INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS DISTRITO INDUSTRIAL
CURSO DE TECNOLOGIA EM ELETRÔNICA INDUSTRIAL

EMYLI BEATRIZ BRAGA PRESTES

SISTEMA INTEGRADO DE DETECÇÃO DE EPIS – SIDE

MANAUS-AM
2025

EMYLI BEATRIZ BRAGA PRESTES

SISTEMA INTEGRADO DE DETECÇÃO DE EPIS – SIDE

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, como requisito parcial, para obtenção do título de Tecnólogo em Tecnologia em Eletrônica Industrial.

Orientador: Me. Alexandre Lopes Martiniano

MANAUS-AM
2025

Dados Internacionais de Catalogação na Publicação (CIP)

Prestes, Emyli Beatriz Braga.
P936s Sistema Integrado de detecção de EPIS - SIDE / Emyli Beatriz Braga Prestes. — Manaus, 2025.
64f.: il. color.

Monografia (Graduação) — Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Distrito Industrial, Curso de Tecnologia em Eletrônica Industrial, 2025.
Orientador: Prof.º Alexandre Lopes Martiniano, Me.

1. Visão computacional. 2. Redes Neurais. l. 3. Monitoramento. I. Martiniano, Alexandre Lopes. II. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas. III. Título.

CDD 629.892

Elaborada por Oziane Romualdo de Souza (CRB11/ nº 734).

ANEXO 7

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos 11 dias do mês de ABRIL, de 2025, às 15:40 h, o(a) discente EMYLI BEATRIZ BRAGA PRESTES apresentou o seu Trabalho de Conclusão de Curso para avaliação da Banca Examinadora constituída pelos seguintes integrantes: Prof(a). Me. ALEXANDRE LOPES MARTINIANO (docente-orientador), Prof(a). DR. JOAS BERNARDO ARANHA RIBEIRO (Membro 1) e Prof(a). Me. HUGO ALVES VELOZO (Membro 2). A sessão pública de defesa foi aberta pelo(a) presidente da banca, que apresentou a Banca Examinadora e deu continuidade aos trabalhos, fazendo uma breve referência ao TCC, que tem como título SIDE - SISTEMA INTEGRADO DE DETECÇÃO DE EPIS. Na sequência, o(a) discente teve até 30 minutos para a comunicação oral de seu trabalho. Cada integrante da banca examinadora fez suas arguições após a defesa do mesmo. Ouvidas as explicações do(a) discente, a banca examinadora, reunida em caráter sigiloso, para proceder à avaliação final, deliberou e decidiu pela APROVAÇÃO com média final 9,5 (NOVE, CINCO) do referido trabalho.

Foi dada ciência ao(à) discente que a versão final do trabalho deverá ser entregue até o dia 15/04/2025, com as devidas alterações sugeridas pela banca. Nada mais havendo a tratar, a sessão foi encerrada às 16 h 40 min, sendo lavrada a presente ata, que, uma vez aprovada, foi assinada por todos os membros da Banca Examinadora e pelo(a) discente.

Prof.(a) Orientador(a)/Presidente: Alexandre Lopes Martiniano.

Prof.(a) Avaliador 1: João Bernardo A. Ribeiro

Prof.(a) Avaliador 2: Hugo Alves Velozo

Discente: Emyli Beatriz Braga Prestes

À minha família e aos meus amigos do IFAM.

AGRADECIMENTOS

Em primeiro lugar, expresso minha gratidão a Deus, pois reconheço que é graças a Ele que tenho vida e que Ele me trouxe até onde cheguei hoje.

Agradeço imensamente à minha avó, Maria Lourença, e ao meu saudoso avô Raimundo Prestes, que sempre me colocaram em primeiro lugar em suas vidas, me criando como sua filha e transmitindo seus valores e conhecimentos sobre a vida a mim.

Agradeço à minha mãe Glaucia Braga e ao meu pai, Fábio Prestes. Eles me trouxeram ao mundo e trabalharam arduamente para que nada me faltasse e eu não me sentisse só. Eles me apoiaram em momentos de dúvidas.

Agradeço à minha irmã Letícia Prestes, que foi minha confidente, amiga e ouvinte. Com ela compartilhei minhas dores, meus medos e anseios. E mesmo estando em outro país seguindo seu sonho, nunca deixou de me apoiar no meu sonho.

Também expresso minha gratidão aos meus amigos, como Paulo Fernandes, Carlos Santiago e outros que conheci ao longo dos anos de estudo no IFAM. A companhia deles proporcionou momentos fundamentais para minha estabilidade emocional, não me deixaram desistir, me incentivaram e caminharam comigo nessa jornada imprevisível e dificultosa, trazendo leveza e momentos felizes e engraçados.

Agradeço ainda ao Hugo Moraes, cuja presença constante e incentivo fizeram diferença nos momentos mais desafiadores desse percurso. Seu apoio foi essencial para que eu mantivesse o foco e concluísse meus objetivos.

E por último, e não menos importante, agradeço ao professor Me. Alexandre Lopes Martiniano, que me orientou nesse trabalho.

“Não adianta a gente ficar sentado se preocupando. O que tiver de ser, será, e nós o enfrentaremos quando vier.”

– Rúbeo Hagrid.

RESUMO:

Este trabalho apresenta o desenvolvimento de um sistema de visão computacional voltado à detecção automática de Equipamentos de Proteção Individual (EPIs) em ambientes de trabalho. A proposta busca integrar tecnologia de inteligência artificial com segurança ocupacional, contribuindo para a prevenção de acidentes e o cumprimento das normas regulamentadoras. O sistema utiliza o modelo YOLOv8, uma rede neural de detecção de objetos em tempo real, treinada para identificar os principais EPIs: capacete, óculos de proteção, máscara facial e colete refletivo. A anotação e preparação do dataset foram realizadas com o auxílio da plataforma Roboflow, que também facilitou o redimensionamento, organização e diversificação das imagens. Para a captura das imagens, foi utilizada uma câmera conectada a um Raspberry Pi, que envia os dados para uma interface gráfica desenvolvida em Python, com o framework Streamlit. Essa interface permite a visualização em tempo real dos equipamentos detectados, facilitando a análise pelo usuário. O sistema foi projetado para ser de fácil implantação e operação, sendo uma solução de baixo custo com potencial de uso em diversos setores industriais. Os testes demonstraram resultados promissores quanto à acurácia da detecção, mesmo com variações de iluminação. A pesquisa evidencia o potencial do uso de visão computacional e IoT na promoção de ambientes de trabalho mais seguros e monitorados de forma inteligente. A combinação entre hardware acessível, software intuitivo e modelos de deep learning mostrou-se eficaz na construção de sistemas de apoio à segurança do trabalho.

Palavras-chave: EPI. Monitoramento. Visão Computacional. UI. Python. Redes Neurais. YOLOv8. Streamlit. Roboflow. Raspberry.

ABSTRACT:

This work presents the development of a computer vision system aimed at the automatic detection of Personal Protective Equipment (PPE) in workplace environments. The proposal seeks to integrate artificial intelligence technology with occupational safety, contributing to accident prevention and compliance with regulatory standards. The system uses the YOLOv8 model, a real-time object detection neural network trained to identify the main PPE items: helmet, safety glasses, face mask, and reflective vest. The dataset annotation and preparation were carried out using the Roboflow platform, which also facilitated the resizing, organization, and diversification of the images. For image capture, a camera connected to a Raspberry Pi was used, sending the data to a graphical interface developed in Python using the Streamlit framework. This interface allows real-time visualization of the detected equipment, making analysis easier for the user. The system was designed to be easy to implement and operate, offering a low-cost solution with potential application in various industrial sectors. Tests showed promising results regarding detection accuracy, even under different lighting conditions. The research highlights the potential of using computer vision and IoT to promote safer and more intelligently monitored workplaces. The combination of accessible hardware, intuitive software, and deep learning models proved effective in building systems that support occupational safety.

Keywords: PE. Monitoring. Computer Vision. UI. Python. Neural Networks. YOLOv8. Streamlit. Roboflow. Raspberry.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

HTML: HyperText Markup Language
CSS: Cascading Style Sheets
IA: Inteligência Artificial
CNN: Redes Neurais Convolucionais
YOLO: You Only Look One
YOLOv8: You Only Look One versão 8
IHM: Interface Homem-Máquina
CNI: Confederação Nacional da Indústria
EPI: Equipamentos de Proteção Individual
CFTV: Circuito Fechado de Televisão
SGBDs: Sistemas de Gerenciamento de Bancos de Dados
LGPD: Lei Geral de Proteção de Dados
API: Interface de Programação de Aplicações
OS: Sistema Operacional

SUMÁRIO

1. INTRODUÇÃO

A segurança no ambiente de trabalho é um aspecto fundamental, especialmente em indústrias que envolvem atividades de risco elevado, como a construção civil e o setor manufatureiro. O uso de Equipamentos de Proteção Individual (EPIs) é essencial para preservar a integridade física dos trabalhadores e garantir a segurança no ambiente de trabalho (SANTOS; SANTOS, 2023). Segundo dados do Observatório de Segurança no Trabalho, entre 2012 e 2022, Manaus registrou 71.942 acidentes de trabalho, liderando o ranking na Região Norte. Foram contabilizados 451 óbitos no período, indicando que um trabalhador morreu a cada 8 dias devido a acidentes laborais na capital amazonense (RIO DE NOTÍCIAS, 2024).

Os métodos tradicionais para monitorar o uso de EPI, como a inspeção visual realizada por supervisores ou o uso de tecnologias de radiofrequência (RFID), têm limitações evidentes como a necessidade de integração com a infraestrutura existente, alto custo de implementação e dificuldades na detecção precisa do uso correto dos EPIs, uma vez que a simples presença do item não garante sua utilização adequada (OLIVEIRA et al., 2020). A observação humana está suscetível a erros, como a fadiga e a subjetividade das interpretações, além de não ser capaz de garantir monitoramento constante e eficaz (DELHI; SANKARLAL; THOMAS, 2020).

Sistemas baseados em RFID, apesar de automáticos, dependem da implementação de dispositivos físicos nos trabalhadores, o que pode ser invasivo e nem sempre garante a precisão necessária para detectar não conformidades (CPCON, 2021). Fatores como interferência eletromagnética, alto custo de implantação e limitações operacionais em ambientes metálicos ou úmidos podem comprometer a confiabilidade desses sistemas. Diante dessas dificuldades, podem surgir novas soluções tecnológicas que utilizam inteligência artificial.

Uma das abordagens mais inovadoras para a detecção automatizada do uso de EPI é a visão computacional, que utiliza algoritmos de aprendizado profundo, como as redes neurais convolucionais (CNNs), para analisar imagens e vídeos em tempo real (ZHANG et al., 2021). Esses sistemas são capazes de identificar automaticamente se os

trabalhadores estão utilizando os EPIs obrigatórios, como capacetes e coletes de segurança, sem a necessidade de intervenção humana ou de dispositivos adicionais nos trabalhadores (AKBARZADEH et al., 2020). A eficiência dessa tecnologia reside na capacidade de processar grandes quantidades de dados visuais rapidamente, garantindo um monitoramento contínuo e preciso. Além disso, elimina a necessidade de dispositivos físicos adicionais nos trabalhadores, o que a torna uma solução menos invasiva e mais adequada para ambientes industriais (HUSSAIN, 2023; AKBARZADEH et al., 2020).

A visão computacional tem se destacado como uma ferramenta poderosa não apenas na detecção de EPIs, mas em diversas aplicações na indústria, como na inspeção de qualidade de produtos e no controle de processos (DELHI; SANKARLAL; THOMAS, 2020). Entre os métodos de visão computacional para esse fim estão os algoritmos YOLO (REDMON et al., 2016) e Faster R-CNN (REN et al., 2015), que são amplamente reconhecidos por sua capacidade de realizar detecções em tempo real com alta precisão. Em locais de construção, essas tecnologias têm sido usadas para identificar comportamentos inseguros, como a falta de capacetes ou coletes de segurança, emitindo alertas automáticos que ajudam a prevenir acidentes e garantir a conformidade com as normas de segurança (DELHI; SANKARLAL; THOMAS, 2020; AKBARZADEH et al., 2020).

Dessa forma, o uso de visão computacional representa uma evolução significativa no campo da segurança do trabalho, oferecendo soluções robustas para monitoramento automatizado em tempo real. Ao aliar precisão tecnológica e capacidade de adaptação a diferentes ambientes industriais, essa abordagem se apresenta como um passo essencial para a redução de acidentes e para o fortalecimento da cultura de segurança nas organizações.

A implementação de sistemas baseados em visão computacional exige uma infraestrutura mínima que já está amplamente disponível em muitas organizações industriais: câmeras de monitoramento. Em muitos casos, essas câmeras fazem parte de sistemas de CFTV (Circuito Fechado de Televisão), tradicionalmente utilizados para vigilância passiva. A reutilização dessas câmeras como sensores visuais inteligentes

reduz custos e acelera a adoção da tecnologia. Ao integrar os algoritmos de detecção a sistemas de CFTV já existentes, é possível transformar um sistema originalmente passivo em um ativo de segurança inteligente, capaz de reagir em tempo real a situações de risco (VISUALCAM, 2023).

Apesar de sua eficácia, os modelos de visão computacional apresentam limitações conhecidas que devem ser consideradas em aplicações industriais. Fatores como interferência de iluminação, oclusões parciais dos EPIs, vieses nos datasets de treinamento e o tempo de inferência elevado em dispositivos com hardware limitado podem comprometer a precisão e a confiabilidade das detecções (LI et al., 2020). Esses desafios exigem cuidados no momento da coleta e rotulação dos dados, além da escolha adequada da arquitetura do modelo e da infraestrutura computacional utilizada.

Além da detecção, sistemas baseados em IA podem realizar a classificação de riscos e atribuir níveis de criticidade para cada evento detectado. Isso permite priorizar intervenções imediatas, como a ausência de capacete em áreas de risco elevado, em vez de emitir alertas generalizados. Essa priorização contribui para otimizar recursos e reforça a tomada de decisão baseada em dados, um dos pilares da Indústria 4.0 (ZHANG et al., 2021).

Além da detecção, sistemas baseados em IA podem realizar a classificação de riscos e atribuir níveis de criticidade para cada evento detectado. Isso permite priorizar intervenções imediatas, como a ausência de capacete em áreas de risco elevado, em vez de emitir alertas generalizados. Essa priorização contribui para otimizar recursos e reforça a tomada de decisão baseada em dados, um dos pilares da Indústria 4.0.

Outro benefício importante da aplicação de inteligência artificial na segurança do trabalho é a rastreabilidade histórica dos eventos. Cada ocorrência de não conformidade pode ser armazenada com registros visuais, horários e localização, o que facilita auditorias, relatórios gerenciais e análises de tendências. Dessa forma, a empresa pode implementar ações corretivas com base em dados reais, reduzindo a subjetividade no planejamento da segurança.

O treinamento de modelos de visão computacional para esse fim requer um dataset diversificado e bem anotado. A utilização de plataformas como o Roboflow auxilia na padronização do processo de anotação, aplicação de aumentos de dados (data argumentation). Isso garante um processo mais ágil e preciso no desenvolvimento de modelos de alta performance.

Um dos grandes desafios na detecção de EPIs é a variação de iluminação e ângulos de câmera. Ambientes industriais estão sujeitos a interferências visuais, poeira, sombras e movimentações bruscas. Modelos como YOLOv8 se destacam por sua robustez em diferentes condições, mantendo desempenho consistente e respostas rápidas.

A combinação de inteligência artificial com sistemas embarcados de baixo custo, como o Raspberry Pi, amplia ainda mais o alcance dessa tecnologia. Pequenos dispositivos podem ser alocados em locais estratégicos, processando imagens localmente e enviando apenas os alertas ou dados processados à nuvem.

A aceitação dessa tecnologia por parte dos trabalhadores também deve ser considerada. É importante que a implementação seja acompanhada por ações de conscientização e treinamentos, mostrando que o objetivo do sistema é preservar vidas e não fiscalizar de forma punitiva. A transparência sobre o uso dos dados coletados é essencial para criar confiança no sistema.

Do ponto de vista normativo, os sistemas inteligentes de monitoramento precisam estar alinhados às exigências da legislação trabalhista e das normas técnicas brasileiras, como a NR-6 e a NR-12. Além disso, a Lei Geral de Proteção de Dados (LGPD) deve ser observada para garantir que o tratamento de imagens e dados dos trabalhadores esteja em conformidade com os princípios de privacidade e segurança da informação.

Outro aspecto técnico relevante é a escalabilidade do sistema. Soluções baseadas em visão computacional podem ser adaptadas tanto para pequenas empresas quanto para grandes plantas industriais, variando apenas o número de câmeras e a capacidade de processamento. Essa flexibilidade permite que o investimento seja escalonado conforme o crescimento da organização ou o nível de risco das operações.

A utilização de interfaces gráficas como a desenvolvida com o framework Streamlit permite interação intuitiva com os dados, sem necessidade de conhecimentos avançados em programação. Essa interface pode exibir os EPIs detectados, registros históricos, e conformidade por setor.

Além do uso interno, os dados gerados por esses sistemas podem servir como indicadores de desempenho em auditorias externas e certificações de segurança, como a ISO 45001. Com isso, as empresas fortalecem sua imagem institucional, demonstrando compromisso com o bem-estar de seus colaboradores e com práticas modernas de gestão.

O avanço dessas tecnologias reforça o papel do engenheiro de segurança como um gestor de inovação. Mais do que apenas aplicar normas, espera-se que esse profissional identifique oportunidades para otimizar processos, aplicar tecnologias emergentes e criar ambientes laborais cada vez mais seguros, eficientes e humanos.

1.1 PROBLEMA DE PESQUISA

A motivação deste trabalho reside na busca por soluções que tornem os ambientes industriais mais seguros e eficientes, utilizando avanços tecnológicos para aprimorar a fiscalização e reduzir os índices de acidentes. A visão computacional, baseada em inteligência artificial, oferece um monitoramento em tempo real, minimizando falhas e otimizando a gestão da segurança do trabalho.

Estudos recentes demonstram que algoritmos de aprendizado profundo, como YOLO e Faster R-CNN, apresentam alta precisão na detecção de EPIs, garantindo um monitoramento eficaz sem a necessidade de dispositivos adicionais nos trabalhadores (REDMON et al., 2016; REN et al., 2017).

Além de contribuir para a preservação da vida e integridade dos trabalhadores, essa tecnologia também proporciona ganhos econômicos às empresas, reduzindo custos associados a afastamentos, indenizações e perdas operacionais decorrentes de acidentes (ZHOU et al., 2022).

