Projeto Integrador – Sistema de Segurança com Inteligência Artificial para Portaria do Centro Universitário Salesiano de São Paulo - UNISAL

Engenharia de Computação - 5° Semestre – Campus Maria Auxiliadora – UNISAL – 2023 – Americana/SP

Engenheiros de T.I

Davi Ricci Archângelo, RA: 210017523
Higor Caetano Dias, RA: 210015889
Pedro Henrique V. da Silva, RA: 210017057
Vitor Cesar Lulio, RA: 210018003

Resumo *- O presente projeto apresenta a elaboração de um sistema destinado ao monitoramento da portaria do Centro Universitário Salesiano de São Paulo UNISAL. Por meio de uma aplicação de software e hardware para a finalidade de controle de acesso a unidade de ensino. Como resultados, apresentamos uma aplicação com capacidade para realização de piloto.*

Palavras-chave*:* Controle de Acesso, Sistema de segurança, Monitoramento, Sustentabilidade, Responsabilidade social, Instituição de Ensino.

Abstract *- IT Engineers were hired with the aim of developing a system that monitors the entry and exit of vehicles from the UNISAL college, with the aim of promoting greater safety for students and employees of the education network.*

I. Introdução

A segurança é um tema de extrema importância em qualquer ambiente, especialmente em locais com grande circulação de pessoas e veículos, como é o caso de uma faculdade. Uma portaria é a principal forma de controle de acesso em um ambiente restrito, sendo responsável por identificar e autorizar a entrada de pessoas e veículos.

De acordo com o estudo de Mendes et al. (2019), "a implementação de sistemas de segurança eficientes nas portarias é fundamental para garantir a proteção dos indivíduos e prevenir situações de risco" (p. 78). Para alcançar esse objetivo, é essencial utilizar tecnologias avançadas que possibilitem o controle e monitoramento efetivos de todos os acessos à portaria.

Com o avanço da tecnologia, o desenvolvimento de um software capaz de captar e ler as placas dos veículos emerge como uma solução eficiente e automatizada para otimizar esse processo de controle de acesso (Silva, 2020). Essa abordagem permite um registro preciso e rápido das informações dos veículos que adentram a faculdade, contribuindo para a segurança e organização do ambiente.

II. Referencial Teórico

As portarias de faculdades concentram um grande fluxo de pessoas e veículos, o que aumenta a necessidade de medidas de segurança efetivas. De acordo com o Anuário Brasileiro de Segurança Pública de 2021, o roubo de veículos foi o terceiro crime mais comum no país em 2020, com mais de 160 mil casos registrados (ANUÁRIO BRASILEIRO DE SEGURANÇA PÚBLICA, 2021). A implementação de sistemas de segurança nas portarias de faculdades pode ajudar a prevenir esse tipo de crime e aumentar a segurança dos estudantes, professores e funcionários.

Levando isso em consideração, podemos ressaltar que a utilização de um sistema de controle de acesso seria o ideal, pois segundo o Relatório de Mercado de Sistemas de Controle de Acesso Globais de 2020, o mercado global de sistemas de controle de acesso deve crescer a uma taxa composta de 6,8% entre 2020 e 2027 (RELATÓRIO DE MERCADO DE SISTEMAS DE CONTROLE DE ACESSO GLOBAIS, 2020). Esse crescimento é impulsionado pela crescente preocupação com a segurança em ambientes públicos e privados.

A inteligência artificial tem se mostrado uma tecnologia cada vez mais promissora para aprimorar sistemas de segurança. Segundo o relatório "AI in Security - Annual Review 2020", publicado pela consultoria britânica MarketsandMarkets, o mercado global de segurança baseada em inteligência artificial deve crescer a uma taxa composta de 31,2% entre 2020 e 2025 (MARKETSANDMARKETS, 2020). Isso indica um grande potencial para a aplicação de IA em sistemas de segurança para portarias de faculdades.

Com o uso da inteligência artificial, iremos fazer o reconhecimento de placas veiculares onde o controle de entrada e saída de veículos é essencial. De acordo com a empresa de pesquisa de mercado Mordor Intelligence, o mercado global de sistemas de reconhecimento de placas veiculares deve crescer a uma taxa composta de 10,7% entre 2021 e 2026 (MORDOR INTELLIGENCE, 2021). Esse crescimento é impulsionado pelo aumento da

demanda por soluções de segurança em aeroportos, estacionamentos e outros ambientes.

