Instituto Politécnico
de Castelo Branco
Escola Superior
de Tecnologia

Protótipo de Solução para Redução do Desperdício Alimentar numa Cantina Institucional

Projeto II

Ana Raquel Ribeiro Correia, 20191266

Clara Alexandra Almeida Aidos, 52010216

Orientadores

Vasco Nuno da Gama de Jesus Soares

João Manuel Leitão Pires Caldeira

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

Junho 2024

Composição do júri

Presidente do júri

Doutor, Eduardo Sabina dos Santos Valente

Professor Adjunto, Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco

Vogais

Doutora, Ângela Cristina Marques de Oliveira

Professora Adjunta, Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco

Doutor, João Manuel Leitão Pires Caldeira

Professor Adjunto, Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco

Dedicatória

Ana Correia:

Dedico este projeto final de curso aos meus pais, por me proporcionarem a oportunidade de seguir os meus sonhos e de me tornar a pessoa que sou hoje.

À minha irmã, que através do seu entusiasmo pela área, teve um impacto profundo na minha escolha, sendo uma constante fonte de inspiração.

Ao meu namorado, que esteve ao meu lado em cada passo desta fase final do curso, e por me de motivar sempre para não desistir.

Quero também dedicar este projeto à minha colega, Clara, com o interesse e empenho dela tornou-se bastante mais simples levar a cabo este projeto.

A todas as pessoas que, de algum modo, caminharam ao meu lado e me apoiaram ao longo do meu percurso.

Clara Aidos:

Dedico este projeto inicialmente à minha colega Ana Correia. Conseguimos, em conjunto, acreditar neste projeto e, com esforço mútuo, superar desafios.

Quero também dedicar este trabalho à minha família: pai, mãe, irmã, sobrinho e cunhado. São a minha rede de suporte e, mesmo estando longe, oferecem o vosso apoio incondicional.

Às minhas amizades "caviar", aquelas que me ajudaram a superar tanto e que, mesmo quando eu não acredito que consigo, elas acreditam.

A todos aqueles que, de alguma forma, acompanham e me apoiam no meu percurso.

Agradecimentos

Queremos agradecer ao nosso orientador Professor Doutor Vasco Nuno da Gama de Jesus Soares e coorientador Professor Doutor João Manuel Leitão Pires Caldeira por nos convidarem a sermos orientadas por eles. Deram-nos liberdade para escolher um tema que nos motivava e traçaram o caminho para que chegássemos a bom porto. Obrigada.

Resumo

O desperdício alimentar tem ganho crescente atenção e debate, dada a sua implicação económica, ambiental, social e nutricional. Embora esteja presente em todas as etapas da cadeia de abastecimento alimentar, é nas fases finais de consumo, como nos domicílios e nos serviços de alimentação, que o problema se torna mais evidente. Este trabalho parte de um estudo anterior dos mesmos autores, apresentado no âmbito do Projeto I, que identificou a visão computacional como uma tecnologia adequada para identificar e quantificar o desperdício alimentar em cantinas institucionais. Partindo deste resultado, neste trabalho descreve-se a proposta e o processo de implementação de um protótipo demonstrador. Tem por base uma plataforma Raspberry Pi 4, um modelo Resnet-50 adaptado com o modelo Faster R-CNN, e um algoritmo para extração de características. Foi utilizado um *dataset* especialmente construído para dar resposta ao desafio de detetar tigelas de sopa e classificar o desperdício no seu consumo. Foi desenvolvida uma aplicação web para visualização dos dados recolhidos, oferecendo suporte à tomada de decisões para uma gestão mais eficiente do desperdício alimentar. O protótipo foi sujeito a validação e testes funcionais, e provou ser uma solução viável de baixo custo.

Palavras-chave

Desperdício alimentar, Cantinas Institucionais, Visão computacional, ResNet-50, Faster R-CNN, Extração de Características, Internet das Coisas, Protótipo.

Abstract

Food waste has gained increasing attention and debate, given its economic, environmental, social and nutritional implications. Although it is present at all stages of the food supply chain, it is in the final stages of consumption, such as households and food services, that the problem becomes most evident. This work builds on a previous study by the same authors, presented as part of Project I, which identified computer vision as a suitable technology for identifying and quantifying food waste in institutional canteens. Based on this result, this paper describes the proposal and implementation process of a demonstrator prototype. It is based on a Raspberry Pi 4 platform, a Resnet-50 model adapted with the Faster R-CNN model, and an algorithm for feature extracting. A specially built dataset was used to meet the challenge of detecting soup bowls and classifying waste in their consumption. A web application was developed to visualize the data collected, supporting decision-making for more efficient food waste management. The prototype was subjected to validation and functional tests, and proved to be a viable, low-cost solution.

Keywords

Food waste, Institutional canteens, Computer vision, ResNet-50, Faster R-CNN, Feature extraction, Internet of Things, Prototype.

Índice geral

Índice de figuras

Lista de tabelas

Lista de abreviaturas, siglas e acrónimos

AP - Average Precision

BLE - Bluetooth Low Energy

CNN - Convolutional Neural Network

DO - Digital Output

GPU - Graphics Processing Unit

HDR - High Dynamic Range

IoT - Internet of Things

mAP - Mean Average Precision

NVM - Node Version Manager

ODS - Objetivos de Desenvolvimento Sustentável

SDRAM - Synchronous dynamic random-access memory

SSD - Solid-state drive

TPU - Tensory Processing Unit

UI - User Interface

1 Introdução

Neste primeiro capítulo, pretende-se apresentar o âmbito deste projeto, definir o problema e identificar o objetivo a alcançar. Será também descrito o planeamento previsto para o desenvolvimento deste trabalho, bem como as principais contribuições que resultam do mesmo.

1.1 Âmbito

O desperdício alimentar é um problema global que se manifesta pela significativa perda de alimentos ao longo da cadeia de abastecimento [1]. Este fenómeno, especialmente visível nas fases de retalho e de consumo final [2], é consequência dos comportamentos adotados por retalhistas e consumidores, que resulta numa redução substancial na quantidade de alimentos consumíveis [3]. A magnitude deste problema é impressionante, já que cerca de um terço dos alimentos destinados ao consumo humano, equivalente a aproximadamente 1,3 mil milhões de toneladas por ano, é perdido ou desperdiçado a nível mundial [1]. A necessidade de abordar esta questão é clara, especialmente considerando a meta 12.3 dos Objetivos de Desenvolvimento Sustentável (ODS), que visa reduzir para metade, até 2030, o desperdício alimentar *per capita* tanto ao nível dos retalhistas e consumidores quanto ao longo das cadeias de produção e abastecimento [4].

Neste contexto, surge a motivação para desenvolver um protótipo de solução para reduzir o desperdício alimentar numa cantina institucional. O objetivo principal é identificar e avaliar tecnologias que permitam criar uma ferramenta capaz de reconhecer e analisar o desperdício numa cantina institucional. Esta ferramenta tem como objetivo ajudar a otimizar processos e a promover práticas mais conscientes, apoiando a tomada de decisões e contribuindo para uma gestão mais eficiente, o que se pretende que resulte numa redução do desperdício alimentar.

1.2 Definição do Problema

Na sociedade atual, o problema do desperdício alimentar surge como uma questão de interesse e discussão crescentes, dadas as implicações económicas, ambientais, sociais e nutricionais que acarreta. Embora o desperdício alimentar se manifeste em todas as fases da cadeia de abastecimento alimentar, nos países desenvolvidos, a sua ocorrência tende a ser mais evidente nas etapas finais de consumo, como nos lares e nos serviços de alimentação. O trabalho realizado no âmbito deste Projeto foca-se nas cantinas institucionais, onde o desperdício alimentar abrange tanto as refeições preparadas que não foram vendidas (ou seja, sobras), como os alimentos que permanecem nos pratos após o consumo da refeição (ou seja, restos).

A relevância do desperdício alimentar em cantinas institucionais reside não só no volume significativo de alimentos desperdiçados, mas também nas oportunidades que este contexto oferece para a implementação de soluções tecnológicas que visem a redução do desperdício. As cantinas institucionais, pela sua natureza padronizada e controlada, constituem um cenário ideal para a aplicação de tecnologias de monitorização e análise de dados.

Um dos principais desafios na quantificação e redução do desperdício alimentar é a falta de métodos automatizados e precisos que permitam monitorizar o desperdício de forma contínua e em tempo real. Neste contexto, o trabalho apresentado neste documento dá continuidade ao que foi realizado no âmbito do Projeto I, que culminou na submissão do artigo "Using Computer Vision for Reducing Food Waste in an Institutional Canteen" [5], atualmente em processo de revisão na revista Waste da editora MDPI. No âmbito desse trabalho, enquadrou-se o problema relacionado com o desperdício alimentar e realizou-se uma análise abrangente do estado da arte, com foco nas técnicas de visão computacional. Foi feita uma revisão crítica dos modelos de redes neuronais convolucionais (do inglês Convolutional Neural Network, CNN) e *datasets* comumente utilizados nesta área, procedendo-se à avaliação de desempenho do Inception-V3 e ResNet-50, juntamente com o *dataset* Food-101.

Dando continuidade ao trabalho realizado no Projeto I, o Projeto II tem como objetivo propor, implementar, testar e validar um protótipo demonstrador, de baixo custo, baseado em Internet das Coisas (do inglês Internet of Things, IoT) e técnicas de visão computacional, que permita quantificar o desperdício alimentar num contexto de uma cantina institucional, em particular da sopa. A escolha do foco no desperdício alimentar incidir na sopa deve-se às dificuldades em detetar o desperdício num segundo prato de refeição, uma vez que as pessoas tendem a espalhar os restos. Além disso, o peso do prato pode não ser um bom indicador, pois podem ser adicionadas cascas de fruta, por exemplo, o que poderia condicionar os resultados. O trabalho aqui realizado para a sopa pode ser aplicado de igual forma a sobremesas.

1.3 Objetivos

Este projeto tem por objetivo principal propor e avaliar uma solução que possibilite quantificar o desperdício alimentar da sopa numa cantina institucional. Para atingir este objetivo principal, foram identificados os seguintes objetivos intermédios:

- Análise do estado da arte e levantamento tecnológico;
- Análise de requisitos;
- Proposta e implementação de protótipo demonstrador;
- Demonstração e testes funcionais;
- Avaliação de desempenho.

Dado que os dois primeiros objetivos intermédios foram atingidos em Projeto I, o foco de Projeto II está direcionado para os restantes.

1.4 Cronograma

Na Tabela 1 é exibido o cronograma das tarefas parciais a serem executadas para alcançar os objetivos intermédios de Projeto II.

- **Tarefa Nº1 – Proposta e implementação de protótipo demonstrador**: desenho e implementação de protótipo demonstrador, de baixo custo, para identificar e quantificar desperdício de sopa, num contexto de uma cantina institucional.
- **Tarefa Nº2 – Implementação de *scripts* no servidor**: elaboração de um *script* que realiza a deteção de tigelas nas imagens utilizando um modelo CNN e um *script* que faz a extração de características das imagens para quantificação do desperdício.
- **Tarefa Nº3 – Demonstração, testes funcionais e avaliação de desempenho**: testes abrangentes das funcionalidades do protótipo, executados em contexto relevante, para verificar se atende às necessidades identificadas, para avaliar o seu desempenho, e para identificar melhorias a aplicar.
- **Tarefa Nº4 – Redação do relatório**: relatório produzido para documentar o trabalho realizado ao longo do Projeto II.
- **Tarefa Nº5 – Redação do artigo**: artigo científico que tem por base o relatório, submetido a uma revista científica internacional, relevante na área, apresentando o trabalho realizado, com o objetivo de disseminar os resultados para a comunidade académica e científica.

Tabela 1 - Cronograma de tarefas.

Ano	**2024**				
Mês / **Tarefa**	Fevereiro	Março	Abril	Maio	Junho
1					
2					
3					
4					
5					

1.5 Contribuições

O trabalho apresentado em Projeto II dá continuidade ao esforço iniciado em Projeto I, que resultou no artigo "*Using Computer Vision for Reducing Food Waste in an Institutional Canteen*" que se encontra em processo de revisão na revista *Waste* da editora MDPI. Considera-se que as principais contribuições do trabalho realizado no âmbito de Projeto II são:

- Proposta de arquitetura para o protótipo demonstrador, de baixo custo, baseada em Internet das Coisas e visão computacional;
- Descrição do processo de implementação do protótipo, incluindo os seus componentes de hardware e software;

- Testes e demonstração do conceito num ambiente de teste controlado;
- Avaliação do desempenho do protótipo funcional;
- Identificação de pontos de trabalho em aberto.

1.6 Organização do Documento

Este documento está organizado em quatro capítulos.

O primeiro capítulo introduz o tema, define o problema, justifica a sua importância e estabelece os objetivos, em conjunto com o cronograma/plano de trabalhos, e destaca as contribuições resultantes deste trabalho.

O segundo capítulo apresenta o protótipo demonstrador desenvolvido no âmbito deste trabalho. Descreve e justifica a arquitetura adotada, bem como as componentes de hardware e software, e o processo de implementação.

O terceiro capítulo descreve os testes conduzidos para validação e prova de conceito do protótipo proposto.

Por fim, o quarto capítulo apresenta as conclusões do trabalho, destacando os resultados obtidos e discute possíveis direções para trabalhos futuros.

2 Protótipo

Este capítulo apresenta o protótipo desenvolvido no contexto deste trabalho. A primeira secção descreve a arquitetura que foi definida e as decisões que foram tomadas na especificação da mesma. A segunda e terceira secções detalham o processo de implementação do hardware e software do protótipo.

2.1 Arquitetura

A Figura 1 apresenta a arquitetura do protótipo da solução desenvolvida, que permite quantificar o desperdício alimentar de sopa numa cantina institucional, no processo de entrega do tabuleiro, após consumo da refeição.

A componente física é composta por uma câmara e um sensor infravermelhos ligado a um Raspberry Pi [6], com o objetivo de capturar imagens de tabuleiros. As imagens capturadas não são guardadas localmente neste dispositivo, dadas as suas limitações de processamento e armazenamento. Pelo que, a imagem do tabuleiro é transferida para um servidor local Ubuntu [7], que está a ser executado numa máquina virtual.

Neste servidor, foi criado um *script* Python [8] que tem por objetivo observar a diretoria de destino das imagens, e de seguida colocar o modelo de deteção e classificação em execução. Em primeiro lugar tenta detetar na imagem capturada a presença de uma tigela, utilizando os modelos de visão computacional Resnet-50 e Faster R-CNN, estudados em Projeto I. Depois, a partir da imagem dessa tigela, realiza a extração de características e calcula o volume de sopa remanescente (i.e., desperdiçado). Este servidor possui uma base de dados MySQL [9], implementada num container Docker [10], que vai armazenar os dados recolhidos na etapa anterior, isto é, o volume desperdiçado associando uma data correspondente.

Para possibilitar a integração e consulta da informação recolhida, bem como a gestão e o apoio à tomada de decisões, foi desenvolvida uma aplicação *web* baseada em microsserviços, constituída por *containers* individuais que comunicam entre si. Todos esses serviços, desde a base de dados até aos microsserviços, estão implementados em *containers* Docker. O microsserviço Frontend é responsável pela interface do utilizador, permitindo a consulta dos dados armazenados. O microsserviço Backend, interage diretamente com a base de dados MySQL. O microsserviço Eureka atua como um serviço de descoberta de microsserviços, facilitando a comunicação entre eles.

A escolha da arquitetura baseada em microsserviços é uma decisão a pensar no futuro, pois oferece várias vantagens importantes. Os microsserviços aumentam a resiliência, garantindo que falhas em componentes não afetem todo o sistema. Facilitam a escalabilidade, permitindo que cada serviço cresça conforme a necessidade. Melhoram a eficiência dos testes ao permitir a verificação isolada de cada serviço. Além disso, possibilitam o uso de múltiplas tecnologias, aumentando a adaptabilidade e inovação contínua [11].

