UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
OFICINAS DE INTEGRAÇÃO 2

MATEUS SILVA
SEBASTIÃO ARAUJO
WILLIAM CHAKUR

CHECK.AI

RELATÓRIO FINAL

CURITIBA

2025

MATEUS SILVA
SEBASTIÃO ARAUJO
WILLIAM CHAKUR

CHECK.AI

Relatório Final apresentada ao Oficinas de Integração 2 da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de "*Ciências" – Área de Concentração: Engenharia de Computação.*

Orientador: Prof. Daniel Rossato

CURITIBA

2025

RESUMO

SILVA, Mateus; ARAUJO, Sebastião; CHAKUR, William. CHECK.AI. 27 f. Relatório Final – Oficinas de Integração 2, Universidade Tecnológica Federal do Paraná. Curitiba, 2025.

O Check.AI é um sistema de autoatendimento inteligente desenvolvido para agilizar o processo de checkout em supermercados, eliminando a necessidade de códigos de barras, especialmente para itens como frutas e legumes.

Utilizando um Raspberry Pi 3B+, uma câmera e uma balança digital, o dispositivo identifica automaticamente os produtos, calcula seus preços com base no peso real e exibe o valor total da compra. A interface, construída com Django, HTML, CSS e JavaScript, permite uma interação simples e direta com o usuário.

As imagens capturadas são enviadas para a API da OpenAI (gpt-4.1-mini) junto a um banco local SQLite com informações sobre os produtos. A IA retorna os itens reconhecidos e suas quantidades, e o sistema valida a identificação comparando o peso real da balança com o peso esperado dos produtos cadastrados, permitindo a detecção de inconsistências.

A lógica de verificação segue o padrão de Strategy, permitindo regras específicas para produtos convencionais e frutas. Ao final, o sistema gera um QR Code PIX via biblioteca pypix, concluindo o processo de forma ágil e sem fricção.

O Check.AI propõe uma solução concreta para os gargalos dos caixas de autoatendimento, promovendo um varejo mais eficiente e inteligente.

Palavras-chave: Autoatendimento, Inteligência Artificial, Validação por Peso, Automação de Varejo

ABSTRACT

SILVA, Mateus; ARAUJO, Sebastião; CHAKUR, William. TITLE IN ENGLISH. 27 f. Relatório Final – Oficinas de Integração 2, Universidade Tecnológica Federal do Paraná. Curitiba, 2025.

Check.AI is an intelligent self-checkout system designed to streamline the checkout process in supermarkets, eliminating the need for barcodes—particularly for items like fruits and vegetables. The system leverages a Raspberry Pi 3B+, a camera, and a digital scale to automatically identify products, determine their prices based on real weight, and display the total amount to be paid.

The user interface, built with Django, HTML, CSS, and JavaScript, ensures a seamless interaction. Captured images are sent to the OpenAI API (gpt-4.1-mini) alongside contextual data stored in a local SQLite database. The AI returns the recognized items and quantities, which are then validated by comparing their expected weight to the actual measurement, enabling error detection.

The verification logic follows the Strategy design pattern, applying distinct business rules for general items and fruits. Finally, the system generates a PIX QR Code using the pypix library, completing the purchase efficiently. Check.AI presents a practical solution to the main pain points of self-checkout systems, offering a smarter, automated retail experience.

Keywords: Self-checkout, Artificial Intelligence, Weight Validation, Retail Automation

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Constantemente, consumidores em supermercados lidam com tecnologias de autoatendimento (WU et al., 2016). Conhecidas como Self-Service Technologies (SSTs), essas soluções foram introduzidas com o objetivo de reduzir custos operacionais, agilizar o atendimento e melhorar a experiência de compra. No entanto, apesar de sua adoção crescente, estudos revelam desafios significativos relacionados à sua aceitação e usabilidade (OREL, 2014). Dentre os principais obstáculos, destacam-se as dificuldades enfrentadas por usuários ao registrar itens em grande quantidade ou produtos sem código de barras, como frutas, pães e verduras (RONDáN et al., 2021).

Esses gargalos comprometem a fluidez da jornada de compra e, em alguns casos, exigem a intervenção de funcionários, o que contradiz o propósito original das SSTs: a autonomia e eficiência no atendimento.

