**UNIVERSIDADE SANTA CECÍLIA**

**ENGENHARIA DA COMPUTAÇÃO**

**ENZO DE SOUZA LIMA**

**GIOVANI MARTINHO DO NASCIMENTO**

**KEVIN BULGARELLI DOSSANTOS**

**PEDRO HENRIQUE SANTOS SILVA**

**MONITORAMENTO DO USO DE EPIs**

**Santos – SP**

**Dezembro/2025**

**UNIVERSIDADE SANTA CECÍLIA**

**ENGENHARIA DA COMPUTAÇÃO**

**ENZO DE SOUZA LIMA**

**GIOVANI MARTINHO DO NASCIMENTO**

**KEVIN BULGARELLI DOSSANTOS**

**PEDRO HENRIQUE SANTOS SILVA**

**MONITORAMENTO DO USO DE EPIs**

**Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do título de Bacharelado da Faculdade de Engenharia da Computação da Universidade Santa Cecília, sob a orientação da Professora Ma. Raquel Galhardo de Carvalho Lopes Araújo e coorientação do Professor Me. Luís Fernando Pompeo Ferrara**

**Santos – SP**

**Dezembro/2025**

Enzo de Souza Lima

Giovani Martinho do Nascimento

Kevin Bulgarelli Dossantos

Pedro Henrique Santos Silva

**Monitoramento do uso de EPIs**

Trabalho de Conclusão de Curso apresentado como exigência para obtenção do título de Engenheiro de Computação à Faculdade de Engenharia de Computação da Universidade Santa Cecília – UNISANTA.

Data da aprovação: \_\_\_\_/\_\_\_\_/\_\_\_\_\_\_ Nota:\_\_\_\_\_\_\_\_\_\_\_\_

Banca Examinadora:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.ª Ma. Kelly Cristina Abou Arabi de Mendonça

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Me. Sergio Schina de Andrade

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Me. Yuri Silva Cruz Storino

**RESUMO**

O uso de Equipamentos de Proteção Individual (EPIs) é essencial para garantir a segurança dos trabalhadores expostos a atividades de risco. No entanto, a não utilização ou o uso inadequado desses equipamentos ainda é um problema recorrente, aumentando o risco de acidentes e penalidades para as empresas. Este projeto propõe o desenvolvimento de um sistema automatizado baseado em visão computacional e aprendizado de máquina para monitoramento em tempo real do uso correto de EPIs. O sistema utilizará uma câmera conectada a uma Raspberry Pi 4, processando as imagens por meio de um modelo treinado no TensorFlow Lite, com suporte de um acelerador USB Coral para otimização do desempenho. A solução será capaz de identificar automaticamente um conjunto específico de EPIs, definido em conjunto com a banca avaliadora do TCC, composto por óculos de proteção, botas, capacete e calça. Caso seja detectada a ausência de um ou mais desses equipamentos, um alerta será acionado e a ocorrência será registrada em um sistema acessível via *web*. Essa abordagem possibilita uma fiscalização mais eficiente, reduzindo falhas humanas na inspeção visual e promovendo maior conformidade com as normas de segurança, como a NR6. O projeto contribui para a segurança jurídica das empresas, minimiza riscos de acidentes de trabalho e reforça a cultura organizacional voltada para a prevenção e proteção dos trabalhadores.

**Palavras-chave**: Visão Computacional; Aprendizado de Máquina; EPIs; Segurança no Trabalho; Reconhecimento de Imagem.

**LISTA DE FIGURAS**

[Figura 1 - Raspberry Pi 4 12](#_Toc216606841)

[Figura 2 - Teste da câmera. 13](#_Toc216606842)

[Figura 3 - Esquemático. 17](#_Toc216606843)

[Figura 4 - Biblioteca 27](#_Toc216606844)

[Figura 5 - Estados 29](#_Toc216606845)

[Figura 6 – CapturaCamera 30](#_Toc216606846)

[Figura 7 – Config 32](#_Toc216606847)

[Figura 8 - RastreadorDeteccoes 33](#_Toc216606848)

[Figura 9 - RastreadorDeteccoes 34](#_Toc216606849)

[Figura 10 - ControladorEPI 36](#_Toc216606850)

[Figura 11 - ControladorEPI 37](#_Toc216606851)

[Figura 12 - GerenciadorFirebase 39](#_Toc216606852)

[Figura 13 - Página inicial do site. 41](#_Toc216606853)

[Figura 14 - Página de Login 46](#_Toc216606854)

[Figura 15 - Página de Usuário 47](#_Toc216606855)

[Figura 16 - Dashboard 48](#_Toc216606856)

[Figura 17 - Histórico de Ocorrência 49](#_Toc216606857)

[Figura 18 – Imagem de Ocorrência 50](#_Toc216606858)

[Figura 19 - Página de Configuração 51](#_Toc216606859)

[Figura 20 - Modelo do encapsulamento 52](#_Toc216606860)

[Figura 21 - Imagem superior 53](#_Toc216606861)

[Figura 22 - Encapsulamento dos componentes fechado 54](#_Toc216606862)

[Figura 23 - Encapsulamento aberto com ventoinha 55](#_Toc216606863)

[Figura 24 - Encapsulamento aberto com componentes visíveis 56](#_Toc216606864)

[Figura 25 - Matriz de confusão do modelo YOLOv8n 58](#_Toc216606865)

[Figura 26 - Treinamento, Validação e Testes 59](#_Toc216606866)

[Figura 27 - Contagem de Anotações por Imagens 60](#_Toc216606867)

[Figura 28 - Análise do Número de Imagens 60](#_Toc216606868)

[Figura 29- Analisando EPI 61](#_Toc216606869)

[Figura 30 - Análise completa e aprovado 62](#_Toc216606870)

[Figura 31 - Análise completa e rejeitada 63](#_Toc216606871)

[Figura 32 - Análise completa e rejeitada com dois EPIs 63](#_Toc216606872)

[Figura 33 - Entrando na zona de detecção 64](#_Toc216606873)

[Figura 34 - Saindo da zona de detecção 65](#_Toc216606874)

[Figura 35 - Análise sem a Coral 66](#_Toc216606875)

[Figura 36 - Zona de detecção vazia 66](#_Toc216606876)

[Figura 37 - Análise de desempenho através do Windows 67](#_Toc216606877)

[Figura 38 - Zona de detecção vazia através do Windows 68](#_Toc216606878)

**SUMÁRIO**

[1 Introdução 8](#_Toc216583220)

[1.1 Machine Learning (ML), ou Aprendizado De Máquina 9](#_Toc216583221)

[1.2 Raspberry Pi 4 10](#_Toc216583222)

[1.3 Inferências de Machine Learning 11](#_Toc216583223)

[1.4 TensorFlow Lite 11](#_Toc216583224)

[1.5 Visão Computacional 12](#_Toc216583225)

[1.6 Processamento De Imagens 12](#_Toc216583226)

[1.7 Firebase 13](#_Toc216583227)

[2. Objetivo 14](#_Toc216583228)

[3. Metodologia 15](#_Toc216583229)

[4. Desenvolvimento 17](#_Toc216583230)

[4.1 Treinamento da IA (Modelo YOLOv8n) 17](#_Toc216583231)

[4.1.1 Roboflow 17](#_Toc216583232)

[4.1.2 Modelo Coral Edge TPU 18](#_Toc216583233)

[4.1.3 Quantização 18](#_Toc216583234)

[4.1.4 Definição de Classes 19](#_Toc216583235)

[4.2 Lógica de Detecção Implementada na Raspberry Pi 19](#_Toc216583236)

[4.2.1 Captura de Vídeo Otimizado com *Thread*ing 19](#_Toc216583237)

[4.2.2 Otimização do Processamento de *Frame*s 19](#_Toc216583238)

[4.2.3 Pré-processamento de *Frame*s para inferência 20](#_Toc216583239)

[4.2.4 Execução da Inferência com YOLOv8 20](#_Toc216583240)

[4.2.5 Delegação para Coral Edge TPU 20](#_Toc216583241)

[4.2.6 Pós-processamento dos Resultados da Detecção 21](#_Toc216583242)

[4.2.6.1 Caixas Delimitadoras 21](#_Toc216583243)

[4.2.6.2 Identificador de Classe e Score de Confiança 21](#_Toc216583244)

[4.2.6.3 Reescalonamento de Coordenadas e Visualização 22](#_Toc216583245)

[4.3 Implementação do Sistema de Alerta e Notificação 22](#_Toc216583246)

[4.3.1 Lógica de Identificação de Ausência de EPIs e Mitigar o Falso Positivo 22](#_Toc216583247)

[4.3.2 Tratamento de Falsos Positivos e Estabilidade da Detecção 22](#_Toc216583248)

[4.3.3 Geração e Formatação de Mensagens de Alerta Personalizadas 23](#_Toc216583249)

[4.3.4 Mecanismo de Controle de Frequência de Alertas Anti-Spam) 23](#_Toc216583250)

[4.3.5 Integração Robusta com Firebase Firestore para Registro de Alertas 24](#_Toc216583251)

[4.3.6 Notificações Imediatas via Telegram 24](#_Toc216583252)

[4.3.7 Consideração de Segurança 25](#_Toc216583253)

[4.3.8 Bibliotecas Utilizadas 25](#_Toc216583254)

[4.3.9 Estrutura do Código 28](#_Toc216583255)

[*4.3.9.1* *Significados dos Estados* 28](#_Toc216583256)

[4.3.9.2 CapturaCamera 29](#_Toc216583257)

[4.3.9.3 Config 31](#_Toc216583258)

[4.3.9.4 RastreadorDeteccoes 32](#_Toc216583259)

[4.3.9.5 MostrarFPS 33](#_Toc216583260)

[4.3.9.6 PreparadorImagem 34](#_Toc216583261)

[4.3.9.7 ControladorEPI 34](#_Toc216583262)

[4.3.9.8 GerenciadorTelegram 48](#_Toc216583263)

[4.4 Desenvolvimento do *Site* 50](#_Toc216583264)

[4.4.1 Tecnologias Frontend Empregadas 51](#_Toc216583265)

[4.4.1.1 HTML (HyperText Markup Language) 51](#_Toc216583266)

[4.4.1.2 Tailwind CSS 51](#_Toc216583267)

[4.4.1.3 Java*Script* (Vanilla JS e Módulos ES6) 51](#_Toc216583268)

[4.4.1.4 Firebase SDK para Java*Script* 51](#_Toc216583269)

[4.4.2 Funcionalidades Implementadas na Interface *Web* 52](#_Toc216583270)

[4.4.2.1 Autenticação de Usuários 52](#_Toc216583271)

[4.4.3 Painel Principal 52](#_Toc216583272)

[4.4.4 Visualização do Histórico de Ocorrências 53](#_Toc216583273)

[4.4.5 Gerenciamento de Contas de Usuário 54](#_Toc216583274)

[4.4.6 Configuração Essencial e Hospedagem da Interface *Web* 54](#_Toc216583275)

[4.4.6.1 Configuração do Firebase para o Frontend 54](#_Toc216583276)

[4.4.6.2 Hospedagem com Firebase Hosting 55](#_Toc216583277)

[4.4.7 Página de Login 55](#_Toc216583278)

[4.4.8 Página de Usuário 56](#_Toc216583279)

[4.4.9 Dashboard 57](#_Toc216583280)

[4.4.10 Histórico de Ocorrência 58](#_Toc216583281)

[4.4.11 Encapsulamento para Raspberry Pi 61](#_Toc216583282)

[4.5 Resultados e Testes 66](#_Toc216583283)

[4.5.1 Teste de otimização da detecção 70](#_Toc216583284)

[4.5.2 Teste com Windows 76](#_Toc216583285)

[5. Conclusão 78](#_Toc216583286)

[6. Referências 81](#_Toc216583287)

# Introdução

Os Equipamentos de Proteção Individual – EPI são instrumentos utilizados para proteção individual do trabalhador, indispensáveis em algumas áreas de atuação, sendo muitas vezes o maior contribuidor para a sobrevivência humana em acidentes de trabalho.

Os primeiros registros da utilização desses meios de proteção remontam a civilizações antigas, como no caso dos egípcios, onde trabalhadores utilizavam almofadas de algodão para proteger os joelhos durante a construção de monumentos [1].

