UNIVERSIDADE FEDERAL DO PAMPA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUCAS DA CRUZ BARBOSA

MINHA HORTA, MINHA VIDA: UMA SOLUÇÃO ESCALÁVEL PARA PRODUTORES BASEADA EM INTERNET ARTIFICIAL DAS COISAS

Bagé
2025

LUCAS DA CRUZ BARBOSA

MINHA HORTA, MINHA VIDA: UMA SOLUÇÃO ESCALÁVEL PARA PRODUTORES BASEADA EM INTERNET ARTIFICIAL DAS COISAS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Leonardo Bidese de Pinho
Coorientador: Sandro da Silva Camargo

Bagé
2025

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

B238m Barbosa, Lucas Da Cruz
MINHA HORTA, MINHA VIDA: UMA SOLUÇÃO ESCALÁVEL PARA PRODUTORES BASEADA EM INTERNET ARTIFICIAL DAS COISAS / Lucas Da Cruz Barbosa.
115 p.

Trabalho de Conclusão de Curso(Graduação)-- Universidade Federal do Pampa, ENGENHARIA DE COMPUTAÇÃO, 2025.
"Orientação: Leonardo Bidese de Pinho; Coorientação: Sandro da Silva Camargo".

1. Agricultura 5.0. 2. Agricultura Urbana. 3. Visão Computacional. 4. Computação em Névoa. 5. Design Science Research. I. Título.

LUCAS DA CRUZ BARBOSA

MINHA HORTA, MINHA VIDA: UMA SOLUÇÃO ESCALÁVEL PARA PRODUTORES BASEADA EM INTERNET ARTIFICIAL DAS COISAS

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Dissertação defendida e aprovada em: 8 de dezembro de 2025.

Banca examinadora:

_____

Prof. Dr. Leonardo Bidese de Pinho

Orientador

(UNIPAMPA)

_____

Prof. Dr. Carlos Michel Betemps

(UNIPAMPA)

_____

Prof. Dr. Julio Saraçol Domingues Jr.

(UNIPAMPA)

Assinado eletronicamente por **CARLOS MICHEL BETEMPS**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 18/12/2025, às 20:43, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **JULIO SARACOL DOMINGUES JUNIOR**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/12/2025, às 11:35, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **LEONARDO BIDESE DE PINHO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/12/2025, às 17:47, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Dedico este trabalho à minha versão de 19 anos, que decidiu se molhar para alcançar seus sonhos. Naquele momento, talvez ele não compreendesse a dimensão da tempestade que enfrentaria ao escolher tornar-se engenheiro de computação pela Unipampa. Mas, como diria Pablo Neruda, "você é livre para fazer suas escolhas, mas é prisioneiro das consequências".

Hoje, esta é a consequência daquelas escolhas.

AGRADECIMENTO

Eu poderia me estender aqui agradecendo a cada pessoa que colaborou, de alguma forma, com a minha formação. No entanto, vou me ater àqueles que tiveram participação mais direta, independentemente de como isso ocorreu. Em primeiro lugar, agradeço a minha irmã, Lecimar da Cruz Barbosa, que em um dos meus momentos mais obscuro foi luz, me lembrou o que é família, me lembrou que não estava sozinho, sem ela nada disso seria possível, sem ela este texto não seria o que é. Aos meus pais, por sempre me incentivarem a estudar e por me proporcionarem as melhores oportunidades que puderam me dar. Sem vocês, eu não teria chegado até aqui. Por fim, registro minha gratidão ao meu orientador e coorientador, pelo acompanhamento, paciência e apoio durante esta etapa final. Estendo ainda meus agradecimentos a todos os docentes que contribuíram, de maneira direta ou indireta, para minha formação acadêmica e pessoal.

Temos todo o tempo do mundo, então pouco tempo.

RESUMO

Este Trabalho de Conclusão de Curso (TCC) tem como propósito demonstrar a viabilidade do desenvolvimento de uma aplicação de visão computacional destinada a otimizar a agricultura urbana na região do Pampa brasileiro. O escopo do projeto ultrapassa a criação de um artefato de software, abrangendo um estudo exploratório que considera as tecnologias utilizadas, o sistema desenvolvido, seus usuários e o desenvolvedor enquanto agente do processo. O objetivo central consiste em aplicar capacidades de visão computacional, Tecnologias da Informação e Comunicação, Internet Artificial das Coisas e Computação em Névoa para aprimorar a eficiência da agricultura urbana em diferentes contextos brasileiros, marcados por desafios relacionados à produção, manejo e acesso a alimentos diante do crescimento populacional e da intensificação da urbanização. Nesse cenário, adotou-se a metodologia *Design Science Research* (DSR), a qual integra desenvolvimento prático e investigação científica. Embora o projeto inicial previsse uma solução completa para apoiar o produtor urbano em todas as fases da produção agrícola, integrando coleta de dados, análise por visão computacional e recomendações, as limitações de tempo e escopo próprias do TCC direcionaram o desenvolvimento para a detecção de frutos por meio de um aplicativo móvel, fornecendo apoio específico à etapa de colheita. Os resultados obtidos evidenciam a viabilidade técnica da abordagem adotada e indicam potenciais evoluções e caminhos de pesquisa para aprimoramentos futuros, contribuindo para o avanço de soluções tecnológicas destinadas à agricultura urbana.

Palavras-chave: Agricultura 5.0; Agricultura Urbana; Visão Computacional; Computação em Névoa; Design Science Research.

ABSTRACT

This Undergraduate Thesis (TCC) aims to demonstrate the feasibility of developing a computer vision application designed to optimize Urban Agriculture in the Brazilian Pampa region. The scope of this project goes beyond the creation of a software artifact, encompassing an exploratory study that considers the technologies employed, the developed system, its users, and the developer as an active agent in the process. The central objective is to apply computer vision capabilities, Information and Communication Technologies, Artificial Internet of Things, and Fog Computing to improve the efficiency of urban agriculture in different Brazilian contexts, which face pressing challenges related to food production, management, and access due to population growth and increasing urbanization. In this scenario, the Design Science Research (DSR) methodology was adopted, integrating practical development with scientific investigation. Although the initial project envisioned a comprehensive solution to support urban growers throughout all stages of agricultural production—integrating data collection, computer vision analysis, and recommendation generation—time and scope constraints inherent to the thesis stages led the development to focus specifically on fruit detection through a mobile application, offering targeted support for the harvesting stage. The results obtained demonstrate the technical feasibility of the adopted approach and highlight potential evolutions and research paths for future enhancements, contributing to the advancement of technological solutions for urban agriculture.

Keywords: Agriculture 5.0; Urban Agriculture; Computer Vision; Fog Computing; Design Science Research.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
AIoT	Artificial Intelligence of Things
AP	Average Precision
API	Application Programming Interface
AR	Average Recall
CPU	Central Processing Unit
COCO	Common Objects in Context
CSV	Comma Separated Values
DSR	Design Science Research
FCN	Fully Connected Network
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet das Coisas
IP	Internet Protocol
IU	Interface de Usuário
MSER	Maximally Stable External Regions
MPCNN	Multi-Path Convolutional Neural Network
MVP	Produto Mínimo Viável
RoIs	Regiões de Interesse
SVM	Support Vector Machine
TIC	Tecnologias da Informação e Comunicação
UNIPAMPA	Universidade Federal do Pampa

SUMÁRIO

1 INTRODUÇÃO

Conforme Alexandratos e Bruinsma (2012), a população global apresentou um crescimento contínuo na demanda por recursos, especialmente água, energia e alimentos. Esse fenômeno, embora não seja recente, foi intensificado por diversos fatores, incluindo o aumento populacional, a elevação da expectativa de vida e a aceleração da urbanização. Como consequência, projetou-se a necessidade de expansão da produção de alimentos em 35% até 2030 (MASSRUHÁ *et al.*, 2020). No entanto, de acordo com Gonçalves e Ventura (2021), o cenário brasileiro foi "marcado pelo contraste entre fome e desperdício, onde existe a produção, porém esta não é acessível para todos". Nesse contexto, a Agricultura Urbana — praticada dentro ou ao redor das cidades (zonas periurbanas), com foco na produção para consumo próprio ou comercialização em pequena escala — consolidou-se como uma estratégia para enfrentar estes desafios. Ao longo dos séculos, os seres humanos empregaram seus recursos para garantir a sobrevivência das gerações presentes e futuras. Como parte desse processo, as sociedades mais avançadas passaram a demonstrar crescente preocupação com a utilização sustentável dos recursos, fundamentadas nos progressos científicos e na implementação desse conhecimento por meio do avanço tecnológico. As Tecnologias da Informação e Comunicação (TIC) constituíram o conjunto de tecnologias destinadas à produção, acesso, transmissão e compartilhamento de informações, além de viabilizarem a comunicação entre pessoas (RODRIGUES, 2016).

Com base nessas inovações, emergiu a Agricultura 5.0, um movimento voltado à integração digital completa do setor agropecuário, incorporando tecnologias inteligentes em todas as fases da cadeia produtiva para aprimorar sua eficiência (MASSRUHÁ *et al.*, 2020). Nesse cenário, destacou-se o conceito de Internet Artificial das Coisas (*Artificial Intelligence of Things* — AIoT), entendido como a evolução da Internet das Coisas por meio da incorporação de algoritmos de inteligência artificial aos dispositivos conectados. A AIoT surgiu da necessidade de não apenas coletar dados no campo, mas também processá-los de forma inteligente tanto localmente, com recursos embarcados, quanto em plataformas de computação distribuída e na nuvem, permitindo análises avançadas, integração de grandes volumes de dados e tomada de decisão contínua. Essa abordagem foi fortalecida pelo avanço das técnicas de Visão Computacional e pelo uso de arquiteturas distribuídas, como a Computação em Névoa (*Fog Computing*), que ampliaram a capacidade de resposta, escalabilidade e autonomia dos sistemas.

A Visão Computacional consiste na área destinada a capacitar máquinas a identificar padrões em dados visuais e extrair informações relevantes. Essa tecnologia empregou câmeras e computadores para reconhecer, rastrear e mensurar objetos, permitindo seu posterior processamento. Com seu avanço, tornou-se amplamente utilizada na automação agrícola, desempenhando papel essencial no progresso do setor (TIAN *et al.*, 2020). Por sua vez, a Computação em Névoa foi concebida para atuar entre a Internet das Coisas (IoT) e a nuvem, fornecendo uma camada computacional intermediária capaz de coletar, agregar e processar dados provenientes de dispositivos IoT. A integração entre névoa e nuvem reduziu a transferência de dados e mitigações de comunicação com a nuvem, além de diminuir a latência, uma vez que os recursos em névoa se encontram mais próximos da borda (BITTENCOURT *et al.*, 2018).

1.1 Problema de Pesquisa

Diante desse cenário, tornou-se relevante pesquisar e desenvolver alternativas tecnológicas de baixo custo que auxiliassem no aumento da eficiência de sistemas produtivos baseados em Agricultura Urbana, utilizando técnicas de Visão Computacional e Computação em Névoa para apoiar produtores em diferentes etapas do ciclo produtivo. Mais especificamente, propôs-se o desenvolvimento de um aplicativo para dispositivos móveis que permitisse a uma central auxiliar produtores no gerenciamento de cultivos, desde a pré-produção até a pós-produção. Isso incluiria tanto a supervisão, por parte da central, do desempenho de grupos de produtores, quanto o suporte às decisões de manejo em cada etapa da produção.

Como o escopo do problema é amplo, optou-se por concentrar esforços em uma das etapas do processo de manejo das hortas urbanas. Para isso, projetou-se um sistema no qual os agricultores urbanos são registrados em uma base de dados, tal que cada um pode manter um ou mais cultivos. As imagens georreferenciadas desses cultivos são coletadas regularmente, seja pelo próprio produtor, seguindo orientações padronizadas fornecidas pela central, seja por um agente responsável pela coleta. O objetivo dessa padronização foi facilitar a identificação automática do estágio de desenvolvimento do cultivo e compará-lo ao progresso esperado, caso o manejo recomendado fosse seguido, possibilitando assim a geração de recomendações personalizadas. Além disso, o sistema permite a criação de relatórios de produção e evolução, futuramente o aplicativo visa a geração de relatórios de previsões por meio de técnicas de mineração de dados, realizar

previsões de produção, monitorando os resultados e ajustando as estimativas até alcançar maior precisão.

Por fim, é importante destacar que o projeto ultrapassou a simples criação de um artefato de software. Tratou-se de uma pesquisa exploratória na qual as tecnologias empregadas, o artefato desenvolvido, os usuários do sistema e o próprio desenvolvedor foram considerados elementos centrais do estudo. Optou-se pelo desafio de aplicar a metodologia *Design Science Research* (DSR), apresentada no Capítulo 2 "Referencial teórico", mais precisamente na seção 2.1 "*Design Science Research*".

1.2 Objetivo Geral

O objetivo central deste projeto consistiu em analisar, com base nos princípios da Agricultura 5.0, a aplicabilidade de Tecnologias da Informação e Comunicação para apoiar a gestão de hortas. Essa análise culminou no desenvolvimento de um aplicativo móvel integrado a métodos de Visão Computacional e Computação em Névoa.

Com uma perspectiva ampla da produção agrícola distribuída, o aplicativo armazenou o histórico de dados dos produtores, possibilitando a geração de relatórios de produção e evolução. Estes relatórios fornecem informações valiosas tanto para o produtor que gerencia sua horta quanto para a central que supervisiona diversos produtores.

Devido às limitações de tempo, este Trabalho de Conclusão de Curso contemplou a implementação de apenas um dos módulos previstos no sistema idealizado: o módulo dedicado à detecção de frutos e à identificação do estágio de maturação, com foco específico na cultura do tomate.

1.3 Objetivos Específicos

O objetivo geral desdobrou-se nos seguintes objetivos específicos de curto prazo (TCC-I) e médio prazo (TCC-II):

- Avaliar a viabilidade do uso de técnicas de Visão Computacional para detecção e contagem de frutos (TCC-I);
- projetar, desenvolver e validar um artefato funcional com processamento em nuvem voltado ao apoio à colheita por meio da detecção automatizada de frutos (TCC-I);

- adaptar a solução desenvolvida para uma arquitetura baseada em Computação em Névoa (*Fog Computing*), visando escalabilidade e distribuição do processamento (TCC-II);
- aprimorar o aplicativo e o modelo de detecção de frutos com base nos resultados obtidos na etapa anterior (TCC-II).

1.4 Metodologia

Para a metodologia de pesquisa, optou-se por seguir a abordagem de Design Science Research (DSR), a qual é iterativa e orientada para a solução de problemas complexos. Para a revisão do estado da arte, optou-se por realizar uma Revisão Sistemática da Literatura (RSL). Já para a metodologia de desenvolvimento do artefato, o método cascata foi aplicado para a criação da versão inicial do produto mínimo viável (MVP) do aplicativo, enquanto o método ágil foi empregado na fase de aprimoramento e manutenção, em razão de sua abordagem iterativa e incremental. Em ambas as situações, previu-se a elaboração de documentação e manuais para a ferramenta.

1.5 Organização do Trabalho

O restante do trabalho está organizado como segue: o Capítulo 2, “Referencial teórico”, que detalha a abordagem adotada para condução da investigação e consolida a literatura relevante, contemplando computação em névoa, visão computacional e aprendizado de máquina aplicados à agricultura, além dos trabalhos correlatos e das decisões de projeto; o Capítulo 3, “Minha horta, minha vida”, apresenta a evolução da solução proposta, iniciando com sua versão em nuvem e sua migração para a arquitetura em névoa, abrangendo análise, modelagem, casos de uso, diagramas, armazenamento de dados e tecnologias empregadas. O Capítulo 4, “Resultados e discussões”, descreve os achados obtidos nos testes realizados e a verificação da ferramenta desenvolvida; por fim, o Capítulo 5 “Considerações finais”, conclui o estudo e aponta possíveis direções para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta a fundamentação teórica para o desenvolvimento do artefato proposto. Inicialmente, descreve-se o processo de Revisão Sistemática da Literatura (RSL), conduzido segundo o protocolo PRISMA, que permitiu identificar tecnologias, métodos e abordagens empregados em pesquisas recentes relacionadas à Agricultura Inteligente e à Agricultura de Precisão. Essa revisão forneceu o embasamento técnico para a definição dos requisitos funcionais e não funcionais do artefato, bem como para as decisões de arquitetura adotadas ao longo do projeto.

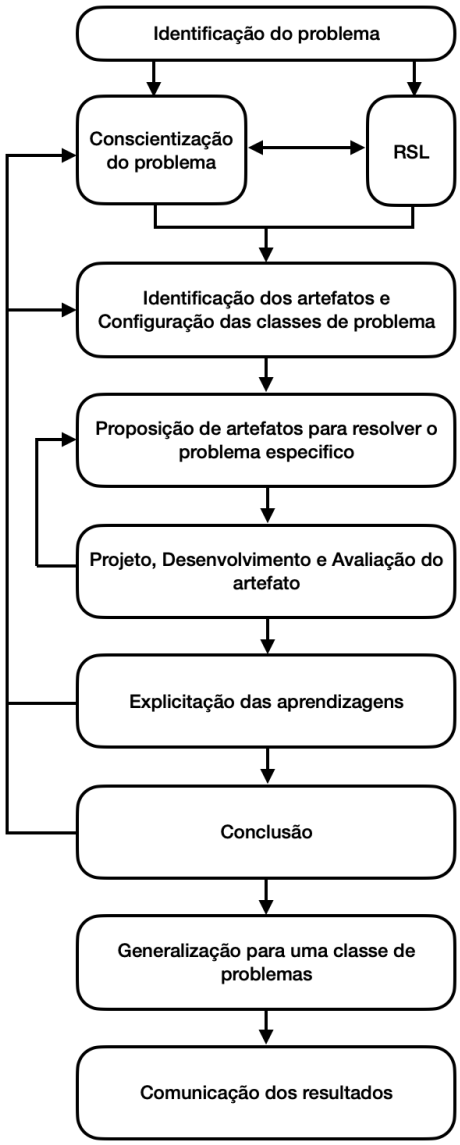
Adicionalmente, apresenta-se o *Design Science Research Model* (DSR), constitui uma síntese de diferentes abordagens metodológicas voltadas à pesquisa orientada à solução de problemas práticos. O modelo é composto por um conjunto de elementos que devem estar coerentemente interligados, assegurando rigor científico ao processo de investigação e desenvolvimento do artefato (PIMENTEL; FILIPPO; SANTOS, 2020).

2.1 *Design Science Research*

O modelo metodológico *Design Science Research* (DSR), que orienta o desenvolvimento do artefato, estrutura o processo de investigação em nove etapas (Figura 1) iterativas de identificação do problema, concepção, construção, avaliação e comunicação da solução, servindo como referência para a modelagem e validação da proposta desenvolvida neste trabalho.

A etapa de identificação do problema deste trabalho está relacionado à necessidade de ampliar a produção de alimentos para atender ao crescimento da demanda populacional global. No contexto brasileiro, observa-se um descompasso entre a produção e a disponibilidade de alimentos, resultando em cenários simultâneos de desperdício e insegurança alimentar. Identificou-se, ainda, a ausência de sistemas de suporte direcionados a indivíduos interessados em cultivar hortas domésticas, seja para consumo próprio ou como fonte complementar de renda. A partir dessa constatação, tornou-se evidente a necessidade de repensar métodos e técnicas de produção agrícola, especialmente em ambientes urbanos e periurbanos. Embora o estudo tenha sido conduzido com foco em regiões periurbanas, a solução proposta apresenta potencial de aplicação em contextos urbanos e rurais.

Figura 1 – Elementos centrais do modelo DSR



Fonte: Adaptado de Pimentel, Filippo e Santos (2020)

Na etapa de conscientização do problema, foram definidos os requisitos específicos do artefato, alinhados aos objetivos do estudo e às funcionalidades esperadas. Inicialmente, identificaram-se as funcionalidades necessárias ao funcionamento da solução, seguidas de sua priorização, de modo a estabelecer quais seriam implementadas no protótipo inicial e quais seriam direcionadas a versões futuras. Paralelamente, foram definidos os requisitos técnicos, incluindo linguagens de programação, bibliotecas e critérios de compatibilidade com sistemas existentes. Também foram estabelecidos critérios de sucesso e métricas de avaliação, formalizados em documento de especificação.

De forma concomitante, realizou-se uma Revisão Sistemática da Literatura (RSL) (KITCHENHAM; CHARTERS, 2007), com o objetivo de compreender o estado da arte e identificar soluções existentes relacionadas à agricultura urbana, à visão computacional e ao uso de tecnologias de computação distribuída. Para a condução da RSL, adotou-se o protocolo *Preferred Reporting Items for Systematic Reviews and Meta-Analyses* (PRISMA) (MOHER *et al.*, 2009), cujo fluxo completo encontra-se ilustrado na Figura 2.

As questões de pesquisa que orientaram o processo de seleção e análise dos estudos foram as seguintes:

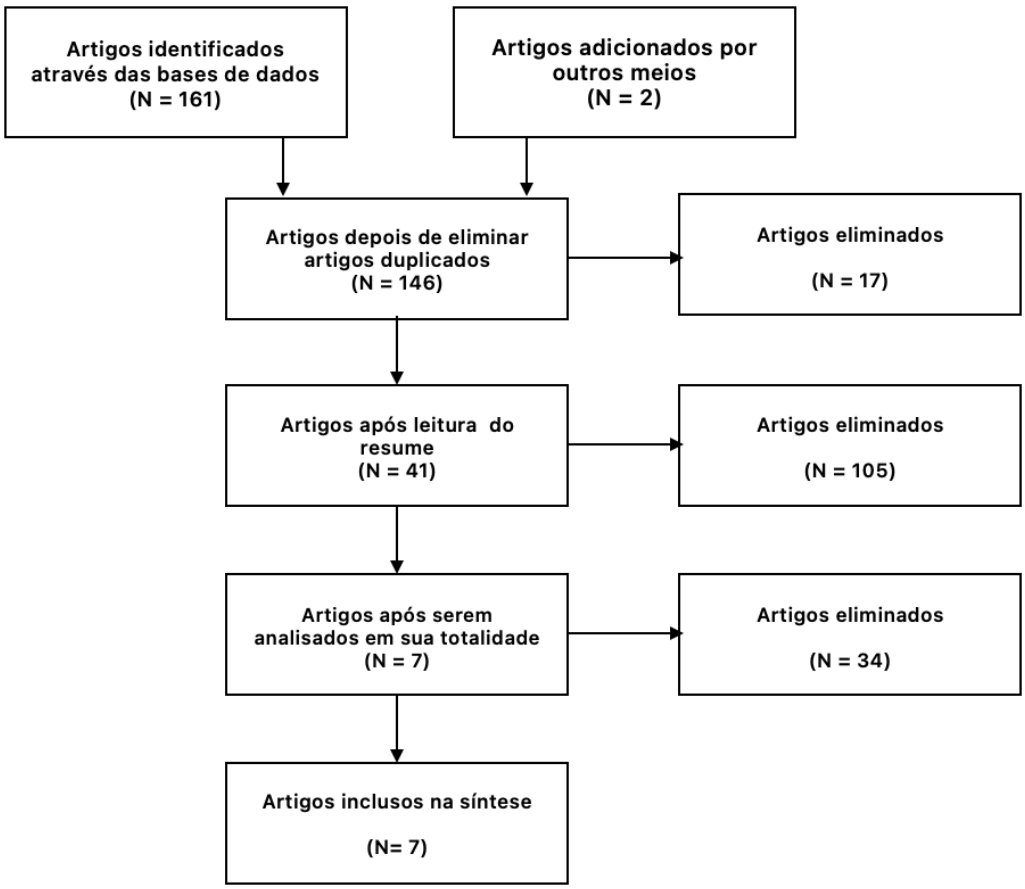
1. O conteúdo da pesquisa era viável para imagens coletadas por meio de celular?
   Essa questão buscou identificar a viabilidade de aplicar as técnicas analisadas a imagens capturadas por dispositivos móveis.
2. Quais tecnologias eram utilizadas e qual era a finalidade das pesquisas para Agricultura Inteligente?
   A questão permitiu compreender quais recursos tecnológicos eram empregados e para quais finalidades eram aplicados.
3. Quais etapas eram contempladas nas pesquisas?
   Essa análise possibilitou identificar abordagens distintas para etapas semelhantes e selecionar aquelas mais adequadas ao escopo deste estudo.
4. Quais técnicas de análise de imagem eram aplicadas?
   A questão permitiu compreender as metodologias adotadas e selecionar as mais compatíveis com a solução proposta.
5. Quais técnicas complementares às CNNs eram aplicadas nas pesquisas?
   O objetivo foi identificar métodos capazes de contribuir para a análise de imagens.
6. Qual a finalidade do uso de Cloud ou Fog Computing nas arquiteturas analisadas?
   A questão visou compreender o papel dessas tecnologias nas arquiteturas avaliadas e como elas poderiam apoiar o desenvolvimento do artefato.

Para a localização dos estudos, definiu-se uma *string* de busca composta por termos tecnológicos e elementos do domínio agrícola, incluindo *fog computing*, *deep learning*, *computer vision*, *machine learning*, *vegetation* e *agriculture*. Apenas artigos publicados a partir de 2020 foram considerados inicialmente. As bases consultadas incluíram Scopus, IEEE Xplore, MDPI, ScienceDirect e Engineering Village.

No total, foram identificados 161 artigos, dos quais 17 foram removidos por duplicidade. Após a análise de títulos, palavras-chave e resumos, 105 estudos foram

excluídos por não se alinharem ao escopo da pesquisa. Os 39 artigos remanescentes passaram por triagem detalhada, resultando na seleção final de 5 estudos plenamente aderentes às questões de pesquisa. Adicionalmente, dois artigos anteriores a 2020 foram incluídos devido à sua relevância técnica e alinhamento conceitual. Esses estudos são discutidos na Seção 2.5.

Figura 2 – Diagrama do processo de seleção dos artigos segundo o PRISMA



Fonte: Adaptado de Kalyani e Collier (2021)

Como resultado da RSL, foram definidos os requisitos do protótipo, classificados em funcionais e não funcionais. Os requisitos funcionais, organizados por ordem de prioridade, incluem:

1. Registro do produtor;
2. Registro de cultivo;
3. Visualização dos cultivos cadastrados;
4. Registro de imagens para análise;
5. Análise de imagens em ambiente de Nuvem;

6. Registro dos resultados das análises;
7. Visualização dos resultados pelo produtor;
8. Geração de relatórios dos cultivos;
9. Análise de imagens em ambiente de Névoa.

Os requisitos não funcionais asseguram que o sistema opere de forma eficiente e confiável, considerando a escolha das linguagens de programação e das plataformas pretadoras de serviço e viabilizam e reforçam aspectos como segurança, escalabilidade, confiabilidade, desempenho e usabilidade.

Na etapa de identificação dos artefatos e configuração das classes de problema, identificam-se os artefatos necessários para a solução do problema, como um aplicativo móvel que utilizasse técnicas de visão computacional para detecção e monitoramento de frutos. Identificaram-se módulos responsáveis pela detecção do estágio de desenvolvimento dos frutos, pela contagem e pela localização em imagens. Assim passamos para a configuração das classes de problemas, buscou-se generalizar a solução proposta para diferentes cenários, avaliando sua viabilidade em contextos semelhantes. Os artefatos foram desenvolvidos com base na RSL e avaliados segundo critérios definidos na etapa anterior, considerando eficácia e eficiência.

A etapa de proposição do artefato envolveu a concepção de módulos autônomos integrados a um aplicativo móvel, permitindo que uma central de processamento oferecesse suporte a produtores urbanos no manejo de plantações. O levantamento do estado da arte possibilitou delimitar as funcionalidades do artefato e sua modelagem.

A etapa de desenvolvimento ocorreu de forma iterativa, contemplando coleta e padronização de dados, treinamento e validação de modelos, bem como a definição da arquitetura do sistema. Os dados foram divididos em 75% para treinamento e 25% para validação, garantindo robustez e capacidade de generalização dos modelos. A arquitetura do aplicativo foi projetada considerando o fluxo de interação entre usuário, banco de dados e *endpoints* de inferência.

A etapa avaliação do artefato foi conduzida inicialmente pelo desenvolvedor e, posteriormente, contou com a participação de um usuário externo. Caso resultados insatisfatórios fossem observados, retornava-se às etapas anteriores para ajustes.

Após a avaliação, passamos para a etapa de explicitação do aprendizado, onde identificaram-se as lições aprendidas ao longo do processo, incluindo decisões técnicas, metodológicas e práticas. Seguimos para etapa de conclusão, onde sintetizaram-se os resultados obtidos, as principais decisões tomadas e as limitações identificadas.

Assim como em etapas anteriores, constatou-se a possibilidade de retorno a fases iniciais para ajustes. Seguindo para a etapa de generalização para uma classe de problemas, avaliou-se a aplicabilidade do artefato em contextos semelhantes, considerando adaptações necessárias para diferentes cenários. Finalmente, na etapa de comunicação dos resultados, documentaram-se as descobertas, metodologias e conclusões do estudo, visando disseminar o conhecimento adquirido e contribuir para a comunidade acadêmica e profissional.

2.2 Visão Computacional

A Visão Computacional é uma área da computação dedicada à automatização da extração e interpretação de informações visuais, permitindo que sistemas computacionais compreendam imagens e vídeos. Inicialmente baseada em técnicas clássicas de processamento de imagens, como filtros, transformações e operações morfológicas, a área evoluiu significativamente com o avanço da Inteligência Artificial.

Entre as bibliotecas tradicionais, destaca-se o OpenCV (OpenCV Team, 2024), amplamente utilizado para detecção de bordas, segmentação e rastreamento de objetos. Atualmente, as Redes Neurais Convolucionais (CNN) desempenham papel central na Visão Computacional aplicada à Agricultura Inteligente, viabilizando análises robustas em cenários complexos (LIN; ADETOMI; ARSLAN, 2021).

As CNN apresentam elevada capacidade de escalabilidade, podendo ser implementadas tanto em dispositivos móveis quanto em ambientes de alto desempenho. Também são investigadas em FPGA, visando otimizações em ambientes com recursos limitados (CZYMMEK *et al.*, 2023). Neste trabalho, destacam-se as arquiteturas Multi-Path CNN e Mask R-CNN, amplamente empregadas em aplicações agrícolas.

A arquitetura Multi-Path CNN (MPCNN) se destacado em ambientes agrícolas complexos ao utilizar caminhos paralelos para extração de múltiplas características visuais. Estudos demonstram sua eficácia na diferenciação entre frutos e elementos do ambiente, alcançando elevados índices de precisão (MAO *et al.*, 2020).

Já o Mask R-CNN combina detecção de objetos e segmentação de instâncias, permitindo segmentação ao nível de píxel. Derivado do Faster R-CNN (REN *et al.*, 2015), o modelo apresenta alto desempenho em cenários com sobreposição e oclusão de objetos, sendo amplamente adotado em aplicações de Agricultura Inteligente (YU *et al.*, 2019).

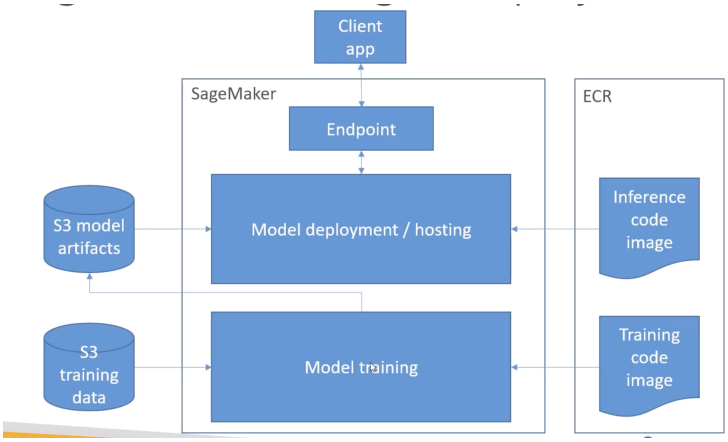
O Common Objects in Context (COCO) é um dos principais referenciais utilizados para a avaliação e validação de modelos de visão computacional, especialmente em tarefas de detecção, segmentação e reconhecimento de objetos (LIN *et al.*, 2014). Além de disponibilizar um amplo conjunto de dados anotados, fornece um protocolo padronizado de métricas e ferramentas de avaliação, amplamente adotado pela comunidade científica, o que possibilita a comparação justa e reprodutível entre diferentes abordagens e arquiteturas de modelos. Nesse contexto, o uso das métricas propostas, como Average Precision (AP) e Average Recall (AR) em múltiplos limiares de Intersection over Union (IoU), contribui para uma análise mais robusta do desempenho dos modelos, indo além de avaliações pontuais.

