Sistema para Monitoramento de Distanciamento Social Utilizando Câmeras Inteligentes

Luiz Alexsandro Ferreira de Vasconcelos Junior

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2023

Luiz Alexsandro Ferreira de Vasconcelos Junior

Sistema para Monitoramento de Distanciamento Social Utilizando Câmeras Inteligentes

Monografia apresentada ao curso Engenharia de Computação
do Centro de Informática, da Universidade Federal da Paraíba,
como requisito para a obtenção do grau de Bacharel em
Engenharia de Computação

Orientador: Dr. Alisson Vasconcelos de Brito

Dezembro de 2023

Catalogação na publicação
Seção de Catalogação e Classificação

V331s Vasconcelos Junior, Luiz Alexsandro Ferreira de.
Sistema para monitoramento de distanciamento social utilizando câmeras inteligentes / Luiz Alexsandro Ferreira de Vasconcelos Junior. - João Pessoa, 2023.
54 f. : il.

Orientação: Alisson Vasconcelos de Brito.
TCC (Graduação) - UFPB/CI.

1. Distanciamento social. 2. Yolo. 3. Visão computacional. 4. Estimação de distância. I. Brito, Alisson Vasconcelos de. II. Título.

UFPB/CI CDU 004.8

Elaborado por Michelle de Kássia Fonseca Barbosa - CRB-738

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Engenharia de Computação intitulado ***Sistema para Monitoramento de Distanciamento Social Utilizando Câmeras Inteligentes*** de autoria de Luiz Alexsandro Ferreira de Vasconcelos Junior, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. Dr. Alisson Vasconcelos de Brito
Universidade Federal da Paraíba

Prof. Me. Maelson Bruno Pacheco Nunes Pereira
Universidade Federal da Paraíba

Prof. Dr. Romulo Calado Pantaleao Camara
Universidade Federal da Paraíba

Coordenadora do Curso de Engenharia de Computação
Veronica Maria Lima Silva
CI/UFPB

João Pessoa, 19 de dezembro de 2023

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

"Faça o que puder, com o que tiver, onde estiver." – Theodore Roosevelt

RESUMO

Sistema Para Monitoramento De Distanciamento Social Utilizando Câmeras Inteligentes proposto neste trabalho tem como finalidade permitir o acompanhamento em tempo real de um espaço, essa aplicação pode ser utilizada em diversas situações, como para reduzir a circulação de pessoas em espaços coletivos, tal como permitir maior controle em um determinado ambiente evitando assim aglomerações. O sistema pode ser utilizada em shoppings para saber em que espaços se dão a maior concentração de pessoas ou até mesmo para conter o avanço de doenças que podem ser transmitidas pelo ar, como aconteceu com a doença causada pelo vírus SARS-CoV-2, popularmente conhecido como coronavírus (DAS et al., 2021). A partir disso, é desenvolvido e implementado um sistema com uma câmera monocular conectado a uma raspberry onde é executado um código capaz de detectar pessoas em uma imagem e calcular a distância entre as mesmas, além disso, também é disponibilizado uma interface web que permite gerenciar os dispositivos bem como acompanhar as informações que são envidas em tempo real pelos mesmos. Os resultados obtidos revelam que o sistema alcançou uma taxa de detecção de aproximadamente 49% considerando um Intersection Over Union de 50% entre as caixas delimitadoras adicionas no pré-processamento e as previstas pelo modelo, para um Intersection Over Union de 50% a 95% a taxa de detecção chega a 99%. O algoritimo responsável por mensurar a distância teve um erro percentual absoluto médio de 18.45%. Esses resultados demonstram que a aplicação pode ser utilizada de maneira viável para ajudar tanto as forças de seguranças públicas bem como para fins comerciais.

Palavras-chave: <distanciamento social>, <yolo>, <estimação de distância>, <visão computacional>.

ABSTRACT

The purpose of the Social Distancing Monitoring System Using Smart Cameras proposed in this work is to allow real-time monitoring of a space. This application can be used in a variety of situations, such as to reduce the movement of people in collective spaces, as well as allowing greater control in a given environment, thus avoiding agglomerations. In addition, this system can be useful for containing the spread of diseases that can be transmitted through the air, as happened with the disease caused by the SARS-CoV-2 virus, popularly known as coronavirus (DAS et al., 2021). From this, a system is designed and implemented with a monocular camera connected to a raspberry where a code capable of detecting people in an image and calculating the distance between them is executed. Furthermore, a web interface is also made available that allows managing devices as well as monitoring the information that is sent in real time by them. The results obtained reveal that the system achieved a detection rate of approximately 49% considering an Intersection Over Union of 50% between the bounding boxes added in pre-processing and those predicted by the model, for an Intersection Over Union of 50% at 95% the detection rate reaches 99%. The algorithm responsible for measuring the distance had an average absolute percentage error of 18.45%. These results demonstrate that the application can be viably used to help both public security forces and commercial purposes.

Key-words: <social distacing>, <yolo>, <distance measure>, <computer vision>.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

YOLO – *You Only Look Once*

API - *Application Programming Interface*

SQL - *Structured query language*

PIB - *Produto Interno Bruto*

OMS - *Organização Mundial da Saúde*

SPA - *Single Page Application*

JSX - *JavaScript XML*

HTML - *Hypertext Mark-up Language*

HTTP - *Hypertext Transfer Protocol*

DOM - *Document Object Model*

SSR - *Server-Side Rendering*

IoU - *Interpolation Over Union*

FPS - *Frames Per Second*

Sumário

INTRODUÇÃO

Tecnologias para monitoramento e gestão de ambiente tem se tornado cada vez mais necessárias, principalmente com a urbanização acelerada. A tecnologia pode ser um enorme aliado em ajudar a garantir espaços mais seguros e eficientes através do monitoramento em tempo real dos ambientes. A necessidade de evitar aglomerações, controlar o fluxo de pessoas e garantir a segurança em ambientes públicos e privados trouxe à tona a importância de sistemas de câmeras inteligente.

Um sistema de câmera inteligente incorporam algoritmos avançados de análise de vídeo, aprendizado de máquina e visão computacional para fornecer informações valiosas em tempo real (BELBACHIR PETER MICHAEL GöBEL (AUTH.), 2010). Essas câmeras possuem a capacidade de detectar objetos e fornecer dados precisos para a tomada de decisões, dessa maneira torna-se evidente como sistemas de câmeras inteligentes podem ser úteis para auxiliar na gestão de espaços e interações fornecendo informações a respeito do ambiente.

Além disso, sistemas de câmeras inteligentes tem se tornado uma peça fundamental para a segurança, fornecendo as forças de seguranças uma visão mais abrangente dos ambientes, monitorando atividades suspeitas e capturando evidências, além de ajudar de maneira preventiva desestimulando atividades criminosas (CUNHA, 2018).

Este trabalho tem como objetivo realizar a construção de um sistema que permita o monitoramento em tempo real da distância entre as pessoas por meio de uma câmera inteligente. Além disso, busca disponibilizar uma aplicação para o acesso às informações geradas pelo sistema. Para atingir esse propósito, utilizaremos uma avançada ferramenta de visão computacional para a detecção de objetos em tempo real. A escolha pelo *YOLO*(You Only Look Once) se baseia na sua eficácia e velocidade quando comparado a outras soluções (SRIVASTAVA et al., 2021).

O trabalho em questão pode trazer diversas contribuições para a sociedade, uma vez que permitirá o acompanhamento em tempo real de ambientes podendo ser utilizado para melhorar a segurança em espaços públicos e privados, otimizar a organização de eventos e podendo ser utilizado em situações de crise, auxiliando a manter o distanciamento social entre as pessoas, medida que foi necessário durante a pandemia causada pelo rápido avanço do COVID-19 (OMS, 2021).

1 ESCOPO DO TRABALHO

A seguir será apresentado os objetivos gerais, objetivos específicos e uma breve descrição do que será abordado no decorrer deste trabalho.

1.0.1 Objetivo geral

Implementar um sistema para monitoramento de distanciamento social utilizando câmeras inteligentes, o sistema deve ser capaz de detectar pessoas bem como mensurar a distância entre elas, também deve ser possível consultar as informações através de uma aplicação.

1.0.2 Objetivos específicos

- Detectar pessoas em uma imagem.
- Alertar quando as pessoas estiverem muito próximas uma das outras.
- Desenvolver uma interface para a consulta das informações processadas.
- Executar o sistema em uma *Raspberry*

1.1 Estrutura da monografia

No primeiro capítulo é possível encontrar a introdução, definição do problema, objetivo geral e o objetivo especifico que pretende-se ser alcançado com a conclusão deste trabalho. No segundo capítulo é realizada uma síntese sobre as tecnologias que serão abordadas no desenvolvimento do trabalho para facilitar o entendimento de quem venha a ter interesse na área. No terceiro capítulo é apresentada a metodologia do trabalho, ou seja os métodos e estratégias utilizadas para chegar a implementação do sistema. No quarto capítulo são apresentados os resultados obtidos com a implementação do sistema, e por último o quinto capítulo onde é exposta a conclusão bem como sugestões para trabalhos futuros.

2 CONCEITOS GERAIS

A seguir são apresentados os conceitos que servem de base para o sistema proposto, a revisão das tecnologias, conceitos e técnicas para mensurar a distância entre as pessoas em uma imagem, imprescindíveis para o desenvolvimento do trabalho.

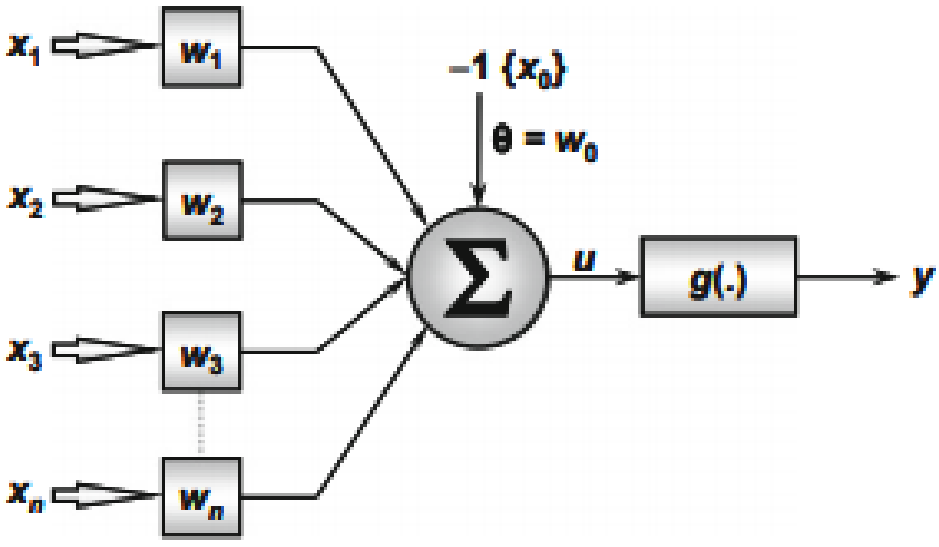
2.1 Aprendizado de Máquina

O aprendizado de máquina está por trás de diversas aplicações na atualidade, desde recomendações de conteúdos em plataformas como YouTube até um filtro de spam nas aplicações de e-mails. Através do aprendizado de máquina computadores se tornam capazes de tomar decisões e realizar tarefas com base em exemplos e experiências fornecida ao mesmo sem a necessidade de serem explicitamente programados para performarem tais atividades (SAMUEL, 1959).

2.1.1 Rede Neural

Uma rede neural é um sistema computacional que funciona de forma similar ao cérebro, enquanto no cérebro a unidade básica é um neurônio, em uma rede neural artificial essa unidade básica seria um perceptron, capaz de realizar processamento de informações simples, os perceptron podem ser combinados formando uma malha de neurônios (SILVA, 2017).