1.1 MOTIVAÇÃO

A motivação central para o desenvolvimento do projeto Check.AI nasceu da observação dessas ineficiências. A dependência de códigos de barras foi identificada como o principal gargalo no processo de autoatendimento, forçando os consumidores a interromperem o fluxo da compra para realizar tarefas manuais de busca, pesagem e etiquetagem de produtos frescos.

O projeto foi impulsionado pelo desafio de superar essa barreira, com o intuito de criar uma experiência de checkout mais automatizada. A proposta consiste em desenvolver um sistema que utiliza tecnologias de inteligência artificial e visão computacional para realizar a identificação e precificação dos produtos de forma autônoma, tornando o processo mais ágil e intuitivo para o usuário.

1.2 OBJETIVOS

O objetivo deste trabalho é desenvolver um protótipo funcional, denominado Check.AI, capaz de reconhecer e precificar automaticamente produtos de supermercado, eliminando a necessidade de códigos de barras. O sistema deve ser capaz de identificar visualmente os itens por meio de uma câmera e calcular seus respectivos preços com base em seu peso real, medido por uma balança integrada.

1.2.1 OBJETIVO GERAL

A solução proposta visa mitigar os principais gargalos observados nos sistemas de autoatendimento tradicionais. Para isso, o Check.AI utiliza um Raspberry Pi 3B+, uma câmera e uma balança digital. A interface foi desenvolvida com Django, HTML, CSS e JavaScript, garantindo simplicidade e acessibilidade ao usuário final.

As imagens capturadas são processadas pela API da OpenAI (gpt-4.1-mini), junto com um banco de dados local em SQLite contendo os produtos disponíveis. A IA retorna os nomes e quantidades dos itens reconhecidos, que são posteriormente validados por meio de verificação cruzada entre o peso real e os pesos esperados dos produtos.

A lógica de verificação adota o padrão de projeto Strategy, permitindo aplicar regras específicas a diferentes categorias de produtos — como frutas, que devem ser pesadas individualmente. Ao final do processo, o sistema gera um QR Code PIX, utilizando a biblioteca pypix, finalizando a compra de forma rápida e sem fricções.

Dessa forma, o Check.AI representa uma resposta concreta aos desafios enfrentados pelos sistemas de autoatendimento atuais, propondo uma alternativa inteligente, escalável e de fácil integração no varejo.

1.2.2 OBJETIVOS ESPECÍFICOS

- **Montar o Hardware do Protótipo:** Construir a estrutura física do dispositivo a partir de um microcontrolador Raspberry Pi, integrando os componentes essenciais: uma câmera e uma balança digital.
- **Implementar o Reconhecimento Visual:** Utilizar a API de visão computacional da OpenAI para permitir que o sistema identifique produtos a partir das imagens capturadas pela câmera.

- **Criar o Mecanismo de Verificação por Peso:** Desenvolver uma lógica de software que compara o peso real medido pela balança com o peso médio esperado de cada produto, armazenado em um banco de dados, a fim de criar uma camada de validação para a identificação.
- **Desenvolver a Aplicação Central (Backend):** Implementar o sistema de gerenciamento com o framework Django, responsável por controlar o banco de dados, as regras de negócio e a comunicação com a interface do usuário.
- **Construir a Interface do Usuário (Frontend):** Criar uma tela de interação clara e funcional para que o usuário possa acompanhar o registro dos produtos e finalizar a compra.
- **Implementar o Sistema de Pagamento:** Integrar uma função para gerar um QR Code no padrão PIX ao final da compra, oferecendo um método de pagamento rápido e digital.

2 DESENVOLVIMENTO

Este capítulo descreve em detalhes o processo de desenvolvimento do Check.AI, abrangendo a arquitetura do sistema, as tecnologias empregadas e a implementação específica de cada componente de hardware e software.

2.1 ESTRUTURA

A estrutura física do Check.AI foi projetada para acomodar de forma funcional todos os componentes do sistema: Raspberry Pi, balança digital, câmera e display de interface com o usuário. O projeto foi inteiramente modelado no *Fusion*, permitindo uma visualização precisa da integração entre as partes e facilitando o planejamento da montagem.

A peça foi impressa em 3D utilizando filamento PLA, o que possibilitou uma produção personalizada, com baixo custo e tempo de fabricação reduzido. A impressão foi realizada em uma impressora 3D. A imagem da Figura 1 ilustra a estrutura final modelada.

