CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

LEANDRO NUNES DUARTE JÚNIOR

SISTEMA DE MONITORAMENTO E GERENCIAMENTO INTELIGENTE DA OCUPAÇÃO DE ESPAÇOS DE TRABALHO COM VISÃO COMPUTACIONAL

Leopoldina

2025

LEANDRO NUNES DUARTE JÚNIOR

SISTEMA DE MONITORAMENTO E GERENCIAMENTO INTELIGENTE DA OCUPAÇÃO DE ESPAÇOS DE TRABALHO COM VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Controle e Automação do Centro Federal de Educação Tecnológica de Minas Gerais, do Campus Leopoldina, como parte do requisito para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Orientador: Marlon Lucas Gomes Salmento

Leopoldina

2025

Duarte Júnior, Leandro Nunes.
D812s Sistema de monitoramento e gerenciamento inteligente da ocupação de espaços de trabalho com visão computacional / Leandro Nunes Duarte Júnior – 2025.
71 f.. : il.

Orientador: Marlon Lucas Gomes Salmento
Bibliografia.

Trabalho de Conclusão de Curso (Graduação) – Centro Federal de Educação Tecnológica de Minas Gerais. *Campus* Leopoldina. Engenharia de Controle e Automação, Departamento de Eletroeletrônica, 2025.

1. Espaço escolar 2. Internet das Coisas.. 3. Minicomputadores. 4. Visão computacional. I. Salmento, Marlon Lucas Gomes. II. Título.
CDU: 681.53

Elaboração da ficha catalográfica pelo bibliotecário Gabriel Jorge Rodrigues Oliveira
CRB 6º 4392 / CEFET-MG c*ampus* Leopoldina.

MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO - LP

FOLHA DE APROVAÇÃO DE TCC Nº 31/2025 - CECALP (11.51.20)

Nº do Protocolo: NÃO PROTOCOLADO

Leopoldina-MG, 15 de dezembro de 2025.

LEANDRO NUNES DUARTE JUNIOR

SISTEMA DE MONITORAMENTO E GERENCIAMENTO INTELIGENTE DA OCUPAÇÃO DE ESPAÇOS DE TRABALHO COM VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Controle e Automação do Centro Federal de Educação Tecnológica de Minas Gerais, do Campus Leopoldina, como parte do requisito para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Aprovado em 17 de dezembro de 2025.

Banca Examinadora:

Prof. D. Eng. Marlon Lucas Gomes Salmento (CEFET-MG) - Presidente
Prof. D. Sc. Vinícius Barbosa Schettino (CEFET-MG)
Prof. D. Sc. Fabiano Pereira Bhering (CEFET-MG)

Leopoldina
2025

(Assinado digitalmente em 19/12/2025 16:29)
FABIANO PEREIRA BHERING
PROFESSOR ENS BASICO TECN TECNOLOGICO
DECOMLP (11.61.11)
Matrícula: ###564#0

(Assinado digitalmente em 20/12/2025 09:00)
MARLON LUCAS GOMES SALMENTO
PROFESSOR ENS BASICO TECN TECNOLOGICO
DEELP (11.61.04)
Matrícula: ###575#9

(Assinado digitalmente em 18/12/2025 09:18)
VINICIUS BARBOSA SCHETTINO
PROFESSOR ENS BASICO TECN TECNOLOGICO
DEELP (11.61.04)
Matrícula: ###138#3

Processo Associado: 23062.066192/2025-54

Visualize o documento original em https://sig.cefetmg.br/public/documentos/index.jsp informando seu número: **31**, ano: **2025**, tipo: **FOLHA DE APROVAÇÃO DE TCC**, data de emissão: **15/12/2025** e o código de verificação: **eb054c04d0**

AGRADECIMENTOS

Agradeço primeiramente a Deus pela oportunidade de viver mais essa experiência, pela força nos momentos de dificuldade e pelo privilégio de conhecer pessoas incríveis ao longo dessa caminhada.

Agradeço ao meu orientador, Marlon Lucas, pela orientação essencial para a conclusão deste trabalho, pela paciência e pelo profissionalismo.

Agradeço à minha família, que compreendeu minha ausência em casa durante fins de semana e feriados dedicados aos estudos — ao meu pai Leandro, à minha mãe Viviane, à minha irmã Jéssica e aos meus irmãos Matheus e Marcos, que em breve também serão engenheiros de controle e automação.

Agradeço também aos meus grandes amigos e exemplos: Lucas Guimarães, Rafael Riguete, Yago Marino, Mateus Costa, Daniel Victor, João Marcos, Lucas Melo, Paloma Souza, Karen Barbosa e Willian Rodrigues, que foram essenciais nessa caminhada. Quando decidi deixar meu trabalho para ingressar na faculdade, recebi a ajuda de um amigo que, inclusive, me deu aulas de logaritmos para a prova do ENEM. Obrigado, Vinicio Souza Coimbra — sua ajuda foi fundamental no início dessa jornada.

SISTEMA DE MONITORAMENTO E GERENCIAMENTO INTELIGENTE DA OCUPAÇÃO DE ESPAÇOS DE TRABALHO COM VISÃO COMPUTACIONAL

Leandro Nunes Duarte Júnior

Marlon Lucas Gomes Salmento

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema inteligente para monitoramento e gerenciamento de ocupação em salas de aula, utilizando Visão Computacional e Internet das Coisas. O sistema é composto por um hardware principal com uma câmera USB conectada a um Raspberry Pi 5, que utiliza o modelo de detecção de objetos e pessoas YOLOv8 para monitorar a ocupação das salas em tempo real. Hardwares secundários baseados em ESP-32 recebem comandos para controlar dispositivos eletroeletrônicos, como iluminação e monitores. As informações obtidas pelo sistema são enviadas para uma API, que centraliza os dados de ocupação em um banco de dados e também atualiza uma página web com visualização de informações de ocupação das salas. Durante os testes realizados em ambiente real, o sistema apresentou desempenho consistente em condições de iluminação adequadas e variação mínima no tempo de resposta, mantendo o sistema sempre atualizado, obtendo nos testes um erro médio absoluto de apenas 0,28 pessoa na comparação entre a média real calculada manualmente e a média calculada pela solução. Além disso, observou-se que o controle automático de dispositivos respondeu corretamente ao estado de ocupação detectado.

Palavras-chave: gestão de ocupação; gestão de espaços; visão computacional; YOLOv8, Raspberry Pi; Internet das coisas.

INTELLIGENT SPACE MANAGEMENT USING COMPUTER VISION: A room occupancy detection, monitoring and control system using YOLOv8, Raspberry Pi and ESP-32

Leandro Nunes Duarte Júnior

Marlon Lucas Gomes Salmento

ABSTRACT

This work presents the development of an intelligent system for monitoring and managing occupancy in classrooms using Computer Vision and the Internet of Things. The system is composed of a main hardware unit with a USB camera connected to a Raspberry Pi 5, which employs the YOLOv8 object and person detection model to monitor room occupancy in real time. Secondary hardware units based on the ESP-32 receive commands to control electronic devices such as lighting and displays. The information obtained by the system is sent to an API that centralizes occupancy data in a database and also updates a web page displaying occupancy information for the rooms. During tests conducted in a real environment, the system demonstrated consistent performance under adequate lighting conditions and minimal variation in response time, keeping the system continuously updated. The tests showed a mean absolute error of only 0.28 person when comparing the manually calculated real average with the average produced by the solution. In addition, the automated device control responded correctly to the detected occupancy state.

Keywords: occupancy management; energy efficiency; computer vision; YOLOv8; Raspberry Pi; Internet of Things.

MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO - LP

TERMO DE RESPONSABILIDADE Nº 288/2025 - CECALP (11.51.20)

Nº do Protocolo: NÃO PROTOCOLADO

Leopoldina-MG, 15 de dezembro de 2025.

Eu, **LEANDRO NUNES DUARTE JUNIOR**, matrícula **201713510243**, discente do Curso de Engenharia de Controle e Automação, declaro para os devidos fins, que o presente Trabalho de Conclusão de Curso, de título **SISTEMA DE MONITORAMENTO E GERENCIAMENTO INTELIGENTE DA OCUPACÃO DE ESPAÇOS DE TRABALHO COM VISÃO COMPUTACIONAL,** é de minha autoria e que estou ciente dos:

- Artigos 297 a 299 do Código Penal;
- Decreto-Lei n.º 2.848 de 7 de dezembro de 1940;
- Lei n.º 9.610, de 19 de fevereiro de 1998, sobre os Direitos Autorais;
- Regulamento Disciplinar Discente do CEFET-MG;
- E que plágio consiste na reprodução de obra alheia e submissão dela como trabalho próprio ou na inclusão, em trabalho próprio, de ideias, textos, tabelas ou ilustrações (quadros, figuras, gráficos, fotografias, retratos, lâminas, desenhos, organogramas, fluxogramas, plantas, mapas e outros) transcritos de obras de terceiros sem a devida e correta citação da referência.

Por ser verdade, e por ter ciência do referido artigo, firmo a presente declaração, isentando o (a) professor(a) **D. Eng. Marlon Lucas Gomes Salmento**(orientador), os membros da banca examinadora e o Centro Federal de Educação Tecnológica de Minas Gerais, Campus Leopoldina, de qualquer responsabilidade.

(Assinado digitalmente em 18/12/2025 10:40)
Leandro Nunes Duarte Júnior
DISCENTE
Matrícula: 2017#######3

Processo Associado: 23062.066192/2025-54

Visualize o documento original em https://sig.cefetmg.br/public/documentos/index.jsp informando seu número: **288**, ano: **2025**, tipo: **TERMO DE RESPONSABILIDADE**, data de emissão: **15/12/2025** e o código de verificação: **0366c453e9**

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

CEP Código de Endereçamento Postal

CSS Cascading Style Sheets

CNN Convolutional Neural Network

HOG Histogram of Oriented Gradients

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

IoT Internet of Things

LED Light Emitting Diode

PIR Passive Infrared

PCB Printed Circuit Board

REST Representational State Transfer

SSD Single Shot MultiBox Detector

SVM Support Vector Machine

UVC USB Video Class

YOLO You Only Look Once

YOLOv8 You Only Look Once versão 8

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização do Tema

Com o crescimento dos centros urbanos e a diversificação das rotinas tanto educacionais quanto corporativas, impulsionadas por modelos presenciais, híbridos e remotos, cresce também a necessidade de melhorar o gerenciamento dos espaços físicos. Ambientes como salas de aula, laboratórios, escritórios, salas de reunião e áreas de uso coletivo frequentemente não são utilizados conforme sua capacidade. Isso pode resultar em salas subutilizadas, ocupações acima da capacidade em determinados horários, desperdício de energia e dificuldades no planejamento da alocação de recursos.

Tanto gestores educacionais quanto administradores de ambientes corporativos geralmente não dispõem de informações em tempo real sobre a ocupação dos espaços, o que torna complexa a tomada de decisões relacionadas à organização de horários, distribuição de equipes, reserva de salas, economia de energia e dimensionamento adequado de recursos.

Avanços em tecnologias de Internet of Things (IoT) e Visão Computacional tornam possível enfrentar esses desafios. Sistemas inteligentes capazes de detectar a presença de pessoas e gerar dados sobre a ocupação dos ambientes, permitem um melhor gerenciamento, automatizado e baseado em evidências. Essas soluções possibilitam a melhoria do uso de salas, a redução de desperdícios, a melhoria na experiência dos usuários e a geração de informações de ocupação valiosas para planejamento acadêmico ou corporativo.

1.2 Justificativa

Este trabalho foi desenvolvido com o propósito de criar uma solução prática e aplicável a diferentes cenários, capaz de gerar resultados relevantes no contexto de gestão inteligente de ambientes. Além disso, também com o objetivo de disseminar o conhecimento sobre o funcionamento de sistemas modernos, que normalmente integram *Application Programming Interface (API)*, banco de dados, página web e hardware embarcado, comunicando-se de maneira estruturada para garantir o correto funcionamento da solução. Essa combinação de tecnologias está presente em grande parte das aplicações utilizadas no cotidiano.

Outro ponto de destaque é demonstrar a viabilidade do uso de um modelo de detecção treinado operando em um hardware com recursos limitados, o que demonstra seus desafios

durante a execução do projeto. A integração desses elementos resulta em uma aplicação funcional e relevante, com potencial de uso real.

Por fim, o sistema proposto abre caminho para novas soluções baseadas no mesmo princípio, permitindo melhorias, adaptações e expansões que podem originar aplicações ainda melhores e inovadoras, contribuindo para facilitar e aprimorar a experiência das pessoas que as utilizam.

1.3 Objetivo Geral

O presente trabalho tem por objetivo desenvolver um sistema de monitoramente e ocupação em ambientes educacionais e corporativos, combinando detecção de pessoas por imagem e integração com API, banco de dados e página web, além de permitir o controle on/off de alguns dispositivos elétroeletrônicos.

É esperado que a implementação deste sistema demonstre que a integração de tecnologias modernas pode resultar em gestão inteligente de ambientes, fornecendo dados valiosos para aumentar o aproveitamento na utilização de salas de aula e escritórios, melhorando a experiência de gestores, colaboradores, professores e alunos, e apoiar a tomada de decisões baseada em informações confiáveis.

1.4 Objetivos Específicos

Os seguintes objetivos específicos foram definidos com o intuito de alcançar o objetivo geral:

1. Detectar presença de pessoas utilizando *hardware* embarcado.
2. Enviar dados de ocupação via API.
3. Manter dados de ocupação armazenados em banco de dados.
4. Desenvolver interface web de monitoramento de dados em tempo real e demonstração de dados estatísticos.
5. Controlar o funcionamento de dispositivos eletroeletrônicos no fomato on/off.

1.5 Estrutura do Trabalho

Este documento está dividido em seis capítulos principais, além de seções como resumo e referências, e está estruturado conforme as seguintes seções:

- **Capítulo 1 -Introdução:** Apresenta a motivação para desenvolvimeno do trabalho, bem como a justificativa do tema e os objetivos com o desenvolvimento da solução.
- **Capítulo 2 -Referencial Teórico:** Mostra os trabalhos utilizados como referência e embasamento teórico para a escolha e utilização das tecnologias utilizadas, como visão computacional, IoT e o modelo *You Only Look Once versão 8 (YOLOv8)*.
- **Capítulo 3 -Metodologia:** Descreve a implementação básica do sistema, os componentes utilizados e a arquitetura proposta.
- **Capítulo 4 -Desenvolvimento:** Descreve com detalhes os passos utilizados para o desenvolvimento do sistema, a parte física do hardware e as tecnogias utilizadas, além de explicar o funcionamento do firmware e dar detalhes sobre as principais estruturas de dados utilizadas.
- **Capítulo 5 -Testes e Análises dos Resultados:** Demonstra os resultados alcançados com a implementação em período de teste da solução, descrevendo o cenário utilizado nos testes e as métricas avaliadas, além dos resultados obtidos na página web.
- **Capítulo 6 -Conclusões e trabalhos futuros:** Traz um resumo dos resultados alcançados, destacando as contribuições do trabalho e apontando perspectivas futuras.

2 REFERENCIAL TEÓRICO

Neste capítulo serão abordadas as referências utilizadas para construção do projeto. Será apresentada uma visão geral sobre detecção de pessoas, protocolos de comunicação entre hardwares, páginas Web e API.

2.1 Detecção de pessoas

O ponto chave para a execução do projeto se dá na detecção de pessoas de forma confiável e na comunicação entre os dispositivos presentes na sala. Para isso, foi realizada uma pesquisa sobre diferentes formas de detectar pessoas, seja por sensores ou por modelos treinados de visão computacional.

Alguns sensores presentes no mercado oferecem soluções interessantes, mas possuem a desvantagem de um raio de detecção relativamente pequeno ou de detectarem apenas movimentos. É o caso dos sensores da família LD2410, como o modelo HI-Link LD2410C (Hi-Link Electronic, 2023), que detectam a presença humana a distâncias de até 5 metros, e dos sensores Passive Infrared (PIR), que se limitam à detecção de movimentos. Essas limitações tornam esse tipo de solução inadequada para ambientes maiores, como grandes escritórios ou salas de aula.

Essa restrição não é observada no modelo de detecção YOLOv8 (ULTRALYTICS, 2025), que foi escolhido por seu desempenho, detectando pessoas com rapidez, alcançando distâncias de detecção de 20 a 30 metros nos testes executados, além da possibilidade de ser executado em dispositivos com baixo poder de processamento.

