UNIVERSIDADE EVANGÉLICA DE GOIÁS - UNIEVANGÉLICA
ENGENHARIA DE COMPUTAÇÃO

MATHEUS GUIMARÃES DOS SANTOS
RUAN DE OLIVEIRA BARBOSA

CANCELA PASS:
CONTROLE DE CANCELAS INTELIGENTE

Anápolis
Novembro, 2021

UNIVERSIDADE EVANGÉLICA DE GOIÁS - UNIEVANGÉLICA
ENGENHARIA DE COMPUTAÇÃO/ENGENHARIA DE SOFTWARE

MATHEUS GUIMARÃES DOS SANTOS
RUAN DE OLIVEIRA BARBOSA

CANCELA PASS:
Controle de Cancelas Inteligente

Trabalho apresentado ao Curso de Engenharia de Computação da Universidade Evangélica de Goiás – UniEVANGÉLICA, da cidade de Anápolis-GO como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Orientador (a): Prof. Ms. Willian Pereira dos Santos

Anápolis
Novembro, 2021

UNIVERSIDADE EVANGÉLICA DE GOIÁS - UNIEVANGÉLICA
ENGENHARIA DE COMPUTAÇÃO

MATHEUS GUIMARÃES DOS SANTOS
RUAN DE OLIVEIRA BARBOSA

CANCELA PASS:
CONTROLE DE CANCELAS INTELIGENTE

Monografia apresentada para Trabalho de Conclusão de Curso de Engenharia de Engenharia de Computação da Universidade Evangélica de Goiás - UniEVANGÉLICA, da cidade de Anápolis-GO como requisito parcial para obtenção do grau de Engenheiro(a) de Computação.

Aprovado por:

_____

Prof. Ms. Willian Pereira dos Santos UniEVANGÉLICA
(ORIENTADOR)

_____

Prof. Ms. Gino Bertollucci Colherinhas UnB
(AVALIADOR)

Prof. Ms. Henrique Valle de Lima UFG
(AVALIADOR)

Anápolis, 10 de dezembro de 2021.

FICHA CATALOGRÁFICA

SANTOS, Matheus, BARBOSA, Ruan. Cancela Pass: Controle de Cancelas Inteligente. Anápolis 2021. (Universidade Evangélica de Goiás – UniEVANGÉLICA).

Monografia. Universidade Evangélica de Goiás, Curso de Engenharia de Computação, da cidade de Anápolis-GO.

1. Cancela. Reconhecimento de imagens. Placas veiculares.

CESSÃO DE DIREITOS

NOMES DOS AUTORES: Matheus Guimarães dos Santos, Ruan de Oliveira Barbosa
TÍTULO DO TRABALHO: Controle de Cancelas Inteligente
GRAU/ANO: Graduação / 2021

_____
Matheus Guimarães dos Santos
Anápolis, 21 de novembro de 2021

_____
Ruan de Oliveira Barbosa
Anápolis, 21 de novembro de 2021

RESUMO

Este projeto tem como foco desenvolver uma aplicação de reconhecimento automático de placas veiculares por meio de imagens usando a biblioteca OpenCV para a leitura e processamento das imagens obtidas, é feito o uso de técnicas de Processamento Digital de Imagens (PDI) e de reconhecimento de caracteres através da aplicação da API Google Vision. A partir destas técnicas propomos desenvolver um sistema que realiza o registro de entrada e a saída de carros em cancelas. A partir da observação da falta de registro de quais veículos entram ou saem do estacionamento do Centro Universitário de Anápolis, verificou-se a necessidade da produção e publicação deste trabalho. Para isso, foi analisado os processos necessários para essa técnica como identificar a placa veicular para assim poder saber a data, hora da entrada saída do veículo. Com essa publicação, espera-se que após a implementação do sistema, haverá uma maior segurança para os que usam o espaço e para a própria Instituição.

Palavras-chave: Cancela. Reconhecimento de imagens. Placas veiculares

LISTA DE ILUSTRAÇÃO

SUMÁRIO

1. INTRODUÇÃO

1.1 Problema

O Centro Universitário de Anápolis possui uma grande extensão territorial somente para o uso de estacionamento e uma grande quantidade de alunos que fazem o uso do mesmo, assim gerando um descontrole nos horários de pico de quem entra e sai de suas dependências.

Atualmente é usando um cartão de acesso para realizar o controle de entrada e saída da unidade, mas o controle de visitas é feito pelo aplicativo #TônaUni, onde é recolhido o CPF da pessoa que está conduzindo o carro e uma foto da mesma, porem não é feito o registro do veículo.

Havendo a falta de registro, há a possibilidade de que o veículo dos visitantes, alunos e funcionários sejam danificados por algum veículo que não teve o seu registro de entrada e saída na unidade, assim a instituição não poderá provar pelo meio convencional que aquele veículo esteve na unidade naquele dia.

Partindo de que é observado os métodos já usados na instituição para o registro e monitoramento dos veículos que entram e saem da unidade e das possíveis falhas, esse trabalho desenvolve uma solução para possíveis falhas que responderão ao seguinte problema de pesquisa: Criar um sistema para registrar a entrada e saída de veículos da unidade resolverá o problema proposto acima?

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um sistema de reconhecimento de placas veiculares para realizar o registro de entradas e saídas de veículos da Instituição.