A arquitetura de sistemas de segurança é um aspecto fundamental para garantir a efetividade e confiabilidade de um sistema de controle de acesso. Segundo o Instituto Nacional de Padrões e Tecnologia (NIST), a arquitetura de segurança deve incluir quatro camadas: a camada de controle de acesso, a camada de detecção, a camada de resposta e a camada de auditoria (NIST, s.d.). A implementação adequada de uma arquitetura de segurança pode garantir um sistema robusto e resistente a possíveis ataques.

III. Desenvolvimento do projeto

O desenvolvimento do projeto iniciou com reflexões e um brainstorm de ideias dos integrantes do grupo para pensarmos em uma solução para facilitar os processos que englobam o controle de acesso à faculdade.

Após várias ideias de soluções e pensando nas experiências obtidas por cada um dos integrantes que compõem o projeto, chegamos a uma solução que agilizaria muito esse processo. Atualmente, o controle dos veículos que acessam a faculdade é feito visualmente pelos guardas que ficam na entrada da faculdade. Inicialmente é feito um cadastro prévio dos veículos autorizados a entrarem na faculdade e o responsável recebe um adesivo para inserir no(s) veículo(s) cadastrados, facilitando a identificação do veículo. O controle se dá em analisar se aquele carro e motorista comumente é visto acessando a faculdade ou se o veículo possui selo da Unisal, recebido após cadastro do veículo. Durante o dia, há uma cancela de controle manual que é mantida baixa e se existir algum acesso, o guarita questiona o motorista e se necessário, pode coletar algumas informações para cadastro prévio, por volta das 18h30 os guaritas realizam a abertura da cancela, devido ao alto fluxo de veículos que acessam a faculdade.

Pensando nas tecnologias e a forma em que é feito o controle de acesso dos veículos, há uma outra maneira mais ágil, eficiente e seguro de realizar esse controle de acesso, que é utilizando técnicas de inteligência artificial para localizar a placa dos veículos e extrair os dígitos da placa, possibilitando construir uma regra de negócio avaliando se aquela placa já foi cadastrada previamente em um banco de dados da instituição. Dessa forma, agilizamos o processo de identificação de um veículo não cadastrado e que pode ser caracterizado como suspeito. Para facilitar a visualização dessas informações, apresentaremos uma página web para os guaritas terem informações úteis dos veículos que acessam a instituição.

Dashboard Web

Para aprimorar a experiência dos usuários do sistema, desenvolvemos uma Dashboard minimalista e de alta funcionalidade. Essa solução foi projetada para proporcionar uma gestão e análise eficiente dos acessos diários, semanais e mensais realizados na faculdade. Com um design limpo e uma interface intuitiva, a DashBoard oferece informações claras e concisas sobre as atividades registradas. O foco principal está na apresentação dos dados de maneira organizada e de fácil compreensão, evitando elementos visuais excessivos. A interface foi cuidadosamente planejada para priorizar a praticidade e a usabilidade, permitindo uma navegação rápida e simplificada. Dessa forma, os usuários podem obter insights significativos e tomar decisões informadas com facilidade, sem distrações visuais desnecessárias. Nossa abordagem minimalista busca otimizar a experiência do usuário, garantindo eficiência e clareza na gestão e análise dos acessos na faculdade.

Ilustração 1 – Ilustração da Dashboard

Fonte: Acervo dos autores

Desenvolvimento

Após levantamento dos desafios técnicos, pensamos nas tecnologias a serem utilizadas para o processo, focando em tecnologias que trariam agilidade para etapa inicial do desenvolvimento e que trouxessem bons resultados e, por experiência com as linguagens, escolhemos Python para trabalhar com identificação da placa e regra de negócio, MySQL como camada de armazenamento dos dados e combo HTML, CSS e JavaScript para apresentação das informações através da página web.

Aplicação Back-end

Python é uma linguagem de programação interpretada, de alto nível e de propósito geral. É conhecida por sua simplicidade, ampla aplicabilidade e abrangência de bibliotecas. Utilizado no projeto para realizar detecção da placa e extração dos caracteres e toda regra de negócio.