A distância de detecção, no entanto, depende de diversos fatores relacionados à câmera, como tipo de lente, foco, resolução das imagens, entre outros. Câmeras de maior resolução permitem detectar pessoas a distâncias maiores, porém exigem mais capacidade de processamento do hardware, enquanto resoluções menores podem limitar esse alcance, mas manter um bom desempenho e rapidez na detecção. As decisões sobre esses parâmetros devem levar em conta o tipo de aplicação e o hardware envolvido.

O artigo Terven, Córdova-Esparza e Romero-González (2023) apresenta uma revisão abrangente das arquiteturas *You Only Look Once (YOLO)* para detecção de objetos em visão computacional. Ele destaca a evolução desde a versão inicial, YOLOv1, até as mais recentes, como YOLOv8 e YOLO-NAS, além de mencionar aplicações práticas em áreas como vigilância e veículos autônomos.

O artigo de Chatterjee, Singh e Agarwal (2024) apresenta um sistema de segurança que utiliza YOLOv8 (ULTRALYTICS, 2025), Python (FOUNDATION, 2025) e OpenCV (OpenCV.org, 2025) para detectar a presença de indivíduos em áreas restritas ou protegidas, descrevendo a arquitetura da solução e realizando comparações com outros modelos. A aplicação desenvolvida neste TCC adota algumas das ferramentas empregadas no trabalho citado, incluindo Python (FOUNDATION, 2025) como linguagem de programação principal e OpenCV (OpenCV.org, 2025) para manipulação das imagens capturadas pela câmera.

2.2 Modelo YOLO de detecção de pessoas

A detecção de pessoas é um campo da visão computacional amplamente estudado devido à sua aplicabilidade em segurança, monitoramento e automação. Historicamente, métodos tradicionais baseados em extração de características e aprendizado de máquina, como o uso de *Histogram of Oriented Gradients (HOG)* combinado com *Support Vector Machine (SVM)* eram comuns. Embora eficazes em cenários simples, esses métodos apresentavam limitações significativas em ambientes complexos, com desafios como variações de iluminação, oclusões e posições variadas das pessoas.

Com o avanço das redes neurais profundas, surgiram modelos mais robustos e precisos. Redes convolucionais abriram caminho para a criação de modelos que extraem automaticamente características relevantes das imagens. Entre esses modelos, destacam-se os sistemas baseados em detecção de objetos, como *Faster R-CNN* (REN et al., 2015), *Single Shot MultiBox Detector (SSD)* (LIU et al., 2016) e a série *YOLO* (REDMON et al., 2016).

O YOLOv8 (ULTRALYTICS, 2025), representa um marco na evolução desses modelos. Ele utiliza uma abordagem unificada para realizar a detecção de objetos em tempo real, analisando toda a imagem de uma só vez. Essa característica torna o YOLOv8 uma boa opção para aplicações em cenários que exigem processamento em tempo real, como monitoramento contínuo em sistemas embarcados. Além disso, a arquitetura foi aprimorada para balancear precisão e velocidade, sendo capaz de detectar objetos com alta confiança mesmo em dispositivos de hardware limitado.

O presente trabalho não foca em como é a estrutura do modelo, mas sim em como o mesmo pode ser utilizado. Porém em Terven, Córdova-Esparza e Romero-González (2023), temos uma análise mais detalhada da evolução da família de modelos YOLO e como é a arquitetura de sua rede. A partir dessa documentação podemos dizer que o YOLO funciona da

seguinte forma:

- Divide a imagem em uma "grade"de células e cada célula fica responsável por prever objetos parcialmente ou totalmente dentro delas.
- Prevê caixas delimitadoras *(bounding boxes)* que indicam onde o objeto está na imagem, fornecendo pontuações de confiança para cada caixa.
- Para cada caixa, o YOLO atribui uma classe de objeto detectado como "gato", "cachorro", "pessoa", etc.
- Filtra previsões, removendo caixas delimitadoras redundantes ou com baixa confiança.

O YOLO não trabalha diretamente com pixels individuais, mas utiliza uma *Convolutional Neural Network (CNN)* Goodfellow, Bengio e Courville (2016), para extrair características visuais, como formas e texturas, a partir de blocos de pixels. Essas características são usadas para prever caixas delimitadoras e classificar objetos. Por essa razão, o YOLO se destaca como uma ferramenta de alto desempenho para sistemas com monitoramento em tempo real, como o desenvolvido neste trabalho.

2.3 Protocolos de comunicação entre Hardwares

Ao definir o método de detecção de pessoas, foi necessário escolher e implementar a comunicação entre o hardware principal com os hardwares secundários para o acionamento dos dispositivos eletroeletrônicos presentes na sala de aula.

A primeira opção de protocolo cogitada foi a de utilizar o protocolo MQTT (OASIS, 2015) para comunicação entre o hardware principal e os hardwares secundários. Porém, esse protocolo necessita de um Broker para fazer o gerenciamento da comunicação entre tópicos, podendo ser *online* ou *offline*. O Broker seria instalado no hardware principal e todos os dispositivos se comunicariam através da rede local, o que não seria vantajoso, pois mais um hardware (roteador) seria acrescentado à solução, e problemas nessa rede poderiam ocasionar falhas de comunicação. O artigo Čabarkapa et al. (2024) demonstra o desenvolvimento de um sistema que combina um aplicativo móvel e web para regular as condições ideais em casas, como temperatura, umidade e iluminação. A solução integra a utilização do microcontrolador ESP-8266 com a comunicação através do protocolo MQTT, ajustando a operação de sistemas de iluminação de acordo com os requisitos do usuário.

A opção escolhida para o presente TCC foi o protocolo ESP-NOW (SYSTEMS, 2025), um protocolo de comunicação sem fio desenvolvido pela *Espressif Systems* que permite a comunicação direta entre dispositivos dessa fabricante, como o ESP-32 e ESP-8266, sem a necessidade de um roteador Wi-Fi. Este protocolo se destaca pelas respostas rápidas, com baixa latência e baixo consumo de energia. Os hardwares escolhidos para controlar os dispositivos eletroeletrônicos utilizam o ESP-8266 como microcontrolador, tornando possível a utilização desse protocolo na aplicação, conectando um ESP-32 via conexão USB ao hardware principal. Esse hardware envia mensagens ao ESP-32 através da conexão USB e o mesmo envia as informações aos outros ESPs através do protocolo ESP-NOW (SYSTEMS, 2025). Em Paguay et al. (2023) temos o exemplo de um trabalho que utiliza esse protocolo de comunicação para automação residencial, utilizando esse protocolo em uma das camadas da solução, em que os dispositivos trocam informações entre si, sem a necessidade de um dispositivo centralizador.

O ESP-NOW(SYSTEMS, 2025) não depende de uma infraestrutura de rede, permitindo a comunicação direta entre os dispositivos. No trabalho citado e em outros que esse protocolo foi testado, destacou-se a baixa latência na comunicação, o baixo consumo de energia e a segurança com criptografia; por esses e outros motivos, esse protocolo de comunicação foi escolhido.

2.4 Interface de Programação de Aplicações

Uma API é um conjunto de regras e definições que permite a comunicação entre diferentes sistemas de software ou hardware. Ela serve como uma ponte, facilitando a troca de informações e a integração entre aplicativos, dispositivos ou serviços.

APIs são amplamente utilizadas para abstrair a complexidade dos sistemas, expondo apenas as funcionalidades necessárias para o uso externo. Elas seguem padrões como o Representational State Transfer (REST), criado por Roy Fielding no ano 2000, que utiliza o protocolo HyperText Transfer Protocol (HTTP) para operações de envio e recebimento de dados e é projetado para ser independente da linguagem de programação utilizada em sua implementação. No trabalho de Ashwath e Priyanka (GOWDA; GOWDA, 2024), são apresentadas boas práticas para o desenvolvimento de APIs REST, com foco em segurança, escalabilidade e organização dos recursos.

Um exemplo comum de API ocorre quando aplicativos acessam serviços externos, como mapas ou sistemas de pagamento, por meio de endpoints específicos. Elas podem ser privadas, utilizadas internamente por organizações para integrar diferentes componentes de um sistema, ou

públicas, acessíveis a qualquer desenvolvedor. Um exemplo amplamente utilizado no Brasil é o serviço ViaCEP (2025), que disponibiliza consultas de endereços por Código de Endereçamento Postal (CEP) de forma simples, gratuita e baseada no padrão REST.

No presente trabalho foi desenvolvida uma API com o framework NestJS (NestJS Framework, 2025), por conta de seu nível de abstração e facilidade de utilição com sua organização em módulos. O NestJS é um framework para criar aplicações Node.js para o lado do servidor, ele é construído com Typescript e oferece suporte completo a essa linguagem, mas também pemite que desenvolvedores programem em JavaScript.

O trabalho Pham (2020) analisa o uso do framework NestJS em projetos, comparando literatura técnica e entrevistas com desenvolvedores. O estudo conclui que o NestJS oferece boa arquitetura, uso de TypeScript, escalabilidade e ferramentas de teste.

O estudo Souza, Lima e Caridade (2022) apresenta o desenvolvimento de um sistema para agendamento de espaços em uma instituição de ensino, utilizando NestJS (NestJS Framework, 2025) no backend e Next.js (VERCEL, 2025) no frontend, demonstrando como essas tecnologias podem compor uma aplicação web moderna, escalável e de fácil manutenção.

2.5 Página WEB

Uma página web é uma estrutura visual interativa acessada por meio de navegadores de internet. Ela geralmente é composta de três tecnologias principais:

- **HyperText Markup Language (HTML):** Define a estrutura e o conteúdo da página.
- **Cascading Style Sheets (CSS):** Adiciona estilo e formatação visual, como cores, fontes e espaçamento.
- **JavaScript:** Torna a página dinâmica, adicionando interatividade, como botões, animações e atualizações automáticas de conteúdo.

Uma página web pode ser estática, o conteúdo é fixo e não muda sem intervenção manual ou dinâmica, o conteúdo é gerado ou atualizado em tempo real, frequentemente com o auxílio de APIs que fornecem informações em segundo plano.

Além disso, as páginas web podem incorporar tecnologias como Asynchronous JavaScript and XML (AJAX) ou WebSocket, que permitem a troca de dados entre cliente e servidor sem recarregar a página, proporcionando uma experiência mais fluida e interativa.

Para o presente trabalho foi desenvolvido um projeto web utilizando Next.js (VERCEL, 2025) que é um framework para construir aplicações web full-stack.

O estudo de Srivastava et al. (2024) apresenta uma revisão abrangente sobre o Next.js, destacando sua evolução, principais recursos e aplicações no desenvolvimento web moderno. Os autores discutem como o framework melhora o desempenho, facilita a navegação e simplifica a estruturação de projetos por meio de recursos como pré-renderização, roteamento baseado em arquivos e otimizações de carregamento. O artigo oferece uma visão consolidada sobre as vantagens do Next.js e seu impacto na construção de aplicações web mais rápidas e organizadas.

2.6 Banco de Dados

Bancos de dados são sistemas responsáveis pelo armazenamento, organização e gerenciamento de informações, permitindo acesso eficiente, seguro e consistente aos dados. Segundo Mostafa, AbuSalim e Saringat (2020) , os Sistemas de Gerenciamento de Banco de Dados (DBMS) surgiram com o objetivo de superar as limitações dos sistemas baseados em arquivos, como redundância de dados, inconsistência e dificuldades de manutenção. Os autores apresentam uma visão geral dos DBMS, abordando seus propósitos, componentes e principais modelos de dados. O estudo compara modelos relacionais, orientados a objetos, hierárquicos e em rede, destacando que a escolha do modelo adequado depende das características da aplicação e da complexidade dos dados armazenados.

Os bancos de dados podem ser classificados em relacionais e não relacionais. Os relacionais utilizam estruturas tabulares e a linguagem SQL, garantindo integridade e consistência dos dados. Já os bancos não relacionais (NoSQL) oferecem maior flexibilidade estrutural e escalabilidade, sendo amplamente utilizados em aplicações web e sistemas distribuídos.

Em aplicações modernas, como sistemas de monitoramento e soluções baseadas em Internet das Coisas (IoT), o uso de bancos de dados é essencial para registrar eventos, históricos e estados do sistema.

No contexto deste trabalho, o banco de dados é empregado para armazenar informações de ocupação dos ambientes, registros temporais e dados provenientes dos dispositivos. O sistema implementa uma arquitetura híbrida que combina persistência em MySQL (Oracle Corporation, 2025) para dados históricos e estruturais, com cache em Redis (Redis Ltd., 2025) para dados de ocupação em tempo real. Além disso, no hardrware principal, é utilizado um banco de dados

SQLite para armazenamento e persistência dos dados locais, ele é utilizado apenas no hardware principal para que os dados de ocupação e configuração não se percam após a reinicialização do mesmo. Esse banco SQLite não está relacionado à arquitetura do sistema, se trata apenas de boa prática no armazenamento de dados locais no hardware principal.

2.7 Trabalhos Relacionados

Diversos trabalhos abordam o monitoramento de ambientes e a contagem de pessoas por meio de sensores e sistemas embarcados. Com o avanço da visão computacional, o uso de câmeras aliado a algoritmos de detecção de pessoas, como redes neurais convolucionais, tem se destacado por oferecer maior precisão e flexibilidade. Essas abordagens permitem o monitoramento em tempo real e a coleta de dados históricos para análise do uso dos ambientes.

Em Curiel et al. (2024) temos um trabalho que propõe um sistema de contagem automática de pessoas em portais de acesso utilizando visão computacional embarcada. A solução emprega uma câmera RGB em visão superior, detecção de movimento por diferença entre quadros, detecção de pessoas com YOLOv4-tiny, rastreamento por Optical Flow de Lucas-Kanade e contagem baseada na análise das trajetórias dos indivíduos entre zonas pré-definidas.

Os experimentos, realizados em ambientes reais como ônibus e entradas de edifícios, apresentaram precisão média de 96,8%, recall de 92% e F1-score de 94,4%, operando em tempo real a aproximadamente 29 FPS, demonstrando a viabilidade da abordagem para aplicações contínuas de monitoramento.

Em Saffari et al. (2021) temos o desenvolvimento de um sistema de detecção de ocupação baseado em câmeras sem bateria, projetado para operar com baixo consumo energético e hardware de baixo custo. As câmeras capturam imagens em baixa resolução, alimentadas por energia ambiente (luz), e transmitem os dados por meio de comunicação backscatter, reduzindo significativamente o consumo energético. O processamento é realizado em um Raspberry Pi 4, onde é implementado o algoritmo de detecção YOLOv5, adaptado para execução em sistemas embarcados utilizando TensorFlow Lite.

Os experimentos foram conduzidos em ambientes internos reais, como residências e laboratórios, considerando variações de iluminação, postura, distância e oclusão parcial dos indivíduos. Os resultados demonstraram acurácia superior a 90% na detecção de ocupação, com

tempo médio de inferência de aproximadamente 100 ms por imagem, operando com apenas 2 GB de RAM. O estudo comprova a viabilidade do uso de visão computacional e deep learning em sistemas embarcados energeticamente autônomos, destacando robustez e confiabilidade para aplicações contínuas de monitoramento de ocupação.

O trabalho realizado por Zhang et al. (2025) realiza uma comparação entre sistemas de detecção de ocupação baseados em visão computacional, utilizando câmeras RGB convencionais e câmeras térmicas em ambientes internos. A metodologia emprega técnicas de processamento de imagens e detecção de pessoas para identificar presença e ocupação sob diferentes condições ambientais, como variações de iluminação, sombras e oclusões.

Os resultados experimentais indicam que câmeras RGB apresentam melhor desempenho em ambientes bem iluminados, oferecendo maior riqueza de detalhes visuais, enquanto câmeras térmicas demonstram maior robustez em condições de baixa iluminação ou ausência total de luz, mantendo elevada taxa de detecção. O estudo conclui que a escolha da tecnologia de sensoriamento deve considerar o ambiente de aplicação, sendo ambas viáveis para monitoramento de ocupação em edifícios inteligentes, com vantagens e limitações específicas.

2.8 Conclusões Parciais

Este capítulo apresentou as referências teóricas utilizadas como base para a concepção do atual projeto. Foram apresentados trabalhos que possuem aplicações similares às utilizadas no desenvolvimento do sistema de monitoramento de ocupação de ambientes, abordando detecção de pessoas através de modelos treinados de detecção, protocolos de comunicação entre hardwares, API e páginas web.

