



FACULDADE DE TECNOLOGIA DO AMAPÁ
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

BRENO KLEVERSON ALVES COSTA
LUCAS LOPES MARTINS

SISTEMA DE CONTROLE DE ACESSO FACIAL (SCAF)

MACAPÁ – AP
2023

BRENO KLEVERSON ALVES COSTA

LUCAS LOPES MARTINS

SISTEMA DE CONTROLE DE ACESSO FACIAL (SCAF)

Trabalho de Conclusão de Curso na Modalidade de Monografia apresentada a Faculdade de Tecnologia do Amapá - META, como requisito final à obtenção do grau de Bacharel em Engenharia de Computação, sob a orientação do Prof. ME. Amerson Riley Cabral Chagas.

**MACAPÁ – AP
2023**

**BRENO KLEVERSON ALVES COSTA
LUCAS LOPES MARTINS**

SISTEMA DE CONTROLE DE ACESSO FACIAL (SCAF)

Trabalho de Conclusão de Curso na Modalidade de Monografia apresentada a Faculdade de Tecnologia do Amapá - META, como requisito final à obtenção do grau de Bacharel em Engenharia da computação.

DATA DE APROVAÇÃO: ____/____/____.

Prof. ME. Amerson Riley Cabral Chagas. (Orientador)

Prof. XXXXXXXXXXXXXXXXXXXXXXXX (Avaliador 1)

Prof^a. XXXXXXXXXXXXXXXXXXXXXXXX (Avaliador 2)

SISTEMA DE CONTROLE DE ACESSO FACIAL (SCAF)

Breno Kleverson Alves Costa¹

Lucas Lopes Martins²

Amerson Riley Cabral Chagas³

Resumo: Diante da crescente demanda por segurança, este estudo propôs apresentar o Sistema de Controle de Acesso Facial (SCAF) como uma solução consolidada. Por meio do emprego de tecnologias avançadas, como Python e OpenCV, pôde ser evidenciada eficácia na identificação de usuários e na restrição de acessos. O SCAF, nesse estágio de desenvolvimento, destaca-se como uma estrutura estável e versátil, capaz de oferecer um controle de acesso confiável e alinhado, considerando os requisitos de segurança. Sua boa performance o posiciona como uma resposta consistente para reduzir riscos e garantir a proteção em contextos reais.

Palavras-chave: Reconhecimento Facial. Inteligência Artificial. Controle de Acesso. OpenCV

¹Acadêmico do Curso de Bacharelado em Engenharia de Computação da Faculdade de Tecnologia do Amapá – META; E-mail: breno.kleverson.alves.costa@gmail.com

²Acadêmico do Curso de Bacharelado em Engenharia da Computação da Faculdade de Tecnologia do Amapá – META; E-mail: lucasmartinsk96@gmail.com

³Especialista em Tecnologia para aplicações WEB – Docente do curso de Engenharia da Computação; E-mail: amersonchagas@meta.edu.br

FACIAL ACCESS CONTROL SYSTEM (SCAF)

Breno Kleverson Alves Costa¹

Lucas Lopes Martins²

Amerson Riley Cabral Chagas³

Resumo: Facing the growing demand for security, this study aimed to introduce the Facial Access Control System (SCAF) as an established solution. Through the implementation of advanced technologies such as Python and OpenCV, the effectiveness in user identification and access restriction was demonstrated. At this stage of development, SCAF stands out as a stable and adaptable structure, capable of providing reliable and aligned access control, while considering security requirements. Its commendable performance situates it as a consistent response to risk reduction and ensuring protection within real-world contexts.

Palavras-chave: Facial Recognition. Artificial Intelligence. Access Control. OpenCV.

¹Student of the Bachelor's Degree in Computer Engineering at the Faculty of Technology of Amapá – META; E-mail: breno.kleverson.alves.costa@gmail.com

²Student of the Bachelor's Degree in Computer Engineering at the Faculty of Technology of Amapá – META; E-mail: lucasmartinsk96@gmail.com

³Specialist in Technology for WEB Applications – Professor of the Computer Engineering course; E-mail: amersonchagas@meta.edu.br

SUMÁRIO

1 INTRODUÇÃO	8
2 REFERENCIAL TEÓRICO	10
2.1 BIOMETRIA	10
2.2 INTELIGÊNCIA ARTIFICIAL	10
2.3 DEEP LEARNING (REDES NEURAIS PROFUNDAS)	11
2.4 REDE NEURAIS ARTIFICIAIS	12
2.4.1 Redes Neurais Convolucional	13
2.5 PYTHON	14
2.6 OPENCV	14
2.7 BIBLIOTECAS	15
2.7.1 Tensorflow	15
2.7.2 Keras	15
2.7.3 Pickle	16
2.7.4 Face Recognition	16
2.8 FRAMEWORK	17
2.8.1 Django	17
2.9 HARDWARE	17
2.9.1 Arduino Uno R3	17
2.9.2 Linguagem C++	18
2.9.3 Biblioteca Serial	18
2.10 CÂMERA IP	19
2.10.1 Protocolo Rtsp	19
2.11 TRABALHOS CORRELATOS	20
3 MATERIAL E MÉTODO	21
3.1 ESQUEMÁTICO DE DESENVOLVIMENTO	21
3.1.1 Esquema Lógico Hardware	21
3.1.2 Diagrama de atividade do sistema de segurança com IA	22
3.2 DESENVOLVIMENTO	23
3.2.1 Back-End	23
3.2.2 Construção Do Conjunto De Dados (Dataset)	25
3.2.3 Treinamento De Máquina (Detector_Faces.Py)	26
3.2.4 Reconhecimento Facial Webcam	30
3.2.5 Abrindo Câmera IP Com Protocolo RTSP	36
3.2.6 Protótipo Fechadura Eletrônica Com Arduino	37
3.3 FRONT-END	42
3.3.1 Tela De Login	42

3.3.2 Tela Home	44
3.3.3 Tela De Cadastro	44
3.3.4 Tela Bloquear Pessoas	47
4 ANÁLISE E DISCUSSÃO DOS RESULTADOS	49
4.1 RESULTADOS DO SISTEMA	49
4.2 RESULTADOS HARDWARE	51
4.3 SISTEMA DE CONTROLE DE ACESSO FACIAL	53
5 CONSIDERAÇÕES FINAIS.....	55
REFERÊNCIAS BIBLIOGRÁFICAS	57

1 INTRODUÇÃO

É inegável que a segurança é uma preocupação evidente no cotidiano das pessoas, e isso inclui também o ambiente empresarial, governamental e pessoal. Infelizmente, crimes como roubos, furtos ou até mesmo ataques em lugares públicos como escolas e hospitais, onde podem ser mais comuns, um exemplo atual é os crescentes ataques terroristas nas escolas, de acordo com o site de notícias aponta “A pesquisadora Telma vinha doutora em Educação e professora da Unicamp apontou o seguinte, registra 22 ataques entre os anos de 2002 e 2023, sendo 10 deles nos últimos dois anos. Isso significa um aumento de cerca de 45% de ataques em escolas no Brasil em um ano” (VIEIRA, 2023), tornando imprescindível a adoção de medidas de segurança mais eficazes.

Atualmente, a evolução tecnológica está passando por avanços significativos no que diz respeito à segurança, leitor biométrico digital, leitor RFID (identificação por rádio frequência), senhas com chave numérica, identificação biométrico facial, Íris, retina e por voz, muitas dessas tecnologias estão ficando mais comum nos dias atuais, pois isso não é à toa, explica assim a redação do site Tiinside.

O setor de segurança eletrônica faturou R\$ 9,24 bilhões em 2021. O Panorama do Mercado, apresentado hoje pela ABESE – Associação Brasileira das Empresas de Sistemas Eletrônicos de Segurança, mostra que o mercado manteve o ritmo e cresceu 14%, superando os resultados do ano anterior (13%). Os números refletem o aumento da adesão de soluções inteligentes durante a pandemia, como a implantação de body cams (câmeras corporais), os equipamentos touchless e câmeras com reconhecimento facial e uso de máscaras, os drones, o rastreamento de frotas e veículos, bem como a retomada de projetos que estavam paralisados ou que surgiram nos últimos 12 meses (TIINSIDE, 2022).

Tendo em vista as tecnologias de segurança apresentadas e sua importância no contexto atual, o presente trabalho tem como objetivo desenvolver um sistema de segurança em seu projeto final de conclusão de curso. A motivação do grupo é integrar empresas privadas, instituições públicas e governamentais, como objetivo geral pretendesse criar uma inteligência artificial capaz de identificar, por meio do monitoramento de câmeras de segurança, indivíduos não autorizados, que possam representar um possível risco para as pessoas presentes no local. Além disso, pretende-se estabelecer perfis de usuários, com um banco de imagens dos rostos dos funcionários e integrantes da empresa.

Entre os objetivos mais específicos do projeto, destacam-se a implementação de protocolos de segurança e a conexão com tecnologias, como travas de portas, alarmes e avisos. Essas medidas serão acompanhadas do envio de notificações aos administradores do sistema,

por meio de mensagens de spam, com a captura de imagens de indivíduos não autorizados que adentram setores restritos.

Através deste complexo sistema de segurança, busca-se o aprimoramento da segurança e no controle de acessos, assegurando a proteção das instalações e das pessoas no local. Ao juntar tecnologias de ponta e inteligência artificial, o objetivo é engendrar um ambiente mais seguro e notavelmente eficiente para as empresas e instituições envolvidas.

No âmbito desse projeto, a utilização de tecnologias distintamente especificadas para o sistema de segurança se faz necessário, com destaque para a linguagem de programação Python. A escolha dessa linguagem altamente usada, está relacionada em sua usabilidade simples e descomplicada, o que a torna ideal para a criação de Inteligência Artificial (IA). Além disso, Python oferece uma extensa gama de bibliotecas acessíveis, onde a integração com OpenCV (Open Source Computer Vision Library) assume um papel particularmente significativo. Esta integração confere a capacidade de processamento de imagem e vídeo, bem como a assimilação perfeita com câmeras de vigilância, tornando uma harmonização indispensável para a criação do sistema de segurança.

Para a realização do projeto, a criação de um protótipo tangível para o sistema de segurança torna-se uma necessidade fundamental. O projeto exige a implantação de componentes eletrônicos específicos, incluindo uma câmera de segurança 4k de última geração, um microcontrolador Arduino Uno R3, trava elétrica solenoide e um alto-falante para simular o alarme. Esses materiais serão utilizados nos testes iniciais da aplicação do sistema. O protótipo físico é fundamental para avaliar e validar o funcionamento do sistema em um ambiente real.

Os testes do sistema, juntamente com o protótipo físico, serão realizados em ambientes controlados. Inicialmente, serão conduzidos testes na instituição de ensino Faculdade Meta, em seguida, em residências e outros ambientes relevantes. Esses testes práticos permitirão a avaliação do desempenho do sistema em diferentes cenários e a identificação de possíveis melhorias. A realização desses testes é essencial para garantir a eficácia e a confiabilidade do sistema de segurança desenvolvido.

2 REFERENCIAL TEÓRICO

Neste capítulo, será apresentado o escopo teórico abrangente deste projeto final de curso. Ele fornecerá uma base sólida para a área em questão, demonstrando a relevância das bases já estabelecidas por meio de dados, artigos científicos, sites e revistas acadêmicas. Essa revisão teórica servirá como embasamento para o desenvolvimento deste trabalho, garantindo que ele esteja fundamentado em conhecimentos pré-existentes e contribua para o avanço do campo de estudo.

2.1 BIOMETRIA

Os sistemas biométricos fundamentam-se na aferição e registro de padrões mensuráveis, que se associam a indivíduos, mediante o emprego de técnicas de inteligência artificial e discernimento de padrões. No contexto atual, as tecnologias biométricas proeminentes englobam a biometria de impressão digital, de geometria das mãos e dedos, faciais, de íris e de voz. O aprimoramento contínuo das tecnologias de identificação e detecção assume um caráter desafiador, impulsionado pelo crescimento das atividades fraudulentas, pelo avanço tecnológico e pelo escopo das ameaças. Por conseguinte, emerge a necessidade premente de uma evolução substancial na salvaguarda das pessoas, de bens e dos recursos pessoais, públicos ou privados. (JAIN; NANDAKUMAR; ROSS, 2016).

O objetivo de um Sistema Biométrico é fornecer mecanismos para que seja possível, através das características do indivíduo, identificá-lo com um grau de certeza aceitável e, se utilizado de forma apropriada pode diminuir, consideravelmente, os problemas relacionados com a segurança. Um sistema biométrico consiste em um conjunto de hardware e software para o reconhecimento de padrões, que opera através da aquisição automática das informações biométricas, extraíndo um modelo a partir dessas informações e esse modelo será armazenado e utilizado para as comparações, ou seja, em uma primeira fase, amostras da característica biométrica são recolhidas, transformadas em um modelo e armazenadas e, em uma segunda etapa, uma amostra da característica biométrica é recolhida e comparada com as previamente armazenadas para ser possível chegar na identidade do indivíduo. (BRENNER; BIZARRIA, 2023, P.7).

2.2 INTELIGÊNCIA ARTIFICIAL

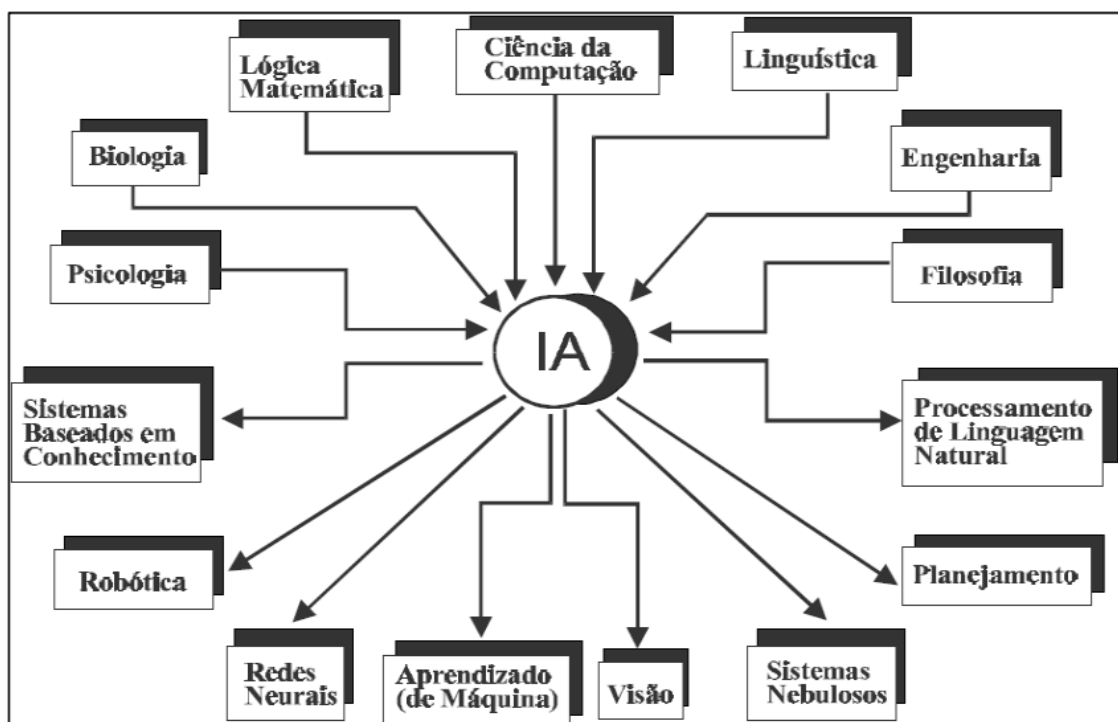
Na segunda década do século XXI, a combinação de diversas tecnologias, como a Internet das Coisas (IoT), blockchain, impressão 3D, robótica avançada e a inteligência artificial (IA), tem superado as previsões anteriores, resultando em máquinas e sistemas inteligentes executando tarefas que costumavam ser exclusivas dos seres humanos. Apesar disso, esses sistemas ainda não possuem a capacidade essencial da inteligência humana, como compreensão de significados e intuição. A IA atual, especialmente o deep learning, tem trazido

avanços notáveis, como diagnósticos médicos superiores e análises de grandes volumes de dados para obter insights valiosos (KAUFMAN; SANTAELLA, 2019).