1.2.2 Objetivos Específicos

- Desenvolvimento de um algoritmo para realizar o reconhecimento das placas veiculares;
- Trabalhar com os dados levantados na criação indicadores e relatórios

1.3 Justificativa

Esse projeto tem como foco desenvolver um sistema que será utilizado para o controle de registros de entrada e saída de veículos da Instituição. Para isso serão utilizado técnicas para o reconhecimento das placas veiculares para registro da data e hora. Com o desenvolvimento dessa produção tecnológica, tem como objetivo apropriar-se da realidade para melhor analisá-la, produzir transformações e discutir sobre os impactos do uso dos registros levantados.

“A automatização de processos consiste em transformar etapas que eram realizadas de maneira estritamente manual em procedimentos que contam com a ajuda de tecnologia.” (Mercado Eletrônico, 2019) O processo de automatização é muito delicado e exige um período muito grande de análises e testes com o sistema que será implantando, mas automatizar um processo significa otimizar tempo, dinheiro e recursos. Analisando a situação da falta de informação levantada nas cancelas será oferecida uma vantagem para a Instituição.

2. FUNDAMENTAÇÃO TEÓRICA

1.1 Imagem digital

Uma imagem digital pode ser definida por uma função bidimensional, da intensidade de luz refletida ou emitida por uma cena (INPE/DPI, AFFONSO, 2002).

Segundo Gonzalez e Woods. (2010, p. 1)

> Uma imagem pode ser definida como uma função bidimensional, f (x, y), em que x e y são coordenadas espaciais (plano), e a amplitude de f em qualquer par de coordenadas (x, y) é chamada de intensidade ou nível de cinza da imagem nesse ponto. Quando x, y e os valores de intensidade de f são quantidades finitas e discretas, chamamos de imagem digital.

A imagem digital possui várias características como, possuir um número finito de bits para representar a dimensão da cena para cada pixel, banda espectral, resolução, resolução espacial, resolução espectral, resolução radiométrica.

A Figura 1 exibe a primeira imagem digital foi feita por Russell Kirsch em 1957 de dimensão de 176x176 pixels, utilizando de um dispositivo que transformava as imagens em matrizes de zeros e uns.

Figura 1: Primeira imagem digital, feita por Russell Kirsch

Fonte: (Mdig, 2010).

1.2 Processamento de imagens

O processamento de imagens são técnicas utilizadas para a análise de dados multidimensionais através da manipulação de uma imagem por computador, sendo que a entrada e a saída do processo sejam imagens. Tem como o objetivo melhorar o aspecto visual para proporcionar uma maior facilidade durante a extração de informações. (INPE/DPI, AFFONSO, 2002)

Normalmente o processamento de imagens envolve procedimentos expressos sob forma algorítmica. Então a maioria das funções de processamento de imagens podem ser implementadas via software. O uso de hardware para o processamento de imagem se faz necessário somente quando o computador onde está sendo feito o processamento possui certas limitações. (MARQUES FILHO; VIEIRA NETO, 1999)

1.3 OpenCV

OpenCV é uma biblioteca “open source” (código aberto) que possui módulos de Processamento de Imagens e Vídeo, onde será usando a função *“cv.adaptiveThreshold”*, que intensifica o que é importante na imagem que será analisada.

Segundo Mordvintsec e K (2013, p. 7)

> O seu processamento de imagens é em tempo real. Atualmente suporta uma grande variedade de linguagens de programação como C++, Python, Java etc. e está disponível em diferentes plataformas, incluindo Windows, Linux, OSX, Android, iOS, etc. Além disso, interfaces baseadas em CUDA e OpenCL também estão em desenvolvimento ativo para alta operações de velocidade de GPU.

A função “*cv.adaptiveThreshold*” determina o limite para o píxel com base em uma pequena região ao redor da mesma. Assim é obtido diferentes regiões na mesma imagem fornecendo melhores resultados para imagens onde a iluminação é variável.

O código abaixo na Figura 2 compara a função “cv.THRESH_BINARY” com as funções adaptativas do Threshold “cv.ADAPTIVE_THRESH_MEAN_C” e “cv.ADAPTIVE_THRESH_GAUSSIAN_C”.

A função cv.THRESH_BINARY é a função padrão para a binarização onde os pixels são divididos em preto ou branco, ela transforma todos os pixel que estão acima 128 da escala de cinza para o preto 255 e para branco os que estão abaixo ou igual a 128.

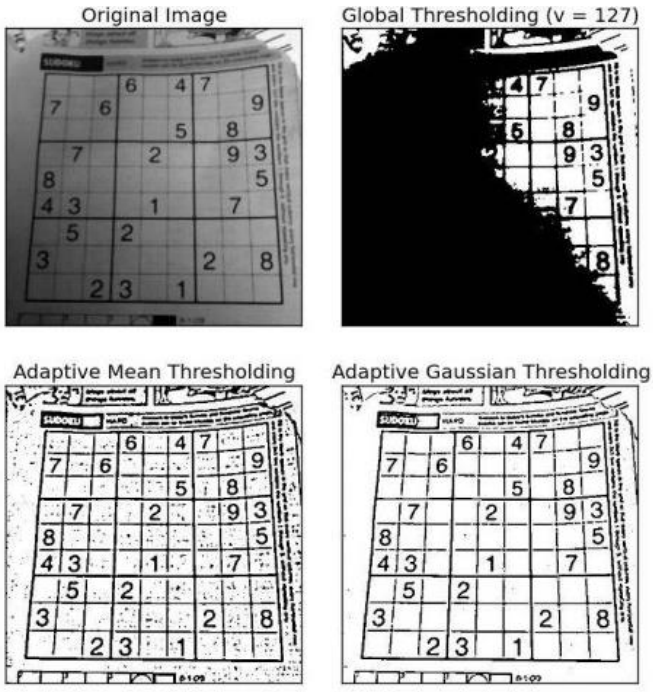
A função cv.ADAPTIVE_THRESH_MEAN_C diferente do binário, o valor do píxel é a média do valor dos pixels vizinhos a ele, assim trazendo uma melhor percepção da imagem quando há sombra. E por último a função cv.ADAPTIVE_THRESH_GAUSSIAN_C onde o valor do píxel será a soma gaussiana ponderada do valor de seus vizinhos, onde diferente das outras a mesma consegue realizar uma melhor retirada dos ruídos da imagem conforme a figura 3 que representa o uso das 3 funções.