3 METODOLOGIA

Neste capítulo, são apresentados os métodos e etapas a serem seguidos para a reprodução da solução proposta, dando visibilidade sobre a arquitetutura de comunicação, detalhando os hardwares utilizados, a versão do modelo de detecção escolhido, como o firmware do hardware principal funciona referente à detecção de pessoas, como é o controle de dispositivos eletrônicos, cálculo de médias de ocupação e gasto estimado de energia, além do detalhamento da API, endpoints utilizados e a forma de armazenamento de dados no banco e suas tabelas.

3.1 Levantamento de Requisitos

Antes do desenvolvimento técnico, se torna necessário um levantamento dos requisitos funcionais e não funcionais, com o objetivo de garantir que a solução atenda às necessidades esperadas.

Requisitos Funcionais

- Detectar presença de pessoas usando hardware embarcado.
- Enviar dados via API usando protocolo HTTP.
- Exibir informações em uma interface Web.

Requisitos Não Funcionais

- Sistema com resposta próxima ao tempo real.
- Solução de hardware de baixo custo.
- API rápida e estável.

3.2 Arquitetura da Solução

A arquitetura da solução foi projetada com foco na separação de responsabilidades e na utilização de tecnologias adequadas para cada componente. A proposta é composta por quatro módulos principais:

1. **Hardware Principal:** É necessário utilizar um hardware compacto que rode um sistema operacional leve, neste caso um microcomputador de placa única, conectado a uma

câmera e a um microcontrolador. O microcomputador realizará a detecção de pessoas por meio de visão computacional com uso de inteligência artificial. O microcontrolador será responsável por controlar dispositivos eletrônicos da sala, recebendo comandos por interface serial e transmitindo via protocolo de comunicação sem fio.

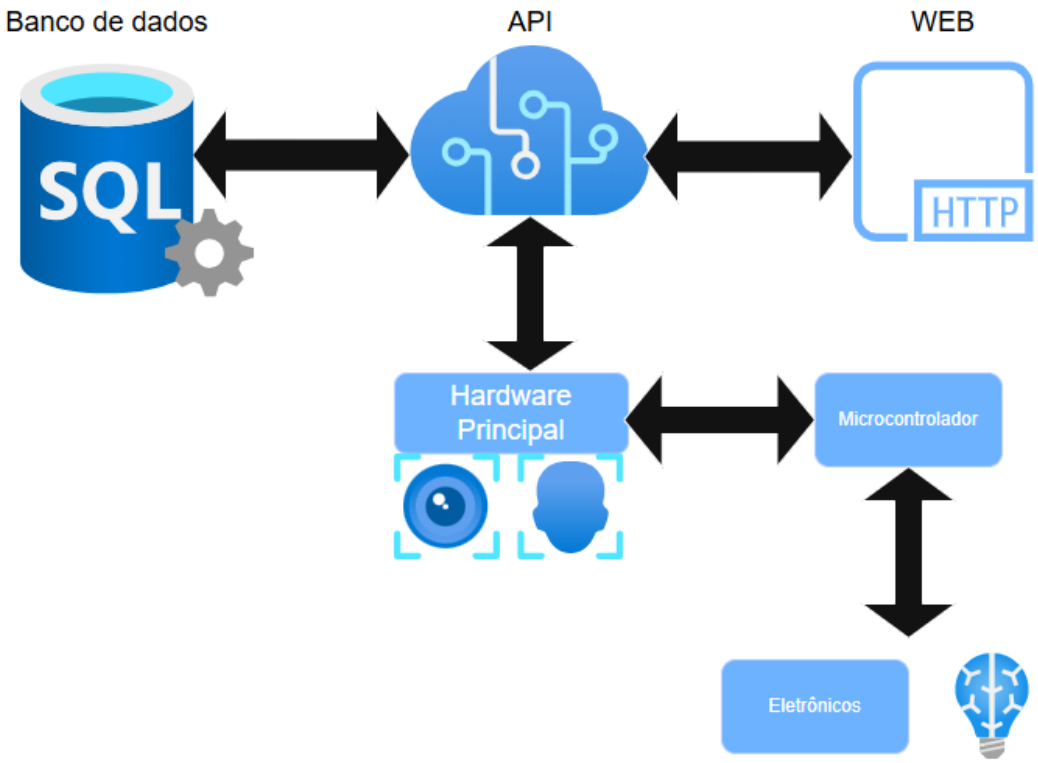
2. **API:** Uma API é utilizada para coordenar a comunicação, ela receberá os dados enviados pelo hardware, armazenando-os em um banco de dados e disponibilizando endpoints para acesso via frontend.

3. **Banco de dados:** Para armazenamento de dados, um sistema de gerenciamento de banco de dados relacional deve armazenar todas as informações necessárias para o sistema, sejam elas referentes às configuraçãoes da organização mapeada, como unidades, prédios, e salas da organização e também relacionadas aos dados reportados pelo hardware, como médias de ocupação e estatísticas.

4. **Interface web:** Uma interface gráfica acessível via navegador deve ser implementada, permitindo a visualização do status de ocupação das salas, gráficos com dados históricos e quais dispositivos eletroeletrônicos a sala em questão possui.

A Figura 3.1 mostra o fluxo de comunicação do sistema. A API é o centro da comunicação entre o hardware principal, o banco de dados e a página web. Pode-se observar também o fluxo de comunicação para controle de dispositivos, que ocorre entre o hardware principal, o microcontrolador e os eletrônicos presentes na sala.

Figura 3.1 – Diagrama do fluxo de comunicação.

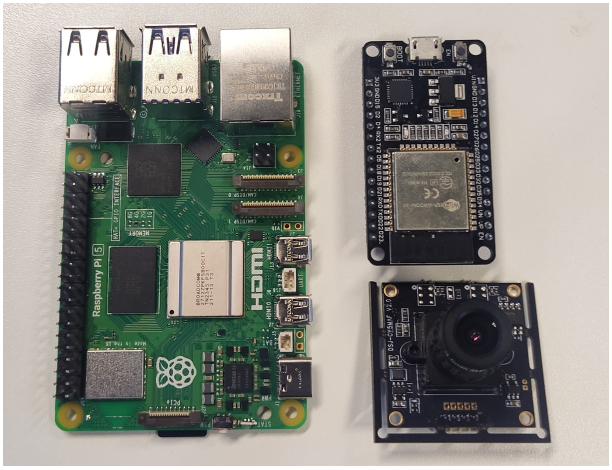


Fonte: Próprio autor

3.3 Hardware principal de controle

Na Figura 3.2 temos a Raspberry Pi 5, a câmera USB e o ESP-32, formando o hardware principal.

Figura 3.2 – Hardware principal



Fonte: Próprio autor

O Raspberry Pi 5 é o elemento central do hardware, desempenhando diversas funções essenciais. Esse computador de placa única é um hardware versátil, utilizado em projetos industriais e educacionais, com aplicações em automação residencial, projetos de robótica, servidores, entre outros. Possui um processador Broadcom BCM2712, quadcore Arm Cortex-A76 64bit, clock de 2.4 GHz, memória RAM de 4 GB, além de diversos periféricos e múltiplas interfaces, incluindo portas USB, GPIOs *(General Purpose Input/Output)*, interface para câmeras CSI *(Camera Serial Interface)* e conectividade integrada por Ethernet, Wi-Fi e Bluetooth. A escolha do Raspberry Pi 5 se deve à sua combinação de desempenho, flexibilidade e custo-benefício. Ele realiza a detecção de ocupação utilizando um modelo YOLOv8 (ULTRALYTICS, 2025) de detecção de objetos e pessoas, implementado em código Python. Além disso, é responsável por enviar as informações processadas para a API e transmitir mensagens para os hardwares secundários de controle através da comunicação serial com um microcontrolador ESP-32.

O Raspberry se comunica diretamente com um ESP-32 conectado a uma porta USB, que será chamado de ESP-HOST. Esse ESP é responsável por receber informações e transmiti-las via protocolo ESP-NOW (SYSTEMS, 2025) para os demais ESPs que controlam os dispositivos eletroeletrônicos.

O sistema operacional utilizado no Raspberry foi o Armbian (DEVELOPERS, 2025), uma distribuição baseada em Linux. Esse sistema operacional foi escolhido por ser "leve"e permitir configurações de fácil acesso ao hardware, como acesso a portas USBs e GPIOs da

Raspberry.

Algumas câmeras com especificações diferentes foram testadas, e a selecionada para a solução foi o módulo de câmera Neocoolcam 5 MP USB, equipado com sensor IMX335 (4K), compatível com o padrão USB Video Class (UVC) e operação plug and play. Suas principais características são:

- Resolução do sensor: 5 MP
- Sensores disponíveis: Sony IMX335 (4K) ou IMX179 (2K)
- Taxa de atualização: até 30 fps
- Interface: USB padrão UVC, compatível com Linux
- Alimentação: 5V via USB
- Formato da imagem: MJPEG / YUY2 (dependendo da configuração)
- Tipo: Módulo PCBA compacto, adequado para integração em projetos embarcados

Este modelo foi escolhido por oferecer boa qualidade de imagem para aplicações de detecção de pessoas, além de apresentar compatibilidade total com dispositivos Linux via UVC, permitindo operação imediata *plug and play* sem drivers adicionais. Assim como outras câmeras USB, qualquer modelo compatível poderia ter sido utilizado, mas esta opção apresentou melhor equilíbrio entre resolução, disponibilidade e custo.

3.3.1 Versão do modelo de detecção utilizado

A eficácia e o alcance máximo de detecção dependem da resolução das imagens fornecidas pela câmera e do modelo de detecção utilizado. O YOLOv8 possui variações otimizadas tanto para dispositivos com recursos limitados quanto para hardwares de alto desempenho (ULTRALYTICS, 2025). Essas versões são diferenciadas pela última letra em sua nomenclatura, sendo elas: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large) e YOLOv8x (extra large). Modelos menores, como o YOLOv8n, apresentam menor consumo de memória e maior velocidade de processamento, sendo adequados para dispositivos embarcados, como o Raspberry Pi 4. Versões mais robustas, como o YOLOv8l ou YOLOv8x, demandam maior poder computacional, aproveitando melhor recursos como no caso do Raspberry Pi 5, garantindo maior precisão nas detecções.

Visando o melhor desempenho com o hardware escolhido, foi feita uma otimização no modelo utilizando o OpenVINO da Intel (CORPORATION, 2025). Essa ferramenta ajusta o modelo para que ele aproveite melhor o hardware disponível, reorganizando cálculos e reduzindo a quantidade de operações necessárias. O OpenVino pode reduzir a precisão dos cálculos sem prejudicar significativamente a qualidade da detecção.

Para escolher a melhor versão, com menor tempo de execução e maior assertividade nas detecções, foram feitos alguns testes no escritório da empresa em que o autor trabalha.

A Tabela 3.1 mostra os resultados obtidos com um script Python apenas de detecção de pessoas, executado em um Raspberry PI 5. Nela pode-se observar como a otimização via OpenVino reduz significativamente o tempo de inferência do modelo.

Tabela 3.1 – Tempo de inferência dos modelos YOLOv8 com e sem otimização via OpenVINO

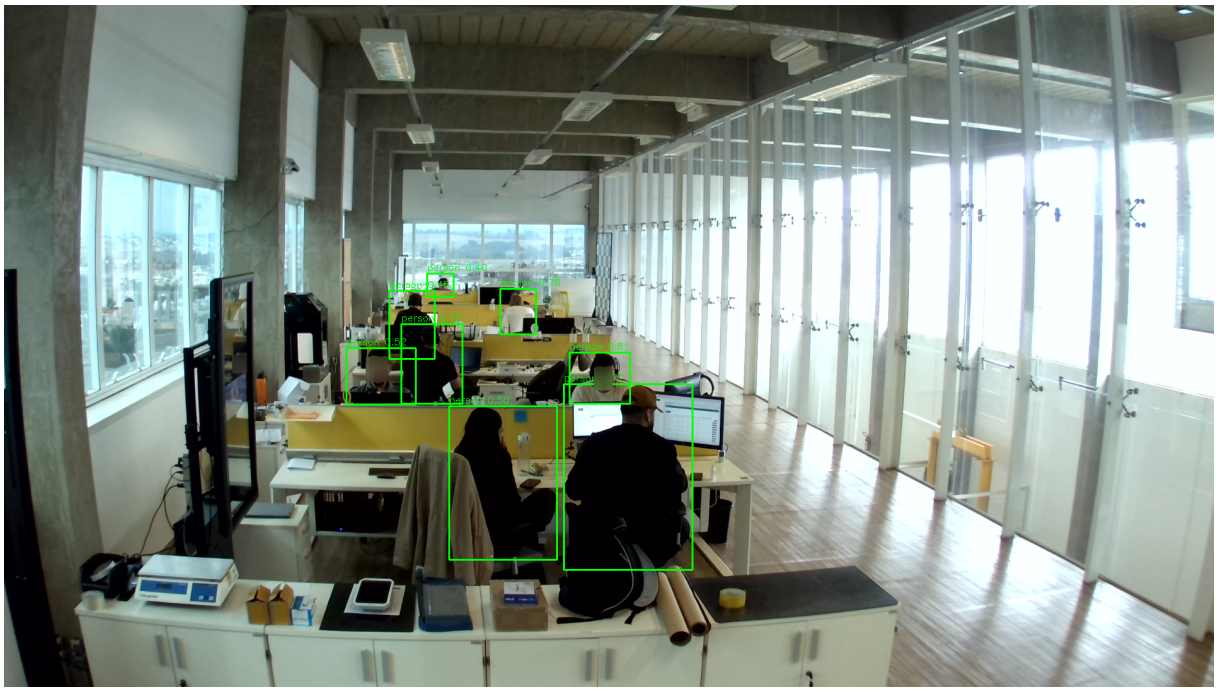
Resolução	**YOLOv8n**	**YOLOv8s**	**YOLOv8m**	**YOLOv8l**	**YOLOv8x**
Sem otimização					
640x360	500 ms	1400 ms	3400 ms	6500 ms	9900 ms
1280x720 (HD)	430 ms	1100 ms	2700 ms	5400 ms	8000 ms
1920x1080 (FullHD)	430 ms	1100 ms	2700 ms	5400 ms	8000 ms
Com otimização (OpenVINO)					
1920x1080 (FullHD)	110 ms	240 ms	530 ms	1100 ms	1700 ms

Com os testes foi possível comprovar que o modelo com otimização obtém respostas mais rápidas, além disso, as detecções foram observadas diretamente nas imagens, concluindo também que o modelo YOLOv8x é superior às outras versões quanto à assertividade nas detecções. Isso pode ser observado na comparação entre a Figura 3.3, que foi capturada utilizando a detecção com YOLOv8n, em que pessoas mais distantes não foram identificadas na imagem, e a Figura 3.4, que foi obtida com o modelo de detecção YOLOv8x, em que todas as pessoas da imagem foram detectadas.

Figura 3.3 – Detecção com YOLOv8n.

Fonte: Próprio autor

Figura 3.4 – Detecção com YOLOv8x.



Fonte: Próprio autor

Após os testes, a versão do modelo escolhida para ser utilizada foi o YOLOv8x com otimização OpenVINO.

3.3.2 Script de controle

A linguagem escolhida para a implementação no hardware foi Python, por ser amplamente utilizada em aplicações envolvendo modelos treinados. Ela oferece uma grande variedade de bibliotecas voltadas para visão computacional e aprendizado de máquina, além de contar com uma comunidade ativa e documentação extensa, o que facilita o desenvolvimento e a resolução de problemas.

Primeiramente, foi criado um ambiente virtual Python, para que o gerenciamento de versões do Python e das dependências não interferisse no Python padrão do sistema. Em seguida, foram adicionadas as dependências para utilizar o YOLOv8 (biblioteca ultralytics) e manipular imagens da câmera.