2.2 ARQUITETURA DO SISTEMA

O Check.AI foi dividida em três partes principais: hardware, software backend e software frontend. Essa divisão ajuda a organizar melhor o sistema, facilita consertos e permite que ele cresça com mais facilidade no futuro.

- **Hardware:** A base física do protótipo é composta por um microcontrolador **Raspberry Pi 3B+**, que atua como o cérebro do sistema. Conectados a ele, estão os periféricos essenciais: uma **câmera USB** para a captura de imagens e uma **balança digital**, construída com uma célula de carga e um módulo conversor HX711, para a medição do peso.
- **Software Backend:** Desenvolvido em Python com o framework **Django**, o backend é o núcleo lógico do sistema. Ele é responsável por orquestrar todas as operações: servir

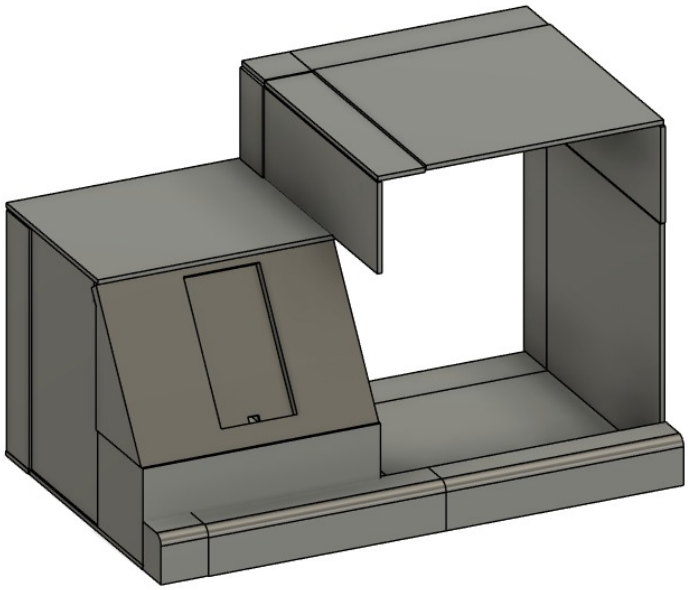

Figura 1: Estrutura modelada no Fusion para impressão em 3D

a aplicação web para o usuário, controlar o hardware (câmera e balança), gerenciar o banco de dados de produtos, processar a lógica de negócio e se comunicar com a API de inteligência artificial.

- **Software Frontend:** A interface com o usuário é uma aplicação web de página única, desenvolvida com **HTML, CSS e JavaScript**. Ela se comunica com o backend através de requisições, permitindo que o usuário inicie o processo de compra, receba o resultado da análise e finalize o pagamento.

2.3 TECNOLOGIAS UTILIZADAS

A implementação do projeto foi viabilizada pela combinação de diversas tecnologias e bibliotecas de software livre, gerenciadas através do `poetry`. As principais foram:

- **Django:** Framework web utilizado para construir a aplicação backend de forma rápida e estruturada, gerenciando o banco de dados, as rotas da API e a lógica de negócio.
- **OpenAI API:** Utilizada como serviço de inteligência artificial para o reconhecimento dos produtos a partir das imagens capturadas. Cada requisição custou em torno de 0,00054 cents

- **OpenCV (`opencv-python`):** Biblioteca de visão computacional empregada para a captura programática de imagens da câmera.
- **HX711 e RPi.GPIO:** Bibliotecas Python que permitem a comunicação de baixo nível com o hardware, possibilitando a leitura dos dados da balança através das portas GPIO do Raspberry Pi.
- **pypix:** Biblioteca utilizada para a geração do QR Code de pagamento no padrão PIX do Banco Central do Brasil, oferecendo uma forma de pagamento moderna e integrada.

2.4 IMPLEMENTAÇÃO DETALHADA

A seguir, são detalhadas as principais funcionalidades implementadas no sistema, desde a interação com o hardware até a lógica de negócio no software.

2.4.1 INTEGRAÇÃO COM HARDWARE

A comunicação com os componentes físicos foi um passo fundamental do projeto.

2.4.1.1 BALANÇA DIGITAL (`BALANCE.PY`)

O sistema interage com a célula de carga através da biblioteca `HX711`. Para garantir uma medição precisa, foi implementada uma rotina que realiza 5 leituras consecutivas do sensor e calcula a média, minimizando o impacto de ruídos elétricos. Os valores de calibração, essenciais para converter os dados brutos em gramas, são lidos de um arquivo externo (`calibracao.json`), permitindo ajustes sem a necessidade de alterar o código.