Figura 2: Código comparando as funções do Threshold

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread('sudoku.png',0)
img = cv.medianBlur(img,5)

ret,th1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
th2 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_MEAN_C,\
            cv.THRESH_BINARY,11,2)
th3 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_GAUSSIAN_C,\
            cv.THRESH_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v = 127)',
            'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
images = [img, th1, th2, th3]

for i in xrange(4):
    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

Fonte: (OpenCV, 2020).

Figura 3: Resultado do uso do código.



Fonte: (OpenCV, 2020).

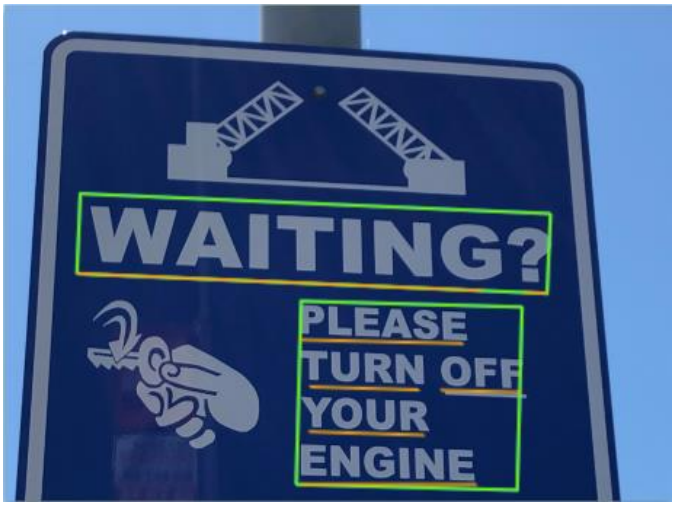
1.4 Reconhecimento óptico de caracteres (OCR - *Optical Character Recognition*)

O reconhecimento óptico de caracteres é uma tecnologia muito importante e está presente em muitos lugares como em balcões de checkout de supermercados ou lojas, máquinas de troca de dinheiro, máquinas de escaneamento e sistemas postais que utilizam scanner como leitor de código, e essa ferramenta pode ser utilizada para transformar documentos digitalizados em dados que podem ser pesquisáveis ou editáveis.

A ferramenta funciona da seguinte forma, é analisado o documento e comparado os seus caracteres com fontes armazenadas em seu banco de dados, podemos usar como exemplo a digitalização de algumas páginas de um livro para enviar suas informações para serem editadas ou simplesmente para transformar essas informações para o digital.

A plataforma Google utiliza dessa ferramenta para consulta de palavras em fotos salvas no Google Drive e no Google Photos do usuário, como apresentado na Figura 6, Figura 7 e Figura 8, e também possui uma API (Interface de Programação de Aplicativos) de reconhecimento ótico de caracteres chamada API Vision e que pode ser utilizada para implementação em outros aplicativos. Trata-se da extração de textos de imagens, utilizando como parâmetro TEXT_DETECTION, como apresentado na Figura 4, podemos extrair qualquer texto da imagem, podendo ser uma placa de rua ou de trânsito, ou usar DOCUMENT_TEXT_DETECTION, como apresentado na Figura 5, para uma extração otimizada para textos e documentos densos.

Figura 4: Exemplo da imagem processada pelo OCR da API Vision usando TEXT_DETECTION.

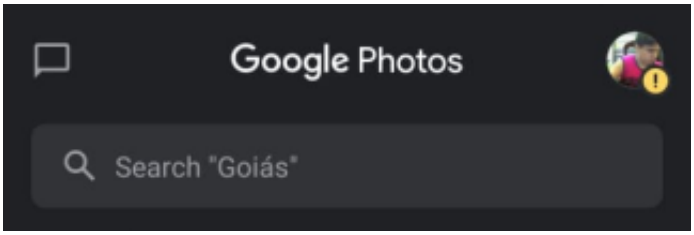


Fonte: (Google Cloud, 2020).

Figura 5: Exemplo da imagem processada pelo OCR da API Vision usando DOCUMENT_TEXT_DETECTION.

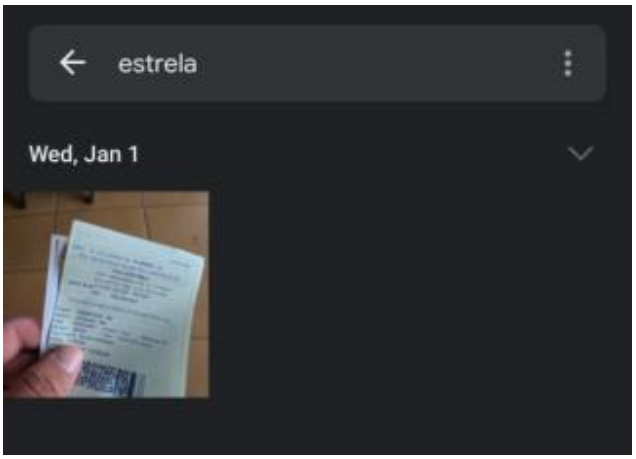
Fonte: (Google Cloud, 2020).