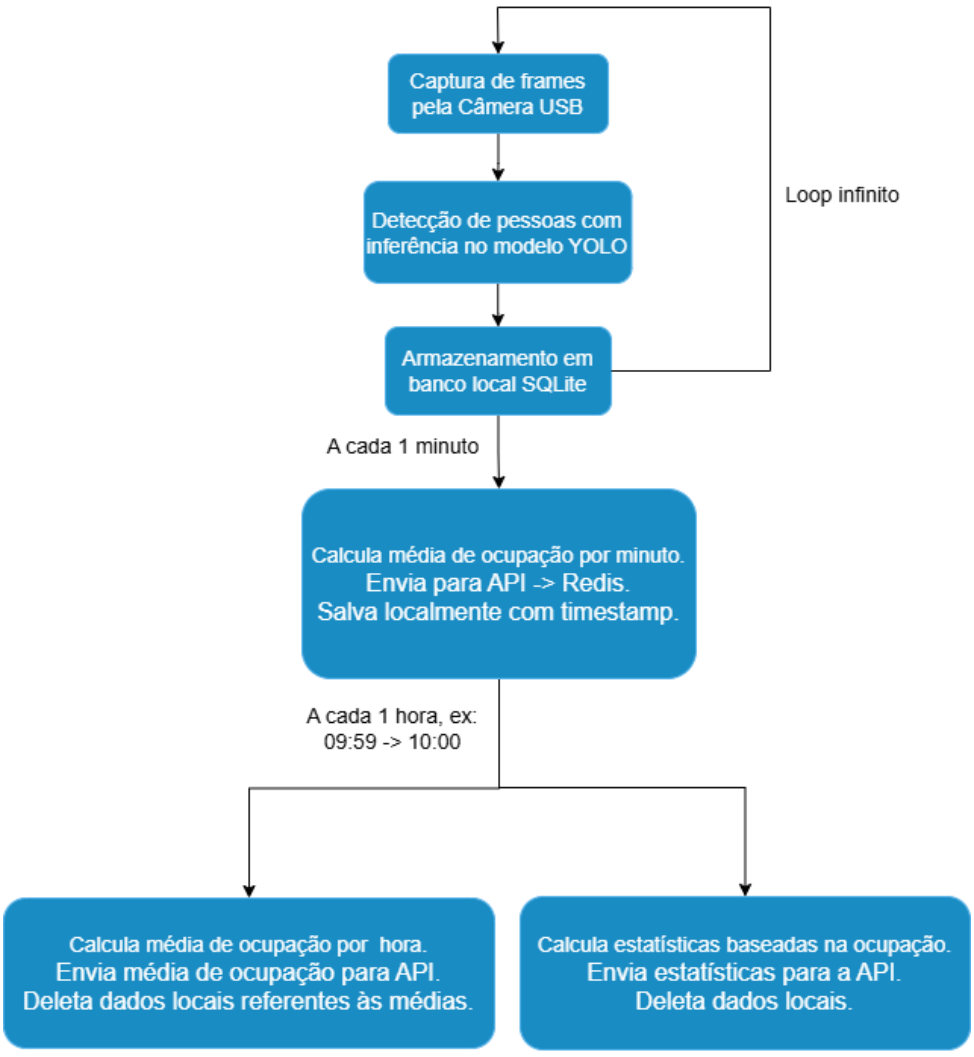
O código principal captura imagens da câmera USB e faz inferências no modelo YOLOv8 (ULTRALYTICS, 2025) de detecção de pessoas, que retorna o número de pessoas detectadas. Esse fluxo acontece em uma tarefa executada continuamente, e o tempo de captura do frame, inferência no modelo e armazenamento das respostas ocorre em torno de 2,5 a 3,5 segundos. O resultado da inferência, contendo o número de pessoas detectadas, é armazenado em um vetor. A cada minuto decorrido, a média de ocupação é calculada somando-se os valores armazenados nesse vetor e dividindo-se pelo seu tamanho (quantidade de valores armazenados). Dessa forma, obtém-se a média de ocupação por minuto no ambiente, e esse valor é enviado para a API, que o armazena em um banco Redis (Redis Ltd., 2025), do qual a página Web consome esses dados como ocupação em "tempo real". Além disso, esse valor é armazenado no banco local SQLite (CONSORTIUM, 2025), utilizado apenas no hardware principal para armazenamento de dados no dispositivo, para manter os dados persistentes após a reinicialização do hardware principal.

Quando ocorre a mudança de uma hora, por exemplo, quando o relógio do sistema passa de 9:59 para 10:00, o sistema calcula a média por hora, somando todos os valores de médias por minuto armazenados no banco local, referentes à hora que mudou e dividindo pelo número de valores de médias por minuto, obtendo assim a média por hora. Nesse momento, a média por hora é enviada para a API, e esses dados serão utilizados nos gráficos exibidos na página Web, como no exemplo da Figura 4.5. Os dados locais do SQLite (CONSORTIUM, 2025) são deletados após o envio de média por hora, para que o ciclo recomece.

Além do envio dos dados de ocupação, dados estatísticos são calculados a partir da presença de pessoas, armazenados localmente e enviados em intervalos de uma hora. Eles contêm a média diária de ocupação, a média máxima de ocupação em um determinado horário do dia, o horário em que essa média máxima ocorreu e o consumo estimado de energia desse dia, que será detalhado posteriormente.

A Figura 3.5 demostra o fluxo de funcionamento do script de controle.

Figura 3.5 – Diagrama de fluxo do script de controle



Fonte: Próprio autor

3.3.2.1 Cálculo estimado do gasto de energia diário

Além do fluxo de dados de ocupação, o firmware do hardware principal controla os dispositivos eletrônicos do ambiente, como detalhado em 3.3.3, dessa forma se torna possível

calcular o consumo diário de energia baseado no tempo real de ocupação do ambiente. A premissa é que todos os dispositivos elétricos permanecem ligados apenas quando há presença de pessoas e desligados quando não há. O cálculo é realizado em duas etapas:

1. Consumo por hora ocupada: O consumo total quando o ambiente está em uso é dado pela soma do consumo de todos os dispositivos:

$$C_{hora} = \sum_{n=1}^{N} q_n \times c_n \tag{3.1}$$

onde q_n é a quantidade de dispositivos do tipo n e c_n é o consumo deste dispositivo.

2. Consumo diário: O consumo total do dia é calculado multiplicando o consumo por hora pelo tempo efetivo de ocupação:

$$E_{dia} = \frac{m_{ocupados}}{60} \times C_{hora} \tag{3.2}$$

onde $m_{ocupados}$ é o número de minutos em que houve pelo menos uma pessoa presente e C_{hora} é o consumo por hora calculado na equação anterior.

Exemplo Prático

Considerando um ambiente com 4 lâmpadas LED (0,01 kWh/h cada), 1 ar-condicionado (1,5 kWh/h) e 2 computadores (0,15 kWh/h cada):

$$\begin{aligned} C_{hora} &= (4 \times 0{,}01) + (1 \times 1{,}5) + (2 \times 0{,}15) \\ &= 0{,}04 + 1{,}5 + 0{,}3 = 1{,}84\,\text{kWh/h} \end{aligned} \tag{3.3}$$

Se o ambiente ficou ocupado por 540 minutos (9 horas) no dia:

$$E_{dia} = \frac{540}{60} \times 1{,}84 = 9 \times 1{,}84 = 16{,}56\,\text{kWh} \tag{3.4}$$

Dessa forma os dados estimados de gasto de energia são calculados e armazenados localmente em uma tabela com as estatísticas diárias. Esses dados são enviados para a API de hora em hora, para manter os dados atualizados no banco.

3.3.3 Controle de dispositivos elétricos

Se o número de pessoas detectadas for maior que zero, uma mensagem é enviada para o ESP-HOST conectado a Raspberry Pi 5 através da porta USB, para que o mesmo envie mensagens aos demais ESPs secundários para ligar os dispositivos eletroeletrônicos da sala. Se o número de pessoas detectadas for igual a zero, é aberta uma contagem de tempo, para que, se permanecer sem presença na sala até o fim dessa contagem, os dispositivos eletroeletrônicos serão desligados.

Para a execução do firmware continuamente, sem interrupções, foi criado um serviço no sistema operacional que inicia a aplicação junto com a inicialização do sistema; assim, a aplicação funciona sempre que o dispositivo estiver ligado, sem parar após reinicialização do hardware.

Os dispositivos como luz e eletrodomésticos presentes em uma sala de aula, geralmente são alimentados por tensão e corrente alternadas de 110 ou 220 volts, sendo que os eletrodomésticos possuem um circuito retificador interno para conversão de energia alternada em contínua. O controle desses dispositivos através da presente solução, foi projetado para cortar a alimentação dos eletroeletrônicos, para isso se torna necessário a utilização de um dispositivo que receba mensagens através do protocolo ESP-NOW (SYSTEMS, 2025a) e interrompa essa alimentação. Seria necessário criar uma *Printed Circuit Board (PCB)* com uma fonte retificadora, um microcontrolador e um relé conectado aos fios de alimentação, porém esses dispositivos já existem comercialmente. Eles são interruptores inteligentes amplamente utilizados em projetos de automação residencial. Para documentação e orientação para trabalhos futuros são mostrados dois exemplos desses hardwares, o Sonoff Mini (ITEAD Intelligent Systems Co., Ltd., 2025b) e o Sonoff Basic (ITEAD Intelligent Systems Co., Ltd., 2025a), porém no desenvolvimento do presente trabalho, apenas o Sonoff-Basic foi utilizados nos testes.

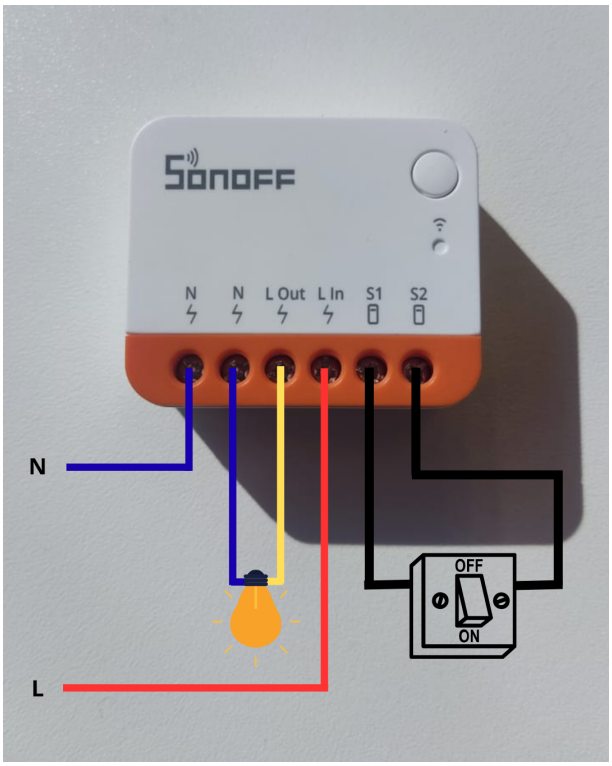
Principais componentes desses hardwares:

- ESP8266: responsável pelo controle lógico e pela comunicação via ESP-NOW.
- Relé eletromecânico: permite o acionamento de cargas até 10A.
- Fonte chaveada: converte a tensão da rede elétrica (AC 90–250V) para 3,3V CC usada pelo ESP8266.
- Terminais de entrada/saída: para conexão de fase, neutro e carga.

- Botão físico: usado para pareamento ou acionamento manual.
- Entradas para interruptor externo (no caso do Sonoff Mini).

A Figura 3.6 mostra um exemplo de Sonoff Mini com diagrama de instalação. Ele é projetado especificamente para ser embutido em caixas de interruptores, permitindo controle tanto remoto quanto local (mantendo o interruptor da parede funcional).

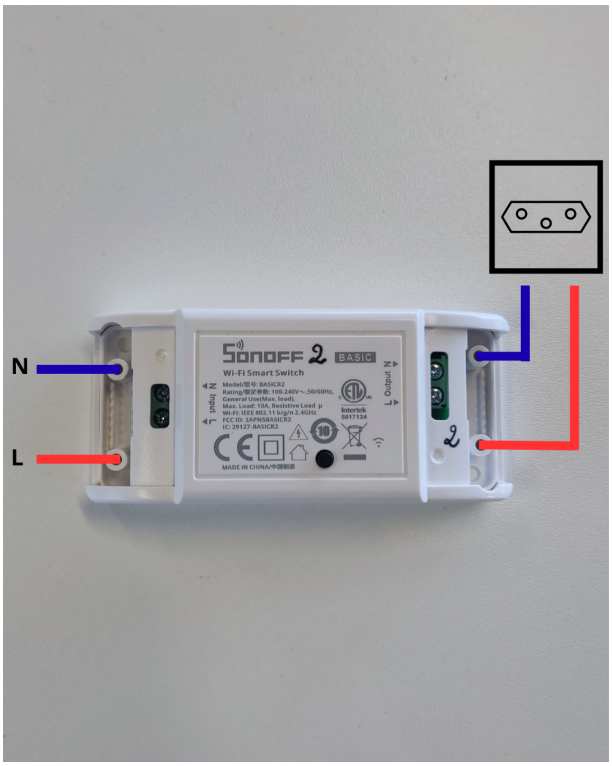
Figura 3.6 – Sonoff mini



Fonte: Próprio autor

A Figura 3.7 mostra um exemplo de Sonoff Basic ideal para ser embutido em caixas de passagem ou adaptado em extensões, funcionando como um controlador de linha.

Figura 3.7 – Sonoff basic

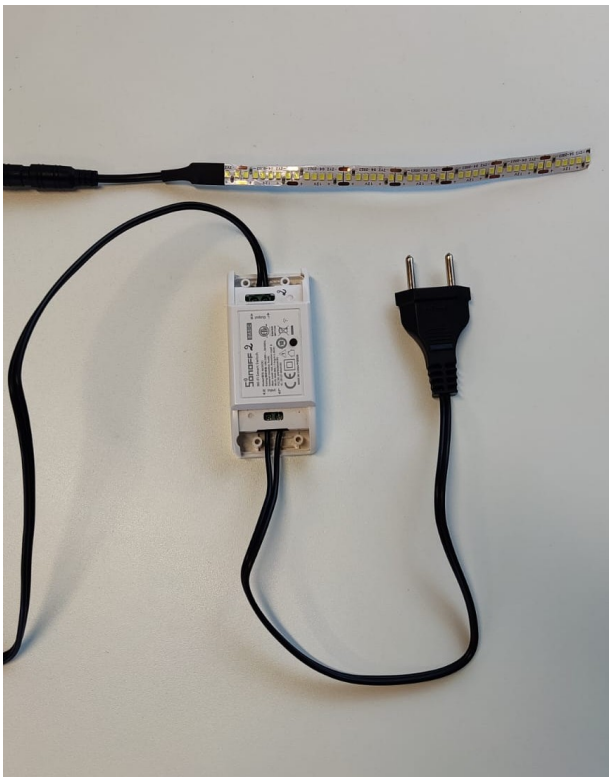


Fonte: Próprio autor

Ambos saem de fábrica com um firmware padrão gravado em seus microcontroladores, o que facilita sua utilização com o aplicativo padrão do fabricante, ou podem ser regravados para utilizar firmware alternativo. No presente projeto, o firmaware utilizado foi criado pelo próprio autor, para utilizar o protocolo ESP-NOW (SYSTEMS, 2025) para comunicação desses dispositivos com o ESP-HOST, criando a comunicação própria de forma alternativa.

Os dispositivos de controle não foram instalados diretamente nos eletrônicos do ambiente, eles foram configurados para acionar uma fita *Light Emitting Diode (LED)* que representa o funcionamento desses dispositivos. Se a fita permanece acesa, demonstra que os equipamentos estariam ligados; se permanece apagada, indica que estariam desligados, conforme o comando do hardware principal. A Figura 3.8 apresenta o dispositivo de controle e a fita LED utilizada, omitindo a fonte de alimentação.

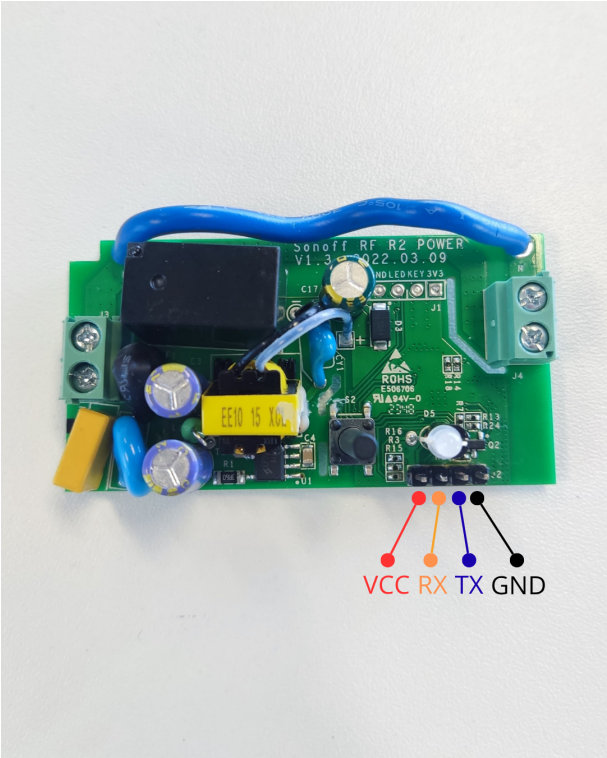
Figura 3.8 – Sonoff basic com fita LED



Fonte: Próprio autor

Para regravar o firmware no Sonoff Basic, foi necessário abrir o hardware comercial e conectar um conversor serial USB aos GPIOs contidos na PCB responsáveis pela gravação (3.3V, GND, RX e TX). A localização dos GPIOs referentes a gravação do firmware pode ser vista na Figura 3.9.