2.4.1.2 CÂMERA (`PYCAMERA.PY`)

A captura de imagens é realizada pela biblioteca OpenCV. Uma função simples é responsável por acessar a câmera, capturar um único quadro (*frame*) e salvá-lo temporariamente para o processamento.

2.4.2 PROCESSAMENTO DA COMPRA E LÓGICA DE NEGÓCIO

O fluxo de processamento da compra é a funcionalidade central do backend.

1. **Requisição do Frontend:** O processo inicia quando o usuário clica no botão para registrar os produtos. A interface envia uma requisição para o endpoint `/polls/process_images/` no servidor Django.

2. **Contextualização para a IA (`utils.py`):** Antes de chamar a IA, o sistema executa um passo crucial: ele consulta o banco de dados para obter a lista de todos os produtos cadastrados e formata essas informações em um texto de contexto. Esse texto, que descreve os produtos disponíveis, é enviado junto com a imagem, servindo como um guia para que a inteligência artificial possa identificar os itens de forma mais precisa.

3. **Análise pela IA (`openai_api.py`):** A imagem capturada e o texto de contexto são enviados para o modelo de visão da OpenAI. A API processa a imagem e retorna um resultado estruturado (JSON) contendo os produtos que ela conseguiu identificar e suas respectivas quantidades.

4. **Validação com Padrão *Strategy* (`product_processing.py`):** Uma vez recebida a resposta da IA, o sistema aplica regras de negócio específicas utilizando o padrão de projeto *Strategy*. Foram criadas duas estratégias:

   - `FruitProcessingStrategy`: Aplicada quando o usuário indica que está pesando frutas. Essa regra verifica se apenas um tipo de fruta foi identificado pela IA, evitando que diferentes frutas sejam pesadas juntas.
   - `ProductProcessingStrategy`: Utilizada para produtos gerais, esta regra garante que nenhuma fruta tenha sido processada por engano neste fluxo.

   Essa abordagem torna o sistema flexível, permitindo que novas regras de validação sejam adicionadas no futuro sem alterar o fluxo principal.

5. **Validação Final e Resposta:** Após a aplicação da estratégia, o sistema realiza a verificação final cruzando a identificação da IA com a leitura da balança e envia a lista de produtos validados, o preço total e o peso para a interface do usuário.

2.4.2.1 INTERFACE COM O USUÁRIO

A interface gráfica foi projetada para proporcionar uma experiência clara, simples e responsiva. O usuário pode inserir produtos, acompanhar o carrinho de compras e finalizar a compra de forma intuitiva. A Figura 3 mostra a tela inicial do sistema, enquanto a Figura 4 exibe a interface após a identificação de produtos e cálculo do valor da compra.

2.4.3 ESTRUTURA DO BANCO DE DADOS

A persistência dos dados é realizada em um banco de dados SQLite, gerenciado pelo ORM do Django. O modelo principal (`src/polls/models.py`) é o **`Product`**, que armazena os atributos de cada item:

- `name`, `description`, `price`: Informações básicas do produto.
- `avg_weight`: Campo numérico que armazena o peso médio esperado do produto, utilizado na lógica de validação cruzada com o peso real da balança.
- `is_fruit`: Campo booleano que permite ao sistema diferenciar frutas de outros produtos e aplicar a estratégia de negócio correta.