A larga adoção e evolução desses equipamentos deve-se à revolução industrial. Os EPIs foram amplamente adotados em setores com altos índices de acidentes, como metalurgia, mineração e fundições [2], adaptando-se às necessidades e evoluções na indústria ao longo do tempo.

No Brasil, a obrigatoriedade do uso de EPIs surgiu em 1º de maio de 1943 com o decreto de Lei N° 5.452, que em seu artigo 160 [3], determina as atividades em que o empregador deve fornecer os equipamentos de proteção individual.

A Norma Regulamentadora n° 6 - NR6, estabelecida pela Lei n° 6.514 em 1977, é a principal norma que regulamenta os Equipamentos de Proteção Individual, abordando os fornecimentos essenciais e a utilização deles. O objetivo primordial do uso da regulamentação é fornecer orientação sobre o uso adequado dos EPIs, estabelecer critérios para fabricantes e importadores, relacionar o equipamento necessário para cada tipo de emprego e proporcionar segurança e bem-estar aos trabalhadores.

A fiscalização da qualidade dos EPIs é de responsabilidade do Ministério da Economia, visando garantir a competência dos equipamentos e proporcionar segurança durante sua utilização pelos trabalhadores.

Os EPIs podem ser utilizados para proteger o indivíduo de diversas formas como a proteção auditiva, respiratória, ocular, entre outras. Os equipamentos mais comuns e utilizados são óculos de proteção, protetores auditivos, luvas isolantes, botas e uniformes para proteção.

Apesar se sua importância muitos trabalhadores negligenciam os EPIs, usando de forma inadequada ou até mesmo não utilizando-os. Milhões de trabalhadores morrem todo ano no mundo por acidentes ou doenças relacionadas ao trabalho [4], muitas dessas mortes poderiam ser evitadas com o uso devido dos equipamentos de proteção.

O não cumprimento da lei da NR6 traz prejuízos tanto para os colaboradores quanto para a empresas. Caso seja identificado o descumprimento de itens exigidos pela norma, a empresa pode ter as suas atividades interrompidas temporariamente até que a situação seja regularizada. Em caso de reincidência a empresa pode ser multada.

Segundo a norma vigente, a corporação tem o dever de fornecer os EPIs, realizar treinamentos e orientar sobre o seu uso adequado. Entretando, se o colaborador não fizer o uso adequado dos equipamentos, a empresa pode aplicar medidas disciplinares, advertências e em caso de recorrência até mesmo a demissão por justa causa [5].

O objetivo desse projeto é monitorar o uso dos equipamentos de proteção individual pelos funcionários que desempenham funções consideradas perigosas, garantindo assim o cumprimento da obrigatoriedade do uso do EPI.

Com isso a corporação tem uma maior segurança jurídica em suas operações, evitando possíveis acidentes de trabalho e o empregado é estimulado a usar os EPIs de forma correta, realizando as suas atividades com maior segurança e menor risco de doenças ocupacionais e acidentes.

Para alcançar esse objetivo, será desenvolvida uma aplicação com base nos conceitos de aprendizado de máquina, estes são sistemas capazes de aprender por meio de algoritmos e dados, em uma semelhança ao processo de aprendizagem humana [6] e visão computacional que são sistemas computacionais capazes de extrair informações de imagens [7] para analisar as imagens capturadas por uma câmera, identificando se as pessoas estão utilizando os EPIs obrigatórios para o local, modificando apenas o determinado EPIs que será utilizado no local apropriado. O processamento de imagem será feito através de uma Raspberry Pi 4 com o auxílio de um acelerador USB coral para melhorar o desempenho do projeto no reconhecimento dos EPIs e das pessoas nas imagens [8].

Caso seja detectada a falta do uso do equipamento, um alarme de advertência será acionado e aparecerá uma notificação no sistema, alertando a empresa da inconformidade.

## Machine Learning (ML), ou Aprendizado De Máquina

É uma área da inteligência artificial que desenvolve algoritmos capazes de aprender a partir de dados, simulando o processo de aprendizado humano. Existem três principais tipos: o aprendizado supervisionado, onde o modelo é treinado com dados rotulados para mapear entradas às saídas corretas; o aprendizado não supervisionado, que trabalha com dados não rotulados para identificar padrões ou agrupamentos ocultos; e o aprendizado por reforço, em que o modelo aprende interagindo com o ambiente, ajustando suas decisões para maximizar recompensas ao longo do tempo. Esses métodos são amplamente aplicados em diversas áreas, como reconhecimento de voz, recomendação de produtos, diagnósticos médicos, estudos científicos, reconhecimento de imagem etc. [9].

## Raspberry Pi 4

O Raspberry Pi, como será mostrado na figura 1, é um minicomputador integrado a uma única placa que tem o seu funcionamento semelhantemente a um PC ou notebook. Seu diferencial é seu tamanho compacto, com comprimento e largura semelhantes a um cartão de crédito, e seu custo-benefício por ser um produto com muita tecnologia embarcada e ter um preço consideravelmente acessível. Este dispositivo traz consigo conexões USB do tipo A nas versões 2.0 e 3.0 e do tipo C, onde a fonte de alimentação é conectada, entradas micro HDMI com suporte a monitores com resolução 4K, porta Gigabit Ethernet para conexões com a internet em alta velocidade, conectividade com o Wi-Fi 2.4 GHz e 5.0 GHz IEEE 802.11ac e ao Bluetooth 5.0. O processador deste modelo é o Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz e contém 4GB de memória RAM LPDDR4-3200 SDRAM. Esta placa é comumente utilizada para a elaboração de projetos, protótipos e fins educacionais [10] [11].

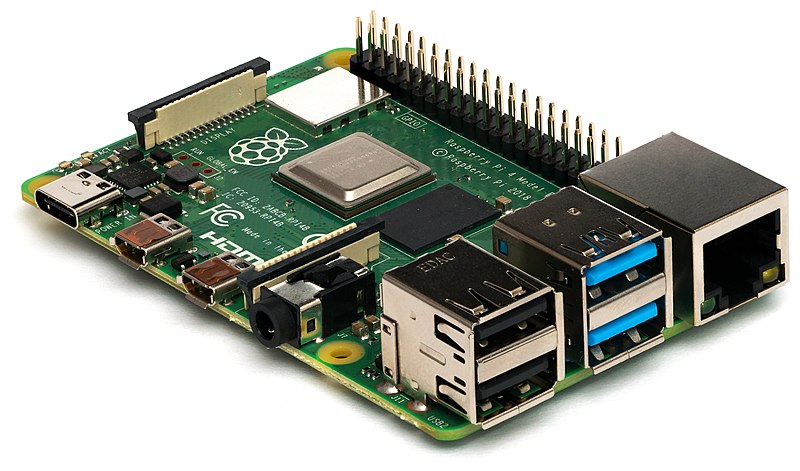


Figura - Raspberry Pi 4

## Inferências de Machine Learning

O processo de inferências consiste na entrada de novos dados a um modelo de *machine learning* e saídas são geradas de acordo com o treinamento realizado na criação deste modelo. Com ele é possível reconhecer em imagens objetos treinados no modelo de ML [12].

## TensorFlow Lite

O TensorFlow Lite é uma biblioteca de código aberto da Google que oferece diversas funcionalidades para o processamento de imagens em projetos de Machine Learning. Com ele é possível treinar e executar modelos já treinados.

Essa biblioteca pode ser utilizada em programas de detecção de objetos, classificação de imagens, reconhecimento de padrões entre outras formas de aplicação. É possível importar dados de imagens em formatos como JPEG ou PNG, onde poderá ser feita a detecção de objetos nesta imagem.

Ele pode ser executado em CPUs (Unidades de Processamento Central), GPUs (Unidades de Processamento Gráfico) ou também tem TPUs (Unidades de Processamento Tensor). Esta última foi desenvolvida pelo Google para acelerar a execução de tarefas que fazem o uso do TensorFlow. A linguagem mais utilizada em projetos que utilizam esta tecnologia é o Python [13] [14] [15] [15].

## Visão Computacional

A visão computacional é uma tecnologia que permite às máquinas reconhecerem e interpretar imagens, bem como os elementos presentes nelas. Essa aplicação utiliza inteligência artificial para processar as imagens, como mostrado na figura 2, identificando objetos, rostos e padrões. Por meio da análise detalhada, a visão computacional capacita as máquinas a entenderem o conteúdo visual de forma semelhante aos seres humanos, possibilitando uma ampla gama de aplicações em diversas áreas, como segurança, medicina, automação industrial, entre outras [16].

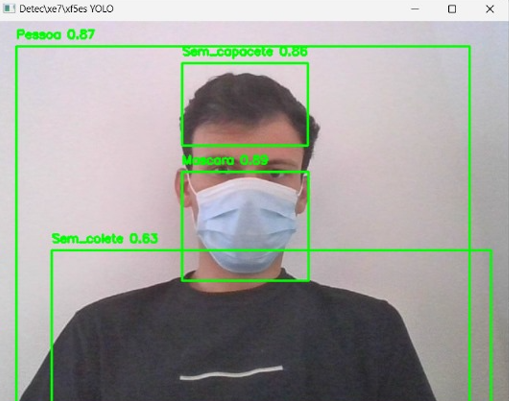
****

Figura - Teste da câmera.

## Processamento De Imagens

O processamento de imagem é uma técnica de manipulação de dados que utiliza imagens ou vídeos como entrada e saída de informações. Através de técnicas de aprendizado de máquina e reconhecimento de padrões, é possível identificar e analisar os elementos presentes nas imagens, permitindo a extração de informações relevantes e a automação de tarefas complexas, como detecção de objetos, análise de textura e reconhecimento de padrões visuais [17].

## Firebase

Firebase é uma plataforma de desenvolvimento de aplicativos do Google, oferecendo um conjunto de serviços de *backend* gerenciados. Com a plataforma BaaS (*Backend* as a Service), o Firebase abstrai muitas das complexidades do desenvolvimento e do gerenciamento da infraestrutura do *backend*.

Utilizando Firestore, um banco de dados NoSQL, flexível e escalável, armazenando dados em formato de documento organizado em coleções. Ele é utilizado para guardar registros de alerta e dados do usuário da interface [18].

# Objetivo

O objetivo do projeto é monitorar os trabalhadores de uma certa área para analisar e registrar se eles estão utilizando seus equipamentos de maneira adequada, onde o uso de Equipamentos de Proteção Individual (EPIs) é obrigatório. A câmera irá capturar imagens que, serão analisadas por um sistema automatizado de reconhecimento de imagem e então tais imagens serão enviadas para um banco de dados que estará conectado com um site criado pelo grupo para avaliar a quantidade de ocorrências e para que o responsável por tal setor fique sabendo quando a falta de EPI foi registrada, também haverá uma notificação através do aplicativo Telegram por questões de praticidade. O conjunto específico de EPIs foi definido em juntamente com a banca avaliadora do TCC, que inclui óculos de proteção, botas e capacete.

# Metodologia

Neste projeto as imagens serão captadas por uma webcam (Webcam USB Full HD 1080p, Modelo W2) conectada via USB a um Raspberry Pi 4, onde esses dados serão processados com o auxílio de um acelerador USB Coral por meio de um algoritmo de aprendizado de máquina desenvolvido na linguagem Python. O programa utilizará um modelo treinado utilizando o TensorFlow Lite, neste modelo há categorização dos objetos, neste caso, os EPIs e o trabalhador que utilizará o EPI, por exemplo: existe uma classe chamada “capacete”, o programa sabe que o que foi detectado é um capacete, pois foi treinado com diversas imagens com pessoas utilizando capacete. Com este modelo será possível analisar as imagens recebidas pela câmera e reconhecer os objetos na imagem, que terão destaque na visualização do vídeo por meio da criação de “caixas delimitadoras” que são gráficos aplicados a imagem para facilitar a visualização de cada objeto detectado.

Ao identificar uma pessoa na imagem e na mesma não forem reconhecidos os EPIs, será demonstrado tanto graficamente (pelas caixas delimitadoras) quanto por meio de uma função dentro do algoritmo de que houve falta de uso dos equipamentos no local monitorado. Esta ocorrência será enviada para um *site*, onde haverá a sua visualização. O acesso a página poderá ser feito por qualquer dispositivo conectado à internet através de um navegador.