2.3 Modelos de Computação Distribuída Escaláveis

A Computação em Nuvem é um paradigma que oferece recursos computacionais como serviço, com suporte à virtualização, escalabilidade e elasticidade (ELAZHARY, 2019). Na Agricultura Inteligente, possibilita armazenamento remoto, processamento intensivo e integração de sistemas. Entretanto, desafios como latência e dependência de conectividade justificam o uso de paradigmas complementares.

A Computação em Névoa atua como camada intermediária entre dispositivos finais e a Nuvem, reduzindo latência e consumo de banda. Já a Computação de Borda desloca o processamento para a extremidade da rede, sendo adequada a aplicações que exigem respostas em tempo real (SHI *et al.*, 2016).

Figura 3 – Diagrama de fluxo do Amazon SageMaker



Fonte: Udemy (2025)

O Amazon SageMaker é uma plataforma de *machine learning* que oferece suporte completo ao ciclo de vida de modelos, desde o treinamento até a implantação. Sua integração com serviços da AWS possibilita a construção de pipelines escaláveis e reprodutíveis. A Figura 3 apresenta uma visão geral desse fluxo.

Na área da segurança dos modelos de computação distribuída, destaca-se o Wazuh, uma plataforma open source voltada para monitoramento de segurança, detecção de intrusões e gestão de eventos, oferecendo suporte completo ao ciclo de coleta, correlação, análise e resposta a incidentes. A solução integra agentes distribuídos, mecanismos de análise centralizada e dashboards para visualização, possibilitando a construção de uma arquitetura escalável e confiável para ambientes on-premises, em nuvem ou híbridos. Sua integração com componentes como Elastic/OpenSearch e APIs externas permite a automatização de fluxos de segurança e a consolidação de logs e alertas em tempo real.

2.4 Modelo de Aceitação de Tecnologia

O Modelo de Aceitação de Tecnologia (TAM), proposto por Davis (1989), estabelece que a adoção de tecnologias é influenciada pela utilidade percebida e pela facilidade percebida de uso. Quanto maior a facilidade de uso, maior tende a ser a utilidade percebida, impactando diretamente a intenção comportamental de uso.

Neste trabalho, o TAM é utilizado como referência para avaliar a aceitação do artefato proposto, fornecendo subsídios para melhorias futuras na interface, nas funcionalidades e na experiência do usuário.

2.5 Trabalhos Correlatos

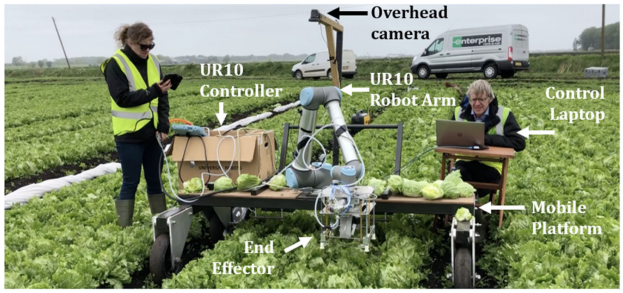
Nesta seção, apresentam-se os estudos correlatos e sua relevância para a compreensão das tecnologias empregadas, fornecendo subsídios para o embasamento do projeto da solução. Esses trabalhos possibilitam responder às questões de pesquisa formuladas anteriormente, a partir dos *insights* obtidos por meio da revisão da literatura.

O estudo de Birrell *et al.* (2020), direcionado à colheita automatizada de alface iceberg, apresenta um panorama da evolução das tecnologias de visão computacional aplicadas à agricultura. Os métodos tradicionais baseavam-se em características manuais, como filtros de borda e classificadores simples, utilizados para detectar culturas como

batata, uva, pepino e brócolis. Embora úteis, tais abordagens eram limitadas diante da complexidade existente nas condições agrícolas reais. Com o avanço de modelos de *deep learning*, como o You Only Look Once (YOLOv3), permitiu-se detectar e classificar alfaces sob diferentes condições, com maior eficiência. O sistema Vegebot (Figura 4) consiste em um braço robótico, câmeras, um cortador e sensores de força.

Para tal, criou-se um banco de imagens específico, composto por diferentes cenários de campo, que serviu de base para o treinamento da rede neural. Durante os testes, o sistema apresentou alta precisão na localização das plantas e em sua classificação em três categorias: prontas para colheita, imaturas e doentes.

Figura 4 – Sistema de colheita Vegebot



Fonte: Birrell *et al.* (2020)

A colheita foi realizada por um mecanismo de dois estágios, envolvendo agarramento suave e corte com altura ajustada, controlado por *feedback* de força. Apesar do desempenho satisfatório na identificação e classificação, a eficiência do processo de colheita ainda necessita de melhorias: verificou-se taxa de sucesso de 52% no destacamento e 38% de danos às folhas externas (Figura 5). Em síntese, o estudo evidencia que, mesmo com limitações, o Vegebot constitui um avanço relevante rumo à automação agrícola, ao integrar visão computacional, aprendizado profundo e robótica na resolução dos desafios da colheita de precisão.

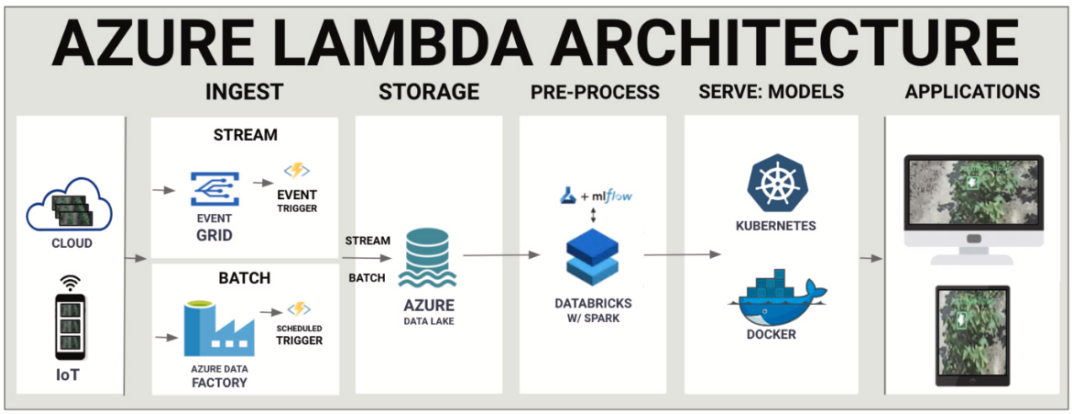
Figura 5 – Resultados da colheita do Vegebot

Fonte: Birrell *et al.* (2020)

O artigo de Issac *et al.* (2023) apresenta uma análise abrangente das metodologias emergentes e consolidadas no campo da agricultura inteligente, com ênfase na convergência entre *big data*, computação em nuvem e visão computacional. Métodos baseados em píxeis e características visuais simples, como o uso de OpenCV para segmentação por cor, foram empregados para identificar cápsulas de algodão.

Com o passar do tempo, modelos mais robustos, como o YOLOv5, passaram a ser utilizados para detectar estruturas mais complexas, como flores e cápsulas, proporcionando maior acurácia e velocidade no reconhecimento de objetos agrícolas. O artigo destaca a relevância da arquitetura de *big data*, denominada Azure Lambda (Figura 6), pelo uso de funções *serverless* da plataforma Microsoft Azure, como solução eficiente para o processamento de grandes volumes de dados agrícolas, incluindo informações meteorológicas e de produtividade.

Figura 6 – Estrutura Azure Lambda



Fonte: Issac *et al.* (2023)

Soluções baseadas em computação em nuvem, como o sistema WalleSmart, distinguem-se por sua escalabilidade e viabilidade econômica, embora ainda haja poucas pesquisas práticas sobre a integração plena de plataformas como o Microsoft Azure nesse contexto. Ademais, a proposta de um pipeline de dados capaz de realizar tanto o processamento em tempo real quanto em lote representa um avanço significativo para aplicações em ambientes agrícolas dinâmicos.

Na arquitetura Lambda implementada na plataforma Azure para aplicações em agricultura inteligente, o fluxo de dados é estruturado em quatro etapas principais: *ingestion*, *store*, *pre-process* e *serve-model*. A etapa de *ingestion* subdivide-se em dois fluxos: *stream* e *batch*. O fluxo *stream*, também chamado de *speed layer*, destina-se

à análise em tempo real dos dados conforme são gerados no campo. Ele aciona automaticamente *pipelines* assim que novas imagens são armazenadas, utilizando gatilhos por eventos em serviços como o *Blob Storage*, integrando-se ao Azure Event Grid e ao Azure Data Factory. Tal fluxo é adequado para decisões rápidas e para lidar com grandes volumes de dados com baixa latência.

O fluxo *batch*, é responsável pelo processamento periódico de grandes volumes de dados armazenados em um *data lake*. Seu acionamento ocorre de forma programada, por meio de expressões *CRON*, possibilitando o reprocessamento de dados históricos ou a preparação de conjuntos de dados para o treinamento de modelos. Esse fluxo permite ainda a segmentação dos dados em lotes menores, favorecendo o treinamento paralelo de modelos com menor consumo computacional.

Após a ingestão, os dados são organizados e armazenados de forma persistente na etapa *store*. Em seguida, passam pela etapa *pre-process*, na qual são limpos e transformados para uso analítico. Por fim, na etapa *serve-model*, os modelos previamente treinados realizam as inferências sobre os dados processados. A arquitetura híbrida, que adota ingestão dual (*stream* e *batch*), proporciona flexibilidade, escalabilidade e respostas rápidas, demonstrando-se eficaz para ambientes agrícolas orientados por dados.

O estudo reforça, portanto, a necessidade de soluções integradas que combinem inteligência artificial, computação em nuvem e análise de dados a fim de superar os desafios contemporâneos da agricultura de precisão.

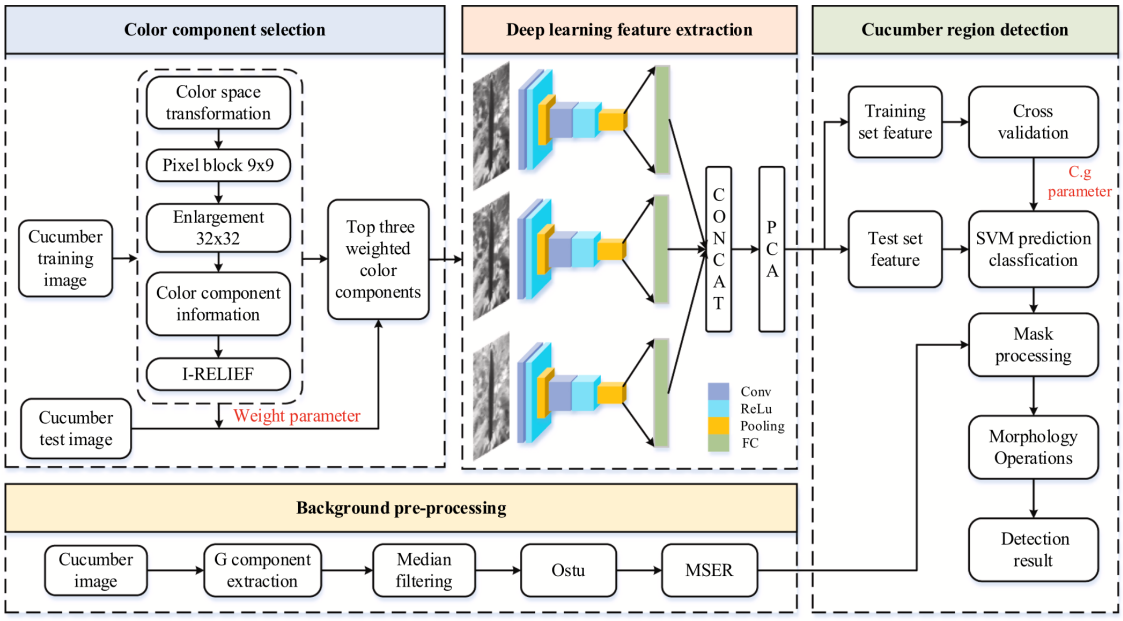
O artigo de Mao *et al.* (2020) apresenta uma abordagem mais sofisticada de detecção de objetos, desenvolvida para culturas específicas. O caso analisado refere-se à colheita automatizada de frutas verdes, como o pepino, que impõe desafios adicionais devido à semelhança cromática com folhas e caules (Figura 7).

Figura 7 – Imagens das amostragens dos pepinos em ambiente natural complexo

Fonte: Mao *et al.* (2020)

Diversas abordagens têm sido propostas para superar tais obstáculos, incluindo análise de textura, espectroscopia e redes neurais profundas, embora enfrentem limitações em cenários ambientais complexos. O autor propõe uma abordagem que combina extração de características e aprendizado profundo (Figura 8), utilizando o algoritmo I-RELIEF para seleção de componentes de cor e uma rede neural convolucional (LeNet-5) integrada a uma *Support Vector Machine* (SVM) para classificação.

Figura 8 – Fluxo MP-CNN



Fonte: Mao *et al.* (2020)

Os dados foram coletados em estufas na China, onde fundos visuais complexos e variações de iluminação representam desafios significativos. A seleção dos componentes de cor mais relevantes, aliada ao pré-processamento com filtragem e segmentação, permite melhorar substancialmente o desempenho da detecção.

Além disso, o uso de *Multi-Path Convolutional Neural Network* (MPCNN) e técnicas de remoção de fundo, como *maximally stable extremal regions* (MSER), intensifica a precisão na segmentação das regiões de interesse. Os resultados (Figura 9) indicam taxa de reconhecimento superior a 90%, com baixa incidência de falsos positivos, evidenciando o potencial da abordagem em sistemas autônomos de colheita de pepinos.

No campo do sensoriamento remoto, o estudo de Csillik *et al.* (2018) apresenta uma revisão abrangente da literatura sobre o uso destas tecnologias na agricultura de precisão. Os avanços em algoritmos de segmentação e extração de características a partir de imagens de alta resolução promoveram melhorias significativas na acurácia das

análises, especialmente com a transição de métodos baseados em píxeis para abordagens orientadas a objetos. Nesse contexto, os *Unmanned Aerial Vehicles* (UAV) emergem como ferramentas eficazes e acessíveis para a coleta de imagens em alta definição.

Figura 9 – Resultados do MP-CNN

(a) imagem original; (b) resultado da segmentação;
(c) area real dos pepinos; (d) detecção de pepinos.

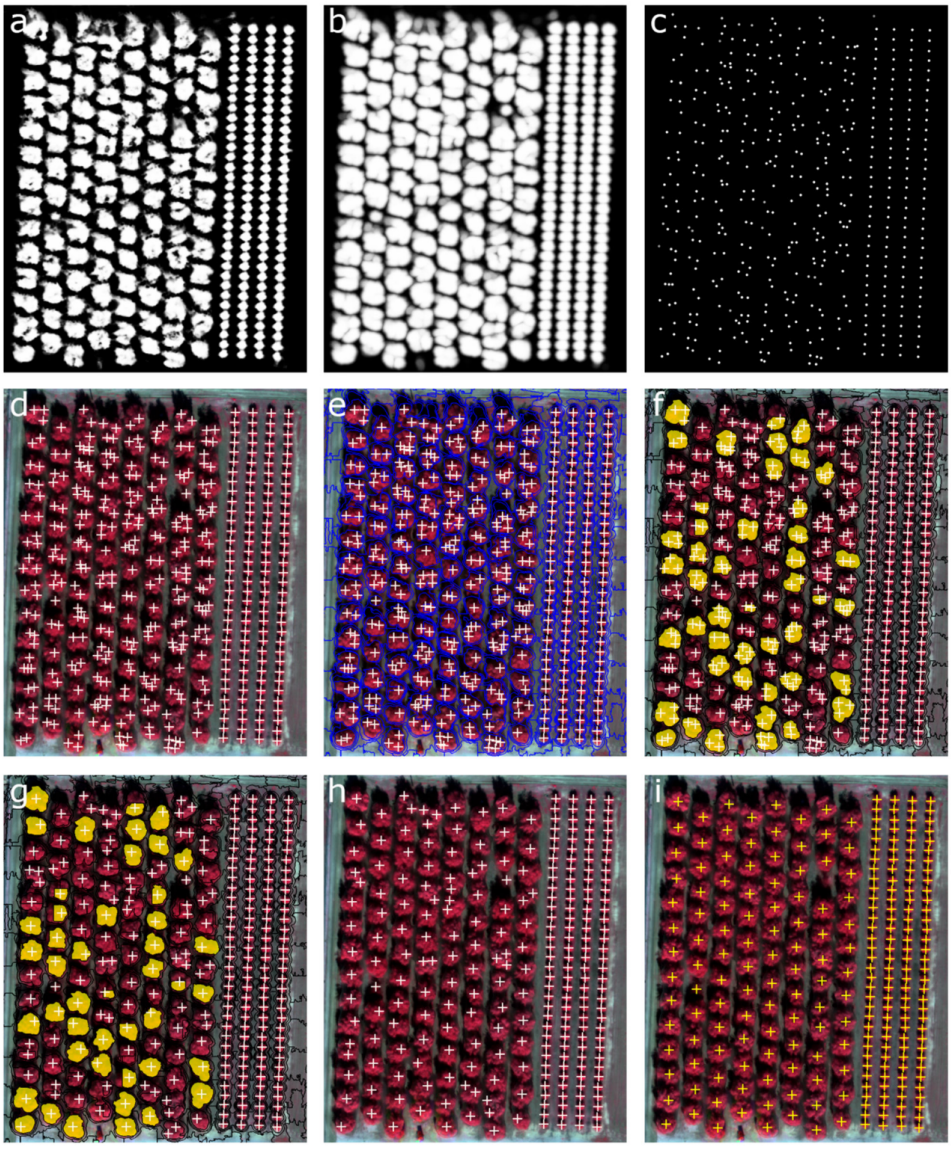
Fonte: Mao *et al.* (2020)

Paralelamente, as redes neurais convolucionais (CNN) têm ganhado destaque no sensoriamento remoto. Estudos comparativos demonstram que tais redes superam métodos tradicionais na detecção de diversas culturas.

A análise das imagens descrita envolve o uso de CNN para detecção e classificação de árvores a partir de imagens multiespectrais obtidas por UAV, utilizando técnicas específicas de treinamento, pré-processamento e pós-processamento. Inicialmente, um modelo simples de CNN foi treinado com amostras de 40×40 píxeis para três classes: árvores, solo exposto e ervas daninhas, utilizando quatro bandas espectrais (verde, vermelho, infravermelho próximo e borda vermelha). O modelo foi ajustado empiricamente, utilizando buffers ao redor dos pontos de amostragem e convoluções com kernels de 11×11 píxeis. Em seguida, aplicou-se um filtro Gaussiano ao mapa

de probabilidade gerado e realizaram-se detecções de máximos locais para localizar as árvores. Para corrigir detecções múltiplas em copas maiores, utilizou-se a segmentação em superpixels com o algoritmo *Simple Linear Iterative Clustering* (SLIC), aplicada ao mapa suavizado e à camada NDVI. Essa abordagem híbrida aumentou significativamente a precisão da classificação, alcançando F-score de 96,24%.

Figura 10 – Processo de análise das imagens



Fonte: Csillik *et al.* (2018)

Na Figura 10, apresentam-se os passos da análise. Um mapa de calor probabilístico da presença de árvores foi gerado a partir de uma CNN, variando de 0 a 1 (Figura 10.a). Esse mapa foi suavizados (Figura 10.b), e as árvores foram identificadas com limiar de 0,5 (Figura 10.c). O subconjunto ilustrado evidencia o problema de

detecção múltipla de copas (Figura 10.d), resolvido com segmentação iterativa via superpixels SLIC maiores que 40×40 píxeis (Figura 10.e). As detecções múltiplas foram identificadas (Figura 10.f) e ajustadas (Figura 10.g). Os resultados refinados (Figura 10.h) demonstraram maior compatibilidade com as amostras de campo (Figura 10.i).

Ainda existe uma lacuna na literatura quanto ao uso de CNN para identificar árvores cítricas de diferentes idades com base em imagens aéreas, sendo este um dos primeiros estudos a abordar tal desafio. Os autores ressaltam a necessidade de novos estudos que consolidem fluxos automatizados padronizados, capazes de apoiar gestores agrícolas no uso eficiente de imagens aéreas para a tomada de decisões.

Outro exemplo de aplicação prática é à automação da colheita de morangos, um dos cenários mais desafiadores para técnicas modernas. Sistemas robóticos, como o Agrobot SW6010 (YU *et al.*, 2019), foram desenvolvidos com o objetivo de automatizar essa tarefa. Entretanto, as condições ambientais — ilustradas na Figura 11 — dificultam a aplicação efetiva de métodos tradicionais de visão computacional.

Figura 11 – Amostragens de morangos em diferentes cenários naturais

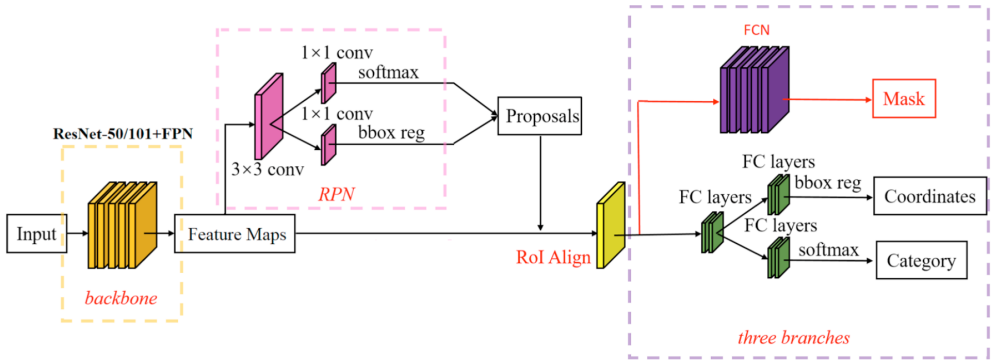
(a) frutos separados; (b) frutos aderentes; (c) frutos sobrepostos; (d) frutos ocluídos; (e) frutos sob iluminação insuficiente.

Fonte: Yu *et al.* (2019)

Para superar tais desafios, o estudo adotou uma abordagem baseada em aprendizado profundo, utilizando o modelo *Mask R-CNN*, que integra detecção de objetos e segmentação de instâncias em uma única arquitetura. As imagens foram capturadas manualmente em plantações de morango na China, resultando em um conjunto com 2.000 imagens, das quais 1.900 foram utilizadas para treinamento.

O *Mask R-CNN* é considerado *state of the art* em detecção de objetos. Yu *et al.* (2019) propõem um fluxo em que o funcionamento se baseia em uma rede totalmente convolucional (*Fully Convolutional Network – FCN*), integrada a camadas totalmente conectadas (*Fully Connected – FC*) (Figura 12).

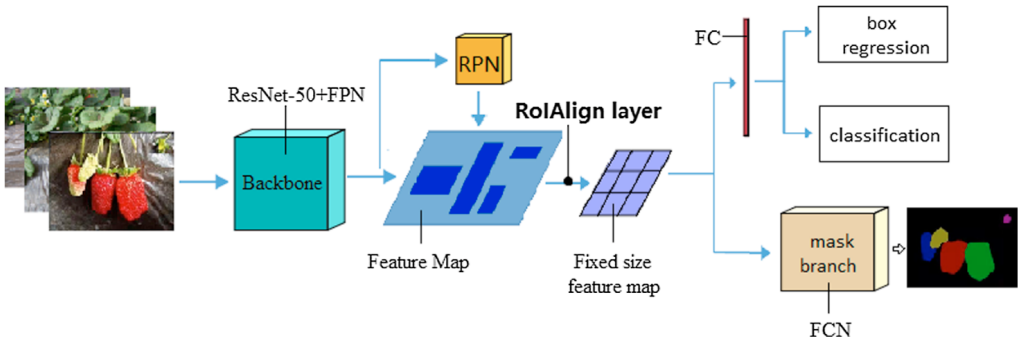
Figura 12 – Arquitetura geral do Mask R-CNN e fluxo de processamento



Fonte: Yu *et al.* (2019)

O fluxo (Figura 13) inicia-se pelo *backbone*, em que a ResNet-50 extrai os mapas de características da imagem de entrada. Em seguida, esses mapas são submetidos à *Region Proposal Network* (RPN), responsável por identificar regiões potencialmente relevantes. A RPN propõe diversas Regiões de Interesse (RoIs), refinadas conforme a probabilidade de conter objetos e sua correspondência com o conteúdo visual. As RoIs passam, então, pelo *RoI Align*, que corrige distorções de quantização, garantindo alinhamento adequado com os dados originais.

Figura 13 – Fluxo de processamento das imagens



Fonte: Yu *et al.* (2019)

O estágio seguinte envolve a *Feature Pyramid Network* (FPN), integrada ao *backbone*, que combina características de diferentes profundidades, unindo mapas de alta resolução e de maior abstração. Cada RoI segue para três saídas da rede: classificação do objeto, regressão da *bounding box* e geração de máscaras binárias. O resultado inclui detecção do objeto, classe associada, localização e máscara de segmentação.

Na Figura 14, apresentam-se exemplos visuais do processo de análise. Já a Figura 15 ilustra os resultados do Mask R-CNN em dois cenários distintos.

A Figura 14, são apresentadas três imagens. Na Figura 14.a, se apresenta a imagem original dos frutos, já a Figura 14.b apresenta as máscaras geradas pela Mask R-CNN, por fim a Figura 14.c) apresenta o resultado da máscara sobre imagem original.

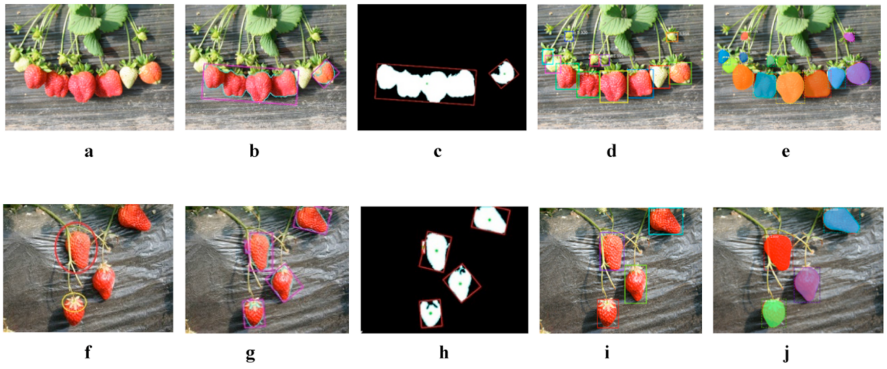
Figura 14 – Exemplos visuais do processo de análise com Mask R-CNN

Fonte: Yu *et al.* (2019)

Na Figura 15, apresentam-se duas situações. Na primeira situação: (Figura 15.a) apresenta imagem original contendo múltiplos frutos aderidos; (Figura 15.b) apresenta o resultado da detecção com métodos tradicionais; (Figura 15.c) apresenta a segmentação desses métodos; (Figura 15.d) apresenta o resultado da detecção com Mask R-CNN; (Figura 15.e) apresenta a segmentação correspondente, que demonstra maior precisão.

Na segunda situação, envolvendo oclusão por folhas e caules: (Figura 15.f) apresenta a imagem original; (Figura 15.g) apresenta a detecção tradicional; (Figura 15.h) apresenta a segmentação correspondente; (Figura 15.i) apresenta a detecção via Mask R-CNN; (Figura 15.j) apresenta a segmentação com desempenho superior na identificação de frutos parcialmente ocluídos.

Figura 15 – Exemplos visuais do resultado da análise das imagens com Mask R-CNN

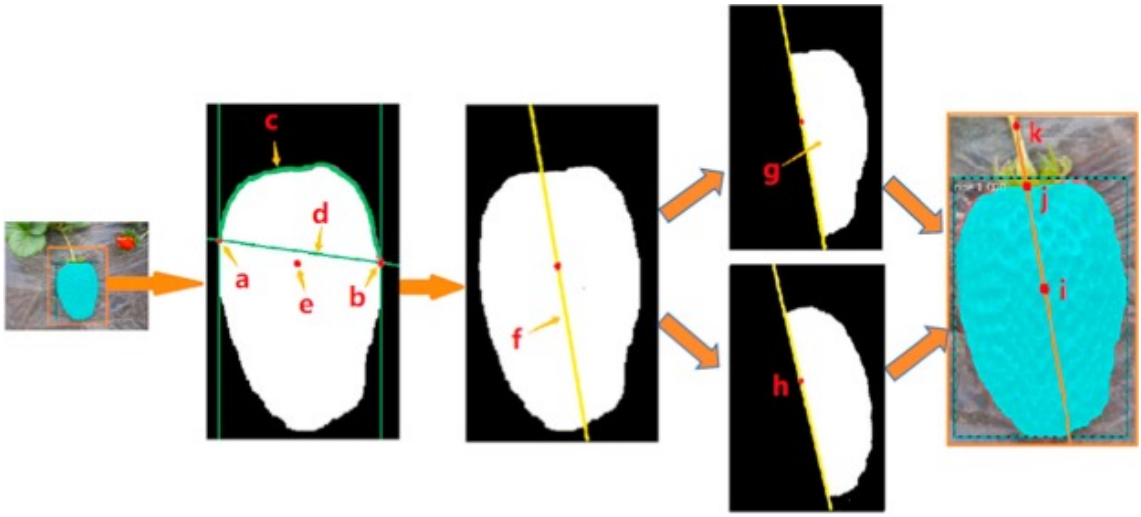


Fonte: Yu *et al.* (2019)

Além da detecção e segmentação das instâncias dos frutos, o estudo também propôs a localização automática dos pontos de colheita. Essa localização foi obtida a partir das máscaras geradas pelo modelo treinado, traçando-se uma linha ao longo do eixo do fruto na direção superior. O ponto de interseção entre essa linha e o contorno do fruto foi marcado como vértice, e o ponto de colheita foi estimado a uma distância de aproximadamente 13 a 20 mm desse vértice, seguindo o mesmo eixo. Essa abordagem considera que morangos crescem predominantemente para baixo, permitindo estimar com precisão o ponto ideal para sua remoção.

Vemos na Figura 16 destacando os parâmetros: (a) mínimo e (b) máximo das coordenadas do contorno; (c) contorno de interesse; (d) linha divisória; (e) baricentro; e (f) eixo do fruto, os quais foram fundamentais para definir o ponto de colheita.

Figura 16 – Localização do ponto de colheita do fruto



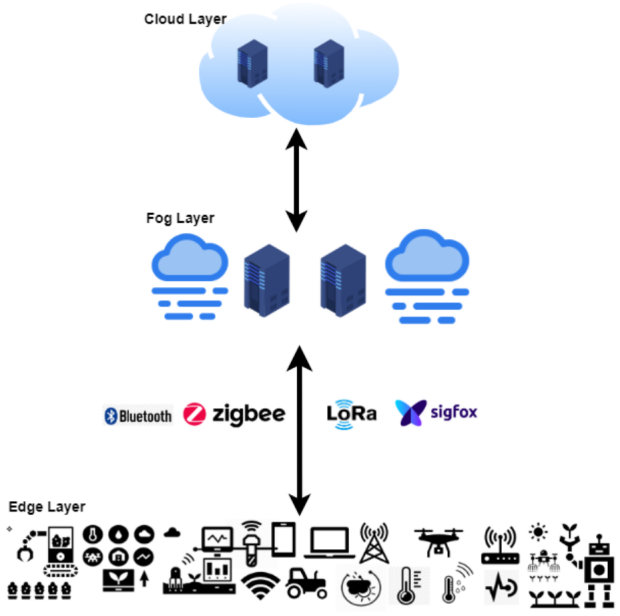
Fonte: Yu *et al.* (2019)

O artigo Kalyani e Collier (2021) afirma que a agricultura tem passado por uma transformação significativa impulsionada por tecnologias emergentes, como computação em nuvem, *Internet of Things* (IoT) e inteligência artificial (IA). Essas inovações possibilitam a coleta de dados em tempo real, a previsão precisa de colheitas e do clima, além de oferecer suporte à tomada de decisão baseada em dados. Com isso, os agricultores podem aumentar a confiabilidade das informações disponíveis, melhorar a eficiência operacional e obter maior lucratividade. A análise de *big data* desempenha papel central nesse cenário, permitindo o uso eficaz de informações provenientes de sensores, máquinas agrícolas e sistemas meteorológicos.

