Instituto Politécnico
de Castelo Branco
Escola Superior
de Tecnologia

CrosSafe - Protótipo de Solução para Detetar e Sinalizar Defeitos em Pavimentos Rodoviários

Projeto II

Gonçalo José Mendes Rosa

João Manuel dos Santos Afonso

Orientadores

João Manuel Leitão Pires Caldeira

Vasco Nuno da Gama de Jesus Soares

Trabalho de Projeto apresentado à Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco, realizado sob a orientação científica do Doutor João Manuel Leitão Pires Caldeira e coorientação do Doutor Vasco Nuno da Gama de Jesus Soares, do Instituto Politécnico de Castelo Branco.

Junho de 2024

Composição do júri

Presidente do júri

Doutora Mónica Isabel Teixeira da Costa

Professora Adjunta do Instituto Politécnico de Castelo Branco

Vogais

Doutor João Manuel Leitão Pires Caldeira

Professor Adjunto do Instituto Politécnico de Castelo Branco

Doutor Osvaldo Arede dos Santos

Professor Adjunto do Instituto Politécnico de Castelo Branco

Resumo

As passadeiras pedonais desempenham um papel fundamental na segurança rodoviária, mas com o tempo, muitas sofrem desgastes que dificultam a sua visualização. Este projeto apresenta uma solução, baseada na utilização de técnicas de visão computacional, para identificação e classificação do nível de desgaste de passadeiras pedonais. Esta solução pretende automatizar a recolhendo dos dados necessários para avaliação de intervenções de melhoria nestes equipamentos por parte das entidades competentes pela sua manutenção. Ao longo deste documento serão descritas as técnicas de visão computacional consideradas para atingir o objetivo proposto e os passos envolvidos na construção de um protótipo demonstrativo. O protótipo desenvolvido foi pensado para montagem num veículo, que se desloca pelas vias de trânsito, recolhendo os dados das passadeiras encontradas. Os dados são depois enviados para uma aplicação *Web* que os disponibiliza para análise. A validação deste protótipo foi realizada através de testes em ambiente real.

Palavras chave

Passadeiras Pedonais, Cidades Inteligentes, Visão Computacional, Redes Neurais Convolucionais, Avaliação de Desempenho.

Abstract

The functionality of pedestrian crossings is of paramount importance in road safety. However, over time, these crossings suffer from wear and tear that makes them difficult to visualize. This project presents a solution based on the use of computer vision techniques for identifying and classifying the level of wear on pedestrian crossings. The aim of this solution is to automate the collection of the data needed to assess improvements to this equipment by the organizations responsible for its maintenance. This project describes the computer vision techniques used to achieve the proposed objective and the steps involved in building a demonstration prototype. The prototype was designed to be mounted on a vehicle that moves along traffic routes, collecting data on the pedestrian crossings it encounters. The data is then sent to a web application that makes it available for analysis. This prototype was validated through real-life tests.

Keywords

Pedestrian Crossings, Smart Cities, Computer Vision, Convolutional Neural Networks, Performance Evaluation.

Índice

Índice de figuras

Lista de tabelas

Lista de abreviaturas, siglas e acrónimos

AP: Average Precision

API: Application Programming Interface

CNN: Convolutional Neural Network

CPU: Central Processing Unit

GPU: Graphics Processing Unit

GPS: Global Positioning System

HTTP: Hypertext Transfer Protocol

IdC: Internet das Coisas

JSON: JavaScript Object Notation

ORM: Object Relational Mapper

RAM: Random Access Memory

SQL: Structured Query Language

UI: User Interface

UML: Unified Modeling Language

USB: Universal Serial Bus

YOLO: You Only Look Once

1. Introdução

Este primeiro capítulo descreve o contexto do trabalho a desenvolver introduzindo a temática onde o mesmo se insere e o seu âmbito. É identificado o problema a explorar e definida a estratégia a adotar para o seu desenvolvimento, nomeadamente a planificação proposta e o objetivo a atingir. Finalmente é apresentada a organização do documento.

1.1 Âmbito

As passadeiras pedonais desempenham um papel crucial na segurança e mobilidade urbana, fornecendo um ambiente seguro para os pedestres atravessarem as vias públicas e contribuindo para a redução de acidentes de trânsito. Em Portugal, apenas no ano de 2023, entre janeiro e novembro, foram registados 33721 acidentes rodoviários, dos quais 13.01% resultaram em atropelamentos [1]. Ainda mais alarmante é o fato de que 11.1% das vítimas desses atropelamentos foram vítimas mortais. Estes números evidenciam a vulnerabilidade dos pedestres nas vias urbanas e realçam a necessidade urgente de medidas para garantir a sua segurança.

1.2 Definição do Problema e Objetivos

As passadeiras pedonais, assim como outras infraestruturas urbanas, estão sujeitas a vários fatores que podem levar à sua degradação ao longo do tempo. Desde a exposição contínua aos elementos naturais até ao desgaste causado pelo tráfego constante, essas passagens enfrentam desafios que comprometem a sua funcionalidade e segurança. Diante deste cenário, torna-se crucial explorar soluções tecnológicas que auxiliem na deteção e classificação do estado de degradação das passadeiras pedonais. A utilização de técnicas de visão computacional possibilita o desenvolvimento de sistemas capazes de identificar, não apenas a presença das passadeiras, mas também de avaliar seu nível de conservação. Estes sistemas permitem a recolha e fornecimento de informações cruciais às autoridades responsáveis pela manutenção destes equipamentos. A análise contínua do estado dessas passagens, por meio desses sistemas, pode desempenhar um papel crucial na melhoria da infraestrutura urbana, contribuindo para a segurança e qualidade de vida nas cidades.

Este projeto tem por principal objetivo o desenvolvimento de um protótipo capaz de detetar e classificar o estado de degradação de passadeiras pedonais. Este protótipo providência dados para consulta numa plataforma *Web*, concebida para as autoridades competentes, onde poderá ser possível visualizar as várias passadeiras pedonais juntamente com o seu estado de degradação.

Para atingir este objetivo principal, identificam-se os seguintes objetivos intermédios:

- Análise do estado da arte e levantamento tecnológico;
- Análise de requisitos;

- Avaliação de desempenho;
- Proposta e implementação de protótipo demonstrador;
- Demonstração e testes funcionais.
- Redação do relatório.

1.3 Cronograma

Para atingir os objetivos acima identificados, prevê-se o plano de trabalhos e cronograma apresentado na Figura 1, constituído pelas tarefas abaixo descritas:

- Tarefa 1 – Estudo do trabalho relacionado (Mês 1): Análise do estado da arte e levantamento tecnológico.
- Tarefa 2 – Análise de requisitos (Meses 2 e 3): Definição dos requisitos que levam à correta solução do problema identificado.
- Tarefa 3 – Proposta e implementação de protótipo demonstrador (Meses 3 e 4): Desenvolvimento do protótipo demonstrador.
- Tarefa 4 – Avaliação de Desempenho (Mês 4): Validação do desempenho do protótipo desenvolvido.
- Tarefa 5 – Demonstração e testes funcionais (Meses 5 e 6): Testes, ajustes e validação do protótipo.
- Tarefa 6 – Redação do relatório (Meses 5 e 6): Elaboração da escrita do relatório.

	Mês 1	Mês 2	Mês 3	Mês 4	Mês 5	Mês 6
Tarefa 1						
Tarefa 2						
Tarefa 3						
Tarefa 4						
Tarefa 5						
Tarefa 6						

Figura 1 - Tarefas realizadas ao longo dos meses do projeto.

1.4 Contribuições

Do desenvolvimento deste trabalho resultaram a seguintes contribuições: (1) análise de técnicas de visão computacional; (2) construção de dois *datasets* contendo passadeiras pedonais, um com 1182 imagens e uma classe, outro com 1440 *labels* de degradação de passadeiras pedonais; (3) melhoria do modelo *YOLOv4-tiny* apresentado em Projeto I (4) descrição do processo de desenvolvimento de um protótipo funcional.

1.5 Organização do Documento

Este documento encontra-se estruturado em 5 capítulos. O primeiro capítulo tem como propósito explicar o trabalho ao leitor bem como contextualizar o mesmo. O

segundo capítulo introduz as técnicas de visão computacional que servem de base a este trabalho. O terceiro capítulo apresenta a construção de um protótipo funcional. O quarto capítulo demonstra os testes de validação efetuados. Finalmente, o quinto capítulo apresenta as conclusões deste trabalho e pontos em aberto que podem ser abordados no futuro.

2. Classificação do Estado de Degradação de Passadeiras com Técnicas de Visão Computacional

Para atingir o objetivo do trabalho proposto, o primeiro desafio passou por dotar o sistema, a desenvolver, da capacidade de deteção e classificação do estado de degradação de passadeiras pedonais, com base em técnicas de visão computacional.

A visão computacional centra-se na capacidade de máquinas computacionais conseguirem interpretar imagens, vídeos e outros dados visuais. Essas informações são utilizadas para uma variedade de finalidades, incluindo deteção e classificação de objetos. A deteção está relacionada com a identificação da localização do objeto enquanto a classificação está relacionada com a atribuição de uma classe a este objeto [2]. Estes conceitos estão ilustrados na Figura 2.

Figura 2 - Ilustração dos conceitos deteção e classificação.

Geralmente, as câmaras são a principal fonte de dados para esse fim, e é crucial possuir um grande conjunto de dados para garantir uma deteção precisa do que se deseja identificar. Além disso, a implementação de algoritmos é essencial para permitir que o modelo aprenda de forma autônoma, sem intervenção humana. Essa tecnologia tem aplicações em diversos campos, como o *machine learning* [3] e o *deep learning* [4], áreas que instruem os computadores a pensar como o cérebro humano, podendo reconhecer padrões complexos em imagens, textos, sons e outros [5]. Estes padrões são possíveis de identificar através da utilização de *Convolutional Neural Network* (CNN). Estas são um conjunto de camadas interligadas através de nós. Estes nós têm como principal objetivo simular neurónios de um ser humano. Na Figura 3 está ilustrada a arquitetura de uma CNN, onde esta recebe como *input* uma imagem com um objeto, por exemplo, e irá retornar uma ou mais classes, de modo a classificar a categoria deste objeto.

Ao longo desta secção serão apresentadas, discutidas e analisadas as técnicas de visão computacional usadas para atingir o objetivo proposto – detetar e classificar o estado de degradação de passadeiras pedonais. Inicialmente é apresentada a técnica de *Adaptive Threshold* e a sua aplicação para a classificação do estado de degradação de passadeiras. Serão discutidas as limitações desta técnica e as configurações

necessárias para a sua aplicação. Esta abordagem, embora inicialmente com resultados promissores, revelou algumas limitações em determinadas condições. Nesse sentido, a secção seguinte apresenta uma segunda abordagem usando o modelo *YOLOv4-tiny*, com um *dataset* melhorado e alterado, para uma classificação de níveis de desgaste de passadeiras mais eficaz. Por fim, é feita uma análise de desempenho comparativa das duas abordagem onde é descrito o cenário de *benchmark*, bem como as métricas utilizadas na avaliação dos resultados obtidos.

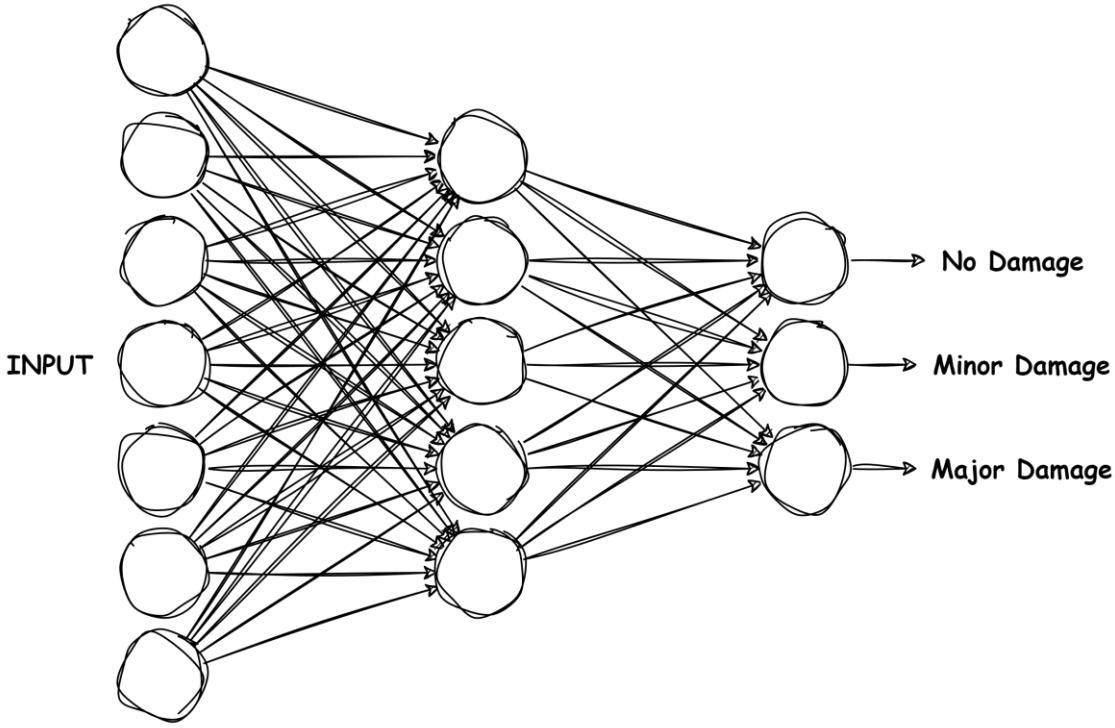


Figura 3 - Ilustração da arquitetura de um CNN.

2.1 Classificação com *Adaptive Threshold*

Adaptive Thresholding [6] é uma técnica empregada no processamento de imagens para distinguir entre o primeiro plano e o plano de fundo de uma imagem. Para alcançar esse fim, seleciona-se um valor de *threshold*, onde todos os pixels com intensidade acima desse *threshold* são considerados como pertencentes a uma classe, representada pela cor branca, enquanto os pixels com intensidade abaixo do *threshold* são categorizados noutra classe, frequentemente usando a cor preta.

Diferentemente da utilização de um *threshold* global, que aplica um único valor de *threshold* a toda a imagem, o *Adaptive Thresholding* calcula valores de *threshold* específicos para diferentes regiões da imagem. Nesse método, o valor de *threshold* para cada pixel é determinado com base nas características da sua vizinhança.

Existem diversos tipos de *thresholding* (ilustrados na Figura 4) utilizados em processamento de imagens, incluindo:

- *Global Thresholding*: Aplica um único valor de *threshold* a toda a imagem.

- *Otsu's Thresholding*: Determina automaticamente o valor ótimo de *threshold* com base no histograma da imagem.
- *Adaptive Mean Thresholding*: Calcula um valor de *threshold* para cada região da imagem com base na média dos valores de intensidade dos pixels naquela região.
- *Adaptive Gaussian Thresholding*: Similar ao *Adaptive Mean Thresholding*, mas calcula o valor de *threshold* usando uma média ponderada baseada na distribuição Gaussiana.

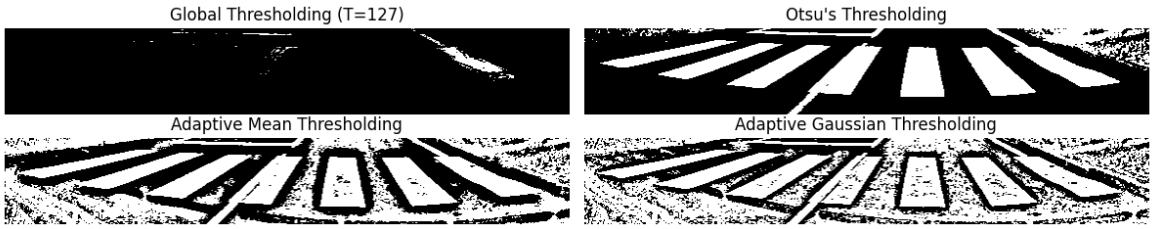


Figura 4 - Comparação dos métodos de *threshold*.