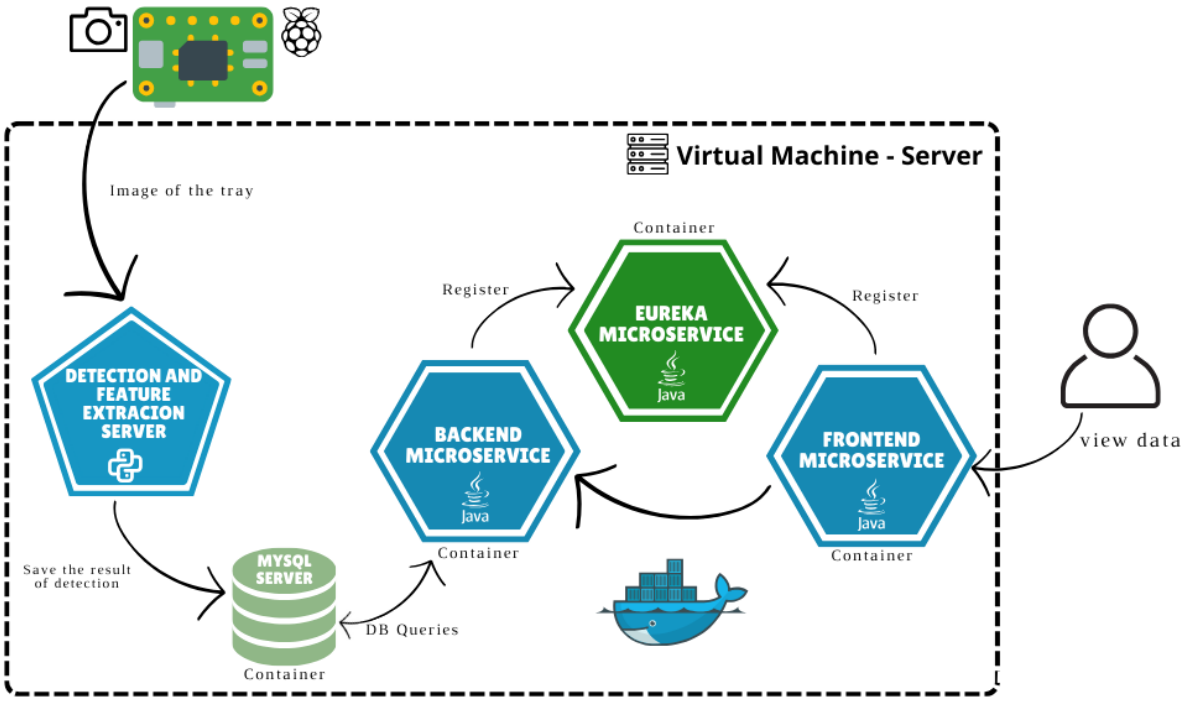

Figura 1- Arquitetura do protótipo.

2.2 Componente de Hardware

Esta subsecção descreve as características dos componentes de *hardware* utilizados na solução implementada.

2.2.1 Raspberry Pi e periféricos

O Raspberry Pi 5 [6] é um dispositivo de computação altamente versátil, concebido para proporcionar um alto desempenho numa variedade de aplicações. Está equipado com um processador *quad-core Broadcom* BCM2712 Arm Cortex-A76 a operar a 2.4GHz. Para facilitar a integração em redes e sistemas existentes, incui dual-band 802.11ac Wi-Fi, Bluetooth 5.0 / *Bluetooth Low Energy* (BLE) e *Gigabit Ethernet* com suporte a PoE+. Tem capacidade de expansão através de uma interface PCIe 2.0 x1 e possibilita a ligação de periféricos rápidos, como unidades *Node Version Manager* (NVM) e *Solid-state drive* (SSD). Com opções de memória LPDDR4X-4267 *Synchronous dynamic random-access memory* (SDRAM) de 4GB ou 8GB e um *slot* para cartão microSD de alta velocidade, nesta solução utilizou-se um microSD Kingston [12] de 32GB com velocidade de 100MB/s.

A câmara utilizada foi uma Raspberry Pi *Camera Module 3* [13]. É uma câmara compacta concebida para Raspberry Pi, equipada com um sensor Sony IMX708 de 12 megapixels e um atuador de foco controlado por I2C. Este módulo apresenta um sensor grande com uma diagonal de 7,4 mm e um ângulo de visão diagonal de 75 graus. Além disso, suporta um modo *High Dynamic Range* (HDR) para garantir uma qualidade de imagem superior e possui um foco

automático ultrarrápido, juntamente com uma biblioteca de comandos de *software* para controlo preciso.

Para a deteção do tabuleiro utilizou-se um sensor de reflexão de infravermelhos da Waveshare ST188 [14]. É um transmissor-refletor de infravermelhos que possui um comparador de voltagem LM393. Apresenta uma sensibilidade ajustável e um indicador de saída de sinal. As suas especificações incluem uma potência de 3.0V a 5.3V, dimensões de 25mm por 15.9mm e buracos de montagem de 2.0mm.

A Figura 2 ilustra a integração do Raspberry Pi 5 e dos seus periféricos.

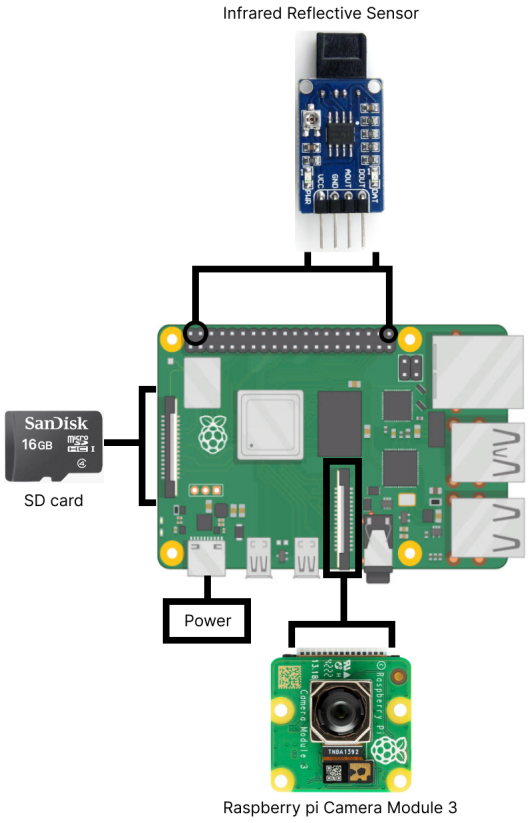

Figura 2 - Integração dos componentes de *hardware*.

Descrevem-se de seguida os passos executados para integrar os componentes de *hardware*. Inicialmente, é necessário inserir o cartão SD no *slot* apropriado do dispositivo. Em seguida, realiza-se a ligação da câmara ao Raspberry Pi, inserindo cuidadosamente o cabo flexível no conetor designado, assegurando que os contactos estejam voltados para a porta HDMI, conforme ilustrado nos tutoriais disponíveis em [15] e [16]. Posteriormente, procede-se à ligação do sensor infravermelhos, utilizando a saída digital (do inglês, Digital Output, DO) para detetar a presença ou ausência de reflexão e ajustando a sensibilidade conforme necessário. O sensor é ligado ao GPIO4 do Raspberry Pi, sendo o pino de alimentação de 5V do

sensor ligado ao pino de 5V do Raspberry Pi, e o pino de terra (*Ground*) do sensor ao pino de terra do Raspberry Pi, que na Figura 3 correspondem aos números 7 (GPIO4), 4 (5V) e 6 (*Ground*) respetivamente [17]. Além disso, o potenciómetro do sensor é ajustado para calibrar a sensibilidade à reflexão infravermelha. Estas orientações foram obtidas nos fóruns de suporte técnico em [18], [19], [20]. Por fim, efetua-se a ligação do cabo de alimentação para fornecer energia ao sistema.

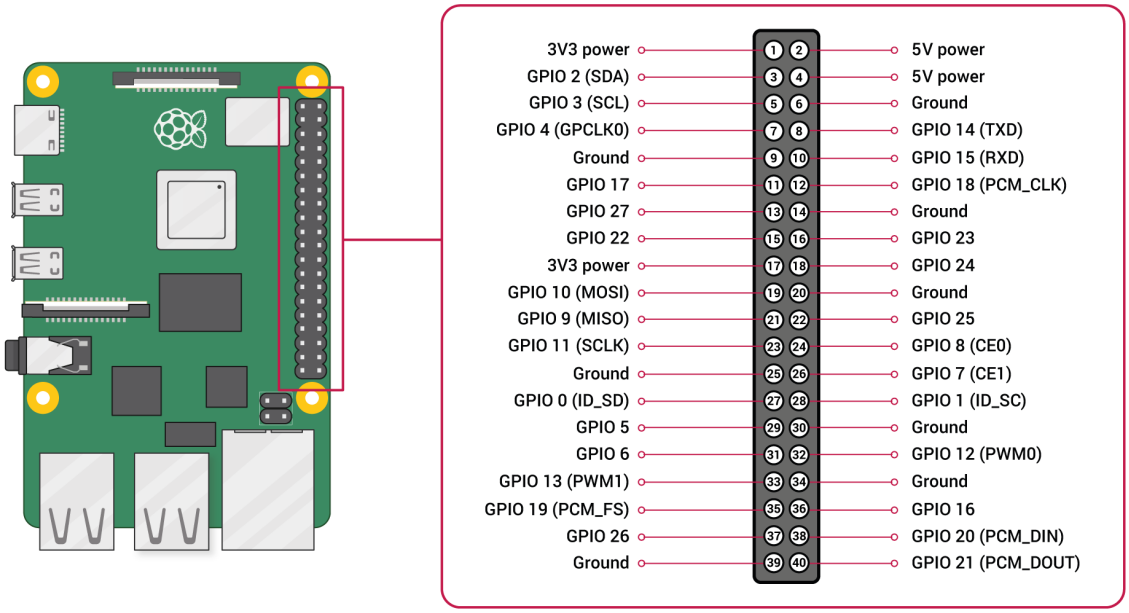

Figura 3 - Diagrama de pinos GPIO do Raspberry Pi Fonte: [17].

A Figura 4 apresenta a integração da Raspberry Pi *Camera Module* 3 e do sensor infravermelhos com o Raspberry Pi 5.

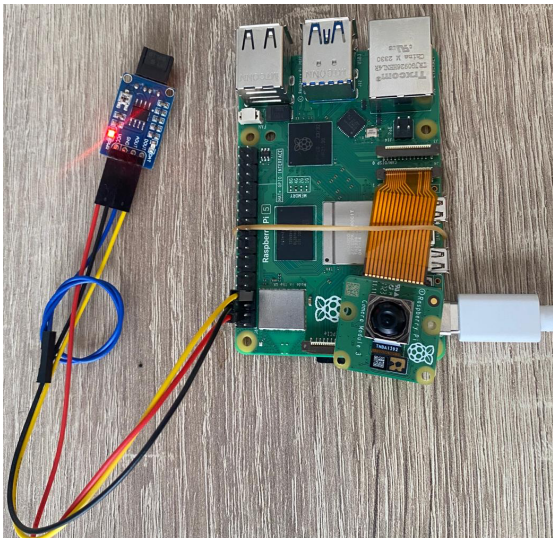

Figura 4 - Integração dos sensores no Raspberry Pi 5.

Na Figura 5 é possível observar a bancada de testes e a componente física do protótipo, simulando o cenário de aplicação do projeto.

Este cenário é uma cantina institucional, onde as refeições são servidas em tabuleiros padronizados, tipicamente compostos por copo, prato e tigela de sopa. No final das refeições, os tabuleiros com os restos dos alimentos são entregues numa janela de recolha com ligação à cozinha, momento em que as imagens são capturadas pelo protótipo.

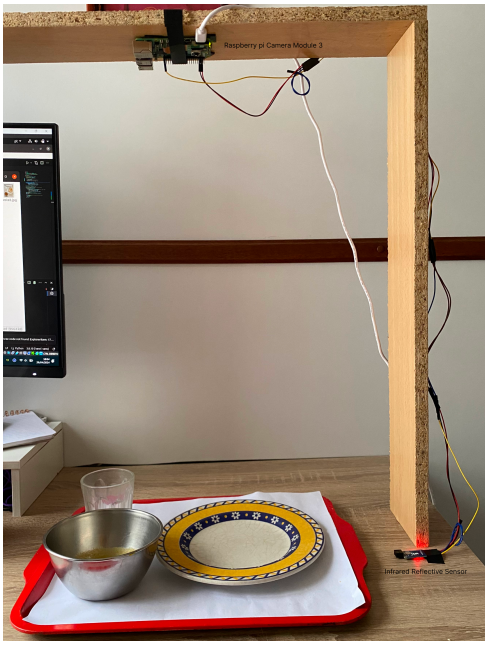

Figura 5 - Bancada de teste do protótipo.

2.2.2 Servidor

Como descrito anteriormente, o Raspberry Pi comunica com um servidor local. Nesta implementação, foi utilizada uma máquina virtual criada com o Oracle VM VirtualBox [21] para atuar como servidor. A máquina virtual opera com o sistema operativo Ubuntu 20.04. Os componentes de hardware configurados para esta máquina virtual incluem 9037 MB de memória RAM, 4 CPUs, e 55 GB de armazenamento em disco rígido virtual do tipo SATA, localizado no arquivo *server-project.vdi*. Esses componentes de *hardware* são virtuais, configuráveis conforme necessário, e permitem a execução eficiente do servidor.

2.3 Componente de Software

Esta subsecção detalha os vários elementos de *software* desenvolvidos e configurados para implementação do protótipo. Começando pela criação do *dataset*, passando pelo treino e

validação dos modelos, configurações dos dispositivos IoT, funcionalidades do servidor local, até ao desenvolvimento de uma aplicação web para visualização dos resultados.

2.3.1 Dataset

Para a criação do *dataset* utilizado no treino do modelo, foram recolhidas imagens dos tabuleiros da cantina da Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco [22]. O *dataset* é composto por 270 imagens de tabuleiros. Ressalva-se que as imagens são bastante semelhantes entre si, variando apenas a posição da tigela no tabuleiro e a quantidade de sopa na tigela. Mas, o objetivo é que apenas esse tipo específico de tigela seja detetado, uma vez que este *dataset* foi construído para funcionar em cantinas institucionais que possuam tigelas desse género, como é o caso da cantina da Escola Superior de Tecnologia. Isto visa evitar, por exemplo, que tigelas de sobremesa sejam identificadas erradamente, pois o propósito deste projeto é detetar tigelas de sopa para calcular o seu desperdício. A Figura 6 apresenta três exemplos de imagens recolhidas que fazem parte do *dataset* criado.

Figura 6 - Exemplos de imagens que compõem o *dataset*.

A plataforma Roboflow [23] foi utilizada para a construção do *dataset*. Inicialmente, foi criado um projeto selecionando o tipo "Classification". Em seguida, fez-se o *upload* das imagens recolhidas. Utilizou-se a opção "Manual Labeling" para anotar manualmente as imagens. No processo de anotação, foi criada a classe "tigela". Após esta etapa, selecionou-se o "Start Annotate" e, utilizando uma "Bounding Box", foi desenhado um contorno em volta da tigela de sopa, conforme ilustrado na Figura 7. Este procedimento foi repetido para todas as imagens capturadas.

Figura 7 - Anotação de imagens no Roboflow.

Depois de todas as imagens terem sido anotadas, procedeu-se à exportação do *dataset* através da opção "Export Dataset". Antes da exportação, foram adicionadas técnicas de "data augmentation", como "flip", "rotation", "blur" e "noise", bem como pré-processamento, incluindo "resize" e "auto-orient", para aumentar o número e a variedade de imagens.

Na Figura 8, é apresentada a composição do *dataset*. A plataforma Roboflow gere a composição e divisão do *dataset*. As imagens foram carregadas e anotadas, mas a plataforma decide quantas devem ser utilizadas para treino, teste e validação. Neste caso, estão incluídas 252 imagens para treino, 11 imagens para validação e 7 imagens para teste. Este *dataset* depois de exportado será utilizado no treino do modelo CNN, conforme descrito de seguida. O dataset está disponível para download em [24].