Figura 1: Ilustração de uma rede Perceptron



Fonte: (SILVA, 2017)

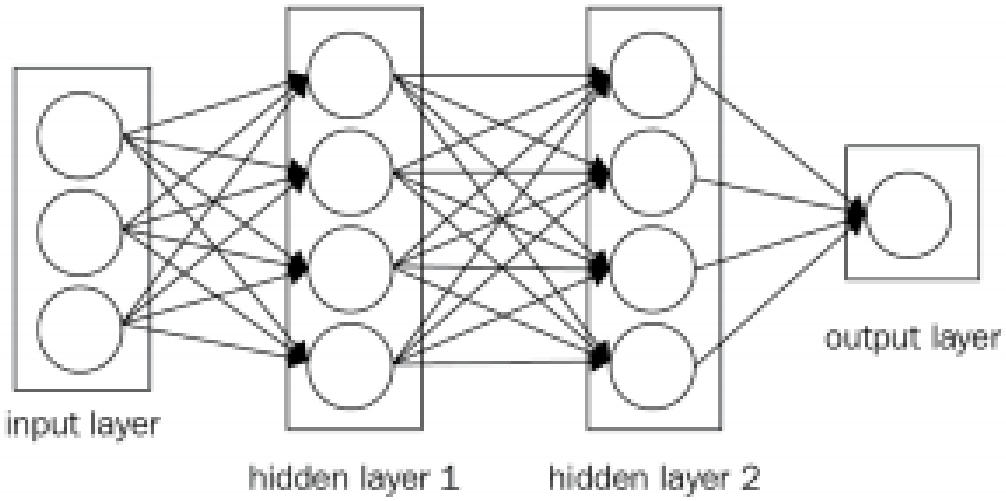
Conforme pode ser observado na imagem acima, as entradas X representam informações do comportamento do processo a serem mapeadas, essas entradas por sua vez serão ponderadas por pesos sinápticos W que quantificarão a importância de cada umas das variáveis de entradas afim de mapear o comportamento de entrada/saída do processo em questão.

O resultado do produto entre as entradas e pesos são combinados linearmente com um bias que é aprendido durante o treinamento, o resultado desse processo é passado para uma função de ativação que pode ser por exemplo uma função Sigmoide que produzirá a saída Y.

2.1.2 Rede Neural Convolucional

Uma rede neural convolucional é muito similar a uma rede neural regular como a perceptron, possuindo pesos sinápticos e bias que são aprendidos durante o processo de treinamento. A rede neural regular recebe como entrada um vetor de uma dimensão onde todos os neurônios estão completamente conectados aos neurônios da camada anterior (SILVA, 2017).

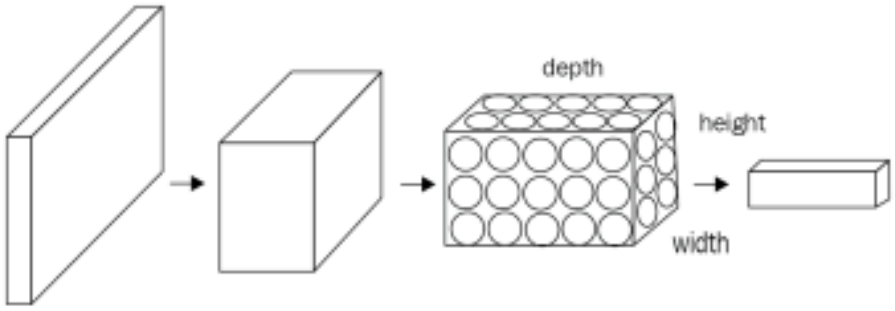
Figura 2: Rede neural regular de três camadas



Fonte: (SEWAK MD. REZAUL KARIM, 2018)

Em uma rede convolucional as camadas tem os neurônios dispostos em três dimensões: largura, altura e profundidade, dessa forma os neurônios de uma determinada camada estarão conectados apenas a uma pequena região da camada anterior, isso torna o processo de aprendizagem mais eficiente além de permitir a redução dos parâmetros da rede. (SEWAK MD. REZAUL KARIM, 2018)

Figura 3: Uma rede neural convolucional com os neurônios dispostos em três camadas



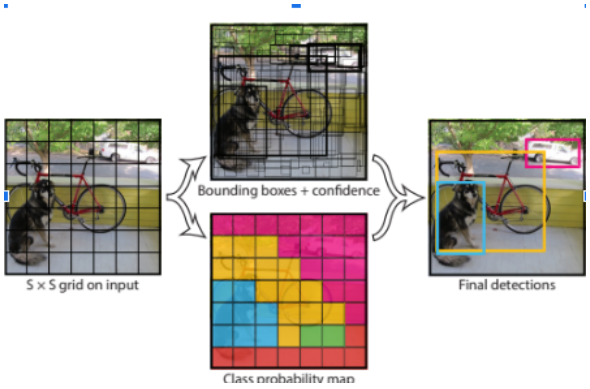
Fonte: (SEWAK MD. REZAUL KARIM, 2018)

2.1.3 YOLOv8

YOLO é uma rede neural convolucional que trata problemas de classificação de forma regressiva, possibilitando a classificação de vários objetos em uma imagem e as coordenadas das caixas delimitadora de cada objeto (REDMON; FARHADI, 2018). O YOLO é capaz de detectar um objeto com uma única rede neural que faz a detecção e classificação na imagem uma única vez, diferente de outras redes neurais que aplicam o modelo em uma imagem em diferentes localizações e escalas da mesma, isso torna o YOLO uma excelente solução para problemas que necessitam ser resolvidos o mais próximo do tempo real (REDMON; FARHADI, 2018).

Existem diversas camada na arquitetura do modelo YOLO. A primeira camada é a de convolução, onde é aplicado filtros afim de extrair características como bordas, texturas, padrões e partes do objeto. A segunda camada é a de pooling, responsável por reduzir as dimensionalidades das características que foram extraídas na camada de convolução, porém preservando as mais importantes, com isso ganha-se tempo de processamento. A última camada é a de detecção, nessa camada a imagem a ser processada é divida em um grid de tamanho S X S, cada célula do grid é responsável por prever um número B de caixas delimitadoras bem como as probabilidades associada a cada classe, a partir disso é aplicado um algoritimo de *NMS*(Non Maximum Supression) com o objetivo de reduzir a quantidade de caixas delimitadoras sobrepostas e mantendo as caixas com mais confiança, esse passo pode ser observado na figura 4 (REDMON; FARHADI, 2018).

Figura 4: Camada de detecção YOLO



Fonte: (REDMON; FARHADI, 2018)

2.2 OpenCV

O OpenCV é uma biblioteca de código aberto para trabalhar com visão computacional, a biblioteca possui uma curva de aprendizado baixa, permitindo a construção de aplicações que permite o computador ver e tomar decisões com base nisso, o OpenCV é escrito em C/C++ e roda em sistema operacionais Linux, Windows e Mac OS, embora a biblioteca seja escrita em C/C++ é possível encontrar interfaces da mesma em diversas outras linguagens como Python, o fato da aplicação ser construída em C/C++ permite que a aplicação a ser construída faço uso de processamento concorrente, isso garante a construção de uma aplicação eficiente para se trabalhar com processamento de imagens (BRADSKI, 2008).

2.3 Spring

O Spring é um framework que pode ser utilizada para a implementação de sistemas baseado em java, como aplicações web ou desktop, a ferramenta fornece recurso para integração dom diversos serviços como banco de dados, mensageria, ftp entre outros, a forma como o spring foi construído permite ao desenvolvedor obter os benefícios ofertado pelo ecossistema gerando o mínimo impacto na aplicação construída (COSMINA, 2017). Dada toda a facilidade do spring para integração com outros sistemas bem como todos os recursos oferecidos pelo framework, a ferramenta é excelente para a construção de uma aplicação web.

2.4 Banco de Dados

O banco de dados é um software composto por duas estruturas, um sistema de armazenamento e um gerenciador, o sistema de armazenamento é responsável por guardar

os dados e suas descrições, já o gerenciador é um sistema que contém uma linguagem que possibilita ao usuário realizar consultas e alterar dados, além disso o gerenciador também é responsável pela a autenticação e autorização no banco (MEIER, 2019). Hoje em dia os bancos são divididos em dois tipos principais, bancos relacionados que utilizam como linguagem o SQL e bancos não-relacionais também conhecidos como NoSQL (MEIER, 2019).

No banco de dados relacional as informações são representadas como um conjunto de tuplas em forma tabular, um tabela permite que os dados sejam visualizados de forma fácil, uma vez que a mesma representa uma entidade e as colunas as características daquela entidade, os dados podem ser inseridos linha a linha realizado o mapeamento entre as colunas e os valores que devem assumir (MEIER, 2019). O MySQL e o PostgreSQL são os banco de dados de código aberto mais utilizados atualmente (DB-ENGINES, 2021).

O banco de dados não-relacional é um sistema altamente flexível, a ideia por trás desse tipo de banco é substituir as tabelas dos bancos relacionais por uma estrutura mais simples, conhecida como documento, esse tipo de estrutura permite a representação de estrutura hierárquicas complexas através de uma única escrita no banco de dados, ao contrário do banco de dados relacionados o documento não é pré-definido, dando ao desenvolvedor maior flexibilidade de como representar os dados da aplicação (CHODOROW, 2019). Atualmente o banco de dados não-relacional mais utilizado é o MongoDB (DB-ENGINES, 2021).

2.4.1 PostgresSQL

PostgreSQL é um popular sistema de gerenciamento de banco de dados, o projeto é completamente mantido por voluntários, não existindo nenhuma corporação por trás responsável por gerenciar o desenvolvimento do mesmo. o PostgreSQL é considerado uma base de dados robusta e bastante performática, além de oferecer diversos recursos como a escrita de funções em diversas linguagens como javascript e customização na criação de atributos. (HSU, 2017)

2.4.2 Elasticsearch

Elasticsearch é um banco de dados NoSQL open-source que disponibiliza uma API Restful para consultas, a utilização da ferramenta é recomendada quando se deseja trabalhar com análise de dados, isso porque a mesma possui uma indexação dos dados quase que em tempo real e permite realizar buscas performáticas, isso deve-se ao fato da ferramente utilizar uma estrutura de dados de índices invertidos do Apache Lucene, que consiste em uma lista de palavras que aparece em cada documento, e cada palavra dessa

lista estão associadas aos documentos em que aparecem. Dessa forma cada campo do documento está indexado, permitindo realizar buscas em larga escala.

2.5 Aplicação WEB

Uma aplicação WEB é um programa computacional que permite que uma pessoa ou uma outra aplicação utilize recursos da máquina em que a aplicação está hospedado para realização de uma tarefa através da internet, essas tarefas podem consistir em ações como armazenar ou buscar alguma informação em um banco de dados, encaminhar uma requisição para um outro serviço e/ou processamento de dados, além disso uma aplicação web também pode ser utilizada para fornecer uma interface gráfica que permite ao usuário encontrar visualizações rápidas dos principais indicadores e dados relevantes sobre um processo (NOBACK, 2020). A aplicação web possui diversas vantagens por estar exposta na internet, podendo ser acessada de qualquer lugar através de um web browser, ademais por ser uma aplicação centralizada todos os usuários acessam uma mesma instância que estará sempre atualizada, o usuário pode escolher qual tipo de dispositivo irá utilizar para consumir os recursos da aplicação podendo ser um computador desktop ou até mesmo um smartphone.

2.6 Estimar distância entre objetos em uma imagem

Uma imagem é composta por pixels, que podem ser representados pela função $f(x, y)$, onde x e y representam as coordenadas espaciais de um pixel na imagem. Uma métrica bastante utilizada para calcular a distância entre dois pontos é a distância euclidiana, essa medida é bastante utilizada em diversos campos, sendo um deles a análise de imagens (GONZALEZ, 2002). As principais escolhas para a utilização dessa métrica é a sua simplicidade para implementação e eficiência, uma vez que o cálculo é relativamente simples permitindo ser aplicado em um grande conjunto de dados, como por exemplo, um stream de video.