Figura 6: Campo de pesquisa do Google Photos.



Fonte: (Autores, 2020).

Figura 7: Resultado ao pesquisar “estrela” no Google Photos.



Fonte: (Autores, 2020).

Figura 8: Imagem retornada Google Photos.

Fonte: (Autores, 2020).

1.5 API Cloud Vision

A API Cloud Vision faz parte da plataforma Google Cloud e a mesma oferece detecção e classificação de objetos utilizando modelos baseados em Redes Neurais (RN) pré-treinadas.

Segundo Google Cloud (2021, p. 1)

> A API Cloud Vision permite que os desenvolvedores entendam o conteúdo de uma imagem ao encapsular modelos avançados de machine learning em uma API REST fácil de usar. Ela classifica rapidamente imagens em milhares de categorias ("barco", "Leão", "Torre Eiffel"), detecta objetos individuais em imagens e encontra e lê palavras impressas contidas nas imagens.

Pressman e Maxim (2016), descreve que o uso de API (Interface de Programação de Aplicativos) desobriga os desenvolvedores a integrar código-fonte de serviço no cliente, em vez disso, o serviço é executado fora do servidor provedor. Sabido que a API Cloud Vision possui serviço de transferência representacional de estado (REST), então para que a API seja utilizada é necessário usar uma PRIVATE_KEY que é a identificação do usuário na plataforma Cloud Vision, no qual a mesma é enviada através de um arquivo

JSON gerado pela Cloud Vision, que é definido na variável ambiente GOOGLE_APPLICATION_CREDENTIALS descrito na Figura 9.

Figura 9: Exemplo de definição da variável ambiente.

```
$env:GOOGLE_APPLICATION_CREDENTIALS="C:\Users\username\Downloads\service-account-file.json"
```

Fonte: (Google Cloud, 2021).

1.6 Padrão das placas veiculares

Na década de 90 as placas do brasil passaram por uma padronização que ainda é usada nos carros o modelo é formado por 3 letras e 4 números inteiros como exibido na Figura 10, possibilitando que tenha 150 milhões de combinações possíveis, permitindo essa quantidade de carros.

Figura 10: Exemplo padrão de placas da década de 90

Fonte: (Qcveiculos, 2020).

Em 31 de janeiro de 2020 as novas placas do Mercosul entraram em vigor e se tornaram obrigatórias em novos veículos e em situações que necessite a troca como quando a mesma está danificada, quando há uma mudança de município ou furto. Na Figura 11 diferente da placa citada anteriormente a nova placa possui 3 letras, 1 número, 1 letra e 2 números nessa ordem.

Apesar de haver dois tipos de placas a quantidade de caracteres que fazem parte da identificação da placa ainda continuam iguais, com isso o tratamento dos caracteres continua o mesmo que é retirar o hífen da placa do modelo dos anos 90.

Figura 11: Itens de segurança da placa Mercosul

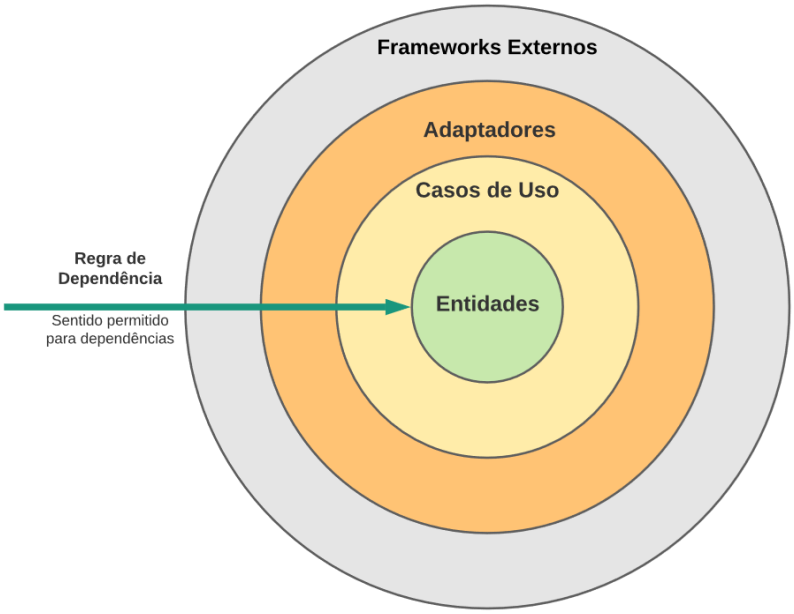
Fonte: (Quatro Rodas, 2020).

1.7 Arquitetura Limpa

Martin (2018), descreve que quanto mais uma arquitetura é engessada a uma forma novos recursos tem a probabilidade de serem mais difíceis de encaixar nessa arquitetura, então as arquiteturas devem ser tão agnósticas em sua forma quanto práticas.