Este estudo delimita-se à aplicação da tecnologia de visão computacional em um ambiente simulado, com condições controladas que representam cenários industriais comuns. A escolha por um ambiente simulado visa garantir a reprodutibilidade do experimento e o controle de variáveis externas, como iluminação e tipos de EPI utilizados. Essa abordagem permite avaliar a eficácia dos algoritmos de detecção, de forma mais precisa antes de sua validação em ambientes reais e complexos.

Ao explorar o potencial da visão computacional na detecção do uso de EPIs, este estudo busca demonstrar como a tecnologia pode ser aplicada para transformar a cultura de segurança nas organizações, promovendo um ambiente de trabalho mais seguro, eficiente e alinhado às exigências normativas.

1.2 OBJETIVOS

1.2.1 OBJETIVOS GERAIS

Desenvolver um sistema baseado em visão computacional utilizando algoritmos de aprendizado profundo para realizar a detecção automatizada de Equipamentos de Proteção Individual (EPIs) em ambiente simulado, visando aprimorar o monitoramento da segurança do trabalho de forma eficiente e em tempo real.

1.2.2 OBJETIVOS ESPECÍFICOS

- Desenvolver uma interface de usuário intuitiva utilizando a linguagem de programação Python.
- Permitir a visualização e controle do processo de detecção em tempo real.
- Implementar a integração do modelo com a interface do usuário através do Streamlit.
- Construir e configurar um modelo de inteligência artificial baseado em visão computacional para detecção de EPIs, utilizando algoritmos como YOLO.
- Realizar o treinamento do modelo com um conjunto de dados anotado, simulando diferentes cenários industriais com variações de iluminação, ângulos e tipos de EPI.

- Validar o funcionamento do sistema em ambiente simulado, verificando sua eficácia na identificação correta do uso ou ausência de EPIs.
- Documentar todas as etapas do desenvolvimento para garantir reprodutibilidade e possibilitar melhorias futuras.

1.3 JUSTIFICATIVA

A importância deste estudo é o avanço tecnológico e a crescente digitalização da indústria, alinhada aos princípios da Indústria 4.0. A aplicação da visão computacional na segurança do trabalho não apenas melhora a eficiência da fiscalização, mas também integrar-se a outras tecnologias, como Internet das Coisas (IoT), criando um ecossistema inteligente e autônomo para a gestão da segurança industrial (GUPTA et al., 2023).

A segurança em ambientes laborais é algo fundamental, e métodos utilizados como o RFID e a própria inspeção, apesar de serem abordagens já estabelecidas, ainda demonstram não serem totalmente eficazes (RIO DE NOTÍCIAS, 2024). Então a detecção automática e configuração da infraestrutura é uma abordagem que promete ser mais eficiente que os métodos tradicionais.

Outrossim, ferramentas como Roboflow e YOLOv8 facilitam de implementação para aplicações de visão computacional, além de permitir execução em dispositivos embarcados. E essa facilidade também traz replicabilidade, permitindo aperfeiçoamento e adaptação deste trabalho para aplicações mais específicas como EPIs não convencionais, ambientes laboratoriais e integração com sistemas já existentes.

1.4 METODOLOGIA

O capítulo dois apresenta os fundamentos teóricos que sustentam esse trabalho, segurança do trabalho e uso adequado de EPIs, destacando dados preocupantes sobre acidentes no Brasil. Também aborda as limitações das inspeções visuais humanas e a adoção de tecnologias como RFID para monitoramento.

Ainda no capítulo dois, são descritas as ferramentas e tecnologias utilizadas no projeto, Python, bibliotecas Python, YOLOv8 e Roboflow, que viabilizaram o desenvolvimento do sistema de detecção de EPIs. Abordando também a parte de hardware, escolhendo o Raspberry Pi e justificando as escolhas tecnológicas e metodológicas, alinhando-as às necessidades do projeto.

No capítulo três é descrito o desenvolvimento do Sistema Integrado de Detecção de EPIs (SIDE), detalhando sua arquitetura, funcionalidades e implementação. O sistema utiliza um Raspberry Pi e uma câmera conectada via USB para capturar imagens e processá-las em tempo real com o modelo YOLOv8, detectando os EPIs capacete, colete, máscara e óculos.

O desenvolvimento seguiu as etapas: levantamento de requisitos, coleta e rotulagem de dados com Roboflow, treinamento do modelo no Google Colab e implementação embarcada no Raspberry Pi.

O capítulo quatro apresenta os experimentos realizados no sistema de detecção de EPIs e os resultados do treinamento do modelo. Os testes foram divididos entre a análise individual de cada classe de EPI (capacete, colete, óculos e máscara) e a aplicação do sistema em tempo real, com funcionalidades como checklist automatizado.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos teóricos que sustentam o desenvolvimento deste trabalho, fornecendo a base necessária para a compreensão do sistema proposto. São abordados temas relacionados à segurança do trabalho e à importância do uso de Equipamentos de Proteção Individual (EPIs), bem como os fundamentos da visão computacional, redes neurais convolucionais e algoritmos de detecção de objetos, com ênfase no YOLOv8. Além disso, discute-se o uso de ferramentas e plataformas como Roboflow e Streamlit, essenciais para a construção e visualização do sistema. A fundamentação teórica visa contextualizar e justificar as escolhas metodológicas e tecnológicas adotadas ao longo do projeto.

2.1 SEGURANÇA DO TRABALHO E EPIS

A segurança em ambientes industriais é de extrema importância, pois envolve não apenas a preservação da integridade física e mental dos trabalhadores, mas também impacta diretamente a produtividade e a continuidade das operações industriais. Os acidentes de trabalho representam um grave problema de saúde pública e econômica. Dados recentes do Observatório de Segurança e Saúde no Trabalho apontam que, em 2021, foram notificadas 612.920 ocorrências de acidentes de trabalho no Brasil, com 2.538 óbitos, representando um aumento de 37% em relação ao ano anterior (CONSELHO NACIONAL DE JUSTIÇA, 2023).

Um estudo revela que 40% dos acidentes de trabalho no Brasil são causados por quedas de altura e, desses, cerca de 80% envolvem a ausência ou o uso incorreto de EPIs, o que reforça a importância do uso adequado desses dispositivos (SINTRICOMB, 2022). Como ilustrado na Figura 1, é possível observar a predominância desse tipo de acidente em relação a outros. A Figura 2 complementa os dados mostrando que a falta de uso adequado dos EPIs está diretamente relacionada ao aumento das quedas. A Norma Regulamentadora nº 6 (NR-6) estabelece que cabe ao empregador fornecer gratuitamente os EPIs adequados aos riscos existentes e garantir sua conservação e

higienização, além de orientar e treinar os trabalhadores quanto ao seu uso correto (UNIVERSIDADE FEDERAL DA FRONTEIRA SUL, 2022).

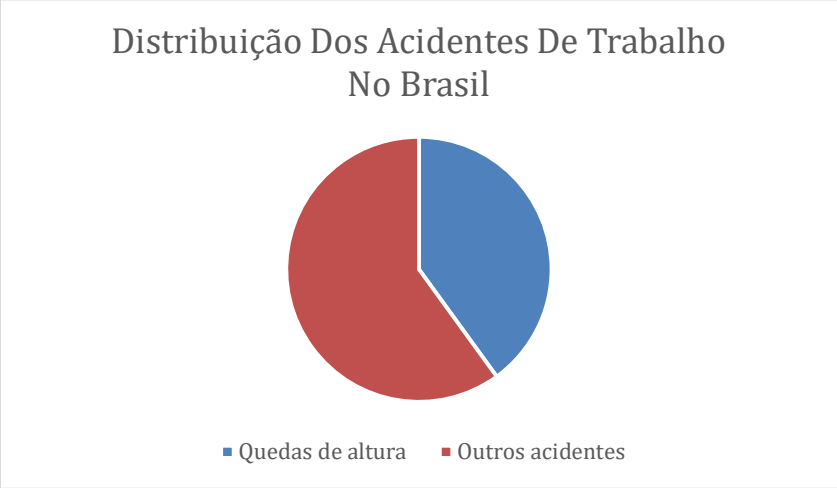

Figura 1: Gráfico de distribuição dos Acidentes de Trabalho no Brasil.7

Fonte: Elaborado pelo autor, 2025.

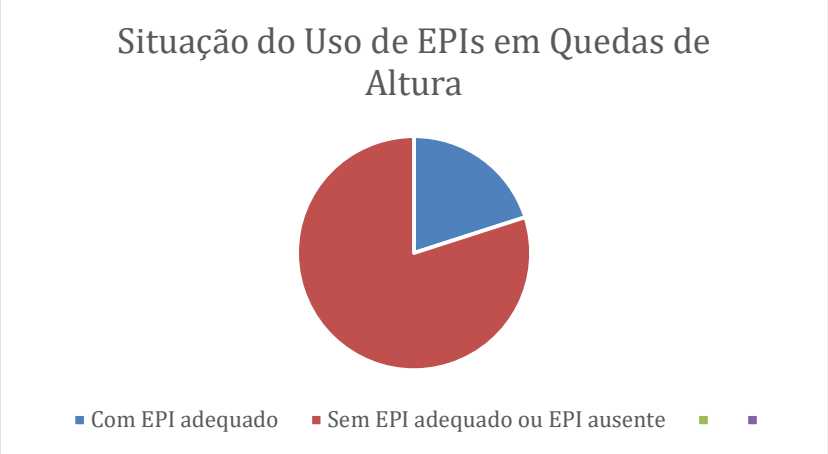

Figura 2: Gráfico de situação do Uso de EPIs em Quedas de Altura.

Fonte: Elaborado pelo autor, 2025.

Ainda assim, muitas empresas negligenciam essa obrigação, o que potencializa o risco de acidentes graves. A inspeção visual humana, frequentemente utilizada como medida de controle e segurança em ambientes industriais, apresenta limitações significativas, como a subjetividade das avaliações, o alto custo com mão de obra especializada e a possibilidade de falhas na detecção de anomalias não visíveis a olho nu, o que

compromete sua eficácia em ambientes que demandam precisão e agilidade (PORTAL POTÊNCIA, 2018).

Nesse sentido, tecnologias como a Identificação por Radiofrequência (RFID) vêm sendo adotadas como alternativas mais eficientes. Sistemas baseados em RFID são capazes de rastrear, monitorar e registrar informações em tempo real, o que otimiza o controle de EPIs e a movimentação de materiais. No entanto, apesar de seus benefícios, o uso de RFID ainda enfrenta desafios como os altos custos de implementação, dificuldades de integração com sistemas legados e interferências em ambientes industriais complexos, o que limita sua adoção em larga escala (ASSOCIAÇÃO NACIONAL DE TECNOLOGIA DO AMBIENTE CONSTRUÍDO, 2017).

2.2 FERRAMENTAS DE SOFTWARE

Para o desenvolvimento do sistema foi necessário conhecimento de ferramentas existentes, primeiramente a linguagem de programação Python, o banco de dados SQLite, o modelo de IA pré-treinado YOLO, a plataforma e biblioteca com ferramentas de visão computacional e treinamento de modelos chamada Roboflow, e por fim a biblioteca de criação rápida de apps Streamlit.

2.2.1 YOLO

Figura 3: Logo YOLO
Fonte: https://kawae.medium.com/deploying-yolov8-object-detection-model-with-tensorflow-serving-9232808c46dc

A visão computacional é um campo da inteligência artificial (IA) que permite que máquinas interpretem e compreendam o mundo visual. Utiliza técnicas de

processamento de imagens e aprendizado de máquina para extrair informações significativas de dados visuais. O aprendizado de máquina, por sua vez, é uma subárea da IA focada no desenvolvimento de algoritmos que permitem que sistemas aprendam padrões a partir de dados e façam previsões ou decisões sem serem explicitamente programados para isso (ZHANG et al., 2023). A Figura 3 ilustra a tecnologia de detecção de objetos.

A detecção de objetos envolve a identificação de instâncias de objetos de determinadas classes (como pessoas, carros ou animais) em imagens digitais. Essa tarefa é desafiadora devido à variabilidade em escala, posição, iluminação e oclusão dos objetos. Modelos baseados em aprendizado profundo, especialmente as redes neurais convolucionais, têm se mostrado eficazes na superação desses desafios, aprendendo representações hierárquicas de características visuais (GUPTA et al., 2022).

O YOLO faz uso de CNN (Redes Neurais Convolucionais) que são usadas principalmente para resolver tarefas difíceis de reconhecimento de padrões baseados em imagens e, com sua arquitetura precisa e ao mesmo tempo simples, oferecem um método simplificado para começar a trabalhar com ANNs (Redes Neurais Artificiais) que são modelos computacionais inspirados biologicamente capazes de superar em muito o desempenho das formas anteriores de inteligência artificial em tarefas comuns de aprendizado de máquina (O'SHEA; NASH, 2015).

O YOLO é um dos modelos mais proeminentes para detecção de objetos em tempo real. Diferentemente de abordagens anteriores que dividiam a imagem em regiões e aplicavam classificadores separadamente, o YOLO trata a detecção como um problema de regressão único, prevendo diretamente as caixas delimitadoras e as probabilidades de classe em uma única passagem pela rede neural. Essa abordagem unificada permite que o YOLO alcance velocidades de processamento significativamente mais altas sem comprometer a precisão (REDMON; FARHADI, 2018; WANG et al., 2023).

O YOLO passou por diversas evoluções desde sua primeira versão, lançada em 2016. Cada nova versão introduziu melhorias substanciais em termos de precisão, velocidade e capacidade de generalização. Por exemplo, o YOLOv2 incorporou ancoragem de

caixas (anchor boxes) e camadas de normalização por lote (batch normalization), enquanto o YOLOv3 utilizou uma arquitetura mais profunda baseada em redes residuais e introduziu detecção em múltiplas escalas. Já o YOLOv4 e versões posteriores, como o YOLOv5, YOLOv6 e YOLOv8, aproveitaram avanços em técnicas de treinamento, arquiteturas otimizadas e bibliotecas eficientes para alcançar um equilíbrio ainda melhor entre desempenho e precisão (WANG et al., 2023). As características das versões do YOLO estão detalhadas na Tabela 1, que apresenta seus pontos fortes e limitações.

Tabela 1: Tabela de características das versões do YOLO

Versão Yolo	Pontos Fortes	Limitações
Yolov1	Processamento rápido em tempo real	Dificuldade em detectar objetos pequenos
Yolov2	Melhor recall, lida melhor com objetos pequenos	Maior demanda computacional
Yolov3	Detecção em múltiplas escalas, bom desempenho em condições diversas	Não é otimizado para dispositivos de baixa potência
Yolov4	Robusto em ambientes visuais complexos	Treinamento complexo para conjuntos de dados personalizados
Yolov5	Muito rápido, adequado para aplicações em tempo real	Pode exigir ajustes finos para cenários específicos
Yolov6	Alta acurácia com profundidade aprimorada	Requer maior poder de processamento para melhores resultados

Yolov7	Alta precisão, eficaz em cenas com muitos objetos	Eficiência computacional reduzida em larga escala
Yolov8	Velocidade muito alta, ideal para ambientes dinâmicos	Pode ter dificuldades com objetos muito pequenos ou em movimento

A principal força do YOLO reside em sua capacidade de inferência em tempo real, o que o torna particularmente vantajoso em cenários de resposta imediata, mesmo em dispositivos com recursos computacionais limitados. Além disso, por tratar a detecção como um problema global, o YOLO tende a apresentar menos falsos positivos em comparação com métodos que dependem de propostas de regiões (*region proposal methods*), como o R-CNN e suas variantes.

Contudo, apesar de suas vantagens, o YOLO também possui limitações. Devido à sua abordagem baseada em grade fixa, ele pode ter dificuldades em detectar objetos muito pequenos ou agrupados próximos uns dos outros. Além disso, erros de localização ainda podem ocorrer, especialmente em classes com alta similaridade visual. Ainda assim, o contínuo aprimoramento de suas versões, aliado à ampla adoção da comunidade científica e industrial, consolidou o YOLO como uma das soluções mais robustas e práticas para tarefas de detecção de objetos em ambientes complexos e dinâmicos (ZHANG et al., 2023; WANG et al., 2023).

O YOLOv8, desenvolvido pela Ultralytics, representa um avanço significativo na detecção de objetos em tempo real, consolidando-se como uma ferramenta essencial na visão computacional. Esta versão aprimorada introduz uma arquitetura mais eficiente, incorporando uma cabeça de detecção livre de âncoras, o que simplifica o modelo e aumenta a precisão na identificação de objetos de diferentes tamanhos e formas (GLENN; ULTRALYTICS, 2023).

Comparado a versões anteriores, como o YOLOv5, o YOLOv8 apresenta melhorias notáveis em termos de precisão e velocidade. A introdução de técnicas avançadas de

aumento de dados, como a combinação de imagens em mosaico durante o treinamento, contribui para uma melhor generalização do modelo em cenários variados. Essas inovações tornam o YOLOv8 particularmente adequado para aplicações que exigem processamento em tempo real e alta acurácia (GLENN; ULTRALYTICS, 2023).

Dessa forma, o YOLOv8 se apresenta como uma excelente alternativa para a implementação de protótipos em pesquisas voltadas à detecção de objetos. Sua arquitetura robusta com está ilustrada na Figura 4.