$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

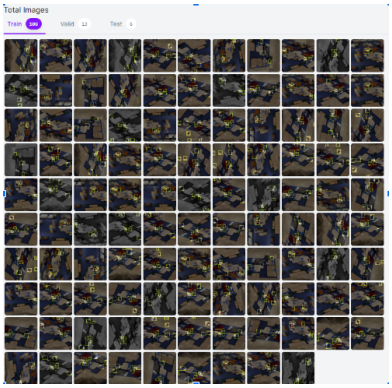
A formula acima é utilizada para calcular a distância euclidiana para dois pontos, através das coordenadas das caixas delimitadoras será possível calcular o centro de cada uma das caixas e utilizar os valores para calcular a distância entre as caixas delimitadoras.

2.7 Roboflow

O Roboflow é uma plataforma que tem como finalidade facilitar o treinamento e criação de modelos de visão computacional, a mesma oferece recursos para a preparação

de dados, treinamento de modelos e implementação. Um dos principais diferencias do roboflow é a facilidade que a ferramente oferece em realizar algumas tarefas repetitivas que são necessário para o treinamento de um modelo, como a aplicação de técnicas que visam aumentar a quantidade de dados disponíveis modificando a imagem original realizando rotação, mudanças nas escalas de cores, zoom e alterando o tamanho das mesmas (B; J, 2022).

Figura 5: Utilização do Roboflow para adicionar rótulos nas imagens



Fonte: Autoria Própria

Na figura 5 o roboflow é utilizado para adicionar rótulos de pessoas nas imagens, o roboflow posteriormente irá gerar um arquivo contendo as coordenadas de cada uma das caixas delimitadoras que foram adicionadas, essas informações são utilizadas no processamento de treinamento e testes do modelo.

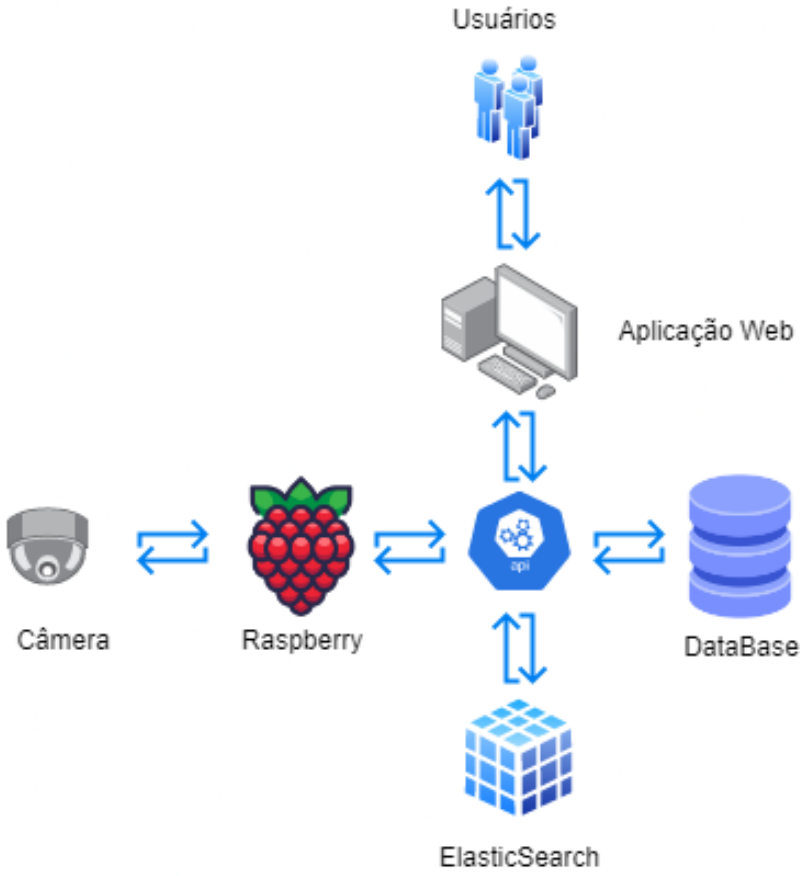
3 METODOLOGIA

Esta secção se concentra nos métodos empregados para atingir o objetivo da construção do sistema para monitoramento de distanciamento social utilizando câmeras inteligentes. O objetivo é entender todas as etapas envolvidas no decorrer do projeto.

O projeto pode ser divido em duas partes, a primeira é a câmera inteligente, responsável por capturar a imagem, processar e enviar as informações para uma aplicação web, que representa a segunda parte do projeto, responsável por receber as informações, realizar a persistência e fornecer um meio do usuário poder visualizar as dados que foram enviados.

3.1 Arquitetura do Projeto

Figura 6: Arquitetura do projeto



Fonte: Autoria Própria

A figura 6 exibe a integração entre todos os componentes da aplicação. Podemos observar que há uma comunicação entre a câmera e a raspberry, nesse conjunto, que

consiste basicamente da primeira etapa do sistema, é realizado a captura da imagem bem como o processamento da mesma, como por exemplo, a detecção de pessoas e o cálculo de distanciamento entre as mesmas, com essas informações a raspberry pode enviar os dados coletadas para uma API.

A API é uma aplicação web construída em java utilizando o framework Spring Boot, essa aplicação disponibiliza diversos endpoints responsáveis pela persistência dos dados recebidos pela raspberry e também pela persistência das câmeras inteligente, onde o usuário pode disponibilizar variáveis de ambientes e tags para as câmeras.

Para a persistência de dados é utilizado uma banco de dados SQL e uma instância elasticSearch, no banco de dados SQL são persistida as informações referente as câmeras, enquanto no elasticSearch é armazenado os logs recebidos através do dispositivo raspberry.

A aplicação web é a responsável por fornecer uma visualização ao usuário das câmeras disponíveis e também das informações enviadas pela mesma, para isso a aplicação utilizada dos endpoints disponibilizados pela API.

3.1.1 Câmera

Devido aos custos de uma câmera foi optador por realizar uma simulação da captura das imagens, sendo utilizado vídeos disponíveis na internet e também uma gravação de autoria própria em um shopping, a gravação foi feita com a câmera traseira de um Iphone 11.

3.1.2 Raspberry Pi

A Raspberry Pi é um computador de baixo custo desenvolvido no Reino Unido pela Fundação Raspberry Pi em colaboração com a Broadcom, o computador inclui CPU, GPU, memória ram e diversos outros componentes como Wi-Fi e conexões USB, recursos que podem ser utilizadas para conexão com hardware externos, como uma câmera e aplicações webs (FOUNDATION, 2023). A flexibilidade e funcionalidade da raspberry pi foram um dos principais motivos para a utilização da mesma no desenvolvimento do projeto, uma vez que é capaz de realizar muitas das funções que um desktop comum.

Nesse projeto foi utilizado o modelo *Raspberry Pi 4 Model B* que é possível observar na Figura 7. O modelo utilizado possui um processador *Quad Core 1.2GHz 64Bit* e *8GB RAM*. Na raspberry é executado um código responsável por detectar as pessoas em um frame e calcular a distância entre as mesmas, em seguida essas informações são enviadas através da internet para uma API.

Figura 7: Raspberry Pi 4 Model B

Fonte: Autoria Própria

É importante salientar que poderiam ser utilizados outros computadores para executar o código responsável pela detecção e classificação dos objetos na imagem, como Jetson Nano, que é uma placa especificamente projetada para trabalhos que envolvem inteligência artificial, uma vez que vem integrada com uma placa de vídeo NVIDIA, no entanto essas placas chegam a custar o dobro de uma placa raspberry.

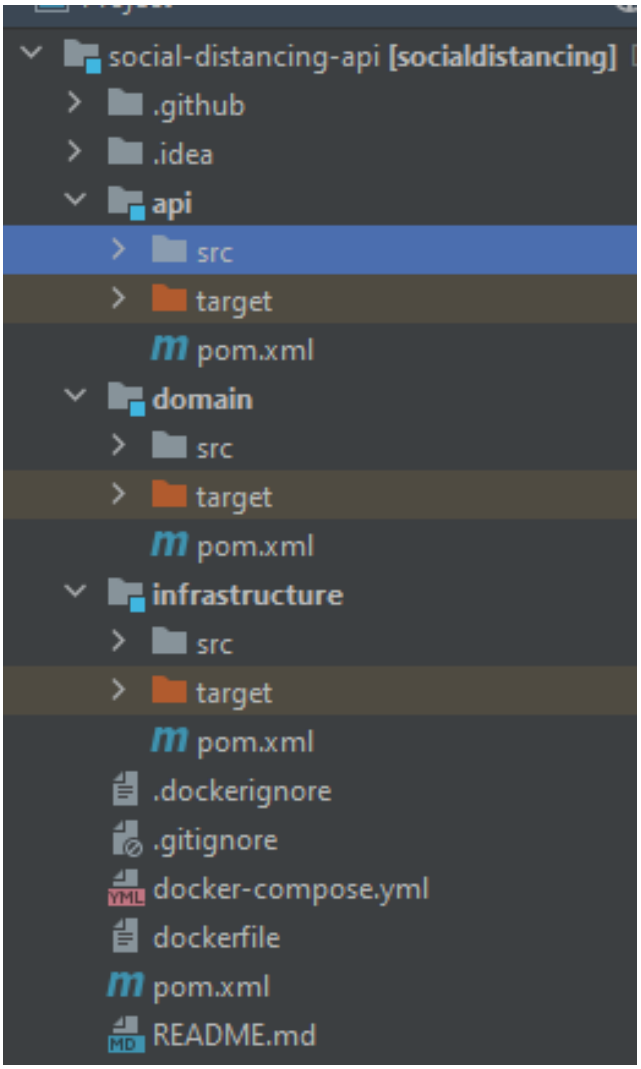
3.1.3 API

A API implementada serve como ponte entre a raspberry pi e a interface de usuário, permitindo a troca de informacões entre eles. Os principais endpoints da API incluem recursos que permitem a receber os dados da raspberry pi e armazenar os mesmos, além de oferecer meios para que a interface do usuário consulte esses dados e exiba ao usuário.

Para a implementação da API foi utilizada a linguagem de programação java, devido a sua escalabilidade, robustez e a grande comunidade de desenvolvedores. Também foi utilizado o framework Spring Boot que oferece diversos pacotes que facilitam a integração

com banco de dados e outros softwares além de auxiliarem no processo de desenvolvimento e reduzir a necessidade de manutenção.

Figura 8: Estrutura da API



Fonte: Autoria Própria

Conforme a imagem acima, podemos observar que a aplicação da API foi estruturada em 3 módulos principais: *api*, *domain* e *infrastructure*. essa abordagem foi adotada pois é uma maneira eficiente de separar responsabilidades e diminuir acoplamentos entre as diferentes partes do sistema, trazendo benefícios para o desenvolvimento e também para a manutenção do sistema. Essa estrutura promove escalabilidade, testabilidade além de contribuir com a facilidade de adição de novas funcionalidades ao produto promovendo também fácil reutilização dos componentes do sistema.

Na camada de *api* está localizada toda a lógica relacionada a interface web, como rotas, configurações de internacionalização, tratamento de exceções, documentação online

da interface e validações de dados. Essa separação clara facilita a manutenção e permite que futuras atualizações ou expansões na API sejam feitas com maior agilidade e precisão.

Em seguida o módulo *domain*, onde todas as regras e entidades estão encapsuladas. Essa separação traz benefícios significativos, pois garante que a lógica de negócio permaneça independente da infraestrutura e de outros componentes externos. Como resultado, como resultado a regra de negócio fica coeso e reutilizável, facilitando a realização de testes unitários e possibilitando a evolução do sistema de forma mais ágil. Ao isolar as preocupações de negócio nesse módulo, o desenvolvimento torna-se mais focado. Isso também permite seja realizada alterações nas demais parte do código sem afetar a lógica de negócio, oferecendo maior flexibilidade ao sistema.