Para a realização dos experimentos e validação do sistema, estabeleceu-se um cenário controlado de monitoramento. A câmera foi posicionada a uma altura de aproximadamente 1,5 metros do solo, com uma distância operacional por volta de 5 metros em relação à área de passagem dos trabalhadores. Essas medidas foram definidas para garantir o enquadramento ideal do corpo inteiro para a detecção dos EPIs (capacete, botas e óculos) com as limitações de hardware do grupo.

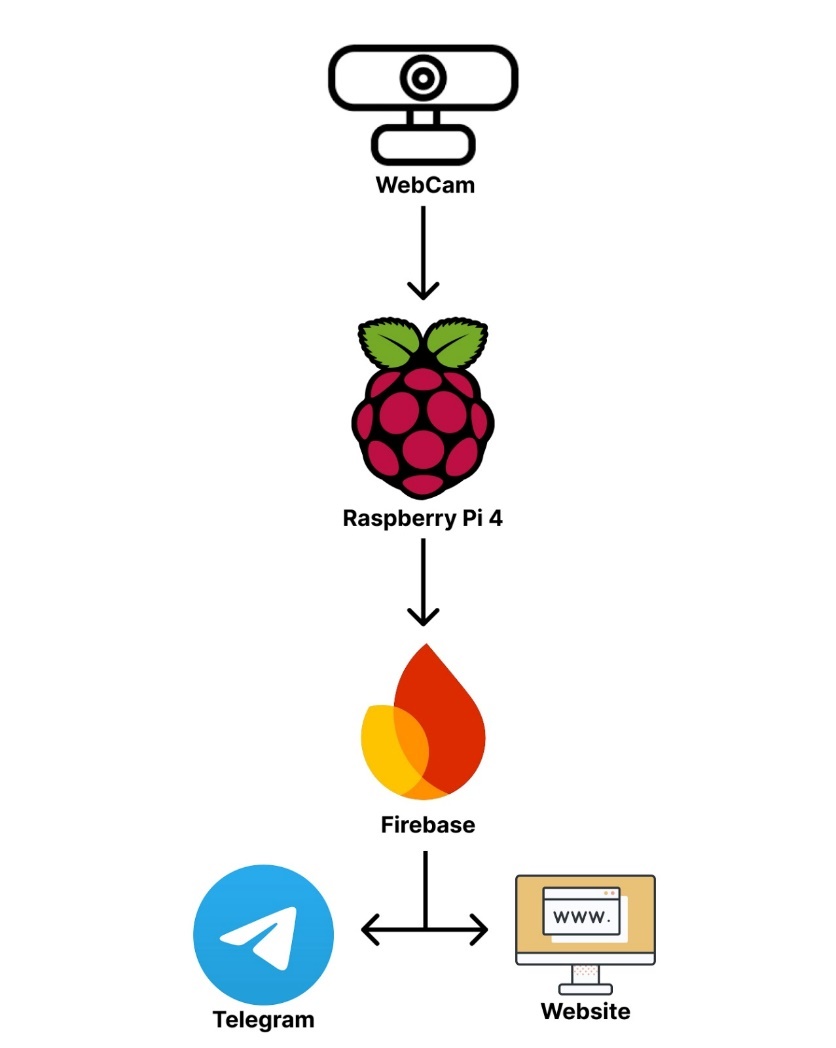


Figura - Esquemático.

# Desenvolvimento

O desenvolvimento do projeto iniciou-se com o treinamento de um modelo de inteligência artificial pré-existente, utilizando um DataSet público para o refinamento de sua capacidade de detecção de Equipamentos de Proteção Individual (EPIs). O algoritmo foi configurado para identificar tanto a utilização adequada quanto a ausência desses equipamentos. Na etapa de implementação, o modelo foi integrado a uma placa Raspberry Pi 4, responsável pelo processamento das imagens capturadas por uma câmera USB. Paralelamente, foi desenvolvida uma interface *web* com autenticação de usuários para acesso aos registros armazenados no Firebase, complementada por um sistema de notificação via *Chat*Bot no Telegram, programado para enviar alertas condicionais em caso de detecção de não conformidades.

## Treinamento da IA (Modelo YOLOv8n)

Para o treinamento do modelo YOLO (You Only Look Once), uma técnica capaz de detectar com precisão objetos, foi utilizado a plataforma Google Colab, um ambiente gratuito de *notebooks* Jupyter na nuvem, que possui suporte a GPUs e TPUs, facilitando assim o treinamento de modelos de aprendizado profundo sem a necessidade de infraestrutura local. O ambiente, inicialmente, foi configurado com a instalação do Python 3.9 e a criação de um ambiente virtual para garantir compatibilidade com as bibliotecas necessárias, Ultralytics e PyTorch.

PyTorch é uma biblioteca em Python amplamente utilizada em aplicações de *deep learning*, área da inteligência artificial que utiliza redes neurais com várias camadas para aprender padrões complexos. É por meio desse tipo de aprendizado que modelos como o YOLO conseguem reconhecer e detectar objetos de forma eficiente.

O treinamento foi realizado com o YOLOv8n, que é voltada a detecção de EPIs. Após o treinamento, o modelo foi convertido para o formato TensorFlow Lite otimizado para Coral Edge TPU, permitindo sua execução em tempo real na Raspberry Pi 4 com acelerador USB.

### Roboflow

Para a criação e preparação do DataSet, foi utilizado módulo para treinamento de detecção de objetos, chamada de Roboflow, que é uma ferramenta *online* amplamente adotada para gerenciamento, anotação e exportação de DataSets de visão computacional. As imagens foram obtidas através da funcionalidade Universe do Roboflow, permitindo explorar e utilizar imagens públicas disponíveis por outros objetos. As imagens foram selecionadas por diferentes conjuntos da aplicação, garantindo diversidade e representatividade das classes de interesse. Após as seleções, as imagens foram anotadas manualmente utilizando a interface gráfica do Roboflow, que cada objeto de interesse foi marcado com Bounding Boxes, associando as respectivas classes do projeto. As anotações foram revisadas cuidadosamente e ajustadas quando necessário, garantindo maior precisão na delimitação dos objetos.

Após as anotações revisadas, foi criada a versão final do DataSet, contendo a estrutura completa necessário para o treinamento. O Roboflow permite exportar o DataSet em diversos formatos e sendo escolhido o formato compatível com o *frame*work YOLO.

### Modelo Coral Edge TPU

Após o treinamento, o YOLOv8n precisa ser convertido para o TensorFlow Lite. Para o formato de arquivo seria para .tflite, que seria um formato para o modelo otimizado para implantar em dispositivos de borda como o Raspberry Pi.

### Quantização

A Quantização é uma técnica de otimizar modelos que visa reduzir o tamanho do arquivo para poder acelerar o tempo de inferência, geralmente isso acontece por perda de precisão. Ela converte os pesos e as ativações da rede neural de números de ponto flutuante de alta precisão, por exemplo: 32 bits, para formatos de menor precisão, como exemplo: inteiros de 8 bits.

Para o Coral Edge TPU, é necessário que o modelo .tflite seja totalmente quantizado para inteiros de 8 bits (INT8). Isso significa que todos os parâmetros e todas as operações matemáticas durante a inferência são realizados usando aritméticas de inteiros de 8 bits. Maximizando a eficiência de executar no Edge TPU. Este processo geralmente requer um pequeno conjunto de dados de calibração para poder determinar os parâmetros ótimos de quantização.

O modelo final, yolov8n\_full\_integer\_quant\_edgetpu.tflite, é o arquivo que é efetivamente carregado e executado pelo *script* deteccao.py na Raspberry Pi com o auxílio do Edge TPU.

### Definição de Classes

O arquivo classes.txt é um arquivo de texto que lista o nome das classes que o modelo foi treinado para detectar. O *script* deteccao.py lê este arquivo para mapear os Ids numéricos das classes retornados pelo modelo para nomes legíveis por humanos, por exemplo: “pessoa”, “capacete”, “bota, “óculos”.

## Lógica de Detecção Implementada na Raspberry Pi

### Captura de Vídeo Otimizado com *Thread*ing

Utilizando o *script* deteccao.py, o programa principal que manuseia o processo de detecção.

A classe CapturaCamera foi criada para encapsular a lógica de captura de *frame*s da *webcam* usando OpenCV.

A principal otimização é o uso de *threading*. Uma *thread* separada é dedicada exclusivamente a ler os *frames* da câmera o mais rápido possível e manter o *frame* mais recente disponível. A *thread* principal do programa, que realiza a inferência e outras logicas, pode então acessar este *frame* mais recente sem ter que esperar pela operação de I/O (entrada/saída) da câmera, que pode ser bloqueado. Isso melhora a fluidez geral e a responsividade do sistema.

### Otimização do Processamento de *Frame*s

Analisando cada *frame* de um fluxo de vídeo com um modelo de IA pode ser computacionalmente intensivo, mesmo com um acelerador. Para poder gerenciar a carga de processamento e manter uma taxa de quadros razoável, o *script* implementa uma forma de pular *frames* (ou *frame skipping*). Ele processa um *frame* cada N *frame* capturados. No código, existem uma linha de comando que processa aproximadamente 1 a cada 6 *frame*s. Isso é um *trade-off*, essa técnica reduz a carga computacional e aumenta a taxa de quadro de processamento, mas introduz uma pequena latência na detecção.

### Pré-processamento de *Frame*s para inferência

Antes que um *frame* de vídeo possa ser alimentado na rede neural YOLOv8n, ele precisa ser formatado de acordo com os requisitos de entrada do modelo.

O OpenCV é usado para redimensionar o *frame* para as dimensões exatas que o modelo espera, no caso do projeto: 640 x 640 (px).

Uma outra etapa, como normalizar dos valores dos pixels, exemplo: para o intervalo [0, 1] ou [-1, 1] e a ordenação correta dos canais de cor (BGR ou RGB), são geralmente tratadas pela biblioteca ultralytics ou pelo interpretador TensorFlow Lite durante o carregamento do modelo e a preparação dos dados para inferência.

### Execução da Inferência com YOLOv8

Inferência é um termo técnico para o processo de usar um modelo de *machine learning* treinado para fazer previsões sobre novos dados que ele não viu durante o treinamento. “Fazer previsão” significa analisar um *frame* de vídeo e identificar os objetos presentes e suas localizações.

O *script* carrega o modelo .tflite otimizado para Edge TPU usando a biblioteca ultralytics.

### Delegação para Coral Edge TPU

Quando o *script* é executado em uma Raspberry Pi com um Coral Edge TPU conectado e os *drives* corretos instalados, o interpretador TensorFlow Lite, através da utilização da biblioteca ultralytics, automaticamente reconhece a presença do Edge TPU e delega a maior parte dos cálculos da rede neural para este acelerador. Isso resulta em uma significável aceleração em comparação com a execução apenas na CPU da Raspberry Pi.

Existe um comando que executa a inferência no *frame* redimensionado. O parâmetro conf=0.3 estabelece um limiar de confiança ou *confidence threshold*.

### Pós-processamento dos Resultados da Detecção

A saída do modelo YOLOv8n é uma estrutura de dados que contém informações sobre cada objeto detectado.

* + - 1. Caixas Delimitadoras

Um conjunto de coordenadas que definem um retângulo ao redor do objeto detectado no *frame*. Essas coordenadas são inicialmente relativas ao tamanho da imagem de entrada do modelo, no caso 640x640 px.

* + - 1. Identificador de Classe e Score de Confiança

O programa possui um arquivo que relaciona cada classe a um número. Esse número é usado para identificar os objetos detectados pela câmera.  
O score de confiança é um valor entre 0 e 1 (onde 1 equivale a 100%) e indica o quanto o modelo está seguro da detecção naquele instante.

* + - 1. Reescalonamento de Coordenadas e Visualização

As coordenadas das caixas delimitadoras são convertidas de volta para a escala do *frame* de vídeo original (redimensionamento para 640x640 px), para que possa ser desenhada corretamente na imagem exibida.

Utilizando funções do OpenCV, o *script* desenha as caixas delimitadoras ao redor dos objetos detectados diretamente no *frame* do vídeo. Rótulos contendo o nome da classe e o score de confiança são exibidos próximos a casa caixa. Cores diferentes são usadas para as caixas de classes diferentes para melhoras a distinção visual.