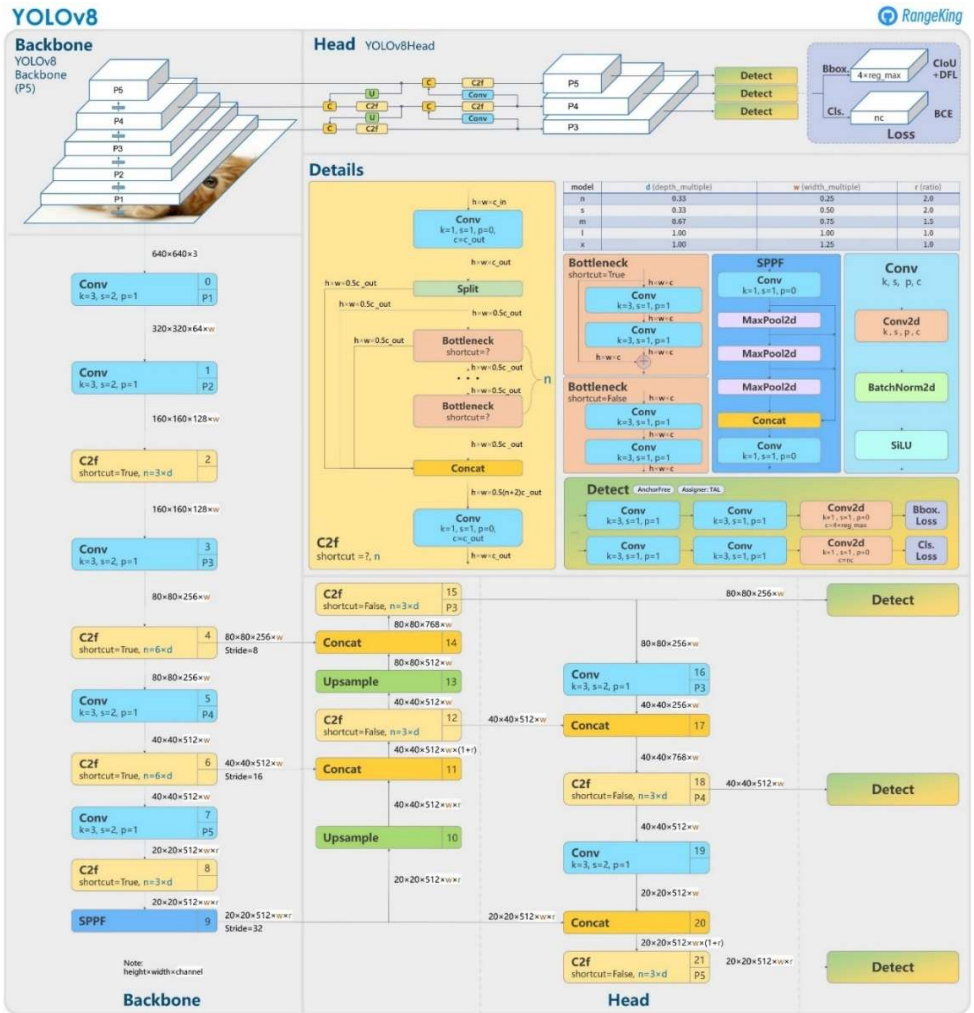

Figura 4: Arquitetura YOLOv8

Fonte: https://github.com/ultralytics/ultralytics/issues/189

2.2.2 ROBOFLOW

Figura 5: Logo Roboflow
Fonte: https://roboflow.com

O Roboflow é uma plataforma de software como serviço (SaaS) projetada para facilitar o desenvolvimento de aplicações de visão computacional. Fundada em 2019 por Joseph Nelson e Brad Dwyer, a empresa tem como missão tornar a visão computacional acessível a desenvolvedores de todos os níveis de experiência. A plataforma oferece ferramentas que abrangem todo o ciclo de vida de um projeto de visão computacional, desde a gestão de datasets até a implantação de modelos treinados. Atualmente, mais de 1 milhão de engenheiros utilizam o Roboflow para criar datasets, treinar modelos e implantar soluções em produção (ROBOFLOW, 2023). A Figura 5 apresenta o logo do Roboflow

O Roboflow permite que os usuários explorem, visualizem, filtrem e organizem seus dados de forma eficiente. A plataforma oferece ferramentas para compreender melhor os datasets, identificar possíveis lacunas e aprimorar a qualidade dos dados utilizados no treinamento de modelos.

Com infraestrutura otimizada, o Roboflow oferece treinamento hospedado de modelos de última geração. Os usuários podem escolher entre diferentes tamanhos de modelos para equilibrar entre rapidez e precisão. Além disso, a plataforma permite avaliar o desempenho dos modelos treinados, identificando áreas de melhoria e orientando a coleta de novos dados.

O Roboflow se destaca por sua capacidade de integração com uma variedade de ferramentas e plataformas. Entre as integrações disponíveis estão serviços de armazenamento como AWS S3 e Google Cloud, bancos de dados como Supabase, e frameworks de treinamento como TensorFlow e PyTorch. Além disso, a plataforma oferece suporte a diversos formatos de anotação e é compatível com câmeras e dispositivos de captura de diferentes fabricantes.

O Roboflow se destaca como uma plataforma robusta e eficiente para o desenvolvimento de protótipos em visão computacional, especialmente quando se utiliza o modelo YOLO para tarefas de detecção de objetos. A integração direta com o YOLO simplifica o processo de treinamento de modelos personalizados, permitindo que desenvolvedores importem datasets, realizem anotações e treinem modelos.

2.2.3 PYTHON

Figura 6 – Logo Python.
Fonte: https://www.python.org

Python é uma linguagem de programação de alto nível, interpretada, de tipagem dinâmica e com forte ênfase na legibilidade do código. Criada por Guido van Rossum e lançada em 1991, Python foi desenvolvida com a filosofia de que "código legível é tão importante quanto o código funcional" (PYTHON, 2023). Ao longo dos anos, essa linguagem tem se consolidado como uma das mais utilizadas no mundo em áreas como desenvolvimento web, ciência de dados, inteligência artificial, automação, computação científica, entre outras. A figura 6 apresenta o logo do Roboflow.

A sintaxe simples e intuitiva, combinada com uma vasta biblioteca padrão e uma comunidade ativa, fez de Python uma linguagem versátil e poderosa. Seu design limpo e sua ênfase na legibilidade facilitam a aprendizagem para iniciantes e tornam o desenvolvimento mais ágil e eficiente para programadores experientes.

No que se refere ao desenvolvimento de protótipos, Python demonstra particular eficiência, sobretudo em ambientes que demandam agilidade, experimentação e interação constante. A linguagem permite uma rápida transição da concepção teórica para uma implementação funcional, fator essencial em contextos de pesquisa aplicada, validação de hipóteses e testes de usabilidade (ZELLER, 2020).

Isso se deve, na maioria, ao ecossistema robusto de bibliotecas e frameworks disponíveis, que oferecem soluções prontas para tarefas comuns e complexas, desde manipulação de dados até criação de interfaces gráficas e integração com serviços web.

Do ponto de vista da engenharia de software, Python favorece a prototipagem por permitir o desenvolvimento incremental. Em comparação com linguagens compiladas como C++ ou Java, que exigem configurações mais rigorosas de ambiente e uma arquitetura inicial mais formalizada, Python permite que o desenvolvedor concentre esforços na lógica de negócio e na resolução do problema central. Tal característica contribui para ciclos de desenvolvimento mais curtos e para uma redução significativa no tempo de validação funcional.

Python apresenta também vantagens importantes quando se trata da interoperabilidade com outras ferramentas e linguagens, facilitando a integração de módulos desenvolvidos em diferentes tecnologias. Essa característica é especialmente relevante em projetos multidisciplinares, nos quais a comunicação entre sistemas heterogêneos é frequente.

Outro aspecto relevante é a extensa documentação e o suporte comunitário, que reduzem barreiras técnicas e promovem a autonomia de estudantes e pesquisadores na construção de soluções experimentais. Uma vez que permite a incorporação de modelos computacionais sofisticados desde as fases iniciais do projeto.

Dessa forma, a escolha de Python como linguagem para prototipação se justifica não apenas pela sua sintaxe acessível, mas principalmente por sua capacidade de acelerar processos de inovação, viabilizar testes de conceito a curto prazo e manter

compatibilidade com soluções em larga escala que possam ser implementadas futuramente.

2.2.4 BIBLIOTECAS PYTHON

O desenvolvimento deste trabalho contou com o uso de diversas bibliotecas da linguagem Python, fundamentais para a implementação do sistema de detecção automatizada de EPIs. A biblioteca Torchvision foi usada para aplicar Non-Maximum Suppression, que visa evitar a detecção duplicada de EPIs. O OpenCV foi utilizado para o processamento em tempo real das imagens capturadas pela câmera, além da exibição dos resultados com caixas delimitadoras sobre os objetos detectados. A biblioteca ultralytics, responsável por disponibilizar uma interface otimizada para os modelos da família YOLO. E O Streamlit foi a biblioteca escolhida para desenvolver a interface gráfica interativa, permitindo o controle do sistema e a visualização dos resultados de forma clara e acessível ao usuário final.

2.2.4.1 SQLITE

Os bancos de dados relacionais têm desempenhado um papel fundamental no armazenamento e gerenciamento de informações nas últimas décadas. Entre as diversas soluções disponíveis, o SQLite destaca-se por sua leveza, eficiência e ampla adoção em aplicações que vão desde dispositivos móveis até sistemas embarcados (GAFFNEY; HIPP, 2022). Este capítulo explora a evolução dos bancos de dados relacionais, a linguagem SQL e, em particular, o surgimento e as características distintivas do SQLite.

A necessidade de organizar e acessar dados de maneira eficiente levou ao desenvolvimento de sistemas de gerenciamento de bancos de dados (SGBDs). Entre os modelos propostos, o modelo relacional, introduzido por Edgar F. Codd na década de 1970, revolucionou a forma como os dados são estruturados, utilizando tabelas para representar entidades e seus relacionamentos (GRASER, 2020).

Para interagir com bancos de dados relacionais, foi criada a Structured Query Language (SQL), uma linguagem padronizada que permite definir, manipular e consultar dados. O

SQL tornou-se essencial para operações como inserção, atualização e recuperação de informações, sendo adotado amplamente em diversos SGBDs (LEARNSQL.COM, 2019).

O SQLite é uma biblioteca escrita em linguagem C que implementa um banco de dados SQL embutido. Diferentemente de outros SGBDs que seguem a arquitetura cliente-servidor, o SQLite é integrado diretamente às aplicações, eliminando a necessidade de um processo de servidor separado. Toda a base de dados, incluindo definições, tabelas, índices e dados, é armazenada em um único arquivo no disco, facilitando a portabilidade e a simplicidade no gerenciamento de dados (GAFFNEY; HIPP, 2022).

Essa arquitetura simplificada torna o SQLite particularmente atraente em contextos de prototipagem de sistemas. Durante as fases iniciais de desenvolvimento de software, é comum que equipes busquem soluções que ofereçam agilidade, baixo custo de configuração e facilidade de uso. O SQLite atende a esses requisitos ao permitir a criação de protótipos funcionais com mínima sobrecarga técnica, dispensando a instalação e configuração de servidores de banco de dados complexos (CHEN; LI; YU, 2020).

A ausência de dependências externas e o suporte nativo em diversas linguagens de programação tornam o SQLite ideal para testes rápidos e validações de conceito. Seu desempenho é mais do que suficiente para aplicações de pequeno e médio porte, permitindo que desenvolvedores foquem no design da aplicação (GAFFNEY; HIPP, 2022).

2.2.4.2 ULTRALYTICS

A biblioteca Ultralytics foi escolhida como a interface para utilizar o modelo de aprendizado de máquina previamente treinado e carregado a partir de um arquivo. Essa biblioteca oferece uma API de alto nível que facilita a implementação, execução e avaliação de modelos de visão computacional, como os baseados na arquitetura YOLO.

2.2.4.3 OPENCV

A biblioteca OpenCV (Open Source Computer Vision Library) é uma das ferramentas mais amplamente utilizadas no campo da visão computacional e processamento de imagens. A OpenCV oferece uma vasta gama de funcionalidades para análise de imagens, detecção de objetos, rastreamento de movimento, reconhecimento facial, entre outras aplicações.

No contexto deste trabalho, a OpenCV foi utilizada como a principal ferramenta para capturar vídeos e realizar a detecção e marcação de Equipamentos de Proteção Individual (EPIs) em imagens.

2.2.4.4 TORCHVISION

A biblioteca Torchvision é um componente do ecossistema PyTorch, desenvolvida para facilitar a implementação e experimentação de algoritmos de visão computacional. Essa biblioteca oferece um conjunto abrangente de ferramentas que inclui conjuntos de dados padronizados, métodos de transformação e pré-processamento de imagens, além de modelos pré-treinados em tarefas de classificação, detecção e segmentação.

Nesse trabalho, foi usado o Torchvision para aplicar o algoritmo Non-Maximum Suppression (NMS), que visa evitar a detecção duplicada de EPIs nas imagens capturadas.

2.2.4.5 STREAMLIT

Figura 7: Logo Streamlit
Fonte: https://streamlit.io

Streamlit é um framework de código aberto em Python que tem ganhado destaque no desenvolvimento de aplicações web interativas voltadas para a análise e visualização de

dados. Lançado com o objetivo de simplificar a criação de dashboards e ferramentas analíticas, o Streamlit permite que cientistas de dados e engenheiros de machine learning transformem scripts de dados em aplicativos web funcionais com poucas linhas de código. Esta abordagem democratiza o acesso a ferramentas interativas, eliminando a necessidade de conhecimentos aprofundados em desenvolvimento web (STREAMLIT INC., 2023). A figura 7 apresenta o logo do Streamlit.

A arquitetura do Streamlit é projetada para ser intuitiva e eficiente. Utilizando uma abordagem declarativa, os desenvolvedores podem escrever scripts Python que, ao serem executados, geram interfaces web automaticamente. Entre as principais funcionalidades do Streamlit, destacam-se widgets interativos que facilitam a criação de elementos como sliders, botões e caixas de seleção, permitindo interatividade em tempo real com os dados.

Suporte a bibliotecas de visualização, integra-se de forma nativa com bibliotecas populares como Matplotlib, Plotly e Altair, ampliando as possibilidades de visualização de dados. Atualização automática, detecta alterações no código-fonte e atualiza a aplicação em tempo real, agilizando o processo de desenvolvimento. O desenvolvimento simplificado elimina a necessidade de escrever código HTML, CSS ou JavaScript, permitindo que o foco permaneça na lógica da aplicação e na análise dos dados.

O Streamlit destaca-se como uma ferramenta excepcional para prototipagem rápida de aplicações web interativas. Sua sintaxe simples e abordagem declarativa permitem que desenvolvedores transformem scripts Python em interfaces funcionais com rapidez e eficiência. A capacidade de iterar rapidamente sobre o design e a funcionalidade da aplicação facilita a experimentação e o refinamento de ideias, tornando o Streamlit uma escolha estratégica para projetos que demandam agilidade e flexibilidade no processo de desenvolvimento.

2.3 TECNOLOGIAS DE HARDWARE

A implementação prática de sistemas de detecção de objetos em ambientes reais depende fortemente do uso de hardware embarcado. Esse tipo de hardware é composto

por dispositivos eletrônicos compactos, de baixo consumo energético, capazes de executar tarefas computacionais específicas em tempo real. O uso de hardware embarcado é fundamental para viabilizar soluções de visão computacional em cenários industriais, especialmente em aplicações que exigem mobilidade, autonomia ou integração com sensores e atuadores físicos (MOREIRA et al., 2019).

A integração entre visão computacional e dispositivos físicos, como motores, alarmes ou sistemas de iluminação, possibilita a automação de respostas a eventos visuais detectados por algoritmos inteligentes. Por exemplo, sistemas que identificam falhas visuais em produtos podem, ao detectar um defeito, acionar mecanismos de rejeição automática na linha de produção. Essa combinação de processamento visual com interação física permite desenvolver sistemas mais responsivos, eficientes e seguros (LUCAS; DIAS; FONSECA, 2020).

Entre os dispositivos mais populares para aplicações embarcadas está o Raspberry Pi, um microcomputador de baixo custo e tamanho reduzido, que possui capacidade suficiente para executar algoritmos de inteligência artificial e visão computacional básica. O Raspberry Pi opera com sistemas Linux, como o Raspberry Pi OS, e suporta linguagens como Python e C++, o que o torna altamente acessível para pesquisadores e desenvolvedores (RASPFOUNDATION, 2023).

2.3.1 RASPBERRY

A Raspberry Pi utilizada neste projeto foi a Raspberry Pi 4 Model B mostrada na figura 8, com 4GB de memória RAM. Esse modelo oferece desempenho suficiente para tarefas de visão computacional básica, além de apresentar bom custo-benefício para ambientes industriais e acadêmicos. A placa possui múltiplas interfaces de comunicação, incluindo USB 3.0, HDMI, Ethernet, Wi-Fi, Bluetooth, além de um conjunto de 40 pinos GPIO que permitem o controle de sensores e atuadores como mostra a figura 9.

Figura 8: Raspberry Pi 4 Model B.

Fonte: Próprio autor, 2025.

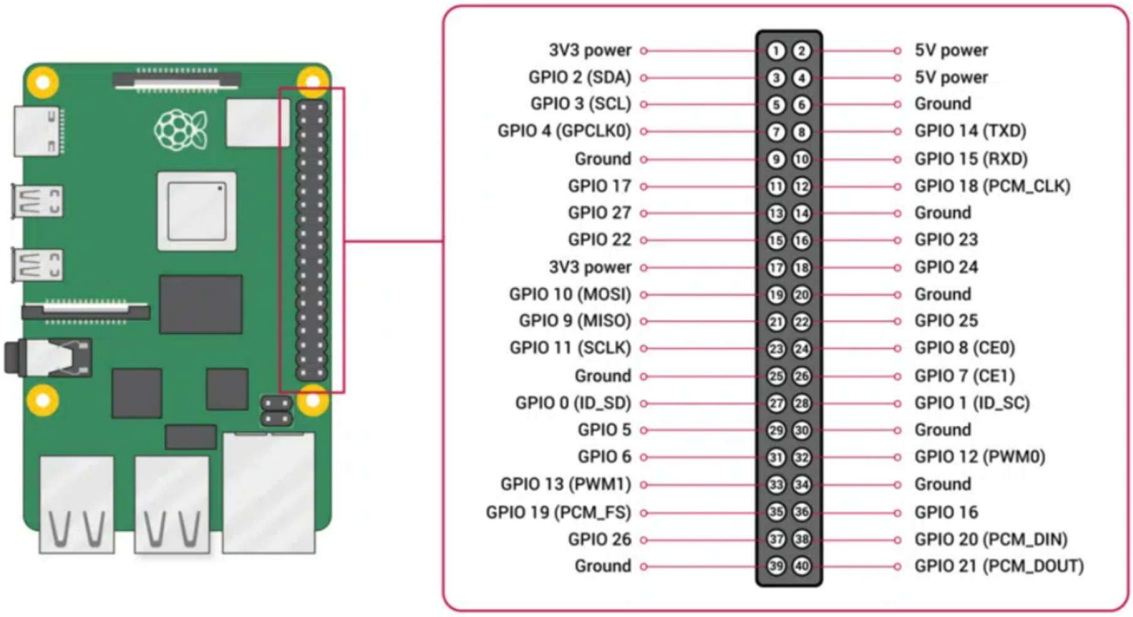

Figura 9: Tabela de pinos Raspberry Pi 4 Model B.

Fonte: https://www.makerhero.com/produto/raspberry-pi-4-model-b/.

O Raspberry Pi se destaca por sua conectividade e versatilidade. Ele possui interfaces para conexão com USB, HDMI, Ethernet, Wi-Fi, Bluetooth e GPIOs, que permitem a integração com uma grande variedade de sensores e atuadores. Esses recursos tornam o dispositivo útil em diferentes setores da indústria, como agricultura de precisão, monitoramento de máquinas, inspeção visual de peças e segurança patrimonial (MONTEIRO; SANTOS, 2022).