A possibilidade de realizar a combinação de diferentes métodos de *threshold* pode ajudar na deteção de desgaste em passadeiras pedonais. Essa abordagem é especialmente útil em ambientes onde as condições de iluminação são variáveis. Ao utilizar as vantagens de cada técnica, é possível obter resultados mais confiáveis e precisos na identificação de áreas desgastadas das passadeiras pedonais, objetivo crucial deste projeto. Na Figura 5 são apresentados alguns exemplos da aplicação de *Adaptive Thresholding* na identificação do desgaste em passadeiras pedonais. Respetivamente, na Figura 5(a) é possível visualizar o resultado da aplicação deste método a uma passadeira pedonal com algum desgaste, na Figura 5(b) temos o resultado obtido para uma passadeira pedonal sem desgaste e finalmente, a Figura 5(c) apresenta o resultado para uma passadeira pedonal com bastante desgaste. Como se pode verificar, esta técnica apresenta resultados bastante satisfatórios para o objetivo pretendido.

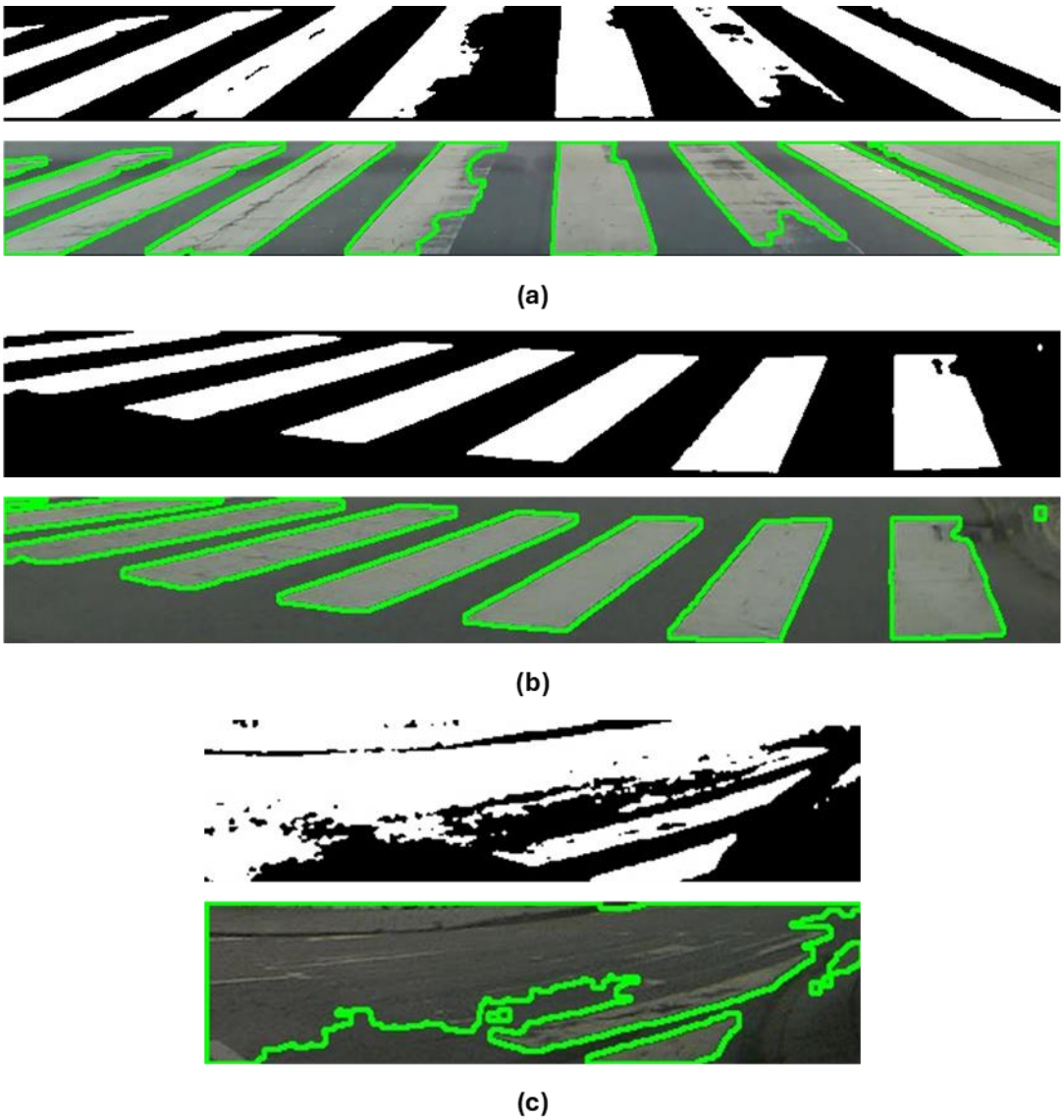


Figura 5 - Utilização de *Adaptive Threshold* para determinar estado de degradação.

2.1.1 Valores de *Threshold*

Conforme mencionado anteriormente, é fundamental estabelecer valores de *threshold* adequados para alcançar os resultados desejados. Para obter esses valores de forma eficiente, foram selecionadas aleatoriamente 15 imagens do *dataset* e incluídas num questionário. O objetivo desta tarefa era pedir a vários participantes que avaliassem o estado de cada uma das passadeiras pedonais de acordo com a sua sensibilidade. Esta etapa foi primordial para tentar ultrapassar uma dificuldade encontrada, na identificação de um padrão que permitisse classificar o estado de uma passadeira. A recolha dos dados recebidos ajudou na definição de um padrão que posteriormente foi aplicado às demais imagens do *dataset*. Na Figura 6 é exemplificada uma das perguntas incluídas no questionário. O questionário foi respondido, até ao momento, por 48 participantes, e alguns dos resultados podem ser observados na Figura 7.

Figura 6 - Exemplo de uma questão colocada no formulário.

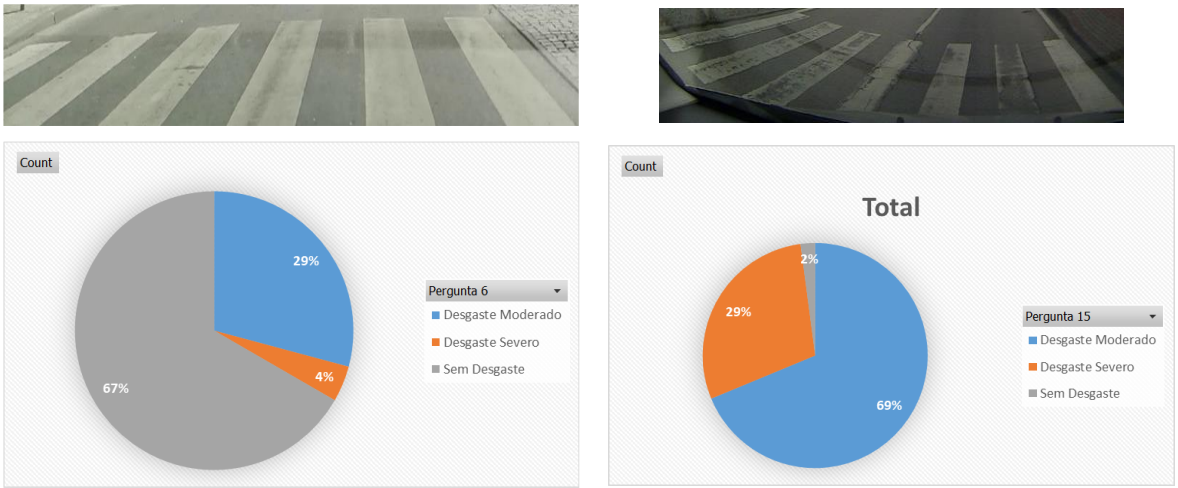


Figura 7 - Exemplo de resultados obtidos no questionário.

Como mencionado anteriormente, o processo de classificação do estado de degradação das passadeiras pedonais envolveu a identificação de padrões em imagens contendo exclusivamente passadeiras. Utilizando métodos de *Adaptive Threshold* [7], foi possível calcular a área branca correspondente às listras da passadeira, essencial para extrair os padrões necessários para determinar o estado de degradação das passadeiras pedonais noutras imagens. Estes padrões estão detalhados na Tabela 1, e podem ser utilizados para determinar o estado de degradação das passadeiras pedonais.

Tabela 1 - Intervalo de desgastes calculado através de *Adaptive Threshold*.

Intervalos	***Estado de Degradação***
<30	Sem desgaste
[30, 50[	Desgaste moderado
>50	Desgaste severo

Ao longo do desenvolvimento deste trabalho, a primeira abordagem seguida, para a deteção e classificação do desgaste de passadeiras pedonais, passou pela utilização de um modelo *YOLOv4-tiny* para identificar passadeiras pedonais. Em seguida, sobre as passadeiras identificadas, foram aplicados métodos de *Adaptive Thresholding* para determinar o estado de degradação dessas passadeiras. Durante a implementação desta abordagem, foram identificados alguns problemas, especialmente quando a passadeira pedonal apresentava um alto nível de desgaste. Esta situação dificultava a identificação das áreas das listras. Para solucionar esse problema, uma possível abordagem seria aumentar a intensidade dos pixels brancos na imagem, tornando-os mais visíveis e facilitando a sua utilização no cálculo da área. Esta abordagem pode ser visualizada na Figura 8.

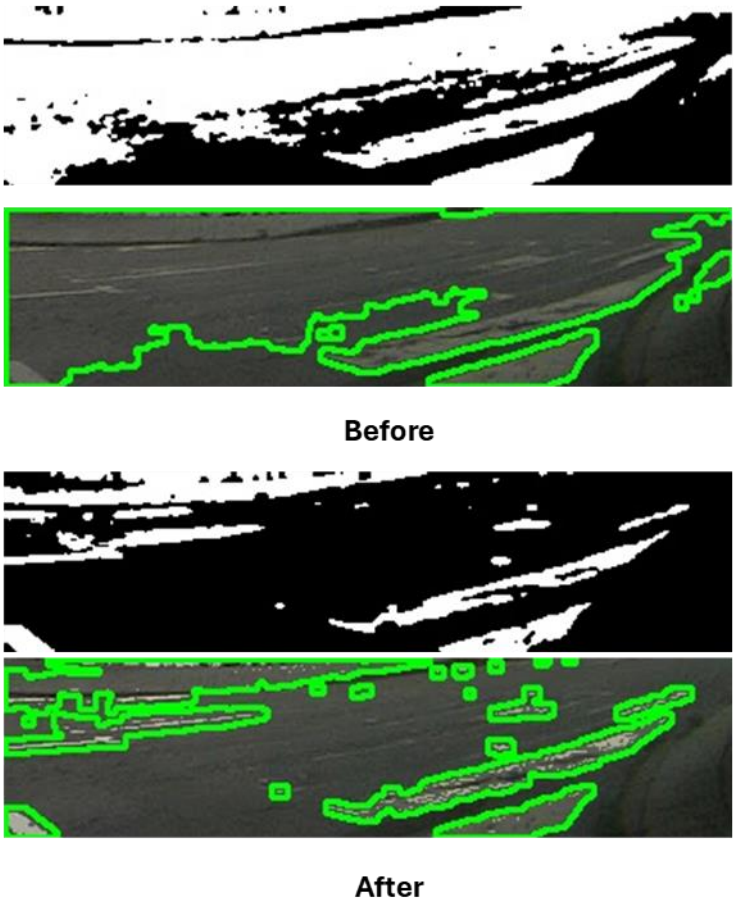


Figura 8 - Exemplo de como o aumento dos pixels pode influenciar os resultados.

Embora esta solução melhorasse ligeiramente o resultado obtido, não o tornava satisfatório. Por outro lado, tal como será demonstrado na secção de avaliação de desempenho (2.3), o efeito de sombras projetadas sobre as passadeiras dificultou a sua identificação e correta classificação. Considerando estas limitações, pela aplicação singular da técnica *Adaptive Threshold* na classificação do estado de degradação das passadeiras, foi adotada uma segunda abordagem, usando o modelo *YOLOv4-tiny*. Na secção seguinte serão descritas as técnicas usadas nesta segunda abordagem, bem como a abordagem em si.

2.2 Classificação com YOLOv4-Tiny

O estudo realizado anteriormente pelos mesmos autores [8], demonstrou que perante os vários modelos CNN testados, aquele que melhor se adequaria ao objetivo a atingir seria o modelo *YOLOv4-tiny*. Sendo assim, foi este o modelo adotado para a deteção de passadeiras, uma vez que, devido à sua arquitetura, apresentou bons resultados em dispositivos com poucos recursos computacionais.

O modelo *YOLO* (*You Only Look Once*) divide a imagem em uma grelha, identificando assim a *bounding box,* níveis de confiança e ainda a classe respetiva para o objeto. Resultando numa combinação de *bounding boxes* de uma determinada classe e com um nível de confiança associado. A versão *tiny* deste modelo utiliza a *CNN CSPDarknet53-tiny* [9] como base, sendo esta composta por três camadas de *convolutional* e três módulos *CSPBlock*. *CSPDarknet53-tiny* é uma versão simplificada da *CSPDarknet53* e utiliza o módulo *CSPBlock* em vez do módulo *ResBlock*. O módulo *CSPBlock* melhora a capacidade de aprendizagem da *CNN* em comparação com o módulo *ResBlock*.

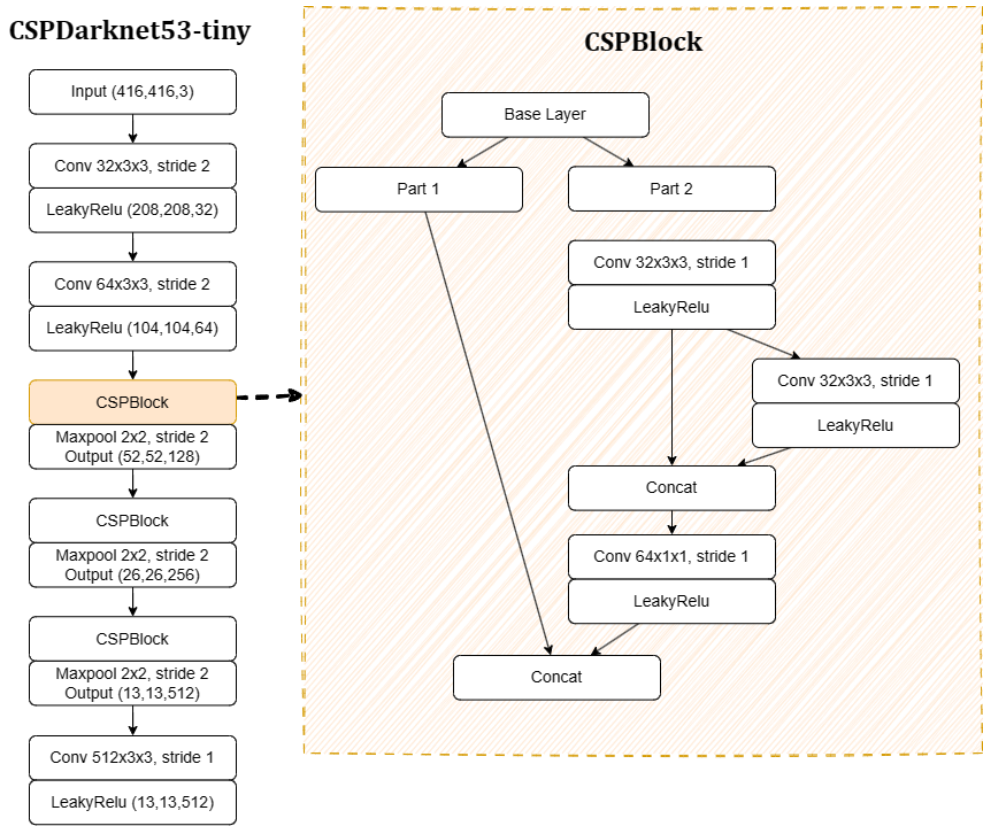


Figura 9 - Ilustração da *CNN CSPDarknet53-tiny*. Adaptado de [9].