Entre essas tecnologias, a computação em nuvem se destaca como um paradigma amplamente adotado, originado a partir de definições estabelecidas por empresas como Google e Amazon em 2006. Ela oferece serviços como *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) e *Software as a Service* (SaaS), com características como escalabilidade e elasticidade. Utiliza-se para armazenamento remoto, análise de dados e previsões climáticas, embora enfrente desafios como latência, consumo de banda larga e preocupações com segurança e privacidade. Para lidar com essas limitações, surgem as abordagens de *Fog computing* e *Edge computing*, que descentralizam o processamento e aproximam os serviços dos dispositivos que geram dados.

A *Fog computing* atua como uma ponte entre a nuvem e os dispositivos de borda, oferecendo suporte ao processamento em tempo real e reduzindo a latência da comunicação. Já a *Edge computing* realiza o processamento diretamente nos dispositivos finais, aumentando a eficiência e a autonomia dos sistemas agrícolas. Ambas as abordagens compartilham características como suporte à mobilidade e baixa latência, sendo frequentemente confundidas, mas com funções distintas: enquanto a Fog envolve recursos de rede e armazenamento intermediários, a Edge se concentra exclusivamente no processamento local. A integração entre nuvem, neblina e borda, denominada *Cloud–Fog–Edge Computing model* (Figura 17), mostra-se essencial para a construção de sistemas agrícolas inteligentes mais resilientes e seguros.

Figura 17 – Cloud–Fog–Edge Computing model

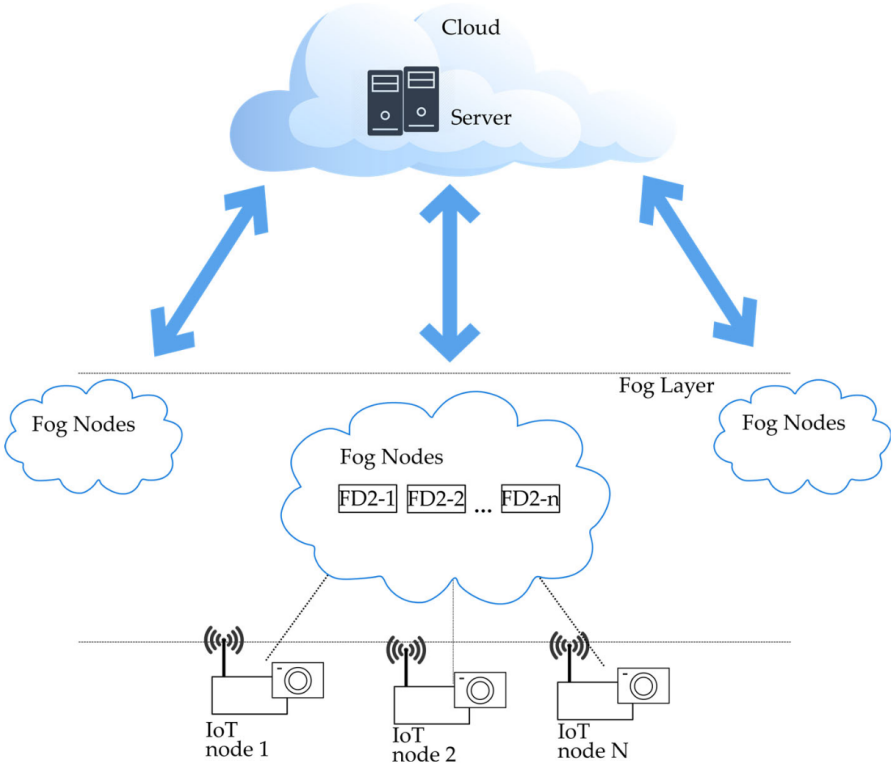


Fonte: Kalyani e Collier (2021)

A pesquisa de Marković *et al.* (2024) propôs um modelo de classificação de imagens otimizado para dispositivos com recursos limitados, com foco em aplicações como detecção de doenças, pragas e plantas daninhas. O uso de CNNs compactas permite que os dados sejam processados localmente ou em camadas intermediárias da rede, como *Fog nodes*, reduzindo atrasos e sobrecarga da nuvem e mantendo os níveis de precisão.

A estrutura do sistema proposto (Figura 18) baseia-se em dispositivos IoT conectados a nós Fog, com modelos de CNN adaptados por técnicas como quantização e compressão. Os dados são processados no local ou em dispositivos intermediários, como *Field Programmable Gate Arrays* (FPGAs). Dois artigos se destacam ao detalhar a estrutura e o fluxo do sistema: ambos abordam o treinamento inicial dos modelos em servidores e sua posterior conversão para formatos compatíveis com execução eficiente em dispositivos de borda, como o PYNQ Z2. A utilização do TensorFlow Lite e do framework Tensil viabilizou tal adaptação para ambientes com restrições de hardware.

Figura 18 – Estrutura do sistema proposto por Marković *et al.* (2024)



Fonte: Marković *et al.* (2024)

A coleta de imagens é padronizada conforme o tipo de cultura e problema agrícola. Por exemplo, imagens de folhas de tomate foram classificadas em 10 categorias e redimensionadas para 128×128 píxeis, enquanto imagens de pragas foram redimensionadas para 96×96 píxeis. Essa padronização é essencial para assegurar compatibilidade com modelos leves de CNN (*Compact CNN* – CCNN) e manter a acurácia durante o treinamento e a inferência. O pré-processamento inclui normalização de cores, extração de contornos e técnicas de *foreground segmentation*, otimizando a extração de características relevantes para as tarefas de classificação.

A avaliação do sistema demonstrou que o processamento na camada Fog reduziu significativamente a latência e o consumo de energia, mantendo desempenho de até 99% de acurácia em aplicações específicas. Redes neurais compactas superaram modelos profundos em eficiência, sendo mais adequadas para ambientes agrícolas remotos. A pesquisa reforça o potencial da computação Edge–Fog para aplicações em agricultura inteligente e indica caminhos futuros com técnicas como *transfer learning*, visando superar limitações relacionadas à escassez de dados agrícolas anotados.

2.6 Considerações Finais da Revisão Sistemática

Esta seção apresenta as considerações obtidas a partir da RSL, reunindo a análise dos trabalhos relacionados, suas limitações, as tecnologias utilizadas, as arquiteturas propostas e o posicionamento deste projeto em relação ao estado da arte.

As técnicas modernas de visão computacional, especialmente aquelas baseadas em CNNs, mostram-se eficazes na detecção e classificação de frutas a partir de imagens, mesmo quando capturadas em resoluções reduzidas, como 320×320 píxeis, conforme demonstrado por arquiteturas como YOLO. Essa capacidade de adaptação a diferentes resoluções é fundamental para aplicações móveis. Estratégias de pré-processamento, normalização e operações morfológicas integram os *pipelines* de processamento utilizados, contribuindo para maior robustez e precisão dos modelos.

Além disso, observa-se um avanço contínuo na otimização dos algoritmos para execução em dispositivos com restrições de hardware, indicando um cenário promissor no qual sistemas de visão computacional poderão operar em tempo real diretamente em smartphones, ampliando a aplicabilidade de soluções agrícolas móveis.

A Tabela 1 apresenta as etapas contempladas pelos trabalhos analisados. Nota-se que apenas três estudos mencionam explicitamente uma arquitetura destinada ao treinamento ou à execução dos modelos. A maior parte concentra-se em etapas como pré-processamento, processamento por CNNs e, em alguns casos, pós-processamento, enquanto poucos abordam todo o fluxo, da coleta das imagens ao pós-CNN.

Tabela 1 – Comparação entre trabalhos relacionados

Trabalhos	Arquitetura	Coleta de imagens	Pré-processamento	Processamento CNN	Pós-processamento CNN	Processamento Pós-CNN
(CSILLIK *et al.*, 2018)	✗	✗	✓	✓	✗	✗
(YU *et al.*, 2019)	✗	✓	✓	✓	✓	✗
(MAO *et al.*, 2020)	✗	✓	✓	✓	✓	✗
(BIRRELL *et al.*, 2020)	✗	✓	✓	✓	✗	✗
(KALYANI; COLLIER, 2021)	✓	✗	✗	✗	✗	✗
(ISSAC *et al.*, 2023)	✓	✓	✓	✓	✗	✗
(MARKOVIć *et al.*, 2024)	✓	✗	✗	✓	✗	✗
Minha Horta Minha Vida	✓	✓	✓	✓	✓	✓

Fonte: Autor (2025)

O projeto *Minha Horta Minha Vida* diferencia-se por propor uma abordagem completa e padronizada, abrangendo desde a coleta sistemática das imagens até o processamento com CNNs e refinamento dos resultados. Essa padronização contribui para a consistência dos dados e a confiabilidade dos modelos treinados, favorecendo aplicações robustas e escaláveis em cenários agrícolas reais.

A aplicação de tecnologias na agricultura de precisão tem-se diversificado, incorporando desde ferramentas clássicas de visão computacional até modelos de aprendizado profundo altamente especializados. Cada estudo analisado evidencia que diferentes culturas agrícolas demandam soluções específicas.

Abordagens tradicionais, como o uso do software Trimble eCognition, mostram-se eficazes no mapeamento de árvores cítricas. Já soluções baseadas em redes neurais, como ResNet-50 e YOLO, destacam-se pela precisão em tarefas complexas, como colheita automatizada e detecção de estruturas delicadas, como cápsulas de algodão. Estratégias híbridas, como a combinação de LeNet-5 e técnicas de extração de cor, ilustram soluções adaptadas para contextos desafiadores, como o cultivo de pepinos. A Tabela 2 sintetiza os estudos, destacando culturas, tecnologias e objetivos, permitindo uma análise comparativa das soluções utilizadas na agricultura inteligente.

Tabela 2 – Comparação entre as tecnologias aplicadas à agricultura de precisão

Estudo	Cultura	Tecnologias	Objetivo
Csillik *et al.* (2018)	Cítricos	Trimble eCognition	Identificação e mapeamento de árvores
Yu *et al.* (2019)	Morango	ResNet-50	Colheita automatizada precisa
Birrell *et al.* (2020)	Alface	OpenCV e YOLOv3	Colheita automatizada
Mao *et al.* (2020)	Pepino	LetNet5	Classificação e colheita automática
Issac *et al.* (2023)	Algodão	YOLOv5	Detecção de cápsulas e flores

Fonte: Autor (2025)

A análise de imagens na agricultura de precisão evoluiu significativamente, passando de métodos baseados em características manuais para técnicas avançadas de aprendizado profundo. No centro dessa evolução estão as CNN, que transformaram a forma de detectar e classificar objetos em ambientes agrícolas complexos.

Com o avanço das tecnologias, arquiteturas como YOLOv3 e YOLOv5 elevaram a velocidade e a precisão da inferência em tempo real, enquanto variantes como a *Multi-Path Convolutional Neural Network* (MPCNN) oferecem melhorias em cenários de alto ruído visual, como a colheita de frutas verdes. Mais recentemente, modelos como o *Mask Region-Based Convolutional Neural Network* (Mask R-CNN) têm-se destacado pela capacidade de segmentação de instâncias, identificando frutas sobrepostas com precisão superior. A Tabela 3 sintetiza as principais técnicas identificadas, evidenciando a trajetória evolutiva das abordagens aplicadas à agricultura inteligente.

Tabela 3 – Comparação entre técnicas de análise de imagens

Trabalhos	CNN	MPCNN	Mask R-CNN
Csillik *et al.* (2018)	✓		
Yu *et al.* (2019)			✓
Birrell *et al.* (2020)	✓		
Mao *et al.* (2020)		✓	
Issac *et al.* (2023)	✓		

Fonte: Autor (2025)

A comparação entre os trabalhos analisados evidencia a diversidade de técnicas complementares empregadas para aprimorar o desempenho dos modelos de visão computacional aplicados à agricultura de precisão. Entre essas técnicas destacam-se:

- Simple Linear Iterative Clustering (SLIC);
- Region Proposal Network (RPN);

- Feature Pyramid Network (FPN);
- Support Vector Machine (SVM);
- Regiões Extremas Estáveis Maximamente (MSER).

Enquanto alguns estudos utilizam técnicas clássicas, outros combinam métodos para otimização dos resultados, a exemplo da integração entre SVM e MSER em uma MPCNN. Também se destacam combinações como RPN, FPN e RoIAlign, amplamente utilizadas em redes como Faster R-CNN e Mask R-CNN. A Tabela 4 apresenta essas técnicas e como se integram ao processamento de imagens em diferentes aplicações.

Tabela 4 – Comparação entre as técnicas complementares à CNN

Trabalhos	**SLIC**	**FPN**	**RPN**	**MSER**	**SVM**
Csillik *et al.* (2018)	✓				
Yu *et al.* (2019)		✓	✓		
Mao *et al.* (2020)				✓	✓

Fonte: Autor (2025)

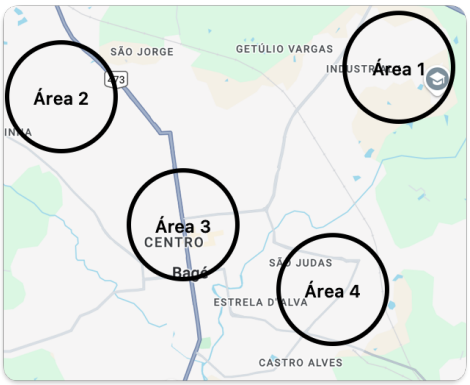
Apenas três dos artigos analisados abordam explicitamente a arquitetura utilizada. Em Issac *et al.* (2023), adota-se uma solução totalmente em nuvem, estruturada em duas camadas — *Speed layer* e *Batch layer*. A primeira recebe as imagens enviadas pelos usuários, enquanto a segunda realiza o treinamento dos modelos, alimentada por dados selecionados da etapa inicial. Os estudos de Kalyani e Collier (2021) e Marković *et al.* (2024) propõem uma arquitetura *Fog*, transferindo a *Speed layer* para dispositivos Fog e mantendo a *Batch layer* na nuvem. Entretanto, Kalyani e Collier (2021) não detalha a implementação dessas camadas, ao passo que Marković *et al.* (2024) apresenta tanto a implementação quanto uma otimização baseada em FPGAs.

3 MINHA HORTA, MINHA VIDA

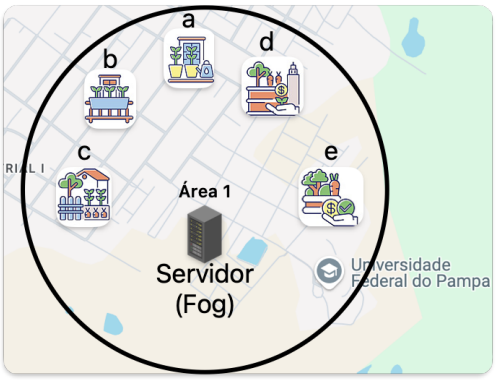
Este trabalho apresenta uma proposta de solução para a detecção e contagem de frutos em imagens agrícolas, com base na arquitetura Cloud–Fog–Edge e utilizando um pipeline modular e escalável inspirado no modelo Azure Lambda. A estrutura organiza os dados por áreas geográficas (Figura 19a), como bairros, compostos por pequenos produtores (Figura 19b), que capturam e enviam imagens pontuais por meio de um aplicativo móvel. Os produtores podem optar por ter o plantio dentro de casa (Cenário a), ter o plantio fora de casa em vasos (Cenário b) ou ter o plantio fora de casa direto no terreno (Cenário c), e os motivos para a plantação podem ser para fins de renda extra (Cenário d) ou para renda e consumo próprio (Cenário e).

Figura 19 – Estrutura do projeto

(a) Estrutura das áreas



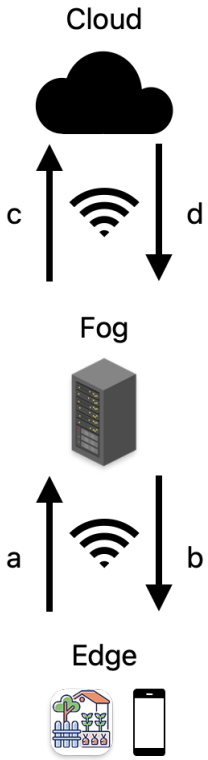
(b) Estrutura dentro das áreas



Fonte: Autor (2025)

Após a captura das imagens, estas são enviadas para a camada Fog (fluxo na Figura 20.a). Essas imagens são processadas localmente, onde se realizam as inferências com um modelo Mask R-CNN previamente treinado, se retornam os resultados ao usuário (fluxo na Figura 20.b) e armazenam as saídas para envio periódico à nuvem (fluxo na Figura 20.c). A nuvem, por sua vez, recebe essas imagens para aprimorar continuamente os modelos, incorporando novos dados ao processo de revalidação e retreinamento, e posteriormente envia versões atualizadas dos modelos de volta aos dispositivos de névoa (fluxo na Figura 20.d). A nuvem é responsável por treinar e versionar os modelos, além de armazenar as imagens recebidas da Fog.

Figura 20 – Fluxo da Arquitetura



Fonte: Autor (2025)

A estrutura do capítulo segue a ordem lógica do fluxo de dados na aplicação: inicia com uma visão geral da arquitetura proposta, seguida pelas etapas de coleta e armazenamento das imagens. Em seguida, são abordados o treinamento do modelo, o processamento dos dados e a inferência do modelo, e o pós-processamento dos resultados. Cada uma dessas etapas será explorada em mais detalhes nas seções a seguir.

3.1 Proposição do Artefato

O projeto visava desenvolver uma solução capaz de apoiar o produtor urbano em todas as etapas da produção agrícola. A proposta incluía coleta de dados, análise por visão computacional e recomendações de manejo. No entanto, ao longo do desenvolvimento, constatou-se que a abrangência planejada excedia o tempo disponível. Assim, o escopo foi reformulado para concentrar-se na etapa de colheita, culminando no desenvolvimento de um aplicativo móvel capaz de detectar frutos, identificar seu estágio de maturação e determinar sua localização nas imagens capturadas.

Propõe-se, portanto, um artefato voltado à detecção e análise de frutas em imagens capturadas por dispositivos móveis, integrado a uma arquitetura distribuída baseada nos paradigmas Cloud–Fog–Edge. As decisões de projeto foram guiadas por critérios de viabilidade técnica, eficiência computacional, escalabilidade e acessibilidade em contextos agrícolas reais. A arquitetura proposta organiza-se em três camadas:

- **Camada de Borda (Edge):** formada por dispositivos responsáveis apenas pela captura das imagens, sem execução de inferência local, o que permite o uso da solução em equipamentos de baixo custo.
- **Camada de Processamento (Fog):** composta por dispositivos embarcados de capacidade moderada, responsáveis por executar modelos em sua etapa de *ingest*. Essa camada fornece respostas rápidas e maior autonomia operacional.
    - Inferência com modelos treinados, adotando uma versão da Mask R-CNN para detecção e contagem de frutos;
    - persistência temporária e sincronização posterior, garantindo operação mesmo sob instabilidade de conectividade.
- **Camada de Persistência e Treinamento (Cloud):** composta por serviços capazes de armazenar dados, treinar modelos e monitorar seu desempenho.
    - Treinamento e retreinamento contínuo dos modelos com base em imagens recebidas das camadas inferiores;
    - gerenciamento de versões dos modelos, posteriormente disponibilizadas para a camada Fog;
    - coordenação dos fluxos de ingestão, curadoria e organização das imagens;
    - monitoramento e orquestração dos dispositivos Fog, incluindo distribuição de modelos e diagnóstico de falhas.

Opta-se pela plataforma AWS, especialmente pelo Amazon SageMaker, em contraste com a arquitetura "Azure Lambda" mencionada na seção 2.5. Essa opção decorre de requisitos específicos do paradigma Fog–Cloud. O SageMaker oferece integração com dispositivos de borda por meio do AWS Greengrass, monitoramento nativo com o SageMaker Model Monitor, automação com o SageMaker Pipelines e otimização de modelos para dispositivos embarcados com o SageMaker Neo. O Edge Manager facilita a sincronização entre os modelos implantados localmente e a nuvem, aumentando a autonomia operacional em ambientes com conectividade limitada.

A camada de nuvem demanda um ambiente capaz de hospedar modelos em *endpoints* escaláveis, permitindo testes controlados, avaliações comparativas e integração contínua com a camada Fog. A arquitetura também requer recursos robustos de observabilidade, como rastreamento de experimentos e registro de métricas, fundamentais para garantir confiabilidade e evolução contínua dos modelos.

Para o processamento de imagens, propõe-se o uso da Mask R-CNN, arquitetura amplamente reconhecida por sua capacidade de segmentação de instâncias. Sua robustez frente a desafios comuns em ambientes agrícolas — como oclusões, sobreposições, ruído e variações de iluminação — justifica sua adoção. Ao fornecer máscaras de segmentação mais precisas que simples caixas delimitadoras, a Mask R-CNN possibilita contagens mais acuradas, aspecto crucial para decisões agronômicas baseadas em produtividade.

3.2 Modelagem da ferramenta

A organização da seção corresponde à sequência do desenvolvimento do artefato: começa com uma visão geral da arquitetura sugerida, seguida pelas fases de modelagem para a coleta e armazenamento das imagens, treinamento do modelo, tratamento dos dados e a inferência do modelo, bem como o pós-processamento dos resultados e a apresentação dos mesmos. Cada uma dessas fases será examinada com mais profundidade nas seções subsequentes.

São apresentadas duas arquiteturas distintas desenvolvidas para suportar um pipeline de processamento e inferência de imagens. A primeira delas, denominada arquitetura Cloud, mantém todo o processamento concentrado na *Cloud*, seguindo princípios semelhantes à arquitetura Azure Lambda, mas opta por um ambiente de desenvolvimento e execução dos modelos diferente. Essa escolha decorre de características específicas do modelo Mask R-CNN. Já a segunda, denominada Arquitetura Fog, representa uma evolução da proposta inicial ao incorporar o paradigma de *Fog Computing*, distribuindo parte do processamento para a borda.

A arquitetura Cloud, ilustrada na Figura 21, faz algumas modificações nas etapas dá *pipeline* apresentada na arquitetura Azure Lambda (Figura 6). A etapa "Ingest" permanece dividida em dois fluxos: "Stream" e "Batch". Onde o fluxo de "Stream" passa de um modelo de streaming contínuo para um modelo de streaming orientado por eventos, assíncrono e sob demanda, baseado em imagens coletadas pelos próprios produtores. Ao invés de depender de um fluxo constante e ininterrupto de

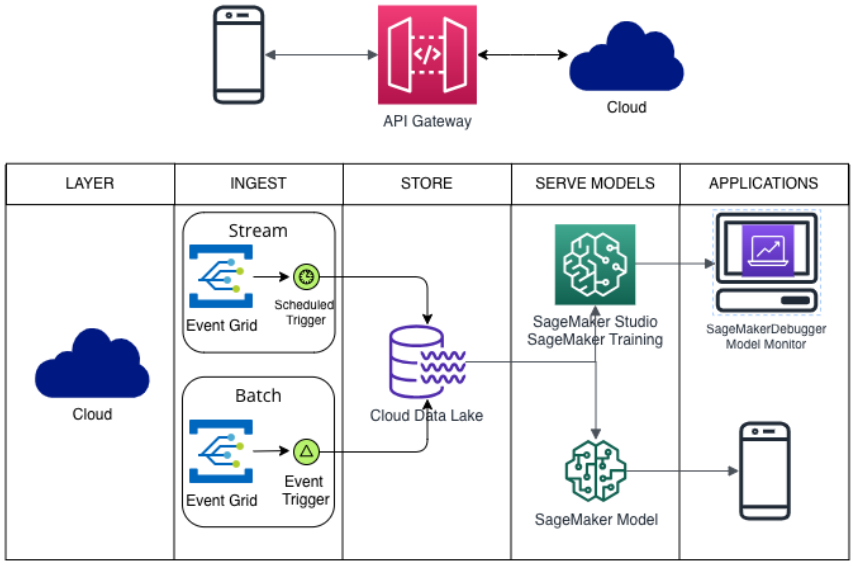
imagens enviadas automaticamente, o sistema passa a trabalhar com eventos esporádicos que refletem momentos mais relevantes na coleta de dados. Esses gatilhos se integram ao AWS EventBridge, permitindo a ativação automática dá *pipeline*.

O fluxo *batch* permanece responsável pelo processamento periódico de grandes volumes de dados, armazenados em um *data lake*. Ele funciona com acionamentos programados, por meio de expressões *Command Run On* (CRON) configuradas no AWS EventBridge, permitindo o reprocessamento de dados históricos e a preparação de grandes conjuntos de dados para o treinamento de modelos de aprendizado de máquina.

Após a ingestão, os dados passam pela etapa de "Store", onde são organizados de forma persistente em um *data lake*, as imagens do "Speed Layer" são armazenadas e periodicamente se escolhem algumas delas para serem passadas para a "Batch Layer" para serem usadas em aprimoramento do modelo.

A etapa de "Pre-process" é eliminada, já que não usamos compressão da imagem. Finalmente, na etapa de serve-model, temos um ambiente para treinamento e análise de modelos, como também um ambiente para hospedar os modelos previamente treinados para gerar inferências sobre as imagens recebidas via "Speed Layer".

Figura 21 – Arquitetura Cloud

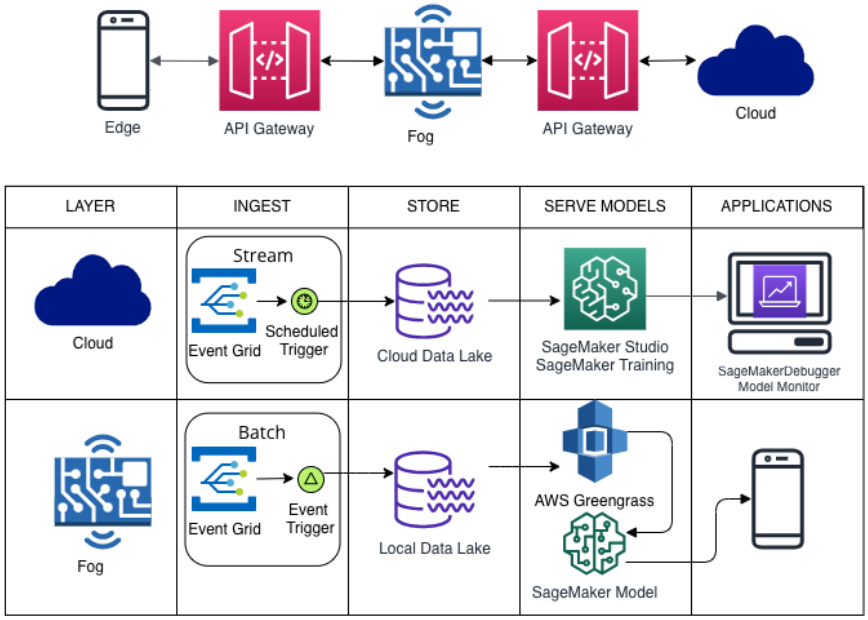


Fonte: Adaptado de Issac *et al.* (2023)

Se incorporar o paradigma de Fog Computing à arquitetura Cloud (Figura 21), resultando na Arquitetura Fog (Figura 22), formando um ecossistema hierárquico e distribuído com três camadas principais: Edge, Fog e Cloud. Onde as etapas dá

pipeline mantêm a separação lógica entre os fluxos “Stream” e “Batch”, mas alteram a distribuição do processamento ao adicionar a camada de *Fog (Fog Layer)*, executada em um microcontrolador. Nesta configuração atualizada, o fluxo da *“Speed Layer”* deixa de ser dependente de uma infraestrutura na nuvem para inferência e começa a funcionar de maneira independente no próprio dispositivo local, empregando um modelo previamente treinado. A cada inferência, os resultados são guardados localmente com a imagem original, sendo posteriormente enviados para a nuvem, onde podem ser incorporados à camada de “Batch”. Esta última continua na nuvem, encarregada de consolidar grandes quantidades de dados e executar reprocessamentos regulares. Assim, os dados oriundos da *Fog Layer* alimentam gradualmente o repositório de dados históricos, possibilitando sua curadoria manual antes de integrarem futuros ciclos de re-treinamento de modelos.

Figura 22 – Arquitetura implementando o paradigma Fog



Fonte: Adaptado de Issac *et al.* (2023)

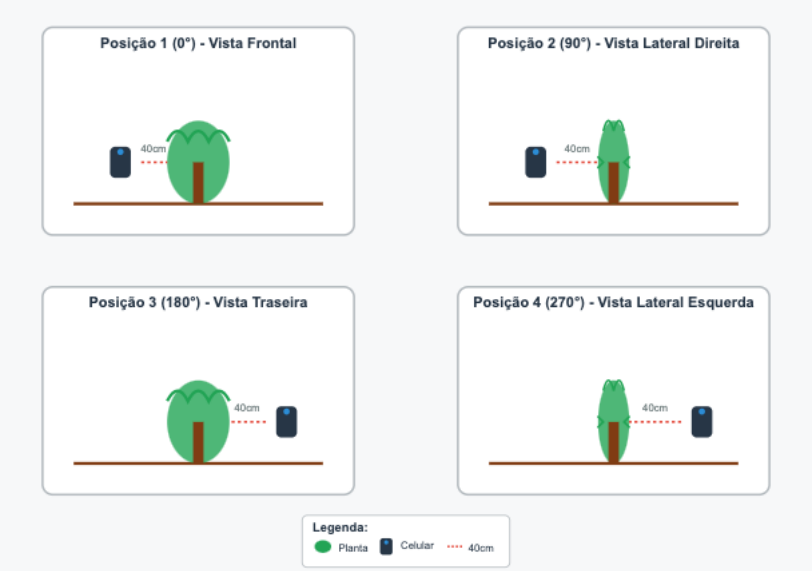
A padronização do processo de captura de imagens visa garantir uma base de dados variada e representativa. As figuras 23b e 23a ilustram o processo da coleta das imagens, onde, para cada planta observada, serão captadas quatro imagens com ângulos de 90° entre si, posicionadas a uma distância em que possa se ver toda a planta. No caso do estudo de caso do tomate-cereja, adotou-se 40 cm.

A coleta foi repetida em diferentes horários do dia, sob diversas condições de iluminação, a fim de contemplar variações ambientais que influenciam diretamente o desempenho dos modelos de visão computacional. Além disso, para aumentar a

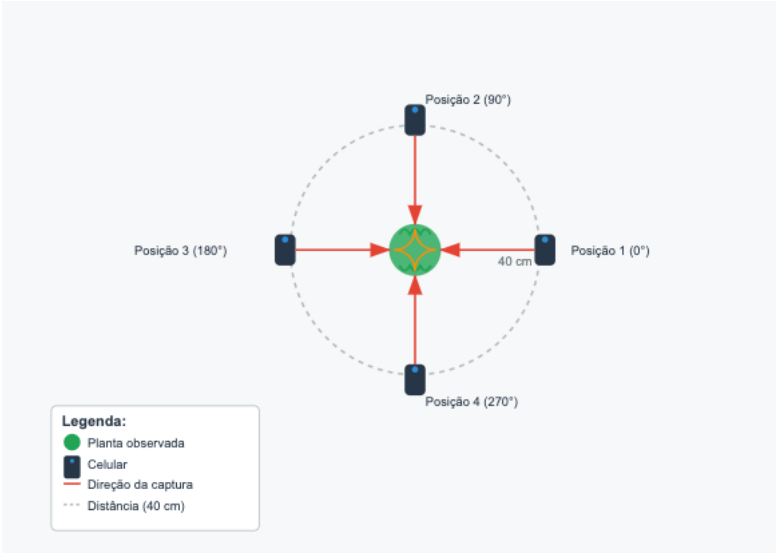
diversidade e a quantidade de amostras, propõe-se a captura de imagens adicionais com desvios angulares de 30° e 60° em relação à posição inicial.