No entanto, por ser limitado em capacidade de processamento e memória, o Raspberry Pi pode apresentar dificuldades na execução de modelos mais complexos de redes neurais profundas, especialmente em tempo real ou em aplicações de alta resolução.

Diversos projetos industriais já utilizaram o Raspberry Pi como plataforma principal para coleta de dados visuais e controle de sistemas automatizados. Na indústria têxtil, por exemplo, o Raspberry Pi tem sido utilizado para monitorar falhas de costura por meio de câmeras acopladas à linha de produção. No setor logístico, ele tem servido como base para sistemas de rastreamento e leitura de etiquetas visuais em tempo real (OLIVEIRA; VIEIRA; BASTOS, 2021).

A conectividade com câmeras, sejam USB ou por interface MIPI CSI, é um dos diferenciais do Raspberry Pi. A câmera oficial (Raspberry Pi Camera Module) fornece resolução de até 12 MP com foco automático e compatibilidade com bibliotecas como OpenCV e libcamera. A qualidade da imagem, no entanto, depende de fatores como iluminação, lente utilizada e velocidade de captura. Em condições ideais, o módulo permite a captura de imagens em alta definição, suficiente para tarefas como reconhecimento de EPIs ou leitura de códigos visuais (RASPFOUNDATION, 2023).

Apesar da boa qualidade de imagem oferecida, a taxa de quadros por segundo (FPS) e a latência do sistema podem limitar aplicações que exigem resposta ultrarrápida ou análise de fluxos de vídeo de alta complexidade. Em alguns casos, utiliza-se o Raspberry Pi apenas como intermediador, delegando o processamento pesado a servidores externos ou ao uso de aceleradores como o Google Coral ou o NVIDIA Jetson Nano (FONSECA et al., 2021).

Além do uso com câmeras fixas, o Raspberry Pi pode ser embarcado em drones, robôs ou sistemas móveis de inspeção. Sua portabilidade e o baixo consumo o tornam ideal para operações remotas, como inspeção de áreas perigosas ou monitoramento agrícola. Na indústria 4.0, essas soluções estão alinhadas ao conceito de IoT (Internet das Coisas), onde dispositivos inteligentes comunicam-se em rede para tomada de decisões autônomas (SOUZA; OLIVEIRA, 2022).

2.3.2 INSTALANDO E ATUALIZANDO O RASPBERRY

Nesse trabalho o OS utilizado foi o RaspBerry pi OS, uma distribuição oficial baseada no Debian, otimizada para o hardware do Raspberry Pi. Essa escolha foi motivada pela sua estabilidade, suporte a atualizações regulares e ampla compatibilidade com bibliotecas e ferramentas de desenvolvimento, como Python e OpenCV. O sistema operacional também oferece uma interface gráfica amigável, facilitando a configuração inicial e o gerenciamento de recursos.

Para a instalação do sistema operacional são necessários os seguintes dispositivos:

- Cartão microSD de pelo menos 8 GB
- Um dispositivo com leitor de cartão microSD para escrever a imagem do OS no cartão
- Fonte de Alimentação de 5V com entrada USB-C para o Raspberry
- Cabo HDMI para conectar o monitor com o Raspberry
- Monitor compatível com HDMI para utilizar o OS no Raspberry
- Teclado e Mouse para interagir com o Raspberry
- Conexão com a internet

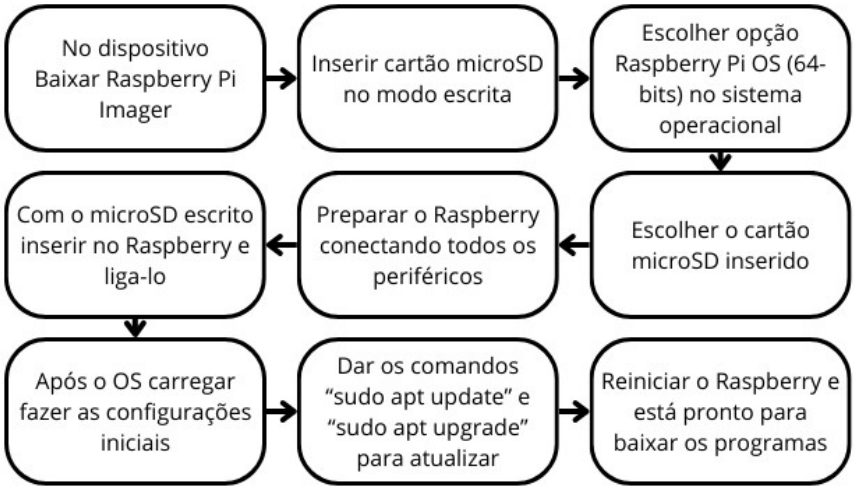

Figura 10: Diagrama de Blocos para instalar Raspberry Pi OS
Fonte: Elaborado pelo autor, 2025.

Como descrito na figura 10, no dispositivo baixar o programa Raspberry Pi Imager, insira no cartão micros no modo escrita, no programa escolha a operação de sistema operacional o “Raspberry Pi OS (64-bits)”, escolha o cartão SD inserido e espere ele gravar. Após a gravação, prepare o Easpberry conectando todos os cabos de mouse, teclado, HDMI, monitor e cabo de internet (se necessário) e fonte. Então conecte o microSD no Raspberry e ligue-o.

Após o sistema operacional carregar, faça as configurações iniciais e atualize o sistema com os comandos “sudo apt update” e “sudo apt upgrade”, e reinicie o Raspberry. Após esses passos o sistema está pronto para uso, basta baixar o programa e suas dependências e executa-lo (FAGNER et al., 2025).

Em suma, o Raspberry Pi representa uma solução viável e acessível para a prototipagem e implementação de sistemas de visão computacional na indústria. Apesar de suas limitações de hardware, sua capacidade de integração com câmeras e sensores, combinada ao suporte de bibliotecas modernas, permite desenvolver aplicações robustas em variados setores industriais. A escolha por seu uso, no entanto, deve ser feita considerando as exigências específicas de processamento, latência e escalabilidade de cada projeto.

3. SIDE - SISTEMA INTEGRADO DE DETECÇÃO DE EPIS

Este capítulo apresenta o desenvolvimento do Sistema de Detecção de EPIs (SD-EPI), destacando sua arquitetura, funcionalidades e implementação. O sistema foi projetado para identificar automaticamente o uso de equipamentos de proteção individual como colete, capacete, óculos e máscara, por meio de um módulo embarcado baseado em Raspberry Pi.

Além da detecção de imagens estáticas e em tempo real, o sistema possibilita a criação de setores com diferentes combinações obrigatórias de EPIs. Por meio da interface de checklist, o acesso é liberado caso todos os equipamentos exigidos sejam detectados e negado caso contrário.

3.1 ARQUITETURA DO HARDWARE

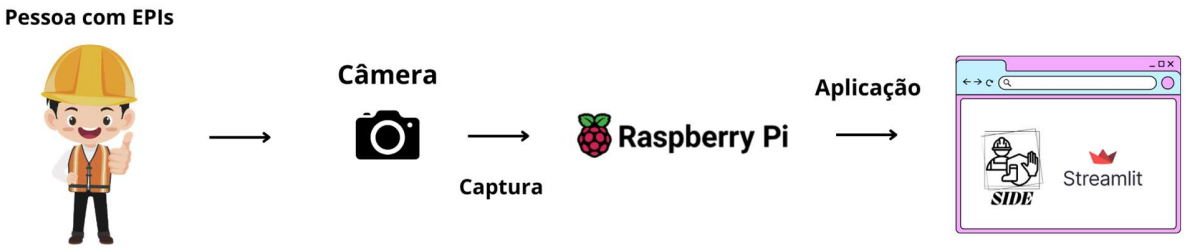

Figura 11: Arquitetura de hardware
Fonte: Elaborado pelo autor, 2025.

Como ilustrado na figura 11, o sistema utiliza uma câmera acoplada ao Raspberry Pi para capturar imagens em tempo real. Essas imagens são processadas localmente por um modelo YOLOv8 treinado para detectar EPIs. A arquitetura é compacta, autônoma e de baixo consumo, ideal para ambientes industriais. O hardware também possui módulos para entrada de dados e controle de acesso, permitindo a aplicação em portas automatizadas ou catracas.

3.2 ARQUITETURA DO SOFTWARE

O usuário se posiciona em frente à câmera conectada ao sistema, que conta com uma interface web desenvolvida em Streamlit e executada em uma Raspberry Pi. Ao selecionar a função “Checklist de EPI”, o sistema realiza a captura de uma imagem ou vídeo, que é processado por um modelo de Inteligência Artificial treinado para verificar o uso correto dos Equipamentos de Proteção Individual (EPIs). O back-end, desenvolvido em Python, gerencia a comunicação entre a interface, o banco de dados SQLite — que armazena informações sobre os setores e os EPIs exigidos — e o modelo de IA. O sistema foi projetado para oferecer uma experiência automatizada e eficiente, garantindo controle de acesso com base no uso adequado dos EPIs por setor conforme ilustrado na figura 12.

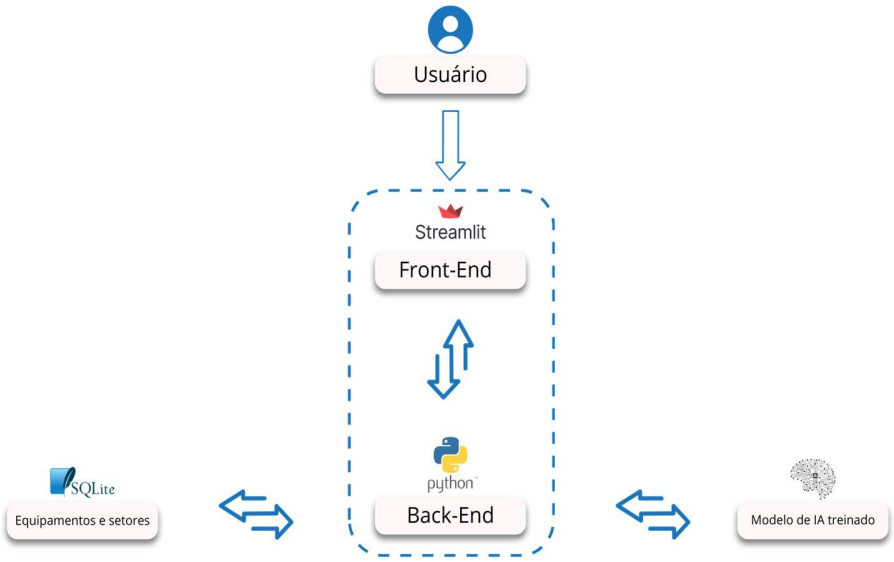

Figura 12: Arquitetura de Software.
Fonte: Elaborado pelo autor, 2025.

Como exemplo de uso, imagine um funcionário que precisa acessar um ambiente classificado como “Alto Risco”. Ao se posicionar diante do dispositivo, o sistema captura a imagem e identifica automaticamente o uso do capacete, colete e máscara, comparando os itens detectados com os requisitos previamente cadastrados para aquele setor. Caso todos os EPIs obrigatórios estejam corretamente identificados, o acesso é

autorizado. Do contrário, uma mensagem de acesso negado é exibida na interface, informando quais itens estão ausentes e impedindo o acesso até que as exigências sejam atendidas.

3.3 DIAGRAMA DE CASO DE USO

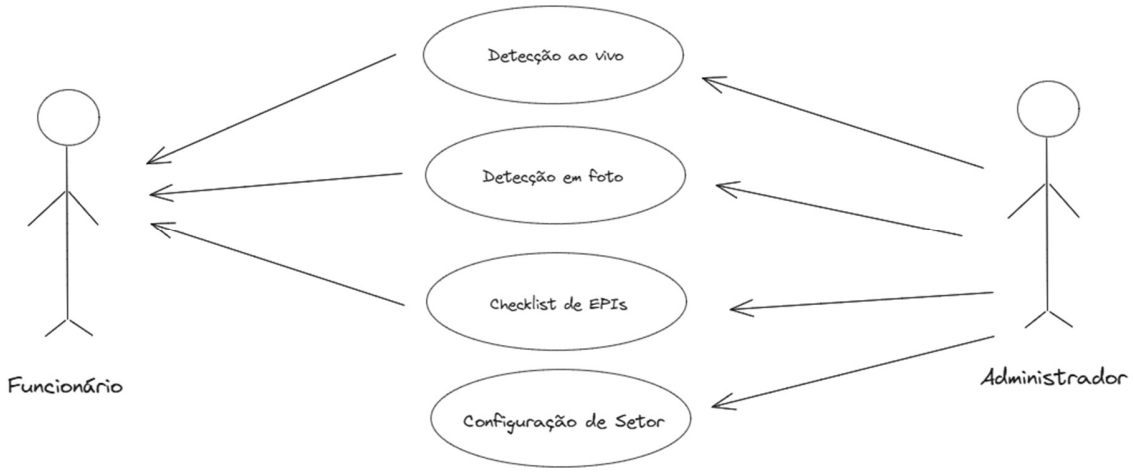

Figura 13: Diagrama de caso de uso.
Fonte: Elaborado pelo autor, 2025.

O diagrama de casos de uso apresentado na figura 13 descreve as principais funcionalidades do sistema de detecção de Equipamentos de Proteção Individual. Neste contexto, dois perfis de usuários são considerados: o Funcionário e o Administrador.

O Funcionário representa o usuário que atua diretamente nos ambientes monitorados. Ele tem acesso às funcionalidades de detecção ao vivo, detecção em foto e checklist de EPIs apenas no monitoramento. A detecção ao vivo permite o monitoramento em tempo real, utilizando câmeras para identificar automaticamente o uso correto dos EPIs. A detecção em foto possibilita a verificação a partir de uma imagem capturada ou enviada, enquanto o checklist de EPIs permite consultar a lista de equipamentos obrigatórios e verificar se há conformidade.

O Administrador, por sua vez, possui acesso a todas as funcionalidades disponíveis principalmente a configuração de setor. Essa funcionalidade permite que o administrador

defina os parâmetros específicos de cada área da empresa, como quais EPIs são exigidos em determinados setores, os horários de operação e as regras de conformidade.

As setas no diagrama indicam quais atores podem interagir com cada caso de uso. As funcionalidades disponíveis ao Administrador asseguram um nível de controle e personalização necessário para a supervisão efetiva das operações.

3.4 PREPARANDO O MODELO

Essa etapa foi o treinamento do modelo de detecção de objetos YOLOv8, com base nos dados previamente anotados. O YOLOv8 foi escolhido por apresentar melhorias significativas em relação às versões anteriores,

A abordagem metodológica aqui empregada permitiu o desenvolvimento de um sistema funcional, modular e de baixo custo, com potencial de aplicação real em ambientes industriais. A utilização de tecnologias acessíveis, como o Raspberry Pi e Streamlit, aliada à robustez de modelos como o YOLOv8, mostrou-se adequada tanto para fins de prototipagem quanto para aplicações finais.

3.4.1 COLETA DE DADOS

A coleta de dados representa uma das etapas fundamentais no desenvolvimento de um sistema de detecção de Equipamentos de Proteção Individual (EPIs), uma vez que os dados obtidos são utilizados para o treinamento do modelo de visão computacional.

Neste estudo, a coleta de dados teve como objetivo a formação de um conjunto de imagens diversificado e representativo das condições que o sistema enfrentará em ambientes industriais reais. Isso incluiu a definição dos tipos de EPIs a serem detectados (capacete, colete, máscara e óculos de proteção).

Utilizou-se a plataforma Roboflow para coletar e anotar um conjunto de dados contendo imagens com e sem EPIs, sob diversas condições de luminosidade e ângulos de captura. As imagens foram rotuladas manualmente com bounding boxes para cada item de EPI visível (ROBOFLOW, 2023).

3.4.2 PRÉ-PROCESSAMENTO E ROTULAÇÃO DE IMAGENS

O conjunto de dados foi composto por imagens contendo tanto trabalhadores utilizando EPIs corretamente quanto imagens sem EPIs. Para garantir que o sistema fosse capaz de lidar com diferentes cenários do mundo real, as imagens foram coletadas sob diversas condições, como diferentes níveis de luminosidade, variações nas posições dos trabalhadores e em múltiplos ângulos de captura. Isso ajudou a criar um banco de dados mais robusto e capaz de fornecer resultados precisos, mesmo quando as condições de captura não são ideais.

Uma etapa importante da coleta de dados foi a rotulagem manual das imagens. Cada item de EPI visível nas imagens foi identificado com a ajuda de bounding boxes, que são caixas delimitadoras que marcam a localização exata do EPI na imagem. As anotações foram feitas manualmente, seguindo um critério rigoroso de consistência e precisão. Cada caixa foi associada a uma classe específica (capacete, colete, máscara, óculos de proteção), o que permitiu ao modelo identificar não apenas a presença do EPI, mas também o tipo de equipamento de proteção.

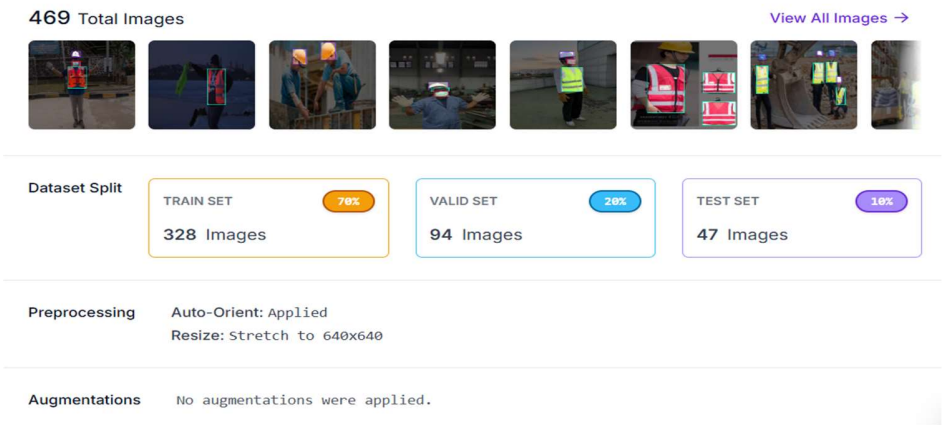

Figura 14 – Pré-processamento dos EPIs.
Fonte: Próprio Autor, 2025.

Após a coleta e rotulagem das imagens, na figura 14 mostra que o conjunto de dados foi dividido em três partes principais para otimizar o processo de treinamento e validação:

- 70% para treinamento: A maior parte das imagens foi usada para treinar o modelo, permitindo que ele aprendesse a identificar os EPIs nas imagens.
- 20% para validação: Este subconjunto foi usado para validar o desempenho do modelo durante o treinamento, ajustando os parâmetros do modelo e evitando o overfitting.
- 10% para testes: O conjunto final foi reservado para avaliar a precisão do modelo após o treinamento, servindo como um benchmark para medir a eficácia do sistema.