Assim, o domínio de IA se caracteriza por ser uma coleção de modelos, técnicas e tecnologias (busca, raciocínio e representação de conhecimento, mecanismos de decisão, percepção, planejamento, processamento de linguagem natural, tratamento de incertezas, aprendizado de máquina) que, isoladamente ou agrupadas, resolvem problemas de tal natureza. Para tal, podem utilizar paradigmas distintos, sendo os principais os paradigmas simbólico, conexionista, evolutivo e probabilístico. (SICHMAN, 2021, P.39)

Assim demonstrado na Figura 01, sobre a alta complexidade de conteúdos que a IA poderá ter e abordar.

Figura 01: Inteligência Artificial



Fonte: MONARD; BARANAUKAS, 2000, p.2.

2.3 DEEP LEARNING (REDES NEURAIS PROFUNDAS)

Deep Learning é um conjunto de técnicas que utiliza Redes Neurais Artificiais Profundas e Machine Learning (Aprendizado de Máquina), com muitas camadas intermediárias formando a em uma estrutura de entrada e saída (LECUN, 2015).

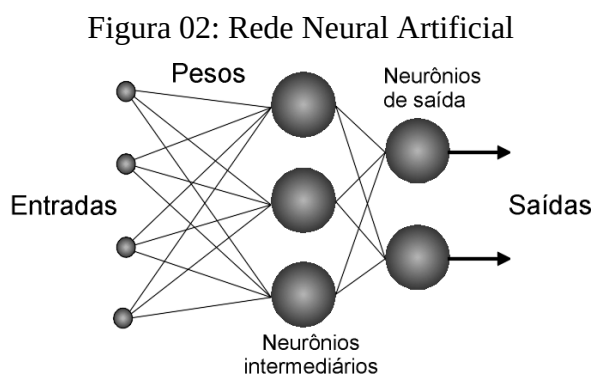
Deste modo uma pequena diferença entre a utilização entre Machine Learning e Deep Learning, Aprendizado de máquina diz respeito ao uso de algoritmos para organizar dados, reconhecer padrões e fazer com que computadores possam aprender. Isso para gerar insights

inteligentes sem necessidade de pré-programação e redes neurais profundas é a parte do aprendizado de máquina que, por meio de algoritmos de alto nível, reproduz a rede neural do cérebro humano, podemos dizer que machine learning estabeleceu as bases para deep learning evoluir (HAMAGUTI; BREVE, 2020).

Deep Learning utiliza algoritmos de aprendizado de máquina que trabalham em vários níveis, com diferentes graus de abstração. Esses modelos aprendidos correspondem a diferentes graus de conceitos, onde os conceitos de alto nível são definidos a partir de níveis inferiores, e os mesmos conceitos de nível inferior ajudam definir muitos conceitos de nível superior concentrando a aprendizagem de múltiplos níveis de representação e abstração que ajudam a dar sentido aos dados, como imagens, sons e textos. (HOSAKI; RIBEIRO, 2021, P.3).

2.4 REDE NEURAIIS ARTIFICIAIS

As redes neurais artificiais são amplamente utilizadas em tarefas de reconhecimento facial, um campo que tem ganhado cada vez mais importância em segurança, vigilância e identificação de pessoas. Eles são compostos por camadas de neurônios artificiais que se comunicam entre si por meio de conexões ponderadas e podem aprender com os dados, conforme mostrado na Figura 2. Esses processos de aprendizado são baseados no ajuste dos pesos das conexões entre os neurônios para minimizar o erro entre a saída prevista da rede e a saída esperada. Esse ajuste é realizado por meio de algoritmos de treinamento como backpropagation, que empregam técnicas de otimização para encontrar os pesos que minimizam o erro (SILVA; OLIVEIRA, 2019).



Fonte: Cerebromente (1998)

As redes neurais possuem a notável capacidade de realizar reconhecimento facial com precisão e eficiência surpreendentes. Por meio dos algoritmos de aprendizado de máquina, eles aprendem a perceber e entender as complexas características faciais dos indivíduos a partir de uma coleção cuidadosamente selecionada de imagens de treinamento. Essas redes extraem e codificam astutamente esses atributos faciais em representações numéricas distintas e compactas, conhecidas como vetores de recursos. Esses vetores possuem uma qualidade muito atrativa para a criação de projetos, pois guardam as características da identidade única de cada

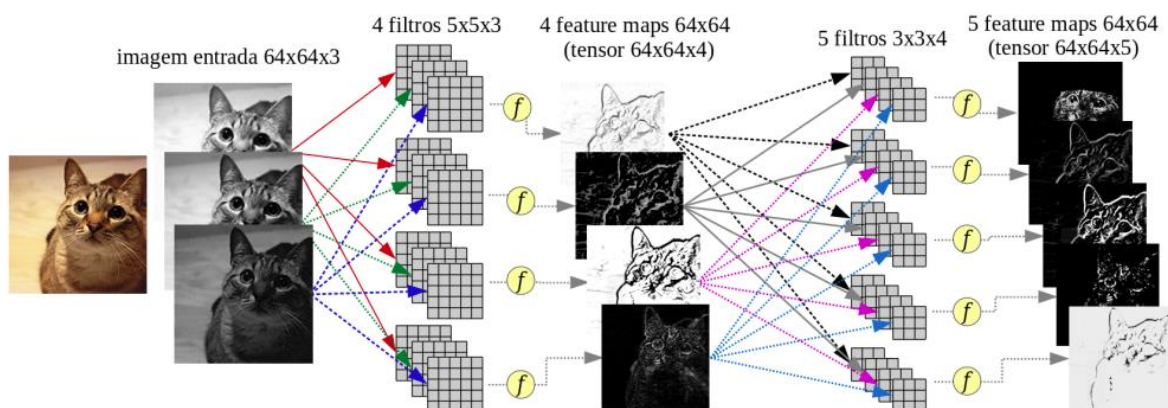
um, onde pode comparar com outros vetores para identificar se realmente é o mesmo indivíduo (CARVALHO, 2018).

2.4.1 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (CNNs) emergem como um dos modelos mais reconhecidos e amplamente empregados no domínio do Aprendizado Profundo. A característica distintiva dessas estruturas reside na sua configuração predominantemente composta por camadas convolucionais, engajadas na análise das entradas por meio de campos receptivos locais. Além disso, tais redes incorporam operações como o pooling, que desempenha um papel crucial na redução da dimensionalidade espacial das representações. No cenário contemporâneo, as notáveis CNNs abarcam notavelmente as arquiteturas ResNet e Inception (PONTI; COSTA, 2017), apontando para a contínua evolução desses paradigmas de aprendizado profundo, na Figura 03 irá ser demonstrado um exemplo de camadas convolucional, mas antes as autoras a seguir irão aprofundar mais no entendimento sobre:

A principal aplicação das CNNs é para o processamento de informações visuais, em particular imagens, pois a convolução permite filtrar as imagens considerando sua estrutura bidimensional (espacial). Considere o exemplo da base de dados MNIST apresentado na seção anterior. Note que ao vetorizar as imagens estamos desprezando toda a estrutura espacial que permite entender a relação entre os pixels vizinhos em uma determinada imagem. É esse tipo de estrutura que as CNNs tentam capturar por meio da camada convolucional (RAZENTE; BARIONI, 2017, p. 74)

Figura 03: Camadas convolucionais.



Fonte: RAZENTE; BARIONI, 2017.

A representação visual de duas camadas convolucionais destaca-se: a primeira apresenta 4 filtros de dimensões $5 \times 5 \times 3$, que recebem como entrada uma imagem RGB de dimensões $64 \times 64 \times 3$, resultando na geração de um tensor com 4 mapas de características. Na segunda camada convolucional, 5 filtros de dimensões $3 \times 3 \times 4$ são aplicados para filtrar o tensor da

camada anterior, culminando na criação de um novo tensor de mapas de características com dimensões $64 \times 64 \times 5$. Os círculos após cada filtro representam funções de ativação, como a conhecida ReLU. A convolução convencional é executada com um passo (stride) de 1, ou seja, cada pixel é percorrido pelos filtros, resultando em uma nova imagem do mesmo tamanho da entrada, ou seja, 64×64 . A adoção de passos maiores que 1 é frequente quando a intenção é agilizar a execução, pulando pixels e gerando imagens menores. Por exemplo, ao utilizar um passo (stride) de 2, a saída será uma imagem de dimensões 32×32 (RAZENTE; BARIONI, 2017, P. 76).

2.5 PYTHON

A linguagem de programação Python foi criada por Guido van Rossum em 1989. Voltada a microcomputadores Intel/AMD com Linux, Windows (IBM-PC) e Mac OS (Intel Macintosh), busca eficiência e produtividade. Inspirado por ABC e desmotivado por C e Shell Script, Guido lançou a versão 0.9 em 1991. Mantida pela Python Software Foundation, Python homenageia "Monty Python's Flying Circus". O nome, relacionado a uma cobra python, surgiu por acaso na capa de um livro da O'Reilly. É uma linguagem de programação muito usada, pois combina aplicações científicas e comerciais (MANZANO, 2018).

O Python é uma linguagem versátil, se destaca muito na inteligência artificial graças a suas bibliotecas especializadas, como NumPy e pandas, que simplificam manipulação de dados. Também existem diversas outras, como Scikit-learn, TensorFlow, OpenCV e PyTorch que impulsionam aprendizado de máquina e aprendizado profundo. Ele se destaca na área da inteligência artificial devido à sua sintaxe clara e legível, tornando a tradução de algoritmos complexos para código acessível, capacitando desenvolvedores a criar soluções avançadas para desafios complexos, reforçando seu papel como líder na vanguarda tecnológica da inteligência artificial (MANZANO, 2018).

2.6 OPENCV

Originalmente desenvolvido pela Intel, o OpenCV (Open Computer Vision) é uma biblioteca gráfica notável. Fornece recursos especializados para processamento de imagens, seja para componentes fotográficos ou videográficas. Criada em 1999, esta biblioteca foi construída com o objetivo principal de oferecer aos programadores e pesquisadores uma plataforma comum para visão computacional. Tem a capacidade de estender sua

disponibilidade para a maioria dos sistemas operacionais e é compatível com várias linguagens de programação, como Python, C++, Java e MATLAB (RIBEIRO, 2020).

Desde seu desenvolvimento, o OpenCV continua sendo muito usado, principalmente em domínios de aplicação, como o reconhecimento facial e à detecção de objetos, assim como ao rastreamento de pessoas ou objetos em movimento e realidades aumentadas. Esta extraordinária biblioteca abrange uma vasta gama de técnicas de processamento de imagem, desde uma simples filtragem até a mais complexa precisão da segmentação, como reconhecimento de padrões, classificação e principalmente o reconhecimento de características (OLIVEIRA, 2017).

No campo da visão computacional, o OpenCV se destaca como um recurso inestimável para desenvolvedores da área da computação. Sua natureza versátil e compatibilidade com várias linguagens de programação, onde entrega diversas possibilidades, onde tem se mostrado uma ferramenta valiosa. Um dos principais motivos para sua permanência no âmbito computacional, é a sua constante atualização por uma comunidade de usuários e desenvolvedores dedicados que contribuem incansavelmente para seu crescimento e garantem sua relevância eterna no campo em constante evolução da visão computacional (OLIVEIRA, 2017).

2.7 BIBLIOTECAS

Nesse tópico será abordar as bibliotecas utilizadas no projeto para o treinamento e inteligência artificial.

2.7.1 Tensorflow

Neste projeto será utilizado uma biblioteca que oferece diversos níveis de abstração de código aberto para aprendizado de máquina e inteligência artificial. Este pode ser empregado em diversas tarefas, com destaque para sua aplicação específica no treinamento e inferência de redes neurais profundas (ABADI; AGARGWAL; BARHAM, 2016, p.1-3).

Tensorflow é uma plataforma que facilita a construção da arquitetura de aprendizado de máquina e redes neurais através da implementação de Interfaces de Aplicação de Programação (API) de alto nível para a modelagem do sistema, auxiliando a configuração de camadas ocultas e a transferência de aprendizado. Sua principal característica é o processamento de dados organizados como tensores, estruturas matemáticas descritas como generalizações de escalares e vetores. Ela possui suporte aos drivers CUDA da NVIDIA, permitindo assim utilizar GPU como paralelismo computacional para reduzir o tempo de treino e aumentar a performance de execução das rotinas de detecção de objetos (JÚNIOR, 2020, p.45).

2.7.2 Keras

Keras, uma interface de programação de aplicações (API), foi idealizada com foco nos seres humanos, não nas máquinas. Guiando-se pelas práticas mais eficientes para diminuir o

esforço mental, Keras oferece interfaces de programação coerentes e simplificadas, minimizando as intervenções necessárias por parte do usuário em casos usuais e proporcionando mensagens de erro nítidas e orientadas à ação. Além disso, Keras atribui a mais alta prioridade à elaboração de documentações e guias do desenvolvedor de qualidade excepcional (KERAS, 2023).