Ilustração 2 – Logo Python

Fonte: https://www.python.org/

Para o início do projeto, procuramos seguir as boas práticas da comunidade quanto à estruturação de pastas, arquivos, dependências e também utilizando a documentação do Python. Foram incluídas as bibliotecas necessárias para o desenvolvimento. Após a configuração inicial do projeto, foram criadas as regras para funcionamento.

As regras se dividem em três partes, processamento do vídeo, reconhecimento da placa e controle de acesso, abaixo abordaremos cada um deles sucintamente.

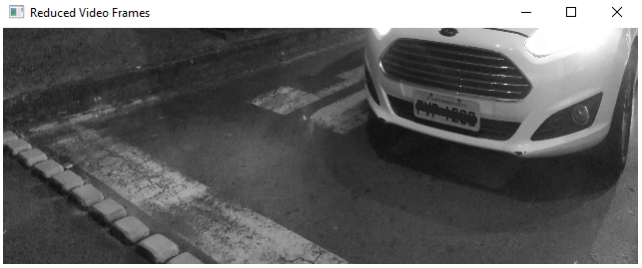
Processamento do vídeo

Realizamos o desenvolvimento nos baseando em um vídeo gravado, porém, tentando nos aproximar ao máximo da realidade simulando uma câmera fixada na guarita. A primeira etapa consiste em um desafio de tratar o vídeo, que consiste numa sequência de imagens. Implementamos um algoritmo responsável por:

1. Reduzir a quantidade de frames do vídeo, utilizando um fator de redução em 100x, isso é, de 2.100 frames (aproximadamente) reduzimos para 6 frames (aproximadamente)
2. Redimensionar os frames, reduzindo a quantidade de pixels, resultado em menos dados para processar
3. Conversão dos frames para escala de cinza, que também contribui em um menor processamento de dados
4. Delimitação do frame, o que auxilia no descarte de pixels fora da área desejada.

As etapas foram estrategicamente pensadas para seguir a ordem acima. Outro fator estratégico para garantirmos melhor desempenho, foi trabalharmos com armazenamento do frame em memória, trazendo acesso mais rápido aos dados e uma redução de operações de leitura/gravação.

Ilustração 3 – Demonstração de um frame otimizado



Fonte: Acervo dos autores

Reconhecimento da placa

Tendo uma quantidade de frames reduzida e otimizada, partimos para os demais desafios. Para reconhecimento da placa, foi utilizado a biblioteca Vision AI do Google, biblioteca poderosa e referência em detecção de objetos e extração de texto.

Ilustração 4 – Logo Google Cloud Vision API

Fonte: google.com

O funcionamento para reconhecimento da placa consiste nas seguintes etapas:

1. Detecção de objetos: fornecemos o frame e a biblioteca é responsável por identificar os objetos da imagem, em seguida, filtramos apenas placas e armazenamos temporariamente as coordenadas de onde a placa se encontra no frame.
2. Delimitação da placa: com as coordenadas armazenadas anteriormente, realizamos um tratamento no frame para recortar a imagem resultando em uma imagem com apenas a placa identificada. O que adiante, nos auxilia na extração de texto, tanto no âmbito de processamento, quanto de textos paralelos
3. Extração do texto: com a delimitação da placa, fornecemos novamente o frame para biblioteca do do google extrair os textos identificados, porém realizamos algumas validações para constatar se o texto se origina de uma placa, no caso de textos paralelos.

Dessa maneira, resta invocar o controle de acesso para realizar as devidas regras de negócio a partir da placa identificada.

Ilustração 5 – Detecção do objeto (placa)