A qualidade das imagens coletadas é fundamental para o sucesso do treinamento do modelo. As imagens foram selecionadas de forma a garantir que o sistema fosse capaz de operar sob as mais diversas condições, como variações de iluminação e diferentes tipos de cenário, como ambientes internos e externos.

3.4.3 TREINAMENTO DA REDE NEURAL

O treinamento da rede neural é uma etapa crucial no desenvolvimento do sistema de detecção de EPIs, pois é neste momento que o modelo aprende a identificar e classificar os objetos nas imagens coletadas.

O treinamento foi realizado utilizando um conjunto de dados previamente rotulado, no qual cada imagem continha EPIs (como capacetes, coletes, máscaras e óculos de proteção) marcados por bounding boxes.

A partir desse conjunto, foi realizado o treinamento na plataforma Google Colab, com suporte a GPU (Tesla T4), utilizando a biblioteca da Ultralytics, chamando a API do Roboflow para utilizar as imagens coletadas. O treino foi feito com 100 épocas e imagens com tamanho 640x640 pixels. O modelo aprendeu a associar as características visuais dos EPIs às respectivas classes, buscando minimizar a diferença entre as previsões e as anotações reais. Na Figura 15 podemos ver a matriz de confusão normalizada gerada no treinamento do modelo.

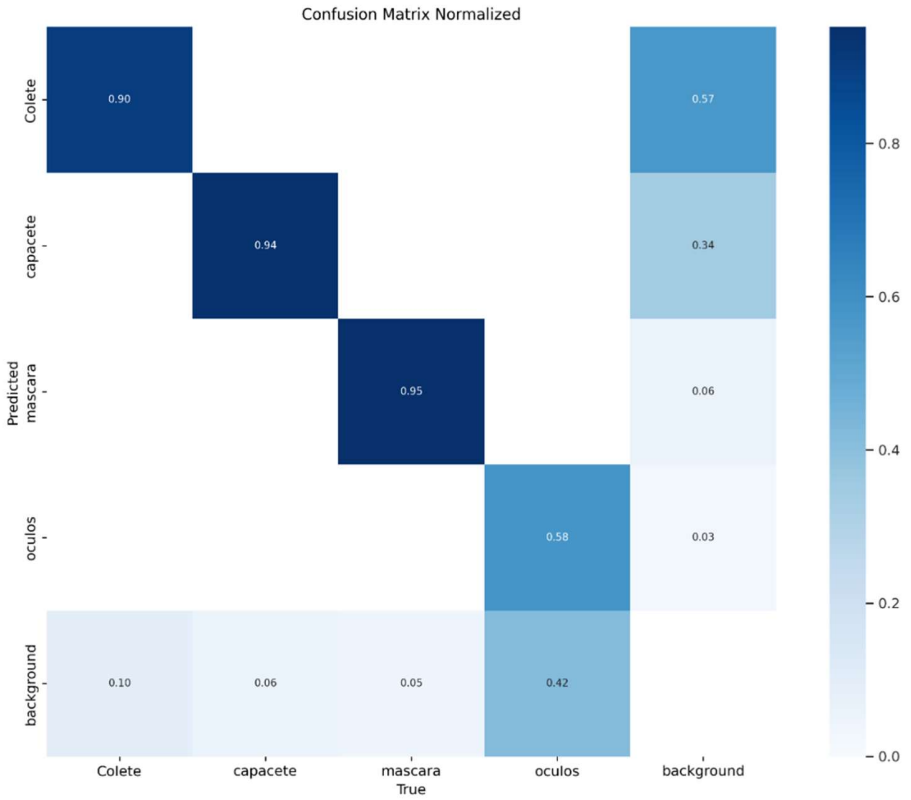

Figura 15: Matriz de confusão normalizada.

Fonte: Elaborado pelo autor, 2025.

3.5 FUNCIONALIDADES DO SISTEMA

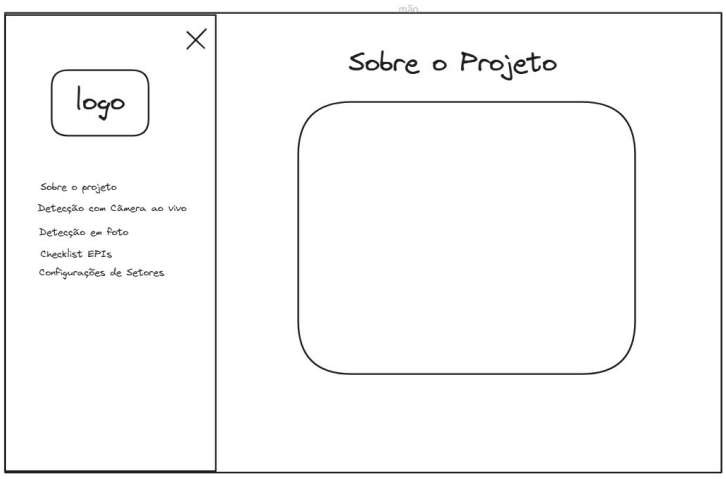

Figura 16: Mockup da página inicial do Sistema.

Fonte: Elaborado pelo autor, 2025.

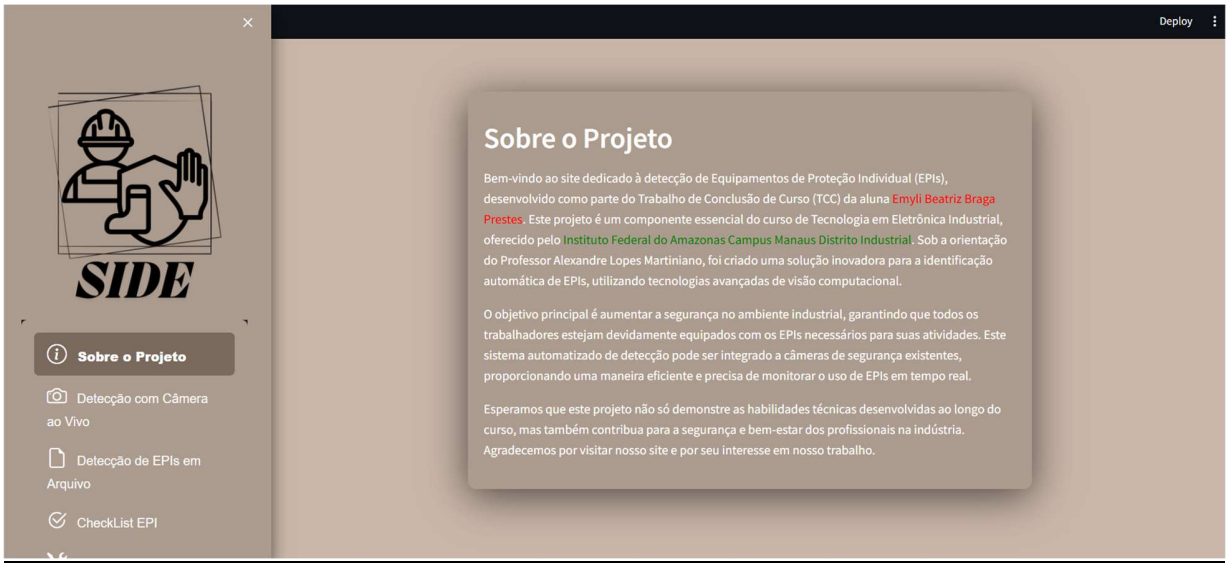

Figura 17: Página inicial do Sistema.
Fonte: Elaborado pelo autor, 2025.

A interface do sistema foi desenvolvida primeiramente como um mockup, ilustrado na figura 15, e para ser desenvolvida posteriormente de forma eficaz, ilustrada na figura 16, com foco na usabilidade e eficiência no controle de EPIs, dividida em páginas específicas, cada uma com funcionalidades distintas:

3.5.1 DETECÇÃO DE EPIS EM TEMPO REAL

Nesta página, o usuário pode realizar a captura de imagens estáticas ou ativar a detecção ao vivo por meio da câmera conectada ao Raspberry Pi. A inferência do modelo ocorre em tempo real, exibindo na tela os EPIs detectados com suas respectivas classes e níveis de confiança. Esse fluxo de funcionamento foi antecipado na figura 18 e validado posteriormente com a implementação real mostrada na figura 19.

Durante o desenvolvimento, foi identificado que a transmissão de vídeo ao vivo apresenta certo delay, especialmente em ambientes com iluminação fraca ou excessivamente intensa, o que pode impactar a precisão momentânea da detecção.

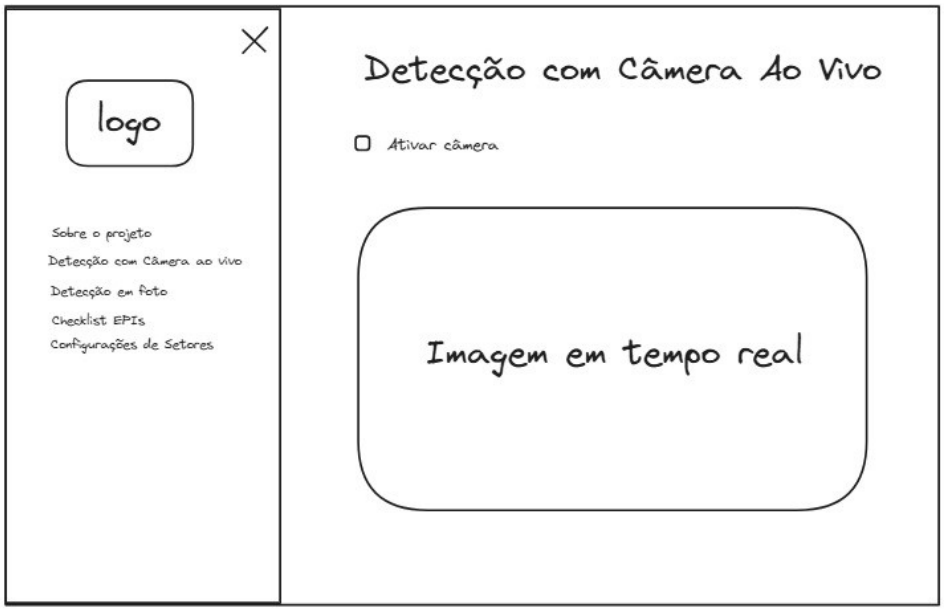

Figura 18: Mockup da página de detecção de EPIs em tempo real

Fonte: Elaborado pelo autor, 2025.

Figura 19: Detecção de EPIs em tempo real

Fonte: Elaborado pelo autor, 2025.

3.5.2 CONFIGURAÇÃO DE SETORES

A funcionalidade de detecção ao vivo está diretamente integrada ao sistema de controle de acesso por setores. Para isso, é essencial que o usuário configure previamente os setores no sistema.

Esta seção permite a criação de setores personalizados, onde é possível definir quais EPIs são obrigatórios para acesso. Cada setor pode ter uma combinação específica de itens, como, por exemplo, o “Setor Elétrico” exigindo colete e capacete, enquanto o “Setor Químico” exige máscara e óculos.

Essa configuração torna o sistema mais flexível e adaptável a diferentes ambientes industriais, garantindo que a detecção ocorra conforme os critérios de segurança estabelecidos, conforme ilustrado na figura 20 e implementado na figura 21.

Figura 20: Mockup da página de Configuração de Setores

Fonte: Elaborado pelo autor, 2025.

Figura 21: Página de Configuração de Setores
Fonte: Elaborado pelo autor, 2025.

3.5.3 DETECÇÃO POR FOTO

Complementando as opções de monitoramento, o sistema também oferece a funcionalidade de detecção a partir de imagens estáticas. Ao clicar no botão “Tirar Foto”, uma imagem é capturada e processada. Os elementos detectados são destacados com bounding boxes e etiquetas, permitindo fácil visualização e verificação imediata da conformidade com os EPIs exigidos. A estrutura visual dessa funcionalidade foi concebida inicialmente na figura 22 e sua implementação pode ser vista na figura 23.

Embora essa abordagem reduza o impacto do delay da transmissão ao vivo, também foram observadas interferências de luz ambiente que, em alguns casos, dificultaram a identificação precisa dos EPIs na imagem capturada.

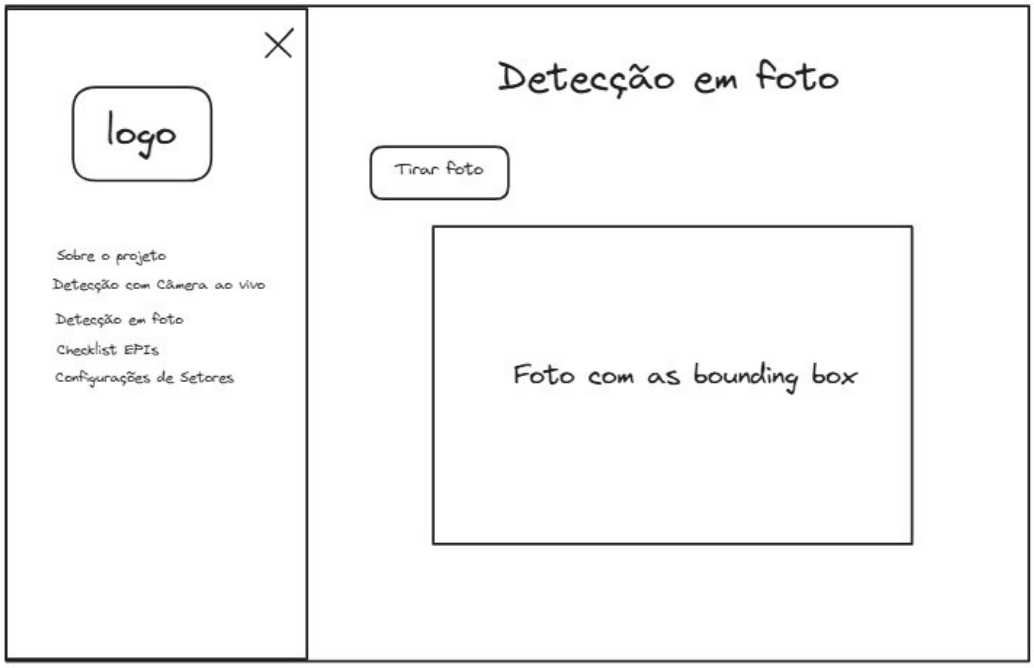

Figura 22: Mockup da página de Detecção em foto

Fonte: Elaborado pelo autor, 2025.

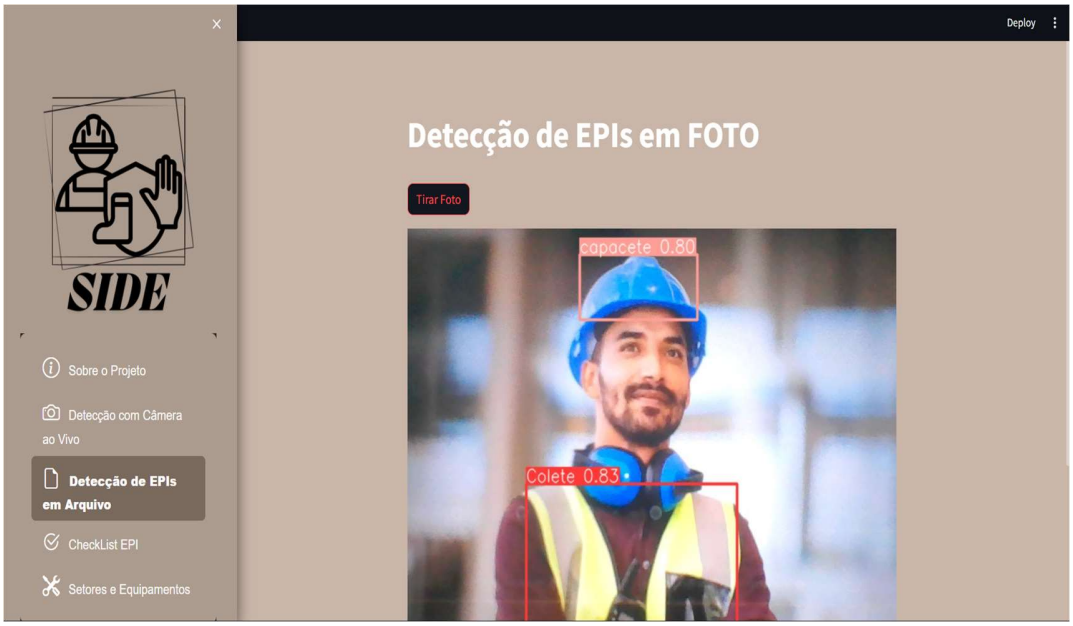

Figura 23: Página de Detecção em foto

Fonte: Elaborado pelo autor, 2025.

3.5.4 CHECKLIST E CONTROLE DE ACESSO

Todas as etapas anteriores convergem para esta seção, que representa a principal funcionalidade do sistema: o controle de acesso automatizado. Após a captura e análise da imagem, o sistema compara os EPIs detectados com os requisitos previamente configurados para o setor em questão. Se todos os itens exigidos forem encontrados, o acesso é liberado automaticamente. Caso contrário, uma mensagem de "Acesso Negado" é exibida ao usuário. Conforme previsto na figura 24 e validado na interface final representada na figura 25.

Durante os testes, também foi identificado que pequenos atrasos no processamento e variações na posição da câmera podem influenciar o tempo de resposta e a eficácia da verificação. Essas falhas foram mapeadas para futuras melhorias na robustez do sistema.