A arquitetura limpa (The Clean Architecture) é um padrão que foi definido por Robert Martin, mais conhecido como Uncle Bob(Tio Bob), com a premissa de desenvolver uma implementação que poderia facilitar o reuso do código, coesão e testabilidade, é considerada uma arquitetura de camadas onde é proposto a independência entre suas classes sendo elas de comunicação externa ou interna. A figura 12 é a representação gráfica da arquitetura.

Figura 12: Arquitetura Limpa



Fonte: (Engenharia de Software Moderna, 2021).

Cada camada é implementada de forma independente, porém elas comunicam entre si através da responsabilidade atribuída.

A camada de frameworks externos é responsável pela comunicação com outras interfaces e a exposição de rotas da aplicação, é uma camada responsável pela conexão com o banco de dados, com outra aplicação de terceiros e com a interface para usuários.

A camada de adaptadores é uma camada composta por classes e interfaces e é responsável pela comunicação e mapeamento das entidades da camada externa com a camada interna e o caminho inverso também.

A camada de casos de usos é um conjunto de classes responsáveis pelas regras de negócio da aplicação, onde que deve ser implementado regras específicas para cada negócio.

A camada de entidades é a camada que possui as classes comuns de uma aplicação e são classes responsáveis por serem a representação dos dados de uma determinada entidade, podendo conter regras de negócio de forma genérica.

3. METODOLOGIA

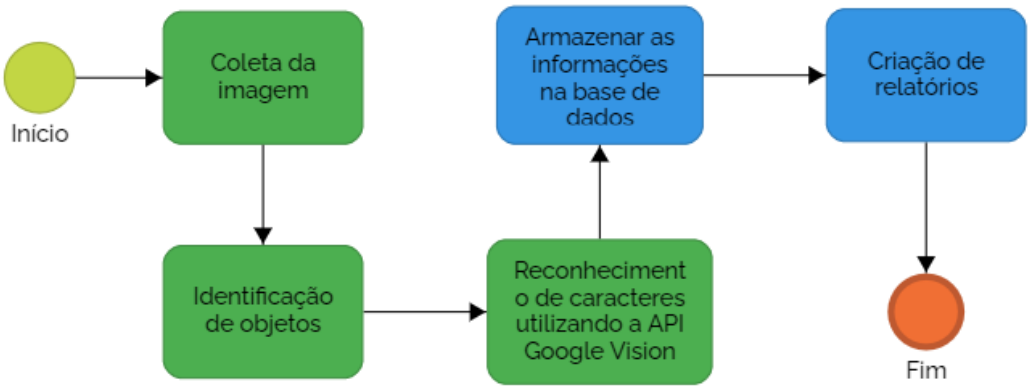
O trabalho foi iniciado com a coleta de referencial teórico sobre inteligência artificial em processamento de imagens, seguido pelos itens referentes a algoritmo de reconhecimento de placas veiculares e desenvolvimento de uma plataforma web para tratar os dados levantados.

Como apresentado na Figura 12, as imagens serão coletadas e depois processadas utilizando o OpenCV com o uso do método "*INTER_AREA"* para realizar a interpolação da imagem afim de melhorar a qualidade e evitando que os caracteres dentro da imagem fiquem distorcidos, após realizado esse tratamento da imagem, será utilizado a API do Google Vision para coletar os caracteres de dentro da imagem.

Com o algoritmo implementado, o mesmo será testado em placas reais de carros, e avaliado para que seja realizado o levantamento de dados, como data e hora do reconhecimento.

Com a análise dos dados levantados possibilitará a criação de relatórios informando data e hora de entrada e saída do estacionamento da faculdade.

Figura 13: Diagrama de processos metodológico do trabalho.



Fonte: (Autores, 2021).

4. DESENVOLVIMENTO

Este capítulo irá apresentar os passos realizados para o desenvolvimento da aplicação como solução o gerenciamento de registros de entrada e saída de veículos dentro da Instituição ou em locais que possuem cancelas para controle de entrada e saída através do uso de reconhecimento de imagens para identificar a placa veicular.

4.1 Projeto

Nesta etapa foi analisado as linguagens a serem usadas para o desenvolvimento, definimos que seriam dois sistemas, um responsável pela identificação e o reconhecimento das placas veiculares, e o outro sistema responsável pelo registro das informações obtidas registrando as entradas e saídas dos veículos. Para o sistema de identificação e reconhecimento das placas veiculares foi utilizado Python e para o sistema de gerenciamento foi utilizado React para a criação da interface, no backend foi utilizado Java e o banco de dados PostgreSQL.

4.2 Reconhecimento das placas veiculares

Nessa etapa foi utilizada duas entradas de vídeo para simular a posição de cada câmera posicionada na cancela, na figura 13 podemos ver como é feito a utilização da câmera utilizando a biblioteca OpenCV.

Figura 14: Função VideoCapture.

```
cap_entrada = cv2.VideoCapture('videos/video.mp4')
cap_entrada.set(3, 640)
cap_entrada.set(4, 480)

cap_saida = cv2.VideoCapture('videos/video.mp4')
cap_saida.set(3, 640)
cap_saida.set(4, 480)
```

Fonte: (Autores, 2021).

A função pode ser utilizada para abrir a utilização de uma câmera ligada a máquina ou para a leitura de um vídeo informando o caminho dele dentro da máquina, neste caso utilizamos um vídeo sendo importado de dentro da pasta “vídeos” no formato de mp4.

Para a identificação de veículos, a captura de imagens e a gravação dessas imagens foi utilizado alguns comandos da biblioteca OpenCV juntamente com um conjunto de dados pré-treinados.