Fonte: Acervo dos autores

Ilustração 6 – Delimitação da placa

Fonte: Acervo dos autores

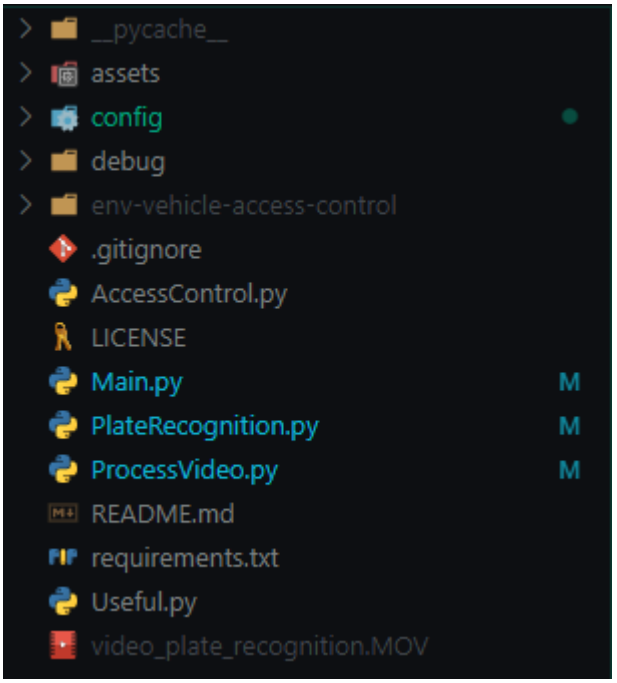
Controle de Acesso

Desenvolvemos um algoritmo que realiza a interação com um banco de dados MySQL para verificar e registrar informações relacionadas a veículos em um sistema de controle de acesso. Esse algoritmo foi escrito para assumir o papel de orquestrador do nosso projeto, onde ele faz o controle dos veículos que são autorizados a acessar (alunos, funcionários, etc que já estão cadastrados em uma base de dados), não autorizados (visitantes, Uber, taxi, etc que ainda não estão cadastrados na base de dados mas que pode ser cadastrado futuramente) e não identificados (onde o algoritmo da identificação de placa não conseguiu capturar os caracteres da placa e isso gera um alerta para a portaria ficar atenta com o acesso suspeito). Além disso, verificamos se o veículo já acessou anteriormente ao registro de acesso, para que seja possível validar sua saída e identificar a nova entrada ao Campus.

Os detalhes de implementação podem ser observados abaixo onde expomos os códigos que realizam as regras citadas anteriormente.

As configurações de pasta e códigos ficaram da seguinte forma:

Ilustração 7 – Estrutura de pastas do projeto



Fonte: Acervo dos autores

Ilustração 8 – Arquivo main

```
from distutils.log import debug
import PlateRecognition, ProcessVideo, Useful

# Variaveis globais para controle
debug = 1
video_path = 'video_plate_recognition.MOV'

def main():
    if debug:
        Useful.delete_directory_files('debug/reduced_video_frames/')
        Useful.delete_directory_files('debug/cut_plate/')
        Useful.delete_directory_files('debug/identified_plate/')

    # Processo video
    ProcessVideo.main()

    # Reconhece objetos e extrai texto da placa
    # plateRecognition = PlateRecognition.main()

    # Valida a placa
    # accessControl = AccessControl.main()


if __name__ == "__main__":
    main()
```

Fonte: Acervo dos autores

Ilustração 9 – Processamento do Vídeo

```python
import datetime
import cv2
import os
import PlateRecognition
import Main

# Variaveis globais para controle
# Fator de redução da taxa de quadros (exemplo: reduzindo pela metade)
fator_reducao = 100
# Define o tamanho desejado para redução do video
largura = 640
altura = 480

def main():
    """
        Ponto de entrada do arquivo, garante execução principal do programa

        Args:
            empty

        Returns:
            array: array com frames.
    """

    try:
        decreaseFramesArray()
    except Exception as e:
        print(e)

def decreaseFramesArray():
    """
        Faz uma série de tratamentos no video para melhorar a performance:
        1. Reduz quantidade de frames
        2. Redimensiona os frames
        3. Converte para escala de cinza
        4. Delimita área especifica (corte no frame)

        Args:
            empty

        Returns:
            empty
    """

    try:
        # Abre o video
        cap = cv2.VideoCapture(Main.video_path, cv2.CAP_FFMPEG)

        # Verifica se o video foi aberto com sucesso
        if not cap.isOpened():
            raise Exception("Não foi possivel abrir o video")

        # Lê e processa os frames reduzidos
        count = 0
        while cap.isOpened():
            ret, frame = cap.read()

            if not ret:
                break

            # Processa o frame atual apenas a cada fator de redução
            if count % fator_reducao == 0:
                # Redimensiona o frame
                frame = cv2.resize(frame, (largura, altura))

                # Converte para escala de cinza
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

                # Focar somente em uma parte do video
                frame = frame[altura//2:,:]

                if Main.debug:
                    file_name = datetime.datetime.now().strftime("reduced_video_frames_%H-%M-%S_%f")
[:-3] + ".jpg"
                    # Salve o quadro como uma imagem na pasta
                    cv2.imwrite(os.path.join('debug/reduced_video_frames/', f'{file_name}'), frame)

                    # Acompanhar frames
                    cv2.imshow('Reduced Video Frames', frame)

                # Reconhece objetos e extrai texto da placa
                PlateRecognition.main(cv2.imencode('.jpg', frame)[1].tobytes())

                # Espera por uma tecla pressionada para avançar para o próximo frame
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break

            count += 1

        # Libera os recursos
        cap.release()

        if Main.debug:
            cv2.destroyAllWindows()

    except Exception as e:
        print("Ocorreu um erro:", str(e))
```