A *CNN CSPDarknet53-tiny* inclui três camadas *convolutional* e três módulos *CSPBlock*. As duas primeiras camadas *convolutional,* incluem uma camada *convolutional* e uma camada *LeakyReLU*, com um tamanho de *kernel* de 3 e um *stride* de 2, respetivamente. A camada *convolutional* final tem um tamanho de *kernel* de 3 e um *stride* de 1, como ilustrado em Figura 9. A *CSPDarknet53-tiny* utiliza ainda a função de *LeakyReLU* como função de ativação em vez da função de ativação *Mish*, o que simplifica o processo de cálculo [10]. A Figura 10 ilustra esta arquitetura.

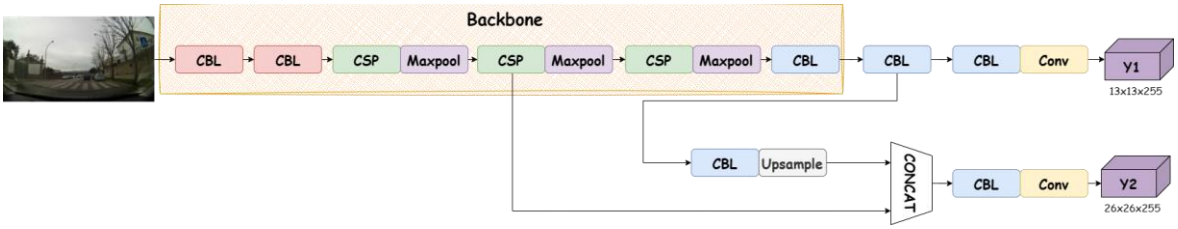


Figura 10 - Arquitetura do modelo *YOLOv4-tiny*. Adaptado de [11].

YOLOv4-tiny utiliza dois *feature maps*, cada um com dimensões de 13 × 13 e 26 × 26, para prever os resultados da deteção. Simultaneamente, cada mapa de características utiliza três *anchor boxes* de tamanhos variáveis para prever três *bounding boxes*.

Conforme mencionado na secção anterior, a primeira abordagem utilizada para a classificação do estado de degradação das passadeiras apresentou algumas limitações em determinadas condições. Para melhorar os resultados, foi adotada uma segunda abordagem. Esta consistiu na construção de um novo *dataset*, que inclui três classes de

degradação das passadeiras pedonais, já mencionadas anteriormente. Estes estados de degradação foram definidos utilizando métodos de *Adaptive Threshold*. Para implementar esta nova abordagem, foi necessário modificar o *dataset* previamente criado pelos mesmos autores [12], originando um novo *dataset*. Na secção seguinte, serão descritas todas as ações realizadas sobre este novo *dataset*.

2.2.1 Dataset

Para melhorar o *dataset* [12] existente, foram adicionadas novas imagens de passadeiras pedonais, passando de um total de 642 para 1182 imagens. Este novo *dataset* está disponível em [13]. As imagens foram capturadas com uma câmara com um ângulo de visão de 160º, conforme demonstrado na Figura 11. Foram tiradas durante o dia, no entanto em um período de céu nublado, o que permitiu diversificar o conjunto de imagens e obter mais informações sobre as passadeiras pedonais.

Figura 11 - Exemplo de uma imagem capturada.

Para a captura destas imagens foi usada uma câmara *Arducam 1080P Low Light WDR USB Camera Module*. Esta possui um suporte de metal que permite que seja facilmente adaptado a um suporte de telemóvel. Esta câmara possui um ângulo de visão *Super-ultrawide* de 160º. Isso facilita a captura de detalhes que poderiam estar ocultos em câmaras com ângulos normais, como 90º, como ilustrado na Figura 12.

Figura 12 - Ilustração dos objetos captados por uma câmara de 90º vs câmara de 160º.

Na Figura 12, mostra a comparação da abertura angular de uma câmara convencional com 90º (a amarelo) e a usada neste projeto de 160º (a vermelho). Como é possível verificar, no caso de câmaras com um ângulo de 90º as passadeiras localizadas à esquerda e direita da imagem não seriam captadas.

Para determinar o estado de degradação das passadeiras pedonais e posteriormente classificar as imagens presentes no *dataset* [14], o *labeling* de todas as imagens presentes neste foi realizado com o método apresentado no capítulo 2.1. Ou seja, a cada imagem foi aplicado o método de *Adaptive Threshold* para determinar qual estado de degradação em que esta melhor se enquadraria. Após esta ação, foram alteradas as *labels* presentes no *dataset,* passando de um *dataset* contendo somente uma classe para três. Assim, o *dataset* passou a ser constituído por, 413 *labels* referentes a passadeiras pedonais classificadas como *Desgaste Moderado*, 471 como *Degaste Severo* e 555 como *Sem Desgaste*, dando um total de 1440 imagens sem utilização de *Data Augmentation*. A Tabela 2 resume a constituição atual do *dataset* criado. Na secção seguinte será apresentada uma avaliação de desempenho comparativa da utilização das duas abordagens descritas na deteção e classificação do estado de degradação de passadeiras pedonais.

Tabela 2 - Número total de *labels* por classe.

Classe	***Total labels***
Sem Desgaste	555
Desgaste Moderado	413
Desgaste Severo	471

2.3 Avaliação de Desempenho

Esta secção descreve a avaliação de desempenho comparativa da utilização dos dois métodos previamente discutidos para a identificação e classificação de desgaste em passadeiras pedonais. Serão abordados o cenário de *benchmark*, as métricas de desempenho e os resultados obtidos nos testes realizados.

2.3.1 Cenário de *Benchmark*

O modelo *YOLOv4-tiny* apresentado foi treinado e testado na plataforma *Google Colab* [15]. Foi utilizado um *CPU* Intel(R) Xeon(R) CPU @ 2.00GHz, 12 GB de *RAM* e uma *GPU* Tesla T4, para a realização dos treinos na plataforma *Google Colab*. Posteriormente este modelo foi testado num *Raspberry PI 5,* com um *CPU Broadcom BCM2712 2.4GHz quad-core* com uma *GPU VideoCore VII* e com 8GB de RAM.

2.3.2 Métricas de Desempenho

Conforme mencionado em [16], o número de iterações recomendado para o modelo *YOLOv4-tiny* é de 12000, sendo este valor especialmente aconselhável para *datasets* com 3 classes. O número de *epochs* pode ser determinado usando a Equação 1.

$$Epochs = \frac{number\ of\ iteractions}{\frac{number\ of\ training\ images}{batch}} \tag{1}$$

A Equação 2, permite o cálculo de *mean Average Precision*, comumente empregada na avaliação de modelos CNN para deteção de objetos, calcula a *mean Average Precision* (mAP), representando a eficácia geral. Esta soma as *average Precision* (AP) de todas as classes, fornecendo um indicador abrangente da capacidade do modelo em identificar objetos.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \tag{2}$$

2.3.3 Resultados e Discussão

Como já referido anteriormente, a utilização do método *Adaptive Threshold*, permitiu determinar o estado de degradação de passadeiras pedonais, presentes em imagens. No entanto, para aplicação mais efetiva desta técnica na classificação do degaste, deveria ser usada uma imagem já com uma passadeira identificada e isolada. Para obtenção dessas imagens, foi utilizado o modelo *YOLOv4-tiny*, já apresentado na etapa anterior deste projeto, mas neste caso re-treinado usando o *dataset* melhorado, este contendo mais imagens. A Tabela 3 apresenta os resultados obtidos anteriormente e usando o novo *dataset*.

Tabela 3 - Comparação dos resultados obtidos entre o modelo apresentado em [8] e o treinado para este projeto (*).

Modelo	***Imagens***	***Input Size***	***mAP (%)***
YOLOv4-tiny [8]	642	608x608	87
YOLOv4-tiny*	1182	608x608	90

Com base nos resultados presentes na Tabela 3, é possível verificar que o modelo usado neste projeto teve uma melhoria de aproximadamente três pontos percentuais de *mAP*. A utilização deste modelo assegura que apenas passadeiras pedonais sejam fornecidas ao método de *threshold* o qual posteriormente determinará o seu estado de degradação. Esta operação pode ser visualizada na Figura 13, que ilustra o referido mecanismo.

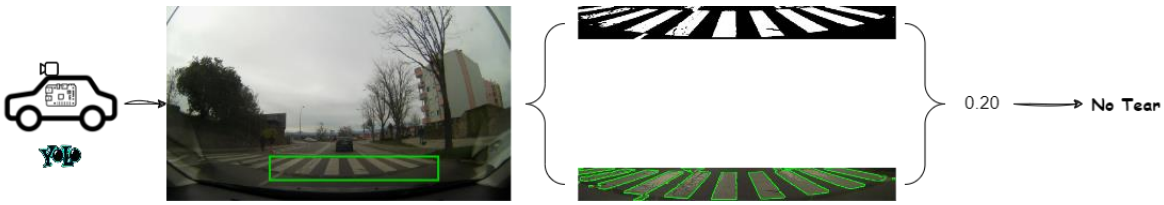


Figura 13 - Diagrama de funcionamento da primeira abordagem.

Embora os resultados obtidos através desta abordagem, até ao momento, demonstrarem poder ser uma solução eficaz, quando aplicada num cenário real, verificou-se alguma dificuldade em identificar o estado de degradação de algumas passadeiras pedonais. Muito embora esta dificuldade possa, em parte, também estar relacionada com a qualidade da câmara utilizada, o excesso de luminosidade do ambiente e até a interferência de sombras, de diferentes elementos, projetadas nas passadeiras revelou ser o principal problema desta abordagem. Na Figura 14 são apresentadas algumas situações que demonstram os problemas detetados. A Figura 14(a) 1), 2) e 3) apresenta o recorte das *bouding boxes* de várias passadeiras pedonaias detetadas pelo modelo *YOLOv4-tiny*. A Figura 14(b) 1), 2) e 3) apresentam, respetivamente, o resultado da utilização do método *Adaptive Threshold* para determinar o estado de degradação de cada uma delas.

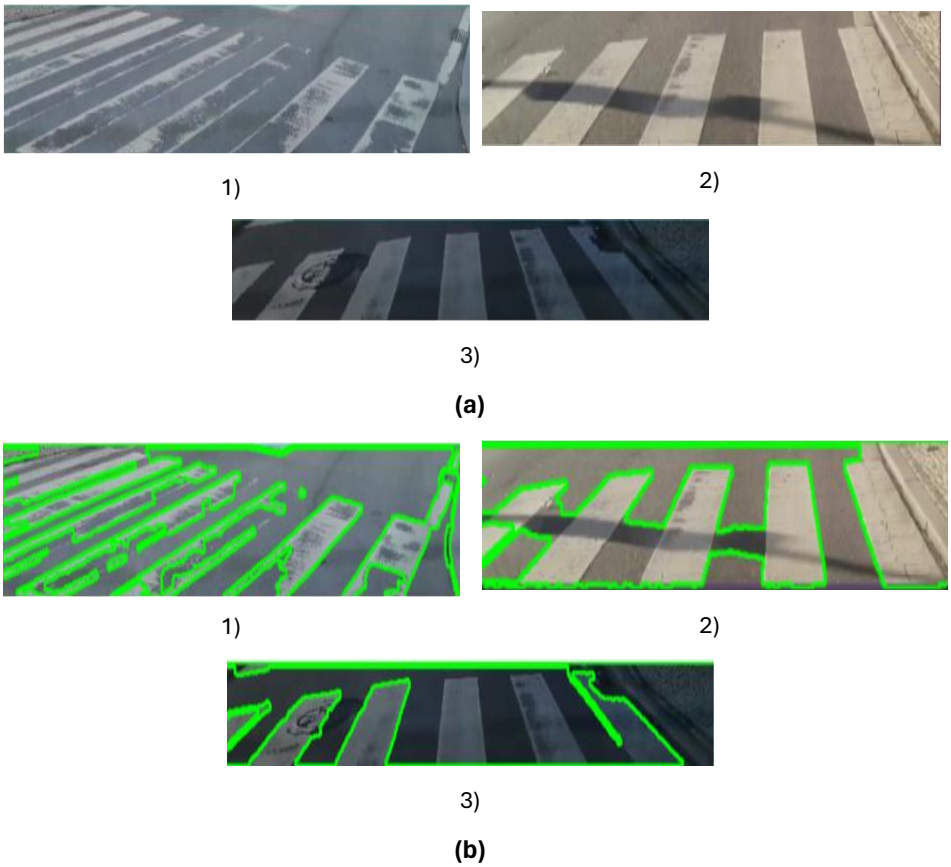


Figura 14 - Exemplos de deteção através de *Adaptive Threshold*.

Como é possível verificar, na Figura 14(b) 1) a deteção dos contornos teve o resultado esperado, demostrando os locais onde existe degradação. Já no caso das Figura 14(b) 2) e Figura 14(b) 3), o resultado obtido não foi o esperado. Nestas figuras é possível visualizar locais onde a técnica não consegue delimitar a passadeira de maneira que seja possível calcular o seu nível de degradação. Por exemplo, na Figura 14(b) 2) a sombra projetada sobre a passadeira influenciou a definição das delimitações dos seus elementos, afetando assim o resultado obtido. Já no caso da Figura 14(b) 3) a aplicação da técnica não conseguiu detetar algumas faixas da passadeira. Com o intuito de investigar o impacto das sombras nos resultados anteriores, esta abordagem foi testada durante um período do dia em que as sombras de possíveis árvores, edifícios ou outros elementos urbanos não incidiam diretamente sobre as passadeiras pedonais. Alguns resultados obtidos podem ser verificados em Figura 15.

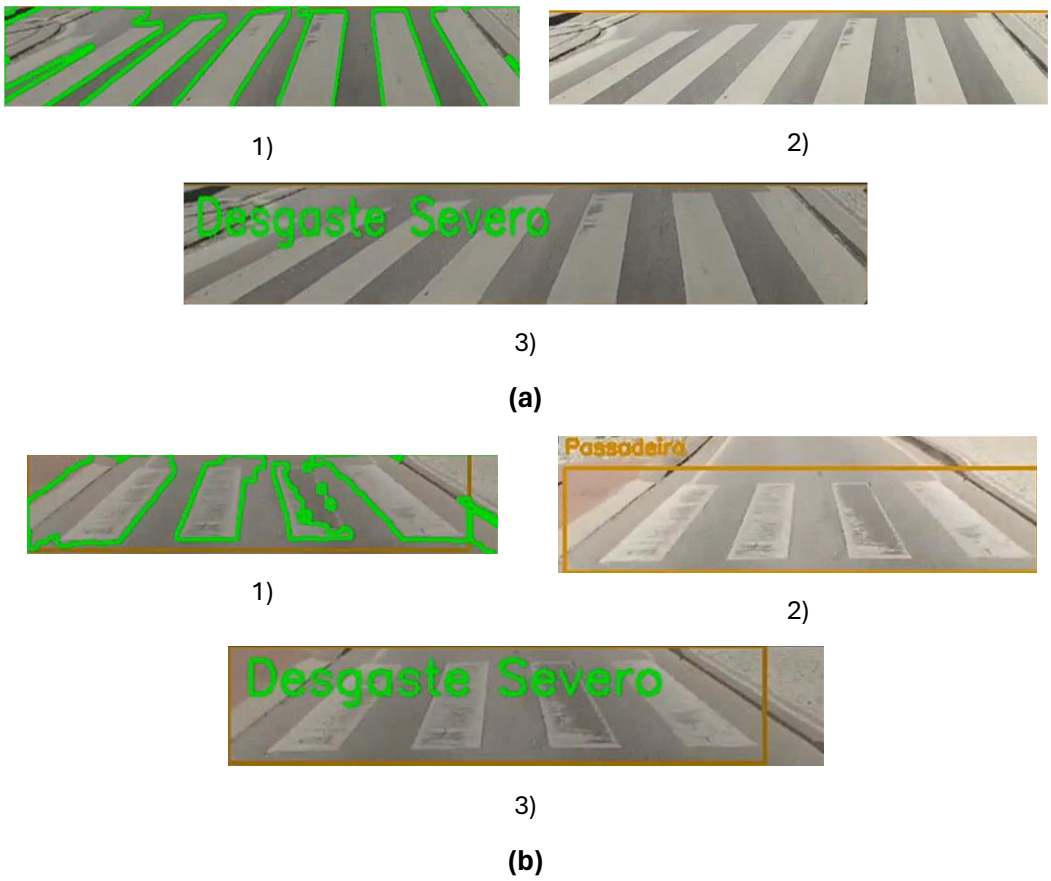


Figura 15 - *Adaptive Threshold* em cenários sem sombras.