Figura 23 – Fluxo da coleta de imagens

(a)



(b)



Fonte: Autor (2025)

Adere-se à proposta de Yu *et al.* (2019), onde o processo de captura inclui propositalmente diferentes situações encontradas em cenários reais de cultivo, como frutos claramente individualizados, frutos aderentes entre si, casos de sobreposição parcial dificultando a segmentação, obstruções provocadas por folhas ou caules, e condições de iluminação insuficiente (Figura 11). Essa abordagem busca garantir que a base de

dados represente adequadamente a complexidade do ambiente agrícola, promovendo o desenvolvimento de modelos mais robustos e generalizáveis.

As imagens são enviadas diretamente por dispositivos móveis para a camada *fog*, via *endpoint*. Posteriormente, as imagens são enviadas para a Cloud para seu armazenamento "permanente". Na *Cloud*, se insere uma camada a mais de organização, cada área envia suas imagens para uma pasta separada.

O processo de treinamento do modelo inicia-se com a preparação e organização dos dados. As imagens são previamente anotadas para identificar as regiões de interesse, incluindo máscaras de segmentação, caixas delimitadoras e rótulos de classe. Em seguida, esses dados são organizados no *Batch Layer*, divididos em conjuntos de treino e validação.

Com os dados organizados, cria-se o ambiente de desenvolvimento no SageMaker, utilizando uma instância de notebook. Esse ambiente serve para configurar o pipeline de treinamento, definir os parâmetros e preparar o modelo.

A seguir, o código de treinamento é encapsulado em um *script* e configurado como um *job* de treinamento no SageMaker. Esse *script* é responsável por carregar os dados, aplicar as transformações necessárias, inicializar o modelo com os parâmetros ajustados e executar o processo de treinamento propriamente dito, utilizando instâncias com GPU, para acelerar a execução. Durante o treinamento, são monitoradas métricas de desempenho como perda (*loss*), precisão de detecção e segmentação, a fim de acompanhar a evolução do modelo.

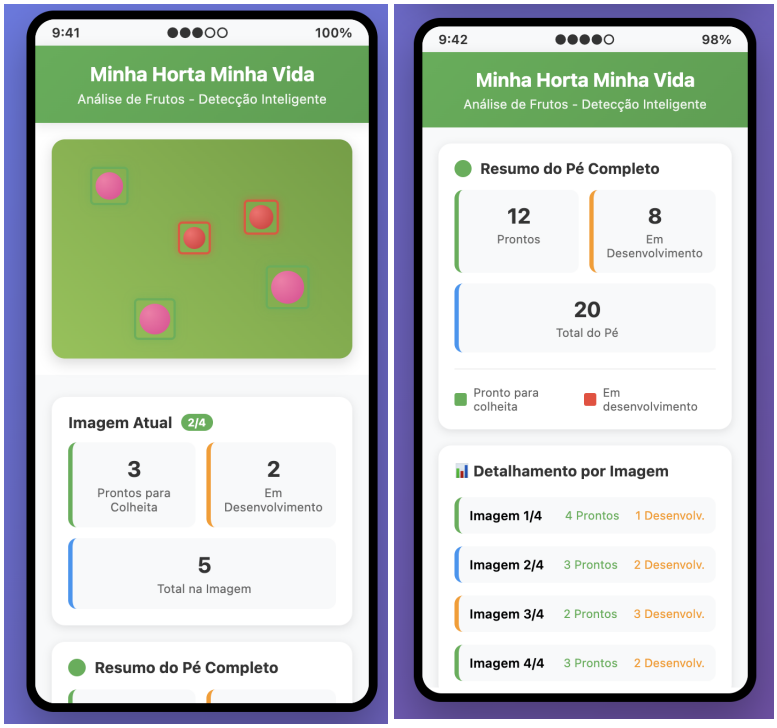
Ao final do processo, o modelo treinado é salvo em um *bucket* no S3. Esse artefato resultante pode então ser implantado em um endpoint de inferência e transferido futuramente para a camada Fog.

Adota-se o fluxo proposto por Yu *et al.* (2019), ilustrado nas Figuras 12 e 13. Este começa com a extração de mapas de características pela ResNet-50, que atua como backbone da arquitetura. Esses mapas são enviados à Region Proposal Network (RPN), responsável por gerar as Regiões de Interesse (RoIs), que são então refinadas e alinhadas com a imagem original por meio do RoI Align. A Feature Pyramid Network (FPN), integrada ao *backbone*, enriquece as representações combinando informações de diferentes escalas. Por fim, cada RoI é processada por três ramificações: uma para classificação do objeto, outra para regressão da bounding box e uma terceira para geração da máscara de segmentação.

Os resultados da inferência são utilizados para gerar visualizações das imagens analisadas (Figura 24). Nessas visualizações, são desenhadas caixas delimitadoras ao redor dos frutos: caixas verdes indicam frutos prontos para a colheita, enquanto caixas vermelhas indicam frutos ainda em desenvolvimento. Abaixo de cada imagem, é exibida uma contagem com o número total de frutos identificados na planta e na imagem específica, que faz parte de um conjunto de quatro imagens. Essa contagem é organizada em três categorias: frutos maduro, frutos em desenvolvimento e total de frutos.

Figura 24 – Proposta inicial de tela de resultado da análise

(a) (b)



Fonte: Autor (2025)

3.3 Implementação do artefato versão Cloud

Nesta seção, aborda-se a implementação da versão Cloud do artefato proposto, incluindo a explicação da construção do projeto, desde linhas de comando até ajustes necessários para que ele funcione com as tecnologias propostas. Contemplando a parte de autenticação e configuração da comunicação com o banco de dados. Em seguida, se explica a proposta de estrutura de banco de dados, assim podendo apresentar os fluxos disponíveis para os usuários e explicitar como os mesmos estão ligados ao banco de dados.

Uma vez explicado o funcionamento da estrutura de dados, passa-se ao desenvolvimento do modelo Mask R-CNN no SageMaker e às configurações no AWS EventBridge, para o disparo de acionamento do modelo ao se inserir imagens novas em determinada pasta do nosso *bucket* e o armazenamento dos resultados do acionamento no *bucket*. Com isso, o usuário pode acessar o aplicativo para visualizar os resultados da análise.

Inicia-se o desenvolvimento do artefato ao criar um projeto Flutter, o diretório inicial contém uma pasta para uso via navegador web ou para cada sistema operacional em que funciona - Android, IOS, Linux, MacOS e Windows -, a pasta "test", a pasta "node_modules" e a pasta "lib", onde vamos desenvolver o projeto, além de alguns arquivos que não serão relevantes no momento, com exceção do "pubspec.yaml". Em seguida, instala-se o Amplify CLI via npm (Node Package Manager), via linha de comando. Após a instalação, deve-se configurar o Amplify com o comando "amplify configure", que abrirá uma página web para fazer login na conta AWS e criar um usuário IAM (Identity and Access Management) com as permissões necessárias para o Amplify. Após configurar o usuário, deve-se voltar ao terminal e inserir as credenciais criadas.

Para o *backend*, opta-se pelo uso do Amplify, uma vez que se trata da abordagem considerada "padrão" pela AWS para acesso a seus recursos a partir de aplicativos. Inicialmente, cria-se uma *sandbox*. Esse tipo de ambiente é exclusivamente experimental, trata-se de um ambiente isolado, destinado a testes e ao desenvolvimento local.

Após a conclusão da fase de testes e a validação da primeira versão funcional — já capaz de executar o fluxo inicial de análise da planta — procede-se à criação de uma aplicação completa no Amplify Console. A partir desse momento, o desenvolvimento passa a ocorrer em uma *branch* específico do repositório Git, dedicado ao ambiente de desenvolvimento, permitindo evoluir o código e a infraestrutura de forma controlada. Uma vez alcançada a versão final e estável, realiza-se o *merge* para a *branch* main, que está vinculada ao ambiente de produção.

A configuração de um projeto no Amplify exige um repositório Git, pois o serviço utiliza o histórico de versões para orquestrar o ciclo completo de construção e atualização dos recursos. Sempre que um novo *commit* é inserido na *branch* associado ao ambiente, o Amplify executa automaticamente o processo de reconstrução e sincronização. A utilização de *branchs* adicionais possibilita a criação de ambientes independentes, adequados para testes com integração real entre código e infraestrutura.

Adicionando-se as linhas apresentadas a seguir ao arquivo “pubspec.yaml”. Com essas modificações, o projeto passa a gerenciar a autenticação de usuários na nuvem por meio de e-mail e senha. Utiliza-se o conceito de *user pool*, que consiste em um serviço da AWS destinado ao armazenamento e gerenciamento de usuários e suas credenciais.

- “amplify_flutter” para conectar seu aplicativo com os recursos do Amplify.
- “amplify_auth_cognito” para conectar aos recursos do Amplify Cognito.
- “amplify_authenticator” para usar os componentes da interface do Amplify.

Em seguida, procede-se para a configuração da parte de dados para a aplicação, partindo de um exemplo de esquema de dados até a estrutura e forma de consumo dos dados pela aplicação. A pasta “Amplify”, criada no passo anterior, contém uma pasta “data”, a qual contém o arquivo “resource.ts”. Este arquivo possui um *backend* pré-configurado. O exemplo padrão criará um modelo *Todo*, uma coleção de itens, com o atributo *content*. Neste arquivo, configuram-se todos os modelos de objetos do banco de dados.

Figura 25 – *Backend* do artefato

```
    HortaPlanta: a
    .model({…
    })
    .authorization((allow) => [allow.authenticated()]),

    Analise : a
    .model({…
    })
    .authorization((allow) => [allow.authenticated()]),
});

export type Schema = ClientSchema<typeof schema>;

export const data = defineData({
  schema,
  authorizationModes: {
    defaultAuthorizationMode: 'identityPool',
  },
});
```

Fonte: Autor (2025)

Optou-se por um banco não relacional por sua flexibilidade na modelagem e por permitir ajustes futuros, o que é importante para este projeto, que visa abranger diversos modelos e funcionalidades ao longo do tempo, e não apenas o módulo

atualmente implementado. Essa escolha viabiliza a evolução da arquitetura sem a necessidade de reestruturações complexas no armazenamento de dados, contribuindo para a escalabilidade e manutenção da solução.

O *backend* do artefato opta pelo método de autenticação identityPool, em que o usuário terá que cadastrar um e-mail e senha para acessar a aplicação. Os objetos estão com a "authorization" configurada para "allow.authenticated", o que restringe o acesso ao banco de dados apenas a pessoas autenticadas (Figura 25). Ao finalizar a implementação dos modelos a serem usados pela aplicação, devemos adicionar a dependência "amplify_api" no arquivo "pubspec.yaml", que gerenciará toda a comunicação com o banco de dados.

Finalmente, executa-se o código de geração dos modelos, o que gerará o código do cliente GraphQL para a aplicação Flutter. O Amplify Data usa GraphQL internamente para fazer solicitações de consulta, mutação e assinatura. O código do cliente GraphQL gerado nos ajuda a criar solicitações de API totalmente tipadas sem precisar criar solicitações GraphQL manualmente e mapeá-las manualmente para o código Dart. O código é armazenado na pasta indicada no comando, neste caso, "lib/models".

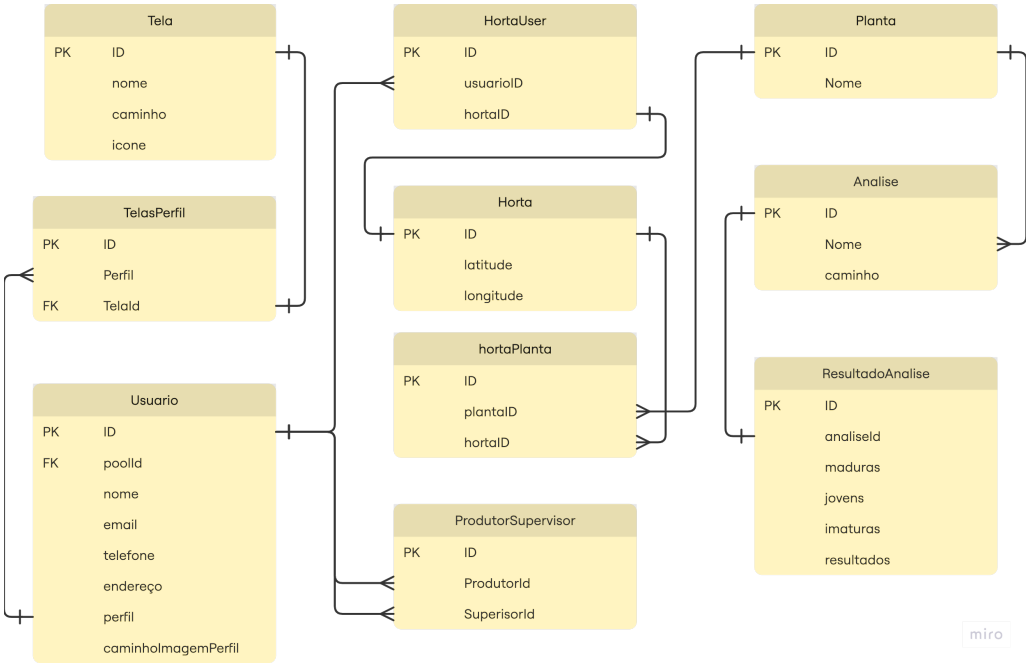
3.4 Proposta de estrutura de dados

A estrutura do banco de dados apresentado é composta por dez entidades inter-relacionadas, com controle de acesso e organização de dados geoespaciais e botânicos. As principais entidades são: Usuário, Horta, Planta, Tela e Análise, cada uma representando elementos essenciais da aplicação. A entidade Usuario contém informações pessoais e de autenticação dos usuários, incluindo seu perfil de acesso e um poolId vinculado ao Amazon Cognito. A entidade Horta representa uma horta com localização geográfica (latitude e longitude), enquanto a entidade Planta armazena dados sobre as plantas disponíveis no sistema.

As relações entre essas entidades são mediadas por coleções de junção, configurando relacionamentos do tipo muitos-para-muitos. A entidade Tela representa as telas da interface do sistema, e sua associação com perfis de usuários é feita por meio da coleção TelasPerfil, que permite controle de acesso baseado em perfis (modelo RBAC – *Role-Based Access Control*). A coleção HortaUsuario estabelece um vínculo entre usuários e hortas, indicando quais usuários participam ou têm acesso a quais hortas. Da mesma forma, a coleção HortaPlanta relaciona hortas com plantas específicas cultivadas

nelas. Seguindo a mesma lógica temos produtorSupervisor, que relaciona produtores a seus supervisores. A coleção Analise armazena registros de análises realizadas, com caminhos para a imagem analisada e caminho para um JSON com o resultado da análise, os mesmos se encontram na *Cloud*, com armazenamento em nuvem como o Amazon S3. Por último temos resultadoAnalise que armazena o resultado do tratamento da análise. Esse modelo contempla separação clara de responsabilidades entre entidades, normalização adequada para evitar redundância, e flexibilidade para escalabilidade e controle de acesso granular.

Figura 26 – Estrutura do Banco de Dados



Fonte: Autor (2025)

As coleções Usuário, TelasPerfil, HortaUsuario, HortaPlanta, Analise, ProdutorSupervisor e Relatorio utilizam índices secundários, que permitem a execução de consultas otimizadas em campos específicos além do identificador principal. Esses índices viabilizam a recuperação eficiente de dados relacionados, como buscar todas as hortas associadas a um determinado usuário, todas as plantas de uma horta ou todas as análises vinculadas a uma planta específica, sem necessidade de varrer toda a coleção. Além de melhorar o desempenho e reduzir custos de leitura no DynamoDB, os índices secundários facilitam a implementação de relacionamentos do tipo muitos-para-muitos e possibilitam o uso de *subscriptions* filtradas no AppSync, garantindo atualizações em tempo real apenas para os dados relevantes a cada usuário.

O modelo Usuario foi definido utilizando a sintaxe declarativa do AWS Amplify, com suporte a GraphQL. Este modelo representa os dados essenciais dos usuários cadastrados no sistema e é protegido por regras de autenticação, permitindo acesso apenas a usuários autenticados via Amazon Cognito. Além de um índice secundário para consultas otimizadas por poolId, que é o identificador único do usuário no Cognito.

Figura 27 – Modelo Usuario (*Schema*)

```
Usuario: a .model(
  {
    poolId : a.string().required(),
    nome : a.string().required(),
    email : a.string().required(),
    telefone : a.string().required(),
    endereco : a.string().required(),
    perfil : a.string().required(),
    caminhoImagemPerfil : a.string().required(),
  }).secondaryIndexes((index) => [ index('poolId'),]),
```

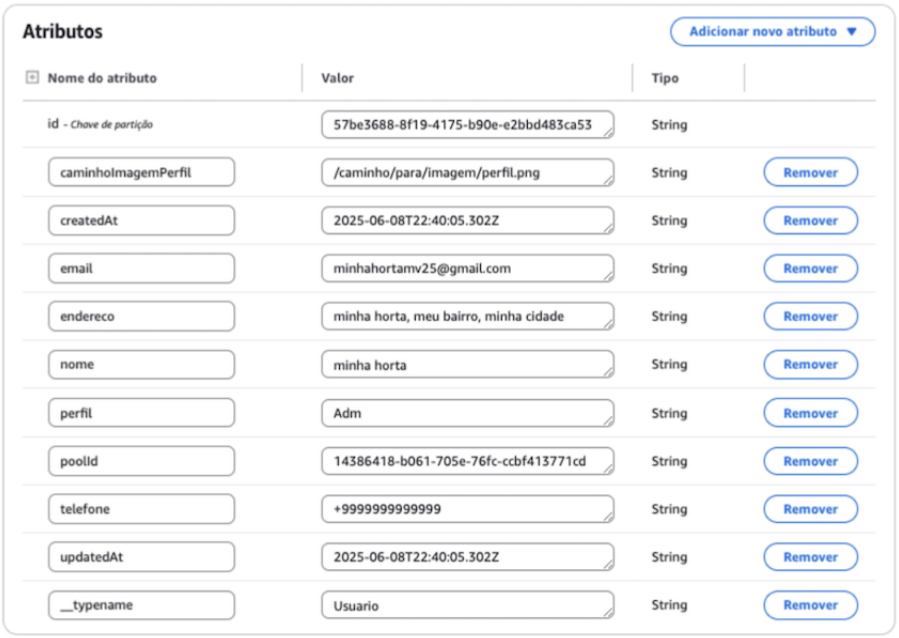
Fonte: Autor (2025)

A entidade Usuario (Figura 27) é composta por atributos obrigatórios, todos definidos como required() no *schema*, sendo estes validados no momento da criação, o que evita inconsistências no banco de dados.

- poolId: Identificador único do usuário no Cognito (User Pool), garantindo a associação entre autenticação e os dados da aplicação.
- nome: Nome completo do usuário.
- email: E-mail utilizado para contato e autenticação.
- telefone: Número de telefone para comunicação.
- endereco: Texto livre representando localização ou descrição.
- perfil: Nível de permissão ou função do usuário no sistema (ex: Adm, Produtor).
- caminhoImagemPerfil: Caminho da imagem de perfil do usuário no *bucket* S3.

Conforme padrão em bancos de dados não relacionais, os dados armazenados incluem informações de auditoria para controle e rastreabilidade. Como na coleção `Usuario`, são registrados automaticamente os campos de criação e atualização de cada item, permitindo acompanhar quando um registro foi criado e quando foi modificado pela última vez, sendo gerenciado de forma automatizada pela própria infraestrutura da AWS. A Figura 28 ilustra a visualização de um item na coleção no DynamoDB.

Figura 28 – Exemplo de registro de Usuario no DynamoDB



Fonte: Autor (2025)

Neste projeto, foi adotada uma arquitetura baseada na separação de responsabilidades para o desenvolvimento de um aplicativo Flutter integrado ao AWS Amplify. O principal objetivo dessa organização foi garantir que os dados se mantivessem atualizados em tempo real, sem exigir múltiplas requisições redundantes, utilizando o recurso de *subscriptions* do AppSync em conjunto com StreamController e Provider.

A estrutura central do projeto foi dividida em três camadas principais: modelos gerados pelo Amplify, arquivos de requisição de dados e provedores de estado. Os arquivos contidos na pasta models/aws são gerados automaticamente pelo AWS Amplify a partir do schema GraphQL definido previamente. Cada modelo representa uma entidade no banco de dados e segue a estrutura e tipagem especificadas nesse *schema*.

A camada de requisições, concentrada no arquivo requests.dart, foi projetada para ser o único ponto responsável por acessar os serviços da AWS. Essa abordagem centralizada permite que toda a lógica de leitura, escrita e *subscriptions* GraphQL esteja encapsulada e reutilizável. Por exemplo, a classe HortaRequest lida tanto com a obtenção das hortas associadas a um usuário quanto com a escuta de alterações em tempo real que possam afetar essa lista, como a inserção ou remoção feita por outros usuários.

Acima dessa camada, foi estruturado o gerenciamento de estado com Provider, usando classes como ListaHortas, ListaPlantas, ListaTelas, entre outras, localizadas na

pasta models/listas. Essas classes utilizam StreamController para propagar mudanças no estado dos dados e servem como ponte entre os widgets e a camada de acesso à nuvem. Isso permite que a interface do usuário seja reativa: ao navegar para uma tela que consome determinada lista, os dados já estarão disponíveis e atualizados, sem a necessidade de novas consultas ao banco.

A estrutura de telas foi organizada com base nos diferentes perfis de usuário. As telas compartilhadas por todos os tipos de usuário foram reunidas na pasta telas/home, representando a base da navegação comum da aplicação, como o menu lateral (drawer_app.dart). A partir daí, a interface se divide em três grupos principais:

- telas/produtor/: onde ficam concentradas as funcionalidades destinadas ao produtor, como o gerenciamento das hortas, plantas e envio de imagens para análise;
- telas/supervisor/: que contém as telas voltadas para supervisores, como a listagem e gestão de usuários vinculados;
- telas/adm/: para funcionalidades específicas de um administrador do sistema.

Essa divisão reflete diretamente a lógica de controle de acesso e facilita a manutenção, pois permite que cada tipo de usuário tenha seu fluxo de navegação isolado em seu próprio espaço dentro da estrutura do projeto.

Complementando essa organização, cada conjunto de telas possui uma subpasta componentes, onde ficam widgets reutilizáveis, mantendo o código mais limpo, modular e fácil de testar. A navegação entre as telas é centralizada no arquivo app_routes.dart, onde todas as rotas nomeadas são definidas de forma clara e padronizada.

3.5 Treinamento do modelo Mask R-CNN no SageMaker

Inicialmente, o treinamento do modelo Mask R-CNN foi realizado utilizando o framework MMDetection, configurado em um notebook Jupyter no ambiente do Amazon SageMaker. Apesar da modularidade e praticidade oferecidas por esse framework, optou-se por migrar para uma implementação direta em PyTorch visando reduzir a dependência de bibliotecas externas e evitar a necessidade de um gerenciador específico de treinamento e inferência. Essa abordagem oferece maior controle sobre o fluxo do modelo, facilita ajustes finos e amplia a flexibilidade da solução.

A Tabela 5 apresenta um resumo das principais características do conjunto de dados utilizado neste treinamento inicial, denominado `tomato_mixed`. Este *dataset* é

composto por um total de 804 imagens, sendo 643 destinadas ao treinamento e 161 ao teste. As imagens foram capturadas em alta resolução (3024×4032 e 3120×4160 píxeis), sob diferentes condições de iluminação, ângulo e fundo, simulando com fidelidade o ambiente real de cultivo.

O conjunto contempla seis classes de tomates, organizadas com base em dois critérios: o estágio de maturação — imaturo, jovem e maduro — e o tipo da variedade observada — *tomato big* (prefixo `b`) e *tomato little* (prefixo `l`). Assim, cada classe representa uma combinação única de estágio e variedade, permitindo ao modelo aprender a diferenciar não apenas a coloração, mas também aspectos morfológicos relevantes à detecção e classificação precisa. Ao todo, o conjunto de treinamento possui 7.781 caixas delimitadoras (*bounding boxes*), anotadas manualmente segundo o formato COCO.

Tabela 5 – Descrição geral do conjunto de dados *tomato_mixed*

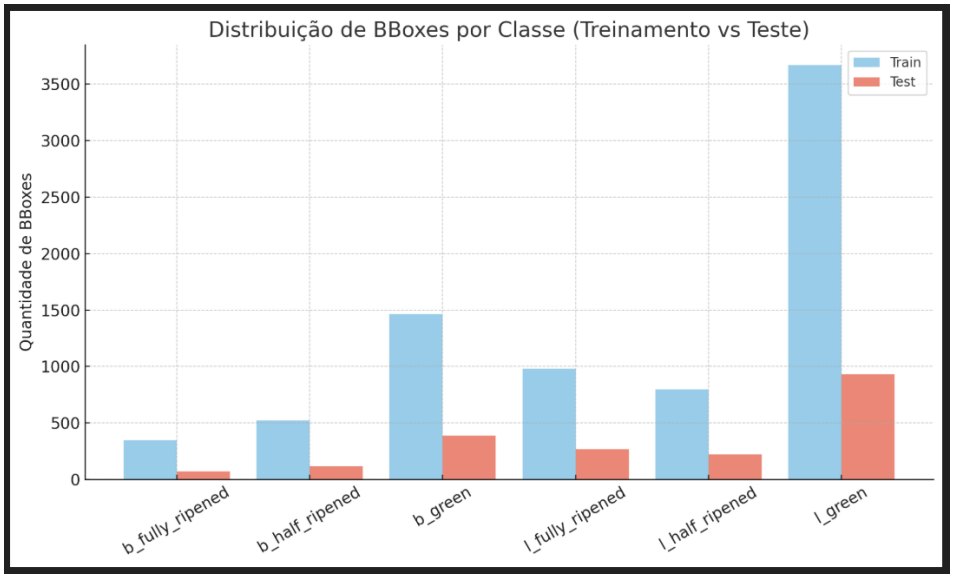
Item	**Descrição**
Nome	`tomato_mixed`
Imagens	643 (treinamento), 161 (teste)
Classes	6
Nomes das Classes	`b_fully_ripened`, `b_half_ripened`, `b_green`, `l_fully_ripened`, `l_half_ripened`, `l_green`
Total de BBoxes	Treinamento: 7.781
Resoluções	3024×4032, 3120×4160

Fonte: Autor (2025)

A Figura 29 detalha a distribuição das anotações por classe, tanto no conjunto de treinamento quanto no de teste. Observa-se que há uma maior incidência de tomates verdes “grandes” (`l_green`), seguidos pelos verdes “pequenos” (`b_green`).

A adoção de um conjunto de dados próprio e padronizado, como proposto neste projeto, representa uma evolução importante no processo de construção de modelos mais precisos e confiáveis. Diferentemente de *datasets* públicos genéricos, um conjunto construído com base em um domínio específico permite treinar modelos especializados, mais alinhados com o contexto e as demandas práticas do produtor.

Figura 29 – Gráfico de distribuição de *bounding boxes* `tomato_mixed`



Fonte: Autor (2025)

É importante destacar que esse novo conjunto de imagens não substitui os dados previamente utilizados, mas se soma a eles como uma etapa evolutiva. A intenção é promover maior equilíbrio entre as classes, tanto em termos de quantidade quanto de representatividade visual, enriquecendo a diversidade e a qualidade da base de dados.

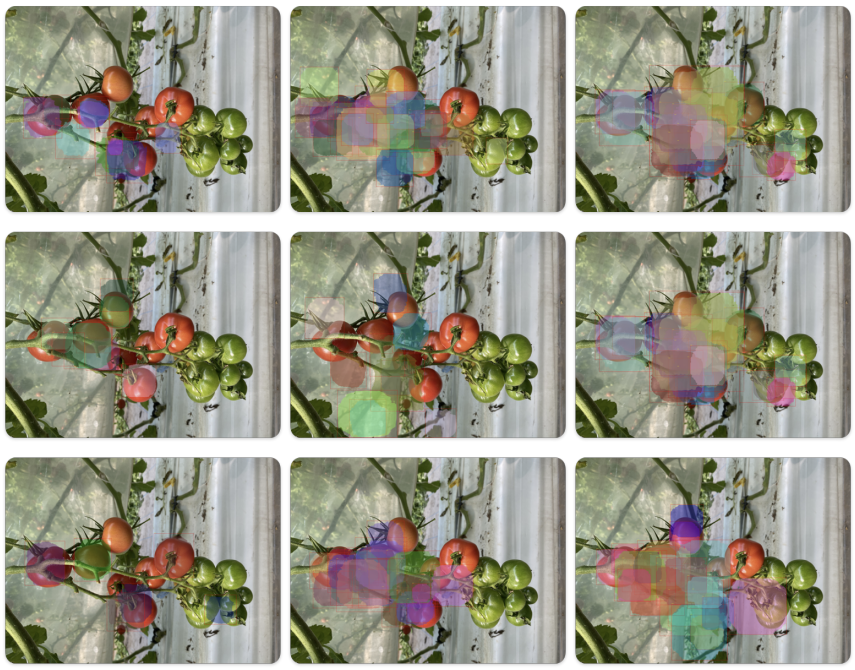
Os resultados obtidos com o modelo treinado a partir desse conjunto ampliado serão comparados aos resultados anteriores, permitindo verificar se a adição de imagens padronizadas produz o efeito esperado na acurácia, generalização e robustez da solução.

Para utilizar o MMDetection, seguiu-se o manual de instalação oficial (CHEN *et al.*, 2019), com a instalação de todas as dependências indicadas. Ainda assim, apesar de atender a todos os pré-requisitos indicados pela documentação, o processo exigiu uma série de ajustes manuais baseados em depuração (*debug*), devido a incompatibilidades entre versões e lacunas nos exemplos fornecidos. Diversas exceções surgiram mesmo após a configuração inicial, sendo necessário inspecionar logs, ajustar arquivos de configuração e, em alguns casos, modificar diretamente trechos do código-fonte.

O treinamento foi dividido em duas etapas, seguindo o manual LABORO.AI (2023): uma fase inicial com 48 épocas e outra complementar com 16, totalizando 64 épocas. Os principais hiperparâmetros incluíram taxa de aprendizado (*learning rate*) igual a 0,01, *momentum* de 0,9 e *weight decay* de 0,0001, com decaimento da taxa de aprendizado nos marcos definidos por *step* = [32, 44]. Na segunda etapa, que visou o refinamento do modelo com base nos pesos previamente treinados, os mesmos

hiperparâmetros foram mantidos, mas o agendamento de aprendizado foi adaptado para o novo ciclo, com *step* = [8, 14], respeitando a duração de 16 épocas. Como o ambiente já havia sido previamente configurado para o MMDetection, ao migrar para PyTorch puro, todas as bibliotecas-base já estavam instaladas, eliminando a necessidade de novas dependências para o desenvolvimento da nova versão. Durante o treinamento, foram salvas imagens com previsões intermediárias e os arquivos dos modelos ao final de cada época. Embora não utilizadas como visualizações finais, essas imagens desempenharam um papel importante no monitoramento qualitativo da evolução do desempenho do modelo. Na Figura 30, cada coluna representa momentos distintos do processo de aprendizagem: a primeira corresponde às previsões obtidas nas três primeiras épocas, onde observa-se que o modelo começa com um número baixo de frutos detectados e que a maioria deles eram maduros. A segunda reúne as geradas nas três etapas seguintes, observa-se que o número de frutos detectados aumenta, porém, com muitas detecções repetidas e em sua maioria furtos maduros. Na terceira apresenta as referentes às três épocas seguintes, onde o modelo começa a detectar os frutos imaturos além de melhorar a precisão das detecções de frutos maduros. Permitindo observar de forma comparativa a progressão do modelo ao longo de parte do treinamento. Os resultados serão analisados na seção de Resultados e Experimentos.

Figura 30 – Imagens da evolução das inferências



Fonte: Autor (2025)

3.6 Disponibilização do modelo no SageMaker e integração com S3

Após o treinamento do modelo Mask R-CNN no ambiente Jupyter do SageMaker, a próxima etapa foi torná-lo acessível para inferências automáticas, de forma escalável e integrada ao restante da arquitetura do projeto.

