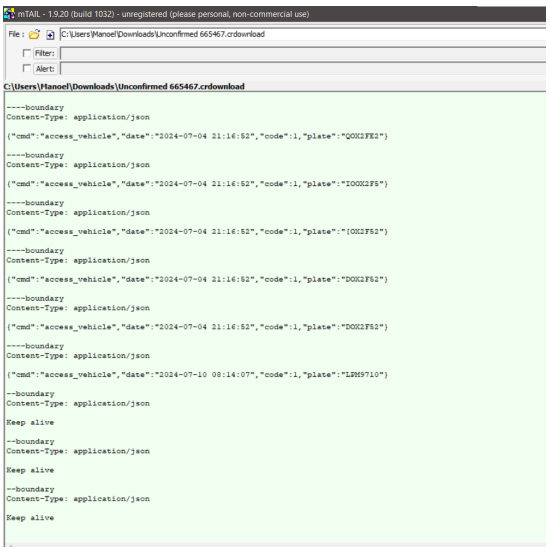
Figura 22 - Eventos de acesso



Fonte: Elaboração própria (2024).

É possível visualizar os eventos de *keep-alive* sendo enviados a cada 30 segundos na Figura 23, indicando que o servidor permanece conectado à aplicação.

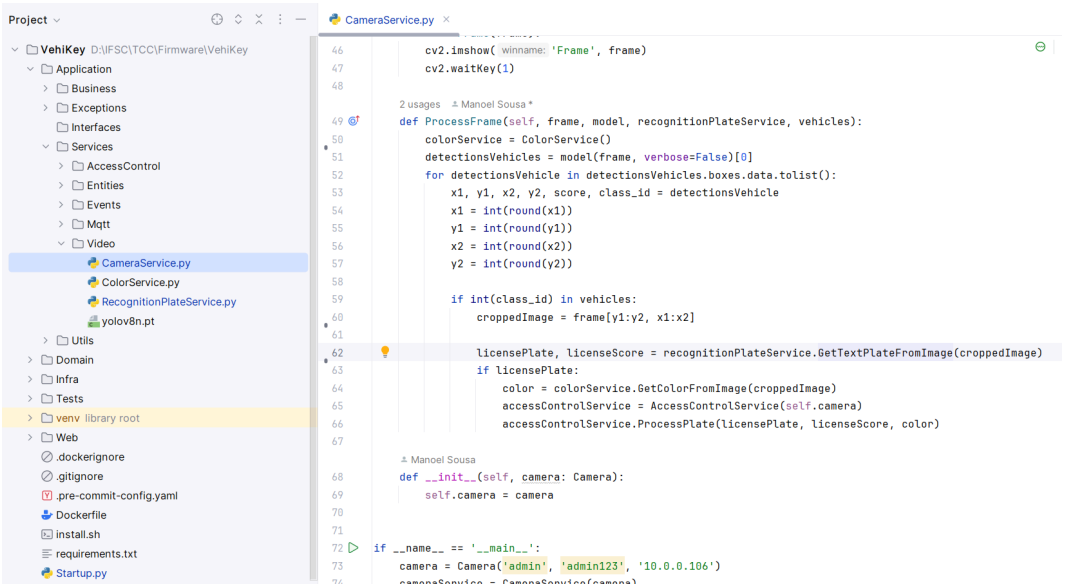
Figura 23 - Eventos de *keep-alive*



Fonte: Elaboração própria (2024).

O código fonte responsável pelo processamento dos frames a partir da stream de vídeo, pode ser observado logo a seguir, na Figura 24. Nele é feito a detecção de todos os veículos presentes na imagem, para posteriormente, recortar a região de interesse da imagem principal e realizar o processamento dos caracteres da placa e cor do veículo.

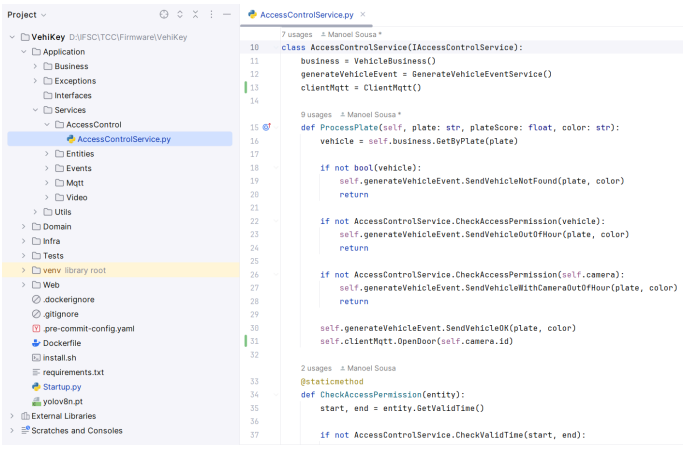
Figura 24 - Processador de frames



Fonte: Elaboração própria (2024).