## Implementação do Sistema de Alerta e Notificação

Uma vez que os objetos são detectados, o sistema precisa de uma lógica para decidir quando uma situação de risco (no caso, a ausência de EPIs) ocorre e como notificação os responsáveis.

### Lógica de Identificação de Ausência de EPIs e Mitigar o Falso Positivo

A condição base para um alerta é a presença de pelo menos uma “pessoa” detectada no *frame*, se uma pessoa é detectada, o sistema então verifica a contagem dos EPIs relevantes (como: “capacete”, “óculos”, “bota”) que também foram detectados no mesmo *frame*. Uma contagem zero para um EPI que deveria estar presente indica uma não conformidade potencial.

### Tratamento de Falsos Positivos e Estabilidade da Detecção

Modelos de detecção de objetos, mesmo os precisos, podem ocasionalmente falhar em detectar um objeto que está presente (falso negativo) ou detectar um objeto que não está lá (falso positivo). Para poder evitar que alertas sejam disparados erraticamente devido a piscadas momentâneas na detecção (como por exemplo: um capacete que fica brevemente obstruído pela mão do trabalhador), o sistema implementa uma lógica de persistência temporal. Como o estado\_epi e duração\_para\_alerta.

O estado\_epi é um dicionário Python quem mantém o estado atual de cada tipo de EPI (ex: {‘capacete’: {‘ausente\_desde’: timestamp, ‘presente’: False}, ...}). Ele armazena se o EPI está sendo considerado ‘presente’ ou ‘ausente’ e, no caso de ausência, desde (ausente\_desde) essa ausência foi notada pela primeira vez.

A duracao\_para\_alerta verifica se um EPI só é classificado como ‘ausente’ para fins de disparo de alerta se ele permanecer não detectado por este período contínuo enquanto uma pessoa está presente. Similar, se um EPI está sendo considerado ‘ausente’ e volta a ser detectado, ele só é marcado como ‘presente’ novamente se for detectado continuamente por duracao\_para\_alerta. Isso adiciona uma “histerese” ao sistema, tornando os alertas mais estáveis.

### Geração e Formatação de Mensagens de Alerta Personalizadas

Quando a lógica de temporização confirma a ausência de um ou mais EPIs em uma pessoa detectada, uma mensagem de alerta textual é dinamicamente formatada. Esta mensagem é clara e informativa, listando especificamente quais EPIs está faltando (ex: “Pessoa sem os EPIs: capacete, óculos”).

### Mecanismo de Controle de Frequência de Alertas Anti-Spam)

Se uma pessoa permanece sem o EPI por um longo período, o sistema poderia gerar um fluxo contínuo de alertas, o que seria indesejável (spam de notificações). Para evitar isso, um controle de frequência foi implementado chamado de ultimo\_alerta e tempo\_entre\_alertas.

O ultimo\_alerta é um dicionário que armazena uma “chave” representando a combinação única de EPIs ausentes no último alerta enviado (ex: “capacete-óculos”) e o timestamp (data e hora) desse último alerta.

O tempo\_entre\_alertas é um novo alerta para a exatamente mesma combinação de EPIs faltantes só será reenviado se este intervalo de tempo tiver decorrido desde o último alerta idêntico. No entanto, se a combinação de EPIs ausentes mudar (ex: antes era só ‘capacete’ e ‘óculos’), um novo alerta é enviado imediatamente, pois representa uma nova situação de risco.

### Integração Robusta com Firebase Firestore para Registro de Alertas

O interagir com os serviços do Firebase a partir de um ambiente de *backend* ou de um dispositivo confiável como a Raspberry Pi, utiliza-se Firebase Admin SDK, ou SKD firebase-admin para Python. Este SDK permite operações privilegiadas, como escrita direta no banco de dados. A autenticação é feita através de um arquivo de credenciais de serviço (firebase\_key.json), que deve ser baixado do console do Firebase e armazenado de forma segura no dispositivo.

A função enviar\_alerta\_imagem no *script* detecção.py é responsável por persistir cada alerta confirmado no Firestore. Um novo documento é criado na coleção alertas\_epi, chamado de mensagem como a descrição textual do alerta (ex: “pessoa sem os EPI: capacete”). O data\_hora, um timestamp no formato padrão ISSO 8601 (ex: "2023-10-27T10:30:00.123456Z"). Usar um formato padronizado e com fuso horário (o ‘Z’ indica UTC) é crucial para a consistência e para facilitar consultas e ordenação cronológica dos alertas, especialmente se o sistema dor expandido para múltiplas localidades.

E, por último, a imagem\_base64 seria o *frame* de vídeo exato do momento em que o alerta foi gerado é primeiro codificado no formato de imagem JPEG (um formato de compressão com perdas, bom para fotografias e para reduzir o tamanho do arquivo) Em seguida, os dados binários dessa imagem JPEG são codificados em BASE64. Base64 é um esquema de codificação que transforma dados binários em uma sequência de caracteres ASCII. Isso é necessário porque o Firestore armazena dados brutos. A string Base64 pode ser facilmente armazenada e, posteriormente, decodificada de volta para uma imagem para visualização na interface *web*.

### Notificações Imediatas via Telegram

O sistema utiliza a API HTTP do Telegram Bot para enviar notificações. Esta é uma interface baseada em requisições *web* (principalmente GET ou POST) que permite a programas externos interagir com a plataforma Telegram através de um bot. A biblioteca requests do Python é usada para construir e enviar essas requisições HTTP para os edpoints da API do Telegram.

Duas funções são utilizadas, chamados de enviar\_telegram(mensagem) e enviar\_imagem\_telegram(imagem\_*buffers*, legenda). O enviar\_telegram(mensagem), ela envia a mensagem de alerta textual para o *chat* especificado. E enviar\_imagem\_telegram(imagem\_*buffers*, legenda) tem a função de enviar o *frame* de vídeo (como um *buffers* de bytes da imagem JPEG) como uma foto para o *chat*, usando a mensagem de alerta como legenda da foto.

As configurações necessárias, chamados de TELEGRAM\_*TOKEN* E ID\_*CHAT*\_TELEGRAM. TELEGRAM\_*TOKEN* é um *token* alfanumérico único fornecido pelo BotFather (o Bot oficial do Telegram para gerenciar outros Bots) quando você cria seu Bot. Este *token* é a “chave de acesso” do seu Bot. Por último, o TELEGRAM\_*CHAT*\_ID é um identificador numérico único do *chat* (seja uma conversa privada com um usuário ou um *chat* em grupo) para ode as notificações devem ser enviadas.

### Consideração de Segurança

Armazenar *token*s e Ids sensíveis diretamente no código não é uma prática recomendada para ambientes de produção. Alternativas mais seguras incluem o uso de variáveis de ambiente do sistema operacional ou arquivos de configuração externos que não são visionados no repositório de código.

### Bibliotecas Utilizadas

A explicação das bibliotecas utilizadas no código, listadas na figura 4.

Texto

O conteúdo gerado por IA pode estar incorreto.

Figura - Biblioteca

* **CV2(OpenCV)**  
  Essencial para todo o processamento de imagem e vídeo. É usada para capturar o vídeo da webcam, redimensionar frames, desenhar retângulos e texto na imagem, e exibir a janela de vídeo.
* **NumPy**  
  Fundamental para computação numérica. O OpenCV e o YOLO retornam dados (como frames e caixas de detecção) no formato de arrays NumPy.
* **Ultralytics**  
  É a biblioteca oficial dos modelos YOLO. A classe YOLO é usada para carregar o modelo .tflite e realizar as predições (inferências) nos frames da câmera.
* **Time**  
  Utilizada para medir o tempo de processamento (cálculo de FPS), controlar a frequência de logs e gerenciar os timers da lógica de análise e alerta.
* **Collections**  
  Fornece as estruturas de dados deque (fila de dois extremos) e defaultdict. O deque é usado para criar os buffers de histórico de detecção, que são essenciais para a lógica de estabilização.
* **Threading**  
  Permite que tarefas sejam executadas em paralelo. É usada para ler a câmera em uma thread separada (evitando que a leitura bloqueie o processamento) e para enviar alertas (Telegram, Firebase) sem travar o loop principal de detecção.
* **firebase\_admin**  
  SDK do Google para interagir com os serviços do Firebase. É usada para inicializar a conexão e para ler/escrever dados no Firestore.
* **Base64**  
  Usada para codificar a imagem do alerta em formato de texto (Base64) antes de salvá-la no Firebase. Isso permite armazenar a imagem diretamente em um campo de texto no banco de dados.
* **Requests**  
  Biblioteca para fazer requisições HTTP. É usada para enviar as mensagens de alerta para a API do Telegram.
* **Logging**  
  Para registrar informações, avisos e erros durante a execução. Ajuda na depuração e no monitoramento do que o sistema está fazendo.
* **OS(OperatingSystem)**  
  Funções do sistema operacional, usada principalmente para verificar se os arquivos de configuração e modelo existem (os.path.exists).
* **Enum**

Permite utilizar “rótulos” fixos para o estado do sistema, exemplo: EstadoEntrada.ANALISANDO ou EstadoEntrada.APROVADO, ao invés de utilizar números ou strings soltas que podem causar erros de digitação.

### Estrutura do Código

* + - 1. *Significados dos Estados*

Os Estados representam o “ciclo de vida” de uma pessoa passando pela câmera. O sistema não analisa cada foto isoladamente, sabendo em que momento da interação estamos.

* **VAZIO:** A câmera não vê ninguém, o sistema fica em modo de economia de energia apenas esperando um movimento.
* **Entrando:** Alguém apareceu na zona, mas ainda não sabemos se vai ficar, apenas alguém de passagem. O sistema inicia uma cronometragem, caso a pessoa permaneça na tela por 1,5 segundo, o sistema confirma que é entrada real e avançada, e caso a pessoa suma antes retorna para “VAZIO”.
* **ANALISANDO:** A pessoa parou na frente da câmera, por ser o momento de análise, o sistema começa a acumular dados nos históricos. Ele tira vários frames para ter certeza se o capacete e o óculos estão lá. Porém nenhuma decisão foi tomada ainda, apenas observação.
* **APROVADO:** Com o fim da análise e todos os EPIs forem detectados, a zona ficara verde na tela e o sistema para de analisar e apenas mantem o status positivo até a pessoa sair.
* **Rejeitado:** Com o fim da análise, EPIs estão faltando. Muda a cor da interface para vermelho, tira print para salvar a foto no Firebase e envia ao Telegram.
* **Saindo:** A pessoa, independentemente de ser rejeitada ou não, começa a sair da zona, o sistema espera um pouco antes de voltar para “VAZIO”.

Texto

O conteúdo gerado por IA pode estar incorreto.

Figura - Estados

* + - 1. CapturaCamera

Esta classe tem o objetivo de resolver um dos problemas mais difíceis e comuns em Visão Computacional com Python, que seria o lag de vídeo. Se usar OpenCV de forma simples, o processamento da IA das com que imagens velhas fiquem acumulando na memória da câmera. O resultado é que o vídeo na tela aparece com segundos de atraso em relação a vida real.

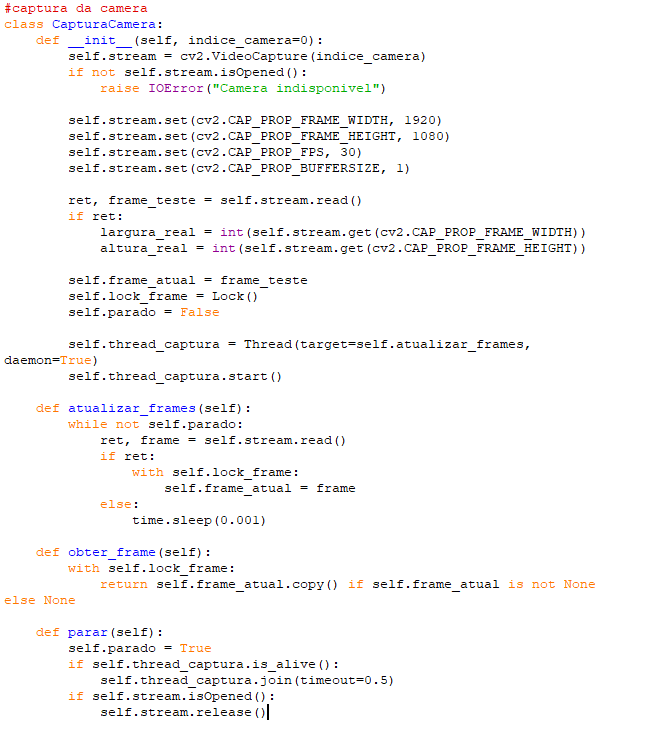


Figura – CapturaCamera

* + - 1. Config

A classe Config atua como o Painel de Controle Central do sistema, em engenharia de software, é má prática espalhar “números mágicos” pelo meio do código logico.