Figura 3.9 – Sonoff basic PCB



Fonte: Próprio autor

O firmware gravado nesse hardware se conecta ao ESP-HOST pelo protocolo ESP-NOW, ele recebe comandos para ligar e desligar o relé presente na PCB, controlando assim a fita LED que representa os dispositivos do ambiente.

3.4 Descrição dos Endpoints e Fluxos da API

Neste projeto foi utilizada uma API REST que integra hardware de detecção, banco de dados relacional MySQL (Oracle Corporation, 2025), Redis (Redis Ltd., 2025) para dados em tempo real, e aplicação web, através de uma arquitetura hierárquica escalável. O sistema utiliza Redis (Redis Ltd., 2025) para cache em tempo real e suporta análises históricas e estatísticas.

Os endpoints foram organizados em dois grupos principais: **Endpoints de Informações de Salas** e **Endpoints de Organização Hierárquica**.

3.4.1 Endpoints de Informações de Salas

O primeiro grupo de endpoints é utilizado para alterar ou obter informações de todas as salas cadastradas no sistema, são disponibilizados oito endpoints REST divididos em três

categorias principais:

- Operações de tempo real com Redis (Redis Ltd., 2025).
- Consulta de dados cadastrais.
- Persistência de dados históricos e estatísticas com MySQL (Oracle Corporation, 2025).

3.4.1.1 Endpoints de Operações Tempo Real

O endpoint `POST /salas/:id` é responsável por atualizar as informações de uma sala em tempo real, recebendo principalmente os dados de ocupação atual. As informações são armazenadas no Redis (Redis Ltd., 2025), garantindo alta velocidade para operações em tempo real. Este endpoint é utilizado pelo hardware embarcado para enviar atualizações periódicas.

O endpoint `GET /salas/:id/ocupacao` permite obter a ocupação atual de uma sala específica, consultando diretamente o Redis (Redis Ltd., 2025). Para garantir dados sempre atualizados, este endpoint desabilita explicitamente o cache HTTP.

3.4.1.2 Endpoints de Consulta de Dados Cadastrais

O endpoint `GET /salas` retorna a lista completa de todas as salas cadastradas no sistema, consultando o banco de dados MySQL (Oracle Corporation, 2025). Este endpoint é utilizado principalmente para exibir a listagem de salas disponíveis.

O endpoint `GET /salas/:id` recupera os dados detalhados de uma sala específica, incluindo informações sobre capacidade, equipamentos disponíveis e localização.

3.4.1.3 Endpoints de Dados Históricos e Estatísticas

O endpoint `POST /historico/:id` salva o histórico de ocupação de uma sala no MySQL (Oracle Corporation, 2025), recebendo um array com as médias de ocupação calculadas pelo hardware.

O endpoint `GET /historico/:id` permite buscar o histórico de ocupação de uma sala nos últimos sete dias, retornando os registros mais recentes ordenados por data e hora.

O endpoint `POST /estatisticas/:id` salva as estatísticas diárias calculadas, incluindo valores de máxima e mínima ocupação e estimativa de gasto energético.

O endpoint `GET /estatisticas/:id` recupera as estatísticas diárias de uma sala dos últimos sete dias, fornecendo dados agregados úteis para análises de padrões de uso.

A Tabela 3.2 resune os endpoints da API de salas.

Tabela 3.2 – Resumo dos Endpoints da API de Salas

Método	**Rota**	**Descrição**	**Armazena**	**Usado por**
POST	/salas/:id	Atualizar ocupação em tempo real	Redis	Hardware
GET	/salas/:id/ocupacao	Obter ocupação atual	Redis	Frontend
GET	/salas	Listar todas as salas	MySQL	Frontend
GET	/salas/:id	Obter dados de uma sala	MySQL	Frontend
POST	/historico/:id	Salvar histórico de ocupação	MySQL	Hardware
GET	/historico/:id	Buscar histórico (7 dias)	MySQL	Frontend
POST	/estatisticas/:id	Salvar estatísticas diárias	MySQL	Hardware
GET	/estatisticas/:id	Buscar estatísticas (7 dias)	MySQL	Frontend

3.4.2 Endpoints de Organização Hierárquica

O segundo grupo de endpoints da API gerencia toda a estrutura hierárquica do sistema, disponibilizando catorze endpoints de consulta REST organizados em seis categorias principais: unidades, prédios, andares, salas, consultas hierárquicas e estatísticas. Todos os dados são consultados no banco de dados MySQL (Oracle Corporation, 2025).

O sistema implementa uma estrutura hierárquica de quatro níveis e cada sala possui um ID único no banco, por isso a hierarquia não precisa ser passada na URL de acersso aos dados da sala.

3.4.2.1 Endpoints de Consulta de Unidades

O endpoint `GET /organizacao/unidades` retorna a lista completa de todas as unidades cadastradas no sistema, ordenadas por nome. Este endpoint fornece informações básicas sobre cada campus ou unidade organizacional.

O endpoint `GET /organizacao/unidades/:id` recupera os dados detalhados de uma unidade específica, incluindo informações de contato, endereço e status de ativação.

3.4.2.2 Endpoints de Consulta de Prédios

O endpoint `GET /organizacao/predios` lista todos os prédios cadastrados, com suporte a filtro opcional por unidade através do parâmetro de query `unidade_id`. Os dados

retornados incluem informações da unidade à qual o prédio pertence.

O endpoint `GET /organizacao/predios/:id` obtém os dados detalhados de um prédio específico, incluindo informações sobre a unidade associada e características do prédio.

3.4.2.3 Endpoints de Consulta de Andares

O endpoint `GET /organizacao/andares` lista todos os andares cadastrados, com filtro opcional por prédio através do parâmetro `predio_id`. Retorna informações completas incluindo dados do prédio e unidade.

O endpoint `GET /organizacao/andares/:id` recupera os dados detalhados de um andar específico, incluindo sua posição na hierarquia organizacional.

3.4.2.4 Endpoints de Consulta de Salas

O endpoint `GET /organizacao/salas` lista todas as salas cadastradas com dados de hierárquia completa, incluindo dados da unidade, prédio e andar. Suporta filtro opcional por andar através do parâmetro `andar_id`.

O endpoint `GET /organizacao/salas/:id` obtém os dados completos de uma sala específica, incluindo toda a hierarquia organizacional.

3.4.2.5 Endpoints de Consultas Hierárquicas

O endpoint `GET /organizacao/hierarquia` retorna a hierarquia organizacional completa através de uma stored procedure, fornecendo uma visão estruturada de todas as unidades, prédios, andares e salas.

O endpoint `GET /organizacao/resumo` fornece um resumo estatístico da hierarquia, incluindo contagem total de unidades, prédios, andares, salas ativas e capacidade total do sistema.

O endpoint `GET /organizacao/unidades/:codigo/salas` busca todas as salas pertencentes a uma unidade específica.

O endpoint `GET /organizacao/predios/:codigo/salas` retorna todas as salas de um prédio específico, incluindo informações hierárquicas completas.

3.4.2.6 Endpoints de Estatísticas Hierárquicas

O endpoint `GET /organizacao/estatisticas/unidades` retorna estatísticas agregadas por unidade para um período específico, recebendo datas de início e fim como parâmetros de query.

O endpoint `GET /organizacao/estatisticas/predios` fornece estatísticas detalhadas por prédio dentro de uma unidade específica, permitindo análise comparativa entre diferentes prédios.

3.4.2.7 Operações CRUD

Além dos endpoints de consulta apresentados, a API disponibiliza endpoints completos para operações CRUD (Create, Read, Update, Delete), em todos os níveis da hierarquia organizacional. Estes endpoints permitem o gerenciamento administrativo de unidades, prédios, andares e salas através de métodos HTTP POST, PUT e DELETE. Essas operações não foram utilizadas no projeto, por isso a omissão das mesmas.

A Tabela 3.3 resume os endpoints relacionados à organização hierárquica.

Tabela 3.3 – Resumo dos Endpoints de Consulta da API de Organização hierárquica

Método	Rota	Descrição	Usado por
Unidades			
GET	/organizacao/unidades	Listar todas as unidades	Frontend
GET	/organizacao/unidades/:id	Obter uma unidade específica	Frontend
Prédios			
GET	/organizacao/predios	Listar todos os prédios	Frontend
GET	/organizacao/predios/:id	Obter um prédio específico	Frontend
Andares			
GET	/organizacao/andares	Listar todos os andares	Frontend
GET	/organizacao/andares/:id	Obter um andar específico	Frontend
Salas			
GET	/organizacao/salas	Listar salas com hierarquia	Frontend
GET	/organizacao/salas/:id	Obter sala com hierarquia	Frontend
Consultas Hierárquicas			
GET	/organizacao/hierarquia	Obter hierarquia completa	Frontend
GET	/organizacao/resumo	Obter resumo hierárquico	Frontend
GET	/organizacao/unidades/:codigo/salas	Buscar salas por unidade	Frontend
GET	/organizacao/predios/:codigo/salas	Buscar salas por prédio	Frontend
Estatísticas			
GET	/organizacao/estatisticas/unidades	Estatísticas por unidade	Frontend
GET	/organizacao/estatisticas/predios	Estatísticas por prédio	Frontend

Todos os endpoints implementam validação de dados e tratamento de exceções apropriado, retornando códigos HTTP adequados para cada situação.

3.5 Banco de Dados

O presente trabalho utiliza como Sistema Gerenciador de Banco de Dados (SGBD) o MySQL (Oracle Corporation, 2025), uma solução relacional de código aberto amplamente adotada por sua robustez, performance e suporte a operações complexas. A instância do banco está configurada para operar tanto localmente quanto em servidores dedicados, com conexões gerenciadas através de variáveis de ambiente armazenadas em arquivo `.env`, garantindo flexibilidade e segurança.

O sistema implementa uma arquitetura híbrida que combina persistência em MySQL (Oracle Corporation, 2025), para dados históricos e estruturais, com cache em Redis (Redis Ltd., 2025) para dados de ocupação em tempo real.

3.5.1 Arquitetura do Sistema de Dados

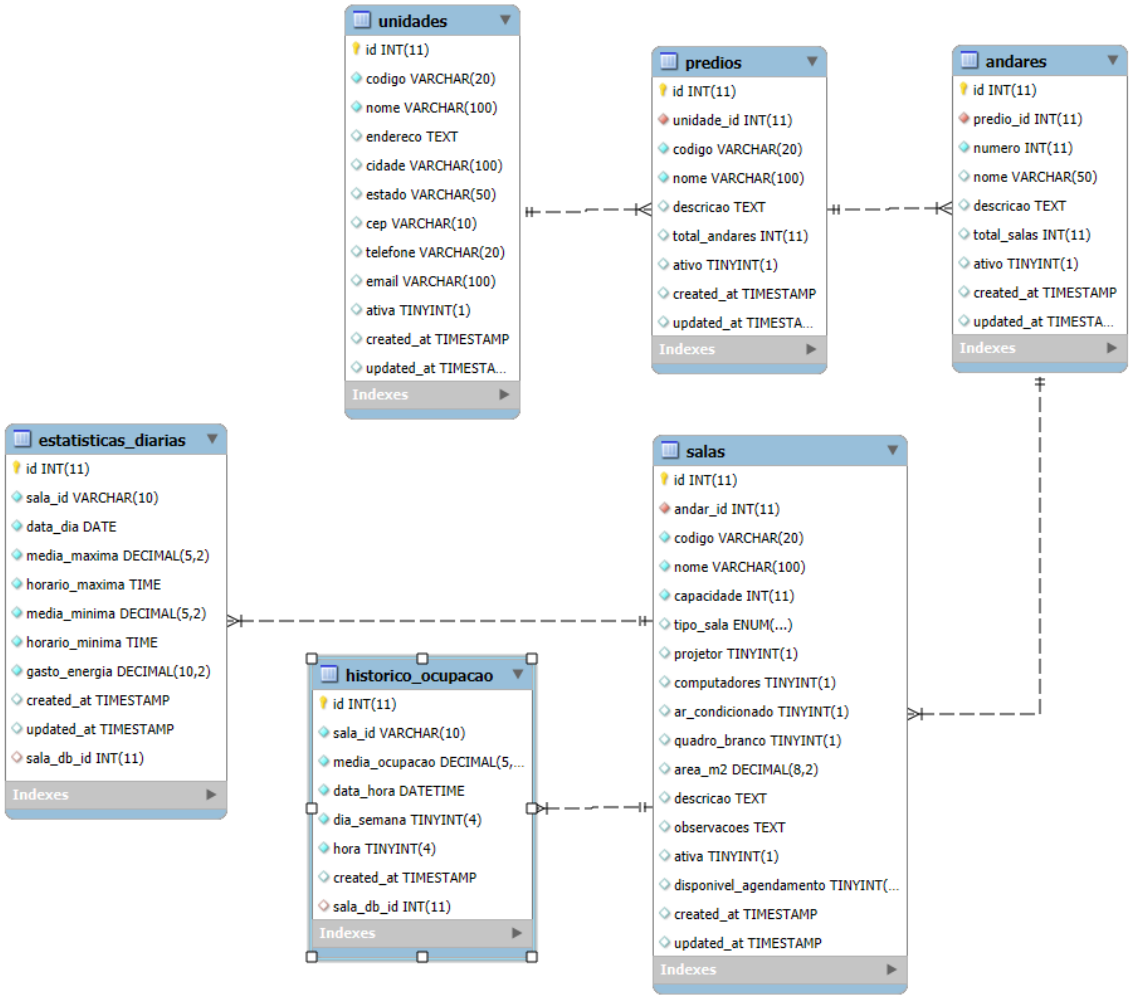
A modelagem do banco de dados considera uma estrutura hierárquica organizacional de quatro níveis, complementada por sistemas de cache e análise temporal:

- **Persistência MySQL:** Dados estruturais, histórico e estatísticas
- **Cache Redis:** Ocupação em tempo real com chaves no formato `sala:id:ocupacao`
- **Limpeza Automática:** Events e procedures para manutenção de dados temporais
- **Views Otimizadas:** Consultas hierárquicas pré-calculadas para performance

3.5.2 Entidades Principais

O sistema implementa seis entidades principais organizadas hierarquicamente, conforme descrito nas Tabelas. Os dados representados são fictícios e o Modelo de Entidade e Relacionamento (MER), pode ser visto na Figura 3.10.

Figura 3.10 – Modelo de Entidade e Relacionamento



Fonte: Próprio autor

A tabela de unidades representa o nível mais alto da hierarquia organizacional, ela armazena informações sobre campus ou unidades administrativas. A Tabela 3.4 apresenta a unidade cadastrada no sistema, incluindo código identificador, nome, localização geográfica e status de ativação.

Tabela 3.4 – Dados Cadastrados - Tabela Unidades

ID	Código	Nome	Cidade	Estado	Ativa
1	CAMPUS_PRINCIPAL	Campus Principal	São Paulo	SP	Sim

A Tabela de prédios armazena as edificações pertencentes a cada unidade. A Tabela 3.5 exibe o prédio cadastrado, mostrando seu vínculo com a unidade através da chave estrangeira,

código identificador, nome descritivo e quantidade de andares.

Tabela 3.5 – Dados Cadastrados - Tabela Prédios

ID	Unidade ID	Código	Nome	Total Andares	Ativo
1	1	BLOCO_A	Bloco A - Salas de Aula	2	Sim

A Tabela de andares representa os pavimentos de cada prédio. A Tabela 3.6 apresenta os andares cadastrados no Bloco A, identificados por número (onde 0 representa o térreo e valores positivos representam andares superiores) e nome descritivo.