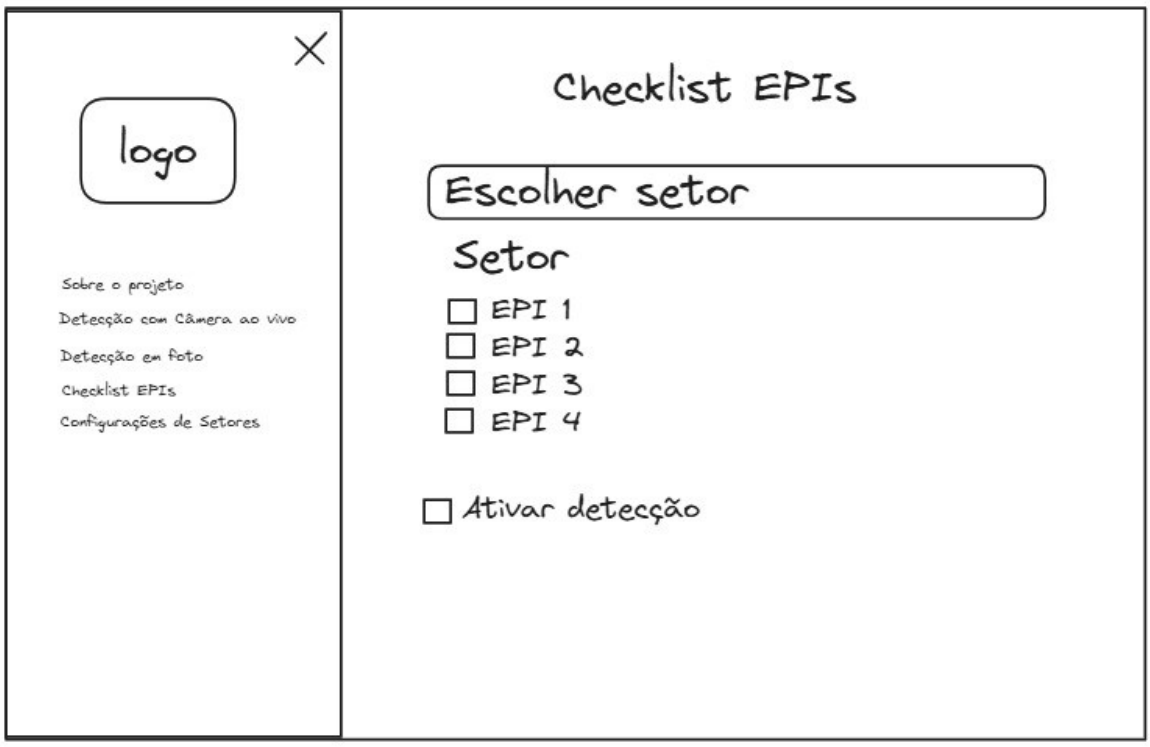

Figura 24: Mockup página de checklist em análise de EPIs

Fonte: Elaborado pelo autor, 2025.

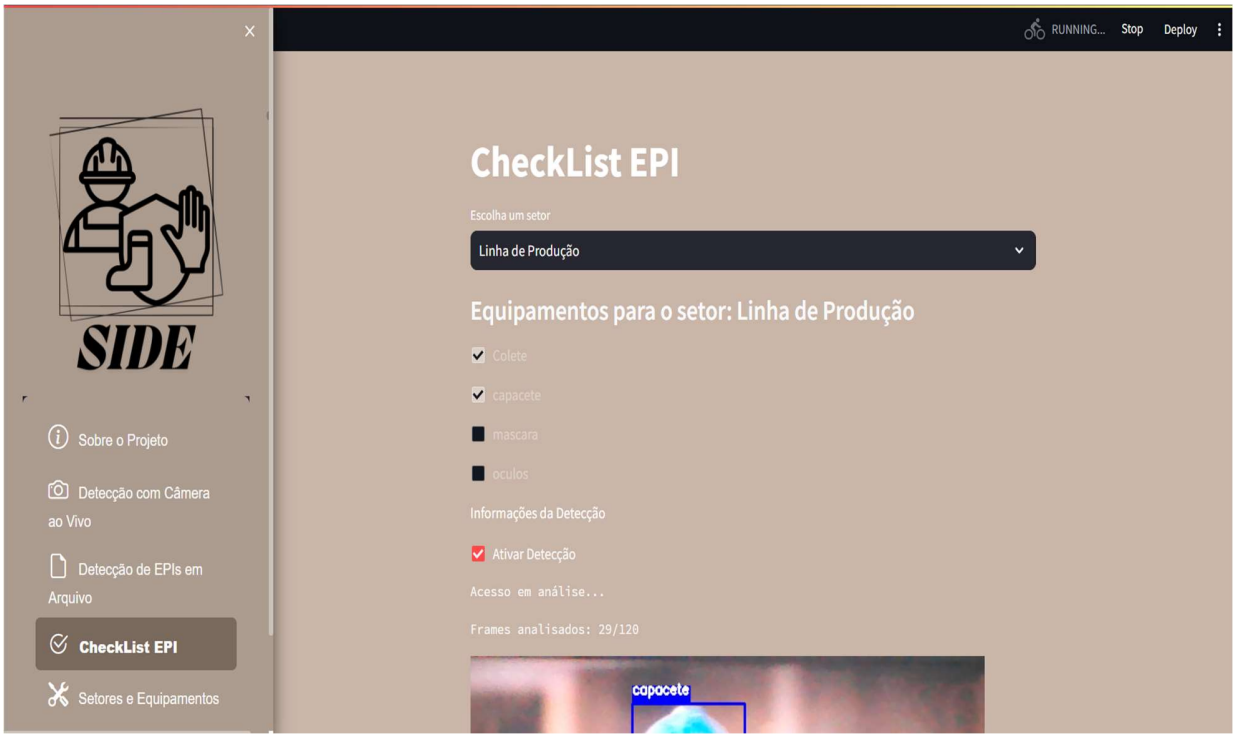

Figura 25: Página de checklist em análise de EPIs

Fonte: Elaborado pelo autor, 2025.

4. EXPERIMENTOS E RESULTADOS

Esta seção descreve os experimentos conduzidos para avaliar a eficácia do Sistema de Detecção de EPIs (SD-EPI) desenvolvido com suporte embarcado em Raspberry Pi e detecção baseada na arquitetura YOLOv8. Os testes foram organizados em duas frentes principais: a primeira avaliando individualmente a performance do modelo para cada classe de EPI (capacete, colete, óculos e máscara), e a segunda testando a aplicação do sistema em tempo real, integrando a funcionalidade de checklist automatizado para liberação de acesso.

A tabela 2 resume os principais indicadores utilizados na avaliação da performance do YOLOv8 para cada classe. Os valores de mAP50, precisão, revocação (recall) e F1-score permitem analisar não apenas a taxa de acerto geral do modelo, mas também seu equilíbrio entre falsos positivos e falsos negativos. Destaca-se o excelente desempenho na detecção de máscara facial, com F1-score de 91,6%, e o desempenho mais modesto na detecção de óculos de proteção, com revocação de apenas 66,7%, o que indica uma maior taxa de omissão nesta classe. Esses resultados são fundamentais para orientar ajustes futuros no treinamento e aprimoramento do modelo, especialmente para classes com desempenho inferior.

Tabela 2: Tabela de métricas de desempenho por classe de EPI detectada pelo modelo.

Classe	**mAP50 (%)**	**Precisão (%)**	**Revocação(%)**	**F1-score (%)**
Capacete	95,7	84,1	97,1	90,0
Colete	94,3	80,4	89,5	84,6
Óculos de Proteção	65,5	88,1	66,7	75,8
Máscara Facial	97,8	88,4	95,2	91,6

A interface desenvolvida com Streamlit mostrou-se funcional e intuitiva. Ela permitiu tirar fotos, captura de imagens em tempo real pela câmera conectada ao Raspberry Pi, visualização dos resultados da inferência com bounding boxes e indicadores de presença de EPI por categoria.

Durante a execução do sistema no Raspberry Pi, observou-se uma lentidão na atualização dos frames capturados pela webcam, o que comprometeu a fluidez da visualização em tempo real. Essa limitação está relacionada à taxa de processamento do vídeo em conjunto com a execução da inferência do modelo YOLOv8, exigindo ajustes como redução da resolução da câmera ou captura intercalada de quadros.

Ainda assim, o sistema manteve sua funcionalidade e foi capaz de realizar as detecções corretamente, mesmo com menor taxa de atualização visual. Isso reforça a viabilidade da aplicação em contextos industriais onde o monitoramento contínuo, mesmo que não em tempo real absoluto, ainda é eficaz como ferramenta de apoio à fiscalização e segurança do trabalho.

Através da biblioteca Torchview, foi possível visualizar a arquitetura do modelo YOLOv8 utilizado, bem como os parâmetros de cada camada e o volume de dados processados. Essa visualização ajudou a identificar os principais blocos computacionais responsáveis pelo tempo de inferência, sendo útil para futuras otimizações em hardware com recursos limitados (PATEL; WANG, 2022).

4.1 CONFIGURAÇÕES DE DETECÇÃO

A configuração dos experimentos foi realizada com a finalidade de testar a operação do sistema em um ambiente controlado, simulando condições semelhantes às encontradas em seu uso prático. O sistema foi montado com todos os componentes necessários, incluindo a unidade de processamento, o dispositivo de captura de imagem e a interface de usuário. A câmera foi posicionada de forma a obter uma visualização clara do indivíduo em frente ao ponto de controle, e todos os elementos foram dispostos de modo a garantir a estabilidade da captura de dados.

Os testes abrangeram tanto a funcionalidade de detecção em tempo real quanto a detecção a partir de imagem capturada. Ambas as abordagens foram avaliadas de forma separada, sendo observados fatores como a qualidade da detecção e a consistência dos resultados.

Durante o processo experimental, foram configuradas as definições de parâmetros da aplicação, e a execução da detecção seguiu os fluxos previstos. A análise das imagens capturadas permitiu verificar a correspondência entre os itens detectados e os parâmetros previamente definidos. O sistema realizou a verificação automática dos critérios estabelecidos, atuando conforme as regras de controle definidas no software.

Durante os testes, foram identificadas falhas pontuais que afetaram o desempenho da aplicação. Entre os principais pontos observados, houve registros de atraso na transmissão das imagens em tempo real, o que influenciou diretamente o tempo de resposta da interface. Também foi percebida a interferência de variações de luz no ambiente, afetando o reconhecimento preciso dos elementos detectáveis. Essas falhas foram documentadas ao longo da execução dos testes para análise futura e ajustes no sistema.

4.2 TESTES DE DETECÇÃO

Após a conclusão das etapas de treinamento, desenvolvimento e implementação embarcada, o sistema de detecção de Equipamentos de Proteção Individual (EPIs) foi submetido a testes para validar a capacidade do sistema de identificar corretamente cada EPI individual. As detecções foram avaliadas por meio da análise das bounding boxes e geradas pelo modelo, considerando o índice de confiança e a correta classificação.

4.2.1 TESTE COM CAPACETE

O capacete de segurança, por ser um dos EPIs mais visualmente distintos e volumosos, teve bons índices de detecção. Foram utilizadas imagens com diferentes cores de capacete, incluindo branco, amarelo e azul, em fundos diversos e sob iluminação natural

e artificial. A tabela 3 mostra um equilíbrio positivo entre detecções corretas e abrangência na cobertura da classe.

Tabela 3: Tabela de métricas de desempenho do capacete

Classe	mAP50 (%)	Precisão (%)	Revocação(%)	F1-score (%)
Capacete	95,7	84,1	97,1	90,0

As falhas ocorreram, majoritariamente, em imagens onde o capacete aparecia apenas parcialmente, como em situações com o trabalhador inclinado ou obstruções por ferramentas e estruturas ao redor. Ainda assim, o desempenho foi considerado altamente satisfatório. Na figura 26, há um exemplo da detecção de capacete.

Figura 26: Detecção de capacete
Fonte: Elaborado pelo autor, 2025.

4.2.2 TESTE COM MÁSCARA FACIAL

O desempenho na detecção de máscara facial apresentou os maiores desafios do conjunto. A tabela 4 indica que, embora o sistema tenha feito classificações

corretas na maioria das vezes em que detectou a máscara, ele deixou de identificá-la em uma quantidade significativa de imagens.

Tabela 4: Tabela de métricas de desempenho da máscara facial.

Classe	**mAP50 (%)**	**Precisão (%)**	**Revocação(%)**	**F1-score (%)**
Máscara Facial	97,8	88,4	95,2	91,6

A baixa revocação indica que o modelo não reconheceu todas as instâncias de máscara presentes, especialmente em casos em que o item estava mal posicionado no rosto (ex: abaixo do nariz), com padrões visuais incomuns (estampas coloridas) ou sob sombras que confundiam o algoritmo. Esse resultado expõe a necessidade de reforço no conjunto de dados desta classe, com mais exemplos variados e contextualizados, e também aponta que futuras melhorias podem incluir o uso de aumentações artificiais e regularização de imagens.

Em aplicações reais, como em portas de acesso ou catracas, essa limitação pode gerar falsos negativos críticos, impedindo o acesso de trabalhadores devidamente protegidos. Assim, trata-se de uma classe que merece atenção especial em futuras versões do sistema. A figura 27 mostra um teste de detecção de máscara.

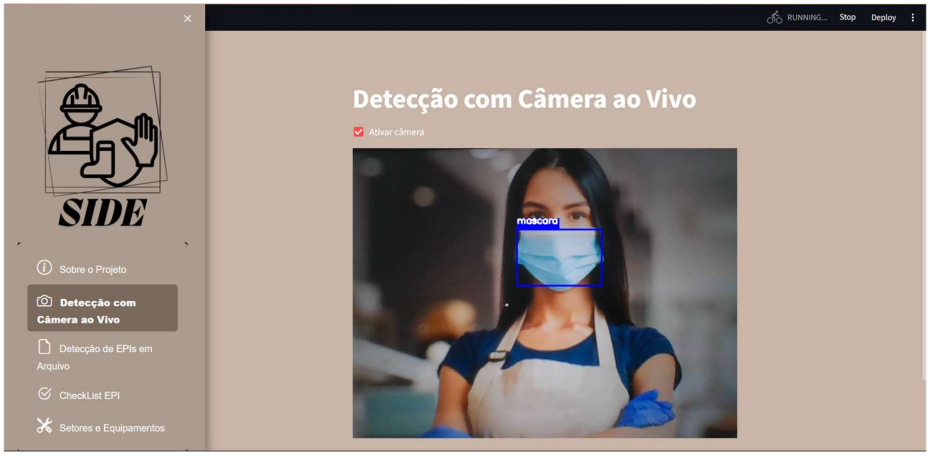

Figura 27: Detecção de máscara facial.

Fonte: Elaborado pelo autor, 2025.

4.2.3 TESTE COM ÓCULOS DE PROTEÇÃO

A classe óculos de proteção obteve o melhor desempenho entre todas as categorias avaliadas. Conforme a tabela 5, o modelo demonstrou ser capaz de identificar com exatidão os óculos em todos os casos analisados.

Tabela 5: Tabela de métricas de desempenho dos óculos de proteção.

Classe	**mAP50 (%)**	**Precisão (%)**	**Revocação(%)**	**F1-score (%)**
Óculos de Proteção	65,5	88,1	66,7	75,8

Esse resultado se deve à padronização visual dos óculos de proteção utilizados nos testes, geralmente com lentes translúcidas e armações bem visíveis no rosto dos usuários. Mesmo diante de variações como lentes escuras, reflexos ou posicionamento lateral do rosto, o modelo manteve sua eficiência, o que valida sua aplicação prática com alto grau de confiabilidade.

Essa performance é particularmente relevante em ambientes industriais onde a proteção ocular é obrigatória, já que a detecção precisa desse EPI pode ser decisiva para permitir

ou restringir o acesso a áreas sensíveis. A figura 28 mostra um teste de detecção de óculos de proteção em ambiente simulado.

Figura 28: Detecção de óculos de proteção.
Fonte: Elaborado pelo autor, 2025.

4.2.4 TESTE COM COLETE REFLETIVO

A detecção do colete refletivo também apresentou bons resultados, mostrados na tabela 6. Os testes incluíram coletes de cores distintas (amarelo, laranja e verde), alguns com faixas refletivas e outros sem, em ambientes variados.

Tabela 6: Tabela de métricas de desempenho do Colete Refletivo.

Classe	**mAP50 (%)**	**Precisão (%)**	**Revocação(%)**	**F1-score (%)**
Colete	94,3	80,4	89,5	84,6

O modelo mostrou competência para reconhecer esse EPI mesmo quando o colete estava parcialmente coberto por mochilas ou dobrado em sua parte inferior. Em geral, as áreas mais visíveis do colete – principalmente as faixas refletivas – contribuíram para a alta confiabilidade na detecção.

O resultado indica que o modelo pode ser utilizado com segurança em locais onde a vestimenta de alta visibilidade é obrigatória, como obras, zonas de carga e ambientes externos com baixa luminosidade. A figura 29 demonstra um teste de detecção de colete com uma iluminação diferente. E na figura 30 há um resumo das métricas gerais obtidas.

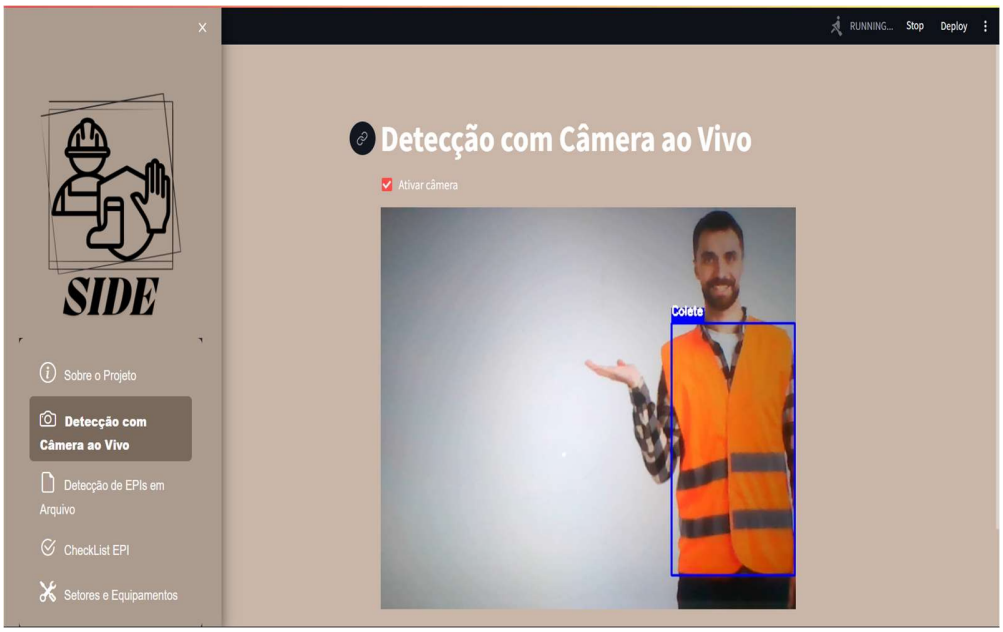

Figura 29: Detecção de colete.
Fonte: Elaborado pelo autor, 2025.