A função representada na figura 14 é responsável pela a identificação do veículo e é chamada de “reconheceVeiculo”, nela podemos verificar se na imagem que foi retirada do vídeo contém um veículo para poder realizar contorno em volta do mesmo, e após isso, é verificado se o veículo está próximo o suficiente para ser detectado a imagem.

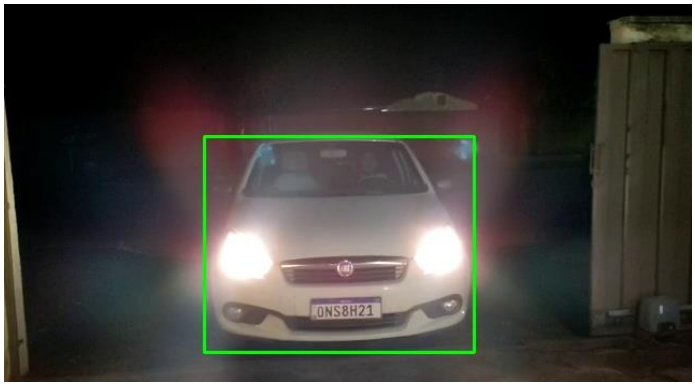
Figura 15: Método responsável pelo reconhecimento do veículo.

```
def reconheceVeiculo(img, tipo):
    class_ids, confs, bbox = net.detect(img, confThreshold=0.5)
    if len(class_ids) != 0:
        for class_id, confidence, box in zip(class_ids.flatten(), confs.flatten(), bbox):
            if class_id == 3:
                cv2.rectangle(img, box, color=(0, 255, 0), thickness=2)

                is_car = (box[0] == 186 and box[1] == 123 and box[2] == 252 and box[3] == 202)
                if is_car:
                    cv2.imwrite("imagens/registro.jpg", img)
                    return 200

    cv2.imshow(str(tipo), img)
    cv2.waitKey(1)
```

Fonte: (Autores, 2021).

Após realizado a identificação do veículo é registrado uma imagem na qualidade capturada para a identificação dos caracteres, como representado na figura 15.

Figura 16: Imagem gerada com o contorno no veículo na resolução de 640x352.



Fonte: (Autores, 2021).

A partir desse passo já temos a primeira imagem onde poderemos trabalhar para a identificação dos caracteres da placa.

Pelo motivo da resolução da imagem está em uma resolução em uma escala menor de 640x352 foi necessário criar um método responsável por redimensionar a imagem para uma escala maior e assim ficar mais fácil a identificação dos caracteres, apresentado na figura 16.

Figura 17: Redimensionamento de imagem

```
def resize(image, tipo):
    imageFile = image
    img = cv2.imread(imageFile, 0)

    scale_percent = 220
    width = int(img.shape[1] * scale_percent / 100)
    height = int(img.shape[0] * scale_percent / 100)
    dim = (width, height)

    resized = cv2.resize(img, dim, interpolation=cv2.INTER_AREA)
    ext = ".jpg"
    cv2.imwrite("imagens/BILINEAR" + ext, resized)

    placa = detect_text("imagens/BILINEAR.jpg")
    if (len(placa) != 0):
        if (len(placa) == 7):
            registrarEstacionamento.registrar(placa, tipo)

    return placa
```

Fonte: (Autores, 2021).

Na figura 17 mostra que foi utilizado a função “imread” para realizar a leitura da imagem passando como 0 (zero) o valor para abrir a imagem em escala de cinza, a função “resize” utilizada para o redimensionamento da imagem para melhorar a qualidade da imagem em pixels, a função “imwrite” utilizada para gravar a nova imagem.

.

Figura 18: Imagem gerada após o redimensionamento na resolução de 1408x774.

Fonte: (Autores, 2021).

Após realizar o redimensionamento para melhorar a qualidade dos pixels retirados da primeira imagem é chamado o método “detect_text” que é responsável pela a detecção de texto dentro da imagem. Na figura 18 utilizamos a chamada da API Vision para enviar a imagem gerada para a identificação dos caracteres.

Figura 19: Método responsável pela comunicação com a API Vision.

```
def detect_text(path):
    client = vision.ImageAnnotatorClient()
    with io.open(path, 'rb') as image_file:
        content = image_file.read()
    image = vision.Image(content=content)
    response = client.text_detection(image=image)
    texts = response.text_annotations

    conv = texts[0].description.splitlines()

    for text in conv:
        if (len(text) == 7):
            conv = text
        elif (len(text) > 7):
            if (text[3] == '-' or text[3] == ' '):
                text = text.replace(text[3], "")

            conv = text

    return conv
```

Fonte: (Autores, 2021).

Para a comunicação com a API Vision é realizado a leitura da imagem redimensionada para converter ela em bytes para realizar a requisição. Sendo assim, possível a obtenção dos caracteres encontrados na imagem, o retorno é um conjunto de

caracteres onde os valores serão validados para identificar o valor correto contido na placa veicular.

Após a identificação dos caracteres é realizado a chamada da função apresentada na figura 19 para enviar as informações para a API responsável pelo registro das informações no banco de dados.

Figura 20: Comunicação com a API de registro.

```
def registrar(placa, tipo_registro):
    response = requests.post("http://localhost:8080/api/gerenciar-estacionamento/",
                             json={"placa": placa, "tipo_registro": tipo_registro}, headers=headers)

    if(tipo_registro == 1):
        tipo_registro = 'Entrada'
    elif (tipo_registro == 2):
        tipo_registro = 'Saida'

    if(response.status_code == 200):
        print("Success, " + tipo_registro + " placa: " + placa)
    else:
        print("Error, " + tipo_registro + " placa: " + placa)
```

Fonte: (Autores, 2021).