Tabela 3.6 – Dados Cadastrados - Tabela Andares

ID	Prédio ID	Número	Nome	Ativo
1	1	1	Primeiro Andar	Sim
2	1	0	Térreo	Sim

A Tabela de salas contém o nível mais detalhado da hierarquia, armazenando informações completas sobre cada ambiente monitorado. A Tabela 3.7 apresenta as salas cadastradas, incluindo código identificador, nome, capacidade de pessoas, tipo de sala e disponibilidade de equipamentos como projetor, computadores e ar condicionado.

Tabela 3.7 – Dados Cadastrados - Tabela Salas

ID	Andar ID	Código	Nome	Capac.	Tipo	Proj.	Comp.	A/C
1	1	101	Sala 101	10	Aula	Sim	Sim	Sim
2	1	102	Sala 102	35	Aula	Sim	Não	Sim
3	1	103	Sala 103	30	Aula	Não	Sim	Não
4	1	104	Sala 104	45	Aula	Sim	Sim	Sim
5	1	auditorio	Auditório Principal	150	Audit.	Sim	Sim	Sim

Legenda: Capac. = Capacidade; Proj. = Projetor; Comp. = Computadores; A/C = Ar Condicionado

A Tabela de histórico de ocupação armazena dados temporais de utilização das salas, mantendo registros dos últimos sete dias. A Tabela 3.8 apresenta uma amostra de dados coletados da Sala 101 durante o dia 19 de novembro de 2025, mostrando a média de ocupação registrada a cada hora pelo sistema.

Tabela 3.8 – Dados Cadastrados - Tabela Histórico de Ocupação

ID	Sala ID	Data/Hora	Dia Semana	Ocupação
2494	101	19/11/2025 09:00	Terça-feira	1,06
2495	101	19/11/2025 10:00	Terça-feira	1,07
2497	101	19/11/2025 11:00	Terça-feira	1,26
2499	101	19/11/2025 12:00	Terça-feira	0,73
2501	101	19/11/2025 13:00	Terça-feira	0,57
2505	101	19/11/2025 14:00	Terça-feira	0,37
2507	101	19/11/2025 15:00	Terça-feira	0,45
2509	101	19/11/2025 16:00	Terça-feira	0,50
2511	101	19/11/2025 17:00	Terça-feira	0,03
2513	101	19/11/2025 18:00	Terça-feira	0,00

A Tabela de estatísticas diárias consolida informações agregadas por dia, armazenando dados por até trinta dias. A Tabela 3.9 apresenta alguns dados estatísticos da Sala 101 coletados entre outubro e novembro de 2025, incluindo as médias máxima e mínima de ocupação diária com seus respectivos horários de ocorrência, além da estimativa de consumo de energia em quilowatt-hora (kWh).

Tabela 3.9 – Dados Cadastrados - Tabela Estatísticas Diárias

ID	Sala ID	Data	Ocupação Máx.	Hora Máx.	Ocupação Mín.	Energia (kWh)
510	101	26/10/2025	0,51	17:00	0,19	1,98
519	101	27/10/2025	1,47	15:00	0,02	9,92
523	101	28/10/2025	5,54	10:00	0,07	18,26
535	101	29/10/2025	0,68	17:00	0,36	3,58
542	101	30/10/2025	5,83	15:00	0,16	22,04
560	101	31/10/2025	1,76	11:00	0,17	15,98
578	101	03/11/2025	2,42	13:00	0,05	16,01
596	101	04/11/2025	5,97	11:00	0,15	18,16
608	101	06/11/2025	6,90	16:00	0,05	19,80
627	101	07/11/2025	1,31	11:00	0,10	11,52

3.5.3 Sistema de Cache Redis

O Redis (Redis Ltd., 2025) é utilizado para armazenamento de dados de ocupação em tempo real, proporcionando acesso instantâneo às informações atuais:

- **Estrutura de Chaves:** `sala:id:ocupacao`
- **Formato de Dados:** JSON com ocupação e timestamp
- **Persistência:** Dados mantidos indefinidamente até nova atualização

- **Configuração:** Host, porta e database configuráveis via ambiente

Exemplo de dados Redis:

```
Chave: "sala:101:ocupacao"
Valor: {
  "ocupacao": 25,
  "timestamp": "2025-09-22T10:30:00.000Z"
}
```

3.6 Conclusões Parciais

Nesta seção foi apresentado o planejamento da solução, bem como os requisitos de hardware e software necessários para reproduzir o sistema de monitoramento, detalhando o hardware, seu firmware, a API, o armazenamento de dados em banco e como foi simulado o acionamento de dispositivos eletrônicos da sala.

4 TESTES ANÁLISE E RESULTADOS

Essa seção tem por objetivo apresentar o resultado da solução, mostrando a interface Web e os testes executados, avaliando se o sistema de detecção e registro de ocupação funcionou como esperado, comparando os dados gerados, com a realidade observada. As avaliações foram divididas em análise quantitativa, para avaliar erros e acertos durante um tempo determinado, comparando os dados obtidos com os dados reais e também de forma qualitativa, avaliando a confiabilidade da comunicação, o comportamento dos dispositivos eletrônicos e o desempenho da interface web.

4.1 Página web

A página web tem como objetivo fornecer uma interface visual e responsiva para monitoramento em tempo real da ocupação das salas cadastradas no sistema. O frontend foi desenvolvido utilizando o framework **Next.js** (VERCEL, 2025), baseado em React (PLATFORMS, 2025), o que permite funcionalidades como renderização do lado do servidor (SSR), geração estática de páginas (SSG) e otimizações de desempenho nativas da plataforma.

A aplicação foi preparada para ser implantada na plataforma **Vercel** (INC., 2025), que oferece integração nativa com projetos Next.js, além de suporte a variáveis de ambiente, monitoramento e logs.

O desenvolvimento da interface web fez uso das seguintes tecnologias e ferramentas:

- **Next.js**: Framework moderno baseado em React para construção de aplicações web escaláveis e performáticas.
- **TypeScript**: Superset de JavaScript que adiciona tipagem estática, aumentando a confiabilidade do código.
- **Tailwind CSS**: Framework utilitário de estilização que permite a construção rápida de interfaces responsivas e modernas.
- **ESLint e PostCSS**: Utilizados para manter a padronização do código e processamento avançado de estilos CSS.

4.1.1 Funcionalidades e Páginas

A aplicação é dividida nas seguintes seções:

- Página inicial de login
- Página de seleção de salas
- Página com mapa e ocupação em tempo real
- Página de detalhes e estatísticas da sala.

4.1.1.1 Página Inicial de login

Exibe uma tela de login básico para permitir o acesso apenas de pessoas autorizadas. A página inicial pode ser vista na Figura 4.1, que mostra os campos de entrada com credenciais válidas de acesso ao sistema.

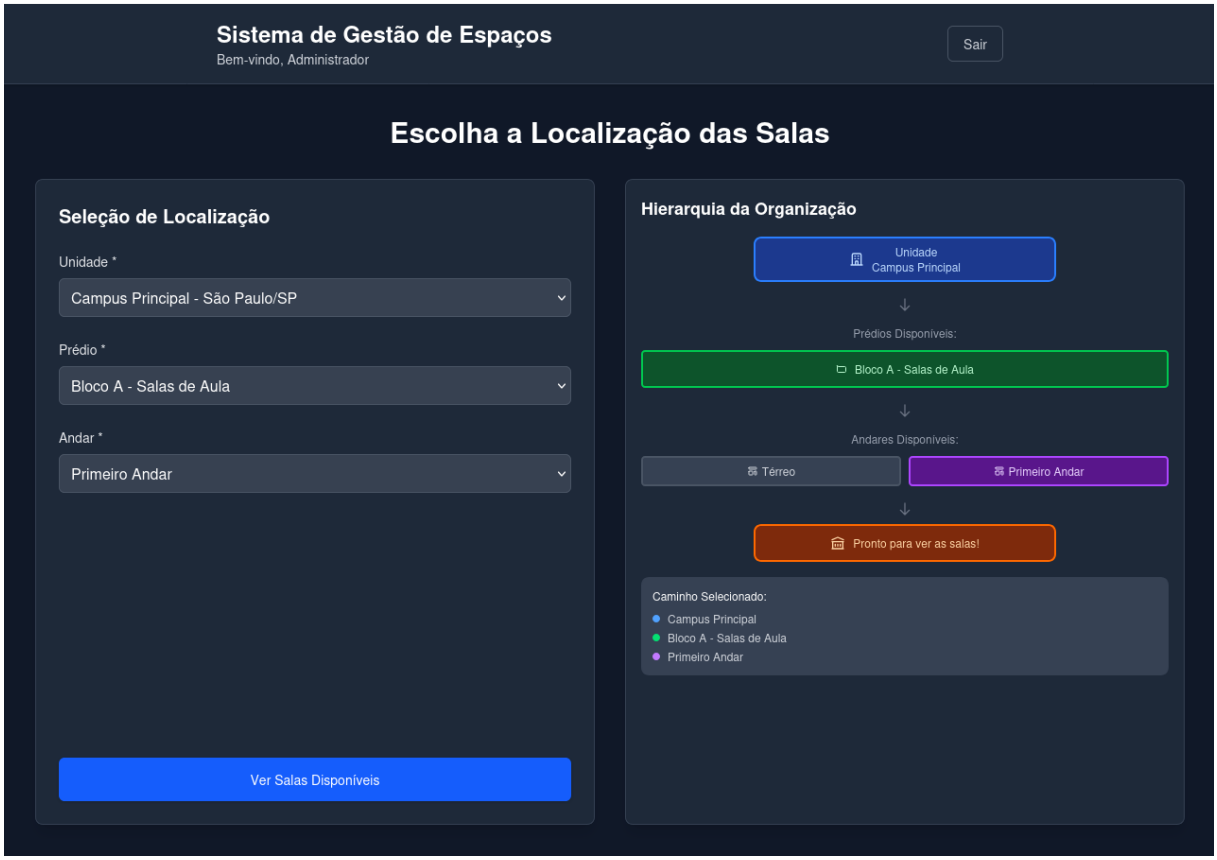
Figura 4.1 – Página de login.

Fonte: Próprio autor

4.1.1.2 Página de seleção do ambiente monitorado

Exibe as unidades cadastradas no sistema e os prédios disponíveis dentro de cada unidade. Ao selecionar um prédio, o usuário é redirecionado para o mapa correspondente. A página de seleção de ambientes pode ser vista na Figura 4.2, que mostra a seleção do primeiro andar do bloco A do Campus principal á esquerda e do lado direito, uma representação em blocos da hierárquia que foi selecionada.

Figura 4.2 – Seleção da localização das salas.



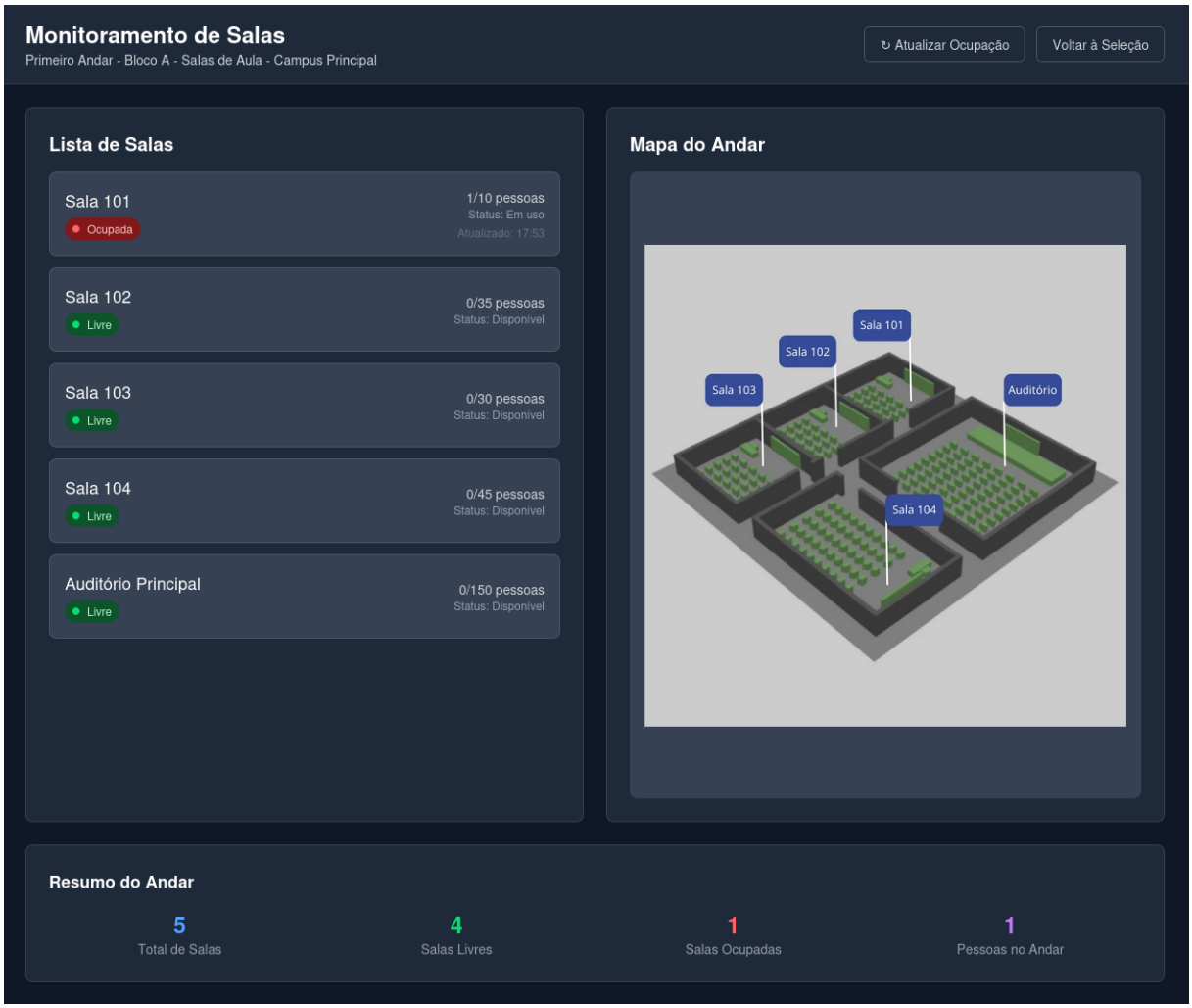
Fonte: Próprio autor

4.1.1.3 Página com mapa e ocupação em tempo real

Representa visualmente as salas de um andar do prédio específico selecionado. Esta página realiza atualizações periódicas, consultando o estado atual de ocupação das salas a cada minuto.A página com o mapa e a ocupação em tempo real pode ser vista na Figura 4.3. Nela, é possível observar, à direita, o mapa das salas existentes no andar selecionado; à esquerda, pode ser vista a ocupação em tempo real de cada sala, indicando o status livre/ocupada e o número de

pessoas presentes.

Figura 4.3 – Mapa e ocupação em tempo real.



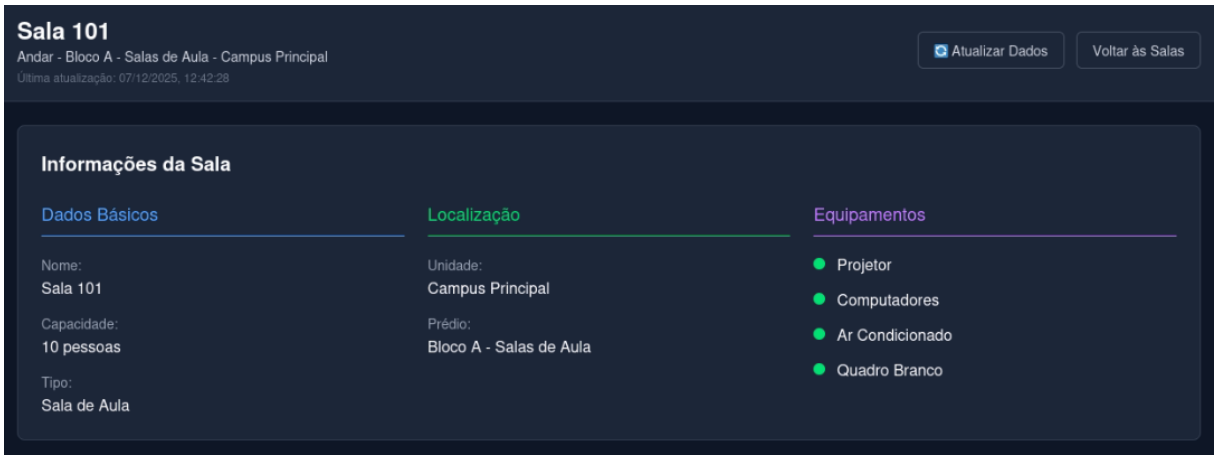
Fonte: Próprio autor

4.1.1.4 Página de detalhes e dados de ocupação da sala

Essa página exibe informações detalhadas sobre uma sala selecionada. Ela foi dividida em 3 seções que podem ser vistas a seguir.