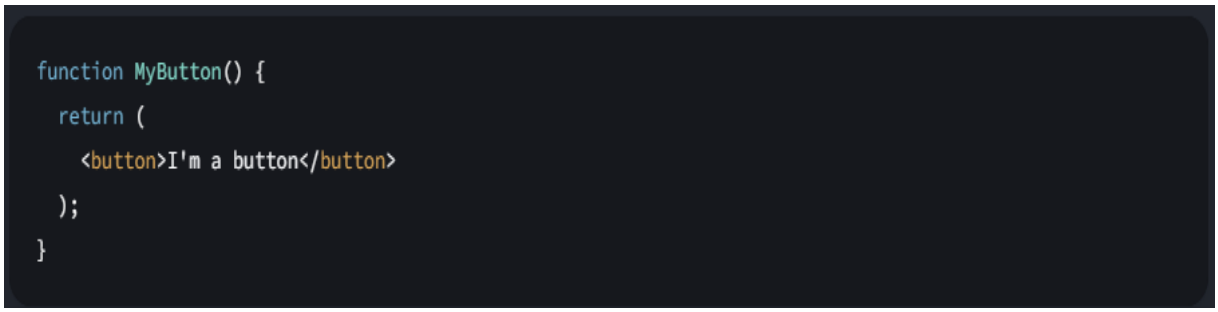
Por fim, o módulo de *infrastructure* trata das preocupações relacionadas à persistência de dados, integrações externas como chamadas de APIs e qualquer outra dependência de infraestrutura. Separar essas responsabilidades em um módulo dedicado permite flexibilidade para trocar ou atualizar componentes de infraestrutura sem afetar diretamente a lógica do domínio ou a interface web. Além disso, permite ser utilizada diferentes fontes para armazenamento e leitura de dados, como é feito na aplicação, onde se tem um sistema de armazenamento em SQL e o outro NoSQL, sem comprometer a estabilidade e a consistência do sistema.

3.1.4 Aplicação web

Uma das ferramentas utilizadas para o desenvolvimento da aplicação web foi o ReactJS, um biblioteca bastante utilizado para a construção de aplicações iterativas e reativas. A ferramenta foi criada pelo facebook afim de oferecer uma maneira de desenvolvimento moderna e eficiente para a construção de aplicações SPA (Single Page Application).

Componentes reutilizáveis é uma das características mais importantes dessa biblioteca, uma vez que permitem aos desenvolvedores dividirem a interface do usuários em diversos componentes independentes reutilizáveis, cada um contento a sua própria lógica e renderização. Os componentes podem ser compostos formando outros mais complexos, isso simplifica o desenvolvimento e manuntenção do código.

Figura 9: Sintaxe JSX



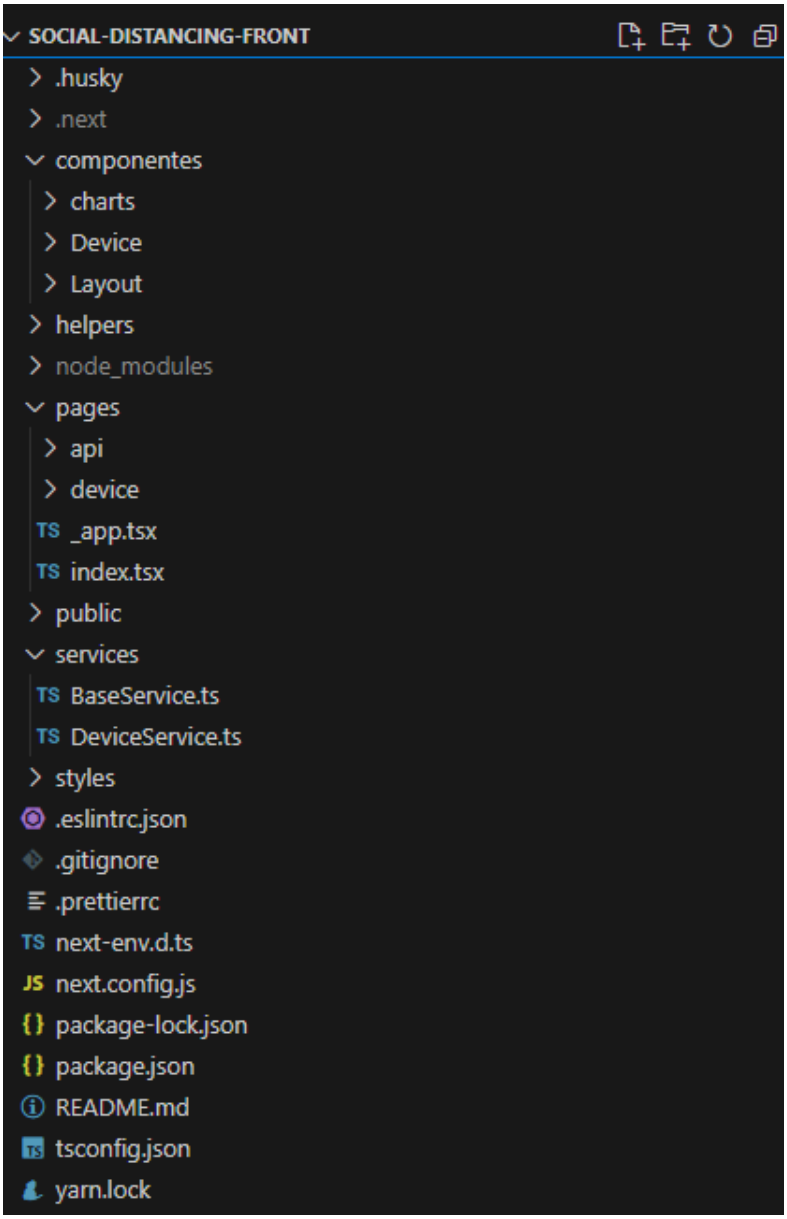
Fonte: Autoria Própria

Na figura acima podemos observar a sintaxe utilizada na construção da aplicação. O JSX é uma extensão do javascript que permite que os programadores escrevam código HTML-like em arquivos javascript (FOUNDATION, 2023), Dessa maneira os desenvolvedores podem escrever o código de maneira declarativa.

A escolha de utilizar o ReactJS para o desenvolvimento da aplicação web traz benefícios significativos. O ReactJS oferece um desempenho excelente, pois utiliza um conceito conhecido como virtual DOM (Document Object Model) que otimiza as atualizações da interface, garantindo uma renderização rápida e eficiente.

Um dos principais desafios desse projeto é sobre como manter o usuário o mais atualizado possível sobre a quantidade de pessoas localizadas pela câmera inteligente, portanto ter serviços com a menor latência possível é essencial, tendo isso em mente, foi utilizado também o NextJS em conjunto com o ReactJS. O NextJS é um framework para o ReactJS que oferece uma série de recursos avançados, sendo uma delas o SSR (Server-Side Rendering), isso significa que as páginas são pré-renderizadas dentro do servidor antes de serem enviadas para o navegador do usuário como HTML, invés de aguardar o carregamento e redenrização do JavaScript, como consequência o usuário irá experimentar tempos de carregamentos mais rápidos e responsivos além de uma experiência mais fluida, especialmente em dispositivos que utilizam de conexões lentas ou instáveis a internet.

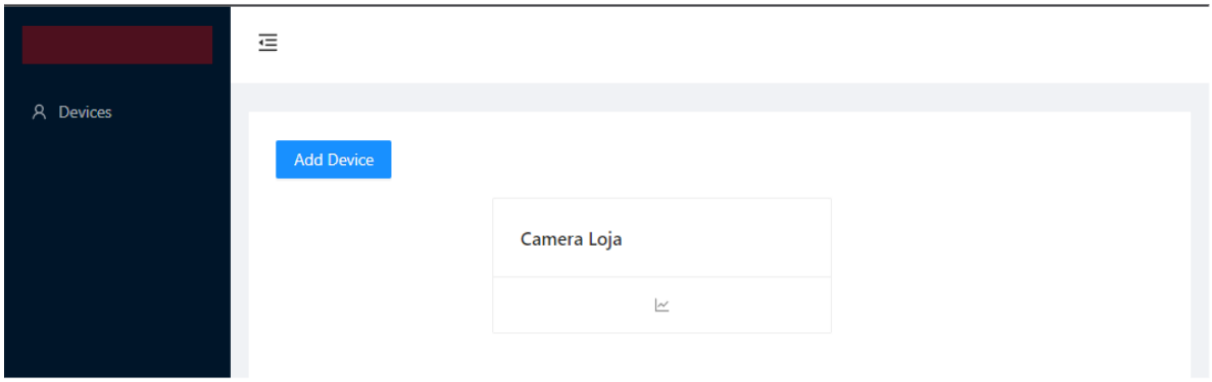
Figura 10: Estrutura da Aplicação Web



Fonte: Autoria Própria

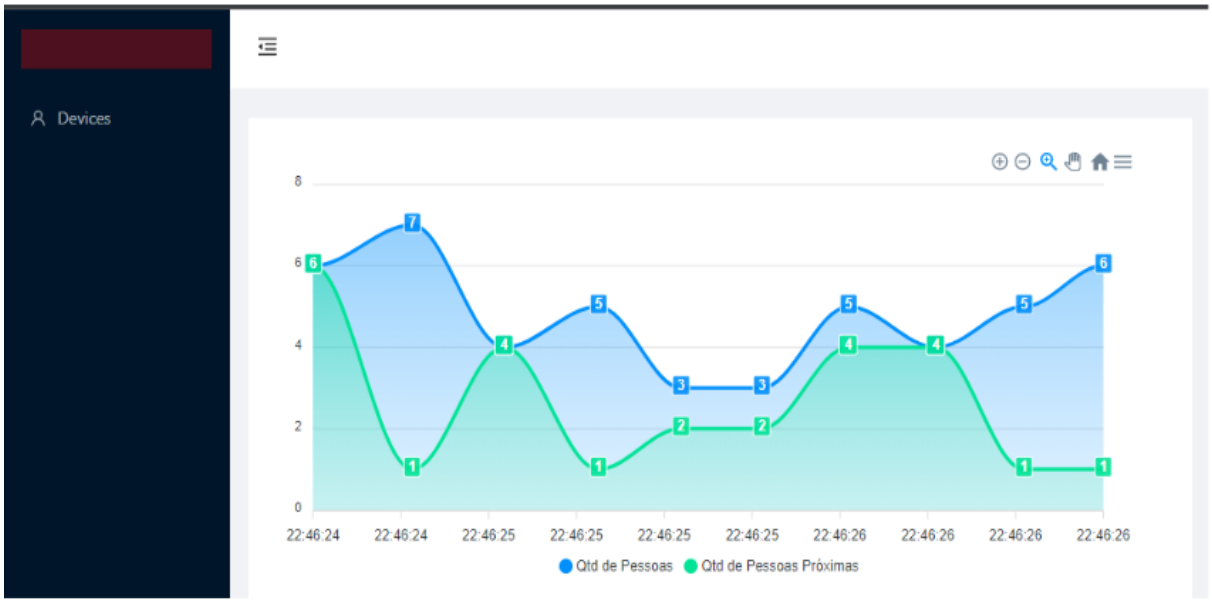
Na figura acima é possível observar a estrutura do projeto front-end. Na pasta *components* é possível encontrar os componentes utilizados para montar cada um dos objetos que podem ser encontrados na interface, por exemplo, dentro da pasta *charts* é possível encontrar os componentes relacionados ao gráfico que mostra a quantidade de pessoas ao longo do tempo, na pasta *Device* estará os componentes responsáveis por renderizar as listas de dispositivos que estão cadastrados no sistema, enquanto na pasta *layout* está o componente que define a estrutura da página. Abaixo podemos observar o resultado de cada componente.