Fonte: Acervo dos autores

Ilustração 10 – Reconhecimento da Placa

```python
import os
import random
import cv2
import numpy as np
from google.cloud import vision
import Main
import datetime

score_detection_object = 0.75

def main(frame_bytes):
    # Variável de ambiente para o arquivo de autenticação do Google Vision
    os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = r'config/google_vision_auth.json'

    # Cria a instância do cliente da API do Google Cloud Vision
    global image_annotator
    image_annotator = vision.ImageAnnotatorClient()

    try:
        plate_vertices = findObjects(frame_bytes)

        # Verifico se houve algum objeto identificado
        if plate_vertices:
            cut_plate_frame_bytes = cutPlateFrame(plate_vertices, frame_bytes)

            extractTextPlate(cut_plate_frame_bytes)
    except Exception as e:
        print(e)

    #return plate

def findObjects(frame_bytes):
    """
        Localiza os objetos de uma imagem utilizando API Google Vision.

        Args:
            frame_bytes: imagem (frame) em formato binario.

        Returns:
            plate_vertices: vertices de onde se localiza a placa.
    """

    image = vision.Image(content=frame_bytes)

    objects = image_annotator.object_localization(
        image=image).localized_object_annotations

    # Armazena os vertices de onde se encontra a placa
    plate_vertices = []

    # Entre os objetos identificados, procura a placa
    for object in objects:
        # Extrair do objecets a confinaça pra salvar no banco
        if object.name == 'License plate' and object.score >= score_detection_object:
            if Main.debug:
                file_name = datetime.datetime.now().strftime("identified_plate_%H-%M-%S_%f")[:-3] +
".jpg"
                # Salvando frame que contem um objeto de placa, pra usar de debug
                cv2.imwrite(os.path.join('debug/identified_plate/', f'{file_name}'),
cv2.imdecode(np.frombuffer(frame_bytes, np.uint8), cv2.IMREAD_COLOR))

            # Tosere em um array as posições X e Y dos pontos que formam a placa
            for vertex in object.bounding_poly.normalized_vertices:
                plate_vertices.append([vertex.x, vertex.y])

    return plate_vertices


def cutPlateFrame(plate_vertices, frame_bytes):
    """
        A partir de coordenadas X e Y monta imagem somente da região desejada

        Args:
            frame_bytes: imagem (frame) em formato binario.
            plate_vertices: vertices de onde se localiza a placa.

        Returns:
            cut_frame_byes: frame cortado em bytes
    """
    # Realiza decode do frame de bytes para imagem
    frame_image = cv2.imdecode(np.frombuffer(frame_bytes, np.uint8), cv2.IMREAD_COLOR)

    vertices = np.array(plate_vertices)

    # Obtenha a altura e a largura da imagem
    altura, largura, _ = frame_image.shape

    # Converta as coordenadas normalizadas para coordenadas de pixel
    vertices_pixel = vertices * np.array([largura, altura])

    # Converta as coordenadas de pixel para inteiro
    vertices_pixel_int = vertices_pixel.astype(np.int32)

    # Obtenha os retângulos delimitadores (x, y, largura, altura) da ROI
    x, y, w, h = cv2.boundingRect(vertices_pixel_int)

    # Recorte a região de interesse da imagem com base nos retângulos delimitadores
    cut_frame = cv2.getRectSubPix(frame_image, (w, h), (x + w/2, y + h/2))

    if Main.debug:
        file_name = datetime.datetime.now().strftime("cut_plate_%H-%M-%S_%f")[:-3] + ".jpg"
        # Salvando imagem da placa recortada
        cv2.imwrite(os.path.join('debug/cut_plate/', f'{file_name}'), cut_frame)

    # Incode da imagem em bytes
    cut_plate_frame_bytes = cv2.imencode('.jpg', cut_frame)[1].tobytes()

    return cut_plate_frame_bytes


def extractTextPlate(cut_plate_frame_bytes):
    image = vision.Image(content=cut_plate_frame_bytes)

    texts = image_annotator.text_detection(
    image=image).text_annotations

    plate = ""
    for text in texts:
        if Main.debug:
            print("text:", text.description)

        # Removendo caracteres da placa
        clean_plate = text.description.replace(" ", "").replace("-", "")

        # Pra que é isso?
        clean_plate = clean_plate.split('\n')

        for plate_txt in clean_plate:
            if len(plate_txt) == 7:
                plate = plate_txt
            else:
                plate = "Placa não identificada"

        if len(clean_plate) == 7:
            plate = clean_plate
        else:
            plate = "Placa não identificada"

    #print("Placa: ", plate)
    return plate
```