2.7.3 Pickle

Sobre o uso da biblioteca pickle e à compreensão mais aprofundada de suas funcionalidades:

O pickle implementa protocolos binários para serializar e desserializar uma estrutura de objeto Python. “Pickling” é o processo pelo qual uma hierarquia de objetos Python é convertida em um fluxo de bytes, e “unpickling” é a operação inversa, por meio da qual um fluxo de bytes (de um arquivo binário ou objeto semelhante a bytes) é convertido novamente em uma hierarquia de objetos. A decapagem (e a decapagem) é alternativamente conhecida como “serialização”, “organização” ou “achatamento”; entretanto, para evitar confusão, os termos usados aqui são “decapagem” e “decapagem” (Python Documentação, 2023).

A persistência é viabilizada por meio do módulo pickle e cPickle (um pickle implementado em C, com desempenho até 1000 vezes superior). Essa biblioteca tem a finalidade de armazenar um objeto no formato de “byte stream” ou converter um arquivo byte stream de volta para um objeto. Importante destacar que não é seguro abrir um “byte stream” de origem desconhecida com este programa, pois ela pode conter código malicioso (ARTÉM; CASTRO; MAURO, 2012).

2.7.4 Face Recognition

A biblioteca de reconhecimento facial, construída com a tecnologia de última geração do dlib e baseada em aprendizado profundo, proporciona uma eficaz capacidade de reconhecimento e manipulação de rostos em Python ou na linha de comando. O modelo integrado apresenta uma notável precisão de 99,38% ao ser referenciado em rostos rotulados na natureza. Além do reconhecimento facial, a biblioteca oferece funcionalidades para a obtenção da localização e contorno dos olhos, nariz, boca e queixo de cada indivíduo (PyPI, 2023).

Embora seu uso seja aplicável em cenários significativos, como reconhecimento facial em tempo real, destaca-se por sua versatilidade ao possibilitar tarefas menos convencionais, como a aplicação de maquiagem digital, similar ao conceito presente em “Meitu”. A biblioteca também proporciona uma ferramenta de linha de comando, denominada “face_recognition”, que facilita o reconhecimento facial em uma pasta de imagens por meio da interface de linha de comando. Essa abordagem flexível e abrangente destaca a utilidade dessa biblioteca em diversas aplicações, desde tarefas técnicas até aplicações mais lúdicas (PyPI, 2023).

2.8 FRAMEWORK

Neste projeto foi utilizado um framework para auxiliar na criação do sistema SCAF, que foi essencial por sua facilidade e suporte.

2.8.1 Django

O Django é um framework de desenvolvimento web em Python, criado por desenvolvedores para simplificar o processo de construção de aplicações web. Ele gerencia tarefas complexas, como tratamento de requisições, mapeamento objeto-relacional e segurança contra ameaças, possibilitando que os programadores se concentrem nas regras de negócio. Sua arquitetura "zero-compartilhamento" oferece escalabilidade, facilitando a adição de recursos conforme necessário. Embora muitos o considerem um modelo MVC (Model-View-Controller), o Django é definido como MTV (Model-Template-View), com ênfase nas Views para determinar o que é apresentado e nos Templates para a formatação desse conteúdo (MACIEL, 2020).

Uma das características marcantes do Django é seu compromisso com a segurança. Ele é projetado para evitar os ataques mais comuns, como Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) e SQL injection. Essa ênfase na segurança torna o Django uma escolha confiável para a construção de aplicações robustas e protegidas (MACIEL, 2020).

2.9 HARDWARE

Nesta parte serão apresentadas as ferramentas e bibliotecas para a parte física do projeto que será interligado com o sistema SCAF.

2.9.1 Arduino Uno R3

O Arduino Uno R3 é um microcontrolador, uma espécie de "cérebro" programável que pode controlar vários componentes eletrônicos. Ele é composto por uma placa com um processador, pinos de entrada e saída, e várias conexões para sensores, luzes, motores e muito mais. O "R3" no nome se refere à revisão da placa, sendo a terceira versão do Arduino Uno (NUSSEY, 2018).

O Arduino é uma ferramenta muito presente nos desenvolvimentos de projetos eletrônicos. De acordo com Nussey (2018), o Arduino R3 é uma placa de desenvolvimento eletrônico altamente versátil, projetada para iniciantes em eletrônica e programação. Com recursos simplificados e uma ampla gama de sensores e atuadores disponíveis, o Arduino R3 permite criar projetos interativos e experimentar conceitos básicos da automação e da Internet

das Coisas. Ideal para educadores e entusiastas, o Arduino R3 oferece uma maneira didática de aprender sobre eletrônica e programação através de projetos práticos e divertidos.

2.9.2 Linguagem C++

O C++ foi originalmente concebido para abordar a manipulação direta de hardware, mantendo uma abstração eficiente de alto nível. Ao longo dos anos, evoluiu de uma solução simples para programação de sistemas nos anos 1980 para uma ferramenta complexa e eficaz em diversas aplicações. O foco do C++ permanece na combinação de mapeamento direto de construções de linguagem para hardware e abstração sem sobrecarga. Essa dualidade é a característica distintiva da linguagem, proporcionando "nenhuma gordura distribuída" e a codificação manual otimizada do que é utilizado. O C++ continua a adaptar-se a novos desafios e estilos de uso, com sua evolução documentada de 2006 a 2020, refletindo as necessidades em constante mudança dos desenvolvedores. É considerado uma linguagem industrial, composta por elementos além do idioma, como bibliotecas, ferramentas, bases de código extensas e suporte da comunidade, proporcionando uma ferramenta abrangente para construção de sistemas robustos (STROUSTRUP, 2020).

O C++ surgiu nos Bell Labs nos anos 80, concebido por Bjarne Stroustrup para desenvolver uma versão distribuída do Unix, mantendo a compatibilidade com C e incorporando elementos do Simula. A escolha da linguagem C como base foi devida à sua proposta genérica, velocidade e portabilidade. A evolução do C++ incluiu a adição de características como funções virtuais, sobrecarga de operadores e herança múltipla. Sua biblioteca padrão, com destaque para a biblioteca de E/S e a Standard Template Library (STL), desempenhou papel crucial em sua distinção em relação ao C. A linguagem influenciou outras como Java, a IDL de CORBA e C# (PANOSSO, 2019).

2.9.3 Biblioteca Serial

A biblioteca Software Serial viabiliza a comunicação serial em pinos digitais específicos de uma placa Arduino, utilizando software para replicar essa funcionalidade. Essa biblioteca permite a criação de múltiplas portas seriais de software, cada uma com velocidades de até 115.200 bps, incluindo a opção de sinalização invertida. Essa ferramenta é empregada para a comunicação entre uma placa Arduino e um computador ou outros dispositivos, tornando-se especialmente útil em placas que possuem limitações nas portas seriais físicas, como o Uno, Nano, Mini e Mega (ARDUINO, 2023).

A comunicação serial (UART) na plataforma Arduino é, sem dúvida, um poderoso recurso que possibilita a comunicação entre a placa e um computador ou entre a placa e outro dispositivo, como por exemplo um módulo GPS ou um módulo GSM. É através desse canal que é realizado o upload do código para a placa (CENTRON, 2018, p.1).

2.10 CÂMERA IP

As câmeras IP (Internet Protocol), conhecidas também por câmeras de rede, proporcionam uma maior qualidade de imagem, facilidade na instalação, além de poder acessar e controlar as gravações através de um navegador web, conectado em uma rede IP, como a LAN, Internet ou intranet (SILVA, 2018).

A primeira câmera IP foi criada em 1996 pela empresa Axis Communications fundada em 1984 por Martin Gren, Mikael Karlsson e Keith Bloodworth. A câmera denominada Axis Neteye 200 digitalizava as imagens analógicas e transmitia pela rede usando o protocolo TCP/IP, ela trabalhava com apenas 3 frames por minuto, era uma revolução na época, pois permitia o acesso remoto em redes já instaladas e contriuiu para uma economia de milhares de dólares em projetos industriais por permitir o monitoramento remoto (SILVA, 2018, P.32).

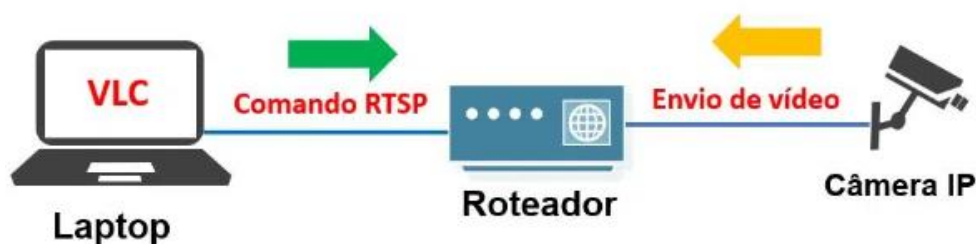
2.10.1 Protocolo Rtsp

As câmeras IP são equipadas com o protocolo RTSP (Real Time Streaming Protocol), o que viabiliza a abertura do fluxo de vídeo em diferentes aplicações, como, por exemplo, o VLC Media Player. Para visualizar a imagem da câmera no seu computador usando o VLC Media Player através do protocolo RTSP (INTELBAS, 2022), este protocolo será utilizado para o desenvolvimento do sistema.

O RTSP (Real Time Streaming Protocol - Protocolo de Transmissão em Tempo Real) em sua forma básica é adequado apenas para redes locais que são cuidadosamente gerenciadas. Ele não oferece nenhum mecanismo para recuperar a perda de pacotes no Protocolo de Transporte em Tempo Real (RTP). As portas UDP usadas pelo RTSP são atribuídas dinamicamente, o que pode ser problemático em ambientes com firewalls, pois essas portas não são facilmente compatíveis com restrições de firewall (KAUFFMANN, 2020).

O protocolo permite solicitar fluxos de mídia sob demanda e sincronizar múltiplos streams em uma apresentação. A configuração é definida em uma descrição de apresentação acessível via HTTP, independente do servidor de mídia, como mostrada na Figura 04 (KAUFFMANN, 2020).

Figura 04: Mídia sob demanda em RTSP



Fonte: APRENDACFTV, 2019

2.11 TRABALHOS CORRELATOS

De acordo com Filho (2018) apresenta um projeto baseado no desenvolvimento de sistemas de reconhecimento facial e automação por meio de microcontroladores. Foram criados dois módulos distintos: um para controle de acesso baseado em reconhecimento facial, operando em ambiente desktop, e outro para o gerenciamento de ambientes, acessível por interface web. O módulo de reconhecimento facial garante acesso apenas a pessoas autorizadas, utilizando algoritmos de treinamento para identificar indivíduos por suas características faciais. O controle de ambientes engloba supervisão de parâmetros como temperatura e umidade, focando em locais como salas de servidores.

De acordo com Gomez (2019) demonstra um projeto que visa desenvolver um sistema de detecção facial utilizando uma plataforma embarcada e de baixo custo com um sistema Linux embarcado, junto a uma webcam integrada no sistema para o reconhecimento facial. Todo o sistema desenvolvido em Python inclui algoritmos de extração de características, processamento de vídeos e visão computacional. Os resultados preliminares mostraram uma baixa taxa de erros e eficácia em condições desafiadoras. O projeto visou aprimorar a segurança através de sistemas embarcados e reconhecimento facial, utilizando tecnologias avançadas e algoritmos robustos.

3 MATERIAL E MÉTODO

Neste capítulo, serão apresentados os principais materiais e métodos utilizados no desenvolvimento deste trabalho, com destaque para criação da parte de software, onde foi utilizado a biblioteca OpenCV e a linguagem de programação Python, que desempenharam papéis fundamentais nas abordagens adotadas para implementar o sistema de segurança com identificação facial. Também será abordada a parte de hardware, no qual foram utilizados câmera de segurança, Arduino, relé e fechadura eletrônica. Essa integração entre o software de detecção facial, e o hardware da trava de segurança com a câmera de segurança, proporcionam um sistema eficiente de segurança.

3.1 ESQUEMÁTICO DE DESENVOLVIMENTO

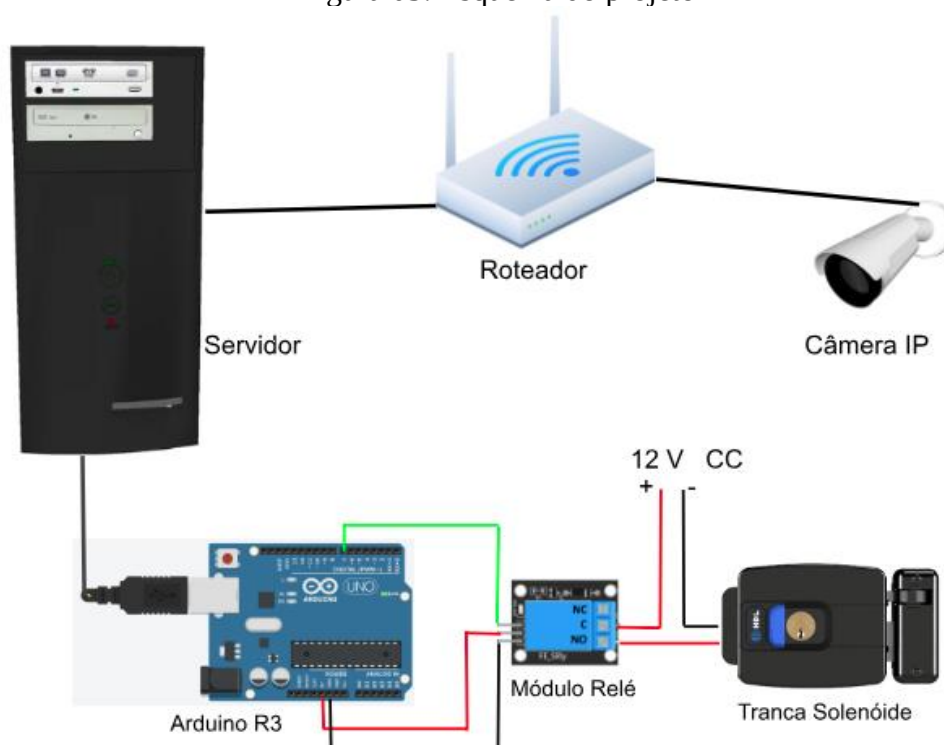
Neste espaço, serão apresentados os esboços referentes à elaboração da parte física, bem como ao avanço do desenvolvimento do sistema no âmbito do projeto.

3.1.1 Esquema Lógico Hardware

O esquema lógico proposto para controle de acesso baseado em reconhecimento facial é composto por vários componentes interconectados para garantir a segurança e eficácia do processo. Inicialmente, uma câmera IP é instalada e conectada a um roteador através do wi-fi, permitindo a captura contínua de imagens do ambiente monitorado. Essas imagens são então transmitidas para um servidor central. O servidor possui um sistema de detecção facial, que emprega algoritmos de aprendizado de máquina para analisar e identificar rostos humanos nas imagens recebidas.