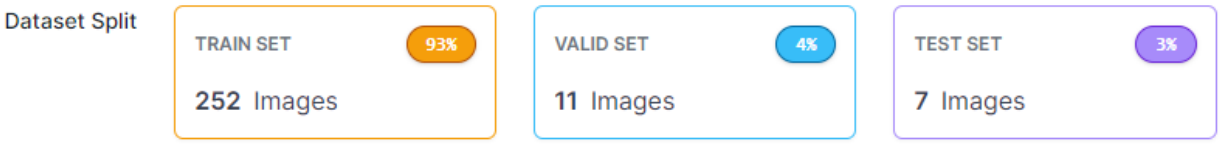

Figura 8 - Divisão do *dataset*.

2.3.2 Modelos Resnet-50 e Faster R-CNN

As CNNs, ou redes neurais convolucionais, são especificamente projetadas para processar dados estruturados em grelha como as imagens e têm-se destacado em tarefas de visão computacional, como classificação, deteção e segmentação de imagens. Os modelos baseados em CNNs são constituídos por camadas convolucionais (*convolution*), de agrupamento (*pooling*) e totalmente ligadas (*fully connected*), que extraem e representam características da

imagem de entrada [25]. Na arquitetura de uma CNN, representada na Figura 9, a camada convolucional é responsável por extrair características da imagem utilizando filtros treináveis, que identificam padrões como bordas, texturas e formas. Os mapas de características resultantes são passados pela camada ReLU, que introduz não-linearidade, permitindo que a rede aprenda tarefas complexas [25], [26], [27].

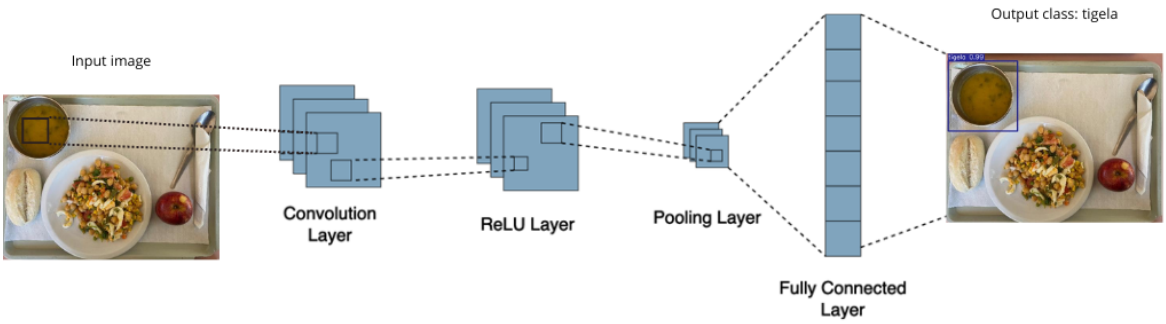

Figura 9 - Arquitetura típica de uma CNN.

Na sequência do trabalho desenvolvido em Projeto I e nas conclusões observadas, optou-se por combinar dois modelos: o Resnet-50 e o Faster R-CNN. Estes dois modelos combinam duas abordagens distintas no campo da visão computacional e *deep learning* para deteção de objetos. O Resnet-50 é uma arquitetura de rede neural profunda conhecida como "rede residual", projetada para superar desafios de treino em redes profundas [28]. Por sua vez, o Faster R-CNN descreve uma abordagem que divide a imagem em regiões e utiliza uma CNN para identificar objetos nessas regiões. Ao unir estas duas abordagens, como ilustrado na Figura 10, é possível aproveitar a capacidade do Resnet-50 para extrair características complexas de imagens, enquanto o Faster R-CNN lida com a deteção dos objetos.

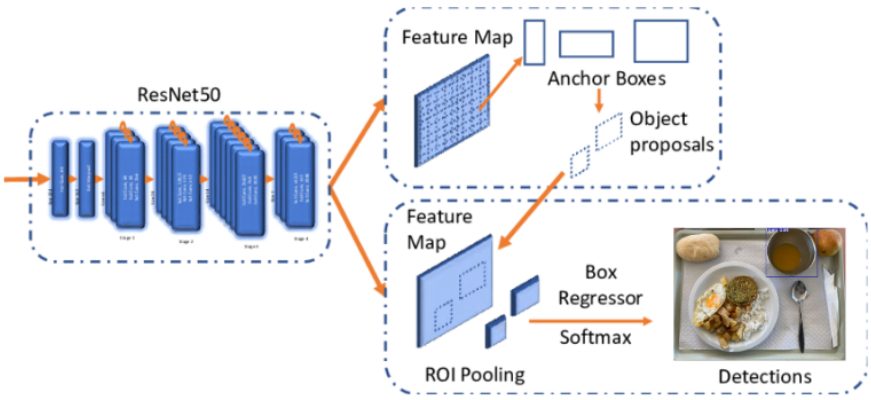

Figura 10 - Arquitetura dos modelos Resnet-50 e Faster R-CNN.

O modelo foi treinado na plataforma Google Colab [29], utilizando um *notebook* disponível em [30]. O *notebook*, com código *Python*, foi adaptado para permitir a importação do *dataset* elaborado para este propósito, através da utilização da API do Roboflow [23]. A plataforma Google Colab, utilizada também em Projeto I, oferece recursos de *hardware* gratuitos, como como Unidades de processamento gráfico (do Inglês, Graphics processing units, GPUs) e Unidades de Processamento Tensorial (do Inglês, Tensory Processing Units, TPU), suporte a diversas linguagens e integração com o Google Drive [31] e o GitHub [32]. A máquina disponibilizada pela plataforma no plano gratuito tem as seguintes características: placa gráfica NVIDIA T4 com 16 GB de VRAM e 13 GB de RAM para o sistema.

Para além do treino do modelo, este *notebook* também permite realizar a validação do mesmo, que consiste em analisar as imagens de validação do *dataset* para avaliar o desempenho. Na Figura 11, é exibida uma imagem exemplo dos resultados de validação efetuados diretamente no próprio *notebook*, mostrando que o modelo identifica com precisão a localização da tigela.

Figura 11 - Resultados de validação com imagens do dataset.

Foram realizados vários treinos do modelo com diferentes versões do *dataset*, e adaptado o número de épocas de treino para evitar o *overfitting*. O *overfitting* representa um desafio significativo durante o treino de CNNs. Ocorre quando um modelo é treinado em excesso, ao ponto de memorizar detalhes específicos dos dados de treino. Como resultado, o modelo apresenta um desempenho excelente nos dados de treino, mas falha ao lidar com novos dados [33].

O modelo utilizado foi treinado por 19 épocas. A decisão de parar o treino foi tomada com base em vários fatores. Quando se verifica que o desempenho do modelo no *dataset* não está a melhorar, isto é, está estagnado ou está a piorar, indicando um aumento de erros, o treino é interrompido [34]. Neste cenário, onde todas as tigelas são consideradas iguais e, portanto, as

imagens dos tabuleiros são muito semelhantes, prolongar o treino por demasiadas épocas não iria melhorar o desempenho do modelo e poderia até piorar.

No que diz respeito às métricas de desempenho na avaliação do modelo, foram consideradas o *mean average precision* (mAP) e o *loss*. O *average precision* (AP) é uma métrica comumente utilizada no contexto da classificação binária e da recuperação de informação para resumir a curva precisão-recuperação. Fornece um valor único que representa a qualidade dos resultados de recuperação classificados para uma classe ou categoria específica, especialmente em tarefas como a deteção de objetos [35]. O cálculo é realizado de acordo com a Equação (1).

$$AP = \sum_{k=0}^{k=n-1} [Recalls(k) - Recalls(k+1)] * Precisions(k) \quad (1)$$

O mAP é obtido pela média das precisões em diferentes níveis de *recall* (avaliação da eficácia do modelo na deteção) [36]. Esta métrica é particularmente importante, pois não é influenciada pelo tamanho dos objetos, o que permite uma comparação eficaz entre modelos. É calculado com base na Equação (2), onde se obtém a média dos AP das classes consideradas.

$$mAP = \frac{1}{k}\sum_{i}^{k} AP_i \quad (2)$$

Por fim, o *loss*, durante o treino, reflete como o modelo se está a ajustar ao *dataset*, esperando-se que diminua à medida que as épocas avançam [37]. Contudo, um valor de *loss* muito baixo não garante um bom desempenho em novos dados, pois pode indicar um ajuste excessivo aos dados de treino. A fórmula para calcular o *loss* é apresentada na Equação (3).

$$Loss = -\frac{1}{Output\ Size} \sum_{i=1}^{Output\ size} y_i * log\, \hat{y}_i + (1 - y_i) * log\ (1 - \hat{y}_i) \quad (3)$$

Na Figura 12 é possível observar que, nas primeiras épocas, o mAP é significativamente elevado. Isto deve-se ao facto de se trabalhar num ambiente muito controlado, onde todas as

imagens do *dataset* são provenientes de uma única cantina, resultando numa grande semelhança entre elas. Por conseguinte, os resultados são extremamente positivos, sendo que a partir da época 5 o mAP é de 1.0.

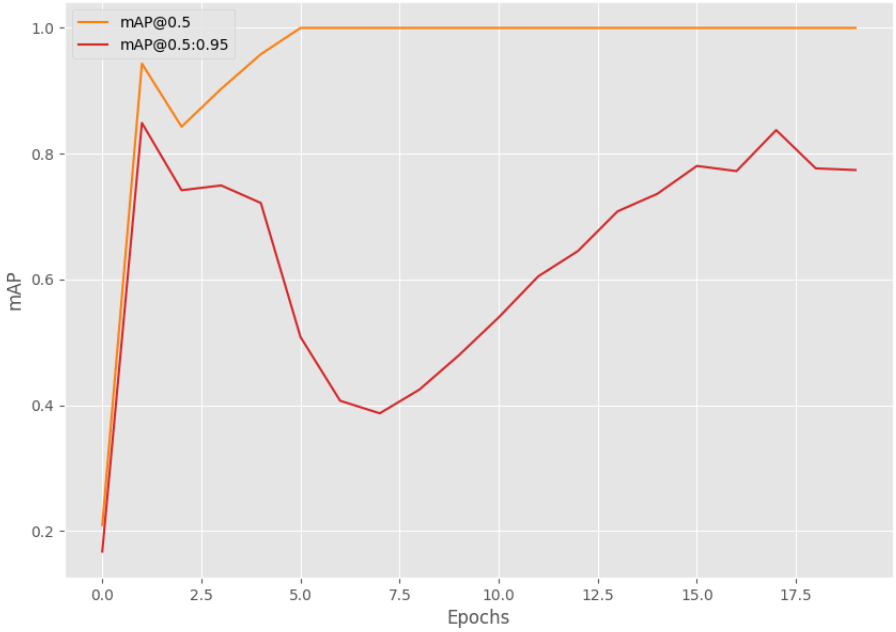

Figura 12 - Resultados de mAP no treino do Resnet-50+Faster R-CNN.

Em relação ao *loss*, pode observar-se na Figura 13 que o seu valor inicial não é excessivamente elevado. Além disso, ao longo das épocas diminui gradualmente, estabilizando-se num valor de 0,0359.

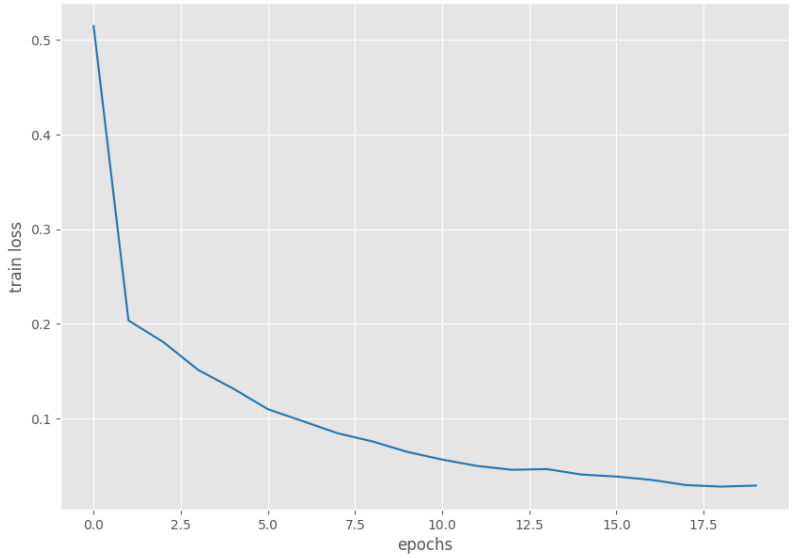

Figura 13 - Resultados de *loss* no treino do Resnet-50+Faster R-CNN.

2.3.3 Configurações de Software nos Dispositivos IoT

A primeira tarefa realizada no Raspberry Pi foi a instalação do sistema operativo, neste caso, o Raspberry Pi OS disponível em [38]. Como descrito anteriormente, foram instalados os seus periféricos, isto é, a Raspberry Pi *Camera Module* 3 [13] e o sensor infravermelhos ST188 [14].

Seguidamente foi desenvolvido, um *script* Python que permite a comunicação entre o Raspberry e o servidor e a integração do sensor de infravermelho para despoletar a ação de capturar a imagem assim que seja detetado um tabuleiro. Na Figura 14 pode observar-se o fluxograma que ilustra o processo executado por este *script* para capturar e transferir as imagens. O processo começa com a inicialização do *script*, onde a câmara e o sensor infravermelhos são configurados para a operação.

Para a configuração da câmara foi utilizada a biblioteca Python Picam2 [39]. Configurou-se o tamanho das imagens como 1920px por 1080px. Quando o *script* é executado inicia-se uma *preview* da câmara, permitindo a visualização em tempo real do que está a ser captado. Quanto às configurações do sensor infravermelhos, a biblioteca Python GPIOZero [40] foi utilizada para a manipular os pinos digitais.

De seguida, foram definidas as variáveis necessárias para a conexão com o servidor remoto, incluindo o endereço IP, o nome de utilizador do servidor e respetiva password e o caminho da diretoria destino no servidor, onde serão armazenadas as imagens capturadas.

Inicialmente, o *script* verifica se o sensor de infravermelhos está ativo. Se o sensor não estiver ativo, o *input* é 0 e o *script* continua a verificar até que o sensor seja ativo. Quando o sensor é ativo, o *input* é alterado para 1 e o *script* aguarda 1 segundo antes de capturar a imagem, para evitar que a mesma fique desfocada. Para que não sejam capturadas imagens repetidas do mesmo tabuleiro, apenas é capturada uma nova imagem se o valor de *input*

anterior for diferente de 1. Ou seja, se um tabuleiro permanecer por um longo período de tempo em frente ao sensor, será capturada apenas uma imagem. Quando o *input* for alterado para 0, indicando a remoção do tabuleiro anterior, e o sensor for ativo novamente pela presença de um novo tabuleiro, será capturada uma nova imagem.

Após a captura da imagem, esta é transferida para o servidor local utilizando a biblioteca Python Paramiko [41], [42]. No cenário de testes, o servidor e o Raspberry Pi estão na mesma rede local, pelo que a comunicação entre ambos é feita através de SSH e utilizando endereços IP privados.

Posteriormente, o *script* verifica se a transferência da imagem foi bem-sucedida. A mesma nunca é guardada localmente no Raspberry, para evitar problemas de armazenamento. Se a imagem for transferida com sucesso, o *input* é redefinido para 0. Se a transferência não for bem-sucedida, é exibida uma mensagem de erro e o *input* é também redefinido para 0, voltando ao passo inicial de verificar se o sensor está ou não ativo.

Este processo é repetido continuamente, permitindo a captura e transferência de imagens sempre que o sensor de infravermelhos detetar a presença de um tabuleiro.