3.3 – Gerenciamento de registros

Nessa etapa de desenvolvimento foi utilizado o framework Spring Boot, com o Java na versão 11, foi utilizando as práticas do Clean Arch (Clean Architecture) separando as suas camadas com as suas respectivas funções.

A camada de controller é responsável pela a exposição da rota da API para ser acessada por algum aplicativo ou outra API. A figura 20 mostra o método do controller responsável pela inserção dos dados dos registros recebidos da API de reconhecimento veicular.

Figura 21: Método responsável por realizar registro.

```
@PostMapping("/")
public ResponseEntity<?> gerenciarRegistroEstacionamento(@RequestBody RegistroEstacionamentoRequest request) {
    try {
        var response : RegistroEstacionamentoResponse = gerenciarRegistroEstacionamentoUseCase.registrar(request);
        return ResponseEntity.ok(new ResponseData<>(response));
    } catch (ServiceException ex) {
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(new ResponseData<>(ex.getMessage()));
    }
}
```

Fonte: (Autores, 2021).

A figura 21 mostra o método do controller responsável pela consulta dos dados dos registros na base de dados.

Figura 22: Método responsável por consultar os registros.

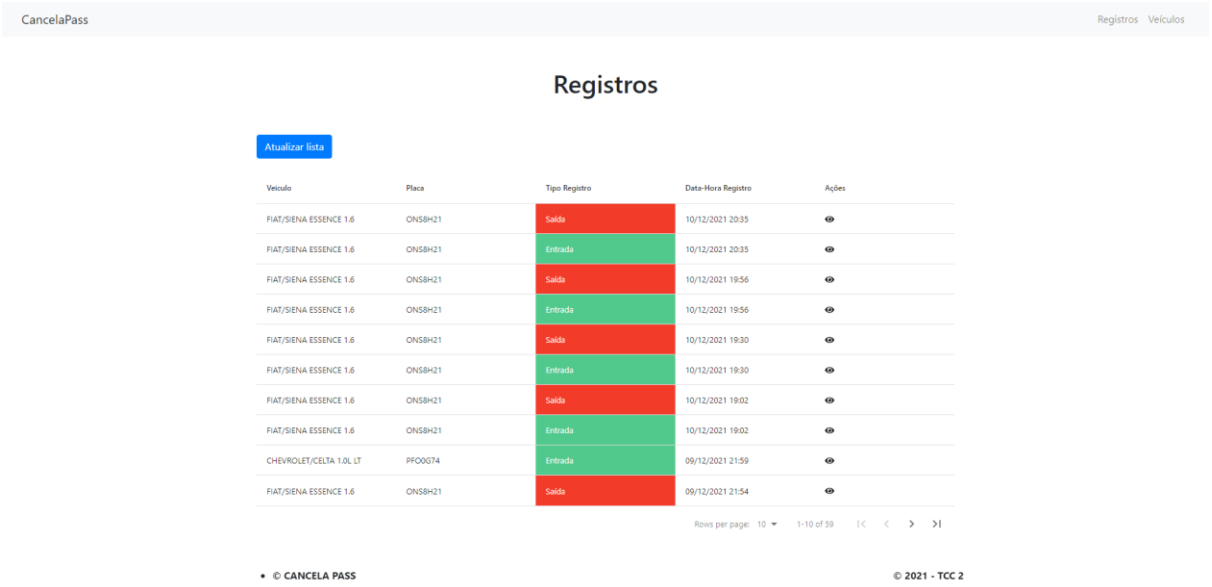
```
@GetMapping("/")
public ResponseEntity<?> getRegistros() {
    try {
        List<RegistroEstacionamentoResponse> response = gerenciarRegistroEstacionamentoUseCase.getRegistros();
        return ResponseEntity.ok(new ResponseData<>(response));
    } catch (ServiceException ex) {
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(new ResponseData<>(ex.getMessage()));
    }
}
```

Fonte: (Autores, 2021).

Este método é o responsável pela a apresentação dos dados apresentados pelo o front-end, neste método é consultado todos os registros já realizados, com as informações dos veículos e contendo as informações sobre os horários dos registros.

Na figura 22 é a tela apresentada no front-end, contendo a listagem com as informações dos veículos e o seu registro de entrada ou de saída da cancela.

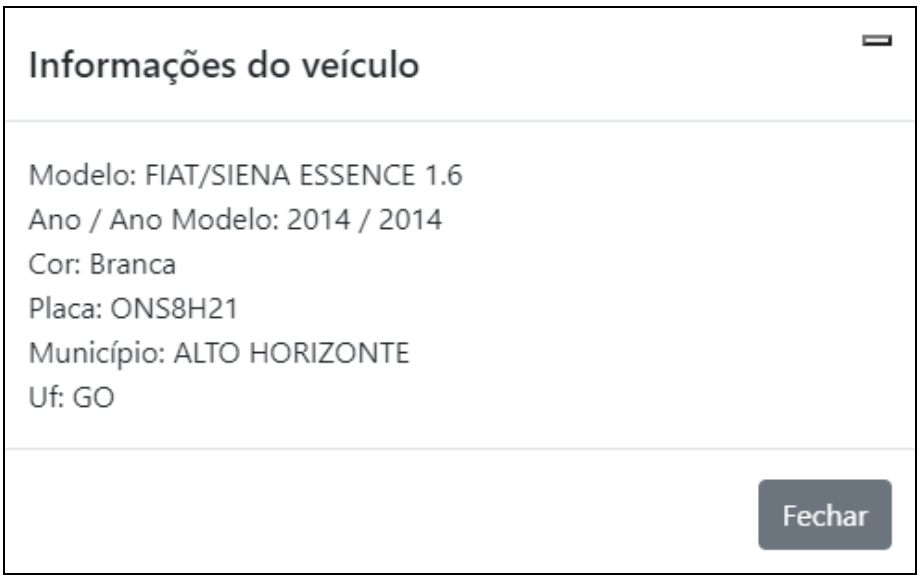
Figura 23: Método responsável por consultar os registros.