Quando um rosto é detectado e reconhecido pelo sistema de detecção facial, o servidor consulta uma base de dados de rostos autorizados. Se houver uma correspondência positiva, o servidor gera um sinal de controle. Esse sinal é enviado ao Arduino R3, um microcontrolador conectado ao servidor. O Arduino, por sua vez, interpreta o sinal e aciona um relé que controla o mecanismo de travamento da porta de acesso, como mostrado na Figura 05.

Figura 05: Esquema do projeto



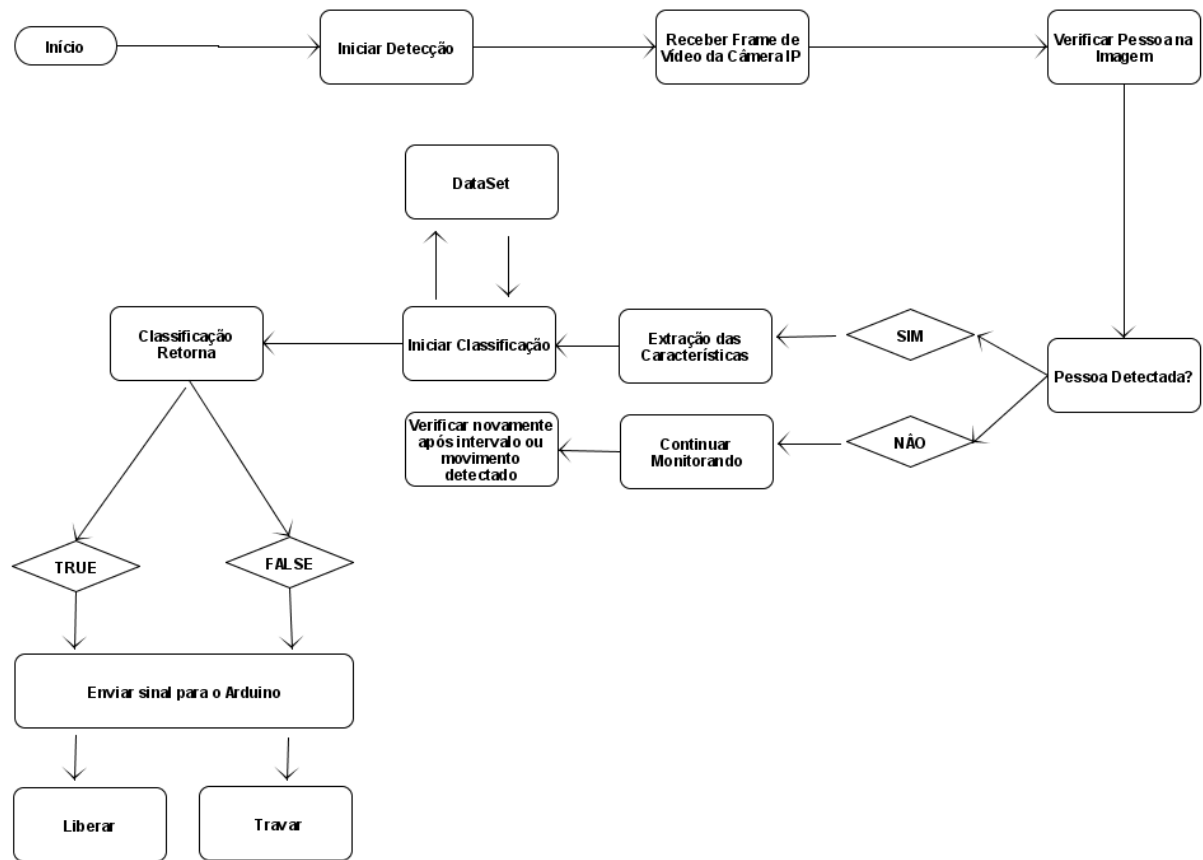
Fonte: Autores, 2023.

3.1.2 Diagrama de atividade do sistema de segurança com IA

Para o desenvolvimento de um sistema foi elaborado um fluxograma, como mostrado na Figura 06, onde mostra de forma simplificada de toda a base do sistema. Nesse fluxograma ilustra o funcionamento de um Sistema de Reconhecimento Facial, destacando as etapas envolvidas no processo. No início, o usuário é apresentado com uma opção de "Iniciar Detecção", quando selecionada a opção, o sistema passa a receber frames de vídeo da câmera IP e verificar a presença de pessoas na imagem. Se uma pessoa for detectada, a detecção de sua identidade é iniciada por meio da classificação facial. O sistema então verifica se a pessoa é autorizada, baseando-se em um conjunto de dados de rostos previamente cadastrados.

A partir dessa verificação, o sistema toma a ação. Se uma pessoa autorizada, um sinal é enviado para um Arduino, liberando a trava da porta. Caso contrário, a porta permanece travada e uma mensagem é exibida. O fluxograma também mostra que o sistema continua monitorando constantemente para novas detecções, garantindo uma segurança contínua.

Figura 06: Fluxograma Base



Fonte: Autores, 2023.

3.2 DESENVOLVIMENTO

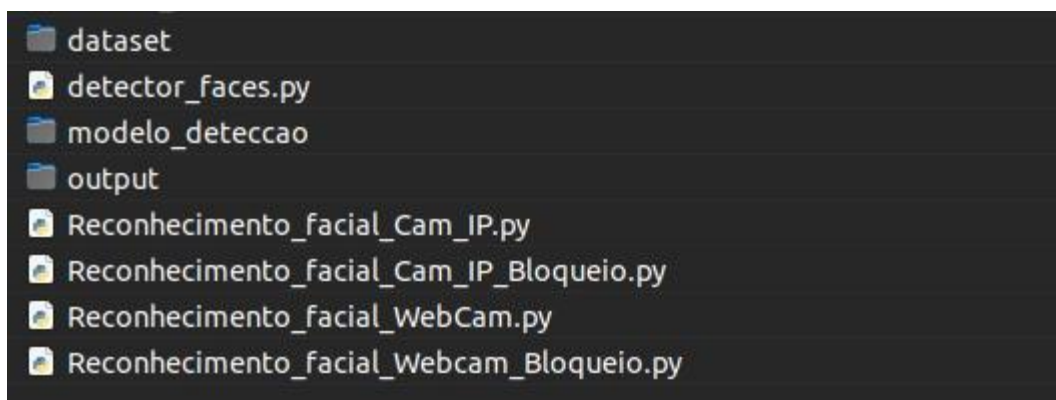
Nesta etapa, foi dedicado tempo ao desenvolvimento e treinamento de IA para dar continuidade ao projeto. Optou-se pela linguagem Python, que proporcionou a utilização de bibliotecas e modelos que facilitam o treinamento mais detalhado para a identificação facial.

3.2.1 Back-End

A infraestrutura fundamental do projeto será meticulosamente preservada no diretório do back-end¹, englobando não somente a estrutura de dados e o conjunto de dados, mas também todos os componentes essenciais referentes ao treinamento e ao reconhecimento facial, conforme explicitado detalhadamente na Figura 07.

¹ Link Github: https://github.com/BrenoKleverson0320/SCAF/tree/main/app_sistema_seguranca/back_end

Figura 07: Divisão das pastas



Fonte: Autores, 2023.

- **dataset:** O dataset consiste em um repositório de imagens de pessoas, categorizadas em três grupos distintos: alunos, funcionários e indivíduos com acesso bloqueado.
- **detector_faces.py:** essa aplicação tem a finalidade de extrair as características faciais das pessoas e armazená-las em um arquivo no formato “.pickle”. Ele utiliza um método para detectar os rostos nas imagens de entrada. Esse procedimento é realizado através da biblioteca dlib, que emprega um detector de rostos baseado em Redes Neurais Convolucionais (CNN), previamente treinado em um extenso conjunto de dados. Posteriormente, a função “face_recognition.face_encodings” é acionada para extrair atributos específicos da face, representando-os em um espaço vetorial numérico. Essa representação numérica é então armazenada para análises posteriores.
- **modelo_deteccao:** é a pasta que contém o modelo pré-treinado "shape_predictor_81_face_landmarks.dat", o qual desempenha um papel crucial na detecção facial através da câmera. Esse modelo possui a capacidade de identificar e localizar características específicas do rosto, possibilitando uma detecção mais precisa e eficiente.
- **output:** é a pasta armazenamento do arquivo resultante da extração das características faciais das pessoas, nomeado como "face_decodificada.pickle". Este arquivo contém as informações essenciais que foram obtidas durante o processo de análise das faces.
- **Reconhecimento_facial_Cam_IP.py:** é a aplicação responsável pela identificação facial das pessoas utilizando uma câmera IP, onde identifica e permite o acesso de pessoas específicas.
- **Reconhecimento_facial_Cam_IP_Bloqueio.py:** semelhante ao programa anterior, essa aplicação também é responsável pela identificação de pessoas utilizando câmera IP, mas focado principalmente no bloqueio de pessoas sem permissão de entrada.

- **Reconhecimento_facial_WebCam.py:** criado para realizar o reconhecimento facial utilizando uma webcam. Sua lógica de funcionamento é semelhante às outras aplicações de reconhecimento facial, com a diferença de que utiliza uma webcam para dá a permissão de acesso as pessoas.
- **Reconhecimento_facial_WebCam_Bloqueio.py:** aplicação criada para fazer o bloqueio de pessoas usando uma webcam, a identificação será feita através desta câmera, onde identifica pessoas do grupo de bloqueados.

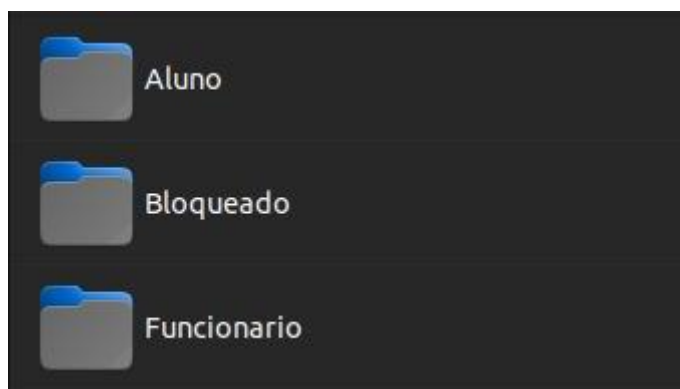
Todos os programas de identificação foram configurados pra utilizar o arquivo "face_decodificada.pickle" para realizar a identificação. Este arquivo contém as características faciais de todas as pessoas registradas no conjunto de dados (dataset), assim seus nomes e o grupo de pessoa. Ele opera comparando os rostos capturado pela câmera com os dados armazenados previamente nesse arquivo.

3.2.2 Construção Do Conjunto De Dados (Dataset)

Durante o desenvolvimento do sistema, a criação de um dataset personalizado foi uma das principais etapas. O dataset foi construído a partir da coleta de imagens de diversos indivíduos, incluindo variações de iluminação, expressões faciais e ângulos de visão. Cada imagem foi anotada manualmente para marcar pontos de referência faciais. Em seguida, aplicamos pré-processamento, redimensionamento e normalização das imagens, além de técnicas de aumento de dados para maior variabilidade.

A divisão do dataset em conjuntos de treinamento, validação e teste foi fundamental para o treinamento e avaliação dos modelos de reconhecimento facial. Esses conjuntos personalizados proporcionaram uma base sólida para o desenvolvimento e aprimoramento do sistema de reconhecimento facial, o qual desempenha um papel crucial do sistema de controle de acesso, podendo assim ser dividido em seguintes pastas, na Figura 08 são destinados a funcionalidade do dataset, logo foi dividido da seguinte forma:

Figura 08: Divisão das pastas



Fonte: Autores, 2023.

3.2.3 Treinamento De Máquina (Detector_Faces.Py)

Na etapa atual do projeto, foi implementada uma abordagem sistemática que faz uso de técnicas de Machine Learning, incluindo Redes Neurais Artificiais e Redes Neurais Profundas. Essa estratégia é crucial para o desenvolvimento do sistema de reconhecimento facial, desempenhando um papel fundamental no controle de acesso.

Para iniciar, como mencionado anteriormente, foi constituído um conjunto de dados personalizado, devidamente anotado. Ao longo do processo de treinamento, a inteligência artificial foi capaz de adquirir a habilidade de identificar traços faciais distintivos. Esses dados são posteriormente empregados na autenticação dos indivíduos, sendo armazenados no arquivo "detector_faces.py"².

Para o treinamento foi necessário fazer o carregamento de modelos já pré-treinados que são disponíveis das seguintes bibliotecas do Dlib e Face Recognition, assim conforme a Figura 09.

² Link Github:

https://github.com/BrenoKleverson0320/SCAF/blob/main/app_sistema_seguranca/back_end/detector_faces.py

Figura 09: aplicação dos modelos treinados.

```
# Carregue o modelo shape_predictor_81_face_landmarks.dat
predictor = dlib.shape_predictor('modelo_deteccao/shape_predictor_81_face_landmarks.dat')

# Inicialização do detector facial e preditores
face_detector = dlib.get_frontal_face_detector()

predictor_68_point_model = face_recognition_models.pose_predictor_model_location()
pose_predictor_68_point = dlib.shape_predictor(predictor_68_point_model)

predictor_5_point_model = face_recognition_models.pose_predictor_five_point_model_location()
pose_predictor_5_point = dlib.shape_predictor(predictor_5_point_model)

cnn_face_detection_model = face_recognition_models.cnn_face_detector_model_location()
cnn_face_detector = dlib.cnn_face_detection_model_v1(cnn_face_detection_model)

face_recognition_model = face_recognition_models.face_recognition_model_location()
face_encoder = dlib.face_recognition_model_v1(face_recognition_model)
```

Fonte: Autores, 2023.

- **shape_predictor:** Prevê os 81 pontos de referência da face, permitindo a detecção de características faciais.
- **get_frontal_face_detector:** Um detector de faces frontal que utiliza o modelo HOG (Histogram of Oriented Gradients).
- **pose_predictor:** Prevê a pose da face como orientação com base em 68 ou 5 pontos de referência.
- **cnn_face_detector:** Um detector de faces que utiliza uma rede neural convolucional.
- **face_encoder:** Um modelo de reconhecimento facial que codifica as características da face em um vetor numérico.

Após carregar os modelos pré-treinados e utilizadores de detectores facial e preditores, dá início a função “load_images_from_folder”. Essa função é responsável por carregar imagens de uma pasta específica onde está localizado o dataset, extrair codificações faciais e nomes associados. É chamada a função “face_recognition.face_encodings” para obter as codificações faciais ilustrado na Figura 10.