Figura 11: Aplicação web: Lista de Dispositivos



Fonte: Autoria Própria

Figura 12: Aplicação web: Gráfico



Fonte: Autoria Própria

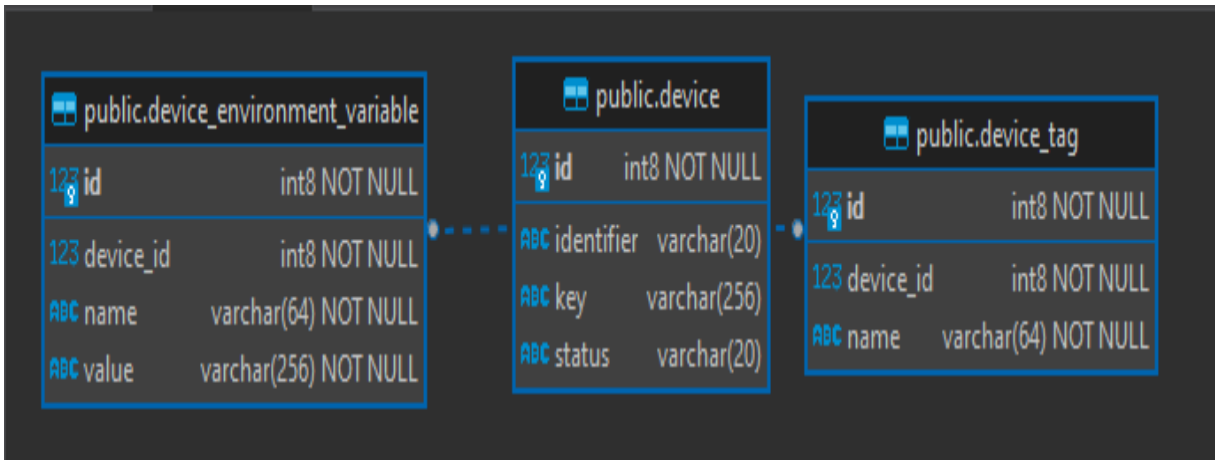
Dentro da pasta *pages* é possível encontrar as rotas da aplicação, cada arquivos e pasta está associado a uma rota, por exemplo, um arquivo index.tsx na raiz da pasta pages está associado a rota "/"da aplicação, já um arquivo localizado dentro da pasta *./pages/-device* está associado a rota "/device/". Essa abordagem baseada em arquivos facilita o roteamento e a navegação no aplicativo, uma vez que esse processo fica automatizado pelo próprio framework.

Na pasta *services* está centralizada toda a lógica responsável pela comunicação com serviços externos, esses arquivos possuem a responsabilidade de realizar chamadas e manipular os dados provenientes dessas APIs, por enquanto no projeto são realizadas chamadas apenas para a API construída com o Spring, são consultados *endpoints* que retornam os dispositivos cadastrados bem como as informações enviadas por eles.

3.1.5 Modelagem do banco de dados SQL

O banco de dados utilizado é o *PostgreSQL*, o banco SQL é utilizado nesse projeto para o armazenamento de informações que possuem baixa quantidade de acessos e que estão altamente relacionadas. O banco de dados de dados é composto por 4 entidades, sendo elas respectivamente: *device*, *device_tag* e *device_environment_variable*.

Figura 13: Modelo do banco de dados



Fonte: Autoria Própria

Na tabela *device* é armazenada informações referentes ao dispositivo físico, a coluna *id* é um identificador auto-gerado, na coluna *identifier* pode ser armazenado um identificador único para o dispositivo, como por exemplo um endereço MAC, na coluna *key* pode ser amazenada uma chave que será utilizada para criptografar a comunicação com o dispositivo caso esteja presente, na coluna *status* é armazenada o estado do dispositivo informado se o mesmo está online, offline ou bloqueado.

A tabela *device_tag* pode ser armazenada tags aos dispositivos, por exemplo, se um dispositivo está em um quarto, ele poderia ter uma tag chamada quarto, o dispositivo pode ter 0 ou mais tags, a coluna *name* armazena o nome da tag.

Por último a tabela *device_environment_variable* responsável por armazenar variáveis de ambiente para o dispositivo, nessa tabela pode ser armazena qualquer informação que

possa ser utilizada posteriormente a respeito daquele dispositivo, um exemplo pode ser uma variável frame_rate, que pode conter um valor representando a taxa de atualizações de métricas para aquele dispositivo, no cenário desse projeto, a taxa de obtenção de novas informações a partir da câmera para processamento, a coluna *name* deve conter o nome da váriavel e a coluna o valor da mesma.

3.1.6 Modelagem ElasticSearch

Para armazenar o resultado do processamento foi o escolhido o *elasticsearch* devido a capacidade da ferramenta de inserir e recuperar dados em tempo real rapidamente, sendo portanto muito utilizado para projetos em que é necessário realizar análise de logs.

Figura 14: Modelo ElastichSearch

{...} _id	# deviceId	# timestamp	payload.number_of...	payload.number_of...
01 {...} 9U320lgBXpgCSy48XGPk	1	1687132986592	10	15
02 {...} 8k320lgBXpgCSy48W2O5	1	1687132986287	11	16
03 {...} 8U320lgBXpgCSy48WmOc	1	1687132986006	8	14
04 {...} 8E320lgBXpgCSy48WWNv	1	1687132985706	16	16
05 {...} 7k320lgBXpgCSy48V2MS	1	1687132985099	14	15

Fonte: Autoria Própria

Na imagem acima temos o modelo de algumas informações enviadas para o *elasticsearch*, a ferramenta foi modelada para que pudesse ser enviada qualquer informação no campo de payload, devendo portanto a aplicação que for consumir essas informações realizar os filtros necessários, no exemplo acima é enviado a quantidade de objetos e quantidade de objetos próximo um ao outro.

3.1.7 Rotas da API

Em uma API (Application Programming Interface), uma "rota"refere-se ao caminho ou URL específico que os clientes podem usar para acessar recursos fornecidos pela API. As rotas são uma parte fundamental do design de uma API, uma vez que definem como os clientes podem interagir com o sistema através de solicitações HTTP.

Os "métodos HTTP"são partes da solicitação HTTP que especifica a ação que o

Tabela 1: Recursos disponíveis na API

Tipo de Método	Rota	Descrição da Operação
GET	/device	Recupera todos os dispositivos
POST	/device	Cria um novo dispositivo
POST	/device/{deviceId}/tag	Cria uma nova tag para um dispositivo específico
GET	/device/{deviceId}/log	Obtém os logs de um dispositivo específico
POST	/device/{deviceId}/log	Adiciona um log a um dispositivo específico
POST	/device/{deviceId}/env	Cria um novo ambiente para um dispositivo específico
DELETE	/device/{deviceId}/tag/{tagName}	Exclui uma tag de um dispositivo específico
DELETE	/device/{deviceId}/env/{envName}	Exclui um ambiente de um dispositivo específico

Fonte: Autoria Própria

cliente deseja executar em um recurso identificado na rota. Os métodos HTTP utlizados no projeto são:

A Tabela 1 apresenta os recursos que podem ser acessados através da API, bem como o método que deve ser utilizado e uma descrição da operação a ser realizada.

3.1.8 Detectando Pessoas

Afim de facilitar o processo, foi utilizado o *ultralytics*, um framework criado para atender tarefas que envolvem visão computacional, o mesmo fornece abstrações para treinar, testar e implementar modelos *YOLO*. (JOCHER et al., 2023).

A implementação é realizada em linguagem de programação python, o modelo é treinado no *ImageNet* dataset, que contém mais de 14 milhões de imagens e é extremamente utilizado para treinar e testar modelos de *deep learning*, o modelo é capaz de detectar e classificar até 80 tipos de objetos (RUSSAKOVSKY et al., 2015). Para esse trabalho o único objeto de interesse é do tipo pessoa .

3.1.9 Mensurando a distância

No pseudo código abaixo é possível visualizar o algorítimo responsável por mensurar a distância entre as pessoas, para mensurar a distância é utilizado um *threshold* em pixels, ou seja, a depender da distância em pixels entre as caixas delimitadoras é definido se está próximo ou não do valor limite definido, utilizando a escala definida e as coordenadas das caixas delimitadoras é calculado a distância a partir do centro das coordenadas das mesmas.

Listing 1: Pseudocódigo para calcular distância entre as pessoas

```
PARA cada pessoa detectada
    Desenhe um retangulo ao redor da pessoa
    Calcule o centro da caixa delimitadora da pessoa atual

    PARA cada pessoa restante detectada
        Calcule o centro da caixa delimitadora da outra

        Calcule a distancia euclidiana entre os centros

        SE a distancia for menor que um valor de limite
            Incremente a contagem de pessoas proximas
            Desenhe uma linha conectando as duas pessoas
```

Como não foi possível criar um ambiente controlado para aferir a eficiência do algorítimo, foram realizado testes apenas através de imagens, observando se o algorítimo apontando corretamente quando pessoas estavam próximas demais uma das outras, nesse caso, lado a lado.

Figura 15: Algorítimo contando 3 pessoas lado a lado

Fonte: Autória Própria

Na imagem acima é possível visualizar que o algorítimo foi capaz de mensurar que há 3 pessoas muito próximas uma das outras, desenhando uma caixa delimitadora ao redor das mesmas.

Em algumas situações de maneira errônea o algorítimo apontou que haveria mais pessoas próximas uma das outras, isso deve-se ao fato da quantidade de caixas delimitadoras sugeridas pelo modelo *YOLO*, existem situações onde o modelo sugeriu mais de uma caixa por pessoa, resultando no erro que pode ser observado na imagem abaixo.

Figura 16: Situação onde erroneamente o algorítimo aponta 5 pessoas próximas

Fonte: Autoria Própria

Foram realizados testes de maneira manual foram apontadas a quantidade de pessoas próximas uma das outras e comparadas com a quantidade de pessoas próximas pelo algorítimo, foi utilizada a métrica de erro percentual absoluto médio, que resultou em 18.45% de erro. Essa métrica foi escolhida por ser tolerante a zero, onde se a diferença entre o valor esperado e o predito for de 0, indica um erro de 100%, além de fornecer o resultando em porcentagem, facilitando a compreensão e comunicação, na tabela abaixo estão os dados comparados.

Tabela 2: Tabela com a quantidade de pessoas próximas vs quantidade de pessoas próximas apontada com algorítimo da distância euclidiana

Quantidade de pessoas próximas	Quantidade previstas
2	2
2	2
2	2
2	2
2	2
2	2
3	4
3	4
3	4
3	4
3	4
3	4
3	4
4	4
3	4
4	4
3	3
6	6
4	6
4	6
4	4
3	3
4	6
4	4
4	6
4	4
4	6
4	4

Utilizando o algorítimo de distância euclidiana é levado em consideração apenas os eixos X e Y, isso significa que se duas pessoas estiverem próximas em X e Y mas distantes em Z, o algorítimo não seria capaz de identificar. Uma maneira de poder contornar esse problema seria utilizando mais de uma câmera em posições diferentes, a diferença entre as imagens seria utilizada para calcular a profundidade (HARTLEY, 2003).

4 RESULTADOS E DISCUSSÕES

Neste capitulo serão apresentados os resultados obtidos através da construção da aplicação utilizando a metodologia proposta.

4.1 Página Inicial

Conforme mencionado anteriormente, a interface web foi construída utilizando ReactJS em conjunto com o NextJS, visando reutilização de componentes e a facilidade de construir aplicações com esses ferramentas.

Ao acessar a página inicial é possível visualizar um botão para cadastrar dispositivos, nessa mesma tela também é possível observar os dispositivos cadastrados.