Ao agrupar tudo na Classe Config, você pode calibrar todo o comportamento do sistema em um único lugar.

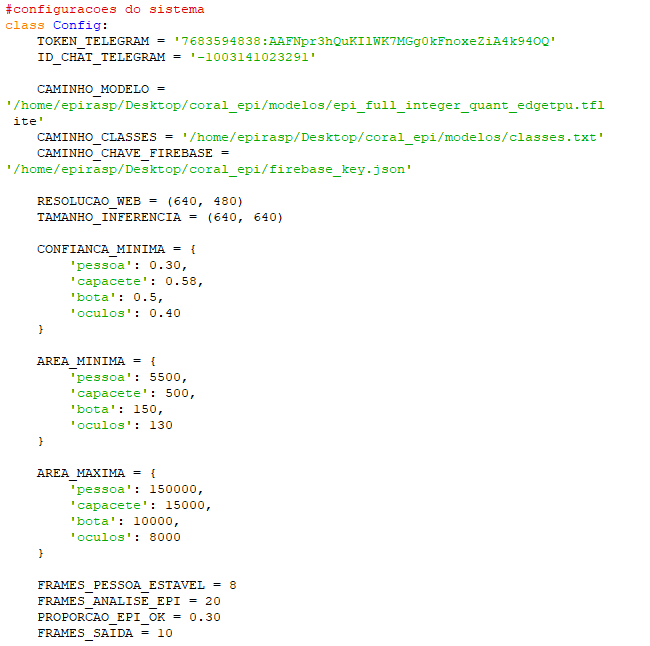


Figura – Config

* + - 1. RastreadorDeteccoes

Esta classe é “Estatístico” do sistema. Ela resolve um problema visual muito comum em IA, o Jitter (Tremulação).

Quando uma IA detecta objetos em vídeo, a caixa nunca fica parada, fica tremula, muda de tamanho e posição levemente a cada milissegundo

Se tiver uma foto aleatória para enviar no alerta, poderia pegar um momento em que a caixa esta torta ou “piscando”. A classe serve para acumular dados durante o período de análise e gera um resultado estável e limpo.

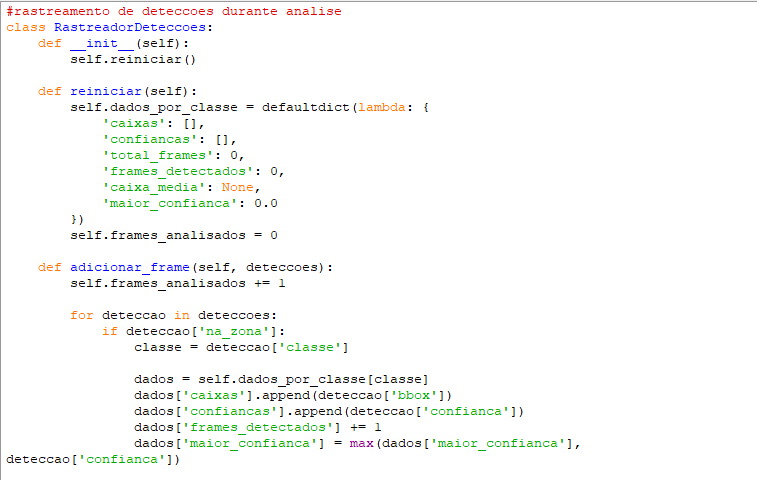


Figura - RastreadorDeteccoes

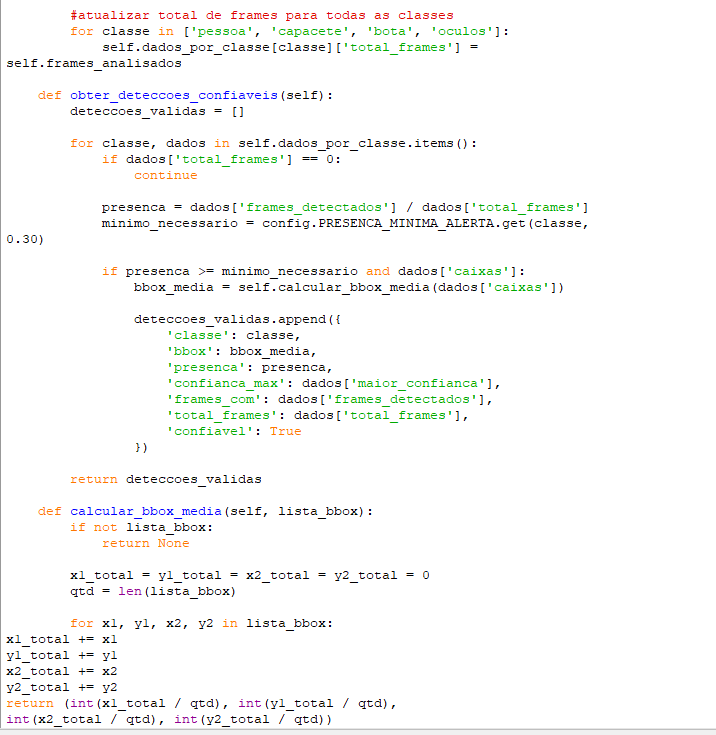


Figura - RastreadorDeteccoes

* + - 1. ControladorEPI

Esta classe seria a mais importante do sistema, ele seria o “Orquestrador Central”. Ele é quem delega as tarefas para as outras classes e toma as decisões finais, as figuras 10 e 11 mostram as classes e um pouco do código com comentários.

Texto

O conteúdo gerado por IA pode estar incorreto.

Figura - ControladorEPI

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Figura - ControladorEPI

* + - 1. GerenciadorTelegram

Esta classe funciona como o responsável por “Comunicar os Alertas” e pela “Inicialização Segura” do sistema.

* **MENSAGEIRO:** Ela possui a função de apenas despachar mensagens de texto para o aplicativo Telegram usando API oficial
* **Envio:** Ele monta o endereço da API inserindo o token que fica na Config, mostrado na Figura.
* **Ponto de Partida:** Ele não começa a rodar sozinho, antes de acionar a IA ela verifica se está tudo no lugar. O sistema verifica se o arquivo da IA e a lista de nomes realmente existam.
* **Ignição:** Quando o ControladorEPI() é chamado, ele dispara uma reação em cadeia, conecta no Firebase, liga a câmera, carrega a IA e prepara o Telegram. O executar() é onde o while True começa. O código ficara preso nesta linha ate alguém desligar o sistema. E o error, seria quando acontece o erro catastrófico que não foi previsto em nenhum lugar, exemplo: Na RAM. Ele captura o erro e salva no log antes do programa morrer.
* **Gatilho:** A função main é a que executa todo o processo do código.

Uma imagem contendo Texto

O conteúdo gerado por IA pode estar incorreto.

Figura - GerenciadorFirebase

## Desenvolvimento do *Site*

O *site* desenvolvido tem como objetivo principal fornecer uma interface *web* funcional, limpa e responsiva, destinada à visualização das ocorrências detectadas pelo sistema. A primeira página do *site,* mostrada na figura 13, é dedicada à autenticação do usuário, exigindo o preenchimento de nome de usuário e senha previamente definidos. Após a autenticação, o usuário é direcionado à página principal da aplicação.

Nesta página, é exibida automaticamente a ocorrência mais recente registrada no Firestore, incluindo a imagem, a classe da ocorrência (relacionada à ausência de determinado EPI) e a data e hora do evento. O sistema também conta com uma página de histórico, onde é possível visualizar uma galeria das últimas imagens registradas, permitindo o acompanhamento visual e cronológico das infrações detectadas.

A estrutura do *site* foi desenvolvida utilizando HTML para a marcação dos elementos, CSS para a estilização e responsividade, e Java*Script* para a implementação das funcionalidades dinâmicas, como a conexão com o Firebase Firestore e a renderização automática dos dados atualizados. A interface foi projetada com foco na simplicidade e usabilidade, permitindo que usuários com diferentes níveis de familiaridade técnica possam acessar as informações de forma clara e eficiente, seja por meio de computadores ou dispositivos móveis.

Tela de celular com fundo preto

O conteúdo gerado por IA pode estar incorreto.

Figura - Página inicial do site.

### Tecnologias Frontend Empregadas

* + - 1. HTML (HyperText Markup Language)

HTML tem a função de uma espinha dorsal de qualquer página *web*, definindo sua estrutura lógica e o conteúdo.

Elementos HTML como <div>, <nav>, <table>, <form>, <img>, etc, são usados par organizar e apresentar as informações.

* + - 1. Tailwind CSS

O Tailwind CSS é uma *frame*work CSS que adota uma abordagem “utility-first”. Em vez de fornecer componentes de interface pré-estilizados (como botões ou cards com aparências fixas), Tailwind oferece um vasto conjunto de classes de utilidade de baixo nível.

Essas classes são aplicadas diretamente nos elementos HTML para compor designs complexos e customizados de forma rápida e consistente. Facilita a criação de interfaces responsivas e a manutenção de um sistema de design coeso.

* + - 1. Java*Script* (Vanilla JS e Módulos ES6)

O Java*Script* é a linguagem de programação que roda no navegador do usuário, tornando as páginas *web* interativas e dinâmicas.

* + - 1. Firebase SDK para Java*Script*

O Firebase SDK é uma biblioteca fornecida pelo Firebase que permite que aplicações *web* frontend interajam de forma segura e simplificada com os diversos serviços do Firebase. Ele lida com a autenticação, a formatação das requisições para a API do Firebase e o tratamento das respostas.

### Funcionalidades Implementadas na Interface *Web*

A interface *web* é composta por um conjunto de páginas HTML, interconectadas, cada uma servindo a um propósito específico no monitoramento e gerenciamento do sistema.

* + - 1. Autenticação de Usuários

A página login.html apresenta ao usuário um formulário para inserir seu nome de usuário e senha. O design é moderno, utilizando Tailwind CSS para estilização. O *script* back.js contém a lógica de autenticação.

A captura de credenciais ocorre quando o formulário de login é submetido, o *script* obtém os valores dos campos de usuário e senha

Utilizando o Firebase SDK, o *script* realiza uma consulta à coleção users no Firebase, buscando por um documento onde o campo user corresponda ao nome de usuário fornecido.

A verificação de senha ocorre quando um usuário correspondente é encontrado, a senha fornecida é comparada com a senha armazenada no documento do Firebase.

Caso as credenciais são validas, informações básicas sobre o usuário logado (como seu nome de usuário e, seu ID de documento no Firebase) são armazenados no sessionStorage do navegador. O sessionStorege é uma API de armazenamento *web* que permite guardar pares de chave-valor que aberta. Isso permite que outras páginas de aplicações verifiquem se o usuário está logado.

O usuário é redirecionado para a página principal do painel (dsahboard.html).

Mensagens de sucesso ou erro (ex; “Usuário não encontrado”, “Senha incorreta”) são exibidas dinamicamente na página de login.

### Painel Principal

Esta página serve como ponto central de navegação e visualização de informações resumidas após o login bem-sucedido, ele apresenta um layout comum de dashboards, com uma barra de navegação lateral para acesso rápido a outras secções, como Histórico de Ocorrências, Gerenciamento de Usuários. E uma área de conteúdo principal.

Além da página apresentar elementos visuais como cartões de estatísticas, por exemplo: “Detecções Totais Hoje”, "Alertas Ativos no Mês", "Taxa de Conformidade de EPIs". E gráficos, como "Linha do Tempo de Detecções", "Distribuição de Tipos de EPIs Ausentes", contêm dados de exemplo diretamente no código HTML. Também apresenta um dashboard útil, adicionando uma consulta o Firebase Firestore para buscar dados reais de alertas, agregasse esses dados para calcular as estatísticas, utilizar uma biblioteca de gráficos Java*Script* para renderizar os gráficos com base nesses dados dinâmicos e uma secção de “Atividade Recente” exibindo exemplos estatísticos de alertas, que poderiam ser substituídos pelos últimos alertas reais do sistema.