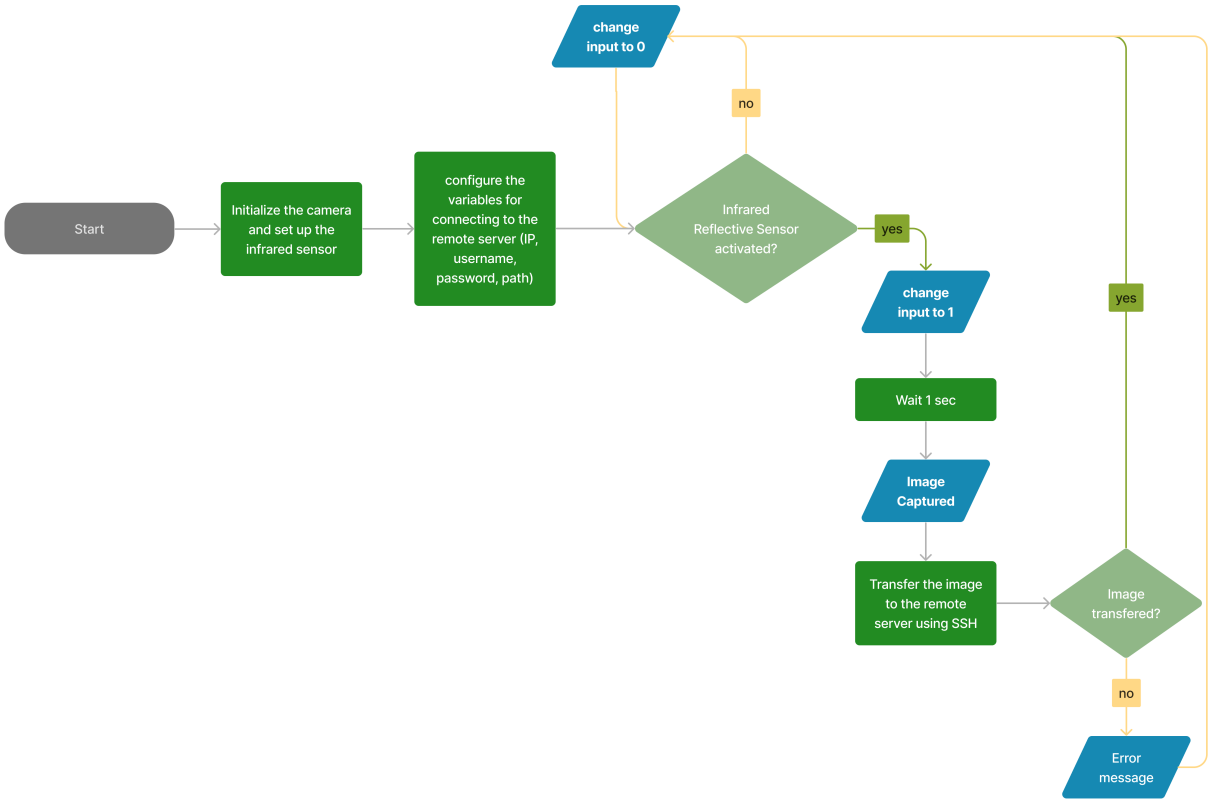

Figura 14 - Fluxograma do *script* Python para captura e transferência de imagens no Raspberry Pi.

2.3.4 Servidor Local Ubuntu

O servidor local está alojado numa máquina virtual, e desempenha diversas funções. Inicialmente este foi criado com o propósito de receber a imagens vindas do Raspberry e executar os modelos de deteção e classificação bem como a extração de características. No entanto, para efeitos de protótipo, ele também tem a função de hospedar tanto a base de dados como a aplicação *web*, utilizando o Docker [10].

Para automatizar o processo, foi criado um *script* Python que envolve todas as funções necessárias para realizar a deteção e classificação, realizar a extração de características, o respetivo cálculo do volume e ainda a inserção dos dados na base de dados. A Figura 15 apresenta um fluxograma que ilustra esse processo.

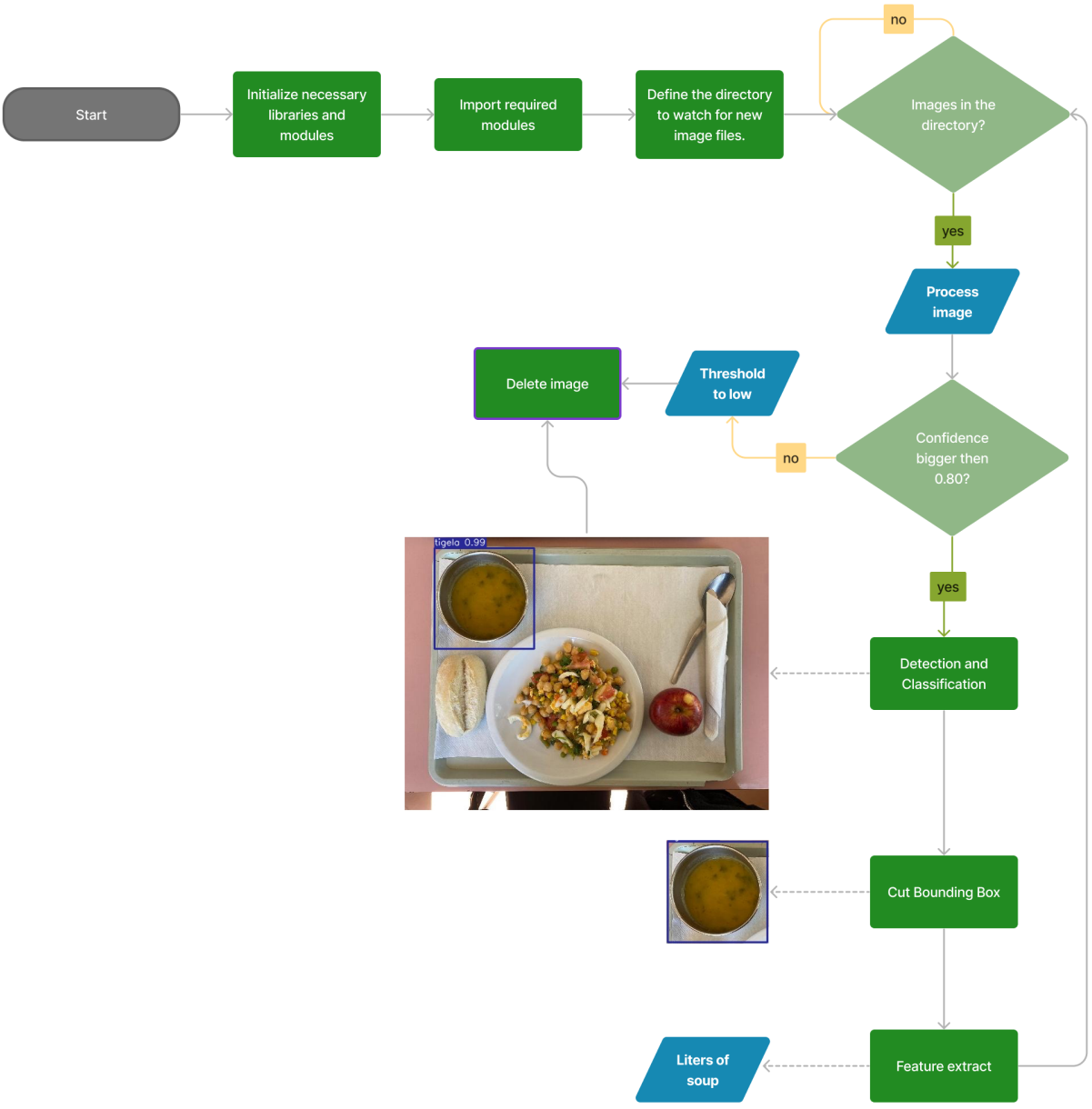

Figura 15 - Fluxograma do *script* Python principal para processamento das imagens no servidor.

A diretoria onde as imagens vindas do Raspberry são armazenadas está sob observação, o que significa que quando uma imagem chega, ela é imediatamente processada. Essa funcionalidade foi implementada utilizando a biblioteca Python Watchdog Observer [43].

A primeira etapa do processamento envolve a deteção e classificação de tigelas nas imagens. Para isso, foi adaptado um projeto disponível *online* do mesmo autor do *notebook* utilizado para treinar o modelo [30]. Este *script* de deteção, explicado no fluxograma da Figura 16, utiliza bibliotecas Python como Pytorch [44], Numpy [45], Cv2 [45], OS [46], YAML [47] e Matplotlib [48].

A etapa "load pre-trained detection model", itera sobre as imagens presentes na diretoria e realiza a deteção e classificação de tigelas em cada uma delas. Caso a tigela seja detetada com um *threshold* superior a 0,8, a imagem é recortada tendo por limites a *bounding box* e é executada a próxima etapa: "feature extraction". Se não foi detetada nenhuma tigela a imagem é eliminada. O valor estabelecido para o *threshold* deve ser considerado aceitável, significando que o modelo só considerará verdadeiramente uma tigela se tiver uma confiança superior a 80%. Noutras palavras, se o modelo detetar uma tigela, mas com uma confiança de 10%, a mesma não será considerada e as *bounding boxes* não serão desenhadas.

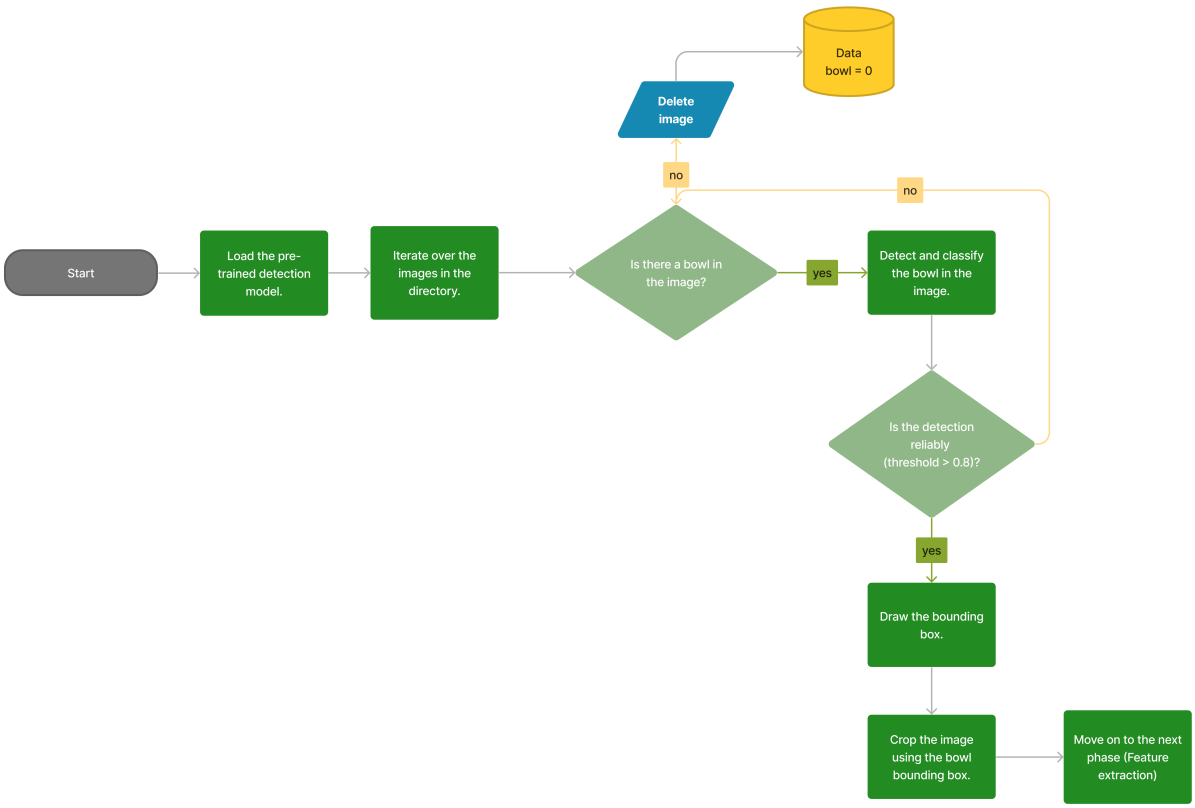

Figura 16 - Fluxograma do *script* Python para a deteção e classificação de tigelas nas imagens.

A etapa seguinte refere-se à "feature extraction". É onde ocorre o cálculo do volume de sopa desperdiçado, ou seja, o volume que permanece na tigela quando é capturada a imagem do tabuleiro. Para implementar esta funcionalidade, utilizaram-se as bibliotecas Python Cv2, Numpy, Copy [49] e Math [50].

O processo de extração de características, ilustrado na Figura 17, começa com o carregamento da imagem e a sua conversão para escala de cinza (*grayscale*). Seguidamente, aplica-se um "gaussian blur" [51] para reduzir o ruído e facilitar a deteção de bordas. A "HoughCircles function" é usada para deteção de círculos e é realizada usando o algoritmo de Hough [52], que identifica os círculos na imagem. Após a deteção dos círculos representando a tigela e a sopa, aplicam-se máscaras para filtrar as áreas laranja da imagem, que representam a sopa. O maior círculo detetado corresponde ao raio da tigela, enquanto o menor círculo representa a sopa na tigela.

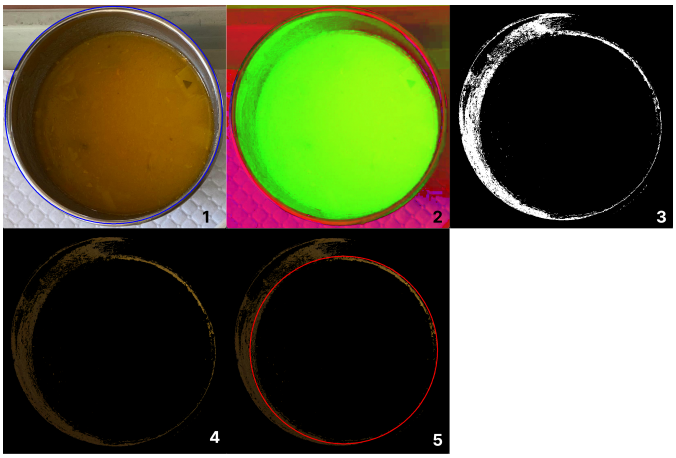

Figura 17 – Processo de tratamento de imagem para deteção de círculos.

A "calculate soup volume function" usada para calcular o volume da sopa utiliza os raios da tigela e da sopa calculados anteriormente. Primeiro, convertendo os valores de pixels para centímetros e ajustando o raio menor da sopa de acordo com a escala da tigela. Em seguida, é calculado o volume utilizando a fórmula do volume de um cone truncado [53], conforme apresentado na Equação (4). A função retorna o total de litros de sopa na tigela (i.e., mililitros).

$$\mathrm{V} = \frac{1}{3} * \pi * h * (r^2 + r * R + R^2),$$

Where:

- *R is the radius of the base of the original cone (bottom surface);*
- *r is the radius of the top surface;*
- *h is the height of our truncated cone.*

(4)

Embora este método forneça uma estimativa do volume da sopa, é importante reconhecer as suas limitações e considerar possíveis fatores de erro, como imprecisões na deteção de círculos e na conversão de pixels para centímetros. Visto que a tigela não representa efetivamente um cone truncado, devido às paredes da tigela não serem retas, deve ser considerada também uma margem de erro no cálculo do volume da sopa. No entanto, este processo oferece uma abordagem automatizada e suficiente para calcular o volume da sopa presente na tigela. O fluxograma da Figura 18, representa o processo da extração de características implementado descrito.