- **Seção 1:** A Figura 4.4 representa essa seção, ela apresenta as informações de uma sala, dados básicos de identificação com nome, capacidade e tipo de sala, sua localização e os equipamentos que essa sala possui.

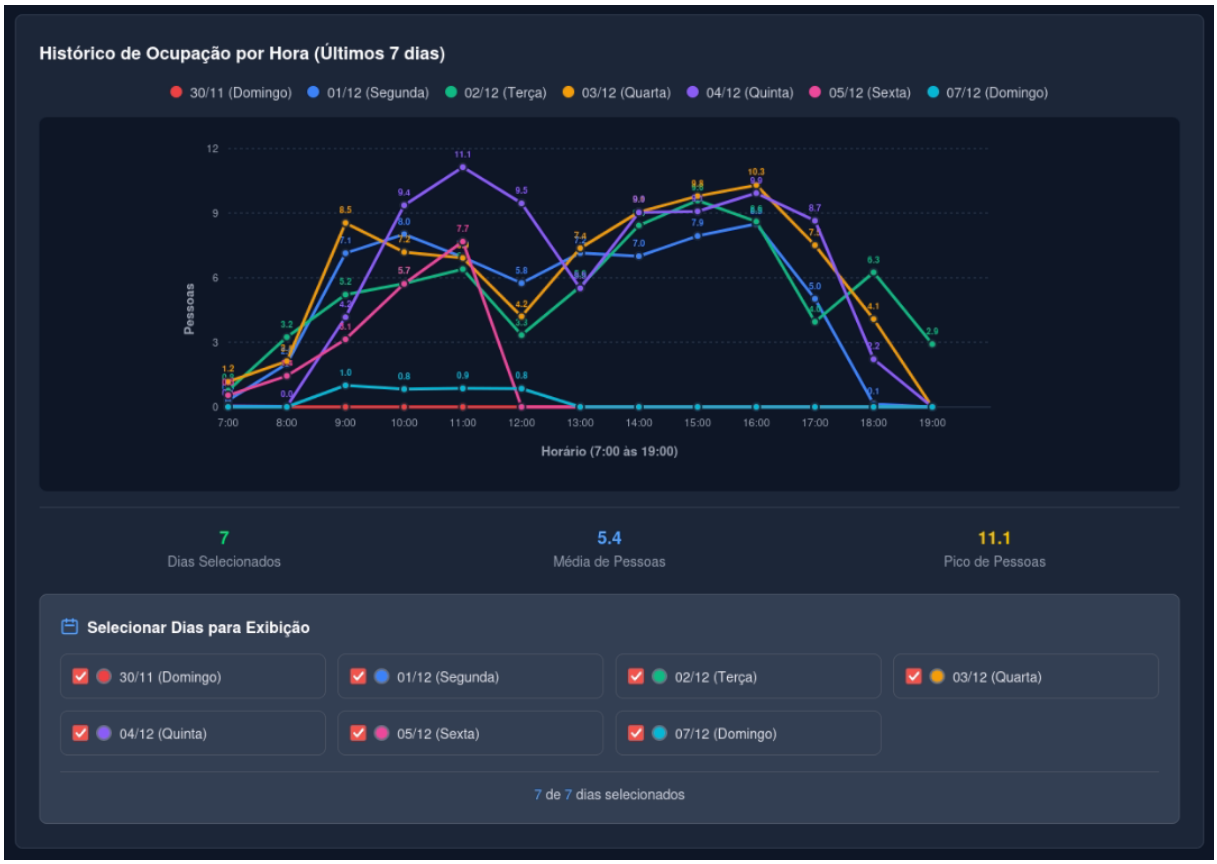
Figura 4.4 – Detelhes e dados 1º parte.



Fonte: Próprio autor

- **Seção 2:** A Figura 4.5 representa essa seção, ela apresenta um gráfico que mostra dados históricos de ocupação média por hora, tendo como referência a data atual. É possível selecionar os dias exibidos no gráfico e visualizar estatísticas que indicam a média diária de ocupação.

Figura 4.5 – Detelhes e dados 2º parte.



Fonte: Próprio autor

- **Seção 3:** A Figura 4.6 representa essa seção, ela apresenta dados de utilização da sala como ocupação máxima, o horário em que essa ocupação máxima ocorreu e o gasto de energia estimado, calculado pelo hardware principal e enviado à API. O consumo de energia é estimado multiplicando o tempo em que a sala permaneceu ocupada pelo consumo individual de cada equipamento. Essa estimativa considera que todos os dispositivos são acionados automaticamente quando há presença detectada na sala e desligados quando não há ocupação.

Figura 4.6 – Detelhes e dados 3º parte.

Fonte: Próprio autor

4.2 Análise Quantitativa

4.2.1 Procedimentos para Coleta de Dados

Para validar a eficácia do sistema desenvolvido, foi instalado um dispositivo em uma das salas da empresa onde o autor trabalha. Este dispositivo operou continuamente durante meses, realizando a coleta de dados sobre a ocupação do ambiente em tempo real. Para a validação da assertividade dos dados de médias informado pela solução, foi feita uma comparação entre os dados informados pela solução na Página WEB e armazenados no banco de dados, com a média real observada no ambiente monitorado.

Os dados de ocupação média de um dos dias mais movimentados do período de testes, foram armazenados para comparação com a média real observada. A média de ocupação real observada no escritório, foi obtida da seguinte forma: paralelamente ao firmaware principal de controle, o hardware principal armazenou frames de dez em dez minutos, esses frames foram agrupados de acordo com o horário em que foram capturados, o autor contou a quantidade de pessoas em cada frame e fez a média de pessoas presentes em uma hora de acordo com o 6 frames capturados de dez em dez minutos.

4.2.2 Resultado da Comparação entre dados de média da solução e dados de médias reais

O dia escolhido para representação nos testes foi o dia 01/12/2025, em uma semana de evento na empresa, que ocasionou maior ocupação na área monitorada pela solução. O gráfico

de ocupação mostrado pela solução neste dia pode ser visto na Figura 4.7

Figura 4.7 – Gráfico de médias de ocupação por hora do dia 01/12/2025.

Fonte: Próprio autor

Neste gráfico é possível observar o comportamento de ocupação do escritório ao longo do dia. No período inicial de monitoramento, entre 7h00 e 8h00, ocorre o horário em que a faxineira limpa o ambiente antes do início do expediente. Nesse mesmo intervalo, chega um dos engenheiros que trabalha no local, o que explica a média de ocupação passando de aproximadamente 0,3 para 2 pessoas. Outro comportamento relevante ocorre no horário de almoço, por volta de 12h00, quando a ocupação média diminui. No entanto, ela não chega a zero, pois os colaboradores realizam suas refeições em horários distintos, mas essa diminuição foi observada em todos os dias. O expediente geralmente termina entre 17h00 e 18h00, e isso pode ser observado pela redução gradual da ocupação até chegar a zero por volta das 19h00.

A Tabela 4.1 mostra o resultado da comparação entre os dados de médias obtidos pela solução e os dados de médias calculados pelo autor através dos frames.

Tabela 4.1 – Tabela de comparação de médias — 01/12/2025 (Segunda-feira)

Horário	Detecção no banco	Contagem Real (Frame)	Erro absoluto
07:00	0.31	0.50	0.19
08:00	2.00	1.34	0.66
09:00	7.14	7.00	0.14
10:00	8.02	7.67	0.35
11:00	6.95	6.50	0.45
12:00	5.75	5.67	0.08
13:00	7.15	6.67	0.48
14:00	7.00	7.00	0.00
15:00	7.94	7.67	0.27
16:00	8.50	8.84	0.34
17:00	5.02	4.67	0.35
18:00	0.14	0.50	0.36
19:00	0.00	0.00	0.00

O erro médio absoluto (MAE, do inglês *Mean Absolute Error*) é uma métrica amplamente utilizada para avaliar a diferença média entre valores reais e valores estimados por um sistema de detecção. Ele expressa, em unidades absolutas, o quão distante o modelo está, em média, do valor correto.

O MAE é definido pela Equação 4.1:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{4.1}$$

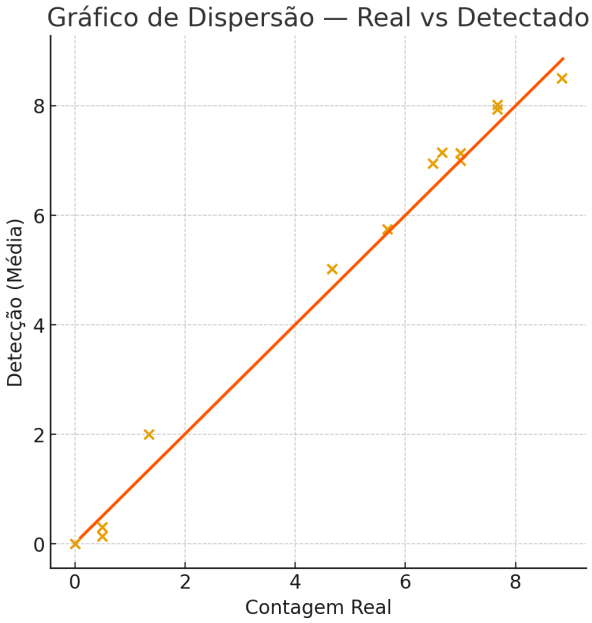
onde y_i representa o valor real, $\hat{y}_i$ representa o valor detectado pelo sistema, e n é o número total de observações analisadas. O erro média absoluto do sitema pode ser vito em na Equação 4.2

$$MAE = \frac{3.67}{13} \approx 0.282 \tag{4.2}$$

O sistema apresentou um erro médio absoluto de aproximadamente 0.28 pessoas, indicando que, em média, a diferença entre a contagem real e a detecção automática ficou abaixo de 1 pessoa ao longo do dia, o que demonstra boa precisão para a aplicação proposta.

Um gráfico de dispersão foi criado para comparar diretamente os valores da contagem real observada e a contagem detectada pelo sistema e pode ser visto na Figura 4.8.

Figura 4.8 – Gráfico de dispersão.



Fonte: Próprio autor

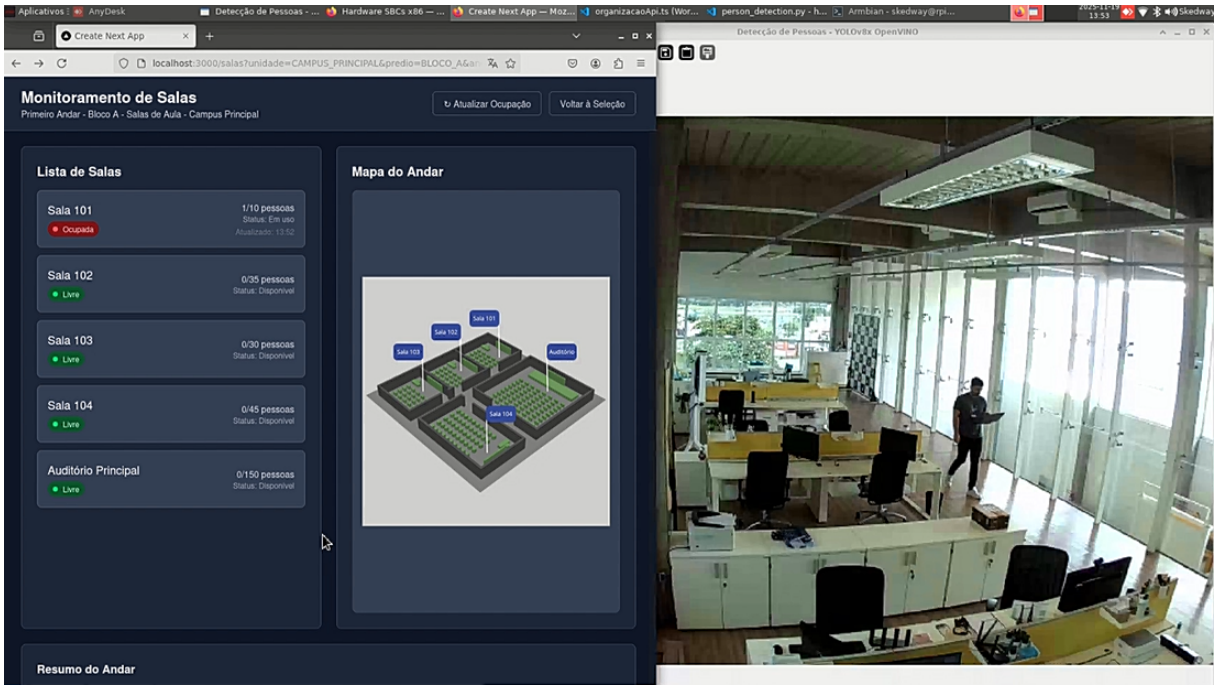
Em um cenário ideal, todos os pontos estariam exatamente sobre a linha y = x, que representa deteções perfeitas. A curva geral acompanha bem o padrão real, isso confirma que o sistema é capaz de capturar o comportamento de ocupação de forma consistente.

4.3 Análise Qualitativa

4.3.1 Confiabilidade de detecção

Para validar a confiabilidade da detecção durante o período de testes, as imagens e o valor de detecção na página web eram monitorados de tempos em tempos, como o exemplo da Figura 4.9, que mostra um print de tela do hardware, em que pode ser vista a ocupação gerada pelo sistema na página web e a respectiva imagem em tempo real.

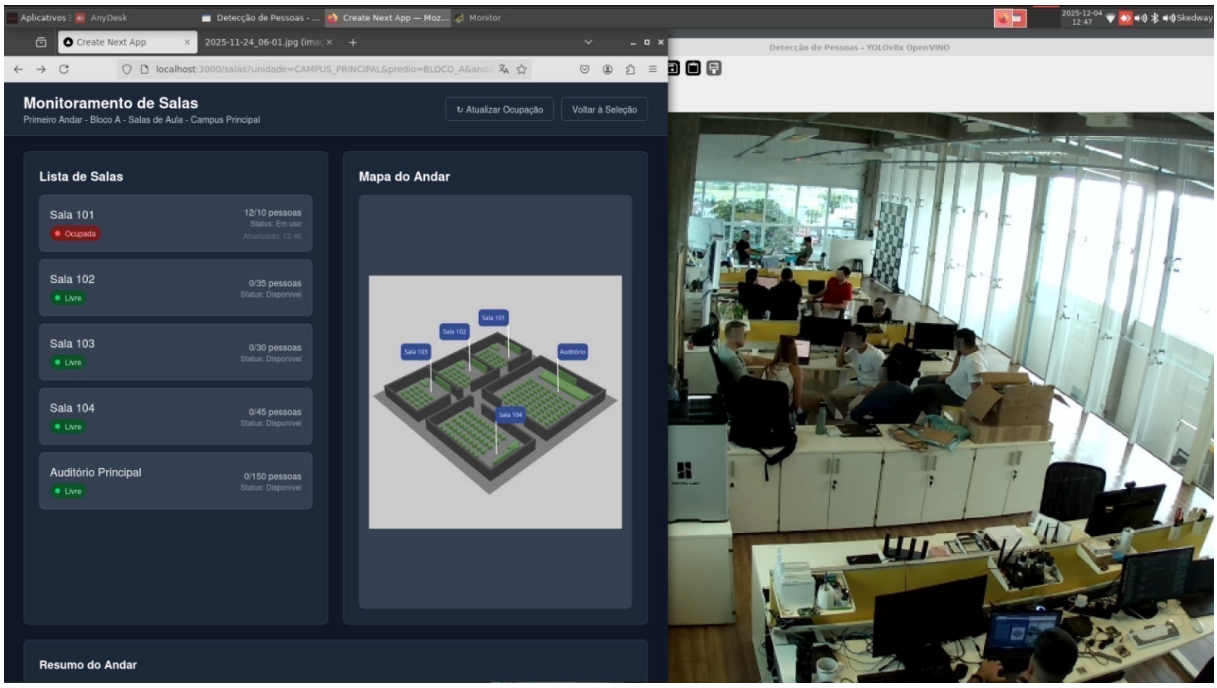
Figura 4.9 – Monitoramento em tempo real 1.