### Visualização do Histórico de Ocorrências

Esta é uma funcionalidade crítica, permitindo aos administradores revisarem todos os alertas de não conformidade de EPIs que foram registrados pelo sistema.

O *script* carregarTabela.js é responsável pelas seguintes funções:

* A Busca de dados no Firestore que ao carregara página, o *script* usa o Firebase SDK para Java*Script* para buscar todos os documentos da coleção alertas\_epi no Firestore.
* A Ordenação de Dados que os alertas podem ser ordenados por data e hora, conforme a seleção do usuário em um controle na página.
* População Dinâmica da Tabela em que o *script* cria dinamicamente as linhas (<tr>) e células (<td>) da tabela HTML, preenchendo com os dados de cada alerta.
* A Visualização de Imagens em Modal que para cada alerta listado na tabela, um botão “Ver imagem” é incluído. Quando este botão é clicado o *script* identifica qual alerta corresponde ao botão clicado, ele recupera a string da imagem codificada em Base64 do objeto de dados daquele alerta. Um modal (uma janela de diálogo que se sobrepõe ao conteúdo da página, muitas vezes escurecendo o fundo) é exibido, a string Base64 é usada como s r c de um elemento <img> dentro do modal (prefixada com data:image/jpeg;base64), fazendo com que o navegador decodifique e renderize a imagem do incidente. O modal pode ser fechado pelo usuário.

### Gerenciamento de Contas de Usuário

Esta seção administrativa permite o controle de quem pode acessar o painel de monitoramento *web*. O *script* userManagement.js implementa a lógica completa de CRUD (Create, Read, Update, Delete).

Inicialização do Administrador Padrão para garantir que o sistema seja acessível imediatamente após a implantação, o *script* verifica, ao ser carregado pela primeira vez, se um usuário administrador padrão já existe na coleção *users* do Firestore. Se não existir, ele é criado automaticamente, as credenciais podem ser alteradas pelo administrador por motivos de segurança.

Adicionar novos usuários seria um formulário na página que permite que um administrador crie novas contas de usuários. O *script* valida se o nome de usuário já existe e se os campos de senha e confirmação de senha coincidem, caso estiver tudo correto, um novo documento de usuário é criado na coleção users do Firestore com o nome de usuário e a senha fornecida.

O *script* busca todos os documentos da coleção users e os exibe dinamicamente em uma lista na página.

Para cada usuário listado (com exceção do administrador padrão “adm”, pois é protegido contra a exclusão pela interface), um botão “Excluir” é fornecido. Ao clicar neste botão, o *script* remove o documento do usuário correspondente do Firestore.

As mensagens de sucesso (“Usuário adicionado com sucesso!”) ou erro (“Senhas não coincidem.”) são exibidas para o administrador após cada operação.

### Configuração Essencial e Hospedagem da Interface *Web*

* + - 1. Configuração do Firebase para o Frontend

Este arquivo Java*script* é absolutamente crucial para o funcionamento da interface *web*. Ele contém um objeto de configuração Java*Script* que inclui todas das chaves e identificadores necessários para que o SDK do Firebase para o Java*Script* possa se conectar ao projeto Firebase correto na nuvem.

Estes detalhes de configuração são obtidos diretamente do console do Firebase, na seção de configuração do projeto, ao registrar um aplicativo da *web*

O repositório inclui um firebaseConfig\_template.js que serve como modelo. O desenvolvedor deve copiar este template para um novo arquivo chamado firebaseConfig.js e preenchê-lo com os valores específicos do seu projeto Firebase.

* + - 1. Hospedagem com Firebase Hosting

O Firebase Hosting é o serviço usado para implantar e servir todos os arquivos estáticos que compõem a interface *web* (arquivos .html, .css, .js, e quaisquer imagens ou fontes). O processo de implantação é realizado através da Firebase CLI (Command Line Interface), uma ferramenta de linha de comando fornecida pelo Google.

Após configurar o Firebase Hosting no projeto (especificando qual diretório local contém os arquivos da interface *web* – neste caso, sistema\_de\_monitoramento/), o comando firebse deploy --only hosting envia esses arquivos para os servidores globais do Firebse.

O Firebse Hosting automaticamente provisiona um certificado SSL para o *site*, permitindo que ele seja acessado via HTTPS (conexão segura). Ele também utiliza uma CDN (Content Delivery Network) global para distribuir o conteúdo, o que significa que os usuários em diferentes partes do mundo podem acessar a interface *web* com baixa latência.

### Página de Login

A página de login (figura 13) permite o usuário a acessar o *site*, preenchendo o nome de usuário e senha, os dados são comparados com os documentos na coleção user do Firestore.

Se caso ocorra correspondência, o usuário é redirecionado para o *dashboard*, caso contrário é exibido uma mensagem de erro.

Tela de celular com fundo preto

O conteúdo gerado por IA pode estar incorreto.

Figura - Página de Login

### Página de Usuário

Permite gerar os usuários cadastrados. Permite o cadastro de novos usuários na coleção user, contendo campos como e-mail e senha como mostrados na figura 14.

Lista todos os usuários cadastrados em uma tabela dinâmica, e todos os dados são gravados e recuperados do Firestore.

Tela de celular

O conteúdo gerado por IA pode estar incorreto.

Figura - Página de Usuário

### Dashboard

Apresenta uma visão geral do sistema de monitoramento. Mostra alertas recentes de uso incorreto de EPI. Exemplo de como é exibido tal dashboard na figura 15.

Os alertas são buscados da coleção em alertas\_epi no Firestore, e cada alerta contém os campos data\_hora, mensagem e imagem\_base64. Os dados são exibidos em cards, gráficos e tabelas, com o visual moderno usando TailewindCSS.

Interface gráfica do usuário, Aplicativo, Teams

O conteúdo gerado por IA pode estar incorreto.

Figura - Dashboard

### Histórico de Ocorrência

Como mostrado nas figuras 16 e 17 sua função é listar todos os alertas registrados. Como recuperar a lista de documentos da coleção alertas\_epi ordenados por data, exibindo a data, mensagem e a imagem base64 associada a cada ocorrência. A imagem codificada é convertida automaticamente para visualização no HTML.

Tela de celular com aplicativo aberto

O conteúdo gerado por IA pode estar incorreto.

Figura - Histórico de Ocorrência

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

Figura – Imagem de Ocorrência

Na aba de configuração da zona de detecção mostrado na Figura 18 disponibiliza o desenho da zona de detecção sobre a imagem da câmera. Ao abrir a página, é carregada a imagem base64 mais recente da coleção alertas\_epi e exibida no centro, ela pode ser desenha sobre a imagem utilizando o mouse.

As coordenadas do retângulo são normalizadas para resolução 640x480 px e salvas na coleção configurações, documentos zona.

Tem a opção de limpar ou salvar uma nova zona. A imagem é carregada dinamicamente para que o administrador possa configurar com base em uma visualização real.

Tela de jogo de vídeo game

O conteúdo gerado por IA pode estar incorreto.

Figura - Página de Configuração

Como mencionado na Figura 19, a zona é configurada pelo administrador e os EPIs será detectado apenas dentro dela, caso estiver uma pessoal fora dela os EPIs não serão detectados.

### Encapsulamento para Raspberry Pi

Um encapsulamento foi feito no programa Autodesk Fusion 360 visto nas figuras 20 e 21, para poder colocar tanto a Raspberry quanto um circuito para poder converter a tensão para 12V, pois a Raspberry Pi 4 aguenta apenas 12V. O encapsulamento será feito de PLA(material usado pela impressora 3D), que tambem conterá uma ventoinha que auxiliará no resfriamento dos componentes, cada parte do encapsulamento pode ser vista nas figuras 22, 23 e 24.

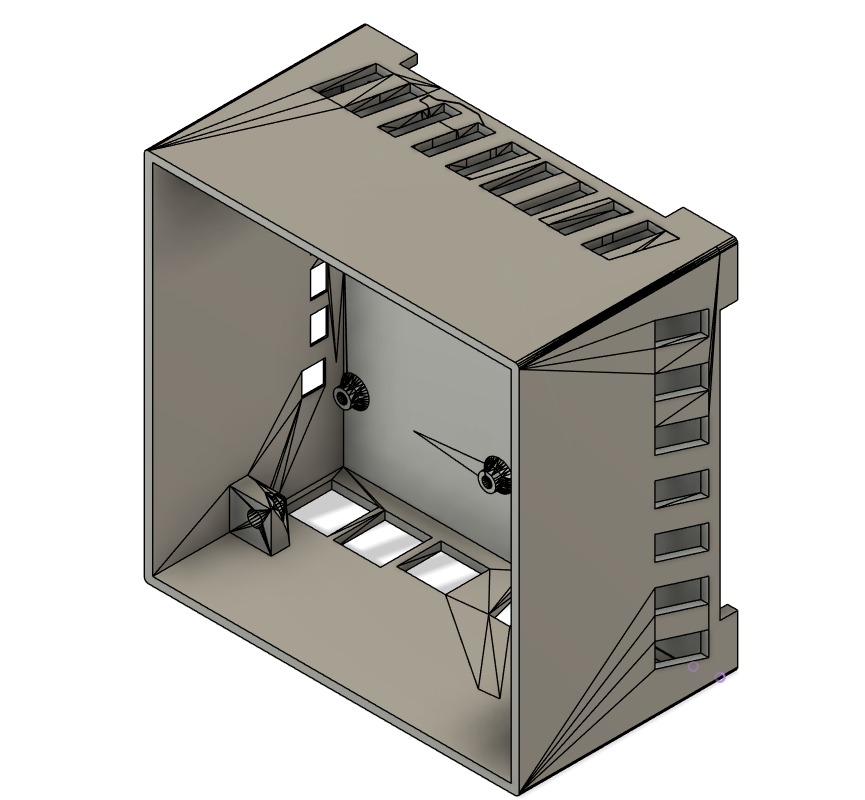


Figura - Modelo do encapsulamento

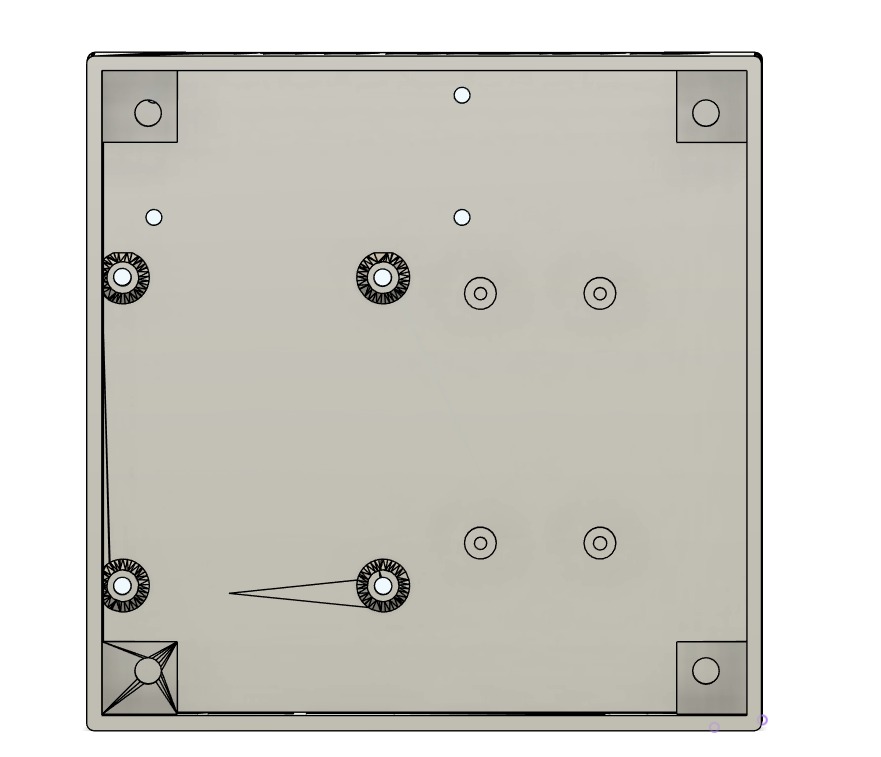


Figura - Imagem superior



Figura - Encapsulamento dos componentes fechado

Uma imagem contendo no interior, foto, mesa, velho

O conteúdo gerado por IA pode estar incorreto.