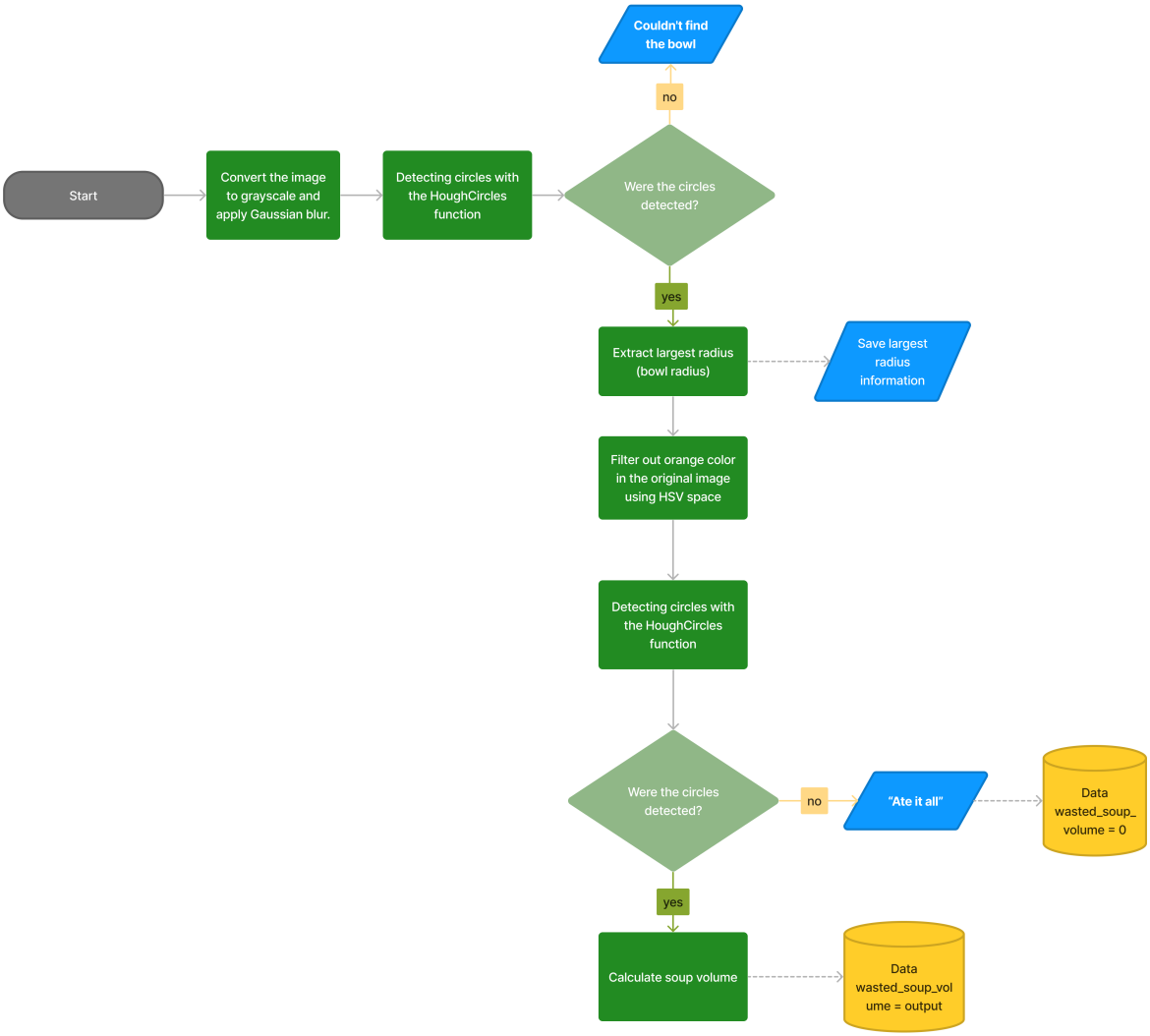

Figura 18 - Fluxograma do *script* Python para a extração de características.

Importa referir que, para melhorar a estrutura e clareza do código, cada fase foi implementada num *script* Python separado, sendo posteriormente chamados num *script* principal "main" pela ordem descrita anteriormente. Este *script* principal é executado num

ciclo contínuo juntamente com a observação da diretoria. Quando uma imagem chega à pasta, os processos dentro do ciclo são executados.

Por fim, existe ainda uma última fase, que consiste na comunicação destes *scripts* com a base de dados MySQL, a fim de armazenar os dados relativos ao cálculo do volume. Assim, foi criado outro *script* que permite realizar a conexão à base de dados, bem como a inserção dos dados na mesma. Este *script* é chamado após a realização do cálculo do volume. Importa salientar que, para fins estatísticos, foram também armazenados dados relativos aos tabuleiros que não continham tigelas, com o objetivo de comparar a quantidade de consumidores que comem ou não sopa. Se não existir uma tigela no tabuleiro, o volume de sopa desperdiçado inserido é igual a zero.

2.3.5 Base de Dados e Aplicação Web

A base de dados, como foi mencionado anteriormente, foi implementada num *container* Docker através da utilização de uma imagem do MySQL (versão 8.4.0) [54] disponível no Docker Hub [55].

A base de dados é constituída por duas tabelas: refeição (*meal*) e ementa (menu). A tabela refeição guarda os dados relativos a cada imagem processada e contém os campos apresentados na Figura 19. A tabela ementa armazena informações inseridas pelos utilizadores do *software* através da interface da aplicação *web*, como as ementas diárias, incluindo o tipo de sopa e a quantidade produzida por dia. Esta opção de inserção de dados pelo utilizador final é facultativa. Os campos da tabela ementa podem ser observados na Figura 19.

Importa referir que os dados presentes na base de dados são, na sua maioria, fictícios, para efeitos de demonstração. Isto é, dado que o protótipo não foi colocado em produção, os dados inseridos na base de dados são provenientes dos testes que foram realizados ao protótipo, mas também foram inseridos outros dados manualmente, com o objetivo de popular a base de dados. Foi desenvolvido um script Python que insere dados nas duas tabelas com valores gerados aleatoriamente adequados ao cenário.

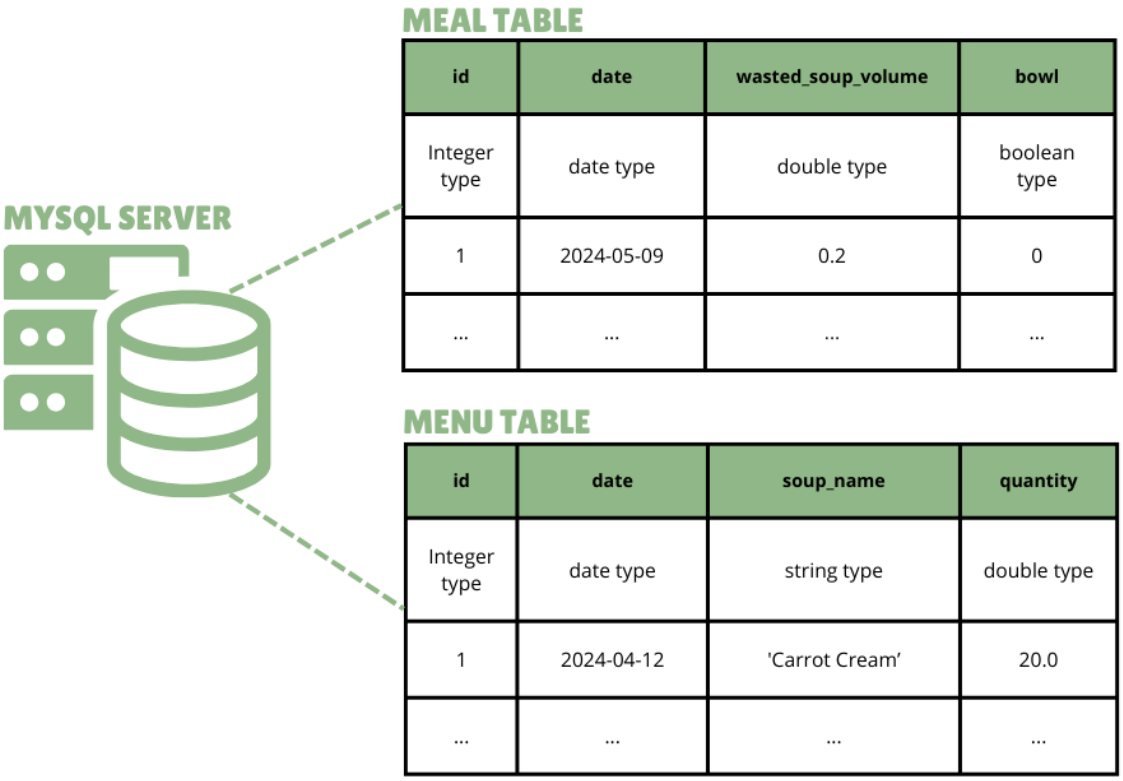

Figura 19 - Composição da base de dados MySQL e respetivas tabelas.

Relativamente à aplicação *web* desenvolvida para permitir a visualização dos dados recolhidos, como mencionado anteriormente, a mesma é baseada em microsserviços. A aplicação foi desenvolvida utilizando a linguagem de programação Java [56] e tem por base a utilização da *framework web* Spring Boot [57]. É constituída por três microsserviços: Frontend, Backend e Eureka Server.

O microsserviço Frontend é responsável por apresentar e gerir a interface do utilizador (do inglês, User Interface, UI) e de fazer pedidos ao microsserviço Backend. Este microsserviço por sua vez é responsável por lidar com a base de dados. O servidor Eureka é um microsserviço com várias funcionalidades, como a de *load balancer*, por exemplo. No entanto, para este projeto, a sua função principal é a descoberta de microsserviços [58]. Ou seja, ele permite que os vários microsserviços se registem, facilitando a comunicação entre eles. A Figura 20 representa a arquitetura da aplicação, conforme explicado anteriormente.

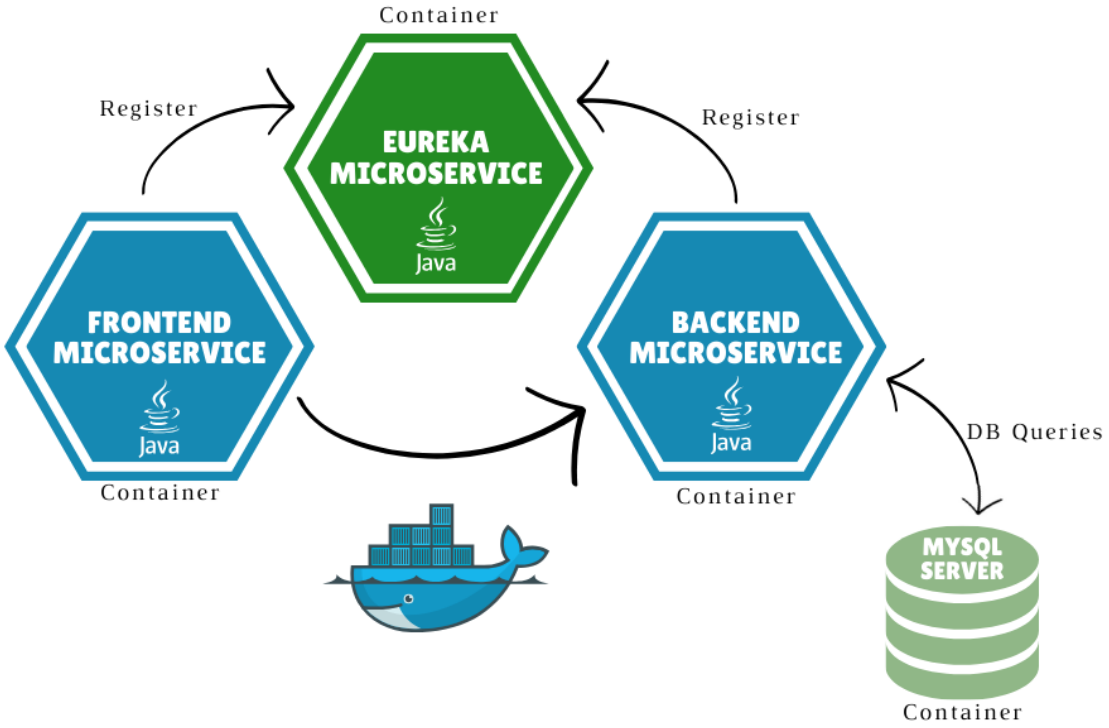

Figura 20 - Arquitetura da aplicação *web*.

Foram criadas as imagens Docker destes microsserviços depois de desenvolvidos, e foi feito "push" para o Docker Hub, para que fosse possível que toda a aplicação ficasse acessível de forma funcional, isto é, acedendo ao IP do servidor e à porta correspondente à aplicação.

As funcionalidades da aplicação estão descritas no diagrama de casos de uso apresentado na Figura 21. Este diagrama ilustra as interações entre os atores e o sistema, destacando as diversas ações que podem ser realizadas na aplicação.

Neste caso as funcionalidades principais são: visualização da página inicial; visualização da página de ementas, visualização da página de estatísticas e visualização da página que contém informações sobre a aplicação. Dentro destas funcionalidades principais existem outras subjacentes. Na página inicial, os dados podem ser visualizados por data, mês ou todos os dados disponíveis. Na página de ementas, é possível inserir ementas diárias e visualizá-las semanalmente. Na página de estatísticas, os dados são apresentados graficamente por mês, semana ou ano.

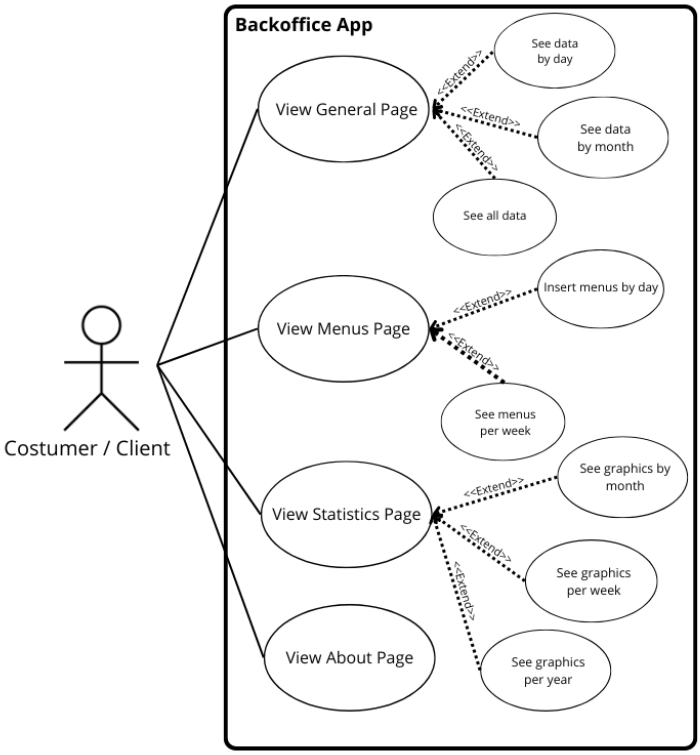

Figura 21 - Diagrama de casos de uso.

De seguida, apresenta-se um conjunto de *screenshots* da aplicação *web*, para proporcionar uma visão detalhada das funcionalidades da mesma. Na Figura 22 é possível visualizar a "General Page", onde o utilizador pode selecionar os dados para visualização por data específica ou mês, e ainda visualizar todos os dados disponíveis.

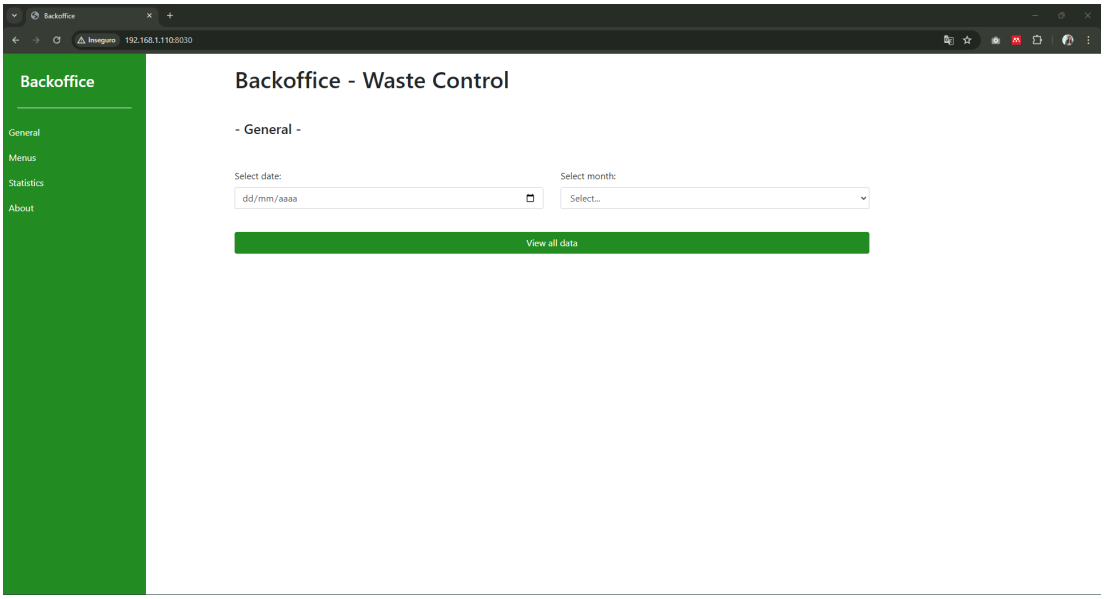

Figura 22 - *App Backoffice: General Page*.