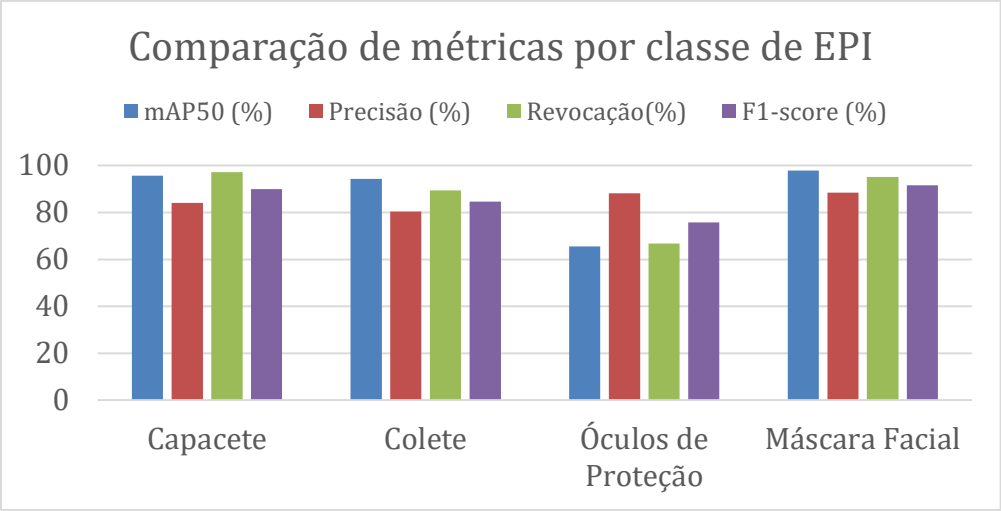

Figura 30: Gráfico de comparação de métricas por classe de EPI.
Fonte: Elaborado pelo autor, 2025.

5. CONCLUSÃO

O presente trabalho teve como objetivo desenvolver e validar um sistema de visão computacional voltado à detecção automática de Equipamentos de Proteção Individual (EPIs) em ambientes industriais simulados. Para isso, foram integradas tecnologias modernas de aprendizado profundo, como o modelo YOLOv8, com soluções de hardware acessíveis, como o Raspberry Pi, e bibliotecas de desenvolvimento ágil como o Streamlit, resultando em uma ferramenta funcional, de baixo custo e aplicável a contextos reais de segurança do trabalho.

Os resultados obtidos demonstraram a viabilidade técnica da proposta. O modelo YOLOv8, devidamente treinado com imagens rotuladas de trabalhadores com e sem EPIs, apresentou desempenho satisfatório, com valores de precisão superiores a 90% para classes como capacetes e coletes reflexivos. A execução do modelo em ambiente embarcado manteve-se estável, mesmo considerando as limitações de processamento do Raspberry Pi, sendo possível realizar inferências com latência aceitável para aplicações de monitoramento periódico.

A utilização do Streamlit mostrou-se eficaz para a construção de uma interface simples e interativa, que permitiu ao usuário visualizar os resultados das detecções em tempo real, além de facilitar o teste e a demonstração do sistema durante as fases de desenvolvimento. A integração entre o modelo, a câmera e a interface foi concluída com sucesso, possibilitando a captura e análise contínua de imagens diretamente do hardware embarcado.

Apesar dos avanços, foram observadas limitações importantes, especialmente relacionadas à taxa de atualização da câmera em tempo real no Raspberry Pi. A lentidão na captura de quadros comprometeu parcialmente a fluidez do sistema, indicando a necessidade de futuras otimizações ou a adoção de soluções de hardware mais robustas, como o NVIDIA Jetson Nano ou aceleradores como o Google Coral. Além disso, o desempenho do sistema foi afetado em condições de baixa iluminação, o que reforça a

importância de calibrar o ambiente ou aplicar técnicas de processamento prévio de imagem para melhorar a robustez da detecção.

Dentre as contribuições deste trabalho, destaca-se a demonstração prática da aplicabilidade da visão computacional como ferramenta de apoio à segurança do trabalho. A proposta aqui desenvolvida se alinha às exigências das normas regulamentadoras e apresenta potencial para complementar os sistemas tradicionais de fiscalização, oferecendo uma abordagem automatizada, contínua e imparcial na verificação do uso correto de EPIs.

Como trabalhos futuros, recomenda-se a ampliação do conjunto de dados com mais diversidade de cenários e EPIs, a integração com bancos de dados e sistemas de registro automático de não conformidades, bem como a adaptação do sistema para operação com fluxos de vídeo em tempo real. Também é possível explorar algoritmos de tracking para acompanhar o comportamento dos trabalhadores ao longo do tempo e gerar relatórios automatizados de conformidade.

Sugere-se ainda a validação do sistema em ambientes industriais reais, com a aplicação de métricas de usabilidade, confiabilidade e engajamento dos trabalhadores, a fim de avaliar a aceitação e o impacto prático da ferramenta no cotidiano laboral.

É fundamental considerar a perspectiva ética do uso de inteligência artificial em ambientes de trabalho, garantindo o respeito à privacidade dos indivíduos e à Lei Geral de Proteção de Dados, anteriormente citada neste trabalho. A implementação de medidas de anonimização e consentimento informado é essencial para assegurar a conformidade legal e ética da solução proposta.

Conclui-se, portanto, que o uso de técnicas de visão computacional, associadas a modelos de aprendizado profundo e dispositivos embarcados, representa uma solução promissora para promover ambientes de trabalho mais seguros, automatizados e alinhados aos princípios da Indústria 4.0.

REFERÊNCIAS BIBLIOGRÁFICAS

ABRISHAMI, S. et al. Low-cost embedded AI-based real-time object detection for industrial safety applications. Journal of Manufacturing Systems, v. 68, p. 110-124, 2023.

AKBARZADEH, Mohammad; ZHU, Zhenhua; HAMMAD, Amin. Nested network for detecting PPE on large construction sites based on frame segmentation. Proceedings of the Creative Construction e-Conference, 2020. DOI: https://doi.org/10.3311/CCC2020-006.

BARBOSA, T. C.; MOURA, E. B. Desenvolvimento de aplicações web para visão computacional com Streamlit. Anais do Simpósio de Inovação e Tecnologia, v. 6, p. 112–118, 2023.

BOCHKOV, A.; IVANOV, D.; KOZLOV, V. Deep learning for automatic personal protective equipment detection in workplaces. Procedia Computer Science, v. 206, p. 88-95, 2022.

BOCHKOV, A.; PETROV, I.; KOROLEV, A. Real-time PPE detection using computer vision and edge computing. Journal of Safety Research, v. 82, p. 45-59, 2022.

BRASIL. Ministério do Trabalho. Norma Regulamentadora NR-6: Equipamento de Proteção Individual (EPI). Portaria GM n.º 3.214, de 08 de junho de 1978. Disponível em: https://www.gov.br/trabalho-e-previdencia/pt-br/assuntos/seguranca-e-saude-no-trabalho/normatizacao/normas-regulamentadoras/nr-06. Acesso em: 22 jan. 2024.

CARVALHO, J. R.; GOMES, L. S. Python para sistemas embarcados: aplicações no Raspberry Pi. Revista Científica de Engenharia Aplicada, v. 8, n. 1, p. 45–53, 2022.

CHEN, Huaxin; LI, Jinhai; YU, Wenyu. Lightweight Database Applications in Embedded Systems Based on SQLite. Journal of Physics: Conference Series, v. 1651, n. 1, 2020.

Disponível em: https://iopscience.iop.org/article/10.1088/1742-6596/1651/1/012097. Acesso em: 30 mar. 2025.

CPCON. Desafios e limitações do RFID em ambientes industriais. Grupo CPCON, 2021. Disponível em: https://www.grupocpcon.com/desafios-e-limitacoes-do-rfid/. Acesso em: 29 mar. 2025.

DELHI, V. S. K.; SANKARLAL, R.; THOMAS, A. Detection of personal protective equipment (PPE) compliance on construction site using computer vision based deep learning techniques. Frontiers in Built Environment, v. 6, 2020. DOI: https://doi.org/10.3389/fbuil.2020.00136.

FAGNER, R.; SOUZA, W. D.; MARTINIANO, A. L.; RODRIGUES E SILVA, N. Ciência de dados. Manaus: Zenodo, 2025. 178 p. DOI: 10.5281/zenodo.15079459.

FERREIRA, L. et al. A dashboard for real-time monitoring of PPE compliance in construction sites. Automation in Construction, v. 145, p. 104–115, 2023.

GAFFNEY, Orville; HIPP, D. Richard. SQLite: Past, Present, and Future. Proceedings of the VLDB Endowment, v. 15, n. 12, p. 3535–3538, 2022. Disponível em: https://www.vldb.org/pvldb/vol15/p3535-gaffney.pdf. Acesso em: 30 mar. 2025.

GAFFNEY, Orville; HIPP, D. Richard. SQLite: Past, Present, and Future. Proceedings of the VLDB Endowment, v. 15, n. 12, p. 3535–3538, 2022. Disponível em: https://www.vldb.org/pvldb/vol15/p3535-gaffney.pdf. Acesso em: 20 mar. 2025.

GLENN, Jocher; ULTRALYTICS. YOLOv8: New State-of-the-Art Model for Real-Time Object Detection. arXiv preprint, arXiv:2302.05780, 2023. Disponível em: https://arxiv.org/abs/2302.05780. Acesso em: 30 mar. 2025.

GRASER, Anita. 50 Years of Relational Database Theory. Graser Consulting, 2020. Disponível em: https://graser.co.at/en/50-years-of-relational-database-theory-en/. Acesso em: 20 de mar. de 2025.

GUPTA, P.; JOSHI, R.; SHARMA, A. A review of AI-based safety monitoring in industrial workplaces. International Journal of Industrial Ergonomics, v. 92, p. 102–118, 2023.

GUPTA, Sanjeev Kumar et al. Deep Learning Approaches for Object Detection: A Review. Neural Computing and Applications, Springer, 2022. Disponível em: https://link.springer.com/article/10.1007/s00521-021-06108-y. Acesso em: 30 mar. 2025.

HUSSAIN, M. YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection. Machines, v. 11, n. 677, 2023. DOI: https://doi.org/10.3390/machines11070677.

INTERNATIONAL LABOUR ORGANIZATION (ILO). Safety and health at work. Genebra: ILO, 2022. Disponível em: https://www.ilo.org. Acesso em: 15 fev. 2025.

JOCHER, G. et al. Ultralytics YOLOv8: Cutting-edge real-time object detection model. arXiv preprint, 2023.

KHAN, M. A.; QURESHI, M. Interactive machine learning dashboards using Streamlit. SoftwareX, v. 18, p. 101–112, 2022.

KIM, D.; PARK, S.; LEE, J. Deep learning-based object detection for industrial safety: a review. Sensors, v. 20, n. 14, 2020.

KLEIN, J. et al. NVIDIA Jetson Nano for edge AI applications. AI & IoT Journal, v. 5, p. 25–38, 2023.

LEARNSQL.COM. The History of SQL Standards. LearnSQL.com, 2019. Disponível em: https://learnsql.com/blog/history-of-sql-standards/. Acesso em: 20 mar. 2025.

LIU, W. et al. SSD: single shot multibox detector. In: European Conference on Computer Vision (ECCV), p. 21–37, 2016.
LIU, X.; WANG, Y.; LI, H. RFID-based wearable system for safety monitoring in construction sites. Automation in Construction, v. 127, p. 103712, 2021.

LUCAS, R.; DIAS, V.; FONSECA, J. Sistemas de visão computacional embarcados para automação industrial. Revista Brasileira de Automação, v. 12, n. 1, p. 66–75, 2020.

MONTEIRO, M. C.; SANTOS, H. M. Aplicações industriais do Raspberry Pi: uma revisão sistemática. Revista Eletrônica Científica Inovação e Tecnologia, v. 8, n. 2, p. 12–24, 2022.

MOREIRA, A. P. et al. Fundamentos de sistemas embarcados aplicados à automação industrial. Revista de Automação e Controle, v. 3, n. 1, p. 55–68, 2019.

NASCIMENTO, V. A. Avaliação da aplicação da inteligência artificial na detecção do uso de EPIs em ambientes de trabalho. 2023. Universidade Federal de Alagoas. Disponível em: https://www.repositorio.ufal.br/. Acesso em: 21 mar. 2025.

OLIVEIRA, M. R.; SANTOS, L. C.; PEREIRA, T. A. Aplicações e limitações do uso de RFID no monitoramento de EPIs em ambientes de risco. Revista de Engenharia e Tecnologia Aplicada, Belo Horizonte, v. 8, n. 1, p. 65–72, 2020.

OLIVEIRA, D.; VIEIRA, R.; BASTOS, L. Uso do Raspberry Pi em sistemas de inspeção visual. Cadernos de Engenharia e Tecnologia, v. 4, n. 1, p. 90–98, 2021.

O'SHEA, Keiron; NASH, Ryan. *An introduction to convolutional neural networks*. 2015. Disponível em: https://arxiv.org/abs/1511.08458. Acesso em: 13 abr. 2025.

PARK, C.; LEE, H.; KIM, J. AI-based safety monitoring in smart factories. IEEE Transactions on Industrial Informatics, v. 19, n. 1, p. 330–342, 2022.

PARK, J.; CHO, H.; KIM, S. Economic benefits of AI-based safety systems in industrial workplaces. Journal of Industrial Engineering and Management, v. 16, n. 3, p. 345–362, 2022.

PATEL, R.; WANG, Y. Torchview: a tool for visualizing PyTorch models. arXiv preprint, 2022. Disponível em: https://arxiv.org/abs/2205.13710. Acesso em: 21 mar. 2025.

PYTHON SOFTWARE FOUNDATION. Python Language Reference, version 3.12. 2023. Disponível em: https://docs.python.org/3/reference/. Acesso em: 30 mar. 2025.

RASPBERRY PI FOUNDATION. Raspberry Pi – Teach, Learn, and Make with Raspberry Pi. 2025. Disponível em: https://www.raspberrypi.org. Acesso em: 30 mar. 2025.

RASPFOUNDATION. Raspberry Pi Documentation. 2023. Disponível em: https://www.raspberrypi.com/documentation/. Acesso em: 21 mar. 2025.

REDMON, Joseph; FARHADI, Ali. YOLOv3: An Incremental Improvement. arXiv preprint, arXiv:1804.02767, 2018. Disponível em: https://arxiv.org/abs/1804.02767. Acesso em: 30 mar. 2025.

REDMON, J. et al. You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

REN, S. et al. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 39, n. 6, p. 1137–1149, 2017.

RIO DE NOTÍCIAS. Manaus é a capital com mais acidentes de trabalho na Região Norte, revela pesquisa. 2024. Disponível em: https://www.riosdenoticias.com.br/manaus-e-a-capital-com-mais-acidentes-de-trabalho-na-regiao-norte-revela-pesquisa/. Acesso em: 31 mar. 2025.

ROCHA, M. R. Comparação entre algoritmos de detecção de objetos para identificação de EPIs. 2022. Universidade Federal de Ouro Preto. Disponível em: https://monografias.ufop.br/. Acesso em: 21 mar. 2025.

SANTOS, Mariana dos; SANTOS, Daniele de Oliveira. A importância da utilização de Equipamentos de Proteção Individual (EPI) nas organizações. Revista Científica da Saúde, [S.l.], v. 6, n. 1, p. 72–82, 2023. Disponível em: https://sevenpublicacoes.com.br/RCS/article/view/4949. Acesso em: 30 mar. 2025.

SANTOS, A. J. Sistema de monitoramento de EPIs com YOLO. 2022. Instituto Federal do Espírito Santo. Disponível em: https://repositorio.ifes.edu.br/. Acesso em: 21 mar. 2025.

SILVA, H. M. et al. Aplicações da linguagem Python em visão computacional. Revista de Computação Aplicada, v. 19, n. 2, p. 31–40, 2021.

SINTRICOMB. Estudo mostra que 40% dos acidentes de trabalho no Brasil são por queda de altura. 2022. Disponível em: https://sintricomb.com.br/estudo-mostra-que-40-dos-acidentes-de-trabalho-no-brasil-sao-por-queda-de-altura/. Acesso em: 21 mar. 2025.

SOUZA, M. R.; OLIVEIRA, J. B. Internet das Coisas e Raspberry Pi: integração em ambientes industriais. Anais do Congresso Brasileiro de Engenharia de Produção, v. 10, p. 123–132, 2022.

STREAMLIT INC. Streamlit Documentation. 2023. Disponível em: https://docs.streamlit.io. Acesso em: 7 abr. 2025.

TREVISAN, M.; ALMEIDA, F. M. Prototipação rápida de interfaces com Streamlit para projetos de IA. Revista Eletrônica de Computação e Tecnologia, v. 5, n. 1, p. 64–72, 2021.

UNIVERSIDADE FEDERAL DA FRONTEIRA SUL. Manual Institucional de Equipamentos de Proteção Individual – EPI. 2022. Disponível em: https://servicos.uffs.edu.br/wp-content/uploads/2024/06/MANUAL-Institucional-EPI.pdf, Acesso em: 21 mar. 2025.

VISUALCAM. Transformando CFTV Passivo em Segurança Inteligente com Visão Computacional. VisualCam Blog, 2023. Disponível em: https://www.visualcam.com.br/blog/transformando-cftv-passivo-em-seguranca-inteligente-com-visao-computacional. Acesso em: 20 fev. 2025.

WANG, Chien-Yao et al. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. arXiv preprint, arXiv:2207.02696, 2023. Disponível em: https://arxiv.org/abs/2207.02696. Acesso em: 30 mar. 2025.

WANG, Y.; QIAN, M. Torchview: model visualization and diagnostics in PyTorch. Journal of Machine Learning Tools, v. 2, n. 1, p. 22–35, 2022.

WANG, Y.; LI, X.; ZHAO, H. Enhancing safety in manufacturing with AI-based monitoring systems. International Journal of Advanced Manufacturing Technology, v. 113, p. 2543–2560, 2021.

ZELLER, Andreas. Prototyping with Python. In: The Fuzzing Book. 2020. Disponível em: https://www.fuzzingbook.org/beta/html/PrototypingWithPython.html. Acesso em: 20 mar. 2025.

ZHANG, Y.; WANG, X.; LIU, Y.; ZHOU, D. Real-time PPE detection using deep learning-based computer vision. IEEE Access, [S. l.], v. 9, p. 14515–14527, 2021. DOI: 10.1109/ACCESS.2021.3052653.

ZHANG, Fan et al. Object Detection for Real-World Applications: A Comprehensive Survey and Future Directions. Visual Informatics, v. 7, n. 2, p. 89–110, 2023. Disponível em: https://doi.org/10.1016/j.visinf.2023.03.001 Acesso em: 30 mar. 2025.

ZHANG, Y.; LI, X.; WANG, Z. AI-driven safety monitoring for construction sites: a systematic review. Journal of Construction Engineering and Management, v. 147, n. 12, p. 04021092, 2021.

ZHENG, F.; CHEN, M.; LU, P. Integration of computer vision and deep learning for PPE monitoring in construction. Construction and Building Materials, v. 368, p. 130563, 2023.

ZHENG, T. et al. PPE detection using deep learning in industrial environments. Expert Systems with Applications, v. 212, p. 118–134, 2023.