Para isso, o modelo foi empacotado em um endpoint do SageMaker Hosting Services, permitindo que ele fosse chamado sob demanda. O deployment foi feito utilizando o SDK do SageMaker, apontando para o artefato .pth treinado e a configuração de inferência personalizada, compatível com MMDetection. Isso garantiu que o endpoint estivesse sempre disponível, sem necessidade de reprocessar o modelo a cada requisição.

A inferência foi automatizada com base em eventos do Amazon S3. Toda vez que uma imagem nova é enviada para uma pasta específica no *bucket* — definida como o diretório de análises pendentes — um evento do tipo s3:ObjectCreated:* é disparado. Esse evento aciona uma função Lambda, cuja responsabilidade é fazer o seguinte:

- Download da imagem recebida.
- Envio da imagem ao endpoint SageMaker via requisição.
- Recebimento da resposta de inferência.
- Processamento e pós-tratamento dos resultados.
- Converter em imagens as máscaras segmentadas.
- Gerar JSON contendo os dados da inferência (classes, scores, bounding boxes).

Esses três resultados (imagem original, imagem com as máscaras segmentadas e o JSON da inferência) são então salvos em suas respectivas pastas no S3. Como etapa final do processo, a função Lambda realiza uma chamada ao AppSync (GraphQL) para registrar uma nova entrada na coleção de análises no banco de dados. Esse registro contém os seguintes dados:

- Referência à planta analisada (por ID).
- Caminhos das imagens geradas e do arquivo JSON salvos no S3.
- Data e hora da análise.
- Identificador do modelo utilizado.

Essa arquitetura permitiu construir um pipeline robusto e assíncrono, em que a inferência é disparada automaticamente com base no envio de arquivos ao S3. Todo o processo, desde a submissão da imagem até o registro da análise no banco de dados,

ocorre de forma automatizada, escalável e resiliente, permitindo que vários usuários alimentem o sistema sem sobrecarregar o aplicativo ou a interface Flutter. O aplicativo, por sua vez, escuta atualizações em tempo real por meio das *subscriptions* do AWS Amplify e exibe imediatamente os resultados ao usuário após a conclusão da inferência.

3.7 Fluxos do artefato

Esta seção descreve os principais fluxos implementados no artefato, dividindo os mesmos em três subseções: fluxos gerais, comuns a todos os usuários, fluxos do perfil de Produtor e fluxos do perfil de Supervisor. Inicialmente, apresenta-se o diagrama de casos de uso (Figura 31), que ilustra as interações entre os diferentes perfis de usuário e as funcionalidades oferecidas pela aplicação.

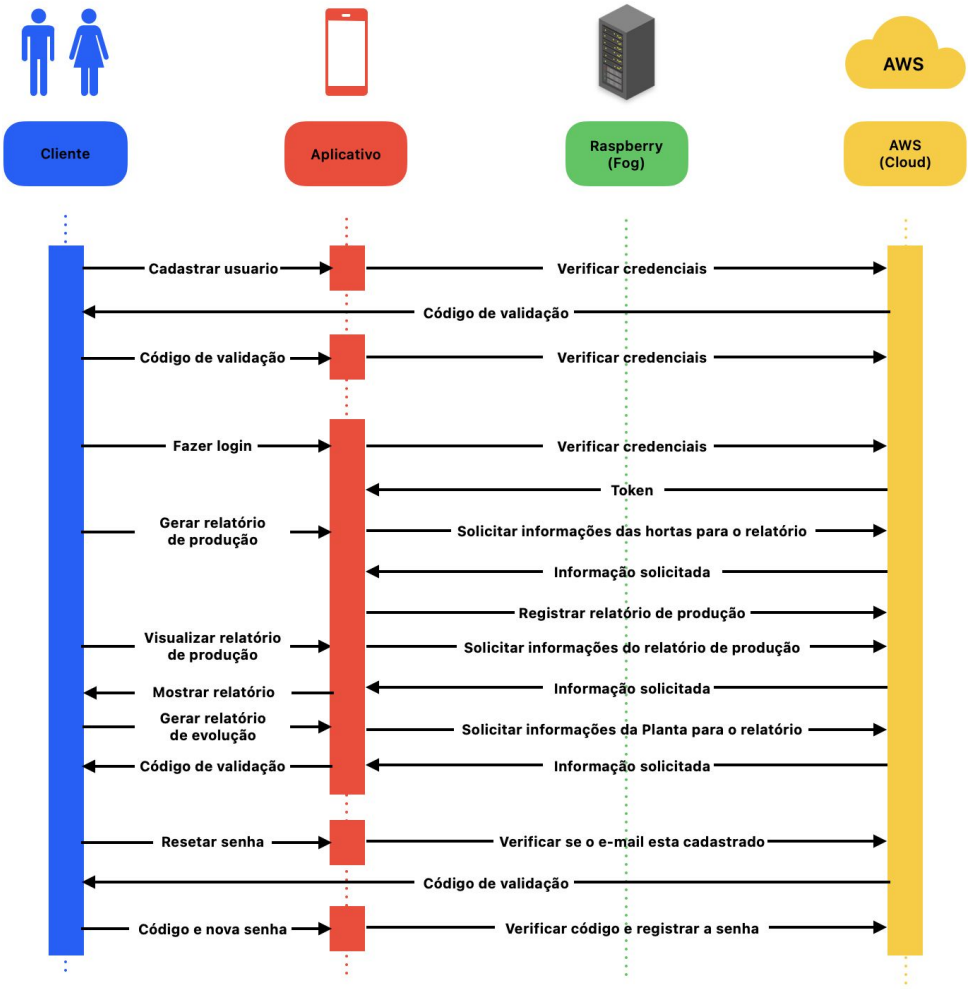
Figura 31 – Diagrama de casos de uso

Fonte: Fonte: Autor (2025).

3.7.1 Fluxos Gerais

Esta subseção descreve os fluxos gerais da aplicação, os quais são comuns a todos os perfis de usuário. Tais fluxos abrangem os processos de cadastro, autenticação, recuperação de senha e geração de relatórios (Figura 32).

Figura 32 – Diagrama de sequência dos fluxos gerais



Fonte: Fonte: Autor (2025).

O fluxo de cadastro de usuário, três primeiras linhas da Figura 32, telas ilustradas na figura 55, foi concebido com o objetivo de garantir a criação de contas seguras, aliando requisitos de segurança a uma experiência de uso adequada. No primeiro acesso à aplicação, o usuário é direcionado à tela de cadastro, na qual deve informar um endereço de e-mail e uma senha (Figura 55a). Nesse momento, o sistema realiza validações automáticas, verificando se o e-mail informado contém o caractere `@` e se a senha atende aos critérios de segurança previamente definidos. Caso o endereço de e-mail ainda não

esteja registrado no sistema, é enviado automaticamente um e-mail contendo um código de verificação. Esse código deve ser informado na tela de verificação de e-mail (Figura 55c) para a conclusão do processo de ativação da conta. Após a confirmação, o usuário é autenticado automaticamente e permanece logado na aplicação. No caso de primeiro acesso, o sistema direciona o usuário para a etapa de finalização do cadastro (Figura 55b), na qual são solicitadas informações complementares, como nome completo, telefone e endereço. Com o preenchimento desses dados, o perfil do usuário é concluído, permitindo o uso integral das funcionalidades da aplicação.

Após a criação da conta, os acessos subsequentes ocorrem por meio do fluxo de login, a quarta e quinta linha da figura 32, telas ilustradas na figura 56. Nessa etapa, o usuário informa o e-mail e a senha previamente cadastrados (Figura 56a). Uma vez validadas as credenciais, o sistema autentica o usuário e o redireciona para a tela inicial (Home) correspondente ao seu perfil (Figuras 56c e 56d).

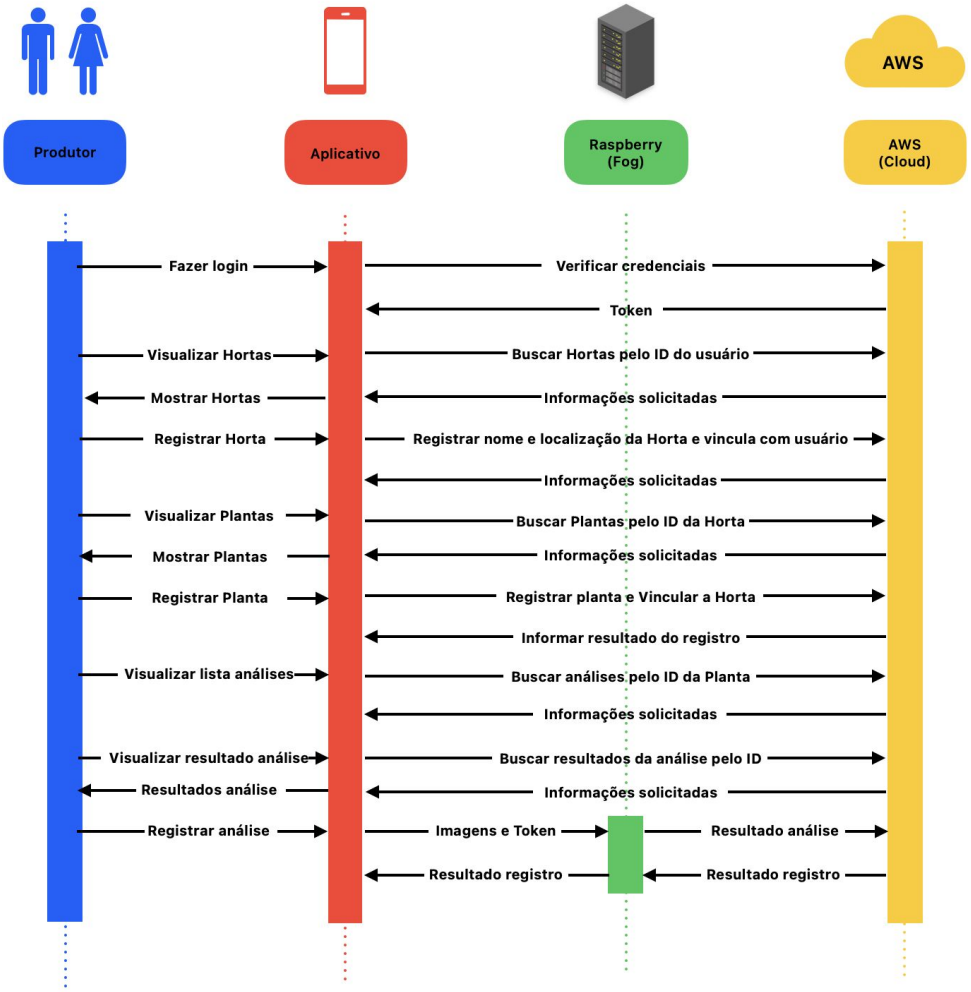
Para situações em que o usuário não se recorda da senha, a aplicação disponibiliza o fluxo de redefinição, ultimas três linhas da figura 32, as telas estão ilustradas na figura 57. Ao selecionar a opção “Esqueci minha senha”, o usuário é direcionado a uma tela na qual deve informar o endereço de e-mail associado à conta. Após a confirmação, o sistema envia automaticamente um e-mail contendo um link seguro para redefinição da senha. Ao acessar esse link, o usuário registra e confirma a nova senha e, após a validação, pode efetuar o login normalmente.

Além dos fluxos de autenticação, o sistema possibilita a geração de relatórios com base nas hortas cadastradas pelo usuário. Após a seleção do tipo de relatório desejado, o sistema processa as informações disponíveis e gera o documento correspondente. Os relatórios de produção, linhas seis a oito da Figura 32, telas na figura 58, apresentam as hortas localizadas dentro do raio de busca definido pelo usuário; caso não existam registros compatíveis com os parâmetros informados, o sistema exibe uma mensagem indicando a ausência de resultados. A visualização dos relatórios de produção ocorre em uma interface específica, ilustrada nas figuras 60 e 61, na qual o usuário pode consultar métricas relevantes para o acompanhamento das hortas cadastradas. De forma análoga, os relatórios de evolução da planta, linhas onze e doze da figura 32, são apresentados em uma tela dedicada (Figuras 62 e 63), permitindo o acompanhamento de indicadores de crescimento, histórico de análises e demais informações relacionadas ao desenvolvimento da planta selecionada.

3.7.2 Fluxos do Produtor

Esta subseção detalha os principais fluxos funcionais implementados para o perfil de Produtor na aplicação (Figura 33). Esses fluxos abrangem desde a visualização e gerenciamento das hortas e plantas até o envio de imagens para análise, geração de relatórios de produção das hortas e de evolução da planta, proporcionando uma experiência completa e intuitiva para os produtores que utilizam a plataforma.

Figura 33 – Diagrama de sequência dos fluxos produtor



Fonte: Fonte: Autor (2025).

O sistema disponibiliza duas formas de visualização das hortas cadastradas, linhas três e quatro da figura 33, telas ilustradas na figura 64. A primeira apresenta os registros em formato de lista (Figura 64b), enquanto a segunda consiste em um mapa (Figura 65b) exibindo a localização de cada horta registrada. Assim como nas telas anteriores, ao

acessar a tela, é apresentado um indicador de carregamento enquanto os dados estão sendo processados. Caso nenhuma horta esteja cadastrada, o sistema exibe uma mensagem informando a ausência de registros.

Para cadastrar uma nova horta, quinta e sexta linha da figura 33, telas ilustradas na figura 65, o usuário deve selecionar o botão correspondente. Ao ativá-lo, o sistema abre um componente que permite escolher a localização da horta no mapa (Figura 65b) e definir um nome para o registro (Figura 65c). Após a confirmação, as informações são enviadas para o *backend*, que armazena o novo cadastro.

O fluxo de visualização das plantas (Figura 66) inicia-se ao selecionar uma horta específica. O produtor é direcionado à tela de visualização de *cards* das plantas associadas àquela horta (Figura 66b). Ao clicar em um destes *cards*, o mesmo expande apresentando dois botões, um para levar à tela de análises da planta e outro para a visualização da evolução da mesma. Essa tela segue a mesma lógica de assinatura reativa, exibindo apenas as plantas pertencentes à horta selecionada e atualizando-se automaticamente conforme alterações ocorram no banco de dados. A assinatura dinâmica é novamente utilizada, de modo que apenas as análises associadas à planta em questão são carregadas e exibidas. Como na tela anterior, ao acessar a tela, é apresentado um indicador de carregamento circular centralizado até que os dados sejam obtidos (Figura 66a). Caso não exista nenhuma horta cadastrada, exibe-se a tela indicando a ausência de registros (Figura 66c).

Na tela de visualização de uma horta (Figura 67a), o produtor pode optar por adicionar uma nova planta. Neste caso, o fluxo de adição de planta (Figura 67) é mais simples: o usuário informa apenas o nome da planta (Figura 67b), ao confirmar, a aplicação cria o registro no banco de dados, associando-o à horta correspondente.

A visualização de análises (Figura 68) inicia-se ao clicar no botão “ver análises" do *card* de uma planta, o usuário é levado à tela de visualização de análises realizadas sobre aquela planta (Figura 68b). Cada análise corresponde a uma execução do modelo de inferência, e os dados exibidos nesta etapa são provenientes do banco de dados (endereço das imagens e resultado das análises) e *bucket* (imagens).

Cada análise contém os resultados das inferências feitas pelo modelo, como mostrado no capítulo anterior. O detalhamento dos resultados será apresentado na subsubseção seguinte. A assinatura dinâmica é novamente utilizada, de modo que apenas as análises associadas à planta em questão são carregadas e exibidas. Como na tela anterior, ao acessar a tela, é apresentado um indicador de carregamento circular centralizado até que os dados sejam obtidos (Figura 68a). Caso não exista nenhum

análise cadastrado, a tela exibe um texto indicando a ausência de registros (Figura 68c). A visualização detalhada da análise (Figura 69) inicia-se ao clicar em uma das análises, acessamos o detalhamento de seus dados. A tela de visualização (Figura 69a) apresenta os dados que incluem quantificação de frutos por estágio de maturação, gráfico de porcentagem dos frutos por estágio (Figura 69b) e segmentação visual, apresentados por meio de imagens processadas (Figura 69c).

O fluxo de registro de análise (Figura 70) é independente, se inicia a partir da Home, ao clicar na opção “Registrar Imagens”. As etapas do processo são:

1. O sistema carrega e exibe as hortas do produtor (Figura 70a).
2. O usuário escolhe uma horta e uma das plantas associadas (Figura 70c).
3. O produtor decide se deseja captar imagens em tempo real usando a câmera do dispositivo ou selecionar imagens já armazenadas em sua galeria.
4. À medida que as imagens são selecionadas, são exibidas em miniaturas (Figura 71).
5. Após a seleção de quatro imagens, o produtor clica em “Enviar” (Figura 70d).

Nesse momento, as imagens são enviadas para o *bucket*, o que aciona uma função Lambda configurada previamente. Essa função realiza as seguintes etapas:

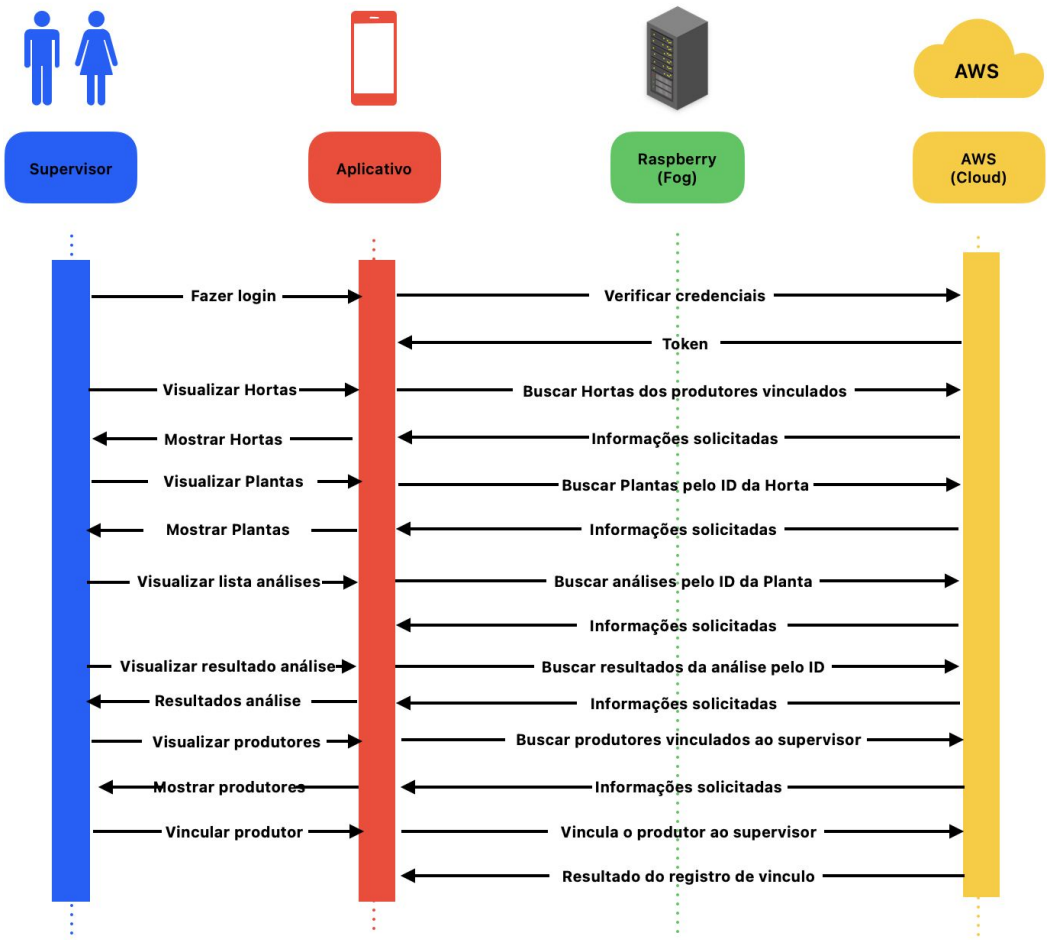
- Recupera as imagens do *bucket* original;
- Executa o modelo de inferência para análise da planta;
- Armazena os resultados processados em outra pasta do *bucket* do S3.

Esses resultados são posteriormente carregados e exibidos na tela de visualização de detalhes da análise, fechando o ciclo completo de coleta, processamento e visualização.

3.7.3 Fluxos do Supervisor

Esta subseção detalha os principais fluxos funcionais implementados para o perfil de Supervisor na aplicação (Figura 34). Esses fluxos abrangem desde a visualização e gerenciamento das hortas e plantas dos produtores vinculados, até a geração de relatórios de produção das hortas e de evolução da planta, proporcionando uma experiência completa e intuitiva para os supervisores que utilizam a plataforma.

Figura 34 – Diagrama de sequência dos fluxos supervisor



Fonte: Fonte: Autor (2025).

O fluxo de visualização das hortas dos meus produtores, linhas três e quatro da figura 34, telas da mesma se encontram na figura 72. Onde o supervisor tem acesso à tela "Hortas dos Meus Produtores", disponível na tela Home. Esta interface apresenta todas as hortas registradas pelos produtores vinculados ao supervisor (Figura 72b). A tela utiliza um sistema de streaming reativo, baseado em uma inscrição (*subscription*) às alterações no banco de dados, exibindo apenas as hortas vinculadas aos produtores associados ao usuário logado. Ao acessar a tela, é apresentado um indicador de carregamento circular centralizado até que os dados sejam obtidos (Figura 72a). Caso não exista nenhuma horta cadastrada, a tela exibe um texto indicando a ausência de registros. Dito isso vale ressaltar que aqui também te temos duas formas de visualizar as hortas, uma em lista e outra em mapa e o botão de gerar relatório de produção.

O fluxo de visualização das plantas do meu produtor, linhas cinco e seis da figura 34, telas da mesma se encontram na figura 73. Inicia-se ao selecionar uma horta

específica. O supervisor é direcionado à tela de visualização das plantas associadas àquela horta (Figura 73b). Ao clicar no *card* de uma planta o mesmo expande apresentado dois botoes, uma para levar a tela de análises da planta e outro para a visualização da evolução da mesma. Essa tela segue a mesma lógica de assinatura reativa, exibindo apenas as plantas pertencentes à horta selecionada e atualizando-se automaticamente conforme alterações ocorram no banco de dados.

O fluxo de visualização da lista de análises, linhas sete e oito da figura 34. Inicia-se ao clicar no em uma planta específica. O supervisor é direcionado à tela de visualização de *cards* das análises associadas àquela planta (Figura 68). Ao clicar em um destes *cards*, o mesmo expande apresentando o botão para a visualização detalhada da análise.

O fluxo de visualização do resultado análises, linhas nove e dez 34, telas da mesma na figura 68. Inicia-se ao clicar no botão “ver análises" do *card* de uma planta, o usuário é levado à tela de visualização de análises realizadas sobre aquela planta (Figura 68b). Cada análise corresponde a uma execução do modelo de inferência, e os dados exibidos nesta etapa são provenientes do banco de dados (endereço das imagens e resultado das análises) e *bucket* (imagens).

Cada análise contém os resultados das inferências feitas pelo modelo, como mostrado no capítulo anterior. O detalhamento dos resultados será apresentado na subsubseção seguinte. A assinatura dinâmica é novamente utilizada, de modo que apenas as análises associadas à planta em questão são carregadas e exibidas. Como na tela anterior, ao acessar a tela, é apresentado um indicador de carregamento circular centralizado até que os dados sejam obtidos (Figura 68a). Caso não exista nenhum análise cadastrado, a tela exibe um texto indicando a ausência de registros (Figura 68c). A visualização detalhada da análise (Figura 69) inicia-se ao clicar em uma das análises, acessamos o detalhamento de seus dados. A tela de visualização (Figura 69a) apresenta os dados que incluem quantificação de frutos por estágio de maturação, gráfico de porcentagem dos frutos por estágio (Figura 69b) e segmentação visual, apresentados por meio de imagens processadas (Figura 69c).

O fluxo de lista dos meus produtores, linhas onze e doze da figura 34. Inicia-se ao clicar na opção “Meus Produtores” disponível na tela Home. O supervisor é direcionado à tela de visualização dos produtores vinculados a ele (Figura 74b). Além de um botão de adicionar produtor. A tela segue a mesma lógica de assinatura reativa, exibindo apenas os produtores vinculados ao supervisor autenticado e atualizando-se automaticamente conforme alterações ocorram no banco de dados.

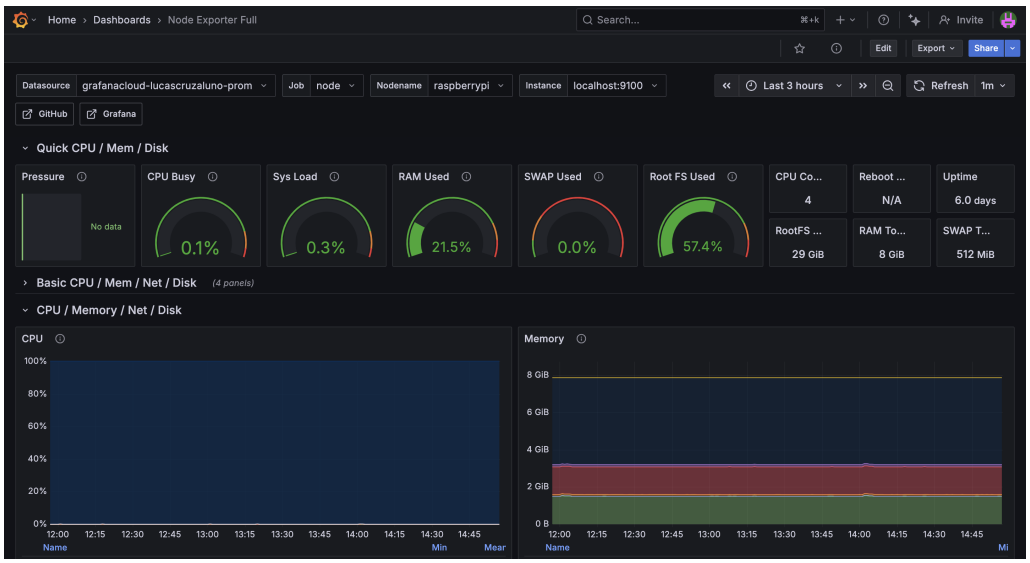
O fluxo de adicionar produtor a supervisor, linhas treze e quatorze da figura 34, telas da mesma se encontram na figura 74. Inicia-se ao clicar no botão “Adicionar Produtor” disponível na tela de lista dos meus produtores. O supervisor é direcionado à tela de adição de produtor (Figura 74b), onde se apresenta todos os produtores cadastrados na plataforma em uma lista de *cards*, caso o produtor já este vinculado ao produtor o carde mostra um símbolo de check, caso contrario um símbolo de mais. Ao clicar no símbolo de mais o produtor é vinculado ao supervisor autenticado. A tela segue a mesma lógica de assinatura reativa, atualizando vínculo do produtor referente ao *card* automaticamente conforme alterações ocorram no banco de dados.

3.8 Migração para arquitetura Fog

Na etapa de implantação do modelo em ambiente Fog, tornou-se necessário avaliar o desempenho do sistema diretamente na Raspberry Pi. Para isso, configurou-se um conjunto de ferramentas de monitoramento que possibilitam a análise em tempo real e a visualização histórica das métricas coletadas.

Inicialmente, utilizou-se a ferramenta Perf, capaz de registrar informações detalhadas sobre utilização de CPU, memória, processos, trocas de contexto e interrupções. Esses dados ilustram o impacto do modelo de visão computacional sobre o hardware limitado do dispositivo.

Figura 35 – Dashboard Grafana



Fonte: Fonte: Autor (2025).

Além disso, foram implantados o Prometheus, responsável pela coleta estruturada das métricas do sistema, e o Grafana (Figura 35), utilizado para a criação de dashboards de visualização. Essa integração permitiu acompanhar a execução dos testes em diferentes cenários de carga (100, 10 e 1 usuário), facilitando a análise comparativa do comportamento do sistema e a identificação dos limites de saturação.

A infraestrutura experimental foi implementada em um Raspberry Pi 5, baseada na arquitetura ARMv8 de 64 bits. A placa é equipada com o processador Quad-Core Cortex-A76 a 2,4 GHz, memória de 8GB, conectada via Ethernet Gigabit diretamente ao modem. Sobre ela, foi instalado o sistema operacional *Raspberry Pi OS (64 bits)* e configurado o ambiente com as bibliotecas da Lista a seguir.

- torch == 2.1.2
- torchvision == 0.16.2
- numpy == 1.26.4
- openmim
- mmengine
- "mmcv == 2.1.0"
- mmdet

Durante a migração, observaram-se problemas de compatibilidade, uma vez que a Raspberry Pi utiliza arquitetura ARM, enquanto muitos tutoriais e exemplos de uso de frameworks de *deep learning* são baseados em arquiteturas da Intel. A solução encontrada partiu da experiência de usuários de macOS, que também utilizam processadores ARM, recomendando o uso das versões exatas das bibliotecas apresentadas anteriormente.

Como complemento à avaliação de desempenho, realizou-se também a análise de segurança da infraestrutura Fog por meio da implantação da plataforma Wazuh. O servidor central do Wazuh foi instalado no MacBook citado anteriormente, utilizando contêineres Docker, o que facilitou a configuração, o isolamento dos serviços e a reprodutibilidade do ambiente. Na Raspberry Pi 5, foi configurado um agente Wazuh responsável pela coleta de logs, monitoramento de integridade, análise de vulnerabilidades e envio contínuo das informações para a central de gerenciamento.

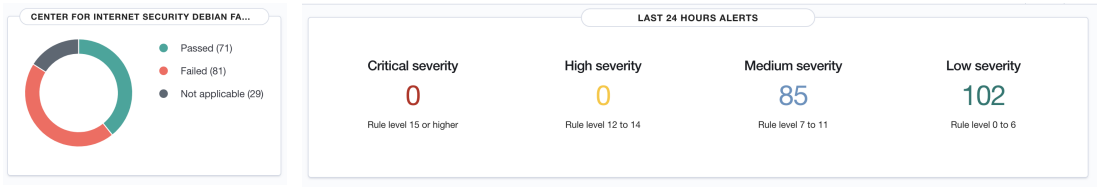
Os resultados obtidos indicaram a ausência de vulnerabilidades graves no ambiente monitorado. A análise de segurança não identificou ocorrências classificadas como críticas ou de alta severidade, tendo sido registrados 0 alertas críticos, 0 alertas de severidade alta, 85 alertas de severidade média e 102 alertas de severidade baixa.

Adicionalmente, foi executada a verificação de conformidade com o *Center for Internet Security (CIS) Debian Family Linux Benchmark v1.0.0*, cujos resultados apontaram 71 verificações aprovadas, 81 não aprovadas e 29 não aplicáveis, resultando em um índice geral de conformidade de 46% (Figura 36).

De forma geral, os achados evidenciam que, embora o ambiente não esteja totalmente aderente às recomendações rígidas de hardening propostas pelo CIS, não foram identificadas falhas críticas que comprometessem a operação do sistema. Esses resultados reforçam a viabilidade da infraestrutura Fog proposta para o cenário experimental, ao mesmo tempo em que indicam oportunidades de aprimoramento futuro no fortalecimento da postura de segurança do dispositivo.

Figura 36 – Resultados da análise de segurança utilizando a plataforma Wazuh

(a) (b)



Fonte: Autor (2025)

4 RESULTADOS E DISCUSSÕES

Este capítulo apresenta os resultados e experimentos realizados visando validar o modelo de detecção e segmentação proposto, bem como o funcionamento do artefato desenvolvido. Inicialmente, descrevem-se os testes utilizando os critérios padronizados do conjunto COCO, com o objetivo de avaliar o desempenho quantitativo do modelo nas tarefas de detecção de objetos e segmentação de instâncias.

Em seguida, apresenta-se o processo de validação do artefato, conduzido com foco na verificação da conformidade funcional dos fluxos implementados, da integridade dos dados, da integração entre serviços e da qualidade da experiência do usuário. Considerando as melhores práticas de *Quality Assurance* (QA), adotou-se uma abordagem sistemática para garantir a cobertura dos pontos críticos da aplicação.