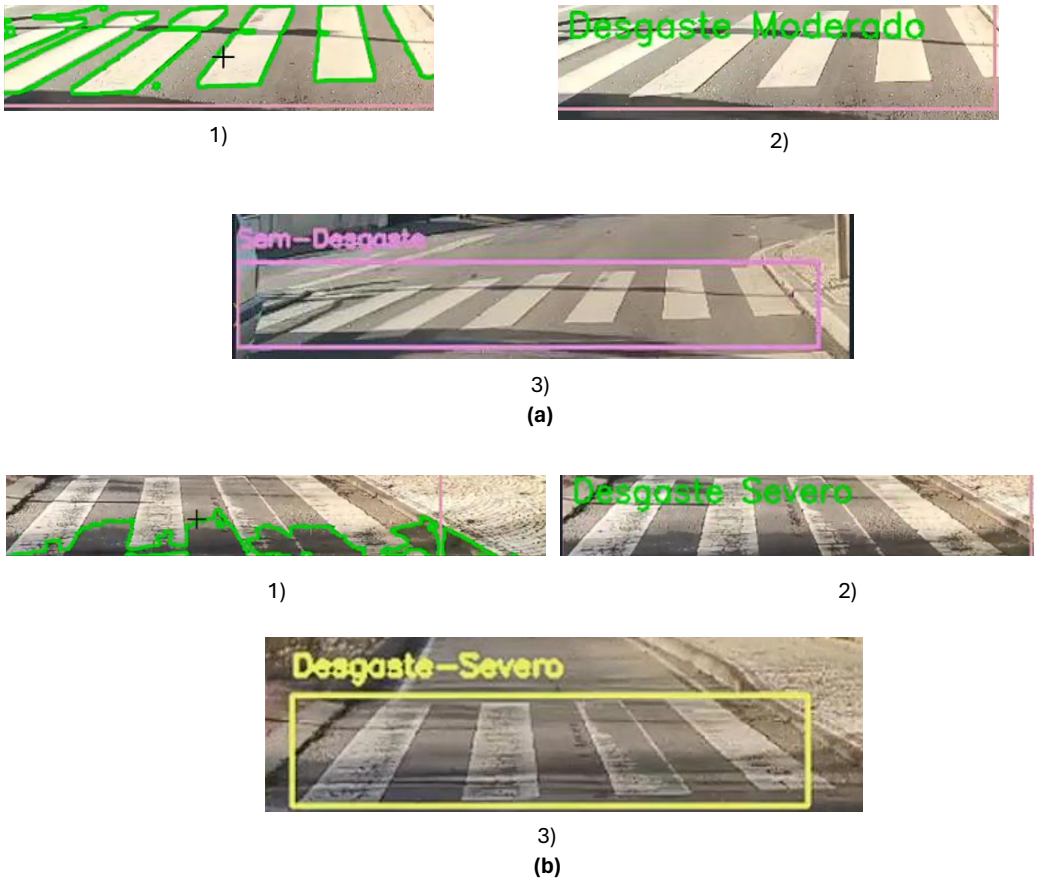
Relativamente à abordagem do modelo *YOLOv4-tiny*, demonstrada no capítulo 2.2, o treino foi realizado utilizando o *dataset* contendo três classes de passadeiras pedonais. Os resultados obtidos são apresentados na Tabela 4. Embora o valor de *mAP* no geral não esteja próximo do modelo do *YOLOv4-tiny* treinado exclusivamente para identificar passadeiras pedonais, e não as classificar, conforme demonstrado anteriormente, há potencial para melhorias com a aquisição de mais imagens. A quantidade de imagens na classe "Desgaste-Moderado", que totaliza 413, influenciou negativamente o resultado de mAP, evidenciando a necessidade de adquirir mais imagens de passadeiras para esta classe.

Tabela 4 - Resultados do modelo *YOLOv4-tiny* para o *dataset* com 3 classes.

Classes	***mAP (%)***	***Overall %***
Sem-Desgaste	***82,13***	***71,21%***
Desgaste-Moderado	***54,72***	
Desgaste Severo	***76,96***	

Comparando as duas abordagens consideradas, é possível verificar que a utilização de um *dataset* já classificado, com os devidos estados de degradação, apresenta resultados superiores. Estes resultados são evidenciados pelo fato desta abordagem

não apresentar dificuldades em determinar o estado de degradação das passadeiras mesmo quando as imagens submetidas as classificações apresentam regiões com sombras ou luminosidade excessiva. Na Figura 16 é possível verificar algumas comparações entre estas duas abordagens. É importante referir que ambas foram aplicadas no mesmo contexto e nas mesmas passadeiras de modo a garantir as mesmas condições de avaliação. É possível verificar em Figura 16(a) 1) e 2) a deteção utilizando a primeira abordagem considerada – *YOLOv4-tiny* e *Adaptive Threshold*. Neste caso, pode ser observado que a passadeira é classificada com desgaste moderado devido a presença de sombras. A Figura 16(a) 3) apresenta o resultado obtido, nas mesmas condições, mas usando a segunda abordagem – *YOLOv4-tiny* com três classes. Neste caso, fica demonstrado que a classificação obtida corresponde ao verdadeiro estado de desgaste desta passadeira, ou seja, sem desgaste. Relativamente à Figura 16(b), ambas as técnicas apresentam a mesma classificação.



1) 2)

3)
(a)

1) 2)

3)
(b)

Figura 16 - *Adaptive Threshold* em cenários com sombras.

Após a análise realizada aos resultados obtidos, será considerado para o desenvolvimento do protótipo, descrito no capítulo seguinte, a utilização da segunda abordagem em análise, ou seja, um *dataset* classificado com três classes de desgaste. Como demostrado, esta abordagem apresento um bom desempenho em todas as condições de utilização, incluindo ambientes com diferentes luminosidades e a presença de projeções de sombras sobre as passadeiras.

3. Protótipo

Neste capítulo, será abordado o processo de desenvolvimento do protótipo destinado à identificação e classificação de desgaste em passadeiras pedonais. É importante salientar, desde o início, que este projeto representa uma prova de conceito e não um produto final. O objetivo principal deste capítulo é detalhar as etapas envolvidas na conceção e implementação do protótipo, oferecendo uma compreensão abrangente da sua arquitetura, componentes de *hardware* e *software* utilizados. Ao longo das próximas secções, serão discutidos os componentes principais do protótipo, os métodos utilizados para a recolha e visualização dos dados.

3.1 Arquitetura

A arquitetura do protótipo foi desenvolvida para oferecer uma solução inteligente e eficaz para controlar o estado das passadeiras pedonais em tempo real, contribuindo para a segurança dos peões e a gestão eficiente do tráfego urbano. O conceito central envolve a integração de módulos *Internet* das Coisas (IdC) em veículos municipais, como camiões do lixo ou carros do município, que circulam regularmente pelas ruas da cidade. Estes módulos IoT estão equipados com câmaras e dispositivos *Global Positioning System* (GPS). Os dados recolhidos são transmitidos para servidores, onde são processados e armazenados numa base de dados *NoSQL*, neste caso *Firebase* [17]. Além disso, uma *Application Programming Interface* (API) alojada na plataforma *Render* [18] é utilizada para facilitar a comunicação entre os dispositivos IdC e a aplicação web. Esta arquitetura permite uma recolha contínua, e em tempo real, de informações sobre o estado das passadeiras, proporcionando uma visão abrangente da situação em toda a cidade onde este protótipo seja aplicado.

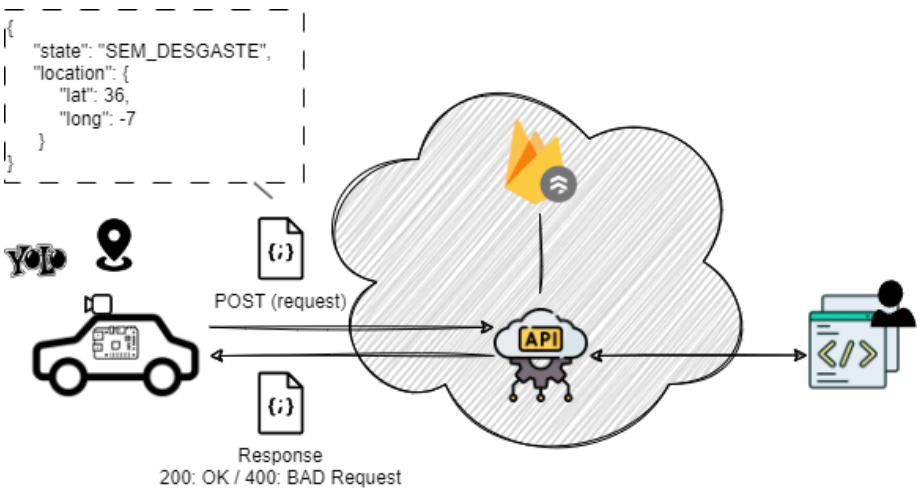


Figura 17 – Arquitetura do protótipo desenvolvido.

A Figura 17 ilustra a arquitetura do protótipo desenvolvido, com os seus vários componentes e a interligação entre eles. Nesta é possível observar a presença dos dispositivos IdC, presentes nos veículos, responsáveis pela captura e transmissão dos dados sobre o estado das passadeiras. Esta transmissão de dados é realizada utilizando dados em formato JSON [19]. A utilização de uma API facilita a comunicação entre o dispositivo IdC e a aplicação *web*. É ainda importante denotar que o dispositivo IdC

necessita de uma API *Key* [20] para comunicar com esta, de forma que somente dispositivos autorizados possam enviar informação. A aplicação *web* desenvolvida proporciona uma interface acessível e interativa para visualização e análise dos dados recolhidos útil num contexto de *smart city* [21]. Nas secções seguintes será detalhado todo o desenvolvimento dos vários componentes presentes na arquitetura apresentada.

3.2 Componentes de *Hardware*

Como elemento de processamento computacional, na construção do dispositivo IoT, a ser integrado nos veículos, foi usado um *Raspberry PI*. Para avaliação de desempenho do dispositivo final fora utilizado um *Raspberry PI* 5 com 8GB RAM [22]. Adicionalmente foi necessário dotar o sistema de um dispositivo de GPS, neste caso foi usado o modelo *GPS NEO-6M* [23]. Finalmente para aquisição de imagem foi necessário a integração de uma câmara, no caso uma *Wide-Angle 1080p UVC-Compliant USB Camera Module with Metal Case* [24]. Para interligar todos estes componentes, foi desenhado um circuito elétrico com recurso à ferramenta *online EasyEDA* [25]. Na Figura 18, é possível observar o esquemático desse circuito elétrico, que inclui todos os componentes descritos:

- *Raspberry PI* (U1)
- *GPS NEO-6M* (U2)
- *Wide-Angle 1080p UVC-Compliant USB Camera Module with Metal Case* (U3)

Tal como referido, o *Raspberry PI* assume o papel de núcleo de processamento, executando o modelo *YOLOv4-tiny* para deteção de desgaste em passadeiras pedonais. O dispositivo de GPS é responsável por fornecer as coordenadas geográficas precisas da localização das passadeiras pedonais. A câmara é responsável pela captura das imagens de qualidade das passadeiras pedonais. O amplo ângulo de visão da câmara selecionada, permite que o modelo *YOLOv4-tiny* obtenha imagens detalhadas da superfície das passadeiras, que facilitam a identificação dos seus sinais de desgaste.

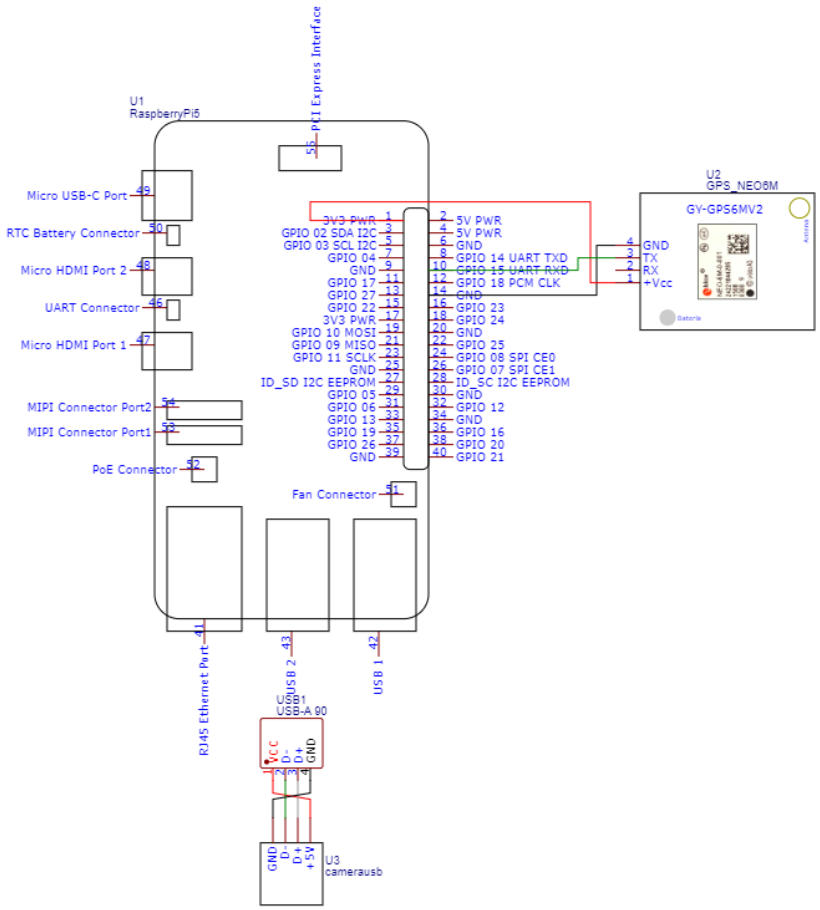


Figura 18 – Esquemático do circuito elétrico dos vários componentes de *hardware*.

Analisando a Figura 18 em detalhe verifica-se que a conexão entre o *Raspberry PI e* a câmara é feita através de uma porta *Universal Serial Bus* (USB). Relativamente à ligação entre este dispositivo e o módulo de GPS, é usada uma ligação *serial* através dos pinos *RX e TX* do *Raspberry PI*. Esta ligação é alimentada usando as ligações 3.3V e *ground* presente no *Raspberry PI*. Na Figura 19 é apresentado os vários componentes de *hardware*, desenvolvidos com base no circuito elétrico apresentado na Figura 18. Nesta é possível verificar a utilização de um suporte de telemóvel, 1) *Phone Mount*, adaptado para suportar a câmara, 2) *Arducam 1080P Low Light*. Esta está conectada via porta *USB 2.0* ao 3) *Raspberry PI 5*, é ainda utilizado o módulo de GPS, 4) *GPS Neo-6m*, através da porta serial RX do *Raspberry PI*, que se interliga a porta TX do módulo de GPS. Finalmente para fonte alimentação é utilizado um 5) *Bluetooth Adapter*, que para além de conter a tecnologia de *Bluetooth,* contém ainda duas portas USB, este é colocado na tomada do isqueiro presente no automóvel e posteriormente ligado ao *Raspberry PI* através de um cabo USB-C. Na próxima secção serão descritas as soluções de *software* desenvolvidas para operação deste *hardware*.