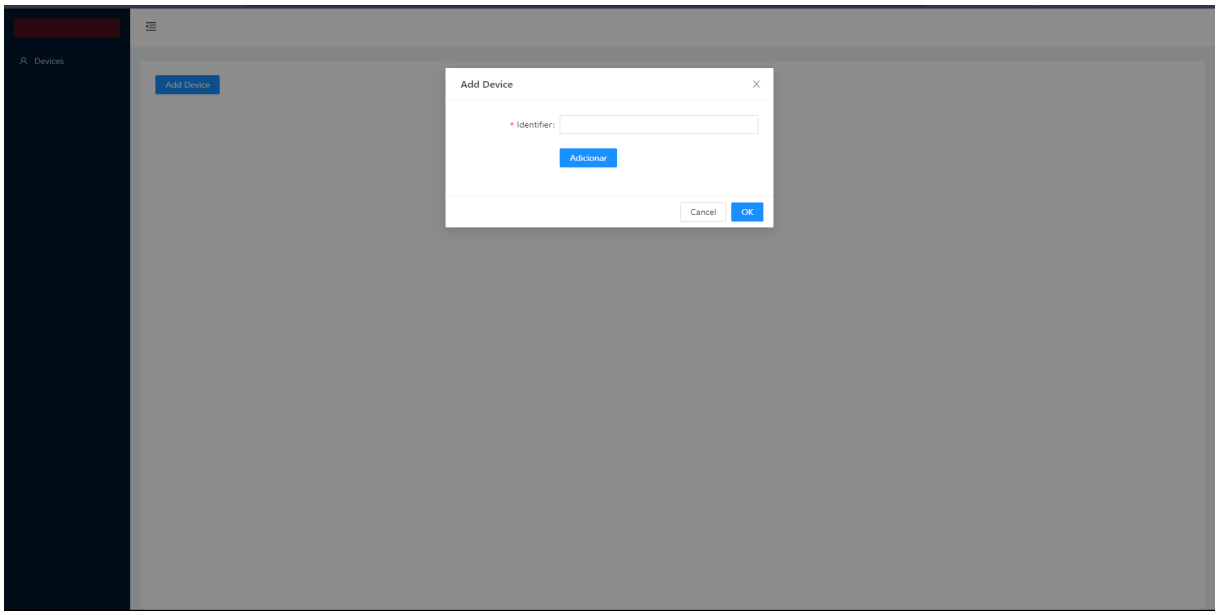
Figura 17: Página Inicial

Fonte: Autoria Própria

4.2 Adicionar Dispositivo

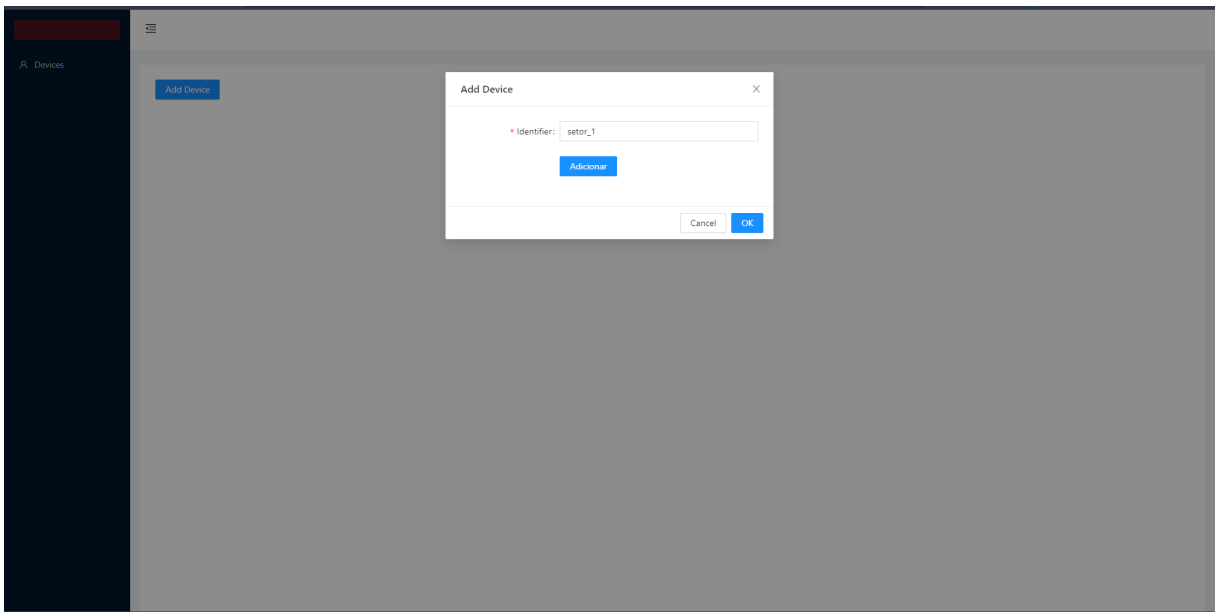
Na página inicial é exibido um botão que permite ao usuário cadastrar um novo dispositivo fornecendo apenas uma informação que permita ao usuário identificar esse dispositivo posteriormente, esse identificador pode ser por exemplo a localização do dispositivo, ao finalizar é enviada uma requisição para a API que persistirá as informações fornecidas no banco de dados.

Figura 18: Página de Cadastro de Dispositivo



Fonte: Autoria Própria

Figura 19: Formulário para cadastro de dispositivo



Fonte: Autoria Própria

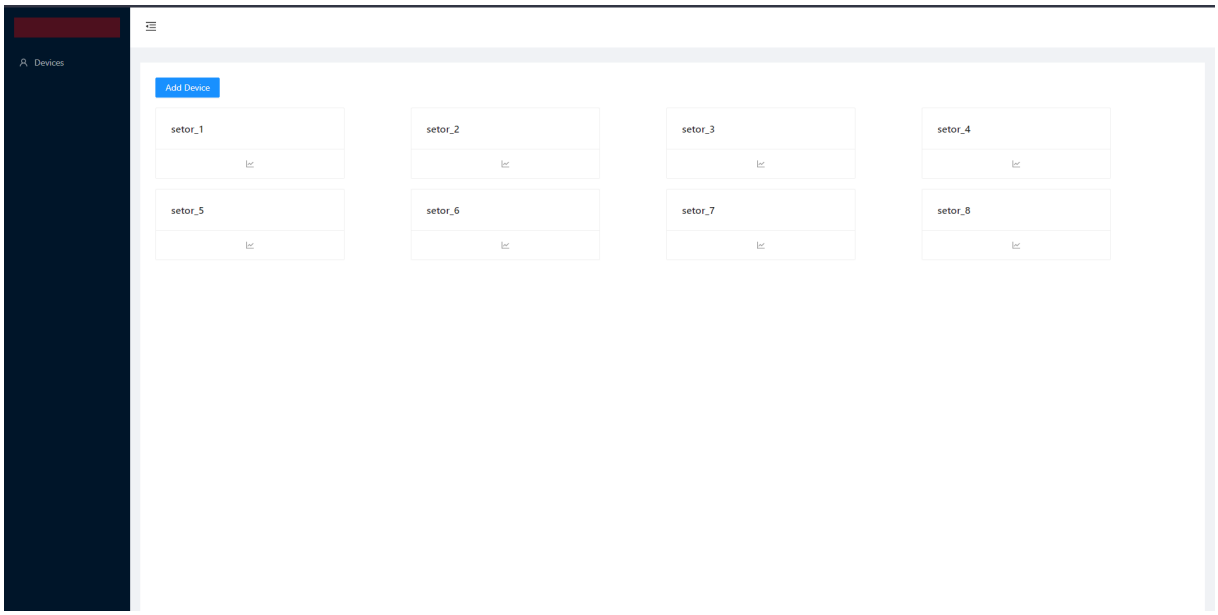
4.3 Visualizar Dispositivos

Quando um dispositivo estiver cadastrado, será possível visualizar o mesmo através da página inicial, cada dispositivo tem uma caixa delimitadora e um botão onde será possível visualizar as informações enviadas pelo mesmo.

Figura 20: Página Inicial com Dispositivo

Fonte: Autoria Própria

Figura 21: Página Inicial com Dispositivos



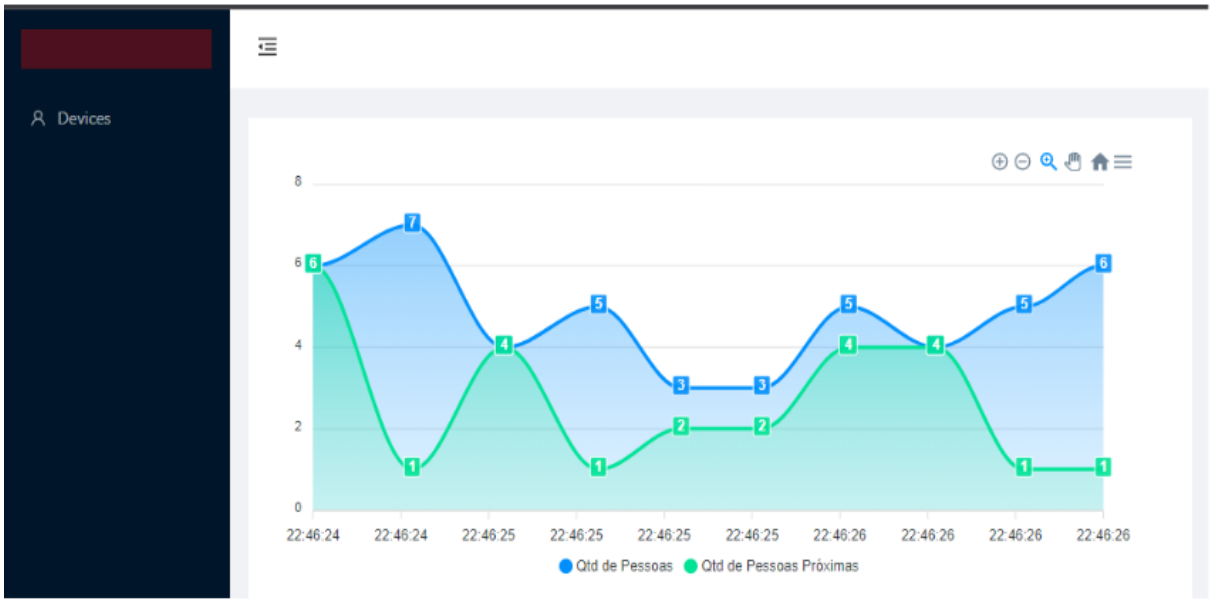
Fonte: Autoria Própria

4.4 Visualizar Informações Enviadas

Ao clicar dentro do botão inserido dentro da caixa delimitadora do dispositivo, será possível visualizar um gráfico contendo as informações enviadas por este dispositivo atualizadas em tempo real.

Na página contém um gráfico onde no eixo horizontal tem-se o tempo e no eixo vertical a quantidade de pessoas, no gráfico é possível visualizar duas informações, a informação em azul é a quantidade de pessoas que foram detectadas e a informação em verde exibe a quantidade de pessoas próximas uma da outra no mesmo momento.

Figura 22: Página com Informações Enviadas



Fonte: Autoria Própria

4.5 Treinamento do Modelo

Para o treinamento do modelo foi utilizado o framework *yolov8*, esse modelo possui 80 classes pré-treinadas, no entanto a única classe de interesse para este projeto é classe *person*, o framework fornece disponibiliza 5 modelos pré-treinados que podem ser visualizados na imagem abaixo.

Figura 23: Modelos YOLOV8

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Fonte: (JOCHER et al., 2023)

Na figura 23 podemos observar através da métrica de mAP(Mean Average Precision) e das métricas de Speed que o modelo *YOLOv8n* possui uma acurácia menor porém

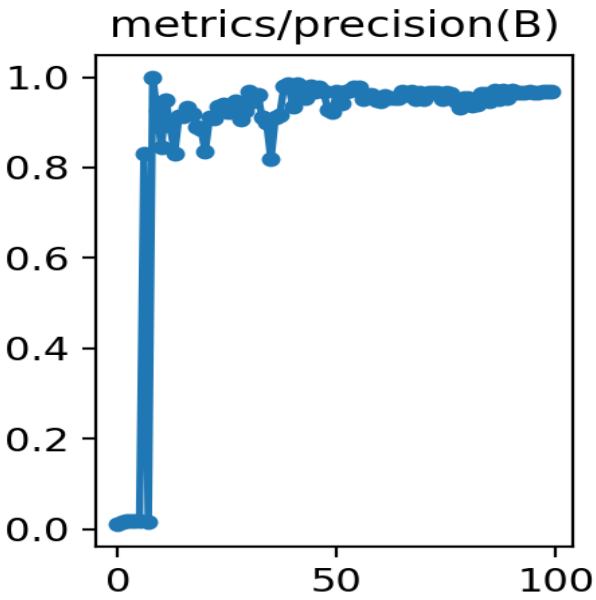
é capaz de detectar e classificar objetos mais rápido que o modelo *YOLOv8x* considerando que ambos os modelos sejam executados no mesmo hardware, como este trabalho irá ser executado em uma *raspberry* foi escolhido o modelo pré-treinado *YOLOv8n*.