Os testes foram organizados em categorias distintas, abrangendo desde as funcionalidades essenciais — como cadastro e autenticação — até os fluxos específicos voltados ao perfil de produtor, à integração com os serviços da AWS e a aspectos relacionados à segurança, ao desempenho e à usabilidade.

Por fim, descrevem-se os experimentos realizados no dispositivo Raspberry Pi, que incluem a validação do modelo em ambiente embarcado, o desenvolvimento da API de inferência e os testes de eficiência sob diferentes níveis de carga. Esses resultados permitem avaliar o comportamento do sistema em um contexto de recursos limitados, aproximando-se do cenário real de uso pretendido.

4.1 Testes com Common Objects in Context

Iniciam-se os testes com o uso da ferramenta COCO. A Tabela 6 apresenta os resultados quantitativos obtidos, respectivamente, para as tarefas de detecção com caixas delimitadoras (*bounding boxes*) e segmentação de instâncias, ambas utilizando o conjunto `tomato_mixed`.

As imagens desse conjunto possuem resolução média de aproximadamente 3000×4000 píxeis. Esse fator é crucial para interpretar os resultados obtidos no contexto das métricas COCO, que categorizam os objetos anotados em três faixas de área — *small*, *medium* e *large* — com base em seus tamanhos em píxeis (ver Tabela 7). Essas categorias são fixas e não podem ser ajustadas para diferentes contextos ou resoluções de imagem.

Instâncias *small* são aquelas com área inferior a 1.024 píxeis (menores que

32×32), o que representa uma fração mínima da imagem em nosso caso. Em imagens com resolução tão alta quanto as utilizadas neste trabalho, objetos com essa dimensão tornam-se praticamente imperceptíveis e, na prática, irrelevantes para a detecção de frutos.

A comparação entre as métricas demonstra que o desempenho foi satisfatório em ambas as tarefas, com valores de AP superiores a 66% na métrica principal (IoU = 0,50:0,95), AR de 77% e precisão elevada para objetos de grande porte (AP de 69% e AR de 79%). Além disso, observa-se que a precisão média sob a métrica mais tolerante (AP@0,50) superou 84% em ambos os casos, enquanto a métrica mais rigorosa (AP@0,75) manteve-se consistente, acima de 76%, evidenciando que o modelo é capaz de localizar e segmentar os frutos com boa sobreposição em relação às anotações de referência. Esses resultados demonstram robustez mesmo sob critérios mais exigentes de avaliação. Os resultados do modelo final são apresentados na Figura 37. Já na matriz de confusão na Figura 38, podemos observar que o modelo apresenta alta precisão na classificação dos frutos, com poucas confusões entre as classes. Os maiores confusões ocorrem entre as classes "verde" (*b_green* e *l_green*) e o *background*, o que é esperado devido à semelhança visual entre esses frutos imaturos e o fundo da imagem.

Tabela 6 – Resultados da avaliação do modelo

Métrica	**Valor bounding boxes**	**Valor segmentação**
AP (IoU=0.50:0.95)	66.500%	67.700%
AP (IoU=0.50)	84.400%	84.000%
AP (IoU=0.75)	76.000%	76.400%
AP (small)	0%	0%
AP (medium)	18.200%	15.400%
AP (large)	69.800%	71.200%
AR (maxDets=100)	76.200%	77.400%
AR (maxDets=300)	76.200%	77.400%
AR (maxDets=1000)	76.200%	77.400%
AR (small)	0%	0%
AR (medium)	26.000%	25.900%
AR (large)	79.600%	80.900%

Legenda: AP = Average Precision; AR = Average Recall.

Fonte: Autor (2025)

Tabela 7 – Critério de classificação de tamanho de objetos segundo o COCO

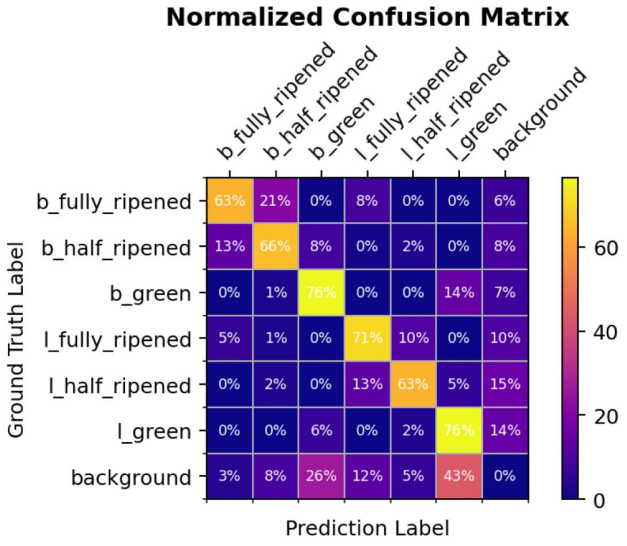
Categoria	**Área (em px^2)**
Small	área < 32 × 32 = 1.024
Medium	32 × 32 <= área < 96 × 96 = 9.216
Large	área >= 96 × 96 = 9.216

Fonte: Autor (2025)

Figura 37 – Resultados do modelo final

Fonte: Autor (2025)

Figura 38 – Matriz de confusão normalizada



Fonte: Autor (2025)

4.2 Testes do Modelo na Raspberry Pi

Com o modelo já funcional na Raspberry Pi, torna-se essencial realizar novos testes de acurácia utilizando o conjunto de dados COCO. Essa necessidade justifica-se pelas diferenças arquiteturais e de ambiente entre a execução original no Amazon SageMaker e a nova implementação na Raspberry Pi. Durante a migração, foi necessário ajustar bibliotecas e dependências para compatibilidade com a arquitetura ARMv8 (ver Seção 3.8), o que pode impactar diretamente o desempenho e a precisão do modelo. Assim, os testes visam validar se o comportamento do modelo permanece consistente e se as modificações realizadas não comprometeram a qualidade das inferências.

Após verificada a eficácia do modelo na Raspberry Pi, desenvolveu-se a API utilizando um *script* em Python. Inicialmente, ela apenas recebia uma imagem, realizava a inferência, armazenava os resultados (imagem com *bounding boxes* e JSON com informações) e devolvia o *status* da requisição e, em caso de sucesso, o nome do arquivo analisado. A função criada anteriormente para a API, agora é responsável por:

1. verificar se o token do *request* é válido;
2. iniciar o temporizador de resposta da requisição;
3. receber uma imagem via *request*;
4. armazenar a imagem localmente;
5. usar o modelo para realizar inferências na imagem;
6. armazenar os resultados localmente;
7. verificar possível sobreposição de inferências;
8. enviar os resultados ao *bucket* S3;
9. eliminar os resultados armazenados localmente;
10. parar o temporizador e devolver a *key* do arquivo criado e o tempo de resposta.

Além disso, adicionou-se uma etapa de validação na API, que exige a passagem de um token criado pela AWS para o usuário específico. Ao final de cada inferência, implementou-se a verificação da sobreposição dos *bounding boxes*, de modo que, caso a sobreposição fosse superior a 30%, prevalecesse a inferência com maior precisão. Após todos os ajustes, abriu-se uma porta no Wi-Fi para permitir o acesso à API via internet, limitando-se as portas acessíveis por motivos de segurança.

Com a API funcional e acessível, iniciou-se a etapa de testes da API. Para essa finalidade, criou-se um *script* em Python, executado no computador pessoal do

pesquisador, um MacBook Pro (8 GB de RAM) com o sistema operacional macOS Ventura. Utilizou-se a base de dados *tomato_mix*, contendo 161 imagens para teste. No *script*, definiu-se uma quantidade *N* de *batches*, correspondente à quantidade de usuários concorrentes, e estabeleceu-se o *batch size* em 4, equivalente às quatro imagens enviadas para analisar uma planta. O *script* realizou um *sort* nas imagens de teste e selecionou aquelas a serem enviadas. Para fins de avaliação, criaram-se três cenários:

1. 1 usuário realizando análise (4 imagens);
2. 10 usuários realizando análise simultaneamente (40 imagens);
3. 100 usuários realizando análise simultaneamente (400 imagens).

4.2.1 Resultados Iniciais

Os resultados obtidos nesta etapa têm como objetivo apresentar o comportamento geral do sistema em três níveis distintos de carga, destacando como a Raspberry Pi 5 responde ao aumento progressivo do volume de requisições. Para isso, analisaram-se os tempos de processamento, a utilização da CPU, os eventos de troca de contexto e a pressão sobre memória, tomando como referência a execução do modelo Mask R-CNN com diferentes quantidades de imagens e usuários.

Neste cenário inicial, apenas um *worker* realizou o processamento. A utilização de CPU permaneceu baixa e o tempo total de execução foi elevado, refletindo a ausência de paralelismo. Os eventos de troca de contexto e os *page-faults* ocorreram em níveis reduzidos, evidenciando baixa competição por recursos e impacto mínimo sobre a memória. O desempenho geral permaneceu estável, mas limitado pela operação isolada de um único *worker*.

Tabela 8 – Métricas de desempenho do sistema: 1º cenário com um *worker*

204.130.36 msec	task-clock	# 0.721 CPUs utilized
38.301	context-switches	# 187.63 /sec
2.589.028	page-faults	# 12.683 K/sec
488.315.084.665	cycles	# 2.392 GHz
570.786.395.867	instructions	# 1.17 insn per cycle
56.824.399.164	branches	# 278.373 M/sec
244.371.150	branch-misses	# 0.43% of all branches

Fonte: Autor (2025)

Observou-se melhora significativa no tempo médio por imagem, passando de 46 segundos por imagem para 26 segundos por imagem, e maior aproveitamento dos recursos de CPU passando de 0,721 CPUs para 1,911 CPUs. O aumento no número de *context-switches* indica maior coordenação entre processos, enquanto a elevação dos *page-faults* sinaliza maior demanda de memória. Ainda assim, o desempenho manteve-se estável, demonstrando a eficiência da execução paralela.

Tabela 9 – Métricas de desempenho do sistema: 2º cenário com um *worker*

1.991.599.29 sec	task-clock	# 1.911 CPUs utilized
390.438	context-switches	# 196.042 /sec
23.589.408	page-faults	# 11.844 K/sec
4.767.329.474.704	cycles	# 2.394 GHz
5.509.308.388.780	instructions	# 1.16 insn per cycle
543.141.768.401	branches	# 272.716 M/sec
2.336.277.954	branch-misses	# 0.43% of all branches

Fonte: Autor (2025)

O sistema apresentou tempos de execução bastante superiores e registrou forte aumento em trocas de contexto e requisições de memória. Apesar da intensificação da carga, o tempo médio por análise manteve-se relativamente estável, indicando que os dois *workers* foram capazes de sustentar a operação, ainda que próximos ao limite.

Tabela 10 – Métricas de desempenho do sistema: 3º cenário com um *worker*

19.766.618.88 msec	task-clock	# 1.975 CPUs utilized
4.068.306	context-switches	# 205.817 /sec
218.138.810	page-faults	# 11.036 K/sec
47.145.025.427.562	cycles	# 2.385 GHz
53.271.389.667.319	instructions	# 1.13 insn per cycle
649.962.435.984	LLC-loads	# 32.882 M/sec
434.452.431.664	LLC-load-misses	# 66.84% of all LLC accesses

Fonte: Autor (2025)

4.2.2 Resultados no Servidor

Os resultados evidenciaram limitações do uso de um único *worker* e a necessidade de paralelismo para melhorar o desempenho. No entanto, instanciar múltiplos *workers*

exige que cada um carregue o modelo individualmente. O terceiro cenário resultou em *timeouts*. O ajuste do parâmetro `-timeout 3000` resolveu o problema, ainda que represente potencial vulnerabilidade. Na sequência, novos testes revelaram estouro de memória no terceiro cenário, solucionado por meio de uma rotinas de limpeza de cache.

Após os ajustes, retomaram-se os testes no servidor com dois *workers* em operação. Os resultados a seguir sintetizam os principais achados em cada cenário.

Com apenas um *worker*, o uso da CPU permaneceu baixo e o sistema operou com pouca pressão sobre memória e processos. Os tempos de execução mostraram-se consistentes, mas o desempenho geral permaneceu limitado pela ausência de paralelismo.

Tabela 11 – Métricas de desempenho do sistema: 1º cenário com dois *workers*

513.28 msec	task-clock	# 0.004 CPUs utilized
1.403	context-switches	# 2.733 K/sec
57	page-faults	# 111.051 /sec
1.084.871.329	cycles	# 2.114 GHz
469.903.380	instructions	# 0.43 insn per cycle
101.918.023	branches	# 198.563 M/sec
4.162.041	branch-misses	# 4.08% of all branches
196.924.35 msec	task-clock	# 1.443 CPUs utilized
31.623	context-switches	# 160.585 /sec
2.369.394	page-faults	# 12.032 K/sec
469.822.285.669	cycles	# 2.386 GHz
554.824.342.374	instructions	# 1.18 insn per cycle
53.505.099.385	branches	# 271.704 M/sec
247.349.934	branch-misses	# 0.46% of all branches

Fonte: Autor (2025)

A presença de dois *workers* resultou em melhor distribuição da carga e maior aproveitamento da CPU. Houve aumento natural nos eventos de troca de contexto e nas solicitações de memória, mas o sistema manteve comportamento estável, processando as imagens com maior rapidez e eficiência em relação ao cenário inicial.

Tabela 12 – Métricas de desempenho do sistema: 2º cenário com dois *workers*

988.822.99 msec	task-clock	# 1.406 CPUs utilized
310.410	context-switches	# 313.919 /sec
10.008.410	page-faults	# 10.122 K/sec
2.357.954.483.793	cycles	# 2.385 GHz
2.524.700.916.712	instructions	# 1.07 insn per cycle
261.465.193.100	branches	# 264.421 M/sec
1.181.612.721	branch-misses	# 0.45% of all branches
1.109.342.74 msec	task-clock	# 1.573 CPUs utilized
325.714	context-switches	# 293.610 /sec
11.341.063	page-faults	# 10.223 K/sec
2.645.083.665.649	cycles	# 2.384 GHz
2.857.135.127.429	instructions	# 1.08 insn per cycle
294.775.723.007	branches	# 265.721 M/sec
1.320.518.736	branch-misses	# 0.45% of all branches

Fonte: Autor (2025)

Sob carga extrema, o sistema elevou substancialmente o volume de operações, tanto em CPU quanto em memória. O número de *context-switches* e de *page-faults* cresceu acentuadamente, indicando maior movimentação interna de processos. Ainda assim, os dois *workers* mantiveram a execução estável e sem falhas críticas.

Tabela 13 – Métricas de desempenho do sistema: 3º cenário com dois *workers*

10.875.407.82 msec	task-clock	# 1.380 CPUs utilized
4.429.873	context-switches	# 407.329 /sec
103.265.385	page-faults	# 9.495 K/sec
25.899.425.137.755	cycles	# 2.381 GHz
27.282.932.413.712	instructions	# 1.05 insn per cycle
2.900.651.446.990	branches	# 266.717 M/sec
12.799.533.530	branch-misses	# 0.44% of all branches
11.191.947.25 msec	task-clock	# 1.419 CPUs utilized
4.511.783	context-switches	# 403.128 /sec
116.206.863	page-faults	# 10.383 K/sec
26.653.291.636.533	cycles	# 2.381 GHz
28.103.619.584.199	instructions	# 1.05 insn per cycle
2.979.477.586.133	branches	# 266.216 M/sec
13.297.266.752	branch-misses	# 0.45% of all branches

Fonte: Autor (2025)

Os cenários intermediário e avançado demonstram que a execução paralela melhora substancialmente o desempenho, reduzindo o tempo total de análise e possibilitando maior *throughput*. Observa-se também que o principal fator limitante não

é a capacidade de processamento bruto, mas a pressão sobre a memória. Isso revela o limite operacional prático para a Raspberry Pi 5 utilizada: o dispositivo sustenta com estabilidade até dois *workers*.

Em síntese, os cenários com dois *workers* apresentaram aumento expressivo de desempenho em comparação ao cenário inicial. A distribuição da carga entre processos melhorou a utilização da CPU e reduziu o tempo total necessário para o processamento. Entretanto, os contadores revelaram que o principal limitador não é a CPU, mas a memória, especialmente o acesso ao LLC.

Tabela 14 – Resultados iniciais do servidor

Imagens	**Tempo de resposta**	**Tempo médio**	**Worker 1**	**Worker 2**
4	186 seg	46 seg.	1.021 CPUs.	–
40	1042 seg. (17 min e 36 seg.)	26 seg.	1.911 CPUs.	–
400	10007 seg. (2 h e 47 min.)	25 seg.	1.975 CPUs.	–
4	136 seg. (2 min 15 seg.)	34 seg.	1.443 CPUs.	0.004 CPUs.
40	703 seg. (11 min 42 seg.)	17 seg.	1.406 CPUs.	1.573 CPUs.
400	7882 seg. (2 h 11 min)	19 seg.	1.380 CPUs.	1.419 CPUs.

Fonte: Autor (2025)

A Raspberry Pi 5 demonstrou capacidade para operar de forma estável com até dois *workers* executando o Mask R-CNN. Acima desse ponto, o sistema enfrenta limites práticos relacionados ao consumo de memória e ao custo computacional do modelo.

Tabela 15 – Resultados finais do servidor

Métrica	**Cenário 1**	**Cenário 10**	**Cenário 100**
CPU Total Utilizada	1.447 CPUs	2.979 CPUs	2.799 CPUs
Instructions per Cycle (IPC)	0.43 – 1.18	1.07–1.08	1.05
L1 Miss Rate	2.7–3.1%	3.0%	3.0%
LLC Miss Rate	62–65%	71–72%	74%

Fonte: Autor (2025)

4.2.3 Análises da Dashboard do Grafana

Nesta subseção apresentam-se as análises dos principais indicadores de desempenho obtidos por meio da *dashboard* do Grafana durante os testes de carga realizados. Essa ferramenta foi utilizada para o monitoramento em tempo real dos

recursos computacionais do sistema, possibilitando a avaliação de métricas como uso de CPU, memória, processos, carga do sistema e comportamento do agendador.

O objetivo consiste em identificar o impacto do aumento de usuários simultâneos sobre o desempenho do sistema, analisando sua capacidade de processamento e a eficiência na distribuição de carga entre os núcleos de CPU. As observações consideram três cenários distintos de execução: testes com 1, 10 e 100 usuários.

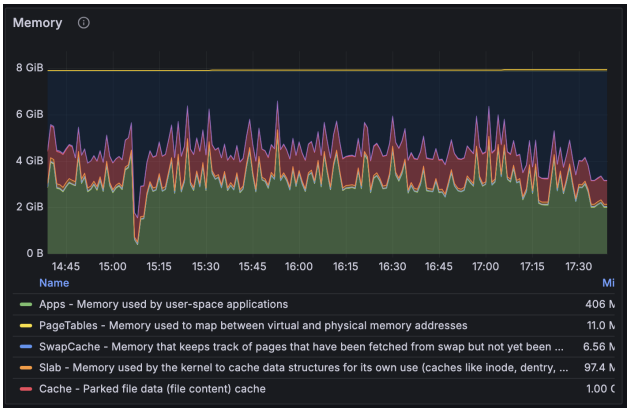
Na Figura 39 observa-se que, a partir das 15:15, quando se inicia o teste com 100 usuários, a utilização da CPU permanece constantemente acima de 80%, sendo a maior parte em modo *user* (processos em espaço de usuário). Isso indica que a carga de trabalho concentra-se no modelo executado pelos *workers*. Após 17:15, quando o teste de 100 usuários é finalizado, há quedas bruscas no uso. Durante o teste com 10 usuários, a variação é mínima; no cenário de 1 usuário há apenas um pequeno pico próximo de 50%, mantendo-se significativamente abaixo dos demais casos.

Figura 39 – Gráfico CPU

Fonte: Autor (2025)

Na Figura 40 nota-se que, durante o teste com 100 usuários, há aumento considerável do uso de memória (entre 4 e 6 GiB), refletindo o processamento intenso das aplicações. Com a redução para 10 usuários, a variação não é significativa; já no teste com 1 usuário, a memória mantém-se abaixo de 4 GiB. Não foram observados indícios relevantes de *swap*, demonstrando que a memória disponível foi suficiente.

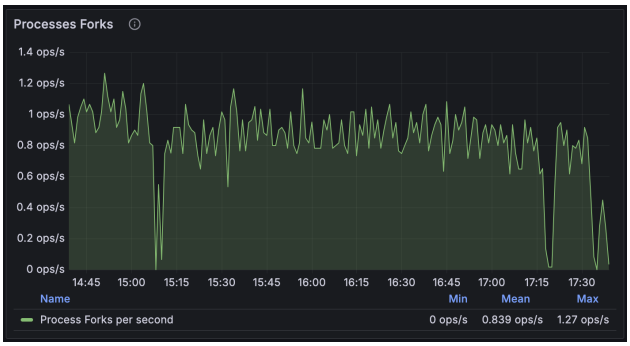
Figura 40 – Gráfico Memória



Fonte: Autor (2025)

Na Figura 41 observa-se que, durante o teste de 100 usuários, a taxa média de criação de processos é de aproximadamente 0,8 *forks*/s. Após 17:15, no teste com 10 usuários, o valor médio mantém-se semelhante, porém sem picos acima de 1 *ops*/s. No teste com 1 usuário, praticamente não há atividade perceptível.

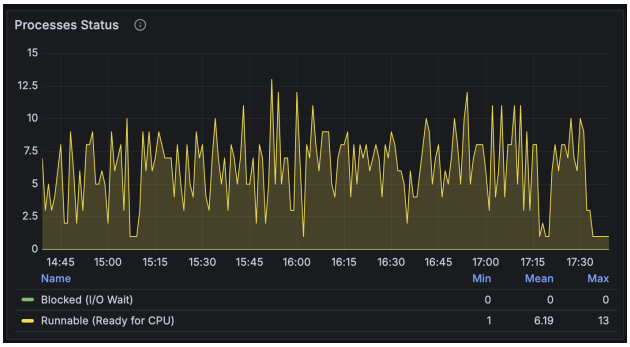
Figura 41 – Gráfico *Processes Forks*



Fonte: Autor (2025)

A Figura 42 demonstra que, durante o teste de 100 usuários, há em média seis processos simultaneamente prontos para execução (*runnable*). Após o término do teste, o valor mantém-se no cenário de 10 usuários, enquanto no de 1 usuário praticamente zera. Não há processos bloqueados por I/O, indicando que o gargalo localiza-se no processamento e não em operações de disco.

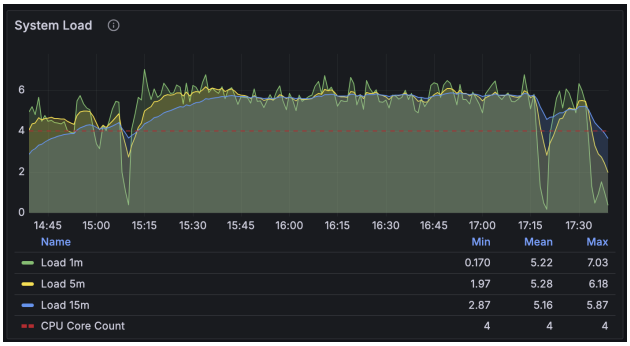
Figura 42 – Gráfico *Processes Status*



Fonte: Autor (2025)

Na Figura 43 nota-se que, no teste com 100 usuários, o *load average* permanece constantemente acima de 4 — linha que representa o número de núcleos do processador. Esse comportamento indica sobrecarga, pois há mais processos demandando CPU do que núcleos disponíveis. Após 17:15, com 10 usuários, o valor aproxima-se de 4; no cenário de 1 usuário, cai para valores inferiores a 2, indicando folga de recursos.

Figura 43 – Gráfico *System Load*



Fonte: Autor (2025)

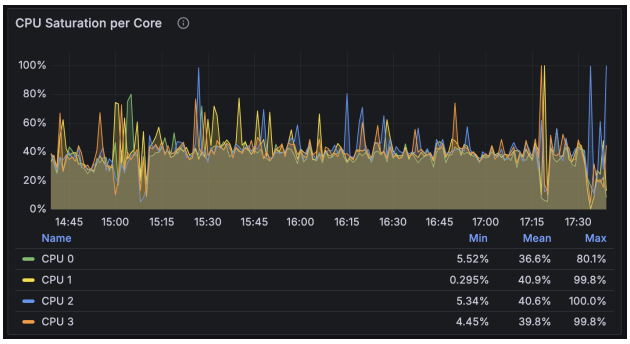
A Figura 44 demonstra que, durante o teste com 100 usuários, o número de *context switches* e *interrupts* mantém-se elevado (cerca de 1,8k *ops*/s). Um pico é observado pouco antes das 15:15, possivelmente decorrente da finalização abrupta de testes anteriores. Com 10 usuários, os valores permanecem próximos da média; com 1 usuário, são inferiores, confirmando que a alta concorrência aumenta a troca de contexto e as interrupções de hardware.

Figura 44 – Gráfico *Context Switches / Interrupts*

Fonte: Autor (2025)

Na Figura 45 observa-se que, durante o teste de 100 usuários, os quatro núcleos apresentam uso equilibrado e picos próximos de 100%, evidenciando boa distribuição de carga. No teste com 10 usuários, os picos diminuem, embora ainda haja flutuações relevantes. Já no cenário de 1 usuário, a utilização média cai para cerca de 50%.

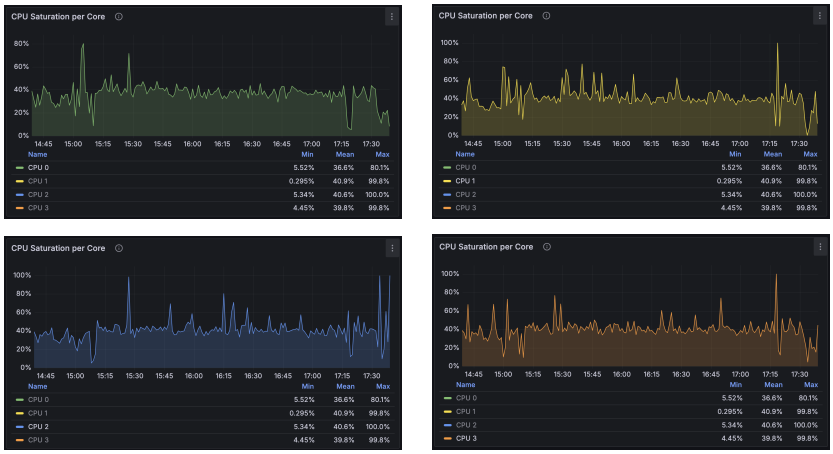
Figura 45 – Gráfico de saturação dos módulos da CPU sobrepostos



Fonte: Autor (2025)

Na Figura 46 verifica-se o comportamento individual de cada núcleo de CPU. O padrão acompanha o da Figura 45, com alta saturação no teste de 100 usuários e queda após o término.

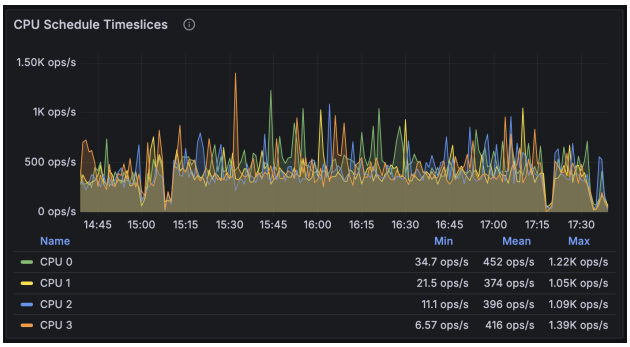
Figura 46 – Gráficos de saturação dos módulos da CPU



Fonte: Autor (2025)

Na Figura 47 verifica-se que, durante o teste de 100 usuários, o agendador do sistema distribui grande quantidade de *timeslices* (até 1,5k *ops*/s) entre os quatro núcleos. Com 10 usuários, esse valor diminui consideravelmente; com 1 usuário, quase não há atividade perceptível, evidenciando a relação direta entre o número de usuários concorrentes e a intensidade do escalonamento.

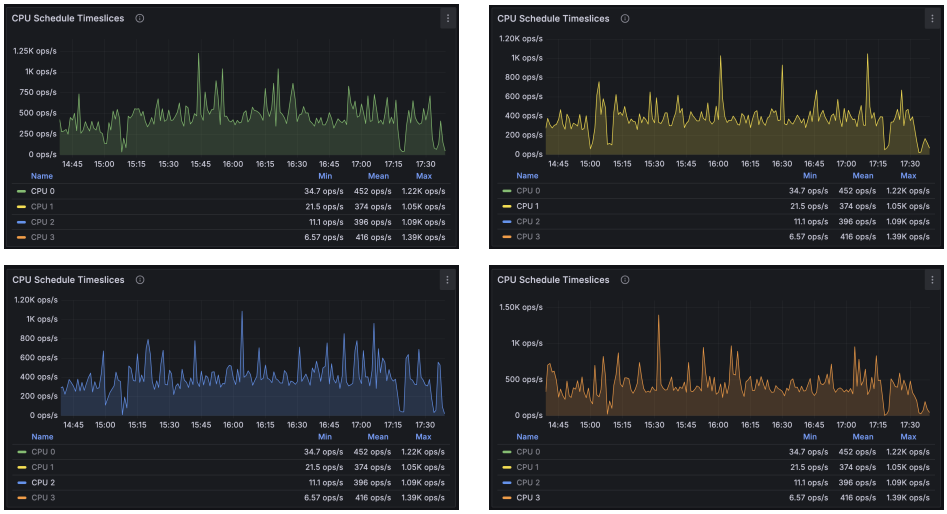
Figura 47 – Gráfico de *schedule* dos módulos da CPU sobrepostos



Fonte: Autor (2025)

A Figura 48 apresenta a análise individual do *schedule* de cada núcleo de CPU. O comportamento permanece consistente com o observado na Figura 47, com alta atividade no teste de 100 usuários e redução progressiva nos demais cenários.

Figura 48 – Gráficos de *schedule* dos módulos da CPU



Fonte: Autor (2025)

Estes gráficos ilustram os resultados mencionados na seção anterior, A Raspberry Pi 5 demonstrou capacidade para operar de forma estável com até dois workers executando o Mask R-CNN. Acima desse ponto, o sistema enfrenta limites práticos relacionados ao consumo de memória e ao custo computacional do modelo. Além de demostrar a eficiência da combinação entre Prometheus e Grafana para monitoramento em tempo real de sistemas embarcados.

4.3 Testes Funcionais

Realizaram-se testes funcionais abrangendo os fluxos de cadastro e autenticação. O processo de cadastro foi testado com diferentes cenários de entrada, incluindo:

- Inserção de e-mails com formatos inválidos (sem @, domínios incorretos);
- senhas que não atendem aos critérios mínimos de segurança (menos de 8 caracteres, ausência de caracteres especiais, números ou letras maiúsculas);
- tentativas de cadastro com e-mails já existentes no banco de dados;
- confirmação por e-mail e ativação da conta;
- login com dados válidos e inválidos;
- recuperação de senha, envolvendo envio e validação do link de redefinição;
- redirecionamento pós-login, tanto para a finalização do cadastro no primeiro acesso quanto para a tela Home em acessos posteriores;

- comportamento reativo da Home, que exibe apenas os módulos autorizados ao perfil do usuário, adaptando-se dinamicamente em caso de mudança de perfil.

Conduziram-se testes específicos para validar a geração de relatórios tanto para o perfil de Produtor quanto para o de Supervisor. No caso do Produtor, testou-se o fluxo de geração de relatório de produção da horta para assegurar que os dados agregados refletissem corretamente as análises realizadas, apresentando informações como quantidade de frutos por estágio de maturação e evolução ao longo do tempo. Para o Supervisor, verificou-se o fluxo de geração de relatórios de produção das hortas dos produtores vinculados, garantindo que os dados fossem corretamente agregados e apresentados de forma clara e organizada. Ambos os fluxos foram testados com o conjunto completo e com filtros de localizações específicas, assegurando a precisão e a integridade das informações exibidas.