Figura 10: Condições faciais.

```

# Função para carregar as imagens de uma pasta e retornar as codificações faciais e nomes
def load_images_from_folder(folder, model='large', num_jitters=50, progress_callback=None):
    face_encodings = []
    face_names = []
    total_files = len(os.listdir(folder))
    unrecognized_photos = [] # Lista para armazenar fotos não reconhecidas
    scanned_photos = 0 # Contador para fotos escaneadas

    for i, filename in enumerate(os.listdir(folder)):
        img_path = os.path.join(folder, filename)
        image = face_recognition.load_image_file(img_path)
        encoding = face_recognition.face_encodings(image, num_jitters=num_jitters, model=model)
        if encoding:
            face_encodings.append(encoding[0])
            file_name, file_extension = os.path.splitext(filename)
            face_names.append(f'{folder}: {file_name}')
        else:
            unrecognized_photos.append(filename)

        if progress_callback:
            progress_callback(i + 1, total_files)

        scanned_photos += 1

    return face_encodings, face_names, unrecognized_photos, scanned_photos

```

Fonte: Autores, 2023.

Na Figura 10 o código está carregando as imagens da pasta dataset e em seguida retorna as codificações faciais dos usuários e os nomes cadastrados, para exemplificar a função que está fazendo será demonstrado no quadro 01.

Quadro 01: codificações faciais

```
encoding = face_recognition.face_encodings(image, num_jitters=num_jitters, model=model)
```

- **encoding:** uma lista de codificações faciais correspondentes às faces presentes na imagem. Essas codificações serão usadas para reconhecimento facial e comparação de faces.
- **image:** A imagem carregada da pasta.
- **num_jitters:** O número de iterações para aplicar pequenas variações nas imagens (rotações, deslocamentos, variações na iluminação) para melhorar a precisão. É uma forma de data augmentation.
- **model:** O modelo a ser usado para extrair as características faciais foi o “large”, onde resultou em uma representação mais abrangente e detalhada das características faciais, o que faz ter uma maior precisão na codificação e no reconhecimento de rostos.

Este código escaneia todos os arquivos no diretório do conjunto de dados, registrando as características faciais, os nomes dos arquivos de imagem e os seus respectivos locais de armazenamento. Posteriormente, essas informações são consolidadas em um arquivo “.pickle”, o qual realiza a serialização dos objetos, convertendo em bytes assim demonstrado na Figura 11.

- **Output_folder:** define o caminho do arquivo pickle usando a variável “pickle_file_path”. O arquivo será chamado de “face_decodificada.pickle” e será salvo na pasta chamada output.
- **Data_to_save:** prepara os dados a serem salvos em um dicionário, que contém duas chaves: “all_face_encodings” e “all_face_names”. Essas chaves correspondem às listas “all_face_encodings” e “all_face_names” que foram geradas ao carregar as imagens.
- **A declaração with:** usada para abrir o arquivo em modo de escrita binária (wb). Isso é feito para garantir que o arquivo seja tratado corretamente em sistemas que usam diferentes convenções de newline.
- **Pickle.dump:** esta função serve para escrever o dicionário “data_to_save” no arquivo pickle.

Portanto, nesta parte do código, os dados são codificações faciais e os nomes associados são armazenados no arquivo pickle chamado “face_decodificada.pickle”³ na pasta output.

3.2.4 Reconhecimento Facial Webcam

Neste tópico, serão abordados os testes realizados por meio do treinamento com o uso da webcam. Ao iniciar o projeto, é crucial configurar cuidadosamente a quantidade de recursos de hardware que os códigos irão consumir. Isso é essencial, uma vez que as taxas de FPS (Frames por Segundo), de acordo com as configurações escolhidas, podem impactar significativamente o desempenho da identificação facial, podendo resultar em uma diminuição da velocidade do processo.

O script “Reconhecimento_facial_WebCam.py”⁴ é responsável por configurar as taxas de processo de acordo com o computador em uso, pois o hardware do mesmo influencia no desempenho dos processos. A Figura 13 ilustra essa configuração, e essa funcionalidade contribui para a usabilidade, eficiência e adaptabilidade do sistema de reconhecimento facial, tornando-o mais robusto e adequado a diferentes cenários de uso.

³ Link GitHub:

https://github.com/BrenoKleverson0320/SCAF/tree/main/app_sistema_seguranca/back_end/output

⁴ Link GitHub:

https://github.com/BrenoKleverson0320/SCAF/blob/main/app_sistema_seguranca/back_end/Reconhecimento_facial_WebCam.py

Figura 13: Controle de usabilidade, eficiência e adaptabilidade do sistema.

```
# Variável global para controlar o modo de exibição
fullscreen = False
num_processes = 11 # Número de processos desejado

# Variáveis para o cálculo do FPS
frame_count = 0
start_time = time.time()
fps_text = "FPS: 0.00" # Inicialmente, o valor do FPS é zero

# Variáveis para controlar o número de reconhecimentos consecutivos
consecutive_recognitions = 0
min_recognitions_threshold = 20
```

Fonte: Autores, 2023

- **Fullscreen (boolean):** Controla o modo de exibição do programa, indicando se está em tela cheia (True) ou não (False).
- **Num_processes (int):** Representa o número desejado de processos.
- **frame_count (int):** Conta o número de frames processados para calcular o FPS.
- **start_time (float):** Armazena o tempo inicial do programa para calcular o FPS.
- **fps_text (str):** Armazena o texto que exibe o valor atual do FPS. Inicialmente, é configurado como "FPS: 0.00".
- **consecutive_recognitions (int):** Controla o número de reconhecimentos consecutivos. Este contador é usado para tomar decisões com base em uma quantidade mínima de reconhecimentos, conforme indicado pela variável “min_recognitions_threshold”.

Agora, será necessário carregar o arquivo pickle, pois o script realizará a comparação entre a base de dados treinada e as imagens provenientes da webcam. Essa comparação é essencial para a identificação do usuário, proporcionando ao sistema a capacidade de correlacionar as características faciais capturadas em tempo real com as representações armazenadas no conjunto de dados treinado, mostrado na Figura 14.

Figura 14: Carregando as informações do pickle.

```
# Função para carregar as informações do arquivo pickle
def load_face_encodings(pickle_file_path):
    try:
        with open(pickle_file_path, 'rb') as file:
            data_loaded = pickle.load(file)
            return data_loaded
    except FileNotFoundError:
        return None
```

Fonte: Autores, 2023

Função de Carregamento “load_face_encodings”, a função recebe o caminho para um arquivo pickle como argumento “pickle_file_path” e tenta abrir o arquivo em modo de leitura binária “rb”. Se o arquivo for encontrado, usa o módulo pickle para carregar os dados contidos no arquivo. Retorna os dados carregados. Se o arquivo não for encontrado, retorna “None”. Essa função é útil para carregar as codificações faciais previamente salvas em um arquivo pickle, permitindo a reutilização desses dados no reconhecimento facial. A exceção “FileNotFoundError” é tratada para lidar com a situação em que o arquivo pickle não existe. Em seguida a função, “process_frame”, é responsável por processar um quadro de vídeo, realizando as seguintes operações na Figura 15:

Figura 15: Processando dados do arquivo pickle.

```
# Função para processar um quadro
def process_frame(frame):
    global consecutive_recognitions # Adicione esta linha

    face_locations = face_recognition.face_locations(frame)
    face_encodings = face_recognition.face_encodings(frame, face_locations)
    vertical_positions = []

    for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
        vertical_position = (top + bottom) // 2
        vertical_positions.append(vertical_position)
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)

    if vertical_positions:
        average_vertical_position = sum(vertical_positions) // len(vertical_positions)
    else:
        average_vertical_position = 100

    threshold = 0.47
```

Fonte: Autores, 2023.

- **Deteção Facial:** Usa a biblioteca “face_recognition” para localizar as faces no quadro “face_recognition.face_locations”. Calcula as codificações faciais para as faces detectadas “face_recognition.face_encodings”.
- **Desenho de Retângulos (cv2.rectangle):** Para cada face detectada, desenha um retângulo ao redor dela no quadro.
- **Cálculo da posição vertical média:** funções “sum” e “len” calcula a posição vertical média das faces detectadas.
- **Reconhecimento Facial:** as funções “face_recognition.face_distance”, “os.path.basename” e “min” itera sobre as faces detectadas e calcula as distâncias entre as codificações faciais e as codificações previamente carregadas “all_face_encodings”. Determina se uma face é reconhecida com base em uma determinada margem de similaridade “threshold”. Se reconhecida, obtém o nome correspondente e a porcentagem de similaridade. Se não reconhecida, atribui o nome "Desconhecido".

- **Exibição de Informações na Imagem:** as funções “cv2.putText” e “cv2.rectangle” exibe o nome da pessoa ou "Desconhecido" sobre o retângulo da face. Adapta a cor do texto com base na média de cor da região da face.

Ao estabelecer as condições para a interpretação da visão computacional durante o processo de identificação facial, incluímos as considerações referentes às dimensões e configurações da abertura da webcam, como ilustrado na Figura 16.

Figura 16: configurando as dimensões dos frames.

```
# Função para exibir o FPS
def display_fps(frame):
    global frame_count, start_time, fps_text
    frame_count += 1

    if time.time() - start_time >= 1:
        fps = frame_count / (time.time() - start_time)
        frame_count = 0
        start_time = time.time()
        fps_text = f"FPS: {fps:.2f}"

    cv2.putText(frame, fps_text, (10, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)

if __name__ == "__main__":
    pickle_file_path = 'output/face_decodificada.pickle'
    loaded_data = load_face_encodings(pickle_file_path)

    if loaded_data:
        all_face_encodings = loaded_data.get('all_face_encodings', [])
        all_face_names = loaded_data.get('all_face_names', [])

        video_capture = cv2.VideoCapture(0)
        desired_width = 640
        desired_height = 480
        desired_framerate = 30
        video_capture.set(3, desired_width)
        video_capture.set(4, desired_height)
        video_capture.set(5, desired_framerate)
```

Fonte: Autor, 2023.

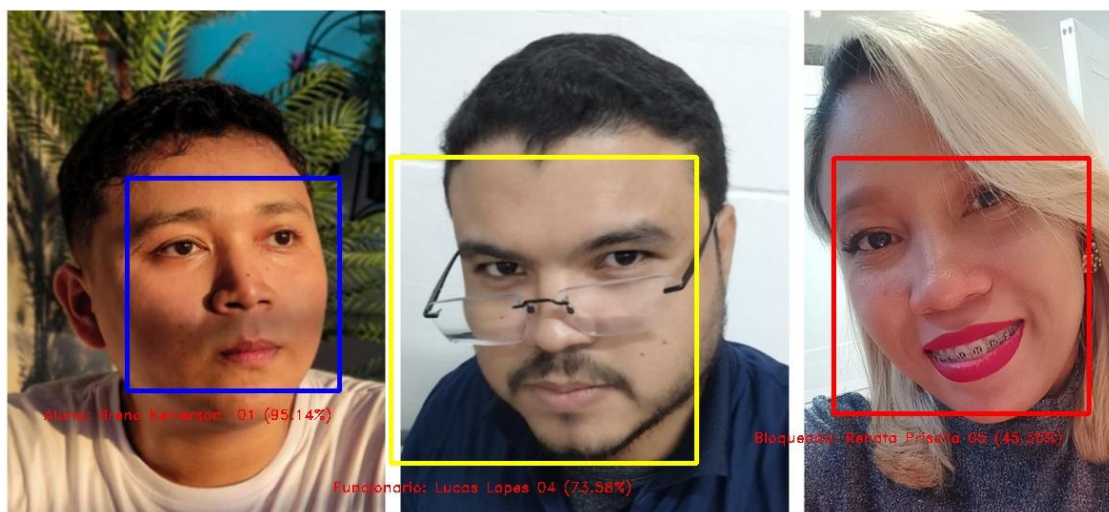
Essa parte do código está relacionada ao controle do FPS e à configuração inicial da captura de vídeo para reconhecimento facial.

- **display_fps Function:** A função atualiza e exibe a taxa de quadros por segundo (FPS) no canto superior esquerdo do quadro e monitora o número de quadros processados e calcula o FPS a cada segundo.
- **__main__ Block:** Carrega as informações previamente codificadas de faces (encodings) a partir de um arquivo pickle. Inicializa a captura de vídeo da webcam usando OpenCV, configurando largura, altura e taxa de quadros desejados. Configura a janela do OpenCV para exibir o reconhecimento facial em modo normal ou tela cheia.
- **Loop Principal:** Dentro do loop infinito ele captura um quadro da webcam e chama a função “display_fps” para exibir a taxa de quadros naquele momento, processa o quadro utilizando a função “process_frame” e exibe o quadro processado na janela do OpenCV

para reconhecimento facial, permitindo a alternância entre tela cheia e modo normal ao pressionar “F” ou “f”. O loop continua até que a tecla “q” seja pressionada.

O propósito fundamental do código é integrar uma câmera webcam para efetuar o reconhecimento facial de uma pessoa, indicando se ela é um funcionário, está bloqueada ou é desconhecida. Os primeiros testes serão realizados para validar essa funcionalidade. Antes de prosseguir, é essencial compreender o processo pelo qual a identificação é conduzida sobre o resultado final da execução do código mostrado na figura 17.

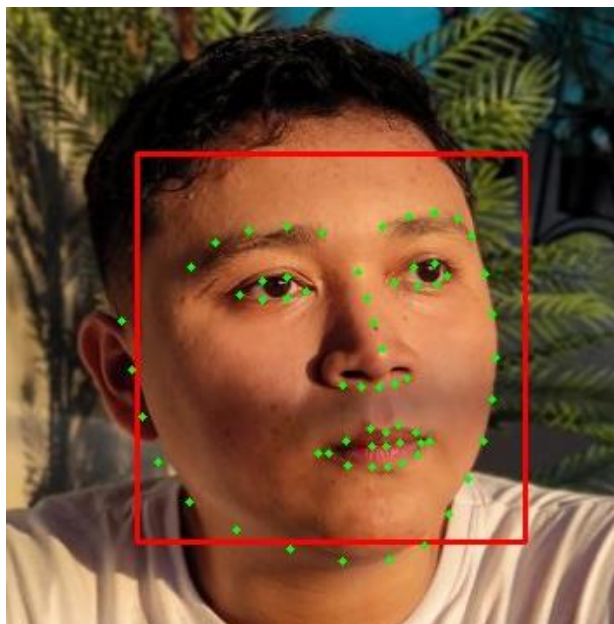
Figura 17: Reconhecendo faces.



Fonte: Autores, 2023.