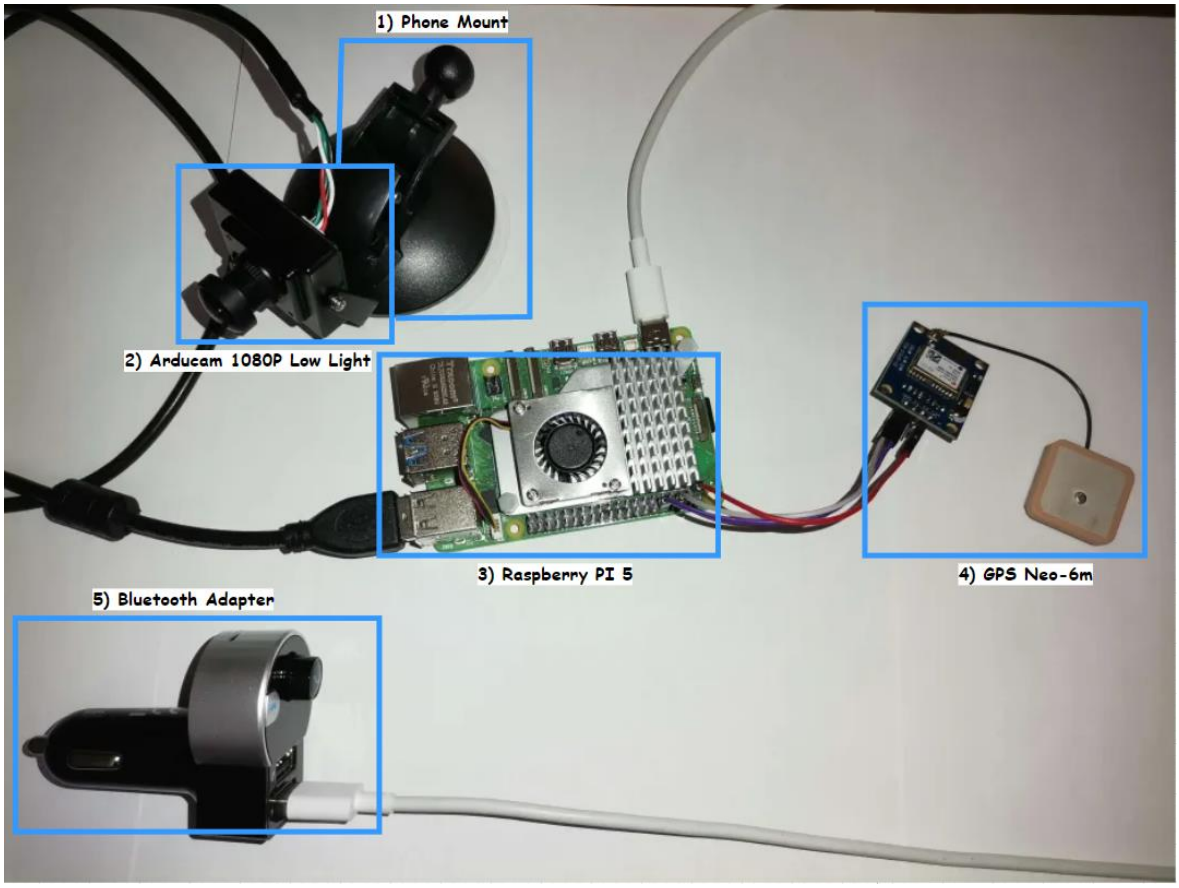


Figura 19 – Componentes de *hardware* do protótipo desenvolvido.

3.3 Componentes de *Software*

Nesta secção, será discutido o desenvolvimento da plataforma integrada, abrangendo a aplicação web, a API responsável por receber os dados do *Raspberry PI* e a base de dados. O objetivo principal desta plataforma é fornecer um espaço onde as entidades responsáveis possam visualizar a localização das passadeiras pedonais e o respetivo estado de degradação.

Nesta primeira parte desta secção são descritos os requisitos da solução a desenvolver. Os requisitos pretendidos pelo utilizador final, definem por vezes as características e iterações presentes numa aplicação [26]. Estes requisitos podem ser separados em dois tipos: funcionais e não funcionais. Os requisitos funcionais estão relacionados com características do produto que os desenvolvedores devem implementar para permitir que os utilizadores atinjam os seus objetivos. Dito isto estes devem estar bem implementados e de forma clara para que os utilizadores e até desenvolvedores possam compreender o sistema [27]. Relativamente aos requisitos não funcionais, estes estão relacionados com as bases do sistema, por outras palavras, relacionados com a velocidade do sistema, segurança e até integridade dos dados [28]. Como forma de compreender os requisitos necessários para o desenvolvimento desta aplicação foi desenhado um diagrama de casos de uso, como demonstra a Figura 20, este construído através de *Unified Modeling Language* (UML) [29]. Um caso de uso tem

como principal objetivo demonstrar como um utilizador ou um sistema pode interagir com um outro sistema [30].

De seguida, são identificados os requisitos funcionais considerados para o desenvolvimento da plataforma. Para cada um desses requisitos é identificado o nome da interação, a sua descrição, as pré-condições e o seu *output*:

1) *Register New Crosswalk Detection*, tem como pré-condição a localização atual do GPS, a execução do modelo *YOLOv4-tiny* e conexão à *internet*. Como *output* esta deteção será guardada pela API, isto se existir acesso à *internet*, caso contrário será guardado localmente.
2) *Send Data to API*, necessita inicialmente que seja detetado uma passadeira pedonal, e posteriormente, verificação da existência de dados guardados localmente a serem enviados. Como *output* a informação é guardada na base de dados através do *endpoint* disponibilizado pela API.
3) *Receive Data from Prototype*, para que este requisito possa funcionar necessita que seja feita uma requisição *Hypertext Transfer Protocol* (HTTP), do tipo POST, por parte do *Raspberry PI*. O *output* será o armazenamento da informação na base de dados.
4) *Send Stored Data,* necessita de uma requisição HTTP, do tipo GET. Serão retornadas todas as deteções guardadas na base de dados.
5) *View Detections*, como pré-condição necessita que o utilizador esteja na página de *dashboard* da aplicação *web*. Como *output* serão demonstradas todas as passadeiras pedonais registadas até ao momento e o seu estado de degradação.

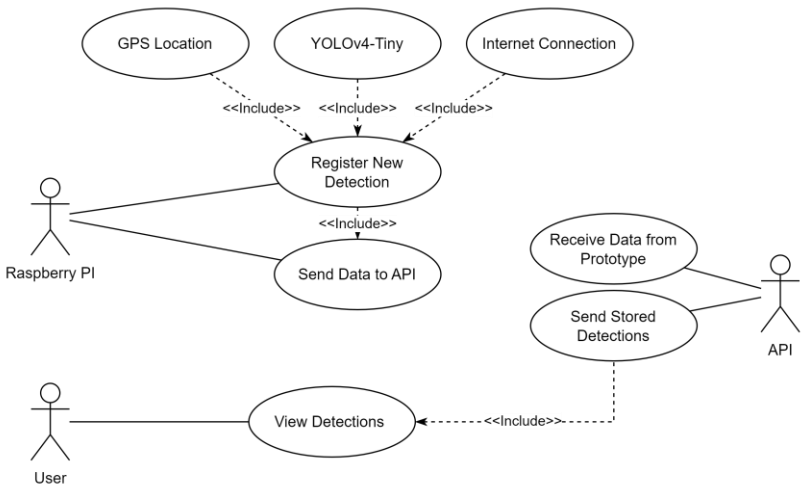


Figura 20 - Diagrama de casos de uso.

Relativamente aos requisitos não funcionais foram tidos em conta requisitos relacionados com a performance aplicação, a construção de interfaces de fácil utilização e intuitivas para os utilizadores e ainda *feedback* de ações/erros que aconteceram durante a utilização da aplicação. No que toca à performance o que foi tido em conta foi a construção de uma aplicação responsiva, rápida e capaz de lidar com elevadas cargas de pedidos HTTP, evitando assim que esta pudesse ficar

indisponível para novas deteções de passadeira pedonais que necessitassem ser guardadas.

Outro requisito inerente à solução a desenvolver está relacionada com a necessidade de armazenamento dos dados recolhidos. Neste caso, para o armazenamento desses dados foi usada uma base de dados não relacional, mas que ao mesmo tempo necessitasse que os seus documentos respeitassem uma determinada estrutura. Esta necessidade garantia que os dados recolhidos pudessem ser apresentados corretamente na *User Interface* (UI) da aplicação, o que foi fundamental para a manutenção e eficiência da mesma. Por último, foi ainda requisito que algumas ações presentes na aplicação CrosSafe pudessem gerar mensagens de erros/informações úteis para o utilizador, facilitando que este pudesse compreender aquilo que está a fazer de errado ou o porquê de uma determinada ação não poder ser completada.

As secções seguintes detalham a implementação dos vários competentes de *software* da solução, tendo em consideração todos os requisitos descritos.

3.3.1 Dispositivo *Internet* das Coisas

No que toca ao dispositivo IdC (*Raspberry Pi*) em si, todo o *software* usado na sua operação foi desenvolvido utilizando a linguagem *Python*, uma vez de esta já oferecer bastantes bibliotecas de fácil utilização para incorporar algoritmos de deteção de objetos, tais como *OpenCV* [31] e *CVLib* [32]. Toda a operação deste dispositivo está representada no fluxograma da Figura 21. Detalhando a sequência de ações realizadas, esta é iniciada assim que o modelo *YOLOv4-tiny* é carregado com sucesso, seguido pelos módulos de GPS e câmara, fundamentais para o funcionamento do sistema. Em seguida, o dispositivo entra num *loop* de espera, aguardando que o modelo detete uma passadeira pedonal e o respetivo desgaste. Assim que esse processo ocorre, as coordenadas atuais são lidas através da porta *serial /dev/ttyAMA0* ligada ao módulo GPS. Após a leituras das coordenadas, estas são convertidas para *encoding UTF-8* através da biblioteca pynmea2 [33]. Os dados são formatados em JSON, de acordo com o esperado pela API (detalhado na secção seguinte), e finalmente enviados. É importante referir que a conexão à *internet* é fundamental para o envio dos dados referentes à sinalização do desgaste das passadeiras. Nesse sentido, se o dispositivo perder a conectividade durante a sua utilização e for necessário enviar dados, estes serão armazenados localmente. Em cada deteção subsequente de uma passadeira pedonal, o sistema verificará a existência de dados locais, para serem enviados juntamente com as novas deteções.

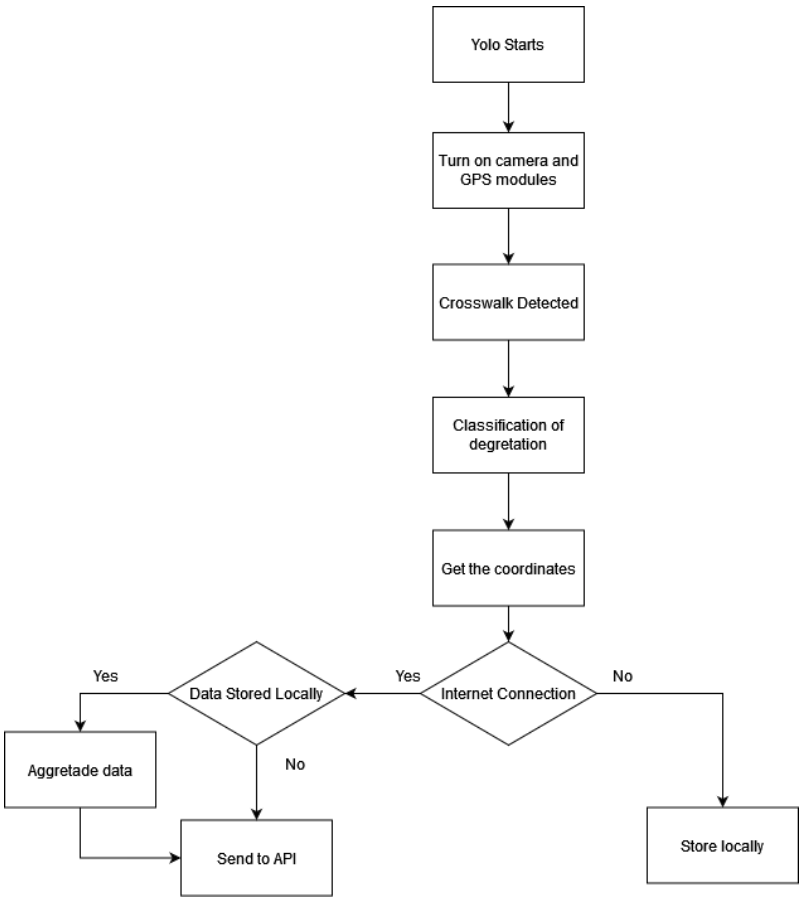


Figura 21 – Fluxograma do funcionamento do *Raspberry PI*.

3.3.2 *Application Programming Interface* e Base de Dados (*Back-end*)

Relativamente à API, esta foi cuidadosamente desenvolvida utilizando a linguagem *NodeJS* [34] em conjunto com a *framework Fastify* [35]. A escolha do NodeJS oferece vantagens significativas, incluindo a sua capacidade para lidar com um grande volume de solicitações de forma eficiente e a sua escalabilidade.

Para o armazenamento de informações, optou-se por utilizar o *Firestore*, da *Firebase* [36]. Sendo esta base de dados NoSQL, oferece uma estrutura flexível e escalável para armazenar e consultar dados em tempo real. Inicialmente foi utilizado uma base de dados *Structured query language* (SQL), neste caso *PostgreSQL* através de PrismaDB [37]. Esta oferecia um *Object Relational Mapping* (ORM) que facilitou a criação de tabelas utilizando *Typescript* para vários tipos de base de dados. No entanto, com o decorrer do desenvolvimento do projeto, esta foi alterada para *Firebase* devido à sua fácil integração e ao facto desta solução ser bastante simples e não necessitar de configurações muito complexas. Todos os *endpoints* implementados foram construídos com base [38], [39]. Como forma de testar todos os *endpoints* localmente foi utilizado a plataforma *Postman* [40]. Na Figura 22 está ilustrado a estrutura dos documentos necessária para o registo de uma nova deteção de uma passadeira pedonal. Estes documentos contêm um campo *state,* indicador do nível de degradação da passadeira, juntamente com o campo *location*, que incorpora a longitude e latitude das coordenadas GPS capturadas.

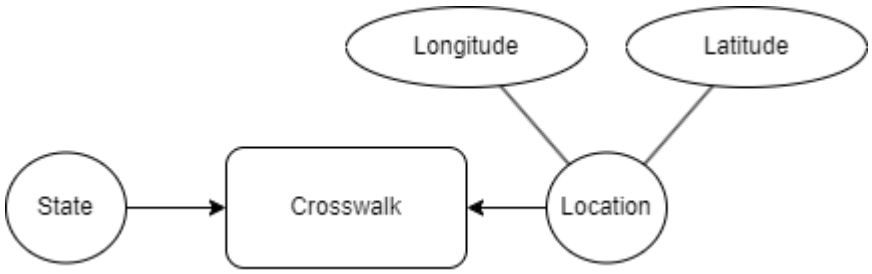


Figura 22 - Estrutura necessária nos documentos.

Os *endpoints* configurados nesta API podem ser visualizadas na Figura 23, onde é possível verificar que somente três métodos HTTP são usados, sendo eles GET, POST e DELETE. O *endpoint /crosswalks* serve tanto para obter todas as deteções de passadeiras pedonais como para registar uma nova. Já o */crosswalks/:id* permite remover uma deteção com um *id* específico.

Resources	GET	POST	PUT	DELETE
/crosswalks	Get current crosswalks	Creates new crosswalk detection	N/A	N/A
/crosswalks/:id	N/A	N/A	N/A	Removes crosswalk detection

Figura 23 - *Endpoints* presentes na API.