Testaram-se os fluxos específicos do perfil de Produtor com foco na gestão de hortas, plantas e análises. A funcionalidade “Minhas Hortas” foi validada para garantir que apenas hortas pertencentes ao usuário logado fossem exibidas. Também se testou o fluxo de criação de novas hortas, incluindo a seleção de localização via Google Maps, a definição do nome e a associação correta com o usuário no banco de dados.

O mesmo critério aplicou-se ao fluxo de gerenciamento de plantas, verificando-se a exibição correta das plantas vinculadas a uma horta específica, bem como a criação de novas plantas e sua associação correspondente.

Testaram-se os fluxos específicos do perfil de Supervisor para assegurar a correta visualização e gerenciamento das hortas e plantas dos produtores vinculados. A funcionalidade “Hortas dos Meus Produtores” foi validada para garantir que apenas hortas associadas aos produtores vinculados fossem exibidas. Também se testou o fluxo de geração de relatórios de produção das hortas, verificando a correta agregação dos dados e a apresentação adequada dos resultados.

Verificou-se igualmente o fluxo de visualização das plantas dos produtores vinculados, assegurando que o supervisor pudesse acessar as informações relevantes de forma clara e organizada. Também foi testado o fluxo de vinculação de produtores ao supervisor, garantindo que a associação fosse realizada corretamente no banco de dados e refletida na interface do usuário.

Testou-se integralmente o fluxo de registro de novas análises, abrangendo desde a tela inicial de “Registrar Imagens” até a finalização da submissão e a visualização dos resultados. Todas as etapas foram verificadas para assegurar a integridade do fluxo

assíncrono e o correto funcionamento da integração com os serviços em nuvem. O processo incluiu a seleção da horta e da planta correspondente, a escolha entre captura de imagens pela câmera ou seleção a partir da galeria, a exibição em miniatura das imagens escolhidas e, por fim, o envio dos arquivos ao Amazon S3.

Realizaram-se testes de integração com o objetivo de validar não apenas a comunicação entre os diferentes módulos do sistema, mas também os mecanismos de segurança associados a essa comunicação. Esses testes asseguraram a consistência dos dados e a fluidez da navegação entre os diversos estados da aplicação, ao mesmo tempo em que verificaram o correto controle de acesso aos recursos disponibilizados.

O processo envolveu a verificação da comunicação entre o *front-end* e as APIs de autenticação, consulta e criação de entidades, incluindo a validação da obrigatoriedade do uso de *tokens* de autenticação nos *requests*. Foram avaliados cenários de tentativa de acesso às APIs sem credenciais válidas, bem como o comportamento do sistema diante de requisições não autorizadas, garantindo a adequada negação de serviço em situações de uso indevido.

Adicionalmente, os testes contemplaram a integração com os serviços da AWS e a sincronização entre os dados persistidos e as interfaces que utilizam assinaturas reativas, assegurando que apenas requisições autenticadas e autorizadas fossem capazes de refletir alterações em tempo real no estado da aplicação.

Conduziram-se testes de interface e experiência do usuário (UX) com o objetivo de avaliar o quão intuitiva se apresenta a aplicação para o usuário final. Nessa etapa, analisaram-se a organização e a disposição dos elementos em tela, a facilidade de compreensão das funcionalidades disponíveis, a clareza das mensagens de erro e de sucesso, bem como a presença de indicadores visuais de carregamento durante operações assíncronas. Adicionalmente, avaliou-se a fluidez da navegação e das transições entre telas, buscando verificar se o usuário consegue executar as principais ações do sistema de forma natural, sem a necessidade de treinamento prévio.

Com base nos testes executados, conclui-se que o artefato encontra-se funcionalmente estável, cobrindo com segurança os fluxos principais definidos na proposta. A abordagem de testes foi fundamental para garantir a confiabilidade do sistema e fornecer uma base para etapas futuras de validação com usuários reais.

4.4 Teste Ponta a ponta

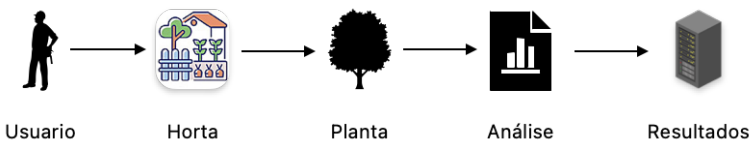
Uma vez estabelecida a eficiência e a eficácia do servidor e do modelo, iniciam-se os testes ponta a ponta, cujo objetivo é avaliar não apenas a eficácia do aplicativo, mas também a usabilidade para o usuário. Os testes foram realizados em um iPhone 15 rodando iOS 18.6.2, conectado ao MacBook mencionado anteriormente para emular a aplicação. Para este fim os usuarios devem completar todos os fluxos disponíveis no aplicativo, desde o cadastro até a visualização dos resultados da análise.

Antes de iniciar os testes, efetuam-se os ajustes necessários no programa. O aplicativo passa a chamar o servidor, e não mais a API criada no SageMaker. Em vez de realizar a chamada ao *endpoint* do SageMaker, solicita-se um token ao AWS Cognito e, em seguida, realiza-se uma chamada direta ao servidor, enviando as imagens com o token no *header*. Os demais fluxos permanecem inalterados. Com os ajustes aplicados, repetem-se todos os testes descritos anteriormente na seção destinada aos fluxos.

Com a execução dos testes, confirma-se a eficiência do programa, embora tenham sido identificadas possíveis melhorias relacionadas à usabilidade, além da necessidade de aperfeiçoar a rotina de eliminação de dados. A seguir, apresentam-se as melhorias implementadas por ordem de relevância:

1. Ajustes na rotina de limpeza (Figura 49).
    - Ao eliminar a análise, eliminam-se todos os arquivos vinculados a ela.
    - Ao eliminar uma planta, eliminam-se todas as análises vinculadas.
    - Ao eliminar uma horta, eliminam-se todas as plantas associadas.

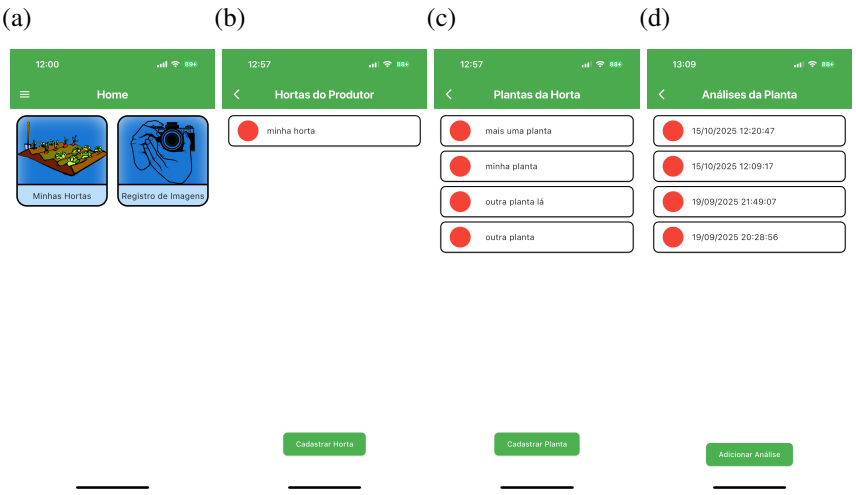
Figura 49 – Fluxo de eliminação de dados



Fonte: Autor (2025)

2. Melhoria no visual das páginas de *home*, hortas, plantas, análises e visualização da análise (Figura 50).

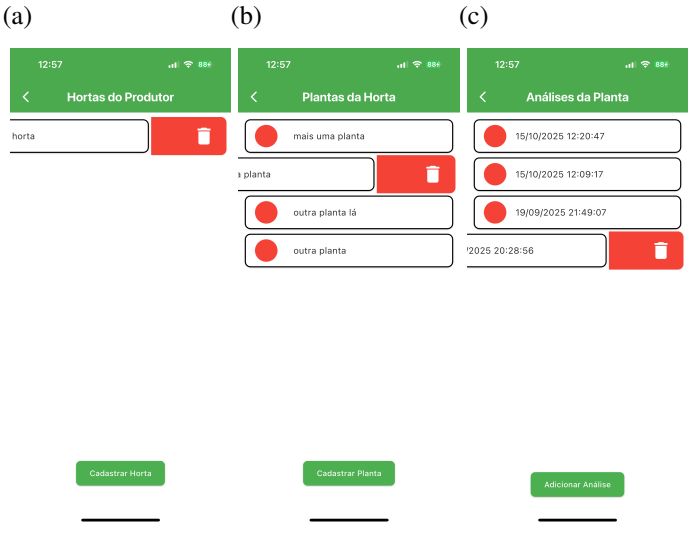
Figura 50 – Telas após melhorias de UX



Fonte: Autor (2025)

3. Melhoria no processo de eliminação de hortas, plantas e análises (Figura 51).

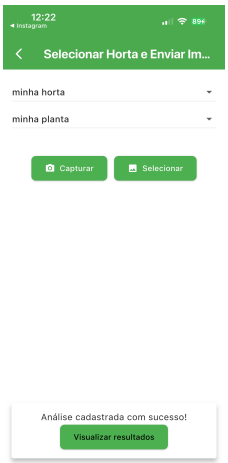
Figura 51 – Processo de eliminação após melhoria



Fonte: Autor (2025)

4. Inclusão de opção para visualizar os resultados da análise diretamente da página de registro de imagens (Figura 52).

Figura 52 – Botão de visualização da análise

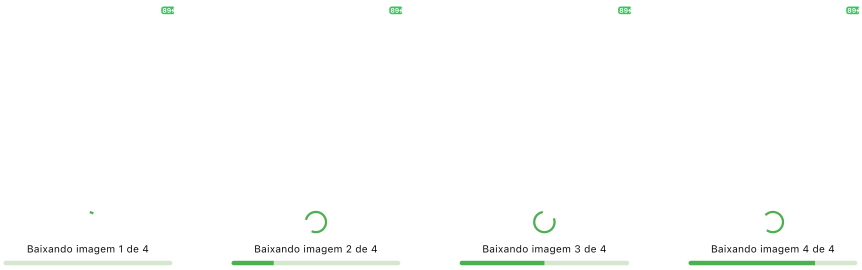


Fonte: Autor (2025)

5. Inclusão de caminho para a página de registro de imagens a partir da página de análises.
6. Adição de barra de progresso na página de registro de imagem ao iniciar a análise e na página de visualização ao iniciar o download das imagens (Figura 53).

Figura 53 – Imagens da tela de carregamento após melhorias

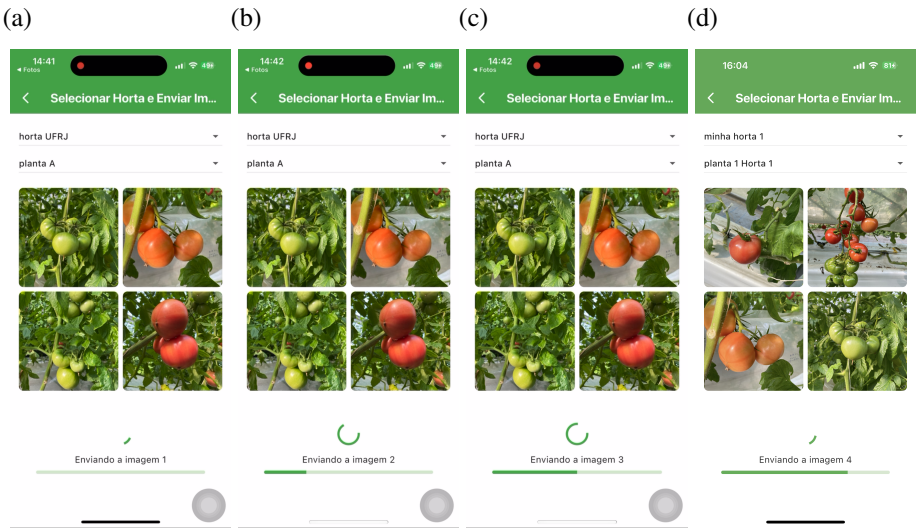
(a) (b) (c) (d)



Fonte: Autor (2025)

7. Adição de círculo de carregamento na página de registro de imagem (Figura 54).

Figura 54 – Imagens da tela de registro de imagem após melhorias



Fonte: Autor (2025)

4.5 Aplicação do TAM

O processo de aceitação de novas tecnologias constitui um fator determinante para o sucesso de sistemas voltados à inovação, especialmente em contextos que envolvem públicos diversos e com distintos níveis de familiaridade tecnológica. No âmbito da agricultura inteligente, compreender como os usuários percebem, avaliam e adotam uma solução digital é essencial para sua incorporação efetiva às práticas cotidianas do campo.

Dessa forma, este capítulo apresenta a aplicação do Modelo de Aceitação de Tecnologia (TAM – *Technology Acceptance Model*) ao sistema "Minha Horta, Minha Vida", uma solução desenvolvida com base na Internet Artificial das Coisas (*IAoT*) e em uma arquitetura Cloud–Fog–Edge, destinada à detecção e classificação de estágios de maturação de tomates por meio de visão computacional. O modelo TAM fornece uma estrutura teórica consolidada para avaliar a aceitação e a intenção de uso de tecnologias emergentes, sendo amplamente utilizado em pesquisas de adoção tecnológica em ambientes acadêmicos e corporativos.

4.5.1 Contextualização do sistema avaliado

O sistema “Minha Horta, Minha Vida” foi concebido com o propósito de oferecer uma solução escalável e acessível para produtores urbanos e rurais que buscam otimizar o cultivo de hortaliças, com foco inicial em plantas de tomate. A aplicação utiliza um modelo de visão computacional baseado em *deep learning* para realizar inferências automáticas sobre o estágio de maturação dos frutos.

A arquitetura proposta segue o paradigma Cloud–Fog–Edge, permitindo a coleta de dados em tempo real por meio de dispositivos móveis, o pré-processamento local em um nó Fog (Raspberry Pi) e o armazenamento e treinamento em nuvem, via AWS SageMaker. Essa estrutura reduz a latência, aumenta a resiliência e garante a escalabilidade do sistema, favorecendo seu uso em diferentes cenários produtivos.

Nesse contexto, torna-se relevante compreender como os potenciais usuários percebem a utilidade, a facilidade de uso e a intenção de adoção dessa tecnologia em suas rotinas de cultivo.

4.5.2 Estrutura conceitual e hipóteses de pesquisa

Para o projeto “Minha Horta, Minha Vida”, o modelo TAM foi adaptado de modo a refletir o contexto agrícola e tecnológico da solução proposta.

Com base nessa estrutura, definem-se as seguintes hipóteses de pesquisa:

- H1: A facilidade percebida de uso influencia a utilidade percebida.
- H2: A facilidade percebida de uso influencia a atitude em relação ao uso.
- H3: A utilidade percebida influencia a atitude em relação ao uso.

4.5.3 Instrumento de coleta e amostragem

A coleta de dados será realizada por meio de um questionário estruturado, composto por 20 questões em escala de aceitação de 5 pontos (1 – discordo totalmente; 5 – concordo totalmente), aplicadas a dois grupos principais:

- Grupo 1: Autor.
- Grupo 2: Irmã do autor.

O questionário tem por objetivo quantificar as percepções relacionadas às variáveis do modelo, permitindo a análise qualitativa dos fatores determinantes da aceitação tecnológica.

4.5.4 Procedimentos de análise

Considerando que a amostra é composta por apenas dois participantes — o autor e sua irmã —, opta-se por uma análise descritiva e interpretativa das respostas obtidas no questionário. Dessa forma, não serão aplicadas técnicas estatísticas complexas, pois o tamanho amostral não permite inferência estatística confiável.

As respostas em escala de aceitação (1 a 5) serão apresentadas em formato tabular, indicando o grau de concordância de cada participante em relação a cada variável do modelo: facilidade percebida de uso, utilidade percebida, atitude em relação ao uso e intenção de uso. A interpretação será conduzida de forma qualitativa, destacando percepções convergentes e divergentes entre os respondentes, bem como *insights* sobre a experiência de uso do sistema.

Essa abordagem permite avaliar, de maneira exploratória, o potencial de aceitação da aplicação, identificando pontos fortes e oportunidades de melhoria para versões futuras do “Minha Horta, Minha Vida”.

4.5.5 Resultados obtidos

Esta seção apresenta os resultados obtidos da aplicação do Modelo de Aceitação de Tecnologia (TAM) ao sistema “Minha Horta, Minha Vida”. O objetivo é avaliar a percepção dos usuários quanto à facilidade de uso, utilidade percebida, atitude em relação à tecnologia e intenção de uso futuro do aplicativo, permitindo identificar fatores que influenciam sua aceitação e possíveis melhorias na experiência de utilização.

Os resultados estão organizados nas tabelas a seguir, distribuídos conforme as quatro dimensões principais do modelo TAM: Facilidade Percebida de Uso (FPU), Utilidade Percebida (UP), Atitude em Relação ao Uso (ATU) e Intenção de Uso (IU). Após cada tabela, apresenta-se uma análise qualitativa dos dados, buscando compreender como cada dimensão contribui para a aceitação e a adoção potencial do sistema “Minha Horta, Minha Vida”.

Tabela 16 – Facilidade Percebida de Uso (FPU)

Item	Avaliação	Grupo 1	Grupo 2
FPU1	“Minha Horta, Minha Vida” é fácil de usar.	4	3
FPU2	As funções principais são fáceis de entender.	4	3
FPU3	A navegação entre as telas é simples e intuitiva.	5	5
FPU4	O processo de cadastro e login é direto e rápido.	5	5
FPU5	Eu me sinto confiante em utilizar o aplicativo sem ajuda de outras pessoas.	5	3

Fonte: Autor (2025)

A análise dos resultados referentes à funcionalidade e à experiência de uso do aplicativo (Tabela 16) evidencia percepções convergentes entre os avaliadores, ainda que com nuances distintas quanto ao nível de complexidade percebida. Em relação à facilidade de uso (FPU1), ambos os grupos reconheceram a interface como organizada e intuitiva, ainda que apontem a necessidade de explicações iniciais para orientar o uso das funcionalidades. Essa observação também se manifesta em FPU2, indicando que as funções principais são compreensíveis, mas poderiam contar com suporte explicativo mais detalhado, como tutoriais ou mensagens interativas. A navegação (FPU3) recebeu avaliações máximas, sendo descrita como fluida e acessível mesmo para usuários com menor familiaridade tecnológica. O processo de cadastro e login (FPU4) também foi bem avaliado, destacando sua eficiência e segurança. Entretanto, em relação à confiança para uso autônomo (FPU5), observou-se divergência: o grupo 1 declarou plena segurança, enquanto o grupo 2 demonstrou necessidade de assistência inicial, reforçando a relevância de recursos didáticos.

Tabela 17 – Utilidade Percebida (UP)

Item	Avaliação	Grupo 1	Grupo 2
UP1	Melhora o acompanhamento do cultivo.	3	3
UP2	As informações apresentadas são relevantes.	4	4
UP3	O sistema ajuda a economizar tempo.	3	4
UP4	O uso do aplicativo pode aumentar a produtividade.	1	1
UP5	Oferece suporte útil para tomada de decisão.	3	3

Fonte: Autor (2025)

No que se refere à utilidade prática (UP1–UP5) (Tabela 17), ambos os grupos apresentaram posicionamentos similares. As avaliações indicaram que o aplicativo auxilia parcialmente no acompanhamento do cultivo (UP1), apontando como limitação a ausência de recursos avançados, como previsão de produtividade e detecção de pragas. As informações apresentadas (UP2) foram consideradas claras e tecnicamente relevantes, embora ainda insuficientes para orientar o manejo de forma personalizada. Quanto à economia de tempo (UP3), as percepções variaram entre avaliações intermediárias, sugerindo que o sistema otimiza algumas etapas, mas ainda depende de registros manuais. O impacto na produtividade (UP4) foi avaliado como baixo, uma vez que o aplicativo atua predominantemente na detecção, e não no apoio à tomada de decisão. Por fim, o suporte ao plantio (UP5) apresentou potencial reconhecido pelos participantes, ainda que limitado às funcionalidades atuais.

Tabela 18 – Atitude em Relação ao Uso (ATU)

Item	**Avaliação**	**Grupo 1**	**Grupo 2**
ATU1	Acho interessante utilizar tecnologias como esta.	5	3
ATU2	Utilizar o aplicativo é uma boa ideia.	5	3
ATU3	O uso do sistema é agradável e satisfatório.	5	4
ATU4	Acredito que vale a pena usar a aplicação no dia a dia.	3	3
ATU5	O sistema transmite confiança durante o uso.	5	4

Fonte: Autor (2025)

Em relação à atitude frente ao uso (ATU1–ATU5) (Tabela 18), as percepções revelaram entusiasmo moderado. O grupo 1 atribuiu notas elevadas, destacando o valor da tecnologia na agricultura urbana e elogiando aspectos como interface e confiabilidade. Já o grupo 2 apresentou avaliações intermediárias, reconhecendo o interesse e a funcionalidade do aplicativo, mas apontando menor relevância para usuários não envolvidos diretamente com o cultivo. Esse contraste sugere que o engajamento está diretamente relacionado às necessidades do usuário final.

Tabela 19 – Intenção de Uso (IU)

Item	Avaliação	Grupo 1	Grupo 2
IU1	Pretendo continuar utilizando o sistema no futuro.	5	1
IU2	Recomendaria o aplicativo para outras pessoas.	3	3
IU3	Se disponível comercialmente, eu usaria este sistema.	3	1
IU4	Eu me sentiria confortável em depender do sistema para o manejo da horta.	4	3
IU5	Tenho interesse em testar novas versões do aplicativo.	5	1

Fonte: Autor (2025)

Os indicadores de intenção de uso (IU1–IU5) (Tabela 19) revelaram padrão semelhante. O grupo 1 demonstrou interesse em continuar utilizando o aplicativo e em testar futuras versões, evidenciando confiança e expectativa de evolução tecnológica. O grupo 2 mostrou menor engajamento, atribuindo tal posição à baixa aplicabilidade em contextos não produtivos. Ambos os grupos, contudo, reconheceram o potencial do sistema, desde que incorpore funcionalidades adicionais, como previsões de rendimento e recomendações de manejo.

5 CONSIDERAÇÕES FINAIS

O objetivo central deste projeto consistiu em analisar a aplicabilidade de Tecnologias da Informação e Comunicação, alinhadas aos princípios da Agricultura 5.0, para apoiar a gestão de hortas urbanas. Esse objetivo desdobrou-se em etapas específicas voltadas à avaliação da viabilidade do uso de Visão Computacional para detecção e contagem de frutos, ao desenvolvimento de um artefato funcional com processamento em nuvem, à posterior adaptação da solução para uma arquitetura baseada em Computação em Névoa e ao aprimoramento contínuo do aplicativo e do modelo de detecção. Considerando o percurso metodológico adotado e os resultados apresentados nos capítulos anteriores, conclui-se que os objetivos foram alcançados de forma plena.

A adoção da Design Science Research (DSR), configurou o desafio de interpretar, à luz do problema real, as possibilidades e limitações das tecnologias disponíveis, bem como selecionar, combinar e ajustar métodos de forma consciente e fundamentada. Ainda assim, o rigor na organização das etapas, a documentação sistemática das decisões de projeto e a validação contínua dos resultados permitiram superar tais dificuldades, garantindo a coerência entre os objetivos estabelecidos e a evolução do artefato ao longo do processo de desenvolvimento. Essa abordagem consolidou a DSR como eixo estruturante do trabalho, assegurando que a solução final fosse não apenas funcional, mas avaliada em seus méritos, limites e contribuições acadêmicas.

Os experimentos realizados com o conjunto de dados COCO e com imagens coletadas em campo demonstraram que o modelo Mask R-CNN apresentou desempenho satisfatório na detecção e classificação de frutos, mesmo sob condições variáveis de iluminação e complexidade típicas do ambiente agrícola urbano. Embora frutos muito pequenos não tenham sido detectados nas fases iniciais de desenvolvimento, essa limitação não comprometeu o processo de contagem ao longo do ciclo produtivo, uma vez que tais frutos se tornam reconhecíveis conforme amadurecem. Assim, o sistema comprovou sua eficácia para o monitoramento contínuo das culturas analisadas.

Do ponto de vista funcional, o aplicativo desenvolvido permite não apenas o registro de imagens, mas também o acompanhamento evolutivo das plantas, possibilitando análises comparativas ao longo do tempo. Entre suas potencialidades, destacam-se a avaliação de diferentes insumos, a observação de variações produtivas entre estações do ano e a identificação de períodos críticos de baixa produtividade — aspectos especialmente relevantes para culturas sensíveis, como o tomate, fortemente influenciadas

pelas condições climáticas e de luminosidade.

A aplicação do Modelo de Aceitação de Tecnologia (TAM) permitiu compreender como fatores como utilidade percebida, facilidade de uso e atitude influenciam a aceitação do sistema. De maneira geral, observou-se boa receptividade entre os usuários, que destacaram o design intuitivo e a facilidade de navegação. Entretanto, foram identificadas demandas por recursos explicativos adicionais e funcionalidades técnicas mais avançadas, evidenciando a necessidade de aprimoramentos voltados à ampliação do valor prático do sistema e à personalização da experiência de uso.

Sob a perspectiva técnica, a arquitetura Cloud–Fog proposta demonstrou-se viável e adequada para o processamento distribuído e para a escalabilidade requerida pela solução. A estrutura modular permite a inserção de novos serviços de detecção, análise ou predição sem comprometer os componentes existentes, viabilizando a expansão da plataforma conforme novas demandas surgirem. Os testes de desempenho indicaram que a solução atual opera próxima ao limite de sua capacidade quando executada com dois workers. Contudo, a própria modularidade da arquitetura facilita a expansão horizontal por meio da adição de novos servidores, assegurando estabilidade e paralelismo entre módulos especializados.

À luz desses resultados, torna-se evidente que o sistema Minha Horta, Minha Vida não apenas cumpre a função de monitoramento, mas também estabelece as bases para uma agricultura urbana mais informada, sustentável e orientada por evidências. A proposta integra coleta de dados, análise de imagens e acompanhamento longitudinal, oferecendo ao produtor subsídios concretos para decisões de manejo fundamentadas na evolução real de suas culturas.

Apesar dos avanços alcançados, o presente trabalho abre espaço para diversas oportunidades de aprimoramento e expansão da solução. Entre as principais recomendações, destacam-se:

- Quantificação de frutos em massa: ampliação do modelo de detecção para estimar a produção em quilogramas, incorporando parâmetros de calibração referentes ao tamanho e à densidade dos frutos.
- Detecção de pragas: desenvolvimento de um módulo específico para identificar doenças e pragas foliares por meio de técnicas de segmentação e classificação baseadas em aprendizado profundo.
- Modelos preditivos de produção: inclusão de algoritmos capazes de estimar datas e quantidades prováveis de colheita para cada cultura monitorada, com base em séries

temporais e aprendizado supervisionado.

- Calendário padronizado de coleta: implementação de um protocolo sistemático de captura de imagens, com duração mínima de seis meses, para garantir a consistência dos dados e possibilitar análises longitudinais mais robustas.
- Expansão de infraestrutura: ampliação do número de workers e implantação de servidores adicionais para suportar o crescimento do sistema sem comprometer sua responsividade.
- Diversificação de culturas: extensão do modelo para outras espécies hortícolas além do tomate, a fim de avaliar a adaptabilidade da solução em diferentes contextos produtivos.

A execução dessas etapas requer tempo, infraestrutura e o engajamento de produtores dispostos a seguir um cronograma regular de coleta e registro. Um ciclo completo de acompanhamento experimental permitirá consolidar um banco de dados longitudinal, fundamental para validar modelos preditivos, aprimorar a precisão da solução e fortalecer sua aplicabilidade em cenários reais de agricultura urbana.

REFERÊNCIAS

ALEXANDRATOS, N.; BRUINSMA, J. World agriculture towards 2030/2050: the 2012 revision. 2012.

BIRRELL, S. *et al.* A field-tested robotic harvesting system for iceberg lettuce. **Journal of Field Robotics**, v. 37, n. 2, p. 225–245, 2020. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21888.

BITTENCOURT, L. *et al.* The internet of things, fog and cloud continuum: Integration and challenges. **Internet of Things**, Elsevier, v. 3, p. 134–155, 2018.

CHEN, K. *et al.* Mmdetection: Open mmlab detection toolbox and benchmark. **arXiv preprint arXiv:1906.07155**, 2019.

CSILLIK, O. *et al.* Identification of citrus trees from unmanned aerial vehicle imagery using convolutional neural networks. **Drones**, v. 2, n. 4, 2018. ISSN 2504-446X. Disponível em: https://www.mdpi.com/2504-446X/2/4/39.

CZYMMEK, V. *et al.* Review of energy-efficient embedded system acceleration of convolution neural networks for organic weeding robots. **Agriculture**, v. 13, n. 11, 2023. ISSN 2077-0472. Disponível em: https://www.mdpi.com/2077-0472/13/11/2103.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS Quarterly**, v. 13, n. 3, p. 319–340, 1989.

ELAZHARY, H. Internet of things (iot), mobile cloud, cloudlet, mobile iot, iot cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions. **Journal of Network and Computer Applications**, v. 128, p. 105–140, 2019. ISSN 1084-8045. Disponível em: https://www.sciencedirect.com/science/article/pii/S1084804518303497.

GONÇALVES, A. G.; VENTURA, E. **Cenário brasileiro é marcado pelo contraste entre fome e desperdício**. 2021. Acesso em: 17 de agosto de 2023. Disponível em: https://cfa.org.br/cenario-brasileiro-e-marcado-pelo-contraste-entre-fome-e-desperdicio/.

ISSAC, A. *et al.* Development and deployment of a big data pipeline for field-based high-throughput cotton phenotyping data. **Smart Agricultural Technology**, v. 5, p. 100265, 2023. ISSN 2772-3755. Disponível em: https://www.sciencedirect.com/science/article/pii/S2772375523000953.

KALYANI, Y.; COLLIER, R. A systematic survey on the role of cloud, fog, and edge computing combination in smart agriculture. **Sensors**, v. 21, n. 17, 2021. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/21/17/5922.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. **EBSE Technical Report, Version 2.3**, 2007.

LABORO.AI. **LaboroTomato: usage guide for MMDetection v3**. 2023. GitHub. Acesso em: 15 dez. 2025. Disponível em: https://github.com/laboroai/LaboroTomato/blob/master/usage_guide/usage_guide_mmdetection_v3.md.

LIN, T.-Y. *et al.* Microsoft COCO: Common objects in context. In: **Proceedings of the European Conference on Computer Vision (ECCV)**. Zurich: Springer, 2014. p. 740–755.

LIN, W.; ADETOMI, A.; ARSLAN, T. Low-power ultra-small edge ai accelerators for image recognition with convolution neural networks: Analysis and future directions. **Electronics**, v. 10, n. 17, 2021. ISSN 2079-9292. Disponível em: https://www.mdpi.com/2079-9292/10/17/2048.

MAO, S. *et al.* Automatic cucumber recognition algorithm for harvesting robots in the natural environment using deep learning and multi-feature fusion. **Computers and Electronics in Agriculture**, v. 170, p. 105254, 2020. ISSN 0168-1699. Disponível em: https://www.sciencedirect.com/science/article/pii/S0168169919327048.

MARKOVIć, D. *et al.* Image processing for smart agriculture applications using cloud-fog computing. **Sensors**, v. 24, n. 18, 2024. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/24/18/5965.

MASSRUHÁ, S. M. F. S. *et al.* A transformação digital no campo rumo à agricultura sustentável e inteligente. In: MASSRUHÁ, S. M. F. S. *et al.* (Ed.). **Agricultura digital: pesquisa, desenvolvimento e inovação nas cadeias produtivas**. Brasilia, DF: Embrapa, 2020. cap. 1, p. 20–25. ISBN 978-65-86056-37-2.

MOHER, D. *et al.* Preferred reporting items for systematic reviews and meta-analyses: The prisma statement. **PLOS Medicine**, Public Library of Science, v. 6, n. 7, p. 1–6, 07 2009. Disponível em: https://doi.org/10.1371/journal.pmed.1000097.

OpenCV Team. **OpenCV: Open Source Computer Vision Library**. 2024. <https://opencv.org/>. Accessed: 2025-06-29.