Fonte: Acervo dos autores

Ilustração 11 – Controle de Acesso

```
#pip install mysql-connector-python

import mysql.connector
import logging
import datetime
from datetime import date

# Configuração do logger - melhorar a visibilidade dos prints tanto de retorno quanto de erro presentes no código
logging.basicConfig(filename='app.log', level=logging.INFO)  # Define o nível de logging para INFO

# Criação de um manipulador de logging para exibir mensagens no console
console_handler = logging.StreamHandler()
console_handler.setLevel(logging.INFO)

# Adiciona o manipulador de console ao logger raiz
logger = logging.getLogger()
logger.addHandler(console_handler)


conn = None  # Declaração da variável de conexão global

class VeiculoStatus:
    CADASTRADO = 1
    NAO_CADASTRADO = 2
    NAO_IDENTIFICADO = 3

def conectar_banco():
    global conn  # Indica que estamos utilizando a variável de conexão global
    try:
        conn = mysql.connector.connect(
            host="127.0.0.1:3306",
            user="root",
            password="root",
            database="ProjetoIntegrador"
        )
        if conn.is_connected():
            logging.info("Conexão estabelecida com o banco de dados.")
            return conn

    except mysql.connector.Error as error:
        logging.error("Erro ao conectar-se ao banco de dados: %s", error)

def desconectar_banco():
    global conn  # Indica que estamos utilizando a variável de conexão global
    try:
        conn.close()
        logging.info("Desconectado do banco de dados.")

    except mysql.connector.Error as error:
        logging.error("Erro ao desconectar-se do banco de dados: %s", error)

def verificar_veiculo(placa):
    conectar_banco()  # Chamada para estabelecer a conexão
    cursor = conn.cursor()

    consulta_verificar = "SELECT COUNT(*) FROM veiculos WHERE placa = %s"
    dados_verificar = (placa,)

    cursor.execute(consulta_verificar, dados_verificar)
    resultado_verificar = cursor.fetchone()

    if resultado_verificar[0] == 0:
        logging.info("Veículo não existe no banco de dados!")
        return VeiculoStatus.NAO_CADASTRADO
    else:
        logging.info("Veículo já existe no banco de dados!")
        return VeiculoStatus.CADASTRADO

    cursor.close()

def inserir_veiculo(placa):
    cursor = conn.cursor()

    consulta_inserir = "INSERT INTO acessos (placa_veiculo) VALUES (%s)"
    dados_inserir = (placa,)

    cursor.execute(consulta_inserir, dados_inserir)
    conn.commit()
    logging.info("Veículo inserido com sucesso!")

    cursor.close()


def verificar_entrada_existente(placa):
    hoje = date.today().strftime("%D-%M-%Y")

    cursor = conn.cursor()

    consulta_verificar = "SELECT COUNT(*) FROM acessos WHERE placa_veiculo = %s AND data_entrada = %s"
    dados_verificar = (placa, hoje)

    cursor.execute(consulta_verificar, dados_verificar)
    resultado_verificar = cursor.fetchone()

    if resultado_verificar[0] > 0:
        logging.warning("Entrada já registrada para o veículo no dia de hoje!")
        return True
    else:
        return False

    cursor.close()
    desconectar_banco()  # Chamada para desconectar do banco de dados

def main():
    placa = "FAI8695"

    conectar_banco()
    status = verificar_veiculo(placa)

    if status == VeiculoStatus.CADASTRADO:
        logging.info("Veículo cadastrado.")

        if not verificar_entrada_existente(placa):
            inserir_veiculo(placa)
    elif status == VeiculoStatus.NAO_CADASTRADO:
        logging.warning("Veículo não cadastrado.")

        if not verificar_entrada_existente(placa):
            inserir_veiculo(placa)
    elif status == VeiculoStatus.NAO_IDENTIFICADO:
        logging.warning("Placa não identificada.")

    desconectar_banco()

# Execução do programa principal
if __name__ == "__main__":
    main()

# Exemplo de uso do logger
logger.info("Esta é uma mensagem de informação.")
logger.warning("Esta é uma mensagem de aviso.")
logger.error("Esta é uma mensagem de erro.")
```