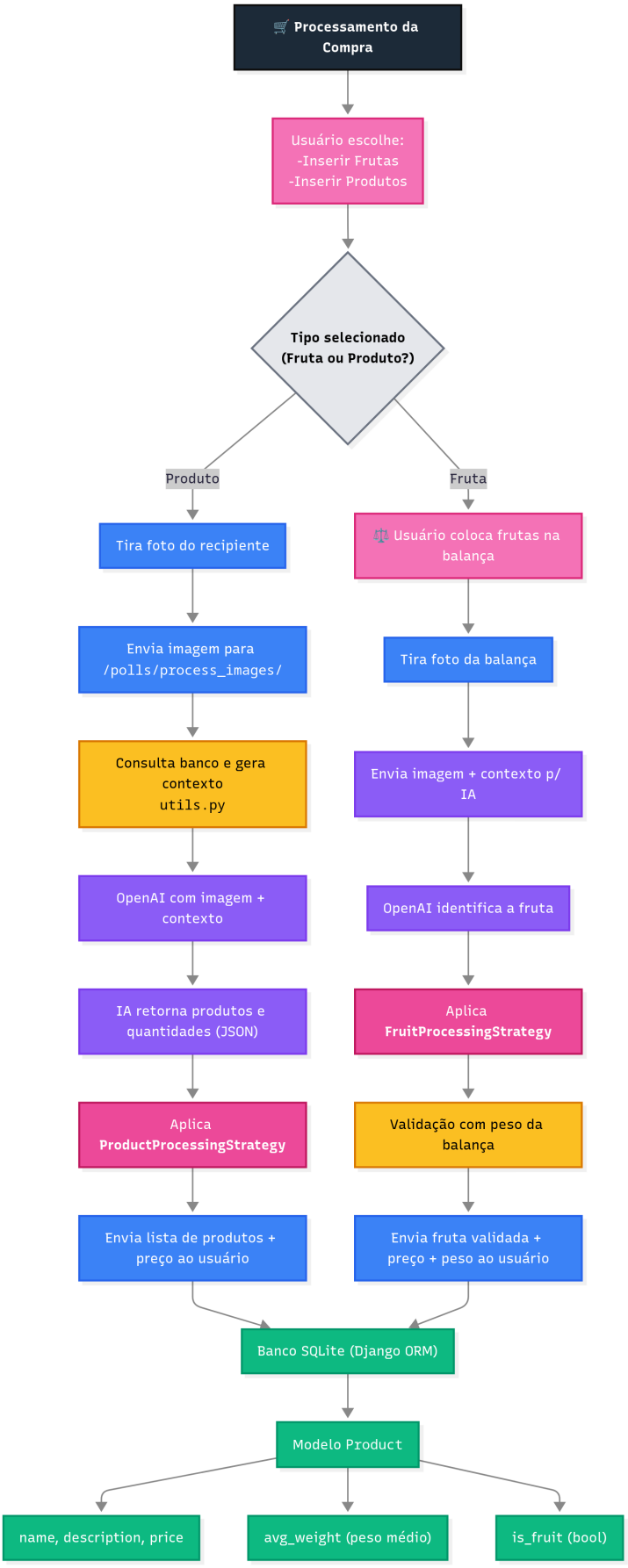

Figura 2: Fluxo de processamento da compra

Figura 3: Tela inicial da interface do Check.AI

Figura 4: Interface com produtos identificados no carrinho

3 CUSTOS

Grande parte dos materiais utilizados no projeto já estava disponível entre os integrantes do grupo, o que reduziu significativamente os custos totais em comparação com o valor estimado inicialmente. A Tabela 1 apresenta os principais componentes e seus respectivos preços para fins de documentação.

Material	**Preço**
Raspberry Pi 3 Model B+	R$405,00
HX711 5kg	R$10,00
Webcam Full HD 1080P	R$59,99
Câmera para *Raspberry Pi* OV5647	R$79,99
Filamento	R$180,90
Total	**R$734,88**

Tabela 1: Tabela de Materiais e Preços

4 DIFICULDADES

Durante o desenvolvimento do projeto, diversas dificuldades técnicas foram encontradas, sendo as principais relacionadas ao posicionamento da câmera, à iluminação do ambiente e à limitação de hardware do Raspberry Pi 3B+.

Inicialmente, o correto posicionamento da câmera mostrou-se essencial para garantir que todos os produtos fossem capturados adequadamente nas imagens. Foram necessários ajustes na angulação e altura até que se atingisse uma configuração que minimizasse erros de detecção. Além disso, a baixa luminosidade do ambiente comprometeu a nitidez das imagens em diversos momentos, afetando diretamente a eficácia do sistema. Como o reconhecimento dos itens depende da API da OpenAI, que realiza a identificação com base em características visuais e marcas descritas no banco de dados local, imagens mal iluminadas dificultam essa análise, resultando em falhas na classificação.

Outro desafio importante foi a calibração da balança digital. Embora tenha sido possível obter leituras consistentes em determinados pontos da superfície, outras regiões apresentaram valores discrepantes ou imprecisos. Essa limitação inviabilizou, em certos momentos, a checagem precisa dos produtos com base na comparação entre o peso real medido e os pesos esperados armazenados no banco de dados.