Nesta página, os dados são apresentados em tabelas, como se pode verificar nas Figura 23 e Figura 24. Ao visualizar por data, é possível obter informações como o volume de sopa desperdiçado nesse dia e o volume produzido (caso tenha sido introduzido pelo utilizador). Também é possível consultar o número total de refeições servidas e comparar as que foram servidas com e sem sopa.

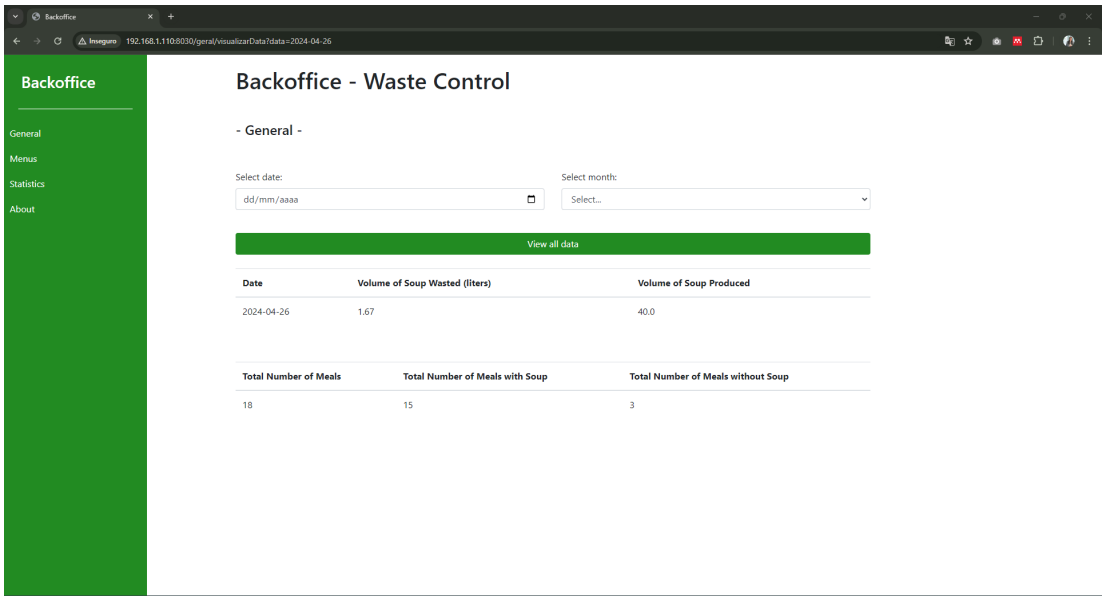

Figura 23 - *App Backoffice: General Page: date.*

Se o utilizador optar pela consulta mensal, pode visualizar o volume de sopa desperdiçado ao longo dos vários dias do mês selecionado, bem como o total de refeições servidas nesse mês. Além disso, é possível visualizar a quantidade total de sopa desperdiçada (em litros) e produzida durante esse período, tal como se pode ver na Figura 24.

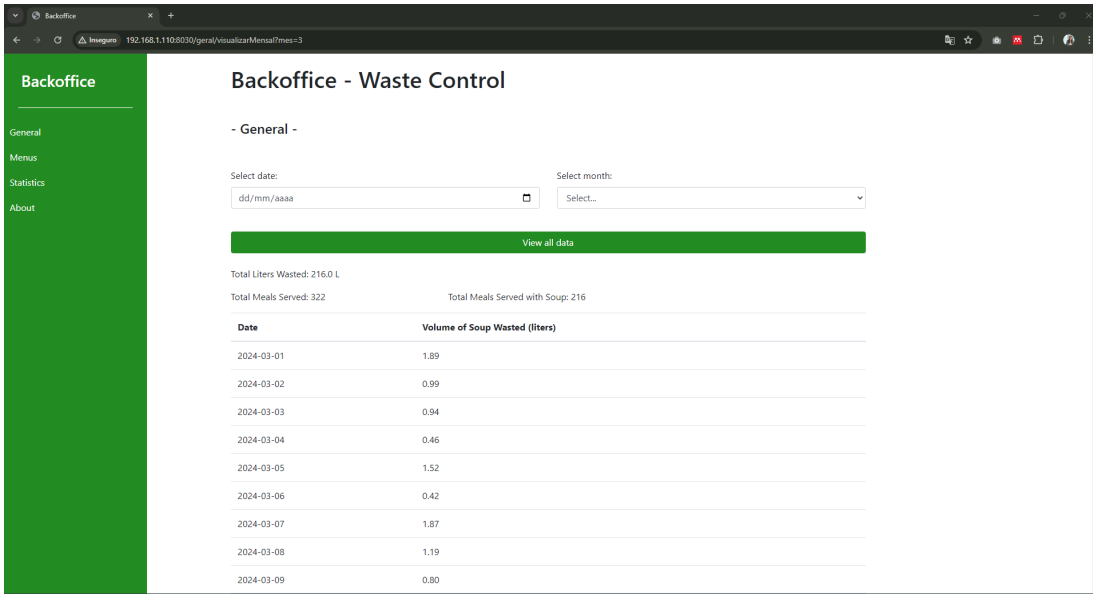

Figura 24 - *App Backoffice: General Page: Month.*

Na página “Menus” disponível na barra de navegação à esquerda, é possível, além de inserir a ementa diária conforme ilustrado na Figura 25, visualizar também as ementas semanais, como se pode observar na Figura 26.

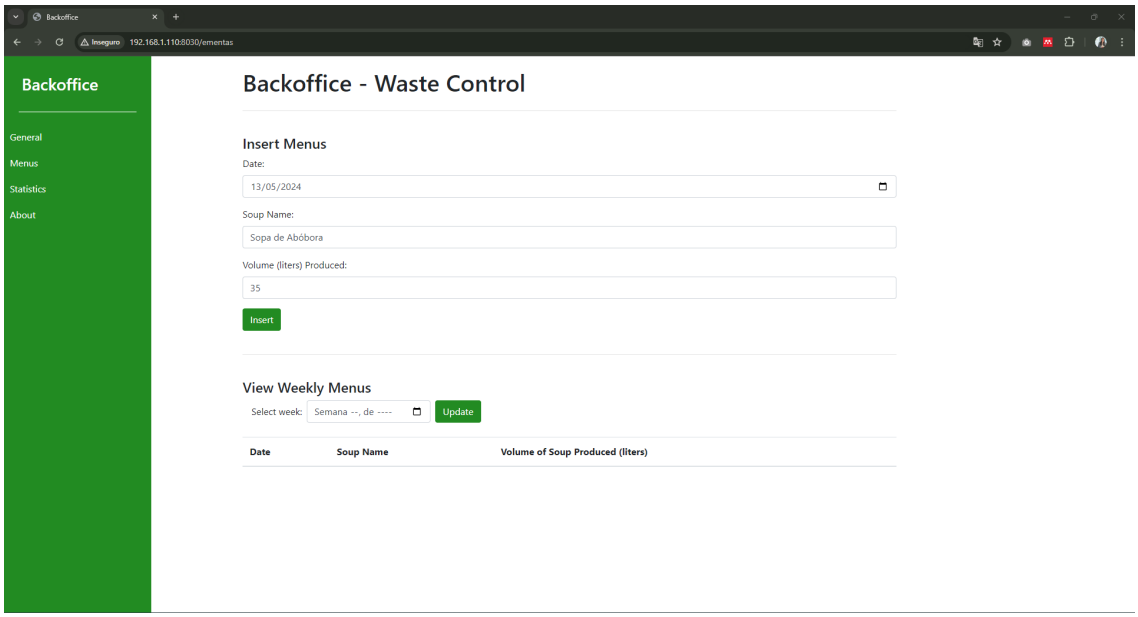

Figura 25 - *App Backoffice: Menus Page:* inserir ementas.

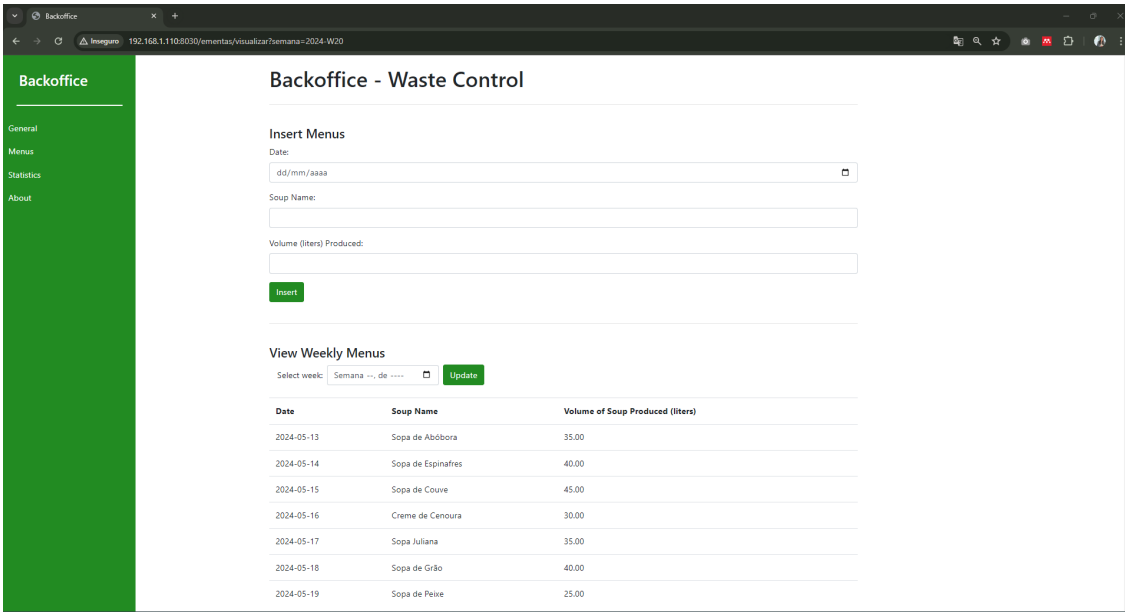

Figura 26 - *App Backoffice: Menus Page:* visualizar ementa.

A página “Statistics”, como o próprio nome indica, permite a consulta de dados estatísticos, elaborados com base na informação alojada na base de dados. Tal como na página inicial, é possível escolher a forma como se pretende visualizar os dados. Na opção “Week”, apresentada na Figura 27, pode consultar-se os dados recolhidos nos dias da semana selecionada pelo utilizador, relativos ao número de tabuleiros que incluem ou não sopa, bem como o volume diário desperdiçado de sopa.

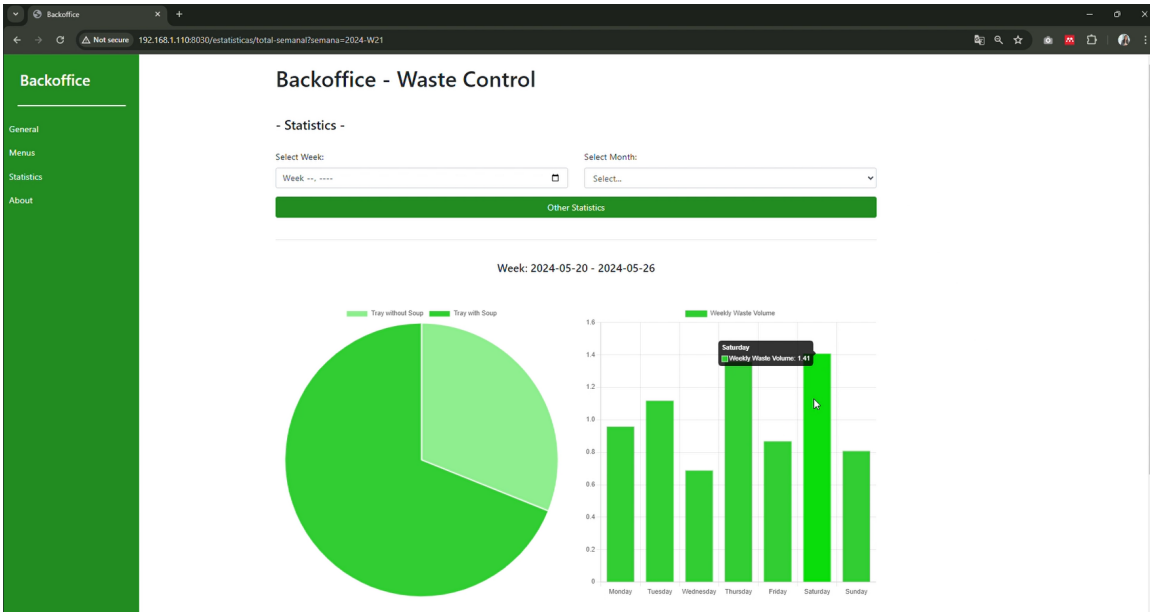

Figura 27 - *App Backoffice: Statistics Page: Week*.

Se o utilizador escolher visualizar os dados estatísticos mensais, apresentados na Figura 28, o gráfico circular é semelhante ao apresentado na visualização semanal, mas com dados relativos ao mês. O gráfico de barras corresponde ao volume de sopa desperdiçada por semana, ao longo do mês selecionado. Enquanto o gráfico de linhas permite a análise dos valores de volume de sopa desperdiçada por dia durante esse mês.

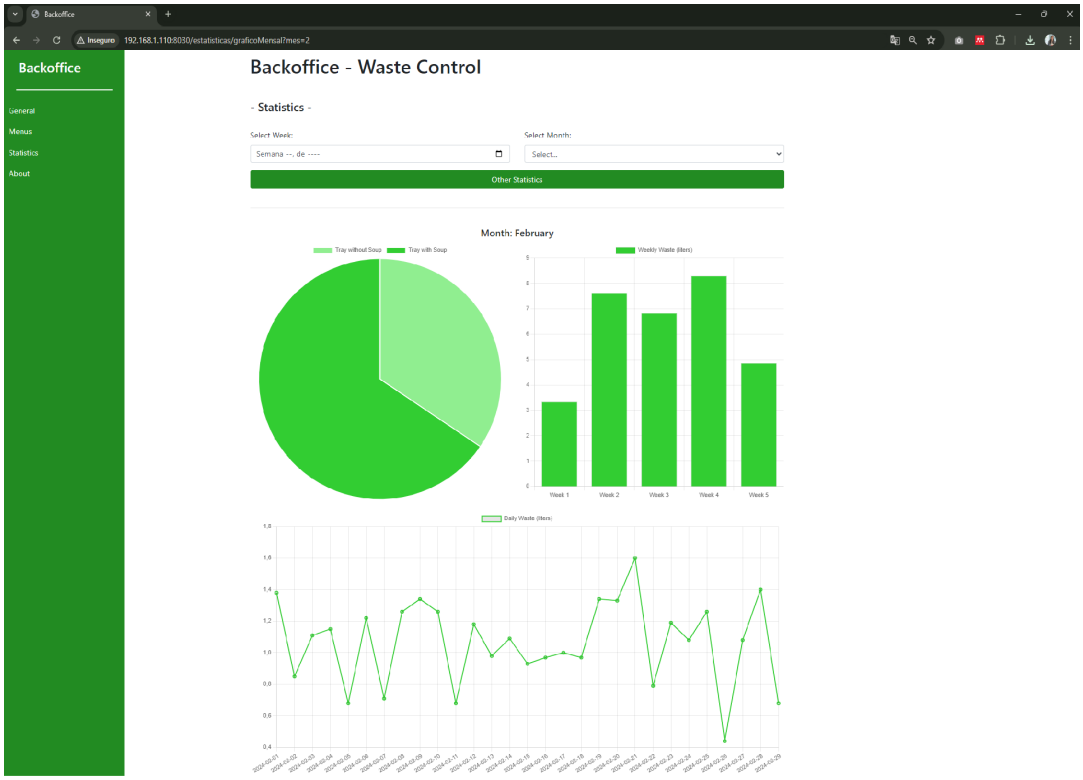

Figura 28 - *App Backoffice: Statistics Page: Month.*

Ainda na página “Statistics”, os dados podem ser visualizados por ano. Na Figura 29, são apresentados os dois tipos de gráficos exibidos quando o utilizador escolhe essa opção. Nesses gráficos, é possível analisar o volume de sopa desperdiçada por mês no ano de 2024. Além disso, é possível comparar o número total de refeições servidas com e sem com sopa, por mês.