As imagens para treinamento foram obtidas através de uma filmagem realizada no Shopping Patteo, localizado na cidade de Olinda no estado de Pernambuco. A partir dessa filmagem, selecionamos diversos *frames* que foram posteriormente submetidos a um processo de pré-processamento utilizando a plataforma *Roboflow*, como por exemplo redimensionamento, adição de ruídos, rotação e recortes. No total foram utilizadas 108 imagens para treinamento, 13 para validação e 6 para teste.

4.5.1 Resultado do Treinamento

Foram realizados 100 iterações (épocas) para o treinamento.

Figura 24: Precisão por Época

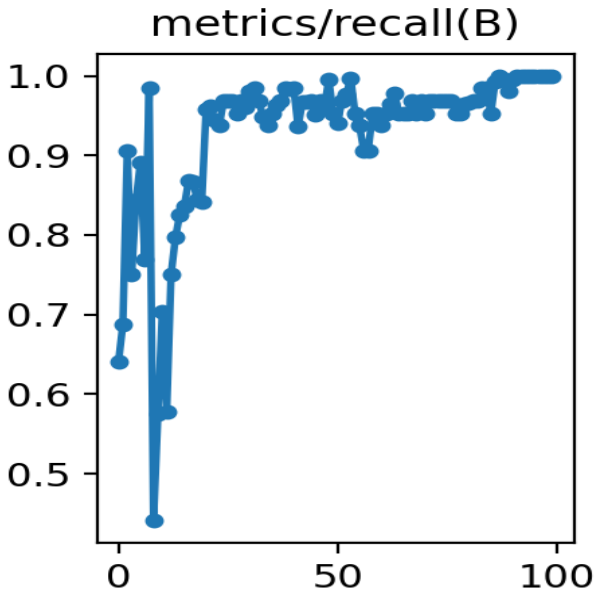


Fonte: Autoria Própria

O gráfico na figura 24 informa a precisão ao longo do tempo, nas primeiras iterações é relativamente baixa mas aumenta consideravelmente após a sétima iteração onde chega a atingir 1.0 de precisão, valor máximo. A precisão é calculado através da proporção entre as previsões positivas e verdadeiras realizadas pelo modelo, por exemplo, quando o modelo diz que há uma pessoa e de fato há uma pessoa na área selecionada e o total de previsões positivas realizada pelo independente delas serem verdadeiras, por exemplo, quando o modelo diz que há uma pessoa mas não há, bem como quando ele diz que há uma pessoa e de fato existe uma pessoa na área sugerida.

Para saber se de fato existe uma pessoa ou não na área sugerida pelo modelo é utilizado durante o treinamento uma métrica conhecida como IoU (Intersection Over Union), que calcula a área entre as caixas delimitadoras preditas e as caixas delimitadoras verdadeiras que foram adicionadas a imagem de maneira manual durante o pré-processamento, para considerar uma caixa predita pelo modelo como correta, é necessária que a sobreposição seja de ao menos 70% (ULTRALYTICS, 2023).

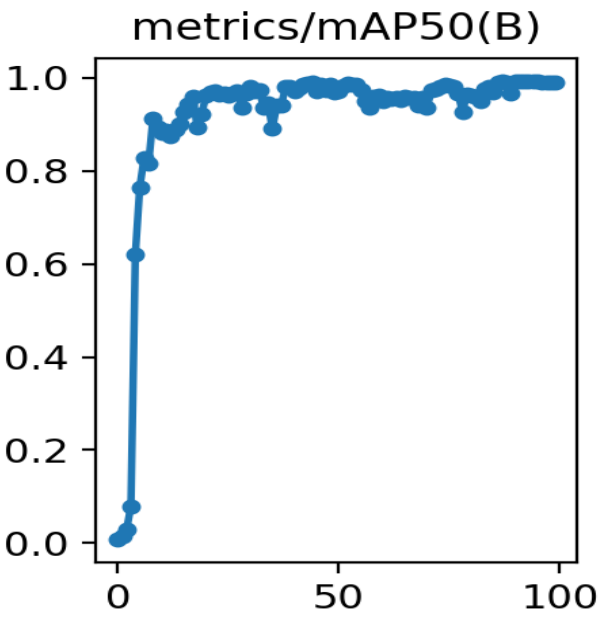
Figura 25: Recall por Época



Fonte: Autoria Própria

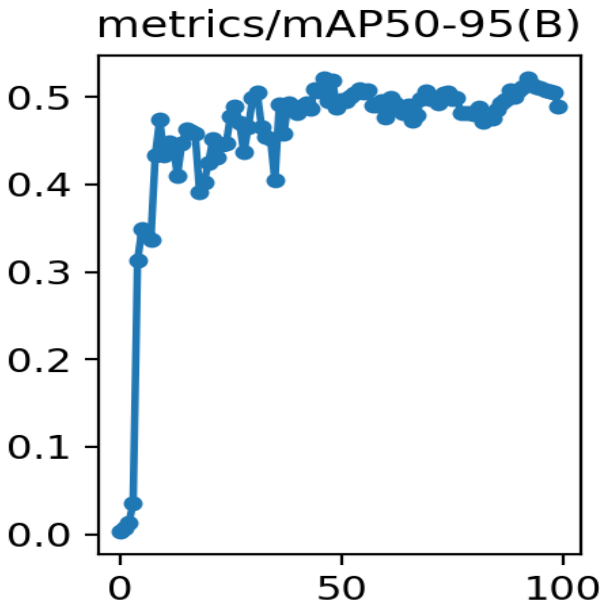
O gráfico na figura 25 informa o quão capaz é o modelo de classificar corretamente todos objetos existentes na imagem, inicialmente o *recall* pode ser baixo, no gráfico é possível observar que na primeira época o *recall* é 0.64 e na décima época cai bruscamente para 0.44, mas a partir da vigésima segunda época o valor se estabiliza, variando entre 0.9 e 1.0. O recall é calculado através da proporção de previsões positivas e verdadeiras realizadas pelo modelo pela quantidade de exemplos do objeto existente independentes do modelo ter acertado ou não.

Figura 26: mAP@50 por Época



Fonte: Autoria Própria

Figura 27: mAP@50-95 por Época

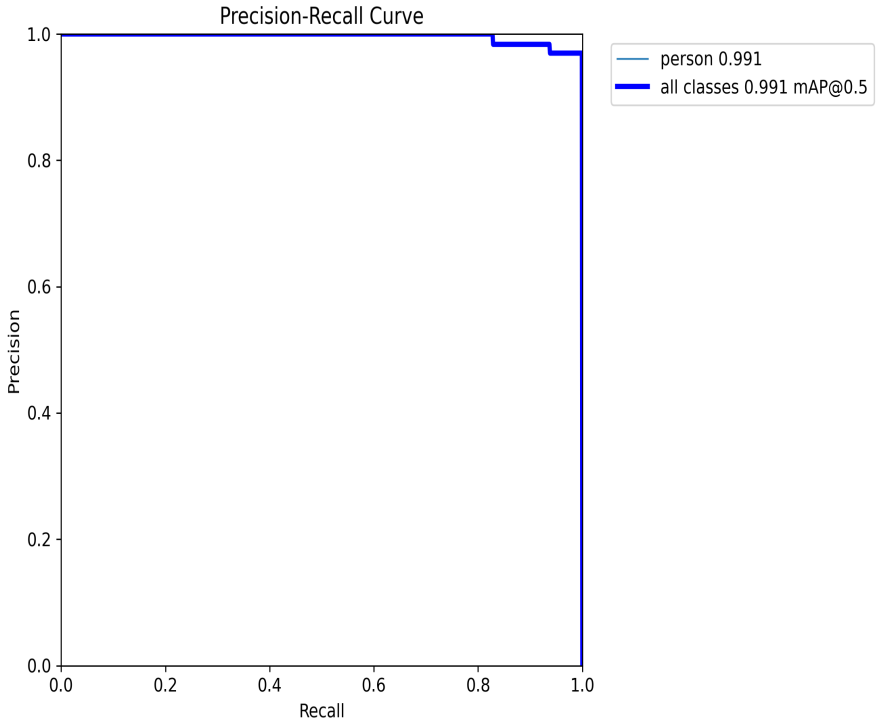


Fonte: Autoria Própria

Os gráficos da figura 26 e da figura 27 informa o quão capaz o modelo é de equilibrar a a precisão e o recall para diferente níveis de IoU, o *mAP@50* é utilizado com um IoU de 50% e para o *mAP@50-95* é utilizado valores entre 50% e 95%. É possível observar que os dois gráficos possuem comportamentos muito semelhantes, iniciando em valores baixos,

próximo a 0. No tentando o *mAP@50* chega a quase 0.99 na última época, enquanto o *mAP@50-95* termina próximo a 49%. O mean average precision é calculado através da interpolação da precisão em diferentes níveis de recall (PROCESSING; LAB, 2023).

Figura 28: Curva de Precision Recall



Fonte: Autoria Própria

Na figura (PROCESSING; LAB, 2023) é possível observar a curva de precision vs recall para um IoU de 50%, na imagem é possível observar como a precisão e o recall se correlacionam, é esperando que conforme o recall aumente a precisão diminua.

Abaixo é possível visualizar os testes de predição realizados nas imagens separadas para a validação do modelo.

Figura 29: Predições no Dataset de validação

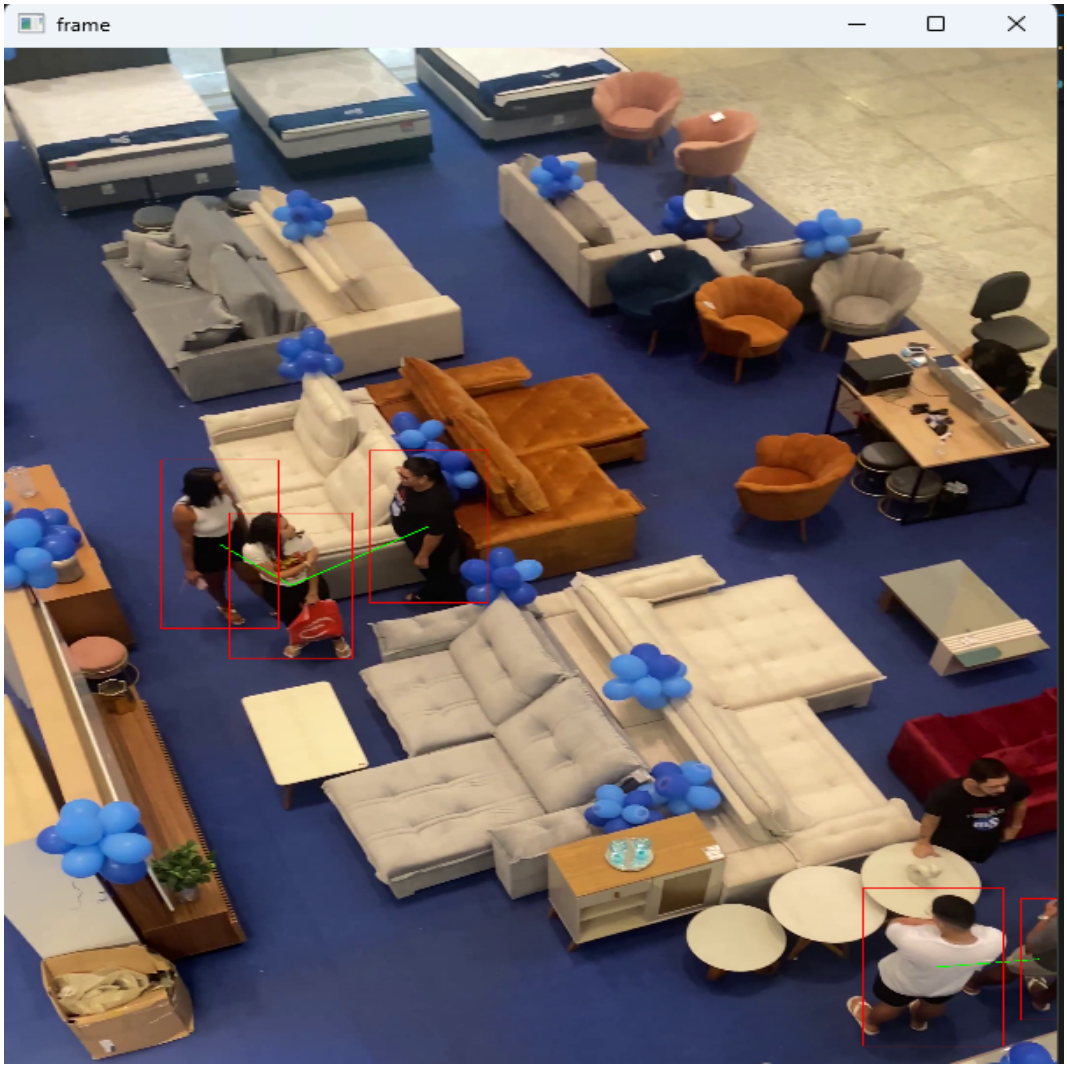
Fonte: Autoria Própria

4.6 Detectando Pessoas e calculando a Distância

A aplicação responsável por detectar as pessoas e calcular a distância entre elas foi construída em python em conjunto com *ultralytics* conforme mencionado na metodologia.