Figura 21 – Encapsulamento aberto com ventoinha

Figura - Encapsulamento aberto com ventoinha



Figura 22 – Encapsulamento aberto com componentes visíveis

Figura - Encapsulamento aberto com componentes visíveis

## Resultados e Testes

O sistema opera aproximadamente 5 metros de distância do alvo e 1,5 metros de altura. O sistema funciona de acordo como foi planejado, quando uma pessoa aparece na área por onde a câmera foi posicionada. Caso esta pessoa esteja utilizando capacete, óculos e bota o sistema detectara que está tudo certo, porém caso nenhum desses objetos seja detectado, é gerado uma ocorrência que é enviado ao banco de dados e, como mostrado na Figura 18, poderá ser vista no *site* criado.

Como demostrado na figura 25, mostra o desempenho do algoritmo na detecção dos EPIs, evidenciando taxas de falsos positivos e falsos negativos, o que é crucial para avaliar sai confiabilidade em alertar corretamente sobre ausência de capacetes, óculos ou botas.

A tabela abaixo, complementa a análise, da figura 25, com métricas com precisão, recall e FI-score, permitindo assim verificar se o sistema identifica EPIs com alta sensibilidade ou se foi registrado um alerta indevido.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Classe** | **Imagens** | **Instâncias** | **Precisão (P)** | **Revocação (R)** | **mAP50** | **mAP50-95** |
| **Todos** | **232** | **704** | **0,947** | **0,873** | **0,939** | **0,599** |
| Bota | 63 | 152 | 0,897 | 0,809 | 0,893 | 0,582 |
| Capacete | 80 | 138 | 0,969 | 0,914 | 0,959 | 0,624 |
| Óculos | 107 | 124 | 0,947 | 0,862 | 0,941 | 0,538 |
| Pessoa | 201 | 290 | 0,974 | 0,908 | 0,963 | 0,654 |

Tabela 1 – resultados de treinamento

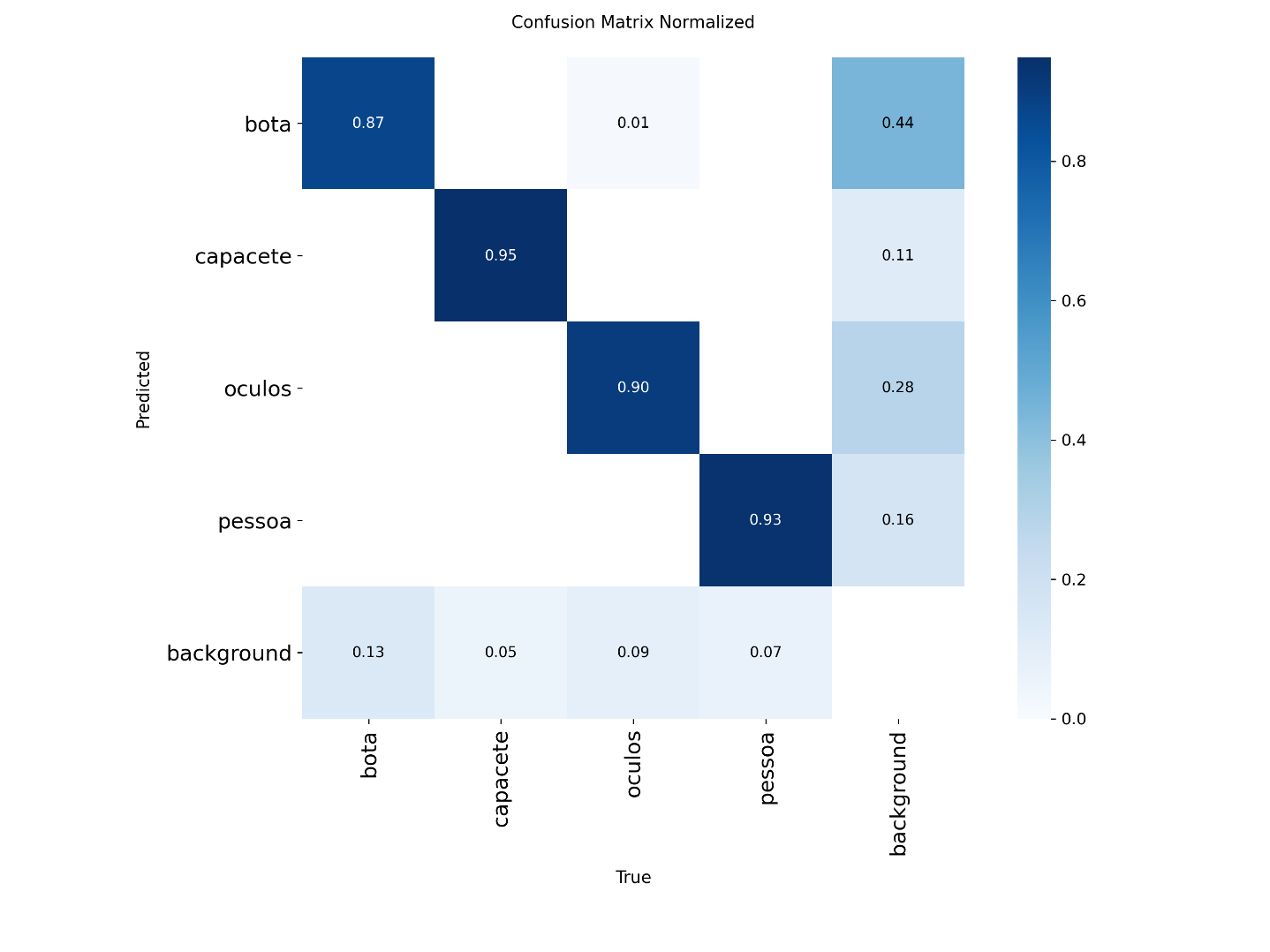


Figura - Matriz de confusão do modelo YOLOv8n

A tabela abaixo e nas figuras 26 e 27, apresenta o treinamento que foi feito para obter os resultados, a validação das imagens e os testes dos objetos encontrados.

Como mencionado na Figura 28, mostra o número de imagens colocadas, que foi 1283 imagens, o tamanho das imagens e o número de anotações feitas durante as imagens treinadas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class Name** | **Total Count** | **Training Count** | **Validation Count** | **Test Count** |
| pessoa | 1622 | 1455 | 110 | 57 |
| bota | 1034 | 922 | 69 | 43 |
| capacete | 832 | 745 | 48 | 39 |
| óculos | 717 | 675 | 31 | 11 |

Tabela 2 – resultados de treinamento

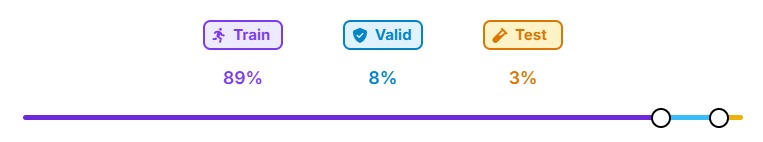


Figura - Treinamento, Validação e Testes

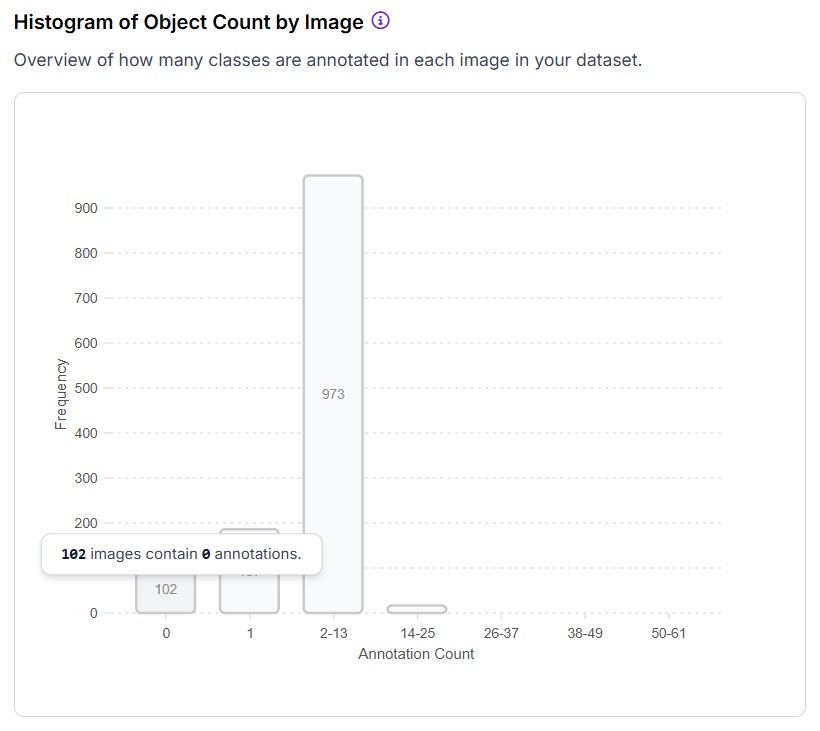


Figura - Contagem de Anotações por Imagens

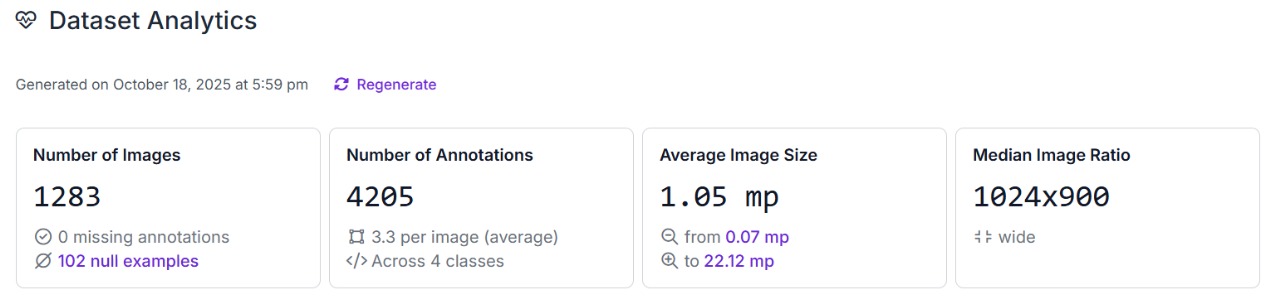


Figura - Análise do Número de Imagens

### Teste de otimização da detecção

Durante os testes com a Raspberry, foi utilizado um aparelho chamado Coral Edge TPU, que é um dispositivo compacto que adiciona um coprocessador Edge TPU no sistema. Permitindo um baixo consumo de energia e um desempenho maior para os modelos TensorFlow.

Utilizado para o arquivo .tflite, as imagens mostram um aumento significativo de FPS para a detecção da pessoa com os EPIs.

Como mostrado na figura 29, enquanto ele analisa a detecção da pessoa, o FPS fica variando de 7 para, no máximo, 9 FPS. Quando termina a análise, estabiliza no máximo 15 FPS (figura 30).



Figura - Analisando EPI



Figura - Análise completa e aprovado

Quando a análise fica como rejeitado, como por exemplo: não possuir nenhum EPI ou estiver faltando, a área de detecção fica vermelha e no fim da análise o FPS, também se estabiliza sendo no máximo 15 FPS e mínimo 7 FPS, assim como nas figuras 26 e 27.



Figura - Análise completa e rejeitada



Figura - Análise completa e rejeitada com dois EPIs

Enquanto a pessoa estiver entrando ou saindo da zona de detecção, o FPS fica variando de 7 a 10 FPS, pois está começando a analisar quem está na zona e detectando os objetos, como mostrado nas figuras 33 e 34.



Figura - Entrando na zona de detecção



Figura - Saindo da zona de detecção

Sem a Coral Edge TPU, o desempenho da detecção fica abaixo do esperado, mostrado na Figura 35 e Figura 36 quando ele analisa o FPS cai para abaixo de 1 FPS. Mesmo quando ele não analisa nada na zona de detecção, o resultado é o mesmo.



Figura - Análise sem a Coral



Figura - Zona de detecção vazia

### Teste com Windows

Foram realizados testes com a versão Windows da detecção de EPIs, feito através de um processador 13th Gen Intel(R) Core(TM) i5-13500 2.50 GHz, 16 GB RAM, pelo sistema operacional Windows 10 de 64 bits.