Na Figura abaixo, podemos ver a lógica do serviço responsável por fazer a verificação se um determinado veículo possui acesso, além de gravar o evento no banco de dados e realizar a abertura de porta, caso necessário.

Figura 25 - Serviço de validação de acesso

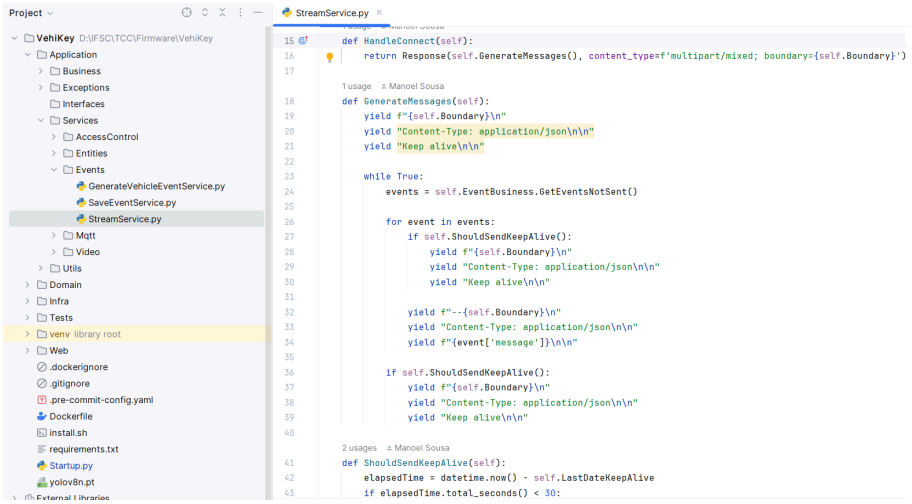


Fonte: Elaboração própria (2024).

Na seguinte imagem, consta a lógica principal do envio efetivo dos eventos para um cliente. Nesta lógica, após estabelecido um *long pooling http* com o

cliente, é solicitado os eventos que não foram enviados até o momento para o serviço responsável por fazer consultas no banco de dados, e realizado o envio dos mesmos. Em casos onde o tempo ultrapasse os 30s, é enviado um keep-alive.

Figura 26 - Publicador de eventos



Fonte: Elaboração própria (2024).

4.1 Análise e discussão dos resultados

Com base nos dados previamente mostrados no tópico anterior, abordaremos com mais profundidade e um olhar analítico sobre os resultados obtidos.

Na Tabela 2, temos os resultados dos testes práticos mostrados anteriormente a partir da leitura de uma placa veicular “QOX-2F52” seguindo os padrões da MercoSul.

Observado que, leituras de até 1 metro de distância do veículo em relação a câmera foram satisfatórias, apesar da assertividade indicada pelo algoritmo de OCR ficar na faixa de 60%, todas leituras seguiram um padrão que pode ser contornado através de alguma alterações em código. Por exemplo, o carácter ‘O’ está sendo facilmente confundido com o carácter ‘Q’.

Ao distanciar em mais 1 metro o veículo em relação a câmera, passando a ter 2 metros de distância no total, a assertividade das leituras indicadas pelo OCR diminuiu em aproximadamente 20%. Notou-se que o algoritmo conseguiu identificar na maioria das vezes os últimos caracteres da placa ‘X2F52’ com sucesso, enquanto

os caracteres iniciais sofreram influência do código ‘QR’ que se encontra no canto superior esquerdo da placa e da sigla ‘BR’ que se encontra no canto esquerdo inferior, tornando a difícil compreensão do caractere correto a distâncias maiores, muitas vezes confundindo com o ‘[‘ ou a letra ‘E’.

Na distância de 3 metros, podemos dizer que é o limite de reconhecimento, dada as condições do teste. Ficando todas com assertividade indicada pelo algoritmo de OCR abaixo de 25%. Herdou-se o mesmo comportamento identificado no teste de 2 metros, onde o código ‘QR’ e a sigla ‘BR’ atrapalharam a leitura dos caracteres iniciais, e dada a distância outros caracteres passaram a ser confundidos, como o número ‘2’ com a letra ‘Z’, dados todos esses erros acumulados, podemos dizer que os resultados nesta distância são tecnicamente inviáveis para ao menos tentar identificar se é uma placa com acesso permitido ou não.