Por fim, enfrentamos sérios entraves relacionados ao desempenho do hardware do Raspberry Pi 3B+, que conta com uma quantidade limitada de memória RAM. Esse fator tornou praticamente inviável a execução de tarefas mais pesadas diretamente no microcontrolador, como o uso de editores de texto (por exemplo, VSCode) ou execução simultânea de múltiplos processos. Como solução, optou-se pela utilização de conexão via SSH entre o Raspberry Pi e um computador externo, permitindo o desenvolvimento remoto do projeto, com edição e execução do código a partir de uma máquina mais potente, contornando assim as limitações do dispositivo.

5 POSSÍVEIS MELHORIAS

Para aprimorar o projeto e viabilizar sua aplicação em contextos reais, algumas melhorias são recomendadas. Primeiramente, a instalação de LEDs ao redor da estrutura do equipamento contribuiria para uma iluminação uniforme e controlada, tornando o sistema independente das condições ambientais e aumentando a confiabilidade na captura das imagens.

A inclusão de uma segunda câmera também representa um avanço relevante. Com uma câmera capturando a visão frontal e outra posicionada na parte superior, seria possível reduzir significativamente os erros de detecção e aumentar a acurácia do sistema, especialmente em casos em que os produtos se sobrepõem ou ocultam detalhes importantes.

Outro ponto de melhoria seria a adoção de uma solução de inteligência artificial que pudesse ser executada localmente, sem depender da API da OpenAI. Embora o custo por requisição da API seja relativamente baixo, em uma aplicação comercial em larga escala, os custos operacionais podem se tornar significativos. A utilização de um modelo leve de detecção de objetos (como YOLOv5 ou YOLOv8, por exemplo), treinado com um conjunto de dados específico do supermercado, permitiria um funcionamento autônomo, com menores custos recorrentes e maior controle sobre o desempenho do sistema.

6 CONCLUSÃO

O projeto Check.AI teve como propósito desenvolver uma solução prática e inteligente para um problema recorrente nos supermercados: a dificuldade no uso dos caixas de autoatendimento, especialmente na pesagem e precificação de produtos sem código de barras, como frutas, legumes e pães. Ao final do desenvolvimento, podemos afirmar que o objetivo central foi alcançado: criamos um protótipo funcional que automatiza o processo de identificação e verificação de produtos por meio de visão computacional e sensores de peso.

Entretanto, o projeto também evidenciou algumas limitações importantes. O posicionamento da câmera e a iluminação ambiente tiveram impacto direto na acurácia do reconhecimento visual, exigindo reposicionamentos e ajustes manuais constantes. A calibração da balança digital mostrou-se sensível ao posicionamento dos produtos, comprometendo, em alguns casos, a precisão das medições. E talvez a limitação mais crítica: o desempenho restrito do Raspberry Pi 3B+, cuja baixa memória RAM inviabiliza o uso de ferramentas mais robustas localmente, como editores de código ou servidores de desenvolvimento, tornando necessário o uso de conexões SSH e desenvolvimento remoto para contornar tais restrições.

Essas dificuldades não invalidam os avanços do projeto, mas apontaram caminhos para melhorias. A adoção de uma iluminação artificial dedicada, como LEDs posicionados estrategicamente, o uso de múltiplas câmeras para melhor cobertura visual, a substituição ou complementação do Raspberry Pi por um hardware mais potente e a adoção de modelos de IA embarcados localmente são algumas sugestões que podem tornar o sistema ainda mais autônomo, preciso e eficiente.

Em resumo, o Check.AI mostra que é possível aplicar tecnologias modernas de visão computacional e automação de maneira criativa e integrada, resultando em uma experiência de compra mais rápida, precisa e acessível.

REFERÊNCIAS

OREL, D. **Supermarket self-checkout service quality, customer satisfaction, and loyalty: Empirical evidence from an emerging market**. 2014. Journal of Retailing and Consumer Services, Volume 21, Issue 2, Pages 118-129. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0969698913000829>. Acesso em: 29 de junho de 2025.

RONDáN, N. et al. **Self-Checkout System Prototype for Point-of-Sale using Image Recognition with Deep Neural Networks**. 2021. Proceedings of the IEEE URUCON 2021, Pages 217–222. Acesso em: 29 de junho de 2025.

WU, B.-F. et al. **An intelligent self-checkout system for smart retail**. 2016. Proceedings of the 2016 International Conference on System Science and Engineering (ICSSE), Pages 1–4. Acesso em: 29 de junho de 2025.