ZHOU, P.; MA, L.; CHEN, R. Economic impact of AI-based safety systems in Industry 4.0. Journal of Industrial Engineering and Management, v. 15, n. 2, p. 256–272, 2022.

APÊNDICE

A.1 Código da Aplicação

https://github.com/EmyliPrestes/TCC-SIDE

Main.py:

```
from gui.layout import set_custom_styles
from gui.navigation import render_sidebar, render_page

def main():
    set_custom_styles()

    selected_page = render_sidebar()
    render_page(selected_page)


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

layout.py:

```
import streamlit as st

def set_custom_styles():
    st.markdown(
        """
        <style>
        [data-testid="stSidebar"] {
            background-color: #ac9c8f;  /* cor de fundo do menu */
            box-shadow: 0 9px 20px rgba(0, 0, 0, 0.6)
        }
        .main {
```

```
            background-color: #c9b6a9;  /* cor de fundo da página */
        }
        </style>
        """,
        unsafe_allow_html=True
    )
```

navigation.py:

```
import streamlit as st
from gui.pages import show_page
from streamlit_option_menu import option_menu

def render_sidebar():
    with st.sidebar:
        st.sidebar.image("./assets/logo2.png", use_column_width=True)
        selected = option_menu(
            "",
            ["Sobre o Projeto",'Detecção com Câmera ao Vivo', 'Detecção de EPIs em Arquivo', 'CheckList EPI', 'Setores e Equipamentos'],
            icons=['info-circle', 'camera', 'file-earmark', 'check2-circle', 'tools'],
            menu_icon="folder",
            default_index=0,
            styles={
                "container": {"padding": "5!important", "background-color": "#ac9c8f"},
                "icon": {"color": "white", "font-size": "25px"},
                "nav-link": {"font-size": "16px", "text-align": "left", "margin":"0px", "--hover-color": "#000000"},
                "nav-link-selected": {"background-color": "#7a695d"},
            }
        )
```

```
    return selected

def render_page(selected):
    show_page(selected)
```

pages.py:

```
import streamlit as st
import cv2
from ml import modelo
import time
from tempfile import NamedTemporaryFile
from pathlib import Path
from db import db

from PIL import Image




def home():
    st.markdown(
        """
        <div style="background-color: #ac9c8f; padding: 20px; border-radius: 10px; box-shadow: 0 9px 50px rgba(0, 0, 0, 0.6);">
            <h2 style="color:white;">Sobre o Projeto</h2>
            <p style='color:white'>Bem-vindo ao site dedicado à detecção de Equipamentos de Proteção Individual (EPIs), desenvolvido como parte do Trabalho de Conclusão de Curso (TCC) da aluna <span style='color:red'>Emyli Beatriz Braga Prestes</span>. Este projeto é um componente essencial do curso de Tecnologia em Eletrônica Industrial,
```

```
oferecido pelo <span style='color:green'>Instituto Federal do Amazonas Campus Manaus Distrito Industrial</span>. Sob a orientação do Professor Alexandre Lopes Martiniano, foi criado uma solução inovadora para a identificação automática de EPIs, utilizando tecnologias avançadas de visão computacional.</p>

        <p style='color:white'>O objetivo principal é aumentar a segurança no ambiente industrial, garantindo que todos os trabalhadores estejam devidamente equipados com os EPIs necessários para suas atividades. Este sistema automatizado de detecção pode ser integrado a câmeras de segurança existentes, proporcionando uma maneira eficiente e precisa de monitorar o uso de EPIs em tempo real.</p>

        <p style='color:white'>Esperamos que este projeto não só demonstre as habilidades técnicas desenvolvidas ao longo do curso, mas também contribua para a segurança e bem-estar dos profissionais na indústria. Agradecemos por visitar nosso site e por seu interesse em nosso trabalho.</p>
    </div>
    """,
    unsafe_allow_html=True
)

def page1():
    st.markdown("<h1 style='color:white;'>Detecção com Câmera ao Vivo</h1>", unsafe_allow_html=True)

    run = st.checkbox('Ativar câmera')

    FRAME_WINDOW = st.image([])

    camera = None
    if run:
        camera = cv2.VideoCapture(0)

    while run:
```

```
if camera:
    ret, frame = camera.read()
    if not ret:
        st.error("Falha ao capturar imagem da câmera.")
        break

    # Realizar a detecção usando o modelo treinado
    results= modelo.detect(frame)

    # Desenhar retângulos ou outras anotações na imagem, se necessário
    if results:
        for result in results:
            for box in result.boxes:
                x1, y1, x2, y2 = map(int, box.xyxy[0])
                assert len(box.cls) == 1
                nome_equipamento_detectado = result.names[int(box.cls[0])]
                color = (255, 0, 0)

                cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)

                (text_width, text_height), baseline = cv2.getTextSize(nome_equipamento_detectado, cv2.FONT_HERSHEY_SIMPLEX, 0.5, 2)

                cv2.rectangle(frame, (x1, y1 - text_height - baseline), (x1 + text_width, y1), color, -1)
                cv2.putText(frame, nome_equipamento_detectado, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

    # Converte o frame de BGR para RGB
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
            # Exibir a imagem com as detecções
            FRAME_WINDOW.image(frame)

            # Adicionar um pequeno atraso para permitir atualização da interface
            # time.sleep(0.1)

    if camera:
        camera.release()

def page2():
    st.markdown("<h1 style='color:white;'>Detecção de EPIs em FOTO</h1>",
unsafe_allow_html=True)
    st.markdown("<p style='color:white;'Tirar Foto</p>", unsafe_allow_html=True)



    '''uploaded_file = st.file_uploader("Escolha um arquivo", type=["jpg", "jpeg", "png"])'''

    botao = st.button('Tirar Foto')

    if botao:

        camera = cv2.VideoCapture(0)
        ret, frame = camera.read()

        if not ret:
            st.error("Falha ao capturar imagem da câmera.")

        else:
```

```
            with NamedTemporaryFile(delete=False, suffix='.jpg') as tmp_file:
                cv2.imwrite(tmp_file.name, frame)

            modelo.predict(tmp_file.name)

            Path(tmp_file.name).unlink()

            st.image('result.jpg', caption='Detecções', use_column_width=True)
```

```
def check_list_page():
    st.markdown("<h1 style='color:white;'>CheckList EPI</h1>", unsafe_allow_html=True)

    setores_equipamentos = db.listar_setores()
    setores = list(setores_equipamentos.keys())

    setor_selecionado = st.selectbox("Escolha um setor", setores)
    checkboxes_estado = {}

    # Mostrar a checklist de acordo com o setor selecionado
    if setor_selecionado:
        st.subheader(f"Equipamentos para o setor: {setor_selecionado}")
        equipamentos_obrigatorios = setores_equipamentos[setor_selecionado]

        if equipamentos_obrigatorios:
            for equipamento in equipamentos_obrigatorios:
```

```
            checkboxes_estado[equipamento] = st.empty()  # Cria um slot vazio para cada
checkbox
            checkboxes_estado[equipamento].checkbox(equipamento,          value=False,
disabled=True)
        else:
            st.write("Nenhum equipamento encontrado para este setor.")

    st.markdown("<p          style='color:white;'>Informações          da          Detecção</p>",
unsafe_allow_html=True)
    run = st.checkbox('Ativar Detecção')
    texto_acesso = st.empty()
    frames_deteccao = st.empty()
    texto_acesso.text("")
    frames_deteccao.text("")

    camera = None
    if run and setor_selecionado:
        camera = cv2.VideoCapture(0)

        NUM_FRAMES_DETECCAO = 120
        todos_equipamentos_detectados = False
        contador_frames = 0
        equipamentos_detectados = []
        while not todos_equipamentos_detectados and contador_frames <=
NUM_FRAMES_DETECCAO:
            if camera:
                ret, frame = camera.read()
                if not ret:
                    st.error("Falha ao capturar imagem da câmera.")
                    break
```

```
            # Realizar a detecção usando o modelo treinado
            results = modelo.detect(frame)

            # Desenhar retângulos ou outras anotações na imagem, se necessário
            if results:
                for result in results:
                    for box in result.boxes:
                        x1, y1, x2, y2 = map(int, box.xyxy[0])
                        assert len(box.cls) == 1
                        nome_equipamento_detectado = result.names[int(box.cls[0])]

                        if nome_equipamento_detectado not in equipamentos_detectados and
nome_equipamento_detectado in equipamentos_obrigatorios:
                            equipamentos_detectados.append(nome_equipamento_detectado)

checkboxes_estado[nome_equipamento_detectado].checkbox(nome_equipamento_det
ectado, value=True, disabled=True)

                        color = (255, 0, 0)
                        cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
                        (text_width, text_height), baseline =
cv2.getTextSize(nome_equipamento_detectado, cv2.FONT_HERSHEY_SIMPLEX, 0.5,
2)

                        cv2.rectangle(frame, (x1, y1 - text_height - baseline), (x1 +
text_width, y1), color, -1)
                        cv2.putText(frame, nome_equipamento_detectado, (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

                        # Converte o frame de BGR para RGB
                        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
                    FRAME_WINDOW = st.image([])
                    # Exibir a imagem com as detecções
                    FRAME_WINDOW.image(frame)

        if set(equipamentos_obrigatorios) == set(equipamentos_detectados):
            todos_equipamentos_detectados = True

        contador_frames += 1
        texto_acesso.text("Acesso em análise...")
        frames_deteccao.text(f"Frames                                    analisados:
{contador_frames}/{NUM_FRAMES_DETECCAO}")

    if todos_equipamentos_detectados:
        texto_acesso.text("Acesso Permitido!")
        frames_deteccao.text(f"Frames                                    analisados:
{contador_frames}/{NUM_FRAMES_DETECCAO}")
    else:
        texto_acesso.text("Acesso Negado!")
        frames_deteccao.text(f"Frames                                    analisados:
{contador_frames}/{NUM_FRAMES_DETECCAO}")

    if camera:
        camera.release()

  """
  criar uma janela de n frames para detectar os equipamentos listados durante x frames,
caso detectado mostrar na tela acesso permitido, caso contrário acesso negado
  """

def sector_page():
  st.header("Gerenciamento de Setores e Equipamentos")
```

```
# Seção para Inserir Equipamentos
st.subheader("Inserir Equipamento")
with st.form("equipamento_form"):
    nome_equipamento = st.text_input("Nome do Equipamento")
    submit_equipamento = st.form_submit_button("Inserir Equipamento")

    if submit_equipamento:
        if nome_equipamento:
            db.inserir_equipamento(nome_equipamento)
            st.success(f"Equipamento '{nome_equipamento}' inserido com sucesso!")
            st.experimental_rerun()
        else:
            st.error("Por favor, insira um nome para o equipamento.")

# Seção para Visualizar e Deletar Equipamentos
st.subheader("Equipamentos Existentes")
equipamentos = db.listar_equipamentos()

if equipamentos:
    for equipamento in equipamentos:
        with st.expander(f"{equipamento}"):
            if st.button(f"Deletar {equipamento}", key=f"del_eq_{equipamento}"):
                db.excluir_equipamento(equipamento)
                st.warning(f"Equipamento '{equipamento}' deletado com sucesso!")
                st.experimental_rerun()
else:
    st.write("Nenhum equipamento encontrado.")

# Seção para Inserir Setores
st.subheader("Inserir Setor")
```

```
    with st.form("setor_form"):
        nome_setor = st.text_input("Nome do Setor")
        equipamentos_disponiveis = db.listar_equipamentos()

        lista_de_equipamentos = st.multiselect("Selecione os Equipamentos", equipamentos_disponiveis)

        submit_setor = st.form_submit_button("Inserir Setor")

        if submit_setor:
            if nome_setor and lista_de_equipamentos:
                db.inserir_setor(nome_setor, lista_de_equipamentos)
                st.success(f"Setor '{nome_setor}' inserido com sucesso com os equipamentos selecionados!")
                st.experimental_rerun()
            else:
                st.error("Por favor, preencha todos os campos.")

    # Seção para Visualizar e Deletar Setores
    st.subheader("Setores Existentes")
    setores = db.listar_setores()

    if setores:
        for setor, equipamentos in setores.items():
            with st.expander(f"Setor: {setor}"):
                st.write("Equipamentos:")
                for equipamento in equipamentos:
                    st.write(f"- {equipamento}")
                if st.button(f"Deletar Setor {setor}", key=f"del_set_{setor}"):
                    db.excluir_setor(setor)
                    st.warning(f"Setor '{setor}' deletado com sucesso!")
```

```
                st.experimental_rerun()
    else:
        st.write("Nenhum setor encontrado.")


def show_page(page_name):
    if page_name == "Sobre o Projeto":
        home()
    elif page_name == "Detecção com Câmera ao Vivo":
        page1()
    elif page_name == "Detecção de EPIs em Arquivo":
        page2()
    elif page_name== "TESTE":
        page3()
    elif page_name == "CheckList EPI":
        check_list_page()
    elif page_name == "Setores e Equipamentos":
        sector_page()
```

__init__.py

```
from .model import Model

MODEL_PATH = 'assets/best.pt'

modelo = Model(MODEL_PATH)
```

database.py:

```
import sqlite3
import json
```

```
class Database:
    def __init__(self, db_name):
        self.db_name = db_name
        self.conn = sqlite3.connect(db_name, check_same_thread=False)
        self.cursor = self.conn.cursor()
        self.create_tables()

    def create_tables(self):

        # Cria tabela equipamentos caso não exista
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS equipamentos (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                nome TEXT NOT NULL UNIQUE
            )
        ''')
        # Cria tabela setores caso não exista
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS setores (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                nome TEXT NOT NULL,
                lista_de_equipamentos TEXT
            )
        ''')
        self.conn.commit()

    # CRUD para tabela equipamentos
    def inserir_equipamento(self, nome):

        try:
```

```
        self.cursor.execute('''
            INSERT INTO equipamentos (nome)
            VALUES (?)
        ''', (nome,))
        self.conn.commit()
    except sqlite3.IntegrityError:
        print(f'Equipamento "{nome}" já existe.')

def consultar_equipamento(self, id):

    self.cursor.execute('SELECT * FROM equipamentos WHERE id = ?', (id,))
    return self.cursor.fetchone()

def listar_equipamentos(self):
    self.cursor.execute('SELECT * FROM equipamentos')
    return [equipamento[1] for equipamento in self.cursor.fetchall()]

def excluir_equipamento(self, nome):
    self.cursor.execute('DELETE FROM equipamentos WHERE nome = ?', (nome,))
    self.conn.commit()

# CRUD para tabela setores
def inserir_setor(self, nome, lista_de_equipamentos_ids):
    equipamentos_json = json.dumps(lista_de_equipamentos_ids)
    self.cursor.execute('''
        INSERT INTO setores (nome, lista_de_equipamentos)
        VALUES (?, ?)
    ''', (nome, equipamentos_json))
    self.conn.commit()

def consultar_setor(self, id):
```

```
    self.cursor.execute('SELECT * FROM setores WHERE id = ?', (id,))
    setor = self.cursor.fetchone()

    if setor:
        id, nome, lista_de_equipamentos = setor
        lista_de_equipamentos_ids = json.loads(lista_de_equipamentos)
        return id, nome, lista_de_equipamentos_ids
    return None

def obter_lista_equipamentos_setor(self, setor_id):
    # Consulta o setor pelo ID
    self.cursor.execute('''
        SELECT lista_de_equipamentos FROM setores WHERE id = ?
    ''', (setor_id,))

    # Recupera o resultado da consulta
    resultado = self.cursor.fetchone()

    if resultado:
        # Recupera a string JSON que representa a lista de equipamentos
        lista_de_equipamentos_json = resultado[0]

        # Converte a string JSON de volta para uma lista
        lista_de_equipamentos = json.loads(lista_de_equipamentos_json)

        return lista_de_equipamentos
    else:
        # Se o setor não for encontrado, retorna None ou uma lista vazia
        return None  # ou [] para uma lista vazia
```

```
def atualizar_setor(self, id, nome=None, lista_de_equipamentos_ids=None):

    setor = self.consultar_setor(id)
    if not setor:
        print(f'Setor com ID {id} não encontrado.')
        return

    if nome is None:
        nome = setor[1]
    if lista_de_equipamentos_ids is None:
        lista_de_equipamentos_ids = setor[2]
    else:
        lista_de_equipamentos_ids = json.dumps(lista_de_equipamentos_ids)

    self.cursor.execute('''
        UPDATE setores
        SET nome = ?, lista_de_equipamentos = ?
        WHERE id = ?
    ''', (nome, lista_de_equipamentos_ids, id))
    self.conn.commit()

def listar_setores(self):
    # Seleciona todos os setores com suas listas de equipamentos
    self.cursor.execute('''
        SELECT nome, lista_de_equipamentos FROM setores
    ''')

    # Recupera todos os resultados
    setores = self.cursor.fetchall()

    # Dicionário para armazenar os setores e suas listas de equipamentos
```

```
    dict_setores = {}

    for setor in setores:
        nome_setor, lista_de_equipamentos_json = setor

        # Converte a string JSON de volta para uma lista
        lista_de_equipamentos = json.loads(lista_de_equipamentos_json)

        # Extrai apenas os nomes dos equipamentos
        nomes_equipamentos = lista_de_equipamentos

        # Adiciona ao dicionário
        dict_setores[nome_setor] = nomes_equipamentos

    return dict_setores

def excluir_setor(self, nome):
    try:
        self.cursor.execute('DELETE FROM setores WHERE nome = ?', (nome,))
        self.conn.commit()
    except sqlite3.Error as e:
        print(f"Erro ao excluir setor: {e}")


def close(self):
    self.conn.close()
```

model.py:

```
from ultralytics import YOLO
import torchvision

class Model:
    def __init__(self, model_path):
        self.model = YOLO(model_path)

    def detect(self, frame, conf_threshold=0.5, iou_threshold=0.4):
        # Realizar a detecção usando o modelo YOLO
        results = self.model.predict(frame)

        # Filtrar resultados com base no limiar de confiança
        filtered_results = []
        for result in results:
            boxes = result.boxes
            if boxes is not None:
                # Aplicar Non-Maximum Suppression (NMS) para evitar detecções duplicadas
                keep = torchvision.ops.nms(boxes.xyxy, boxes.conf, iou_threshold)
                filtered_boxes = [boxes[i] for i in keep if boxes[i].conf.item() > conf_threshold]
                if filtered_boxes:
                    result.boxes = filtered_boxes
                    filtered_results.append(result)
        return filtered_results

    def predict(self, image_path):
        results = self.model.predict(image_path, save=False, imgsz=640, conf=0.7)
        # Processar lista de resultados
        if results:
            for result in results:
                result.save(filename='result.jpg')  # salvar no disco
```