O resultado do desempenho otimizado por conta do *hardware* do sistema, o FPS chegou, no máximo, a 30 como mostrado nas figuras 37 e 38.

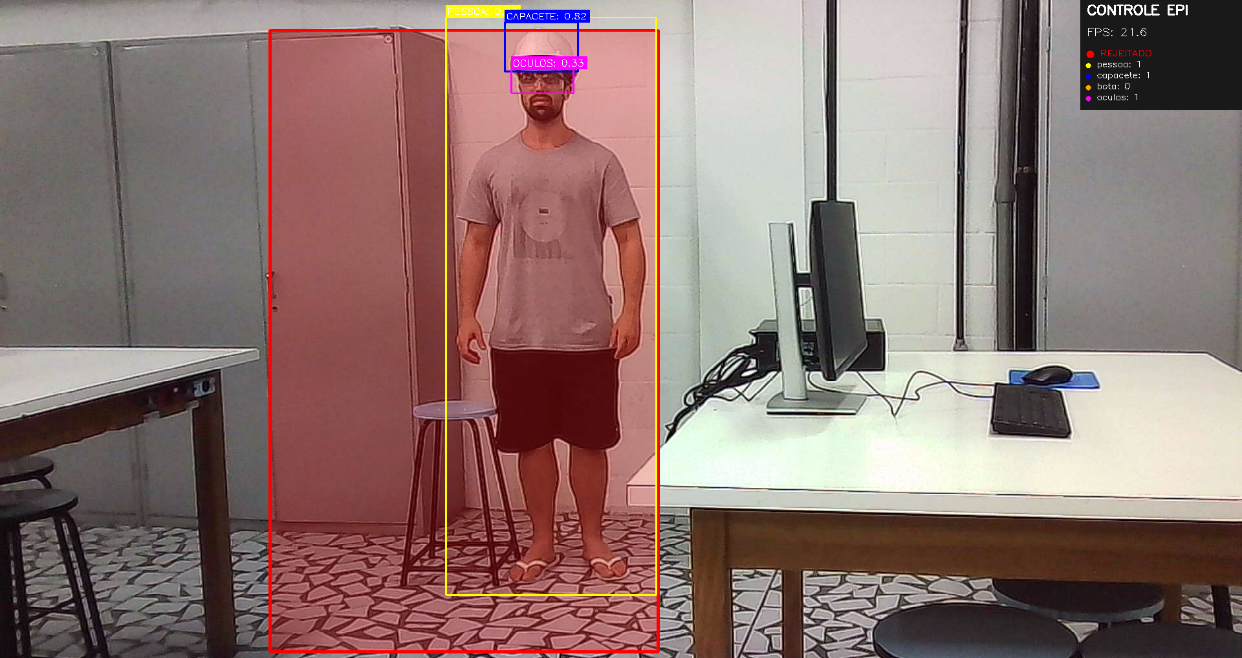


Figura - Análise de desempenho através do Windows



Figura - Zona de detecção vazia através do Windows

# Conclusão

O presente trabalho desenvolveu e implementou com sucesso um sistema automatizado de monitoramento do uso de Equipamentos de Proteção Individual (EPIs) baseado em Visão Computacional e Aprendizado de Máquina, atendendo plenamente aos objetivos estabelecidos de garantir a segurança dos trabalhadores e o cumprimento da Norma Regulamentadora NR6.

O sistema integrado, composto por uma Raspberry Pi 4 com acelerador USB Coral Edge TPU, câmera USB e modelo YOLOv8n otimizado para TensorFlow Lite, demonstrou robustez operacional em ambiente real. O modelo treinado alcançou métricas expressivas de desempenho: mAP@0.5 de 93,9%, mAP@0.5:0.95 de 59,9%, precisão geral de 94,7% e revocação de 87,3%, validando a eficácia da abordagem escolhida. Especificamente, o sistema obteve excelentes resultados na detecção de pessoas (97,4% de precisão), capacetes (96,9%) e óculos (94,7%), com desempenho ligeiramente inferior para botas (89,7%), indicando oportunidades de melhoria através do aumento do dataset para esta classe específica.

A arquitetura de *software* implementada, com destaque para a lógica de máquina de estados e os mecanismos de estabilização temporal, mostrou-se fundamental para mitigar falsos positivos e garantir alertas confiáveis. A implementação de *bufferss* de histórico, limiares de confiança configuráveis e períodos de análise ajustáveis permitiram que o sistema operasse de forma estável mesmo em condições variáveis de iluminação e posicionamento dos trabalhadores. O processamento em tempo real, mantendo média de 7 a 15 FPS durante análises com o acelerador Coral, comprovou a viabilidade técnica da solução em hardware embarcado de baixo custo.

A integração com Firebase Firestore para persistência de dados e a interface *web* desenvolvida em HTML, Java*Script* e Tailwind CSS proporcionaram uma experiência de usuário intuitiva e funcional. O sistema de autenticação, o *dashboard* com visualização de ocorrências em tempo real, o histórico completo com imagens codificadas em Base64, e a ferramenta de configuração de zonas de detecção demonstraram que a solução vai além da mera detecção, oferecendo uma plataforma completa de gestão de segurança do trabalho. As notificações instantâneas via Telegram adicionaram uma camada crítica de resposta rápida a situações de não conformidade.

Do ponto de vista prático, o sistema contribui significativamente para a segurança jurídica das empresas ao automatizar a fiscalização do uso de EPIs, reduzindo a dependência de inspeções manuais sujeitas a falhas humanas. A documentação fotográfica automática de cada ocorrência, com registro de data, hora e EPIs ausentes, fornece evidências objetivas para processos administrativos e educativos. Além disso, o sistema promove uma mudança cultural nas organizações, reforçando a importância da prevenção de acidentes e da proteção contínua dos trabalhadores.

Os testes realizados evidenciaram a importância do acelerador Coral Edge TPU, cuja ausência reduz o desempenho para menos de 1 FPS, inviabilizando o monitoramento em tempo real. Por outro lado, os testes em ambiente Windows com hardware superior (Intel Core i5-13500, 16GB RAM) atingiram 30 FPS, indicando que a arquitetura do software é escalável e pode se beneficiar de plataformas mais robustas quando necessário.

Como limitações identificadas, destaca-se a operação com câmera única, a dependência de condições adequadas de iluminação, e o desempenho inferior na detecção de botas em comparação aos demais EPIs. A distância operacional de aproximadamente 5 metros e altura de 1,5 metro, embora adequadas para os testes realizados, podem requerer ajustes conforme o layout específico de cada ambiente industrial.

Para trabalhos futuros, recomenda-se: (1) expansão do sistema para suportar múltiplas câmeras simultaneamente, permitindo cobertura de áreas maiores; (2) aprimoramento do dataset de treinamento, especialmente para a classe "botas", através da coleta de mais imagens em diferentes ângulos e condições; (3) implementação de técnicas de aumento de dados (data augmentation) para melhorar a robustez do modelo; (4) migração da arquitetura para um servidor dedicado com processamento centralizado acessível via nuvem, eliminando a dependência de hardware local; (5) inclusão de detecção de outros EPIs relevantes como luvas, protetores auriculares e máscaras respiratórias; (6) desenvolvimento de funcionalidades de análise preditiva e geração de relatórios estatísticos sobre padrões de conformidade; e (7) integração com sistemas corporativos de RH e segurança do trabalho para gestão holística de segurança ocupacional.

Em conclusão, o projeto demonstrou que a convergência de tecnologias de Visão Computacional, Aprendizado de Máquina e computação embarcada oferece uma solução viável, eficaz e economicamente acessível para o monitoramento automatizado de EPIs. O sistema desenvolvido representa um avanço significativo na prevenção de acidentes de trabalho, alinhando-se aos princípios da Indústria 4.0 e às exigências regulatórias brasileiras, com potencial de aplicação em diversos setores industriais que demandam alto rigor em segurança ocupacional.

# Referências

|  |  |
| --- | --- |
| 1. | EQUIPASEPI. EQUIPA. [Online].; 2023 [cited 2024 Abril 25. Available from: https://www.equipasepi.com.br/post/curiosidades-evolu%C3%A7%C3%A3o-dos-epis-ao-longo-da-hist%C3%B3ria. |
| 2. | DUAPI. DUAPI. [Online].; 2019 [cited 2024 Abril 25. Available from: https://duapi.com.br/blog/post/a-historia-dos-epis-16. |
| 3. | GOV.BR. GOV.BR. [Online].; 2023 [cited 2024 Abril 25. Available from: https://www.gov.br/trabalho-e-emprego/pt-br/acesso-a-informacao/participacao-social/conselhos-e-orgaos-colegiados/comissao-tripartite-partitaria-permanente/normas-regulamentadora/normas-regulamentadoras-vigentes/norma-regulamentadora-no-6-nr-6. |
| 4. | OIT. Organização Internacional do Trabalho. [Online].; 2023 [cited 2024 Abril 26. Available from: https://www.ilo.org/pt-pt/resource/news/quase-3-milhoes-de-pessoas-morrem-devido-acidentes-e-doencas-relacionados. |
| 5. | SONS W. WILSON SONS. [Online].; 2021 [cited 2024 Abril 26. Available from: https://www.wilsonsons.com.br/pt-br/blog/lei-do-epi-entenda-as-penalidades-para-quem-nao-cumpre/. |
| 6. | IBM. IBM. [Online]. [cited 2024 Junho 01. Available from: https://www.ibm.com/br-pt/topics/machine-learning. |
| 7. | Amazon. AWS. [Online]. [cited 2024 Maio 31. Available from: https://aws.amazon.com/pt/what-is/computer-vision/#:~:text=de%20vis%C3%A3o%20computacional%3F. |
| 8. | Red Hat. Red Hat. [Online].; 2017 [cited 2024 Junho 05. Available from: https://www.redhat.com/pt-br/topics/cloud-native-apps/what-are-cloud-applications. |
| 9. | Costa M. Alura. [Online]. [cited 2024 15 Agosto. Available from: https://www.alura.com.br/artigos/machine-learning. |
| 10. | Olhar Digital. Olhar Digital. [Online].; 2019 [cited 2024 Agosto 14. Available from: https://olhardigital.com.br/2019/02/18/noticias/raspberry-pi-o-que-e-para-que-serve-e-como-comprar/. |
| 11. | Raspberry Pi. Raspberry Pi. [Online]. [cited 2024 Agosto 14. Available from: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/?variant=raspberry-pi-4-model-b-4gb. |
| 12. | Google. Google Cloud. [Online].; 2024 [cited 2024 Agosto 15. Available from: https://cloud.google.com/bigquery/docs/inference-overview?hl=pt-br#:~:text=A%20infer%C3%AAncia%20de%20machine%20learning,de%20machine%20learning%20em%20produ%C3%A7%C3%A3o%22. |
| 13. | Awari. Awari. [Online].; 2023 [cited 2024 Agosto 16. Available from: https://awari.com.br/processamento-de-imagens-com-tensorflow-tecnicas-e-praticas/#:~:text=O%20TensorFlow%20oferece%20ferramentas%20e,formas%2C%20texturas%20ou%20caracter%C3%ADsticas%20visuais. |
| 14. | Databricks. Databricks. [Online]. [cited 202 Agosto 16. Available from: https://www.databricks.com/br/glossary/tensorflow-guide. |
| 15. | Coral. Coral. [Online]. [cited 2024 Agosto 13. Available from: https://coral.ai/products/accelerator. |
| 16. | https://blog.dsacademy.com.br/o-que-e-visao\_computacional/. Data Science Academy. [Online].; 2022 [cited 2024 Agosto 15. Available from: https://blog.dsacademy.com.br/o-que-e-visao\_computacional/. |
| 17. | PUC Rio. Maxwell. [Online]. [cited 2024 Agosto 15. Available from: https://www.maxwell.vrac.puc-rio.br/21365/21365\_6.PDF. |
| 18. | GOOGLE. Firebase. [Online].; 2025 [cited 2022. Available from: GOOGLE. Firebase Documentation. Disponível em: https://firebase.google.com/docs?hl=pt-br. |
| 19. | Alves G. Detecção de Objetos com YOLO – Uma abordagem moderna. IA Expert Academy ed.; 2020. |
| 20. | Ultralytics YOLO Documentos; 2024. |

‌