PIMENTEL, M.; FILIPPO, D.; SANTOS, T. M. dos. Design science research: pesquisa científica atrelada ao design de artefatos. **RE@ D-Revista de Educação a Distância e eLearning**, v. 3, n. 1, p. 37–61, 2020.

REN, S. *et al.* Faster r-cnn: Towards real-time object detection with region proposal networks. In: **Advances in neural information processing systems (NeurIPS)**. 2015. v. 28.

RODRIGUES, R. B. **Novas Tecnologias da Informação e da Comunicação**. Recife, PB: IFPE, 2016. 86 p. ISBN 978-85-9450-008-3.

SHI, W. *et al.* Edge computing: Vision and challenges. **IEEE Internet of Things Journal**, v. 3, n. 5, p. 637–646, 2016.

TIAN, H. *et al.* Computer vision technology in agricultural automation—a review. **Information Processing in Agriculture**, Elsevier, v. 7, n. 1, p. 1–19, 2020.

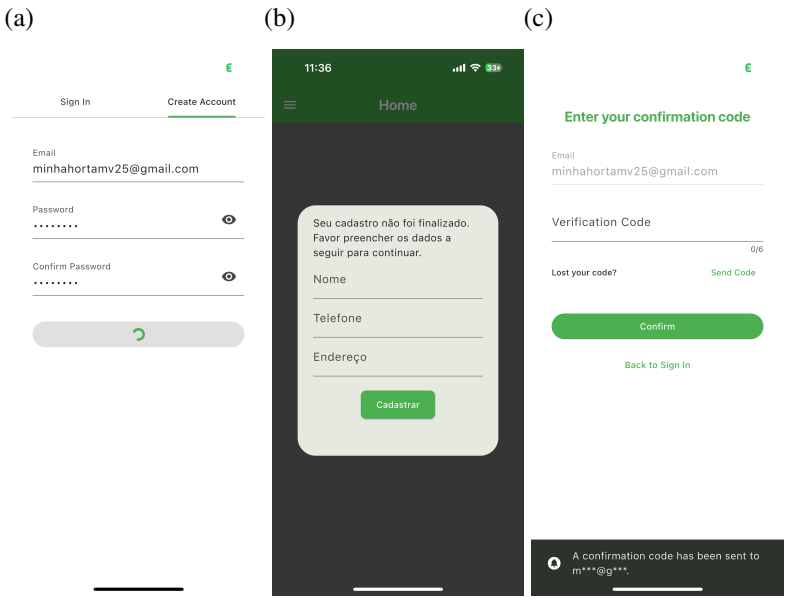
UDEMY, M. R. e equipe. **AWS Certified Machine Learning - Specialty (MLS-C01)**. 2025. <https://www.udemy.com/course/aws-certified-machine-learning-engineer-associate-mla-c01/learn/lecture/45285373>. Aula: SageMaker hosting and endpoints.

YU, Y. *et al.* Fruit detection for strawberry harvesting robot in non-structural environment based on mask-rcnn. **Computers and Electronics in Agriculture**, v. 163, p. 104846, 2019. ISSN 0168-1699. Disponível em: https://www.sciencedirect.com/science/article/pii/S0168169919301103.

APÊNDICE A – TELAS DO APLICATIVO

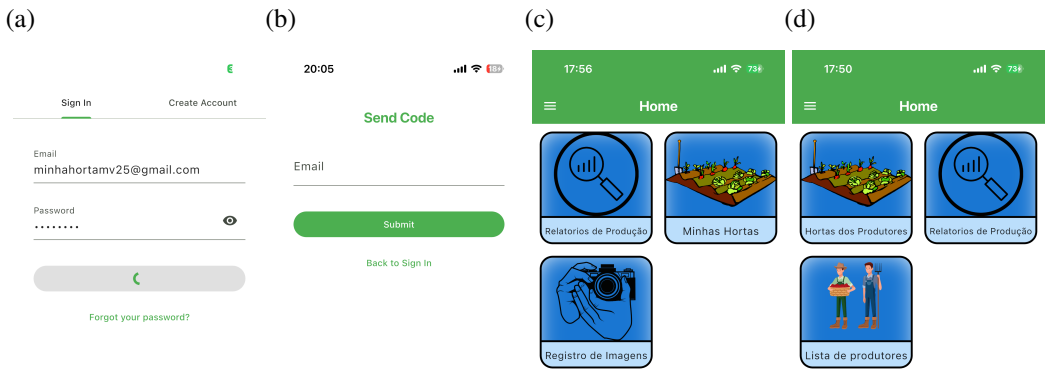
A seguir, são apresentadas as telas dos fluxos do aplicativo desenvolvido neste trabalho.

Figura 55 – Telas do fluxo de cadastro



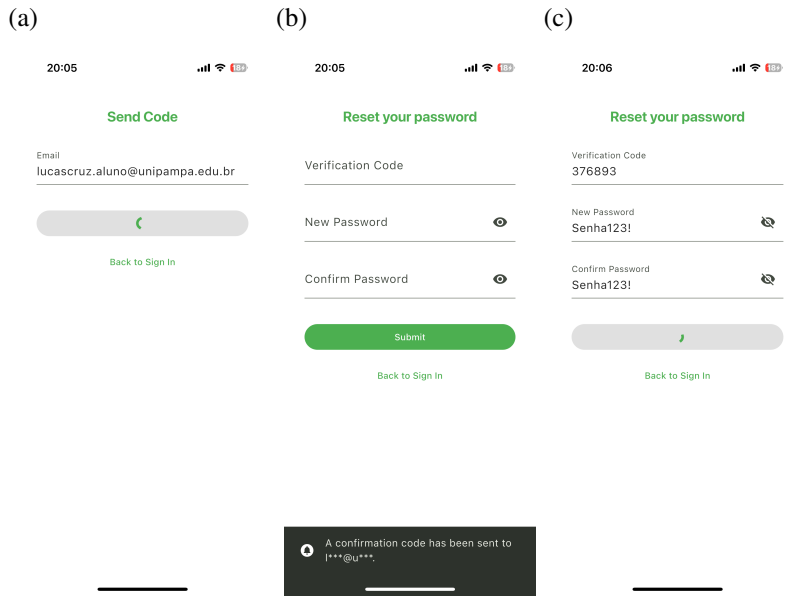
Fonte: Autor (2025)

Figura 56 – Telas do fluxo de login



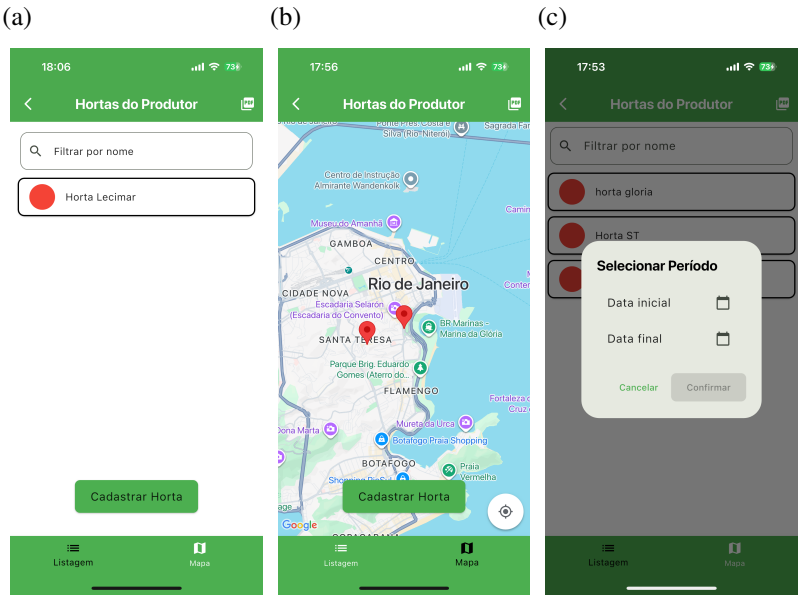
Fonte: Autor (2025)

Figura 57 – Telas do fluxo de redefinir senha



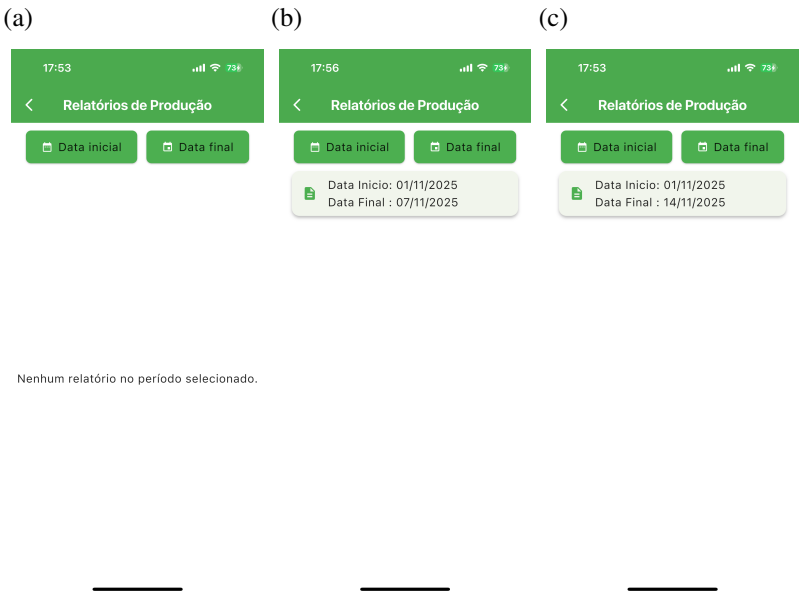
Fonte: Autor (2025)

Figura 58 – Telas do fluxo gerar Relatório Produção



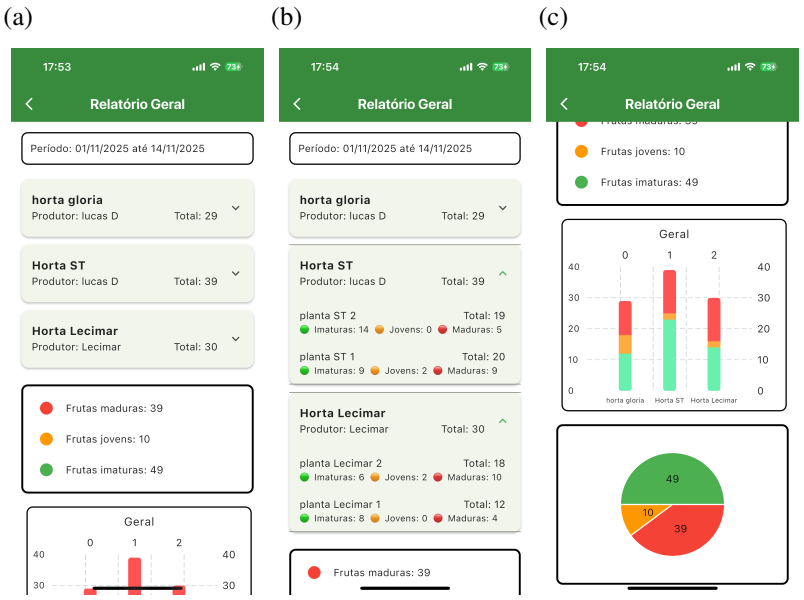
Fonte: Autor (2025)

Figura 59 – Telas Visualizar Relatório de Produção



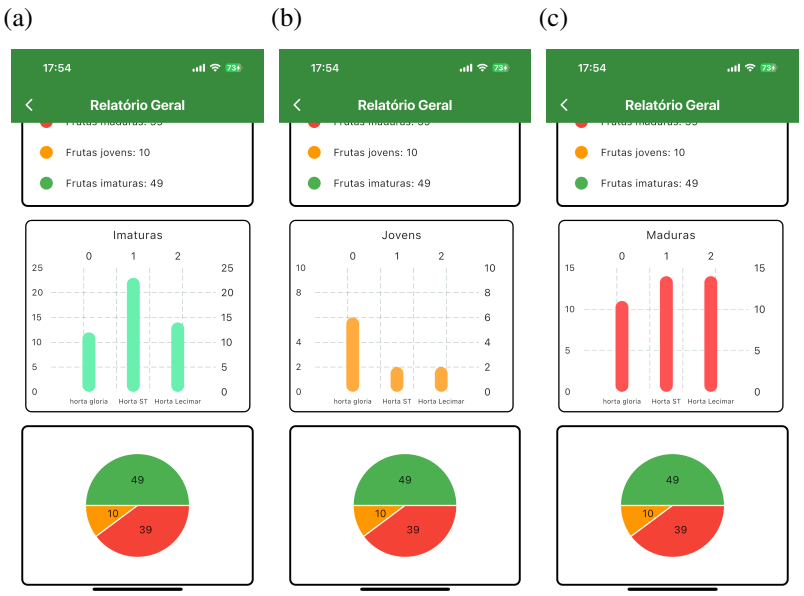
Fonte: Autor (2025)

Figura 60 – Telas do fluxo Visualizar Relatório de Produção parte 1



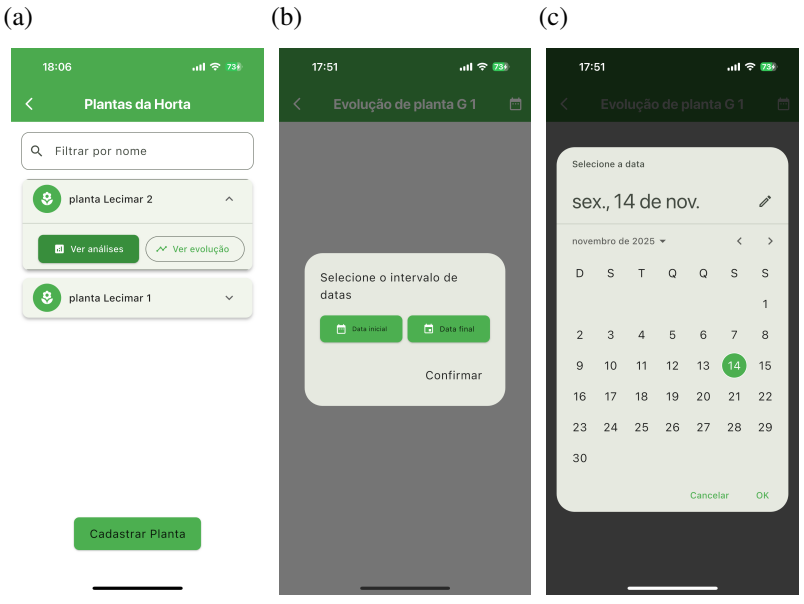
Fonte: Autor (2025)

Figura 61 – Telas do fluxo Visualizar Relatório de Produção parte 2



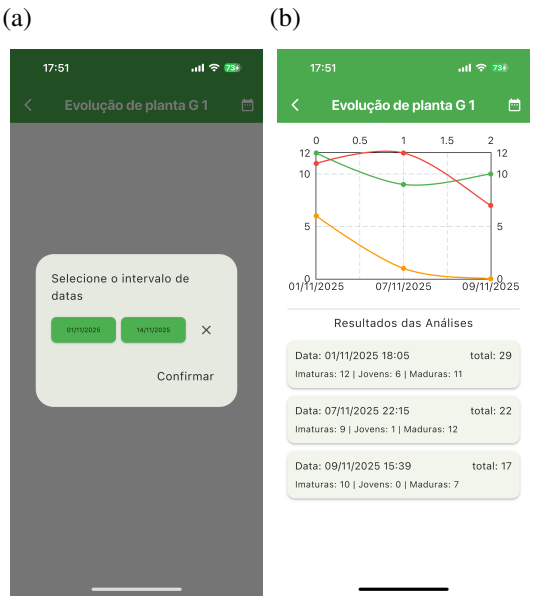
Fonte: Autor (2025)

Figura 62 – Telas do fluxo Evolução da planta parte 1



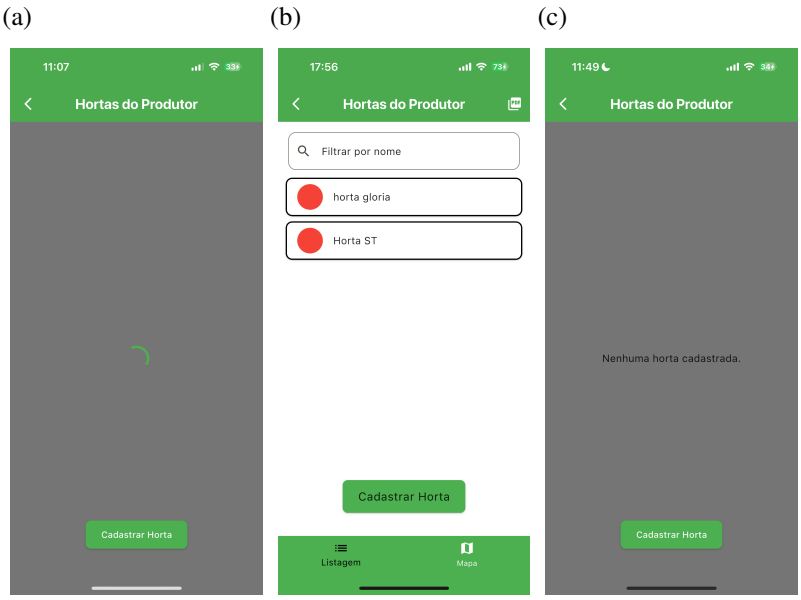
Fonte: Autor (2025)

Figura 63 – Telas do fluxo Evolução da planta parte 2



Fonte: Autor (2025)

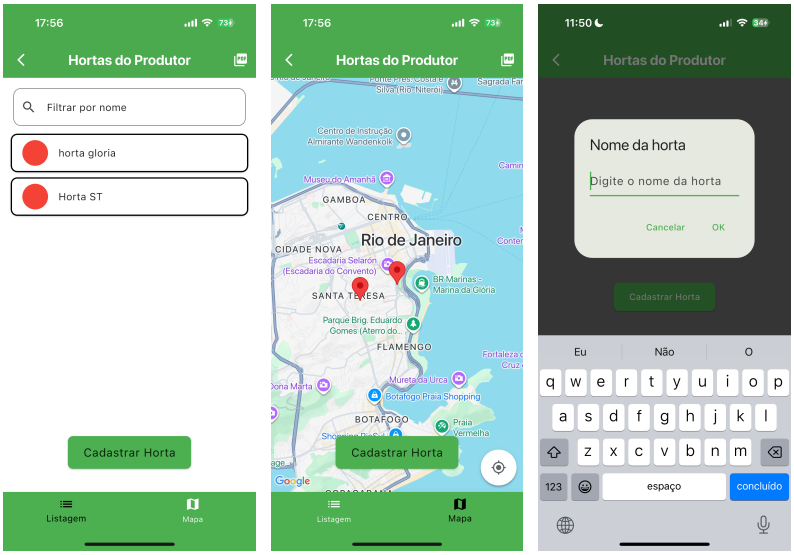
Figura 64 – Telas do fluxo de Visualização de Hortas



Fonte: Autor (2025)

Figura 65 – Telas do fluxo de Adicionar Horta

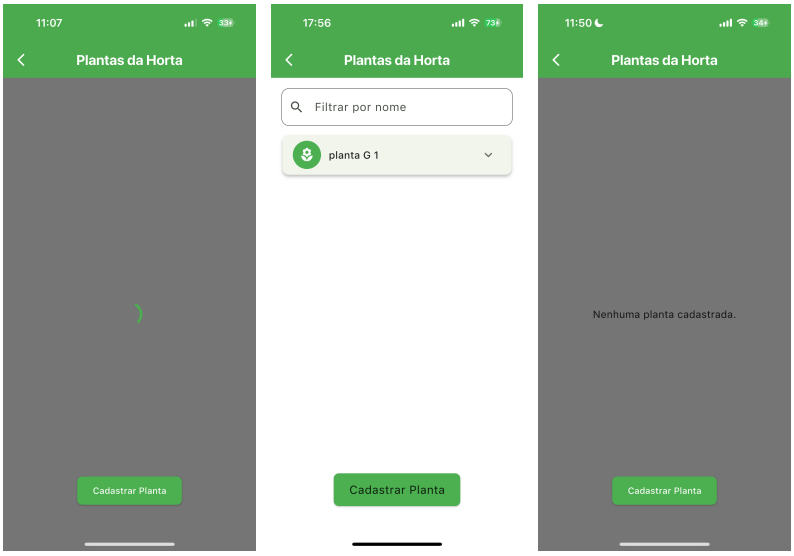
(a) (b) (c)



Fonte: Autor (2025)

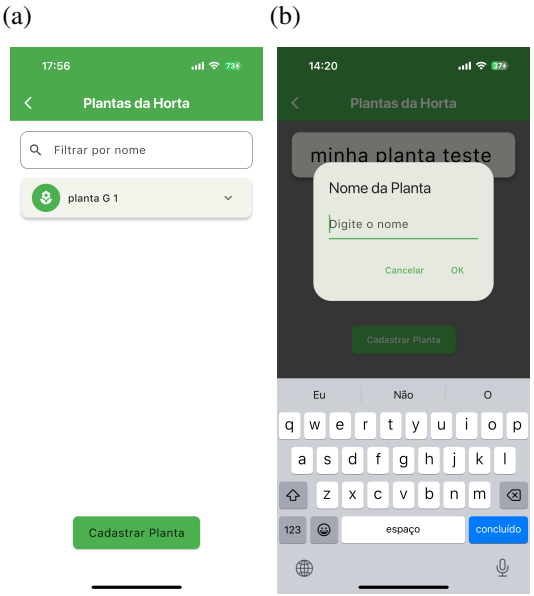
Figura 66 – Telas do fluxo de visualização de Plantas

(a) (b) (c)



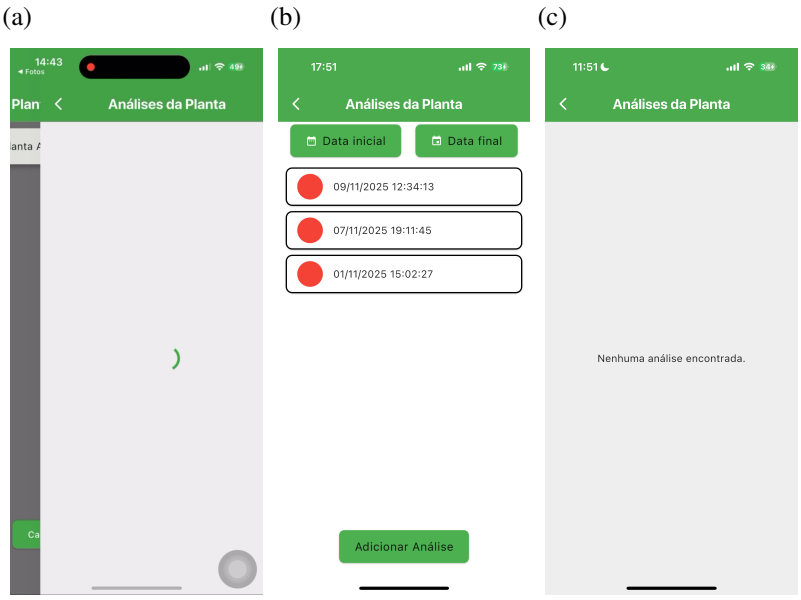
Fonte: Autor (2025)

Figura 67 – Telas do fluxo de adição de Plantas



Fonte: Autor (2025)

Figura 68 – Telas do fluxo de Visualização de Análises

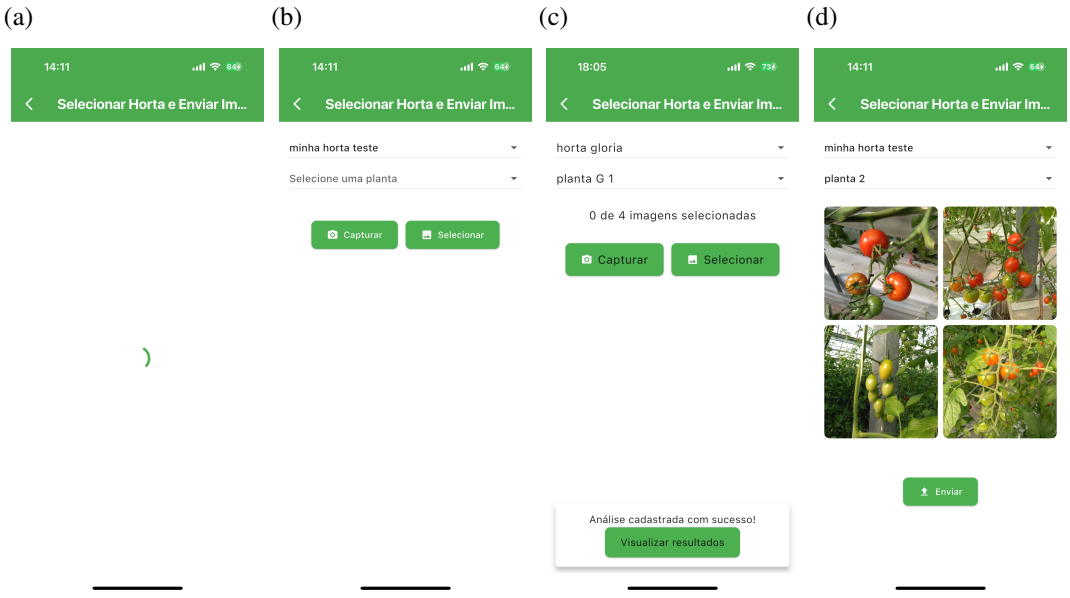


Fonte: Autor (2025)

Figura 69 – Telas do fluxo de Visualização detalhada de Análises

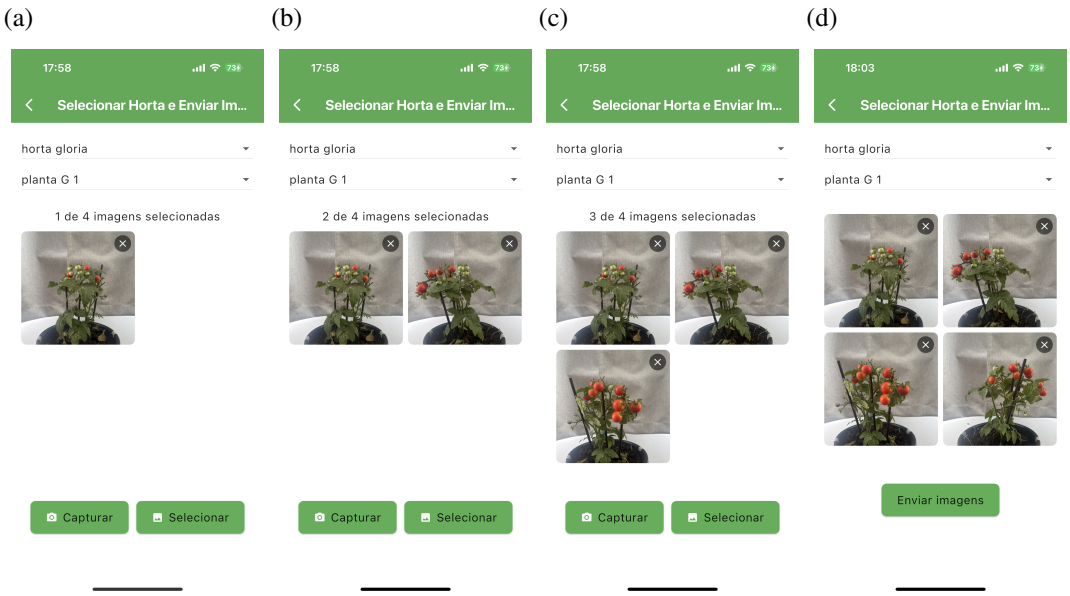
Fonte: Autor (2025)

Figura 70 – Telas do fluxo de Registro de Análises parte 1



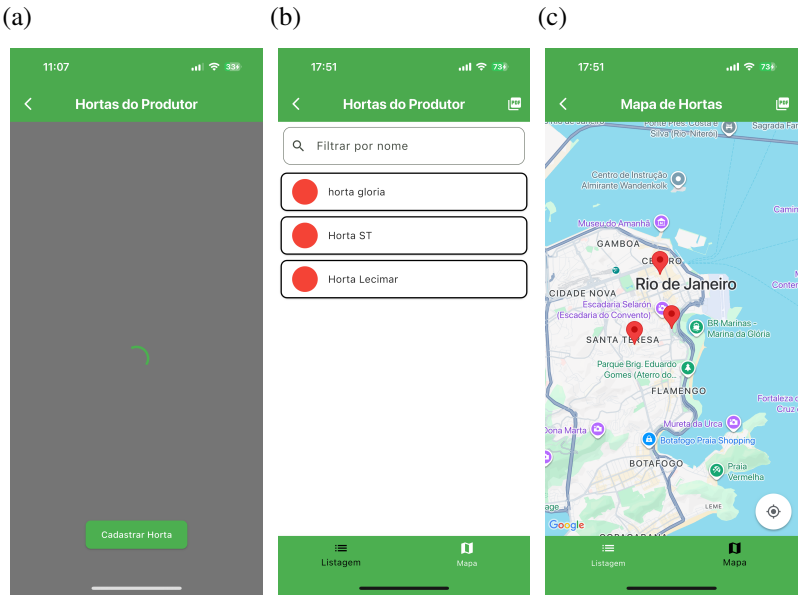
Fonte: Autor (2025)

Figura 71 – Telas do fluxo de seleção das imagens parte 2



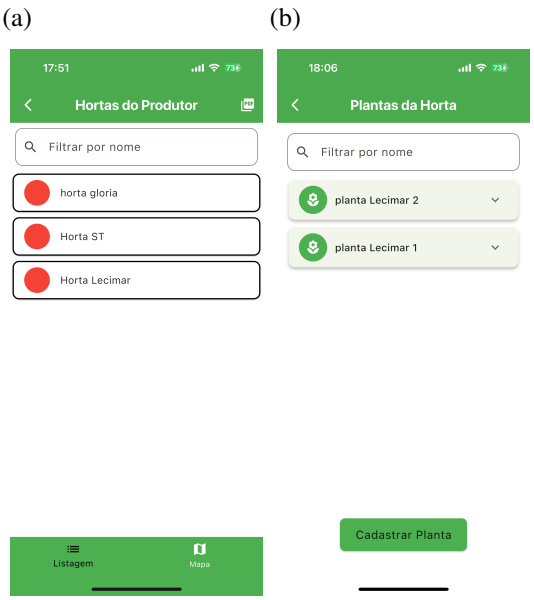
Fonte: Autor (2025)

Figura 72 – Telas do fluxo de seleção das imagens



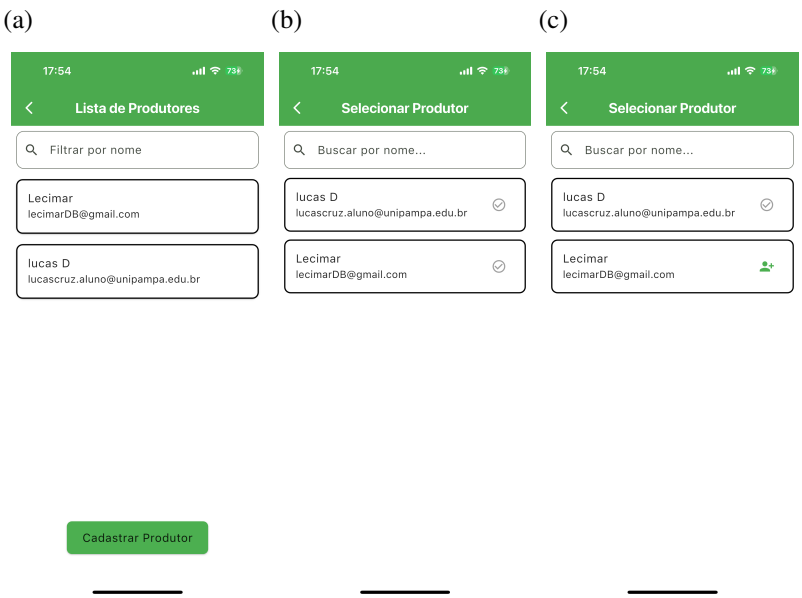
Fonte: Autor (2025)

Figura 73 – Telas do fluxo de visualizar hortas dos meus produtores



Fonte: Autor (2025)

Figura 74 – Telas do fluxo de adicionar produtor a supervisor



Fonte: Autor (2025)