Durante o desenvolvimento da *API* foi verificado que o campo *location*, que contém as coordenadas de onde foi efetuada uma deteção de uma passadeira pedonal, poderia ser utilizado para obter o nome da cidade. Esta informação permitiu facilitar a identificação da localização desta passadeira pedonal ao utilizador. Para isto foi utilizada a API da *Geoapify* [41], através do seu *endpoint* de *reverse geocode*, que através das coordenadas retorna o nome da cidade. Na Figura 24 pode ser observada uma resposta exemplo desta API.

```
"features": [
    {
        "type": "Feature",
        "properties": {
            "datasource": {
                "sourcename": "openstreetmap",
                "attribution": "© OpenStreetMap contributors",
                "license": "Open Database License",
                "url": "https://www.openstreetmap.org/copyright"
            },
            "country": "Portugal",
            "country_code": "pt",
            "county": "Portalegre",
            "city": "Alter do Chão",
            "village": "Seda",
            "lon": -7.7892933,
            "lat": 39.1909641,
            "distance": 0,
            "result_type": "city",
            "formatted": "Alter do Chão, Portalegre, Portugal",
            "address_line1": "Alter do Chão",
            "address_line2": "Portalegre, Portugal",
            "category": "administrative",
            "timezone": {
                "name": "Europe/Lisbon",
                "offset_STD": "+00:00",
                "offset_STD_seconds": 0,
                "offset_DST": "+01:00",
                "offset_DST_seconds": 3600,
                "abbreviation_STD": "WET",
                "abbreviation_DST": "WEST"
            },
            "plus_code": "8CFJ56R6+97",
            "plus_code_short": "56R6+97 Alter do Chão, Portalegre, Portugal",
            "rank": {
                "importance": 0.35000999999999993,
                "popularity": 2.561673546613246
            },
            "place_id": "51c3ceb9803c281fc059e41afa8271984340f00101f901b1446200000000c00208"
```


Figura 24 - Exemplo de uma resposta da API *Geoapify*.

Na Figura 26 é possível verificar o diagrama de sequência entre os vários componentes desta aplicação, desde a deteção realizada pelo dispositivo IdC até ao registo na base de dados não relacional da *Firebase.*

Um dispositivo IdC, como um *Raspberry PI*, deteta e classifica uma passadeira pedonal, recolhendo as coordenadas GPS. Em seguida, esses dados são enviados para a API através do *endpoint /crosswalks*. A API processa os dados recebidos e utiliza uma API de R*everse Geocode* para transformar as coordenadas GPS na informação de localização, como o nome da cidade, o nome da rua, entre outros.

Após isso, a API determina se esta deteção é nova ou se já existe uma similar presente na base de dados. Para isso, esta compara a nova deteção com as existentes, calculando a distância entre as coordenadas. Se a distância entre a nova deteção e uma já registada for menor que um valor de *threshold*, a entrada existente na base de dados é atualizada com as novas informações. Se a distância for maior que esse limite, a nova deteção é adicionada como uma entrada separada na base de dados. Este mecanismo está ilustrado na Figura 25, neste caso o valor de *threshold* definido foi de 5 metros.

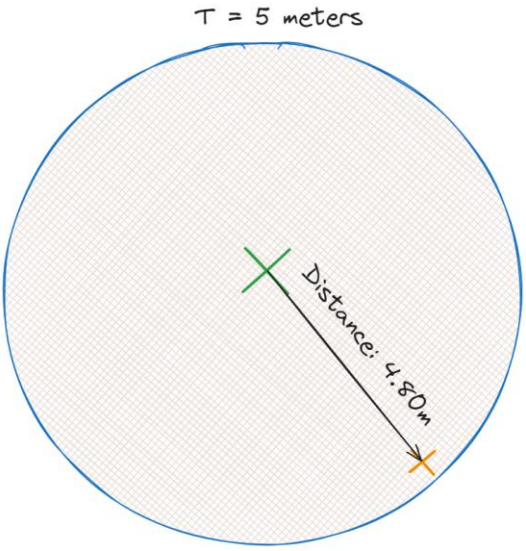


Figura 25 - Ilustração do mecanismo para identificar distância entre coordenadas.

Desta forma, o protótipo garante que as informações sobre passadeiras pedonais sejam precisas e atualizadas, evitando duplicações desnecessárias e mantendo a base de dados organizada.

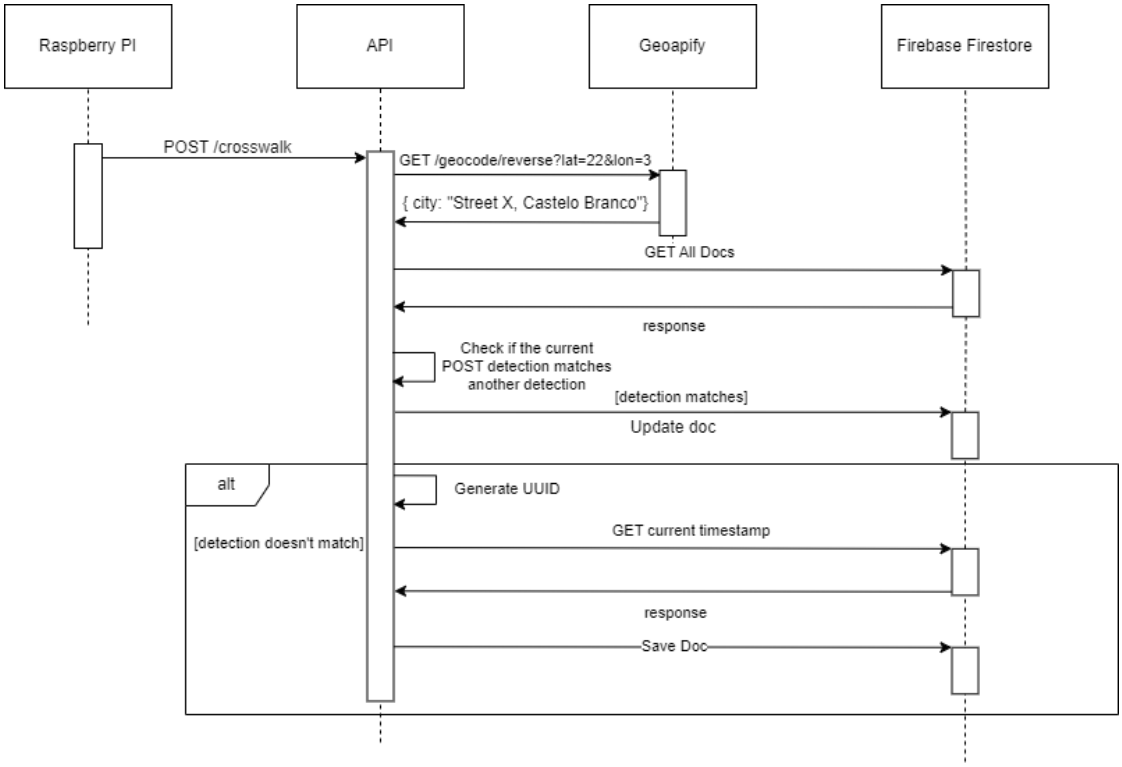


Figura 26 - Diagrama de sequência entre API e Raspberry PI.

Após o desenvolvimento da API foi necessário realizar o seu *deploy*. Para este efeito foi utilizado a plataforma *Render* [18]. Esta dispõe de vários serviços para realizar o *deploy* de diferentes aplicações. No caso foi escolhido a opção *Web Service* [42], para *NodeJS*.

3.3.3 Aplicação *Web* (*Front-end*)

A aplicação *web* "CrosSafe" foi desenvolvida utilizando a *framework ReactJS* [43], com a variante *TypeScript* [44], visando assegurar uma base sólida para o desenvolvimento e manutenção do código. Esta oferece a vantagem de permitir a definição de tipos estáticos para variáveis, propriedades e funções, o que facilita a deteção de erros e contribui para aprimorar a qualidade do código desenvolvido. Adicionalmente, para otimizar o fluxo de desenvolvimento e garantir uma construção eficiente da aplicação, optou-se por integrar o *Vite* [45] como a ferramenta de *build*. Este destaca-se por oferecer um servidor de desenvolvimento extremamente rápido e uma *pipeline* de construção otimizada. Esta ferramenta aproveita ao máximo a capacidade dos navegadores modernos, importando módulos ES diretamente, o que resulta num tempo de desenvolvimento mais ágil e uma experiência de criação mais fluida.

No que diz respeito à estilização das páginas e componentes, adotou-se por uma abordagem moderna e eficiente, combinando o *Tailwind CSS* [46] com a biblioteca de componentes *Shadcn UI* [47]. O *Tailwind CSS* proporciona uma metodologia baseada em classes utilitárias, facilitando a estilização dos elementos de forma rápida e consistente. Por sua vez, a biblioteca de componentes *Shadcn UI* oferece uma ampla variedade de componentes pré-projetados e estilizados, agilizando o processo de desenvolvimento e garantindo uma experiência visualmente coesa em toda a aplicação.

Relativamente à comunicação entre a API e a aplicação CrosSafe, esta pode ser visualizada pelo diagrama de sequência presente em Figura 27. Esta comunicação utiliza dois *endpoints* da API, o *endpoint /crosswalks*, que permite obter todos os documentos registados até ao momento na base de dados e o *endpoint /crosswalks:id*, que permite marcar uma passadeira pedonal como reparada e posteriormente ser removida da base de dados.

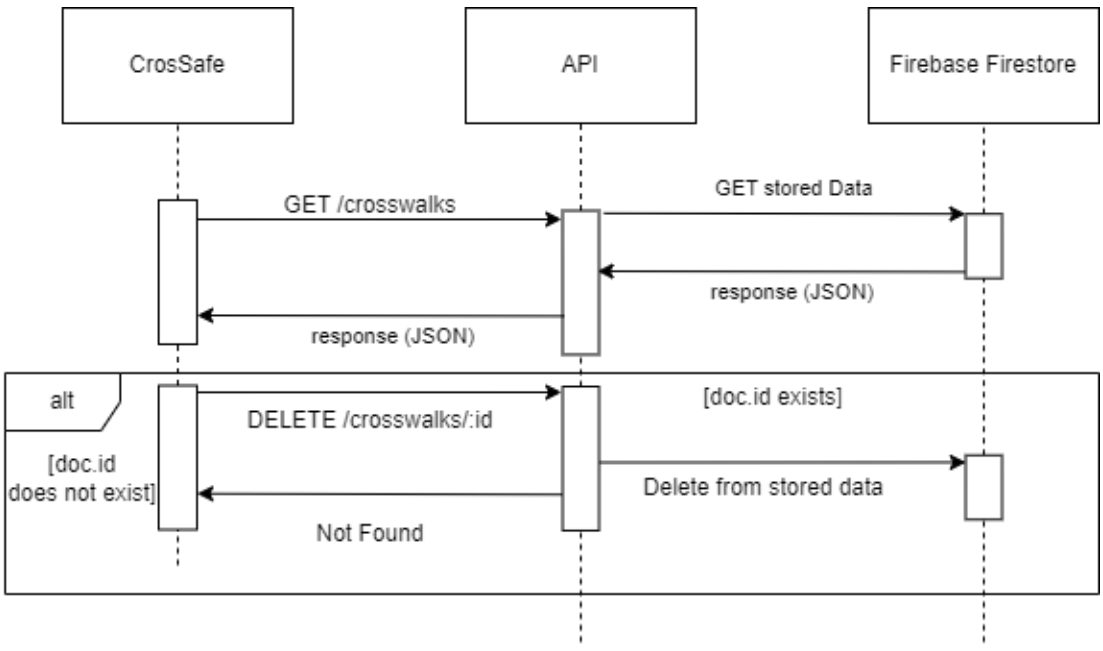


Figura 27 - Diagrama de sequência da comunicação entre a API e a aplicação CrosSafe.

Posteriormente os dados armazenados podem ser visualizados na aplicação *web* da CrosSafe. Esta aplicação contém duas páginas, a página *home* e a página *dashboard*. Os *mockups* destas duas páginas podem ser visualizados na Figura 28 e na Figura 29, respetivamente.

Figura 28 - *Mockup* página *Home*.

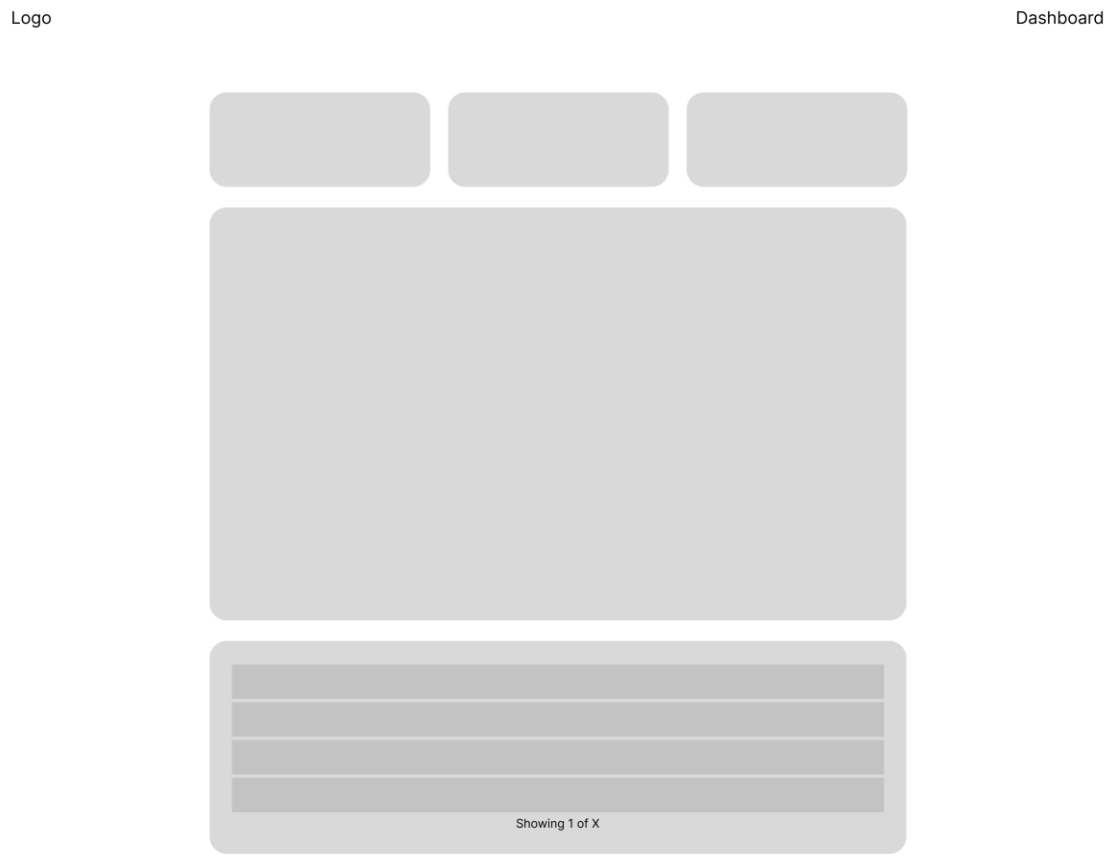


Figura 29 - *Mockup* página de *dashboard*.

Na página *dashboard* são apresentadas todas as deteções presentes na base de dados até ao momento. Esta página é constituída por três componentes, sendo eles:

1) Contagem de passadeiras pedonais por estado de degradação (Figura 30);
2) Mapa demonstrativo da localização das deteções (Figura 31);
3) Tabela com informações adicionais sobre as passadeiras pedonais detetas (Figura 32);

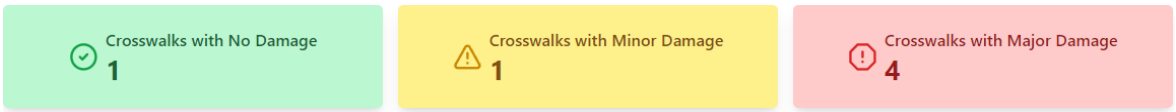


Figura 30 - Componente de contagem de passadeiras por estado de degradação.

Na Figura 30 é apresentado o primeiro componente desta página, onde são apresentadas todas as passadeiras pedonais divididas por estado de degradação, este componente permite sumarizar as informações presentes nos restantes componentes.