Tabela 2 - Assertividades das leituras

Distância do veículo (m)	Placa Veicular Lida	Assertividade dada pelo OCR (%)
1	OOX2F52	**88,17**
1	QOX2F52	**46,58**
1	OOX2F52	**62,20**
1	OOX2F52	**67,68**
1	OOX2F52	**67,17**
1	OOX2F52	**66,04**
2	OOX2F52	**59,47**
2	OOX2F52	**47,08**
2	OOXZF52	**68,85**
2	EOOX2F5	**31,87**
2	[OX2F52	**43,53**
2	EOX2F52	**57,99**
2	[DOX2F5	**28,30**
3	EQOX2F5	**10,14**
3	[QOXZF5	**14,74**
3	FQOX2F5	**15,19**
3	[DOX2F5	**16,83**
3	[OOX2F5	**25,97**
3	QOXZF52	**29,24**
3	FOOX2F5	**29,74**
3	OOOEF52	**11,20**

Fonte: Elaboração própria (2024).

Na Tabela 3, temos as médias e desvio padrão das assertividades indicadas pelo algoritmo de OCR em relação às distâncias da câmera ao veículo.

Observa-se que a cada 1 metro de distância, a assertividade diminui consideravelmente de forma não linear.

Tabela 3 - Médias e desvio padrão das leituras

Distância do veículo (m)	Média (%)	Desvio Padrão (%)
1	66,30	**12,16**
2	48,16	**13,81**
3	19,13	**7,46**

Fonte: Elaboração própria (2024).

A identificação de cores do veículo, em todos os testes, não foi satisfatória. A implementação baseada na média das cores presentes na captura do veículo e com a verificação de qual das 97 cores mapeadas essa média tem de ficar mais próxima, deixou-se muito a influenciar pela luminosidade do ambiente. Cenários com um mínimo de sombra possível no veículo, tendem a deixá-lo um pouco acinzentado, e dada esta condição, todas as tentativas de leitura de cor do veículo indicavam uma variação de cinza.

A interação com o usuário através de uma interface WEB, cumpriu o seu propósito, sendo responsiva, intuitiva e permitindo realizar todas operações de CRUD de veículos e câmeras. Além de conseguir visualizar a câmera remotamente e realizar o acionamento do portão.

A geração de eventos também ficou dentro do esperado, atingindo um resultado satisfatório, mantendo os ouvintes sempre atualizados com os eventos em tempo real e em caso de desconexão com a rede, manteve-se os eventos em buffer e, após a reconexão com a rede, transmite os eventos armazenados aos ouvintes novamente, gerando o efeito planejado desde o início do projeto.

Por fim, a arquitetura *back end* baseada em conceitos elaborados por grandes profissionais da área de programação, foi de extrema importância. Durante o desenvolvimento do código, foi necessário realizar diversos testes de etapas específicas, além de inúmeras alterações e, dada arquitetura Clean Architecture e princípios SOLID, foi possível realizá-los de forma simples e fácil, provando ser eficaz e funcional na prática.

5 CONSIDERAÇÕES FINAIS

O presente trabalho de conclusão de curso apresentou o desenvolvimento de um sistema automatizado de controle de acesso veicular integrado a uma a câmeras IP e microcontroladores ESP32. O sistema foi projetado para capturar e processar imagens de veículos, utilizando técnicas de visão computacional e protocolos de comunicação, como MQTT e RTSP.

Ao longo do desenvolvimento, foram enfrentados desafios relacionados à exatidão do reconhecimento óptico de caracteres (OCR) e à identificação das cores dos veículos. Os testes realizados revelaram que a exatidão do OCR varia significativamente com a distância entre a câmera e o veículo, com resultados satisfatórios até 1 metro de distância, mas com exatidão reduzida em distâncias maiores. A identificação de cores, por sua vez, mostrou-se sensível às condições de luminosidade e ao ambiente de teste, não satisfazendo ao planejado no início do projeto.

A interface web do sistema demonstrou ser uma ferramenta eficiente e intuitiva para a interação do usuário, permitindo a gestão de veículos e câmeras, além de oferecer a possibilidade de monitorar e controlar o acesso em tempo real. A implementação da Clean Architecture no *back end* garantiu uma estrutura modular e escalável, facilitando a manutenção e futuras expansões do sistema.