Na imagem 30 podemos visualizar esse funcionamento, as caixas delimitadoras em vermelho é desenhada quando a aplicação detecta e classifica os objetos como pessoa, quando as pessoas estão próximas uma das outras é adicionado uma linha verde conectando o centro das caixas delimitadoras, essas informações são calculadas e enviadas para API para posterior consulta.

Figura 30: Aplicação que detecta e calcula distância



Fonte: Autoria Própria

Conforme mencionado na metodologia não foi possível calcular a precisão dos cálculos de distância, uma vez que seria necessário um ambiente controlado para tal, onde seria conduzido testes de medidas para posterior calibração do algorítimo.

4.7 Execução na Raspberry

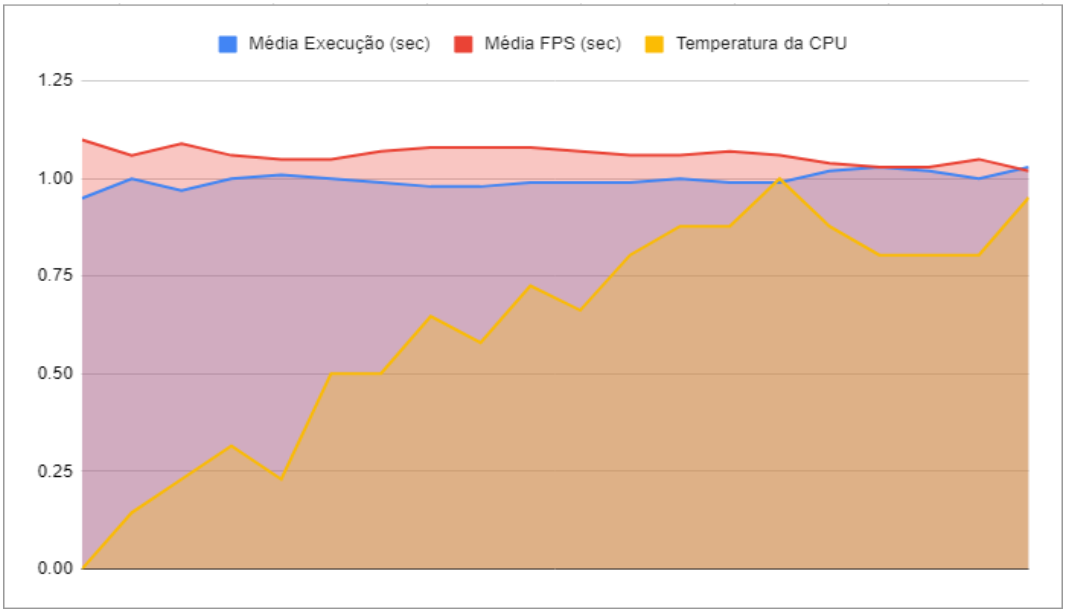
A execução do projeto responsável por detectar as pessoas e calcular a distância é feita a partir do projeto, executando o arquivo *main.py* encontrado na pasta *src*. Foi realizado um *benchmark* na raspberry referente a execução, onde foi coletado métricas como a média de execução e de frames por segundo processados, ao total foram realizado 20 execuções, na tabela abaixo é possível visualizar as métricas coletadas.

Tabela 3: Benchmark da Execução na Raspberry

Média Execução (sec)	Média FPS (sec)	Temperatura da CPU (°C)
0.95	1.10	73.04
1.00	1.06	74.98
0.97	1.09	75.96
1.00	1.06	76.93
1.01	1.05	75.96
1.00	1.05	78.39
0.99	1.07	78.39
0.98	1.08	79.85
0.98	1.08	79.37
0.99	1.08	80.83
0.99	1.07	80.34
0.99	1.06	81.31
1.00	1.06	81.80
0.99	1.07	81.80
0.99	1.06	82.78
1.02	1.04	81.80
1.03	1.03	81.31
1.02	1.03	81.31
1.00	1.05	81.31
1.03	1.02	82.29

Na figura 31 é possível visualizar as informações acima em um gráfico de área, a temperatura foi normalizado para valores entre 0 e 1.

Figura 31: Benchmark da Execução na Raspberry



Fonte: Autoria Própria

Através da imagem acima, é possível visualizar que conforme a temperatura aumenta, a área da média de execução também aumenta enquanto a área da média de *FPS* diminui, no entanto essa variação não é muito significativa.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresenta o desenvolvimento de um sistema de câmera inteligente capaz de detectar pessoas, calcular a distância entre as mesmas e fornecer uma interface web onde seja possível acompanhar as informações enviadas pelo dispositivos. Dado o exposto, os objetivos foram alcançados.

Em relação as melhorias e trabalhos futuros que podem ser adicionados ao projeto, seria propor adicionar um algorítimo que permita o sistema calibrar de maneira autônoma a variável responsável por mensurar a distância entre as pessoas, além disso, pode ser adicionado recursos a interface web que permita aos usuários acompanhar não apenas os resultados gerados pelo dispositivo, mas as imagens que estão sendo processadas.

Além disso, seria possível diminuir o custo do projeto ao utilizar um dispositivo de hardware mais econômico, como uma *ESP-32* que teria a responsabilidade apenas de enviar a imagem para um servidor capaz de processar a imagem, descartando portanto a necessidade de uma raspberry e diminuindo consideravelmente o custo do projeto.

REFERÊNCIAS

B, N. J. D.; J, S. *Roboflow*. 2022. Disponível em: ⟨https://roboflow.com⟩.

BELBACHIR PETER MICHAEL GöBEL (AUTH.), A. N. B. e. A. N. *Smart Cameras*. 1. ed. Springer, 2010. ISBN 9781441909527; 1441909524; 9781441909534; 1441909532; 1441909540; 9781441909541. Disponível em: ⟨libgen.li/file.php?md5=570dff0250b1cf1f4ad68036be23b672⟩.

BRADSKI, A. K. G. *Learning OpenCV*. [S.l.]: O'Reilly, 2008. ISBN 0596516134,978-0-596-51613-0.

CHODOROW, S. B. E. B. K. *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. Third edition. O'Reilly Media, 2019. ISBN 1491954469; 9781491954461. Disponível em: ⟨libgen.li/file.php?md5=4a072a6314a3aa500e5802f604846752⟩.

COSMINA, I. *Pro Spring 5: an in-depth guide to the Spring framework and its tools*. 5th ed. ed. Apress, 2017. ISBN 978-1-4842-2808-1,9781484228074,1484228073,1484228081. Disponível em: ⟨http://gen.lib.rus.ec/book/index.php?md5=550CE675A215B368FAB90EFE0FD936FA⟩.

CUNHA, T. *Sistema inteligente de câmeras integradas à polícia vai monitorar ruas de São Carlos*. 2018. Disponível em: <https://g1.globo.com/sp/sao-carlos-regiao/noticia/2018/07/20/sistema-inteligente-de-cameras-integradas-a-policia-vai-monitorar-ruas-de-sao-carlos.ghtml>. Acesso em: 24 Out de 2023.

DAS, S. P.; MAJUMDAR, D.; GAYEN, R. K. Monitoring social distancing through person detection and tracking using computer vision. In: *2021 5th International Conference on Electronics, Materials Engineering Nano-Technology (IEMENTech)*. [S.l.: s.n.], 2021. p. 1–5.

DB-ENGINES. *DB-Engines Ranking of Relational DBMS*. 2021. Disponível em: <https://db-engines.com/en/ranking/relational+dbms>. Acesso em: 12 Mai de 2021.

FOUNDATION, R. P. *Raspberry Pi - About us*. 2023. Disponível em: <https://www.raspberrypi.com/about/>. Acesso em: 13 Mai de 2023.

GONZALEZ, R. E. W. R. C. *Digital image processing*. 2nd ed. ed. Prentice Hall, 2002. ISBN 9780201180756; 0201180758. Disponível em: ⟨libgen.li/file.php?md5=fc074f258bca010efe764a50f97b2c18⟩.

HARTLEY, A. Z. R. *Multiple View Geometry in Computer Vision*. 2nd ed. ed. Cambridge University Press, 2003. ISBN 9780521540513; 0521540518. Disponível em: ⟨libgen.li/file.php?md5=9c3806b82e6b235e1f8c77310ff1cfda⟩.

HSU, R. O. O. L. S. *Postgresql: Up and Running: A Practical Guide to the Advanced Open Source Database*. 3. ed. O'Reilly Media, 2017. ISBN 1491963417, 9781491963418. Disponível em: ⟨http://gen.lib.rus.ec/book/index.php?md5=86935c06ed4e278d72aed3b0fe5f98ce⟩.

JOCHER, G.; CHAURASIA, A.; QIU, J. *Ultralytics YOLOv8*. 2023. Disponível em: ⟨https://github.com/ultralytics/ultralytics⟩.

MEIER, M. K. A. *SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management*. 1. ed. Springer Vieweg, 2019. ISBN 3658245484, 978-3658245481. Disponível em: ⟨http://gen.lib.rus.ec/book/index.php?md5=B96492DA730DCBC6B1DF98A906CF8552⟩.

NOBACK, M. *Advanced Web Application Architecture*. [S.l.]: Noback's Office, 2020. ISBN 9789082120165.

OMS. *COVID-19: physical distancing*. 2021. Disponível em: <https://www.who.int/westernpacific/emergencies/covid-19/information/physical-distancing>. Acesso em: 10 Mai de 2021.

PROCESSING, L. I.; LAB, C. G. *Visão Computacional::Métricas::Mean Average Precision*. 2023. Disponível em: <https://lapix.ufsc.br/ensino/visao/visao-computacionaldeep-learning/visao-computacionalmetricasmean-average-precision/>. Acesso em: 21 Out de 2023.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv*, 2018.

RUSSAKOVSKY, O. et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, IBM, v. 3, n. 3, p. 210–229, 1959.

SEWAK MD. REZAUL KARIM, P. P. M. *Practical Convolutional Neural Networks: Implement advanced deep learning models using Python*. [S.l.]: Packt Publishing - ebooks Account, 2018. ISBN 1788392302,9781788392303.

SILVA, I. N. da. *Artificial Neural Networks : A Practical Course*. 1. ed. [S.l.]: Springer International Publishing, 2017. ISBN 978-3-319-43162-8,978-3-319-43161-1.

SRIVASTAVA, S. et al. Comparative analysis of deep learning image detection algorithms. *Journal of Big Data*, v. 8, n. 1, p. 66, May 2021. ISSN 2196-1115. Disponível em: ⟨https://doi.org/10.1186/s40537-021-00434-w⟩.

ULTRALYTICS. *Configuration - Ultralytics YOLOv8 Docs*. 2023. Disponível em: <https://docs.ultralytics.com/usage/cfg>. Acesso em: 21 Out de 2023.