Relativamente ao componente do mapa, foi utilizada a biblioteca L*eaflet* [48]. Esta disponibiliza mapas interativos para *Javascript*, o que facilita a integração de um mapa interativo e a visualização exata das deteções. Na Figura 31 é possível visualizar este componente em detalhe. Nele é apresentada a localização das passadeiras pedonais e o seu estado de degradação, estes estados são apresentados mediante 3 cores

diferentes: a verde as passadeiras Sem Desgaste, a amarelo Desgaste Moderado e a avermelhado Desgaste Severo.

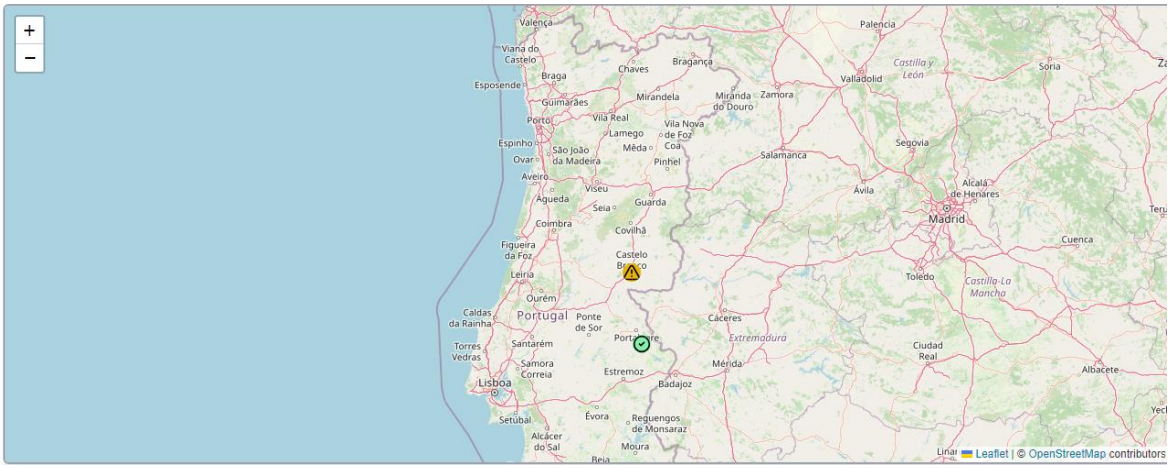


Figura 31 - Mapa interativo presente na aplicação.

No que toca à renderização do elevado número de listas de dados, em aplicações *ReactJS*, esta pode ser feita de várias formas [49]. A tabela presente na página de d*ashboard* utiliza o método de *pagination*, que permite basicamente controlar o número de itens a serem mostrados na aplicação. Este método divide os itens em várias "páginas", caso o número de elementos contidos na lista seja superior a um limiar definido.

Na Figura 32 é possível verificar a implementação do componente tabela, da página *dashboard*, utilizando o método de *pagination*. Nesta tabela são mostradas várias informações desde o *status*, que reflete o estado de degradação da passadeira, a cidade onde esta foi detetada e ainda o dia e hora em que foi detetada. Esta tabela contém ainda uma coluna de *actions* que permite marcar o registo da deteção da passadeira pedonal como reparado. Ao clicar neste botão irá ser apresentado um *popup* de confirmação para que o utilizador garanta que realmente pretende realizar esta ação. De recordar que a marcação de uma passadeira como reparada, elimina esta entrada da base de dados. A Figura 33 ilustra este procedimento.

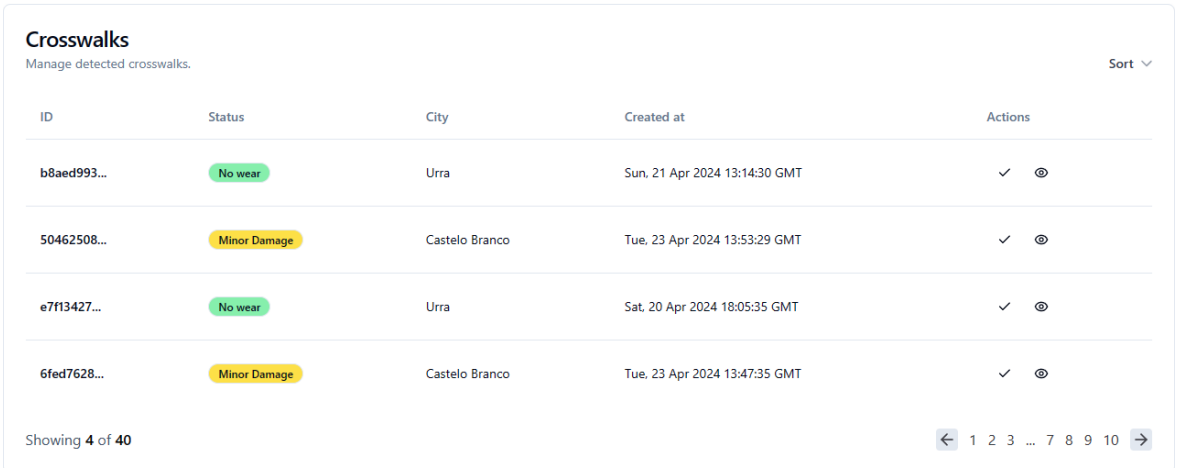


Figura 32 - Tabela com informações adicionais sobre as passadeiras pedonais detetas, utilizando o método de *pagination*.

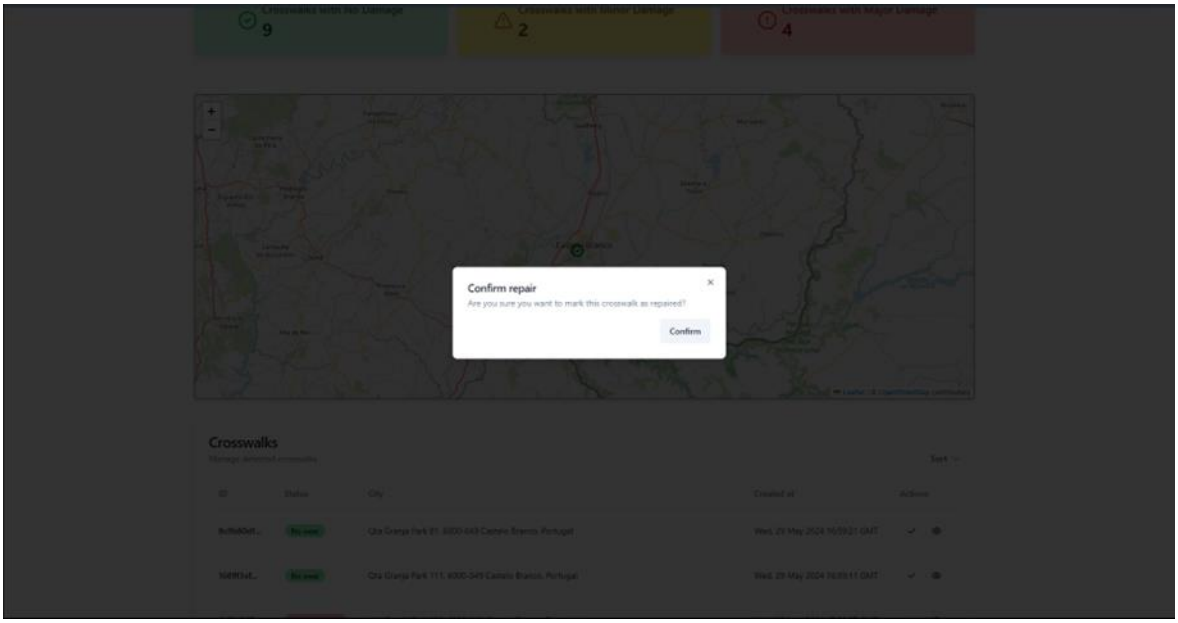


Figura 33 - *Popup* para confirmação de reparação de passadeira.

Após a confirmação por parte do utilizador será apresentada uma mensagem de sucesso no canto inferior direto, presente na Figura 34.

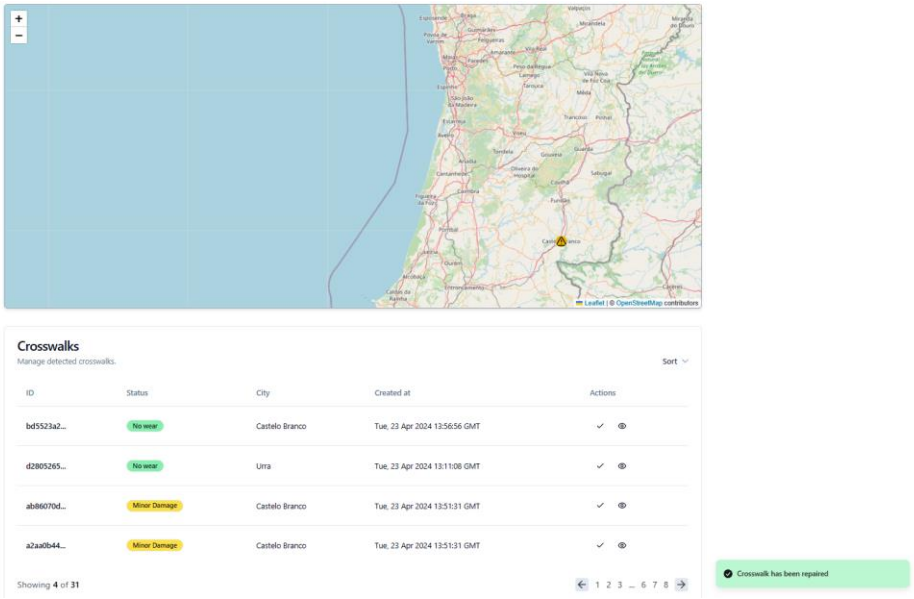


Figura 34 - Exemplo de uma mensagem de sucesso após confirmação de reparação de passadeira.

4. Testes e Validação

Neste capítulo, será delineado o processo de validação do protótipo desenvolvido para identificação e classificação de desgaste em passadeiras pedonais. O foco central deste capítulo é a descrição dos passos envolvidos na validação do protótipo, desde a deteção inicial de uma passadeira, e respetiva classificação, até à visualização dos dados na aplicação *web*.

Durante os testes realizados, utilizando uma *Powerbank* como fonte de alimentação para o *Raspberry PI*, observou-se um gasto adicional de corrente. Esta corrente adicional deve-se ao facto da necessidade de suportar o uso do *cooler*, utilizado para o arrefecimento do *Raspberry PI*. Caso contrário este desligava-se sem aviso prévio. Na Tabela 5, encontram-se registados os consumos identificados com todos os componentes montados.

Tabela 5 - Consumos do *Raspberry PI*.

Hardware	***Watts***	***Volts***	***Amperes***
Raspberry PI 5 8GB c/ cooler	27	5	6
Raspberry PI 5 8GB s/ cooler	27	5	5

O processo de deteção de desgaste em passadeiras pedonais inicia-se com a identificação de uma passadeira pedonal e classificação do seu estado de degradação pelo dispositivo IdC. Na Figura 35 é exemplificado o resultado desta etapa. Em seguida, o dispositivo IdC envia os dados dessa deteção para a plataforma integrada através da API desenvolvida. Estes dados, uma vez recebidos, são armazenados na base de dados *Firestore,* da *Firabase*. Na Figura 36, é exibido um gráfico demonstrativo da quantidade de dados recebidos por esta base de dados, num determinado intervalo de tempo.

Figura 35 - Exemplo de uma passadeira pedonal classificada com desgaste severo.

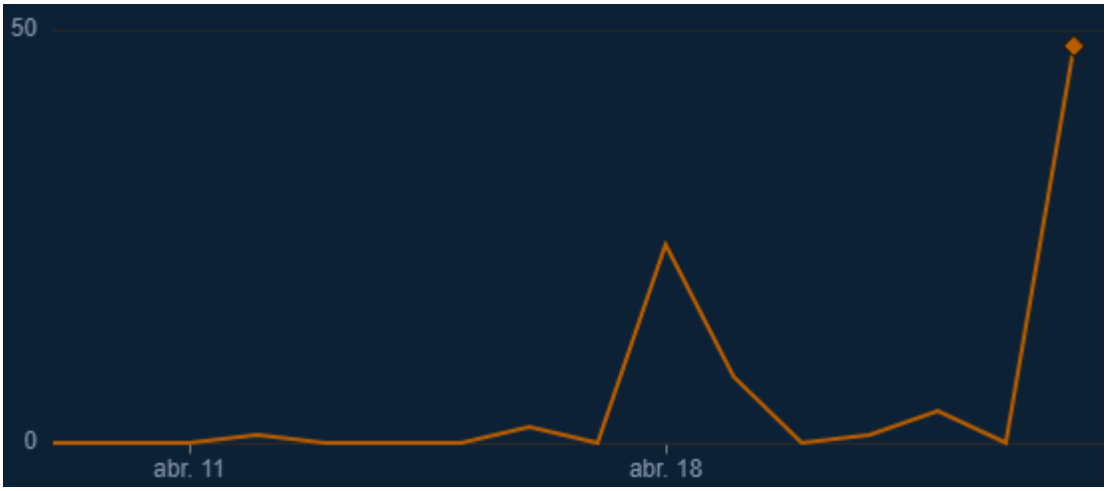


Figura 36 - Exemplo da quantidade de documentos recebidos pela *Firestore*.

Para verificar o correto funcionamento do protótipo criado, foi realizado um percurso pela cidade de Castelo Branco. Este percurso é ilustrado na Figura 37. Escolheu-se uma rota com passadeiras pedonais em diversos estados de degradação, durante um período do dia em que o sol estava relativamente baixo. Esse horário aumentou a dificuldade do modelo *YOLOv4-tiny* em detetar e classificar o estado de degradação das passadeiras pedonais devido à presença de sombras e excesso de luminosidade. Ao longo do percurso, encontraram-se 17 passadeiras. Destas, 9 estavam em boas condições, 3 apresentavam desgaste moderado e 5 mostravam desgaste severo.

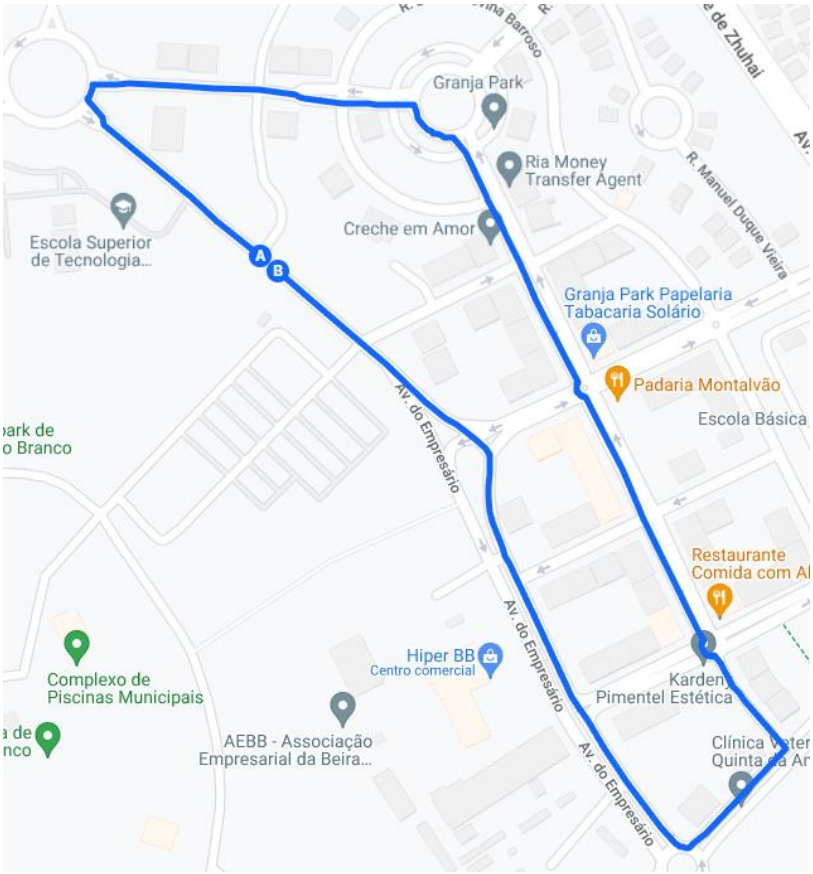


Figura 37 - Ilustração do percurso realizado.