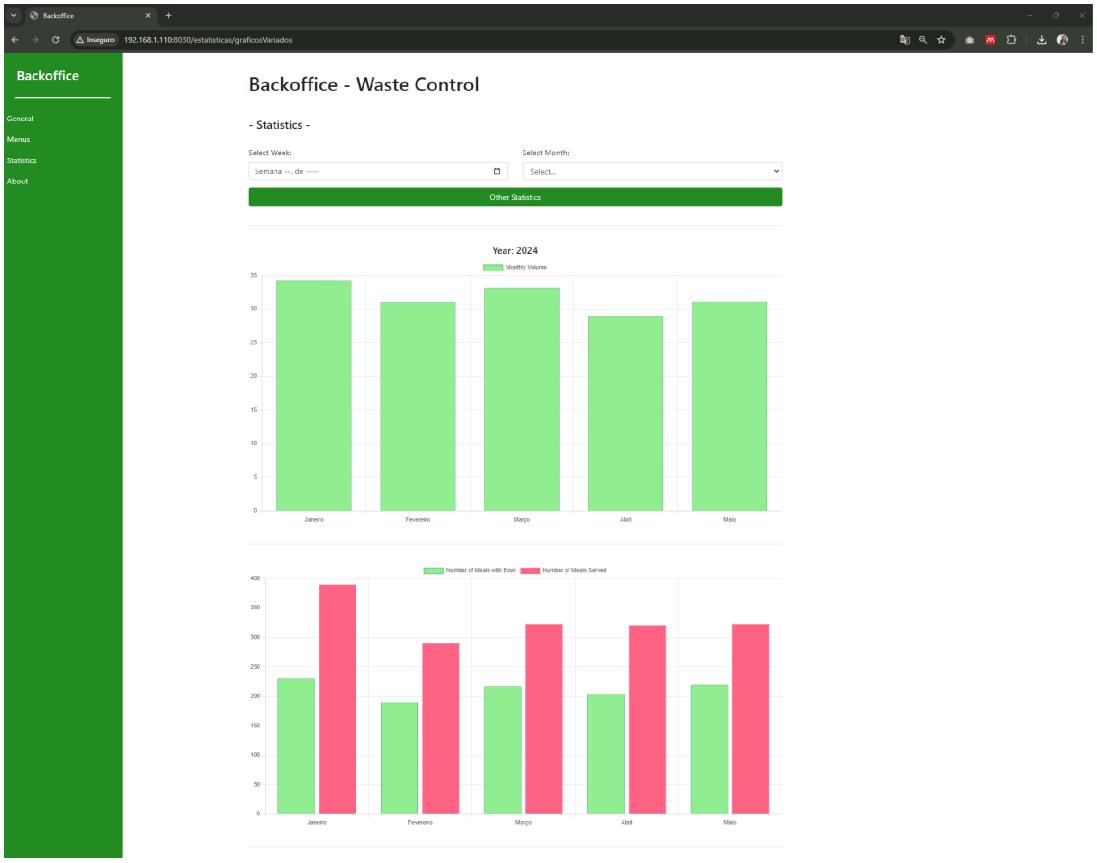

Figura 29 - *App Backoffice: Statistics Page: Year*.

Por fim, na Figura 30, é apresentada a página "About", que contem um breve texto descritivo sobre a aplicação *web* e o projeto desenvolvido.

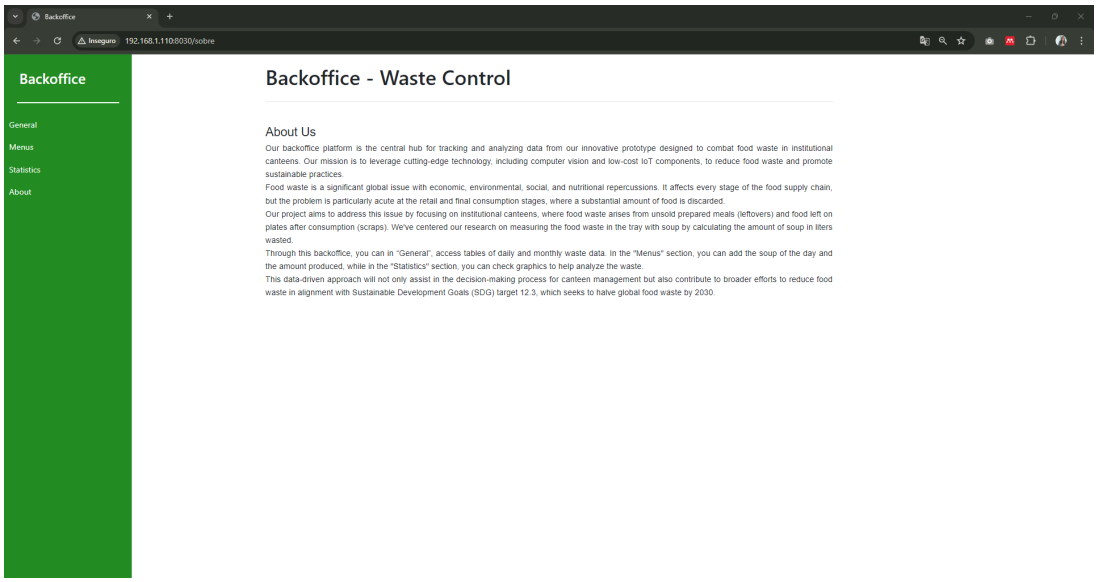

Figura 30 - *App Backoffice: About Page*.

3 Testes de Validação

Neste capítulo são apresentados os testes de validação realizados ao protótipo. A Figura 31 mostra o funcionamento do sensor de infravermelhos ligado ao Raspberry Pi, que é composto por duas luzes vermelhas. Quando apenas uma está acesa, tal indica a ausência de deteção do tabuleiro, e o input é 1. Assim que o tabuleiro é detetado, ambas as luzes são acesas e o input do sensor passa para 0.

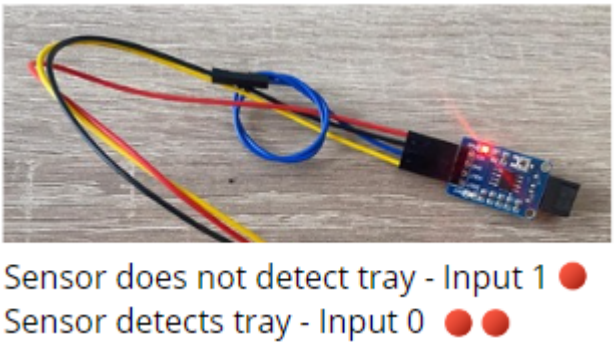

Figura 31 - Funcionamento do sensor infravermelhos.

A Figura 32 demonstra o momento em que o sensor identifica o tabuleiro, evidenciado pelo acender de ambas as luzes vermelhas. Para evitar a captura de uma imagem desfocada, esta não é imediatamente realizada. Assim, a imagem é capturada após 1 segundo, depois transferida para o servidor, e o *input* do sensor é alterado para 0, indicando que o tabuleiro ainda está presente. Este processo é ilustrado na Figura 33, onde se verifica a captura bem-sucedida da imagem e a alteração do *input* para 0.

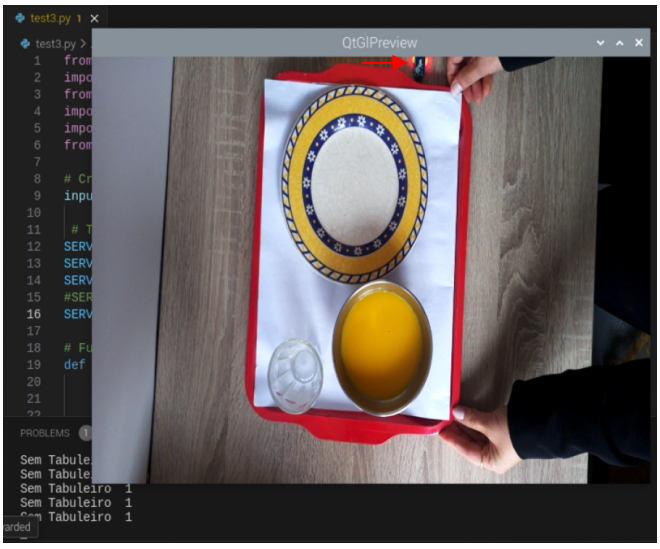

Figura 32 - *Software* de captura e transferência de imagens em execução no Raspberry Pi (1).

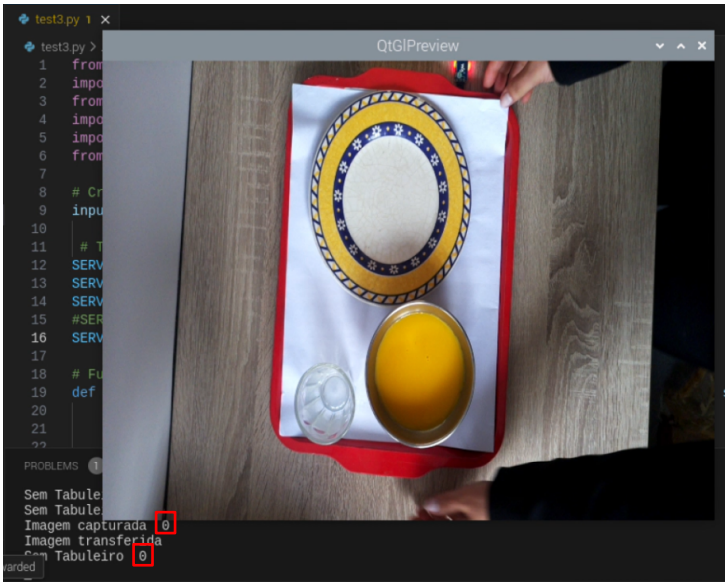

Figura 33 - *Software* de captura e transferência de imagens em execução no Raspberry Pi (2).

No que diz respeito ao servidor, a Figura 34 mostra que a imagem capturada foi armazenada na diretoria "images" após a captura e transferência. De seguida, é estabelecida a conexão com a base de dados. Depois, a diretoria em observação deteta a nova imagem e inicia o seu processamento para tarefas de classificação e reconhecimento de objetos.

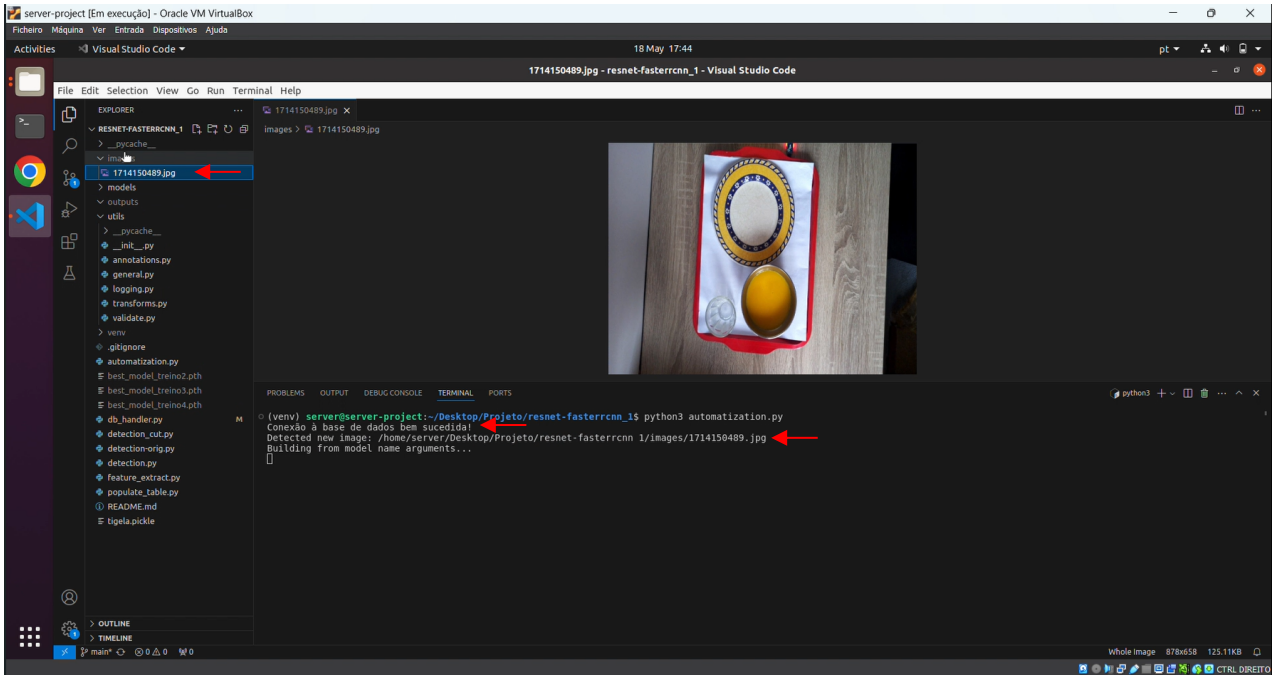

Figura 34 - Início do processamento da imagem no servidor.

Após a conclusão da deteção, a imagem original é eliminada da diretoria "images". É guardada na diretoria "outputs" a imagem com as *bounding boxes* desenhadas, conforme

exemplificado na Figura 35. Para a extração de características, a tigela é isolada e recortada pelas delimitações das *bounding boxes*, sendo posteriormente calculado o seu volume, como evidenciado na mesma Figura.

No teste de validação realizado, a tigela continha um volume restante de 0,12 litros de sopa. Para comparação do valor calculado com o valor real, foi realizada um a medição da sopa da tigela e obteve-se uma quantidade 0,10 litros. Esta pequena discrepância está justificada face ao descrito na subsecção 2.3.4. Por fim, estes dados são enviados e armazenados na base de dados.

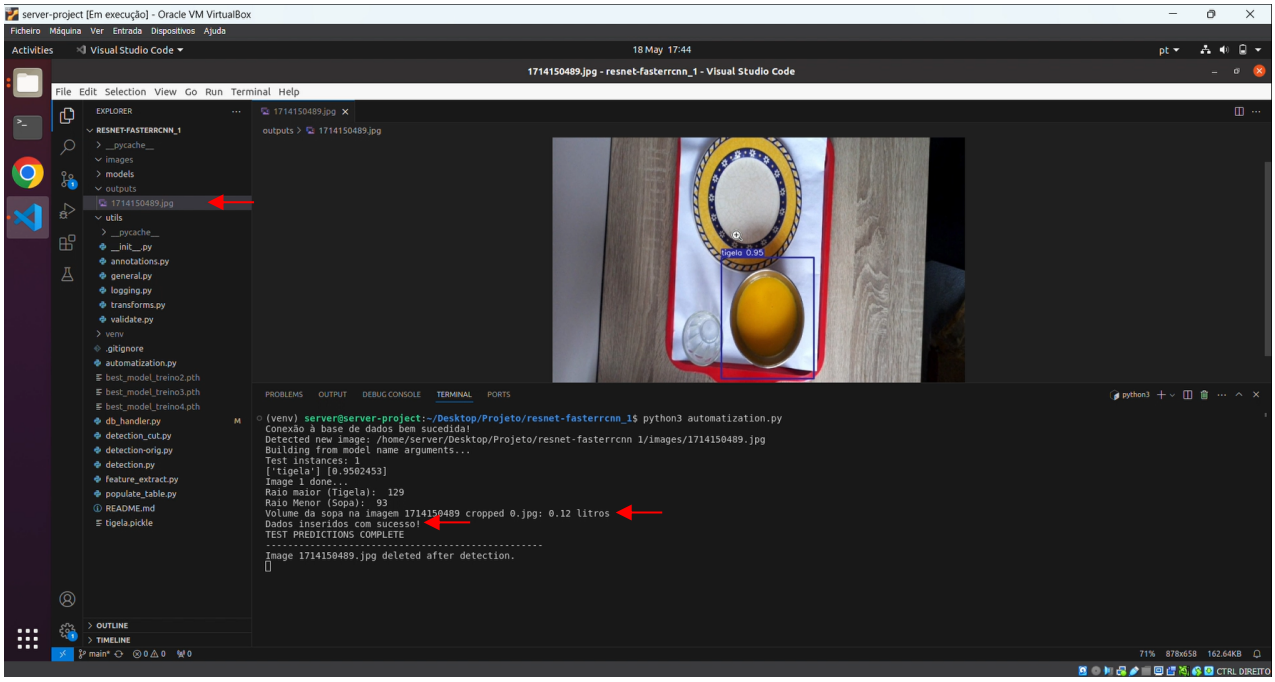

Figura 35 - Deteção da tigela na imagem e cálculo do volume de sopa desperdiçada.