Em resumo, o sistema proposto representa uma solução viável para o controle de acesso veicular em condomínios, melhorando a segurança e a eficiência do processo de entrada e saída de veículos. No entanto, ajustes adicionais são necessários para aprimorar a assertividade do reconhecimento de placas a distâncias maiores e a identificação de cores, bem como para tornar a solução totalmente comercializável.

5.1 Sugestões para trabalhos futuros

Para aprimorar o sistema desenvolvido, sugerem-se as seguintes melhorias e extensões para trabalhos futuros:

Melhoria na exatidão do reconhecimento de placas: Desenvolver e integrar algoritmos mais avançados de OCR e visão computacional para aumentar a exatidão da leitura das placas, especialmente em condições de baixa iluminação ou quando o veículo está em movimento. Utilizar a biblioteca Tesseract com modelos pré-treinados com caracteres de placas veiculares brasileiras, para aumentar a exatidão no reconhecimento dos caracteres, combinado com consultas de dados previamente cadastrados no banco de dados para auxiliar no reconhecimento da placa veicular, como, utilizar os últimos 4 algarismos da placa veicular lida para verificar se coincide com alguma informação na tabela de veículos, no intuito de agilizar o reconhecimento e eliminar falhas de leituras como observado nos testes realizados.

Melhoria na Interface Web: Continuar a aprimorar a interface web para torná-la mais intuitiva e fácil de usar, incluindo a adição de mais funcionalidades, como relatórios de acesso detalhados, gráficos de tráfego e indicadores de eventos de acesso.

Expansão do sistema para ambientes comerciais: Adaptar e expandir o sistema para ser usado em estacionamentos de grandes empresas, shoppings e outras áreas comerciais, onde o controle de acesso veicular é fundamental para a segurança e o gerenciamento do fluxo de veículos.

Reconhecimento de outros tipos de placas: Ampliar o sistema para reconhecer diferentes padrões de placas veiculares, não se limitando apenas aos padrões utilizados no Brasil, permitindo a utilização em outros países.

Melhoria no reconhecimento de cor: Utilizar modelo de cor baseado em HSV ao invés de RGB, para eliminar a influência na cor ocasionada pela iluminação e consequentemente obter resultados mais assertivos.

REFERÊNCIAS

AL-GHAILI, Abba, *et al*. **A new vertical edge detection algorithm and its application**. Acesso em: 16 de agosto de 2024.

ALMEIDA, Pedro. **Detectando objetos com YOLO e Darknet.** Disponível em: <https://medium.com/@argonalyst/detectando-objetos-com-yolo-e-darknet-cdb9a17fbc3f>. Acesso em: 18 de abril de 2024.

ANDRADE, Ricardo. **Desenvolvimento de um aplicativo móvel com OCR e reconhecimento de voz para leitura de consumo de água e gás em condomínios**. Acesso em: 17 de abril de 2024.

CHIOU, Yu-Chian; et al. **Optimal locations of license plate recognition to enhance the origin-destination matrix estimation**, v. 2, 2012. Acesso em: 16 de agosto de 2024.

DU, Shan, *et al*. **Automatic license plate recognition (ALPR): a state-of-the-art review**. Acesso em: 16 de agosto de 2024.

FARADJI, Farhad; REZAIE, Amir Hossein; ZIARATBAN, Majid. **A morphological-based license plate location**. Acesso em: 16 de agosto de 2024.

FILHO, Ogê; NETO, Hugo. **Processamento Digital de Imagens**. Acesso em: 17 de abril de 2024.

GODOY, José. **Técnicas de segurança em condomínio**, v. 5. Acesso em: 14 de abril de 2024.

HE, Xiangjian, *et al*. **Segmentation of characters on car license plates**. Acesso em: 18 de agosto de 2024.

HONGLIANG, Bai; CHANGPING, Liu. **A hybrid license plate extraction method based on edge statistics and morphology**. Acesso em: 16 de agosto de 2024.

MARTIN, Robert. **Clean Architecture: A Craftsman 's Guide to Software Structure.** Acesso em: 17 de abril de 2024.

MEIRELLES, Bruno. **Motivações por trás da escolha de uma portaria remota para condomínios residenciais: um estudo de caso da porter rio de janeiro**. Acesso em: 14 de abril de 2024.

MOHD, Amri; KARIM, Mohamed; SAIFIZUL, Ahmad. **Travel time measurement in real-time using automatic number plate recognition for Malaysian environment**, v. 8, 2009. Acesso em: 16 de agosto de 2024.

SARFRAZ, Muhammad; AHMED, M.J.; GHAZI, S.A. **License plate recognition system: Saudi Arabian case**. Acesso em: 16 de agosto de 2024.