Fonte: Acervo dos autores

IV. Resultados e Discussões

Discutindo sobre os resultados gerados durante o projeto, percebemos que esse sistema irá facilitar para a equipe de segurança da portaria no controle dos acessos, cadastros e identificação dos veículos de forma mais ágil e eficaz, onde uma tela de dashboard irá centralizar todas as informações primordiais, para que seja feito uma validação completa e estará disponível para o porteiro utilizar como quiser. Com o crescimento do projeto, visamos melhorar ainda mais a eficácia do modelo para deixar mais assertivo e seguro.

Missão

Desenvolver um sistema de segurança para portaria de faculdade que integra hardware, software e inteligência artificial para garantir o controle efetivo de veículos e aumentar a segurança dos estudantes, professores e funcionários.

Visão

Tornar-se referência em sistemas de segurança para portaria de faculdade, por meio da constante busca por inovação e aprimoramento, e contribuir para a melhoria da segurança em instituições de ensino em todo o país.

Valores

Inovação: buscar constantemente soluções criativas e tecnologicamente avançadas para aprimorar o sistema de segurança para portaria de faculdade;

Efetividade: desenvolver um sistema de segurança que garanta um controle efetivo de veículos e aumente a segurança dos estudantes, professores e funcionários;

Transparência: agir com ética e transparência em todas as etapas do desenvolvimento do sistema de segurança para portaria de faculdade.

Sustentabilidade

O sistema de segurança para portaria de faculdade que estamos desenvolvendo tem diversas características sustentáveis. Em primeiro lugar, ao aumentar a segurança dos estudantes, professores e funcionários, estamos contribuindo para um ambiente mais seguro e saudável na instituição de ensino. Isso é especialmente importante em áreas urbanas, onde a segurança muitas vezes é um problema.

Além disso, o nosso sistema de segurança para portaria de faculdade usa tecnologia de ponta, como hardware, software e inteligência artificial. Essas tecnologias são mais eficientes do que os métodos tradicionais de controle de acesso, como a utilização de segurança ou catracas. Dessa forma, estamos contribuindo para um uso mais eficiente dos recursos e reduzindo o impacto ambiental do nosso projeto.

Ação Social

Nossa ação social consiste em promover maior segurança e facilidade no controle dos veículos que acessam a instituição de ensino e principalmente, identificar previamente veículos suspeitos, outro fator é o engajamento com a comunidade e contribuir para o desenvolvimento educacional, incentivando o avanço tecnológico em nossa região. Essa ação social complementa o projeto de controle de acesso de veículos em nossa faculdade, expandindo seus benefícios além do âmbito acadêmico e impactando positivamente a comunidade ao nosso redor.

Conclusão

V. Conclusões

Esperamos que, ao implementar o nosso sistema, as instituições de ensino possam melhorar significativamente a segurança de seus alunos, professores e funcionários, bem como otimizar o controle de acesso de veículos e pedestres, garantindo um ambiente mais seguro e saudável.