Relativamente ao *Raspberry PI 5* juntamente com o modelo *YOLOv4-tiny*, identificaram o estado de degradação das passadeiras pedonais, num tempo médio de 130ms, como pode ser observado na Figura 38. Em relação às deteções realizadas, o dispositivo conseguiu identificar 15 das 17 passadeiras presentes no percurso. Esta informação está resumida na Tabela 6.

Tabela 6 - Comparação das classificações reais com as efetuadas pelo modelo *YOLOv4-tiny*.

Classe	***Classificação Real***	***YOLOv4-Tiny***
Sem-Desgaste	9	9
Desgaste-Moderado	3	2
Desgaste-Severo	5	4
Total	17	15

```
Detection and classification took 0.1493 seconds
Sending GPS data and objects detected to server... SEM_DESGASTE
Request successful!
Detection and classification took 0.1489 seconds
Sending GPS data and objects detected to server... SEM_DESGASTE
Request successful!
Detection and classification took 0.1459 seconds
Sending GPS data and objects detected to server... SEM_DESGASTE
Request successful!
Detection and classification took 0.1529 seconds
Sending GPS data and objects detected to server... SEM_DESGASTE
Request successful!
Detection and classification took 0.1247 seconds
Sending GPS data and objects detected to server... SEM_DESGASTE
Request successful!
Detection and classification took 0.1268 seconds
Sending GPS data and objects detected to server... SEM_DESGASTE
Request successful!
Detection and classification took 0.1345 seconds
Sending GPS data and objects detected to server... SEM_DESGASTE
Request successful!
```

Figura 38 - Tempo de deteção e classificação necessário.

Um dos problemas detetados foi a dificuldade em identificar passadeiras que se encontravam na diagonal. Um exemplo deste problema pode ser observado na Figura 39. Assim, torna-se necessário realizar uma recolha maior de imagens para aumentar a precisão desta solução.

Figura 39 - Imagem contendo uma passadeira ligeiramente na diagonal.

Os dados transmitidos pelo *Raspberry PI*, podem ser visualizados na plataforma *web*, como demonstrado na Figura 40. Nesta é possível visualizar exatamente a posição das passadeiras pedonais e o seu estado de degradação. Esta visualização detalhada

facilita a monitorização e manutenção das infraestruturas pedonais, contribuindo para a segurança e comodidade dos peões.

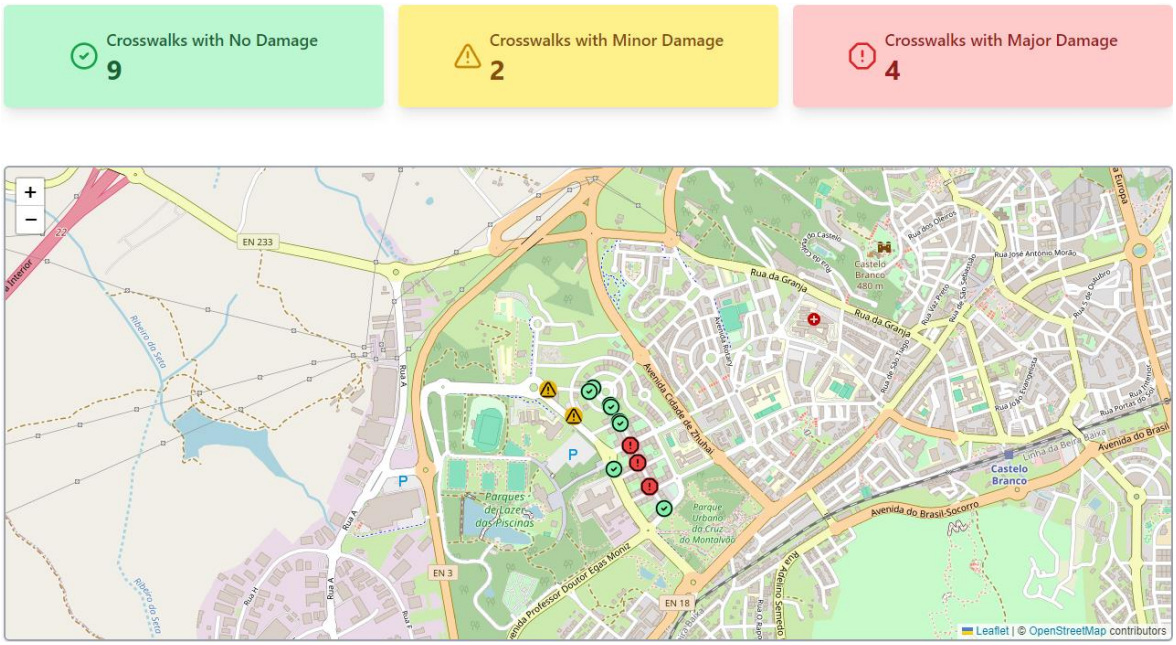


Figura 40 - Visualização dos dados transmitidos pelo *Raspberry PI* na aplicação *web* da CrosSafe.

Com base nos testes realizados e nos resultados obtidos, identificou-se a necessidade de aumentar o número de imagens no *dataset*. Esta melhoria é essencial para aprimorar a precisão na classificação do desgaste das passadeiras pedonais. A ampliação deste *dataset* permitirá um treino mais robusto dos modelos de classificação, resultando em avaliações mais precisas e fiáveis do estado das passadeiras, contribuindo para uma manutenção mais eficaz e segura das mesmas.

5. Conclusão e Trabalho Futuro

A importância da conservação e manutenção das passadeiras pedonais não deve ser subestimada. Não só garantem a segurança dos pedestres, como também desempenham um papel crucial na organização do tráfego urbano. Passadeiras conservadas são essenciais para prevenir acidentes e assegurar que pessoas de todas as idades possam atravessar ruas com segurança. Contudo, este património urbano está em risco devido ao desgaste natural e ao impacto contínuo da atividade humana e das condições climáticas. É necessário, portanto, uma intervenção eficaz para garantir a sua conservação.

O trabalho desenvolvido no contexto deste projeto contribui para este esforço de manutenção. Apresenta o desenvolvimento de um protótipo que utiliza técnicas de visão computacional e uma plataforma integrada para monitorizar o desgaste em passadeiras pedonais. Esta plataforma poderá ser utilizada por qualquer entidade responsável pela manutenção das infraestruturas rodoviárias, dando a possibilidade de controlar e validar reparações que estas necessitem, de modo a proteger os seus cidadãos de eventuais acidentes e assim contribuir para a diminuição da taxa de mortalidade devido a atropelamentos rodoviários.

Permanecem em aberto diversos pontos para trabalho futuro, dos quais se destacam: 1) criação de um *dataset* com suporte a uma quantidade alargada passadeiras pedonais em diversos ambientes climatéricos, com um número alargado de imagens para cada tipo de desgaste de degradação; 2) continuar o processo de validação da aplicação *web* com um conjunto alargado de utilizadores; 3) utilizar o *feedback* dos utilizadores reais para adicionar funcionalidades à interface de utilizador que enriqueçam e facilitem a sua utilização; 4) testar e avaliar outros modelos CNN; 5) testar e avaliar hardware com recursos computacionais superior aos testados neste projeto.

Referências

[1] C. Técnica, "Relatório Novembro 2023," Barcarena, Lisboa , Feb. 2024. Accessed: Apr. 21, 2024. [Online]. Available: https://www.dgeg.gov.pt/pt/estatistica/energia/petroleo-e-derivados/vendas-mensais/

[2] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, "Contextualizing object detection and classification," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1585–1592, 2011, doi: 10.1109/CVPR.2011.5995330.

[3] "What is Computer Vision? | IBM." Accessed: Dec. 08, 2023. [Online]. Available: https://www.ibm.com/topics/computer-vision

[4] I. H. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *SN Comput Sci*, vol. 2, no. 6, pp. 1–20, Nov. 2021, doi: 10.1007/S42979-021-00815-1/FIGURES/11.

[5] "What is Deep Learning? - Deep Learning Explained - AWS." Accessed: Jan. 12, 2024. [Online]. Available: https://aws.amazon.com/what-is/deep-learning/

[6] Y.-K. Huo, G. Wei, Y.-D. Zhang, and L.-N. Wu, "An Adaptive Threshold for the Canny Operator of Edge Detection".

[7] P. Roy, S. Dutta, N. Dey, G. Dey, S. Chakraborty, and R. Ray, "Adaptive thresholding: A comparative study," *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies, ICCICCT 2014*, pp. 1182–1186, Dec. 2014, doi: 10.1109/ICCICCT.2014.6993140.

[8] G. J. M. Rosa, J. M. S. Afonso, P. D. Gaspar, V. N. G. J. Soares, and J. M. L. P. Caldeira, "Detecting Wear and Tear in Pedestrian Crossings Using Computer Vision Techniques: Approaches, Challenges, and Opportunities," *Information*, vol. 15, no. 3, p. 169, Mar. 2024, doi: 10.3390/info15030169.

[9] P. Xu *et al.*, "On-Board Real-Time Ship Detection in HISEA-1 SAR Images Based on CFAR and Lightweight Deep Learning," *Remote Sensing 2021, Vol. 13, Page 1995*, vol. 13, no. 10, p. 1995, May 2021, doi: 10.3390/RS13101995.

[10] D. Misra, "Mish: A Self Regularized Non-Monotonic Activation Function," *31st British Machine Vision Conference, BMVC 2020*, Aug. 2019, Accessed: Apr. 16, 2024. [Online]. Available: https://arxiv.org/abs/1908.08681v3

[11] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," Apr. 2020, Accessed: Apr. 19, 2024. [Online]. Available: http://arxiv.org/abs/2004.10934

[12] "Crosswalks Dataset > Overview." Accessed: Dec. 05, 2023. [Online]. Available: https://universe.roboflow.com/projeto-5fy5m/crosswalks-zbjgg

[13] "Project Overview." Accessed: Apr. 18, 2024. [Online]. Available: https://universe.roboflow.com/passadeiras/crosswalks-one-class

[14] "Crosswalks Dataset > Overview." Accessed: Apr. 18, 2024. [Online]. Available: https://universe.roboflow.com/passadeiras/crosswalks-uo9bq

[15] "Google Colab." Accessed: Apr. 20, 2024. [Online]. Available: https://colab.research.google.com/

[16] "GitHub - AlexeyAB/darknet: YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection (Windows and Linux version of Darknet)." Accessed: Apr. 22, 2024. [Online]. Available: https://github.com/AlexeyAB/darknet

[17] "Firebase | Google's Mobile and Web App Development Platform." Accessed: Apr. 02, 2024. [Online]. Available: https://firebase.google.com/?hl=pt

[18] "Cloud Application Hosting for Developers | Render." Accessed: Apr. 02, 2024. [Online]. Available: https://render.com/

[19] "JSON." Accessed: May 07, 2024. [Online]. Available: https://www.json.org/json-en.html

[20] "O que é uma chave de API? — Explicação sobre chaves e tokens de API — AWS." Accessed: Apr. 21, 2024. [Online]. Available: https://aws.amazon.com/pt/what-is/api-key/

[21] Y. Chuantao, X. Zhang, C. Hui, W. Jingyuan, C. Daven, and D. Bertrand, "A literature survey on smart cities," *18. Sci China Inf Sci*, vol. 58, no. 18, p. 100102, 2015, doi: 10.1007/s11432-015-5397-4.

[22] "Buy a Raspberry Pi 5 – Raspberry Pi." Accessed: May 16, 2024. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-5/

[23] "MÓDULO GPS NEO-6M - UART C/ ANTENA CERÂMICA." Accessed: May 16, 2024. [Online]. Available: https://www.botnroll.com/pt/gps-gnss/4548-m-dulo-gps-neo-6m-uart-c-antena-cer-mica.html

[24] "Arducam 1080P Low Light WDR USB Camera Module with Metal Case." Accessed: May 16, 2024. [Online]. Available: https://www.arducam.com/product/b020201-arducam-1080p-low-light-wdr-usb-camera-module-with-metal-case-2mp-1-2-8-cmos-imx291-160-degree-ultra-wide-angle-mini-uvc-webcam-board-with-microphones/

[25] "EasyEDA - PCB design & simulação de circuitos online." Accessed: Apr. 04, 2024. [Online]. Available: https://easyeda.com/pt

[26] M. Maguire and N. Bevan, "User Requirements Analysis," pp. 133–148, 2002, doi: 10.1007/978-0-387-35610-5_9.

[27] "Functional requirements examples and templates - Jama Software." Accessed: Apr. 16, 2024. [Online]. Available: https://www.jamasoftware.com/requirements-management-guide/writing-requirements/functional-requirements-examples-and-templates

[28] "Nonfunctional Requirements: Examples, Types and Approaches." Accessed: Apr. 16, 2024. [Online]. Available: https://www.altexsoft.com/blog/non-functional-requirements/

[29] G. Booch, J. Rumbaugh, and I. Jacobson, "The Unified Modeling Language for Object-Oriented Development Documentation Set Version 0.9a Addendum", Accessed: Apr. 16, 2024. [Online]. Available: http://www.rational.com

[30] "UML Use Case Diagram Tutorial | Lucidchart." Accessed: Apr. 16, 2024. [Online]. Available: https://www.lucidchart.com/pages/uml-use-case-diagram

[31] "OpenCV - Open Computer Vision Library." Accessed: Apr. 04, 2024. [Online]. Available: https://opencv.org/

[32] "GitHub - arunponnusamy/cvlib: A simple, high level, easy to use, open source Computer Vision library for Python." Accessed: Apr. 04, 2024. [Online]. Available: https://github.com/arunponnusamy/cvlib

[33] "GitHub - Knio/pynmea2: Python library for parsing the NMEA 0183 protocol (GPS)." Accessed: May 08, 2024. [Online]. Available: https://github.com/Knio/pynmea2

[34] "Node.js — Run JavaScript Everywhere." Accessed: Apr. 02, 2024. [Online]. Available: https://nodejs.org/en

[35] "Fast and low overhead web framework, for Node.js | Fastify." Accessed: Apr. 02, 2024. [Online]. Available: https://fastify.dev/

[36] "Firestore | Firebase." Accessed: May 16, 2024. [Online]. Available: https://firebase.google.com/docs/firestore

[37] "Prisma | Simplify working and interacting with databases." Accessed: Apr. 21, 2024. [Online]. Available: https://www.prisma.io/

[38] "Web API design best practices - Azure Architecture Center | Microsoft Learn." Accessed: Apr. 21, 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design

[39] J. Bloch, "How to Design a Good API and Why it Matters".

[40] "Postman API Platform | Sign Up for Free." Accessed: Apr. 18, 2024. [Online]. Available: https://www.postman.com/

[41] "Reverse Geocoding API | Lat/Lon to address | Geoapify." Accessed: Apr. 18, 2024. [Online]. Available: https://www.geoapify.com/reverse-geocoding-api

[42] "Web Services | Render Docs." Accessed: Apr. 08, 2024. [Online]. Available: https://docs.render.com/web-services

[43] "React." Accessed: Apr. 02, 2024. [Online]. Available: https://react.dev/

[44] "TypeScript: JavaScript With Syntax For Types." Accessed: Apr. 21, 2024. [Online]. Available: https://www.typescriptlang.org/

[45] "Vite | Next Generation Frontend Tooling." Accessed: Apr. 02, 2024. [Online]. Available: https://vitejs.dev/

[46] "Tailwind CSS." Accessed: Mar. 31, 2024. [Online]. Available: https://tailwindcss.com/

[47] "shadcn/ui." Accessed: Apr. 21, 2024. [Online]. Available: https://ui.shadcn.com/

[48] "Leaflet - a JavaScript library for interactive maps." Accessed: Apr. 18, 2024. [Online]. Available: https://leafletjs.com/

[49] "Rendering large lists in React: 5 methods with examples - LogRocket Blog." Accessed: Apr. 19, 2024. [Online]. Available: https://blog.logrocket.com/render-large-lists-react-5-methods-examples/