O código foi executado com sucesso, analisando eficazmente o arquivo de treinamento gerado a partir das pastas do dataset. Esse êxito formaliza o teste, indicando que o sistema de reconhecimento facial está operacional e apto para prosseguir em estágios mais avançados de desenvolvimento. Na Figura 18 fornecerá uma demonstração visual do processo que determina os pontos faciais e leva em consideração as dimensões do rosto, baseando-se no quadrado formado pelos eixos X e Y. Este método constitui a base essencial para a identificação facial eficiente dentro do contexto do projeto.

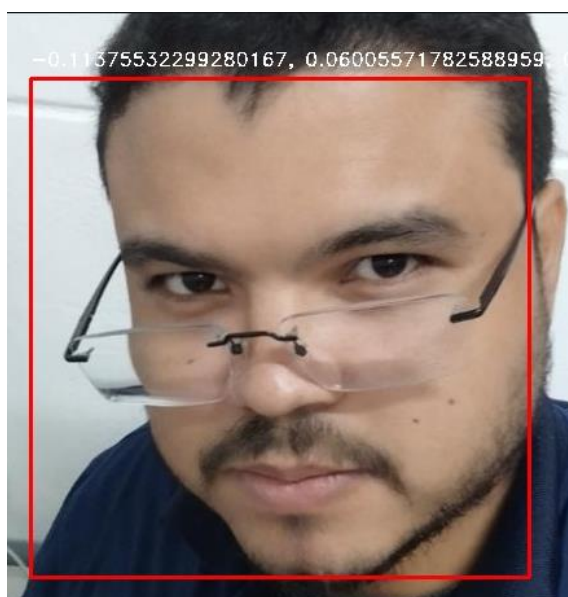
Figura 18: Dimensões do rosto.



Fonte: Autores, 2023.

Na Figura 18, as embeddings (o processo de criação de vetores usando aprendizado profundo) interpretam o rosto humano como um conjunto de características únicas e as representam em um espaço vetorial. Essas embeddings são, essencialmente, uma representação numérica das características faciais, sendo especialmente úteis para diferenciar um rosto dos demais, assim mostrado na Figura 19. Este processo é fundamental para a capacidade do sistema em realizar distinções precisas entre diferentes rostos, contribuindo assim para uma identificação facial eficaz.

Figura 19: Representação numéricas das características faciais.



Fonte: Autores, 2023.

Portanto, ao demonstrar que o projeto de treinamento e identificação facial por meio de um dataset está operacional, fica claro que futuramente ele será substituído por um banco de dados mais robusto. Essa evolução é uma etapa essencial para aprimorar a eficácia e a escalabilidade do sistema de reconhecimento facial.

3.2.5 Abrindo Câmera IP Com Protocolo RTSP

Neste tópico, a implementação da câmera IP no projeto será abordada. Após os testes bem-sucedidos com a webcam, a adição da câmera IP requer algumas atribuições adicionais para integrá-la à lógica do script e ao projeto. Seguindo a mesma lógica aplicada à webcam, serão realizadas as modificações necessárias para assegurar a efetiva incorporação da câmera IP ao sistema.

Para implementar a conexão com uma câmera IP via RTSP em Python, é necessário seguir um procedimento sistemático. Primeiro, é fundamental preparar a câmera IP, assegurando-se de que esteja configurada e conectada à rede, com um endereço IP atribuído corretamente. Em seguida, foi utilizada uma biblioteca adequada para lidar com a comunicação RTSP, sendo a biblioteca OpenCV frequentemente empregada devido à sua versatilidade, requerendo sua instalação prévia.

O código Python constitui a parte central da conexão. Nele, foi definida a URL RTSP da câmera IP no formato "rtsp://endereco_da_camera:porta_rtsp/stream". Por meio da biblioteca OpenCV, inicializa-se um objeto "VideoCapture" com essa URL, estabelecendo a conexão, conforme ilustrado na Figura 20.

Figura 20: Conexão RTSP.

```
#parametros de acesso da camera
USERNAME = 'admin'
PASSWORD = 'SENHATESTE'
IP = '192.168.1.181'
PORT = '554'

#so roda se for ffmpeg
os.environ["OPENCV_FFMPEG_CAPTURE_OPTIONS"] = "rtsp_transport;udp"

#url da camera stream varia conforme o modelo da camera
URL = 'rtsp://{}:{@}:{}/onvif1'.format(USERNAME, PASSWORD, IP, PORT)
print('Conectando com: ' + URL)

cap = cv2.VideoCapture(URL, cv2.CAP_FFMPEG)
```

Fonte: Autores, 2023.

Com esse código é possível abrir a câmera IP, o loop principal do código Python é responsável por capturar os quadros de vídeo em tempo real da câmera e, opcionalmente, realizar qualquer processamento necessário, como exibição ou armazenamento. Essa etapa é crucial para a continuidade da operação eficiente do sistema.

No script "Reconhecimento_facial_Cam_IP.py⁵", é essencial incorporar as mesmas lógicas que foram adicionadas para a câmera webcam. Isso inclui o controle das taxas de FPS, o gerenciamento do processamento, a inclusão do arquivo pickle, a reprodução e o controle das interpretações desses dados. Dessa maneira, torna-se possível realizar identificações utilizando a câmera IP.

O projeto está identificando, as implementações e estratégias foram um sucesso, corroborando a viabilidade e o sucesso do projeto. Este avanço representa um passo significativo na aplicação prática da tecnologia de reconhecimento facial utilizando câmeras IP.

3.2.6 Protótipo Fechadura Eletrônica Com Arduino

Neste capítulo, será explorada toda a estrutura física do o projeto, que consiste em um protótipo projetado com o objetivo de conceder acesso a pessoas autorizadas, ao mesmo tempo em que impede o acesso de pessoas não autorizadas ou desconhecidas. Integramos um mecanismo sonoro como medida preventiva para reforçar a segurança das pessoas bloqueadas pelo sistema. Para implementar esse sistema, utilizamos um Arduino Uno, um relé de 5 volts, uma tranca solenoide de 12 volts, duas câmeras IP e webcam, conforme ilustrado na Figura 21.

⁵ Link GitHub:

https://github.com/BrenoKleverson0320/SCAF/blob/main/app_sistema__seguranca/back_end/Reconhecimento_facial_Cam_IP.py

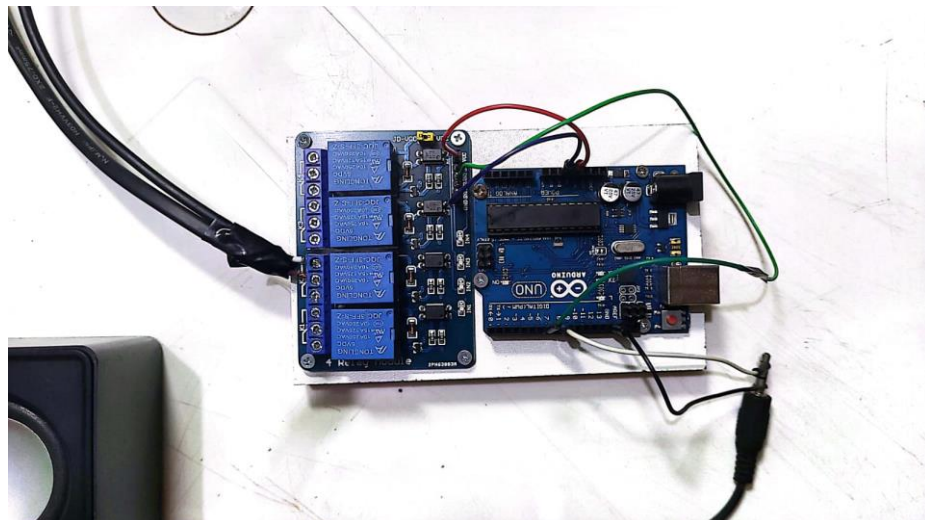
Figura 21: equipamentos usados.



Fonte: Autores, 2023.

O dispositivo Arduino foi conectado de forma direta ao servidor utilizando uma conexão USB para estabelecer a comunicação serial. Os comandos destinados ao dispositivo serão direcionados de maneira direta ao relé e ao alto-falante, utilizando os pinos digitais 7 e 8 do Arduino, respectivamente, incluindo os pinos GND necessários. O alto-falante será submetido a uma frequência de 500 Hz, projetada para gerar um sinal sonoro como resposta à detecção de bloqueio. Ao mesmo tempo, o relé receberá uma alimentação de 5V diretamente do Arduino para sua ativação, desempenhando a função de um interruptor para possibilitar a transmissão de 12V necessários para acionar a trava elétrica, demonstrado na Figura 22.

Figura 22: Montagem do Arduino Uno.



Fonte: Autores, 2023.

A lógica do projeto funcionará da seguinte forma: o código utilizado será em C++ e Python, abordados nas respectivas formas no arquivo "codigo_arduino.ino" na pasta "codigo_arduino". Esse arquivo será responsável pelo controle da tranca solenoide, mais especificamente pelo módulo relé, por meio dos comandos recebidos pelas portas seriais. A parte lógica em Python será encontrada nos códigos "Reconhecimento_facial_webcam.py" e "Reconhecimento_facial_cam_ip.py". Nestes scripts, os dados serão enviados para o C++, que os interpretará para efetuar o controle da tranca solenoide. Essa abordagem visa estabelecer uma comunicação eficiente entre as linguagens de programação, garantindo a integração funcional do sistema.

No script “codigo_arduino.ino” conforme a lógica foi explicada ele funcionara da seguinte forma mostrado no quadro 2.

Quatro 2:Codigo para tranca eletrônica.

```

1.  const int RelPin = 7;
2.  const int BuzzerPin = 8;
3.  int incomingByte;
4.  bool relayState = HIGH;
5.  bool soundOn = false;
6.  unsigned long lastRelayChangeTime = 0;
7.  unsigned long lastCommandTime = 0;
8.  void setup() {
9.    Serial.begin(9600);
10.   pinMode(RelPin, OUTPUT);
11.   pinMode(BuzzerPin, OUTPUT);
12.   digitalWrite(RelPin, relayState);
13.   digitalWrite(BuzzerPin, HIGH);
14. }
15. void loop() {
16.   if (Serial.available() > 0) {
17.     incomingByte = Serial.read();
18.     if (incomingByte == '3') {
19.       relayState = HIGH;
20.       digitalWrite(RelPin, relayState);
21.     }
22.     else if (incomingByte == '1' && millis() - lastCommandTime >= 6000) {
23.       // Se o comando '1' for recebido e passaram-se pelo menos 6 segundos
24.       delay(5000);
25.       relayState = LOW;
26.       digitalWrite(RelPin, relayState);
27.       lastRelayChangeTime = millis();
28.       lastCommandTime = millis();
29.       //soundOn = false;
30.     } else if (incomingByte == '0') {
31.       relayState = !relayState;
32.       digitalWrite(RelPin, relayState);
33.       //relayState = LOW;
34.       soundOn = false;
35.     } }
36.   if (incomingByte == '2') {

```

```

37. relayState = LOW;
38. digitalWrite(RelePin, relayState);
39. soundOn = true;
40. }if (soundOn) {
41. tone(BuzzerPin, 500);
42. } else {
43. noTone(BuzzerPin); }
44. if (relayState == LOW && millis() - lastRelayChangeTime >= 3000) {
45. relayState = HIGH;
46. digitalWrite(RelePin, relayState); }}

```

Fonte: Autores, 2023.

No quadro 2 apresenta um sistema de controle através da comunicação serial, utilizando dois componentes principais: um módulo de relé e uma caixa de som. No início da linha de 1 a 7, são definidas constantes para os pinos do relé e a caixa de som, além de variáveis para armazenar o estado do relé, a ativação do som e informações temporais cruciais.

A função “setup()” linha 8 é responsável pela configuração inicial, estabelecendo a comunicação serial, definindo os pinos como saídas e atribuindo estados iniciais ao relé e ao alarme. A comunicação serial é iniciada a uma taxa de 9600 bps (bit).

O loop principal “loop()” na linha 15 monitora constantemente a disponibilidade de dados na porta serial. Quando um byte está disponível, o código lê o caractere recebido e executa a lógica correspondente. Por exemplo, o comando “0” inverte o estado do relé, enquanto o comando “3” desativa imediatamente o relé.

O código também gerencia a ativação do som, onde o comando “2” desativa o relé e ativa a caixa de som. A lógica de temporização é crucial, como evidenciado pelo comando “1”. Se o comando “1” é recebido e passaram-se pelo menos 6 segundos desde o último comando “1”, o relé é ativado por um curto período, esta lógica foi adicionada para dimensionar a quantidade de tempo que faz com que fique na frente da câmera para poder ativar a trava caso seja conhecido.

Além disso, o código controla o relé para evitar situações indesejadas. Após 3 segundos de inatividade, o relé é reativado automaticamente, garantindo uma operação segura e eficiente. Essa abordagem cuidadosa no controle do sistema torna o código adaptável a diferentes cenários de aplicação.

De agora em diante, será apresentada a lógica aplicada no Python nos scripts "Reconhecimento_facial_webcam.py" e "Reconhecimento_facial_cam_ip.py", uma vez que ambos compartilham a mesma lógica para seu funcionamento, conforme detalhado na Figura 23.

Figura 23: Interligando códigos com Arduino.

```
# Variáveis para controlar o número de reconhecimentos consecutivos
consecutive_recognitions = 0
min_recognitions_threshold = 20

# Encontre a porta do Arduino automaticamente
def find_arduino_port():
    arduino_ports = [port.device for port in list_ports.comports() if 'ttyACM0' in port.description]
    if arduino_ports:
        return arduino_ports[0]
    return None

# Encontra a porta do Arduino
arduino_port = find_arduino_port()

if arduino_port:
    arduino = serial.Serial(arduino_port, 9600)
```

Fonte: Autores, 2023.

Função “find_arduino_port()” busca automaticamente a porta do Arduino com base na descrição encontrada nos dispositivos conectados. Caso a porta do Arduino seja encontrada, o código estabelece uma comunicação serial utilizando a biblioteca serial.

Na função “process_frame(frame)”, após a detecção de rostos utilizando a biblioteca face_recognition, há um bloco condicional que envia comandos para o Arduino com base no reconhecimento facial. Dependendo do nome associado ao rosto reconhecido (por exemplo, "Aluno", "Funcionario", "Bloqueado" ou "Desconhecido"), o código envia diferentes comandos (1 ou 3) para o Arduino, controlando assim o módulo de relé.

A função “process_frame” interage diretamente com o código Arduino assim mostrado na Figura 24, controlando o relé com base no reconhecimento facial. Isso demonstra a integração entre os ambientes Python e Arduino.