Fonte: Próprio autor

A Figura 4.10 mostra mais um exemplo, porém com uma ocupação maior, com um total de 12 pessoas.

Figura 4.10 – Monitoramento em tempo real 2.



Fonte: Próprio autor

Dessa forma foi possível validar que a solução operou como esperado, contando corretamente o número de pessoas no ambiente.

4.3.2 Desempenho da Interface Web

A interface web funcionou corretamente, foram necessários ajustes de tempo de atualização das páginas e também adequação das requisições para utilizar menos recursos da API e do banco de dados, pois os serviços utilizados para hospedagem foram grátis, e possuiam limitações de requisições. Por conta dessas limitações alguns testes e imagens de evidencias foram gerados quando a solução estava sendo executada localmente.

4.3.3 Comportamento dos Dispositivos Eletrônicos

Para os teste com dispositivos eletrônicos, um dos Sonoff Basic (ITEAD Intelligent Systems Co., Ltd., 2025a) ficou ligado controlando uma fita LED, ligando os LEDs quando havia presença detectada e desligando os mesmos quando o ambiente estava vazio. Durante o período de testes não houve erros observados, o dispositivo ficou a uma distância máxima de 10 metros do hardware principal, respeitando a distância máxima para a comunicação via ESP-NOW.

4.4 Limitações Observadas

A detecção de pessoas nem sempre é exata pois o dispositivo analisa os frames de tempos em tempos e alguns erros podem ser gerados. Alguns erros observados foram devidos a pessoas se movimentando ou obstruídas por objetos à sua frente, reflexos em espelhos e confusão com bustos/imagens de pessoas em decoração. O modelo utilizado analisa formas de pessoas presentes em frames, dessas forma ele não consegue distinguir seres vivos de objetos inanimados, assim podemos ver uma limitação na utilização desse dispositivo em salas de aula de anatomia por exemplo, pois pode confundir um boneco utilizado nas aulas com uma pessoa. Salas com vidros podem gerar reflexo e duplicar o resultado de detecção, se o vidro for transparente, pode detectar pessoas fora da sala monitorada, isso deve ser levado em conta em sua instalação em um ambiente real não controlado.

4.5 Conclusões Parciais

Neste capítulo foi possível observar o comportamento da solução sendo aplicada em um ambiente real. O sistema funcionou de forma consistente, realizando a detecção e contagem de pessoas com boa precisão. A interface web operou corretamente após ajustes de atualização e volume de requisições. O controle de dispositivos eletrônicos, exemplificado pelo uso de um módulo Sonoff Basic para manipular uma fita LED conforme detecção de presença, funcionou sem falhas no período de testes. Por outro lado, foram observadas limitações relacionadas ao ambiente, como reflexos, obstruções e objetos que podem ser confundidos com pessoas, indicando que a instalação e o contexto influenciam diretamente o desempenho do sistema.

Um vídeo demonstrativo da solução foi gravado para ilustrar seu funcionamento. Nele é apresentado um pequeno teste em que a tela de um notebook acessa remotamente o hardware principal da aplicação. É possível observar a navegação entre as páginas web mencionadas no texto, bem como uma visualização dividida entre a página que mostra a ocupação em tempo real e a imagem capturada pela câmera do hardware, com o objetivo de evidenciar a atualização da página conforme o número de pessoas presentes na sala.

Além disso, para melhorar a representação visual no vídeo, a fita LED foi substituída por um giroflex como dispositivo controlado pela solução, simbolizando, de forma ilustrativa, o acionamento de iluminação do ambiente ou de outros equipamentos eletroeletrônicos. Os testes foram realizados localmente, pois, para a gravação do vídeo, o intervalo de atualização

da página web e do envio de dados de ocupação foi reduzido para tornar a demonstração mais dinâmica. Essa configuração exigiria grande quantidade de recursos dos servidores utilizados, que são gratuitos, podendo esgotar o limite disponível. Por esse motivo, a execução do teste ocorreu localmente.

O vídeo pode ser acessado no link a seguir: <https://drive.google.com/file/d/1vNkW4JrWqwNZ2fQmEdnwcUAde-_xEVT8/view?usp=sharing>

Os repositórios contendo os códigos desenvolvidos no projeto estão disponíveis nos links abaixo:

<https://github.com/LeandroNunesDuarteJunior/TCC_firmware/tree/main/hardware>

<https://github.com/LeandroNunesDuarteJunior/TCC_API_nestJS>

<https://github.com/LeandroNunesDuarteJunior/TCC_WEB_nextJS>

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo, são apresentadas as conclusões, as considerações sobre o trabalho desenvolvido e, também, discutidas propostas para trabalhos futuros.

5.1 Conclusões

O presente trabalho demonstra como a utilização de tecnologias conhecidas e acessíveis no mercado, podem ser combinadas para gerar uma solução de automação que combina controle e gerenciamento de espaços. Tecnologias amplamente conhecidas como visão computacional com (ULTRALYTICS, 2025), processamento embarcado em Raspberry Pi e integração com APIs, podem ser combinadas para auxiliar no gerenciamento de ocupação de ambientes. Os resultados indicam que, em condições adequadas de iluminação, posicionamento da câmera e fluxo de pessoas, o sistema apresenta um desempenho consistente e capaz de gerar informações úteis para monitoramento e tomada de decisão.

O cálculo do erro médio absoluto evidenciou uma diferença média de apenas 0,28 pessoa na comparação entre a média real calculada manualmente e a média calculada pela solução, resultado considerado satisfatório para aplicações de monitoramento em tempo real. Além disso, a comparação entre valores detectados e valores reais mostrou que o comportamento geral da ocupação ao longo do dia foi corretamente capturado, mesmo com pequenas variações.

Também ficou claro que o desempenho do sistema é sensível ao ambiente onde está instalado. Fatores como variação brusca de luminosidade, presença de superfícies reflexivas, sobreposição de pessoas e ângulos desfavoráveis podem influenciar negativamente a acurácia da detecção. Assim, apesar de robusta, a solução depende de adequações e calibrações específicas para cada local de implantação, a fim de garantir resultados consistentes.

De maneira geral, pode-se notar que a solução é viável, funcional e capaz de gerar informações importantes para gestão de espaços.

5.2 Trabalhos futuros

O sistema pode ser melhorado e utilizado para aplicações diferentes da proposta do presente sistema, alguns exemplos de trabalhos futuros a serem implementados são dados a seguir:

- Implementação de uma solução de reserva de salas: Adicionar uma funcionalidade para organizar a utilização dos espaços, permitindo que várias pessoas com autorização de acesso possam reservar as salas. Essa funcionalidade garantiria melhor gestão dos ambientes e evitaria conflitos de uso.

- Desenvolvimento de um aplicativo móvel: Criar um aplicativo para dispositivos móveis que replique as funcionalidades do console, proporcionando acesso prático e intuitivo às informações e controles do sistema, independentemente da localização do usuário.

- Adição de um modo de emergência: Utilizar o sistema para emitir mensagens por meio de um alto-falante conectado ao Raspberry Pi 5. Em situações de emergência, como a detecção de fogo ou de um objeto específico identificado pelo modelo (ULTRALYTICS, 2025), o sistema pode acionar um alarme sonoro, emitindo alertas claros e imediatos para garantir a segurança dos ocupantes.

REFERÊNCIAS

CHATTERJEE, N.; SINGH, A. V.; AGARWAL, R. You only look once (yolov8) based intrusion detection system for physical security and surveillance. In: **2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)**. [S.l.: s.n.], 2024. p. 1–5.

CONSORTIUM, S. **SQLite Documentation**. 2025. <https://www.sqlite.org/docs.html>. Acesso em: 24 nov. 2025.

CORPORATION, I. **OpenVINO Toolkit Documentation**. 2025. <https://docs.openvino.ai>. Acesso em: 24 nov. 2025.

CURIEL, G. et al. A computer vision based system for human detection and automatic people counting. **Transactions on Energy Systems and Engineering Applications**, v. 5, n. 2, p. 1–14, 2024.

DEVELOPERS, A. **Armbian Documentation**. [S.l.], 2025. Acessado em: 2025-01-01. Disponível em: <https://www.armbian.com>.

FOUNDATION, P. S. **Python 3 Documentation**. 2025. <https://docs.python.org/3/>. Acesso em: 24 nov. 2025.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. Disponível em: <https://www.deeplearningbook.org/>.

GOWDA, P.; GOWDA, A. Best practices in rest api design for enhanced scalability and security. **Journal of Artificial Intelligence, Machine Learning and Data Science**, v. 2, n. 1, p. 827–830, 2024.

Hi-Link Electronic. **HLK-LD2410C Human Presence Sensor Module**. [S.l.], 2023. Acessado em: 10/12/2025. Disponível em: <https://www.hlktech.net/>.

INC., V. **Vercel Documentation**. 2025. <https://vercel.com/docs>. Acesso em: 24 nov. 2025.

ITEAD Intelligent Systems Co., Ltd. **SONOFF Basic: DIY Smart Switch**. [S.l.], 2025. Acesso em: 21 nov. 2025. Disponível em: <https://sonoff.tech/products/sonoff-basicr4-wi-fi-smart-switch?_pos=2&_psq=sonoff+basic&_ss=e&_v=1.0>.

ITEAD Intelligent Systems Co., Ltd. **SONOFF Mini: DIY Smart Switch**. [S.l.], 2025. Acesso em: 21 nov. 2025. Disponível em: <https://sonoff.tech/products/sonoff-zigbee-smart-roller-shutter-switch-mini-zbrbs>.

LIU, W. et al. Ssd: Single shot multibox detector. In: **European Conference on Computer Vision (ECCV)**. [S.l.]: Springer, 2016. p. 21–37.

MOSTAFA, S. A.; ABUSALIM, S. W.; SARINGAT, M. Z. A comparative study of data management systems. **Journal of Soft Computing and Data Mining**, v. 1, n. 1, p. 10–16, 2020.

NestJS Framework. **NestJS Documentation**. 2025. <https://nestjs.com/>. Acesso em: 01 fev. 2025.

OASIS. **MQTT Version 3.1.1 Plus Errata 01**. 2015. <https://mqtt.org>. Acesso em: 24 nov. 2025.

OpenCV.org. **OpenCV-Python Documentation**. 2025. <https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html>. Acessado em: 20 nov. 2025.

Oracle Corporation. **MySQL Documentation**. [S.l.], 2025. Acesso em: 21 nov. 2025. Disponível em: <https://dev.mysql.com/doc/>.

PAGUAY, J. A. C. et al. Secure home automation system based on esp-now mesh network, mqtt and home assistant platform. **IEEE Latin America Transactions**, v. 21, n. 7, p. 829–838, 2023.

PHAM, A. D. Developing back-end of a web application with nestjs framework: Case: Integrify oy's student management system. 2020.

PLATFORMS, M. **React Documentation**. 2025. <https://react.dev>. Acesso em: 24 nov. 2025.

Redis Ltd. **Redis Documentation**. [S.l.], 2025. Acesso em: 21 nov. 2025. Disponível em: <https://redis.io/docs/>.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 779–788.

REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. **Advances in Neural Information Processing Systems**, v. 28, 2015.

SAFFARI, A. et al. Battery-free camera occupancy detection system. In: **Proceedings of the 5th international workshop on embedded and mobile deep learning**. [S.l.: s.n.], 2021. p. 13–18.

SOUZA, F. M. C.; LIMA, E. C. S.; CARIDADE, E. R. de S. Criando sistema escalável de agendamentos utilizando typescript com nestjs no backend e nextjs no frontend. **Revista Ibero-Americana de Humanidades, Ciências e Educação**, v. 8, n. 12, p. 43–57, 2022.

SRIVASTAVA, S. et al. A comprehensive review of next. js technology: Advancements, features, and applications. **Features, and Applications (May 16, 2024)**, 2024.

SYSTEMS, E. **ESP-NOW User Guide**. 2025. <https://www.espressif.com/en/solutions/low-power-solutions/esp-now>. Acesso em: 24 nov. 2025.

TERVEN, J.; CÓRDOVA-ESPARZA, D.-M.; ROMERO-GONZÁLEZ, J.-A. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. **Machine Learning and Knowledge Extraction**, MDPI, v. 5, n. 4, p. 1680–1716, 2023.

ULTRALYTICS. **YOLOv8 Documentation**. 2025. <https://docs.ultralytics.com/models/yolov8/>. Acesso em: 24 nov. 2025.

VERCEL. **Next.js Documentation**. 2025. <https://nextjs.org/docs>. Acesso em: 24 nov. 2025.

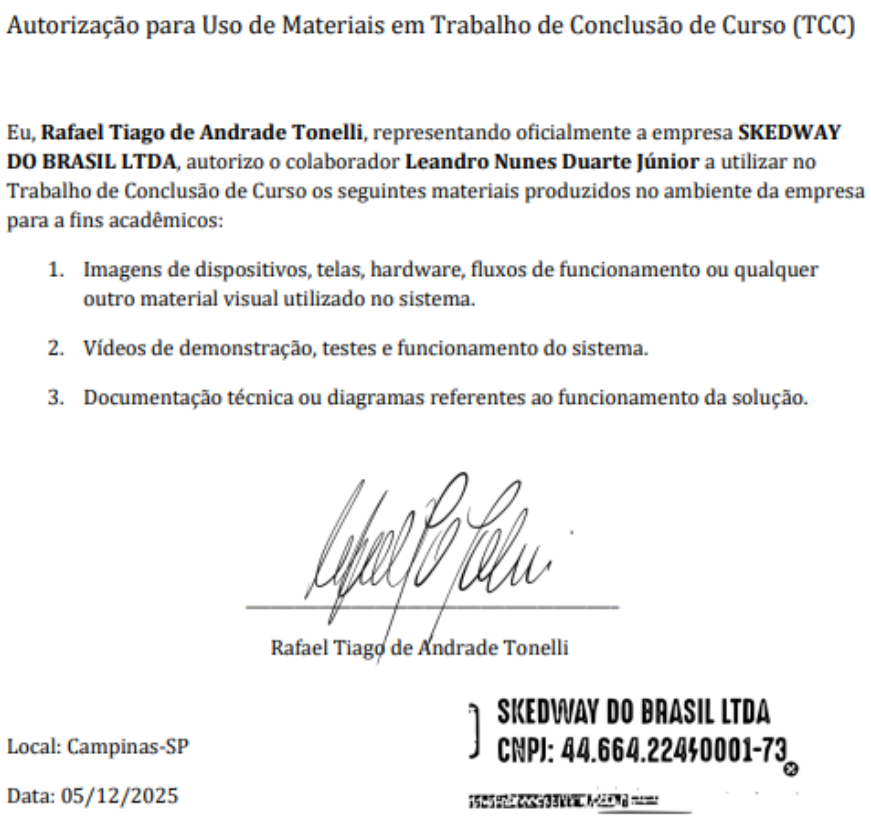
ViaCEP. **ViaCEP: Webservice de CEP**. 2025. <https://viacep.com.br/>. Acesso em: 10 dez. 2025.

ZHANG, W. et al. Vision-based occupancy detection in indoor environments: A comparison of standard rgb and thermal cameras. **Journal of Building Engineering**, Elsevier, p. 114215, 2025.

ČABARKAPA, V. et al. Integration of system for ambient conditions monitoring and regulation in smart homes using the mqtt protocol. In: **2024 23rd International Symposium INFOTEH-JAHORINA (INFOTEH)**. [S.l.: s.n.], 2024. p. 1–6.

APÊNDICE A – AUTORIZAÇÃO DA EMPRESA PARA USO DE MATERIAIS

Figura A.1 – Autorização assinada.



Autorização para Uso de Materiais em Trabalho de Conclusão de Curso (TCC)

Eu, **Rafael Tiago de Andrade Tonelli**, representando oficialmente a empresa **SKEDWAY DO BRASIL LTDA**, autorizo o colaborador **Leandro Nunes Duarte Júnior** a utilizar no Trabalho de Conclusão de Curso os seguintes materiais produzidos no ambiente da empresa para a fins acadêmicos:

1. Imagens de dispositivos, telas, hardware, fluxos de funcionamento ou qualquer outro material visual utilizado no sistema.
2. Vídeos de demonstração, testes e funcionamento do sistema.
3. Documentação técnica ou diagramas referentes ao funcionamento da solução.

Rafael Tiago de Andrade Tonelli

SKEDWAY DO BRASIL LTDA
CNPJ: 44.664.224\0001-73

Local: Campinas-SP

Data: 05/12/2025

Fonte: Próprio autor