Terminados estes processos, o ciclo volta ao início e o sistema fica a aguardar que uma nova imagem chegue ao servidor para iniciar novamente o processamento.

4 Conclusões e Trabalho Futuro

4.1 Conclusões

Reduzir o desperdício de alimentos não é apenas uma opção, mas uma responsabilidade ética com uma urgência temporal. Cada refeição recuperada ajuda a enfrentar desafios importantes, com consequências significativas nas esferas económica, ambiental e social. Compreender a necessidade urgente de resolver estes problemas é fundamental, especialmente considerando a grande quantidade de alimentos perdidos ao longo da cadeia de abastecimento, acentuada pelos hábitos de retalhistas e consumidores.

O trabalho apresentado neste projeto é apenas um ponto de partida para enfrentar a complexidade do desperdício alimentar. As tecnologias de informação, tais como a visão computacional e a inteligência artificial, podem ter um papel importante no combate ao desperdício alimentar. Estas tecnologias permitem identificar e analisar padrões que podem orientar estratégias para reduzir eficazmente as perdas alimentares.

Este Projeto teve por objetivo principal propor e avaliar uma solução que possibilite identificar e quantificar desperdício alimentar numa cantina institucional, contribuindo para o desenvolvimento de estratégias que conduzam à redução do desperdício. Nesse sentido, o presente trabalho procurou, a partir do estudo do estado da arte e avaliação inicial de desempenho conduzido em Projeto I, apresentar uma proposta, implementação, teste e validação de um protótipo demonstrador do conceito, para o alimento sopa.

Consideram-se como principais contributos do trabalho realizado em Projeto II: a criação de um *dataset* adaptado cenário e alimento em questão [24]; o treino e validação do modelo ResNet-50 combinado com o modelo Faster R-CNN para deteção; o estudo da extração de características das imagens; a definição de uma arquitetura, bem como as componentes de hardware e software de um protótipo de baixo custo, baseado em Internet das Coisas e visão computacional (CNNs); a descrição do processo de implementação e configuração do protótipo; o desenvolvimento de uma aplicação *web* baseada em microsserviços para visualizar os dados recolhidos; e finalmente os testes para validação e prova de conceito realizados sobre o protótipo.

O protótipo demonstrou ser eficaz pois atingiu os objetivos propostos na sua conceção. É considerado uma opção economicamente viável e funcional. Em conjunto com a aplicação, pode ser utilizado no apoio à tomada de decisões e contribui para uma gestão mais eficiente, com o objetivo de reduzir o desperdício alimentar.

4.2 Trabalho Futuro

A solução apresentada neste projeto foca-se no combate ao desperdício de sopa, num ambiente controlado desde a criação do *dataset* até à deteção, classificação e cálculo do volume de sopa desperdiçada. O objetivo foi validar um conceito, demonstrando que era

possível quantificar o desperdício, utilizando um hardware de baixo custo e técnicas de visão computacional.

Para replicar esta solução, é necessário adaptá-la ao ambiente em que será utilizada. Isso inclui criar um *dataset* adequado aos pratos comercializados, treinar a CNN com o *dataset* criado e adaptar o hardware ao espaço em que é utilizado, por exemplo ajustando a localização do sensor e da câmara.

Quanto à extração de características, esta também precisa de ser adaptada ao contexto de implementação. No caso desta solução, foi desenvolvida para a deteção de círculos e a cor da sopa, de forma a obter uma estimativa do volume da sopa. É importante reconhecer as suas limitações e considerar possíveis fatores de erro, como imprecisões na deteção de círculos e na conversão de pixels para centímetros. Esta forma de extração de características pode facilmente ser replicada para outros alimentos servidos em tigelas, como, por exemplo, sobremesas, mas teria de ser adaptada a outro tipo de alimentos servidos em outros formatos, como pratos.

No treino do modelo Resnet-50 e Faster R-CNN, as métricas de desempenho na avaliação deste apresentam resultados extremamente positivos devido ao facto de se trabalhar num ambiente muito controlado, onde todas as imagens do *dataset* são provenientes de uma única cantina, resultando numa grande semelhança entre elas. Para melhorar o desempenho do modelo, seria vantajoso diversificar o ambiente de treino, incluindo imagens de várias cantinas ou ambientes alimentares distintos, o que ajudaria o modelo a generalizar melhor para diferentes cenários. Adicionalmente, aumentar a diversidade do *dataset*, incorporando uma variedade de condições de iluminação, ângulos de visão e disposições dos alimentos nos tabuleiros.

Além disso, a aplicação web desenvolvida para visualização dos dados recolhidos pode ser adaptada às necessidades dos utilizadores, oferecendo um melhor apoio à tomada de decisões. Isso pode incluir personalizações na interface, ferramentas de análise mais avançadas e integração com outros sistemas ou processos existentes na cantina, para uma gestão ainda mais eficiente do desperdício alimentar.

Referências Bibliográficas

[1] J. Gustavsson, C. Cederberg, U. Sonesson, R. van Otterdijk, and A. Meybeck, *Global Food Losses and Food Waste: Extent, Causes and Prevention, Rome: Food and Agriculture Organisation of the United Nations*, vol. 365, no. 1554. Rome, Italy, 2011.

[2] Instituto Nacional de Estatística, "Desperdício alimentar - Resultados e perspetivas," Jul. 2022.

[3] Tribunal de Contas Europeu, "Relatório Especial - Luta contra o desperdício alimentar: uma oportunidade para a UE melhorar a eficiência dos recursos na cadeia de abastecimento alimentar," 2016.

[4] United Nations, "Ensure sustainable consumption and production patterns," https://sdgs.un.org/goals/goal12. Accessed: Oct. 05, 2023. [Online]. Available: https://sdgs.un.org/goals/goal12

[5] A. Correia, C. Aidos, J. M. L. P. Caldeia, and V. N. G. J. Soares, "Using Computer Vision for Reducing Food Waste in an Institutional Canteen," *Waste, MDPI. Submitted, pending review*.

[6] "Raspberry Pi." Accessed: May 21, 2024. [Online]. Available: https://www.raspberrypi.com/

[7] "Enterprise Open Source and Linux | Ubuntu." Accessed: May 21, 2024. [Online]. Available: https://ubuntu.com/

[8] "Welcome to Python.org." Accessed: May 21, 2024. [Online]. Available: https://www.python.org/

[9] "MySQL." Accessed: May 21, 2024. [Online]. Available: https://www.mysql.com/

[10] "Docker Desktop: The #1 Containerization Tool for Developers | Docker." Accessed: May 17, 2024. [Online]. Available: https://www.docker.com/products/docker-desktop/

[11] "Microsserviços e o impacto na escalabilidade de aplicações." Accessed: May 21, 2024. [Online]. Available: https://imaginedone.com.br/artigos/inovacao-e-tecnologia/microsservicos/

[12] "Canvas Select Plus microSD Card, A1, Class 10 UHS-I, 64GB to 512GB - Kingston Technology." Accessed: May 18, 2024. [Online]. Available: https://www.kingston.com/en/memory-cards/canvas-select-plus-microsd-card

[13] "Buy a Raspberry Pi Camera Module 3 – Raspberry Pi." Accessed: May 18, 2024. [Online]. Available: https://www.raspberrypi.com/products/camera-module-3/

[14] "Infrared Reflective Sensor - Waveshare Wiki." Accessed: May 18, 2024. [Online]. Available: https://www.waveshare.com/wiki/Infrared_Reflective_Sensor

[15] "Camera - Raspberry Pi Documentation." Accessed: May 18, 2024. [Online]. Available: https://www.raspberrypi.com/documentation/accessories/camera.html

[16] "How To Use Dual Cameras on the Raspberry Pi 5 | Tom's Hardware." Accessed: May 18, 2024. [Online]. Available: https://www.tomshardware.com/raspberry-pi/how-to-use-dual-cameras-on-the-raspberry-pi-5

[17] "Raspberry Pi hardware - Raspberry Pi Documentation." Accessed: May 18, 2024. [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[18] "Raspberry Pi 5 - gpiod vs RPi.GPIO - Raspberry Pi Forums." Accessed: May 18, 2024. [Online]. Available: https://forums.raspberrypi.com/viewtopic.php?t=359742

[19] "Using a IR Reflective Sensor - Raspberry Pi Forums." Accessed: May 18, 2024. [Online]. Available: https://forums.raspberrypi.com/viewtopic.php?t=181544

[20] "2. Basic Recipes — gpiozero 2.0.1 Documentation." Accessed: May 18, 2024. [Online]. Available: https://gpiozero.readthedocs.io/en/stable/recipes.html#pin-numbering

[21] "Oracle VM VirtualBox." Accessed: May 18, 2024. [Online]. Available: https://www.virtualbox.org/

[22] "Instituto Politécnico de Castelo Branco." Accessed: May 21, 2024. [Online]. Available: https://www.ipcb.pt/

[23] "Roboflow: Computer vision tools for developers and enterprises." Accessed: May 15, 2024. [Online]. Available: https://roboflow.com/

[24] "Soup-Bowl-Dataset Dataset > Overview." Accessed: Jun. 04, 2024. [Online]. Available: https://universe.roboflow.com/sopa/soup-bowl-dataset

[25] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow : concepts, tools, and techniques to build intelligent systems*, Third Edition. Sebastopol, CA: O'Reilly Media, 2022.

[26] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J Big Data*, vol. 8, no. 1, p. 53, Mar. 2021, doi: 10.1186/s40537-021-00444-8.

[27] J. Maurício, I. Domingues, and J. Bernardino, "Comparing Vision Transformers and Convolutional Neural Networks for Image Classification: A Literature Review," *Applied Sciences (Switzerland)*, vol. 13, no. 9, p. 5521, May 2023, doi: https://doi.org/10.3390/app13095521.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jun. 2015, [Online]. Available: http://arxiv.org/abs/1506.01497

[29] “Google Colab.” Accessed: May 09, 2024. [Online]. Available: https://colab.research.google.com/drive/1thXa8nF65Iywbcbm4LaZ_KDD1bkIvi9O?usp=sharing

[30] “sovit-123/fasterrcnn-pytorch-training-pipeline: PyTorch Faster R-CNN Object Detection on Custom Dataset.” Accessed: May 09, 2024. [Online]. Available: https://github.com/sovit-123/fasterrcnn-pytorch-training-pipeline

[31] “Personal Cloud Storage & File Sharing Platform - Google.” Accessed: May 24, 2024. [Online]. Available: https://www.google.com/drive/

[32] “GitHub: Let’s build from here · GitHub.” Accessed: May 24, 2024. [Online]. Available: https://github.com/

[33] “Overfitting and underfitting in machine learning | SuperAnnotate.” Accessed: May 21, 2024. [Online]. Available: https://www.superannotate.com/blog/overfitting-and-underfitting-in-machine-learning

[34] X. Ying, “An Overview of Overfitting and its Solutions,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Mar. 2019. doi: 10.1088/1742-6596/1168/2/022022.

[35] “What is Average Precision in Object Detection & Localization Algorithms and how to calculate it? | by Aqeel Anwar | Towards Data Science.” Accessed: May 21, 2024. [Online]. Available: https://towardsdatascience.com/what-is-average-precision-in-object-detection-localization-algorithms-and-how-to-calculate-it-3f330efe697b

[36] “mAP (mean Average Precision) for Object Detection | by Jonathan Hui | Medium.” Accessed: May 21, 2024. [Online]. Available: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

[37] V. Mudivedu, “What is the difference between Training Loss Validation Loss and Evaluation Loss,” Medium. Accessed: Jan. 18, 2024. [Online]. Available: https://medium.com/@penpencil.blr/what-is-the-difference-between-training-loss-validation-loss-and-evaluation-loss-c169ddeccd59

[38] “Raspberry Pi OS – Raspberry Pi.” Accessed: May 09, 2024. [Online]. Available: https://www.raspberrypi.com/software/

[39] “How To Use Picamera2 to Take Photos With Raspberry Pi | Tom’s Hardware.” Accessed: May 17, 2024. [Online]. Available: https://www.tomshardware.com/how-to/use-picamera2-take-photos-with-raspberry-pi

[40] “2. Basic Recipes — gpiozero 2.0.1 Documentation.” Accessed: May 09, 2024. [Online]. Available: https://gpiozero.readthedocs.io/en/stable/recipes.html#pin-numbering

[41] “Paramiko- How to SSH and transfer files with python | by Mokgadi Rasekgala | Medium.” Accessed: May 09, 2024. [Online]. Available: https://medium.com/@keagileageek/paramiko-how-to-ssh-and-file-transfers-with-python-75766179de73

[42] "python - How to transfer a file to ssh server in an ssh-connection made by paramiko? - Stack Overflow." Accessed: May 09, 2024. [Online]. Available: https://stackoverflow.com/questions/11499507/how-to-transfer-a-file-to-ssh-server-in-an-ssh-connection-made-by-paramiko

[43] "Python: Event Monitoring with watchdogs | by Pravash | Medium." Accessed: May 17, 2024. [Online]. Available: https://pravash-techie.medium.com/python-event-monitoring-with-watchdogs-86125f946da6

[44] PyTorch, "PyTorch 2.1 documentation." Accessed: Dec. 29, 2023. [Online]. Available: https://pytorch.org/docs/stable/index.html

[45] "NumPy -." Accessed: May 17, 2024. [Online]. Available: https://numpy.org/

[46] "os — Miscellaneous operating system interfaces — Python 3.12.3 documentation." Accessed: May 17, 2024. [Online]. Available: https://docs.python.org/3/library/os.html

[47] "The Official YAML Web Site." Accessed: May 17, 2024. [Online]. Available: https://yaml.org/

[48] "Matplotlib — Visualization with Python." Accessed: May 17, 2024. [Online]. Available: https://matplotlib.org/

[49] "copy — Shallow and deep copy operations — Python 3.12.3 documentation." Accessed: May 17, 2024. [Online]. Available: https://docs.python.org/3/library/copy.html

[50] "math — Mathematical functions — Python 3.12.3 documentation." Accessed: May 17, 2024. [Online]. Available: https://docs.python.org/3/library/math.html

[51] "Coding Gaussian Blur Operation In Python From Scratch | by Rohit Krishna | Medium | Medium." Accessed: May 21, 2024. [Online]. Available: https://medium.com/@rohit-krishna/coding-gaussian-blur-operation-from-scratch-in-python-f5a9af0a0c0f

[52] "Implementing the Hough Transform from Scratch | by Alberto Formaggio | Medium." Accessed: May 21, 2024. [Online]. Available: https://medium.com/@alb.formaggio/implementing-the-hough-transform-from-scratch-09a56ba7316b

[53] "Truncated Cone Volume Calculator." Accessed: May 17, 2024. [Online]. Available: https://www.omnicalculator.com/math/truncated-cone-volume

[54] "mysql - Official Image | Docker Hub." Accessed: May 17, 2024. [Online]. Available: https://hub.docker.com/_/mysql

[55] "Docker Hub Container Image Library | App Containerization." Accessed: May 21, 2024. [Online]. Available: https://hub.docker.com/

[56] "Java | Oracle." Accessed: May 21, 2024. [Online]. Available: https://www.java.com/en/

[57] “Spring Boot.” Accessed: May 17, 2024. [Online]. Available: https://spring.io/projects/spring-boot

[58] “2. Service Discovery: Eureka Server.” Accessed: May 17, 2024. [Online]. Available: https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-eureka-server.html