Figura 24: Interação dos códigos.

```
if "Aluno" in name:
    border_color = (255, 0, 0)
    consecutive_recognitions += 1
    if consecutive_recognitions >= min_recognitions_threshold:
        arduino.write(b'1')
elif "Funcionario" in name:
    border_color = (0, 255, 255)
    consecutive_recognitions += 1
    if consecutive_recognitions >= min_recognitions_threshold:
        arduino.write(b'1')
elif "Bloqueado" in name:
    border_color = (0, 0, 255)
    arduino.write(b'3') # Envia imediatamente o número 3 para o Arduino
    consecutive_recognitions = 0 # Reinicia o contador de reconhecimentos consecutivos
else:
    border_color = (0, 255, 0)
    arduino.write(b'3') # Envia imediatamente o número 3 para o Arduino
    consecutive_recognitions = 0 # Reinicia o contador de reconhecimentos consecutivos
```

Fonte: Autores, 2023.

Dentro do bloco "try", no loop de processamento de frames, quando um rosto é identificado como "Aluno" ou "Funcionario", é enviado o comando “1” para o arduino

utilizando a função “`arduino.write(b'1')`”. Além disso, o código incrementa o contador “`consecutive_recognitions`”. Se o número de reconhecimentos consecutivos atinge ou ultrapassa o limite definido em “`min_recognitions_threshold`”, o código envia um comando para o Arduino através de “`arduino.write`”. Se o nome contiver “Aluno” ou “Funcionario”, envia “`b'1`”. Se o nome contiver “Bloqueado”, envia “`b'3`”. Em casos em que o nome reconhecido contenha “Bloqueado” ou não corresponda a nenhum padrão conhecido, o código reinicia o contador de reconhecimentos consecutivos.

Por fim, a conexão com o protótipo para a identificação facial foi realizada com sucesso. Após as configurações adequadas, é possível executar o código Python.

3.3 FRONT-END

Para o desenvolvimento do front-end⁶ foi escolhido a framework Django, devido sua produtividade e escalabilidade para desenvolvimento web. A arquitetura utilizada foi a MTV (Model-Template-View), onde se divide em Model, View e Template. Os modelos, na arquitetura identificado como models, representam os dados, onde interagem com o banco de dados. As views irão processar as solicitações e preparar os dados para a exibição nos templates, os quais são arquivos HTML que moldam a estrutura visual. O sistema direciona as solicitações de URL para funções de view correspondentes. Este fluxo permite que dados sejam recuperados, processados e exibidos de forma dinâmica em páginas web.

3.3.1 Tela De Login

A tela de login foi estruturada para permitir que os usuários acessem o sistema com segurança através de um usuário e senha, mostrado na Figura 25. Foi utilizado formulários personalizados e técnicas de validação fornecidas pelo Django, para garantir a autenticação adequada dos usuários, tornando a interface intuitiva e funcional, o primeiro usuário foi cadastrado previamente diretamente no banco de dados utilizado no Django SQLite.

⁶ Link GitHub: <https://github.com/BrenoKleverson0320/SCAF/tree/main>

Figura 25: Tela de Login



A mockup of a login screen. At the top, it says "Bem-vindo de volta!". Below this are two empty, rounded rectangular input fields for username and password. Under the password field is a blue rounded rectangular button with the text "Login". At the bottom, there are two links: "Esqueceu a Senha?" and "Criar uma Conta!".

Fonte: Autores, 2023.

A técnica de validação foi desenvolvida na views⁷, onde está conectado ao “form” do HTML através do `{% csrf_token %}`, onde irá receber os dados do usuário e fazer a validação, como mostrado na Figura 26.

Figura 26: Técnicas de Validação

```
def login_view(request):
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            user = authenticate(request, username=username, password=password)
            if user is not None:
                login(request, user)
                return redirect('index')
        else:
            form = LoginForm()
            return render(request, 'front-end/login.html', {'form': form})

def index(request):
    return render(request, 'front-end/index.html')
```

Fonte: Autores, 2023.

⁷ Link GitHub: https://github.com/BrenoKleverson0320/SCAF/blob/main/app_sistema__seguranca/views.py

3.3.2 Tela Home

A tela inicial atua como o ponto central do sistema, ela fornecendo uma visão panorâmica de todas as funcionalidades disponíveis no sistema. Ela facilita a navegação para diferentes seções do sistema, oferecendo uma experiência amigável ao usuário.

Na tela inicial existe duas principais funções, Cadastros e Monitoramento. Na parte de monitoramento existe o modo Proteção e o Controle de Acesso, como mostrado na Figura 27, O modo Proteção irá iniciar o reconhecimento facial e impedir a entrada de pessoas bloqueadas, O modo controle de acesso irá permitir a entrada somente de pessoas liberadas, como Alunos e Funcionários.

Figura 27: Opção de Monitoramento



Fonte: Autores, 2023.

Na área de cadastros tem duas opções, o de Cadastrar Pessoas e de Bloquear, como mostrado na Figura 28, no caso de cadastrar pessoas, as pessoas cadastradas seriam Alunos e Funcionários. Já opção de bloquear é semelhante à de cadastrar pessoas, mas você irá bloquear o acesso de uma pessoa.

Figura 28: Opção de Cadastros



Fonte: Autores, 2023.

3.3.3 Tela De Cadastro

Para o desenvolvimento do cadastro de funcionários e alunos, mostrado na Figura 29, foi criado um formulário, onde coleta informações como nome, e-mail, endereço, número de

telefone e fotografias, essas fotografias são essenciais, pois são utilizadas para fazer o reconhecimento das pessoas através do sistema. Esses dados são armazenados de forma segura no banco de dados SQLite, utilizando a lógica de models e views do Django para garantir a integridade dos dados.

Figura 29: Tela de cadastro

A interface de cadastro, intitulada "Cadastrar!", apresenta um formulário com os seguintes campos e elementos:

- Campos de entrada para "Nome", "Endereço de Email" e "endereço".
- Campos para "Telefone" e um menu suspenso "Escolha o Tipo" com uma seta para baixo.
- Botão "Adicionar Fotos" em cinza.
- Botão principal "Cadastrar" em azul.
- Link "← Voltar ao Início" na base da interface.

Fonte: Autores, 2023

A construção da lógica do banco de dados, foi feita no arquivo models⁸, como mostrado na Figura 30, onde toda a lógica é construída e enviada para o SQLite através do comando “python manage.py makemigrations” e “python manage.py migrate”.

⁸ Link GitHub: https://github.com/BrenoKleverson0320/SCAF/blob/main/app_sistema_seguranca/models.py

Figura 30: Estrutura Banco de Dados

```
from django.db import models

class Aluno(models.Model):
    nome = models.CharField(max_length=100)
    email = models.EmailField()
    endereco = models.CharField(max_length=200)
    telefone = models.CharField(max_length=15)
    foto = models.ImageField(upload_to='Alunos/')

class Funcionario(models.Model):
    nome = models.CharField(max_length=100)
    email = models.EmailField()
    endereco = models.CharField(max_length=200)
    telefone = models.CharField(max_length=15)
    foto = models.ImageField(upload_to='Funcionarios/')
```

Fonte: Autores, 2023

Para receber os dados do formulário, foi construída uma função para receber todos os dados das pessoas e enviar para o banco de dados, como mostrado na Figura 31, essa função além de salvar as informações no banco de dados, também salva as fotografias no dataset, para serem usadas posteriormente, e modifica o nome das fotografias para ficar igual ao nome da pessoa cadastrada.

Figura 31: Função Cadastro

```

def cadastros(request):
    if request.method == 'POST':
        nome = request.POST['nome']
        email = request.POST['email']
        endereco = request.POST['endereco']
        telefone = request.POST['telefone']
        pessoa_type = request.POST.get('pessoa_type') # '1' para aluno, '2' para funcionário

        # Verificar se o campo 'foto' é um único arquivo ou vários arquivos
        fotos = request.FILES.getlist('foto')

        if pessoa_type == '1':
            folder = 'app_sistema_seguranca/back_end/dataset/Aluno'
        elif pessoa_type == '2':
            folder = 'app_sistema_seguranca/back_end/dataset/Funcionario'
        else:
            return redirect('index')

        for foto in fotos:
            i = 1 # Inicialize 'i' para evitar o erro
            filename = f"{nome}_{i}"

            while os.path.exists(os.path.join(folder, f"{filename}.jpg")):
                i += 1
                filename = f"{nome}_{i}"

            filename = f"{filename}.jpg"
            file_path = os.path.join(folder, filename)

            # Salvar o arquivo na pasta com o nome único
            with open(file_path, 'wb') as destination:
                for chunk in foto.chunks():
                    destination.write(chunk)

            # Salvar os dados do aluno ou funcionário no banco de dados
            if pessoa_type == '1':
                aluno = Aluno(nome=nome, email=email, endereco=endereco, telefone=telefone, foto=filename)
                aluno.save()
            elif pessoa_type == '2':
                funcionario = Funcionario(nome=nome, email=email, endereco=endereco, telefone=telefone, foto=filename)
                funcionario.save()

        return redirect('index')

    return render(request, 'front-end/cadastros.html')

```

Fonte: Autores, 2023

3.3.4 Tela Bloquear Pessoas

A tela de bloquear pessoas tem sua lógica semelhante ao de cadastrar pessoas, nela é possível registrar indivíduos que deseja bloquear, também existe um formulário onde pode ser adicionado detalhes como nome, e-mail, endereço, telefone, além de e um campo para fazer uma descrição completa da pessoa, como mostrado na Figura 32. Também é permitido anexar múltiplas imagens para identificação, essas imagens são salvas no banco de dados e na pasta Bloqueados que se encontra no dataset.

Figura 32: Tela Bloquear.

The image shows a web form titled "Bloquear Pessoa!". It contains the following elements:

- A text input field labeled "Nome".
- A text input field labeled "Endereço de Email".
- A text input field labeled "endereço".
- A text input field labeled "Telefone".
- A dark grey button labeled "Adicionar Fotos".
- A text area labeled "Descrição:" with placeholder text "Digite uma descrição...".
- A large red button labeled "Bloquear".
- A blue link at the bottom labeled "← Voltar ao Início".

Fonte: Autores, 2023

Toda a lógica de conexão da tela de bloqueio está na views, onde a função “bloquear” envia os dados para o banco de dados, além de modificar o nome das imagens para o nome da pessoa bloqueada, como mostrado na Figura 33.

Figura 33: Função Bloquear

```
def bloquear(request):
    if request.method == 'POST':
        nome = request.POST.get('nome')
        email = request.POST.get('email')
        endereco = request.POST.get('endereço')
        telefone = request.POST.get('telefone')
        descricao = request.POST.get('descricao')

        bloqueado = Bloqueado.objects.create(nome=nome, email=email, endereco=endereco, telefone=telefone, descricao=descricao)

        fotos = request.FILES.getlist('foto')
        for index, foto in enumerate(fotos):
            # Renomear a foto
            nome_foto = f'{bloqueado.nome}_{index + 1}.jpg'
            bloqueado_foto = FotoBloqueado.objects.create(bloqueado=bloqueado, foto=nome_foto)

            # Salvar a foto no diretório
            with open(f'app_sistema_seguranca/back_end/dataset/Bloqueado/{nome_foto}', 'wb+') as destination:
                for chunk in foto.chunks():
                    destination.write(chunk)

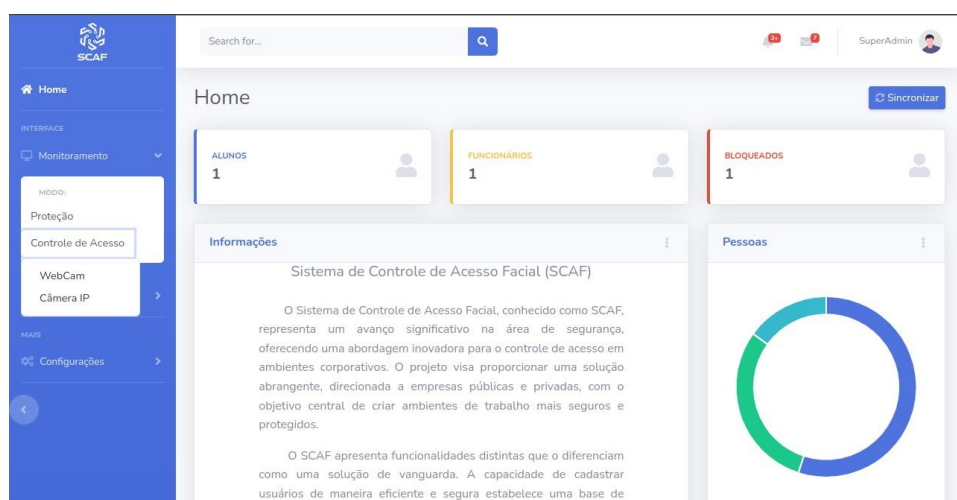
        return redirect('index') |
    return render(request, 'front-end/bloquear.html')
```

Fonte: Autores, 2023.

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Nesta etapa, serão abordados os resultados do desenvolvimento do protótipo, que envolveram a elaboração de hardware e o desenvolvimento de um sistema (software) denominado SCAF (Sistema de Controle de Acesso Facial), conforme mostrado na Figura 34. Sua elaboração demonstrou-se bem-sucedida, pois o treinamento conteve dados funcionais de modelos de redes neurais treinados, resultando em um treinamento de máquina ainda mais bem-sucedido, capaz de reconhecer indivíduos cadastrados no sistema.

Figura 34: tela principal do sistema

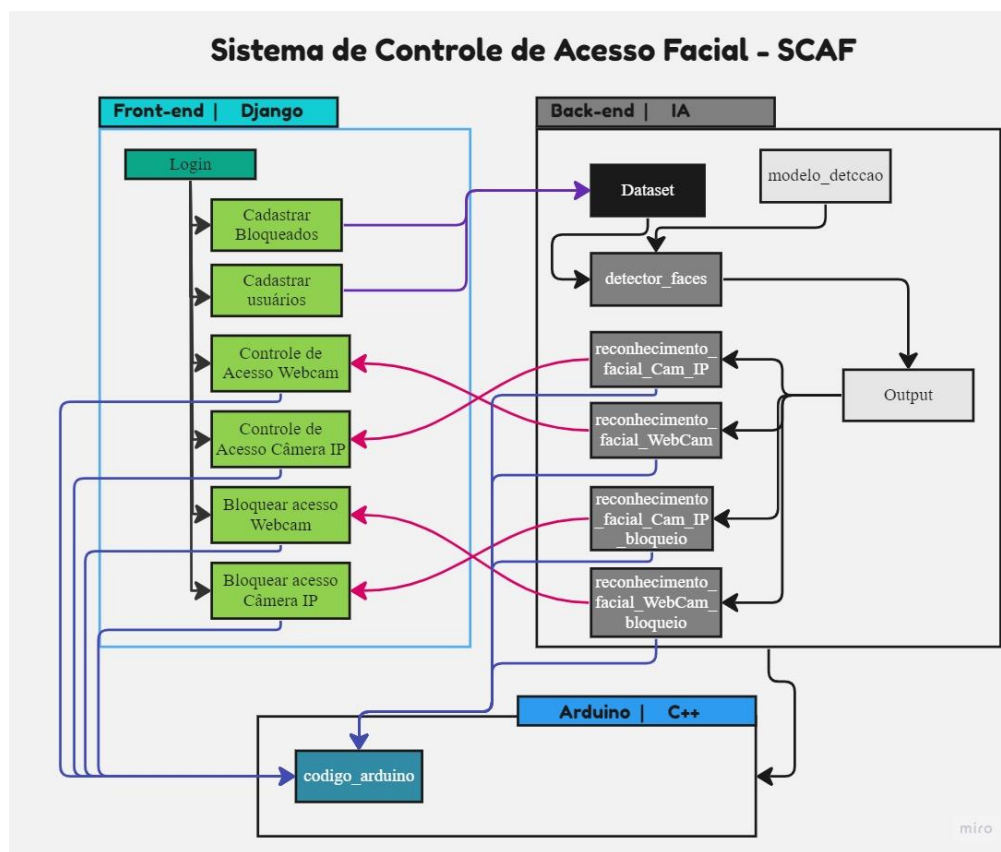


Fonte: Autores, 2023.