Além disso, esperamos que a nossa solução possa ser adaptada e escalada para diferentes tipos e tamanhos de instituições de ensino, atendendo às necessidades específicas de cada uma delas.

Por fim, acreditamos que a sustentabilidade e a responsabilidade social são fatores essenciais para o sucesso do nosso projeto. Esperamos que o nosso sistema de segurança para portaria de faculdade possa contribuir para um mundo mais sustentável e justo, promovendo a inclusão e o desenvolvimento social por meio da educação.

Referências

A SEGURANÇA PRIVADA NÃO CONTROLADA. Disponível em: <https://forumseguranca.org.br/wp-content/uploads/2022/07/18-anuario-2022-a-seguranca-privada-nao-controlada.pdf>
Acesso em: 20 mar. 2023.

INTELIGÊNCIA ARTIFICIAL NO MERCADO DE SEGURANÇA. Disponível em: <https://www.marketsandmarkets.com/Market-Reports/artificial-intelligence-security-market-220634996.html>
Acesso em: 20 mar. 2023.

MENDES, A., OLIVEIRA, F., & PEREIRA, J. (2019). SEGURANÇA EM PORTARIAS: DESAFIOS E SOLUÇÕES. Disponível em: Revista Brasileira de Segurança Pública, 14(2), 75-90.
Acesso em: 20 mar. 2023.

SILVA, R. (2020). DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE DE ACESSO COM RECONHECIMENTO DE PLACAS DE VEÍCULOS. Disponível em: Revista de Tecnologia e Sociedade, 12(1), 45-58.
Acesso em: 20 mar. 2023.

QUATRO CAMADAS DA ARQUITETURA DE SEGURANÇA Disponível em: <https://www.nist.gov/>
Acesso em: 20 mar. 2023.

SISTEMA AUTOMÁTICO DE DETECÇÃO E LEITURA DE PLACAS VEICULARES. Disponível em: <https://repositorio.ufsc.br/handle/123456789/224664>
Acesso em: 20 mar. 2023.

SISTEMA DE RECONHECIMENTO AUTOMÁTICO DE PLACAS VEICULARES. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/2971935/mod_folder/content/0/DM%20-%20Reconhecimento%20Autom%C3%A1tico%20de%20Placas.pdf?forcedownload=1>.
Acesso em: 20 mar. 2023.

GOOGLE VISION. Disponível em: <https://cloud.google.com/vision?hl=pt-br>.
Acesso em: 20 mar. 2023.

ALEXANDRE, H. (OCR OPENCV) Realizando Detecção de Placas através de Contornos (parte 1/2). 2018. (33m). Disponível em: <https://www.youtube.com/watch?v=pDA4mncyJ8Q>.
Acesso em: 08 mar. 2023.

ALEXANDRE, H. (OCR OPENCV) com OpenCV e Tesseract utilizando Python (parte 2/2). 2021. (30m). Disponível em: <https://www.youtube.com/watch?v=3cwkEOUscX>.
Acesso em: 09 mar. 2023.

ALEXANDRE, H. (GOOGLE VISION) Realizando OCR utilizando API de Visão Computacional do GOOGLE. 2021. (30m). Disponível em: <https://www.youtube.com/watch?v=h3H8Li08Idg>.
Acesso em: 10 mar. 2023.

BHATT, B. Extract Text from image OCR using Google Vision API in Python. 2020. (5m19s). Disponível em: <https://www.youtube.com/watch?v=tOVjjo8VJTs>.
Acesso em: 10 mar. 2023.

PINTO, I. Usando o Google Cloud Vision como OCR - com Ítalo da Silva | Orange Talks #21. 2021. (47m31s). Disponível em: <https://www.youtube.com/watch?v=5mLXf-Kor_Y>.
Acesso em: 01 abr. 2023.

CTRL, D. Google Cloud Vision API | Análise de Imagens. 2021. (13m08s). Disponível em: <https://www.youtube.com/watch?v=LnB-rkTR-X4>.
Acesso em: 08 abr. 2023.

OpenCV. [online]. Disponível em: <https://opencv.org/>. Acesso em: 10 abr. 2023.

Python. [online]. Disponível em: <https://www.python.org/>. Acesso em: 15 abr. 2023.