Fonte: (Autores, 2021).

Nesta tela do aplicativo é possível de visualizar todos os registros, organizar os mesmos por horário e por tipo de registro, é possível também realizar a ação de visualizar para ver com mais detalhes as informações sobre o veículo, como mostrado na figura 23.

Figura 24: Tela de informações detalhadas do veículo.



Fonte: (Autores, 2021).

Ao selecionar a opção para visualizar as informações detalhadas é apresentadas as informações sobre o modelo, ano, cor, placa, município e a unidade federativa do veículo, podendo dessa forma mais fácil a identificação caso seja necessário de encontrar o mesmo dentro do estacionamento.

5. CONCLUSÃO E CONSIDERAÇÕES FINAIS

Com a realização deste trabalho possibilitou entender os conceitos de imagem digital, processamento de imagens e a biblioteca OpenCV. Dando embasamento para a criação de um algoritmo de reconhecimento de placas veiculares com a API Google Vision, assim conseguimos desenvolver uma solução para o problema em questão.

Espera-se que com os estudos e algoritmos desenvolvidos seja facilitado com os registros dos veículos que entram dentro do estacionamento da instituição, dando um melhor controle e indicativos para a mesma.

Com o decorrer do projeto diversas ideias foram levantadas, mas o foco foi primeiramente na funcionalidade de registrar os veículos resolvendo os problemas que foram apresentados, mas é pretendido futuramente que seja implementado além da identificação da placa do veículo, também a abertura da cancela automaticamente caso o veículo tenha sido registrado previamente no sistema.

6. REFERÊNCIAS

CENTRO TECNOLÓGICO DE ACESSIBILIDADE. **Ferramentas OCR – entenda o que são e sua relação com a acessibilidade**. Disponível em: https://cta.ifrs.edu.br/ferramentas-ocr-entenda-o-que-sao-como-funcionam-e-qual-sua-relacao-com-a-acessibilidade/. Acesso em: 18 jun. 2020.

FILHO, Ogê Marques; NETO, Hugo Vieira. **Processamento Digital de Imagens**. 1. ed. Rio de Janeiro: Brasport, 1999.

GONZALEZ, Rafael C.; WOODS, Richard C.. **Processamento digital de imagens**. 3. ed. [S.l.]: Pearson Universidades, 2009. p. 0-624.

GOOGLE CLOUD. **Produtos de machine learning e IA** . Disponível em: https://cloud.google.com/vision/docs/ocr. Acesso em: 18 jun. 2020.

MDIG. **A primeira imagem digital da história (1957)**. Disponível em: https://www.mdig.com.br/index.php?itemid=15402. Acesso em: 9 mai. 2020.

MERCADO ELETRÔNICO. **Automatização de processos: O que é e quais são os benefícios?**. Disponível em: https://blog.mercadoe.com/automatizacao-de-processos/. Acesso em: 11 mai. 2020.

MORDVINTSEV, Alexander; K, Abid. **OpenCV-Python Tutorials Documentation**: Release. 1. ed. [S.l.]: Google Summer Of Code, 2013. p. 1-273.

OPENCV. **Image Thresholding**. Disponível em: https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html. Acesso em: 12 mai. 2020.

GOOGLE CLOUD. **Documentação do Google Vision** . Disponível em: https://cloud.google.com/vision/docs/. Acesso em: 18 nov. 2021.

PRESSMAN, Roger S, MAXIM, Bruce R. **Engenharia de Software: Uma abordagem profissional.** 8. Ed. Porto Alegre: AMGH, 2016.

QC VEÍCULOS. **Tipos de placas de automóveis: o que significa cada uma?**. Disponível em: http://qcveiculos.com.br/placas-de-automoveis/. Acesso em: 14 mai. 2020.

QUATRO RODAS. **Novo padrão de placas começará a ser usado no Brasil amanhã**. Disponível em: https://quatrorodas.abril.com.br/noticias/novo-padrao-de-placas-comecara-a-ser-usado-no-brasil-amanha/. Acesso em: 15 mai. 2020.

GOOGLE CLOUD. **Documentação do Google Vision** . Disponível em: https://cloud.google.com/vision/docs/. Acesso em: 18 nov. 2021.

MARTIN, Robert C. **Clean Architecture**: A Craftsman’s Guide to Software Structure and Desing. 1. Ed. 2018 p. 1-442

ENGENHARIA DE SOFTWARE MODERNA. **Construindo Sistemas com uma Arquitetura Limpa** . Disponível em: https://engsoftmoderna.info/artigos/arquitetura-limpa.html. Acesso em: 12 dez. 2021.