4.1 RESULTADOS DO SISTEMA

O Sistema de Controle de Acesso Facial, ao longo do projeto, foi dividido em duas partes: o front-end e o back-end. Sendo assim, ilustrados de maneira ainda mais dinâmica na Figura 35, é capaz de compreender o dinamismo do sistema.

Figura 35: Diagrama do sistema final.



Fonte: Autores, 2023.

Na Figura 35 acima, o diagrama representado é composto das funcionalidades do sistema. É possível notar que tanto o back-end quanto o front-end se comunicam. Após fazer o login, caso os administradores queiram cadastrar e bloquear usuários, basta seguir o esquemático. Após adicionar ou redirecionar em caso de bloqueios, as informações serão encaminhadas para o dataset, onde serão utilizadas no treinamento de faces. Após essa etapa, o sistema estará apto para realizar o reconhecimento facial através das câmeras.

Neste protótipo, foi necessário implementar dois métodos para a abertura de câmeras: um com a webcam e outro com a câmera IP. Essa abordagem se deve ao tempo de reconhecimento, sendo que a câmera IP demandou mais tempo. Em estudos analisados sobre o tema, o artigo "Segurança Câmera IP" (SILVA, 2018, p.39-40) destaca aspectos relevantes para a performance ao comparar câmeras IP e webcams. Fatores como a ausência de conexão cabeada, compressão de dados e perda de dados via wi-fi influenciam na eficiência da transmissão. A qualidade dos vídeos, determinada por resolução da imagem, taxa de quadros por segundo (FPS) e codec (H.264/H.265), desempenha um papel crucial. Silva ressalta a importância da largura de banda e do espaço de armazenamento, relacionados à qualidade da imagem, algoritmo de compressão e modo de gravação. Ambientes com alta movimentação

demandam mais recursos para processar e armazenar os vídeos. Em resumo, a escolha entre câmeras IP e webcams envolve considerações técnicas como resolução, codec e características do ambiente. Ambos os métodos são interligados com o Arduino, proporcionando o acesso aos indivíduos autorizados.

O sistema apresenta duas modalidades distintas de utilização. O primeiro modo é designado para o controle de acesso, ideal para locais que requerem restrição e permissão específica, garantindo a entrada apenas de pessoas autorizadas. Por outro lado, o segundo modo opera de maneira diferente, sendo mais adequado para ambientes onde a porta permanece aberta constantemente. Nesse contexto, o sistema atua como um mecanismo de segurança, impedindo a entrada de indivíduos categorizados como potencialmente perigosos, sendo uma opção mais indicada para comércios e locais nos quais a porta não é trancada. Destaca-se que se trata de um mesmo projeto, flexível o suficiente para se adaptar a diferentes cenários.

Portanto, o Sistema de Controle de Acesso Facial (SCAF) conseguiu atender às expectativas do grupo, realizando a interligação necessária.

4.2 RESULTADOS HARDWARE

O projeto passou por diversas etapas que exigiram um aprofundamento no entendimento de temas como treinamento, redes neurais, visão computacional e OpenCV. Um aspecto crucial que impactou significativamente o projeto foi a utilização de hardware, incluindo computadores e notebooks. Ao todo, foram empregados 2 computadores e 2 notebooks para testes, evidenciando a compreensão de que ferramentas que envolvem aprendizado e treinamento de máquinas demandam recursos de hardware. Esse impacto foi corroborado por estudos e artigos, como "Avaliação de Desempenho de Visão Computacional em Aplicações Móveis" (CUNHA, 2018, p.36-44) e "Protótipo de Sistema de Visão Computacional para Captura, Análise e Transmissão de Imagens Aquáticas" (BARCELOS, 2021, p.51). Esses trabalhos evidenciam que o hardware utilizado desempenha um papel significativo nos resultados, tornando o processo diferenciado para cada configuração de máquina, como demonstrado na Tabela 01 com os equipamentos utilizados pela equipe.

Tabela 01: configurações de computadores

Notebook		
PC1	Lenovo	Intel Celeron 1.10 GHz; 8 GB; SSD 480 GB; Sistema Ubuntu
PC2	Asus	Intel i5 3º 2.50 Ghz; 8 GB; 220 SSD; Sistema Ubuntu
Computador		

PC3	Lenovo	Intel i5 12 ^o 3.10 Ghz; 8 GB; 280 SSD MV; Sistema Ubuntu
PC4	Montado	Ryzen 5 5600; GTX 1070 TI; 8 GB; SSD 120 GB; Sistema Ubuntu

Com base no hardware utilizado, ajustes no treinamento foram necessários. Através de códigos, foi possível configurar taxas de frames, a quantidade de processamento que cada frame consumiria da máquina, e até mesmo a resolução da imagem, sendo crucial adaptar esses parâmetros para máquinas com menor capacidade.

O consumo de hardware nos PCs 1 e 2, onde o projeto apresenta uma variação significativa nos resultados finais ao reconhecer um usuário do sistema. Verifica-se uma interferência de tempo no reconhecimento facial, sendo que no PC1, com FPS de 4.32, leva cerca de 4.6 segundos, enquanto no PC2, com FPS de 7.2, ocorre em torno de 2.1 segundos. Já nos PCs 3 e 4, com FPS de 12.4, o tempo de 1.7 segundos permanece constante. Mesmo em hardwares menos avançados, o projeto manteve seu funcionamento, apresentando resultados positivos conforme as expectativas da equipe.

Ao realizar o protótipo do projeto, foram utilizados os materiais listados na Tabela 02 para a realização de testes. Essa etapa envolveu um custo adicional, contribuindo para o resultado final alcançado no Sistema de Controle de Acesso Facial.

Tabela 02: Matérias do protótipo.

O protótipo foi construído com base nesses matérias.	
Unidade	Material
2	Trancas eletrônicas de 12 V
2	Câmeras WebCam de 720p
2	Câmeras IP – Intelbras IM1; Marca genérica.
1	Arduino uno
1	Relé de 5V
1	Caixa de Som USB

O projeto obteve sucesso ao alcançar as métricas estabelecidas e atingir os objetivos propostos em relação ao protótipo físico. No entanto, em relação ao uso da tranca eletrônica, ela se mostrou adequada para o controle de acesso, mas inadequada para o bloqueio efetivo de pessoas. Nos testes realizados, observou-se que a tranca é programada apenas para ser acionada, abrindo mesmo ao identificar usuários bloqueados, como evidenciado no vídeo¹ de testes.

¹ Link vídeo: <https://drive.google.com/file/d/18UWApRJzdlh7uvOIEsjNMpOgEv94IJbm/view?usp=sharing>

Portanto, os resultados indicam a necessidade de explorar outras soluções de travamento para essa funcionalidade específica.

Portanto, para finalizar, o projeto foi concretizado em uma maquete, conforme ilustrado na Figura 36. Com ela, é possível simular a entrada de usuários cadastrados no SCAF.

Figura 36:



Fonte: Autores, 2023.

4.3 SISTEMA DE CONTROLE DE ACESSO FACIAL

O protótipo, que visa aprimorar a segurança nos setores de trabalho, tem como objetivo selecionar funcionários, desconhecidos e bloqueados, sendo capaz de se comunicar com um hardware para bloquear ou conceder acesso. Em caso de autorização, o sistema permite a entrada; caso contrário, emite sinais sonoros como medida de segurança. Essa operação é demonstrada no vídeo² e vídeo³, onde o projeto concede e bloqueia o acesso, respectivamente.

Portanto, neste projeto de conclusão de curso, nesse presente trabalho alcançou as métricas estabelecidas em seus objetivos. No entanto, em fase de testes e aprimoramento, o estado atual do sistema SCAF ainda é sua versão alfa e passará por diversas melhorias,

² Link vídeo: https://drive.google.com/file/d/1tnbe_qsxiri6FofpmO0vXbFuXxkngh1B/view?usp=sharing

³ Link vídeo: https://drive.google.com/file/d/1i5t0xLNgZVzIWEAyPC_z4xEpv3WeNAhP/view?usp=sharing

conforme mencionado posteriormente. O protótipo físico também será aprimorado ao longo do processo.

5 CONSIDERAÇÕES FINAIS

No decorrer deste trabalho, foi possível explorar a importância crescente da segurança nos mais variados contextos, desde o âmbito pessoal até o empresarial. A introdução de tecnologias avançadas no campo da segurança eletrônica foi um dos principais enfoques, visando mitigar os riscos de incidentes e fortalecer a proteção de espaços e pessoas. O desenvolvimento tecnológico incisivo tem sido um aliado crucial na busca por soluções que garantam a proteção coletiva e individual.

A proposta de desenvolvimento de um sistema de segurança integrado, onde consegue fazer o controle de acesso de pessoas, assim como identificar potenciais ameaças por meio do monitoramento de câmeras de segurança, encontra-se concretizada. Esta proposta não apenas foi concretizada, mas também se mostrou promissora nos testes preliminares, evidenciando a viabilidade e eficácia do projeto.

Os resultados que foram obtidos, baseados na utilização de tecnologias específicas, sobretudo no uso da linguagem de programação Python e sua integração com OpenCV e Face Recognition, reforçaram a capacidade de viabilizar e adaptar o sistema proposto. A capacidade de processar imagem e vídeo, aliada à simplicidade de uso da linguagem Python, foi fundamental na criação do sistema na sua totalidade.

A hipótese inicial, fundamentada na necessidade de criar um sistema eficiente e integrado para a proteção de possíveis ameaças, pode ser confirmada pelos resultados promissores obtidos. E de certa forma, a pesquisa não apenas validou a hipótese como também identificou nuances e possíveis cenários para aprimoramentos futuros.

Porém, é importante destacar as limitações constatadas durante a execução deste projeto. Dificuldades específicas, particularmente relacionadas à câmera de segurança IP. Houve uma significativa redução na taxa de quadros devido à perda de pacotes de dados enviados pela câmera. Além disso, enfrentamos desafios ao executar o programa em dispositivos inadequados, onde, era demandando um alto poder de processamento e capacidade de renderização de vídeo, o que resultou em um progresso mais lento no desenvolvimento. Com isso pode ser evidenciado áreas que precisam de maior atenção em futuras iterações do sistema.

Uma das direções mais significativas para o futuro deste projeto é o aprimoramento contínuo da Inteligência Artificial. Investir em algoritmos mais avançados, como o Estimativa de Pose Humana (EPH), onde consegue identificar figuras na forma humana, poderá aperfeiçoar a sua precisão na identificação de ameaças, proporcionando um sistema mais robusto e confiável. Assim como também, a integração da tela de monitoramento ao sistema web

utilizando JavaScript, e a integração de um NVR (Network Video Records), permitindo ampliar a área de visão do sistema e garantindo sua total conectividade online.

Olhando para o futuro acadêmico e profissional, este estudo apresentou alguns possíveis caminhos. A continuidade no desenvolvimento do sistema, seja na forma de aprofundamentos acadêmicos ou na aplicação prática no mercado, se mostra promissor na contribuição do avanço na segurança e em diferentes aspectos tecnológicos.

Portanto, o desenvolvimento deste sistema de segurança, representa não apenas uma conquista acadêmica, mas também uma contribuição muito significativa para na proteção de espaços públicos e privados. O seu potencial vai além do ambiente acadêmico, podendo ser aplicado em diferentes lugares para garantir um ambiente mais seguro e protegido para todos.

REFERÊNCIAS BIBLIOGRÁFICAS

- ABADI, Martín; AGARWAL, Ashish; BARHAM, Paul. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems**. Documento Técnico Preliminar, 2016. Disponível em: < <https://arxiv.org/abs/1603.04467>>. Acesso em: 21 de agosto de 2023.
- Arduino. **SoftwareSerial Library**. 2023. Disponível em: <<https://docs.arduino.cc/learn/built-in-libraries/software-serial>>. Acesso em: 15 nov. 2023.
- ARTEM, Guilherme; CASTRO, Henrique; MAURO, José. **Uma Visão Geral sobre a Linguagem**. 2013. Disponível em: < <http://www.inf.ufes.br/~vitorsouza/archive/2020/wp-content/uploads/teaching-lp-20132-seminario-python.pdf>>. Acesso em: 20 de novembro.
- BARCELOS, Matheus de Oliveira. **Protótipo de Sistema de Visão Computacional para Captura, Análise e Transmissão de Imagens Aquáticas**. Universidade Federal de Santa Catarina. Campus Araranguá. Curso de Graduação em Engenharia de Computação. 2021. Disponível em:<https://repositorio.ufsc.br/bitstream/handle/123456789/228983/Prot_tipo_de_Sistema_de_Vis_o_Computacional_para_captura_an_lise_e_Transmiss_o_de_Imagens_aqu_ticas_V22.pdf?sequence=1&isAllowed=y>. Acesso em: 15 nov. 2023.
- BRENNER, Grabiell, P. S ;BIZARRIA, W. **Sistema de Controle de Acesso com Biometria da Digital**. 2023. Disponível em: < <https://www.aedb.br/seget/arquivos/artigos11/44914520.pdf>>. Acesso em: 21 de agosto de 2023.
- CARVALHO, V. S. **Reconhecimento de faces baseado em redes neurais profundas**. Saarbrücken: Editora Novas Edições Acadêmicas, 2018.
- CENDRON, Marcelo. **Comunicação Serial**. Curso Técnico em Automação Industrial. Programação aplicada a Microcontroladores. Disponível em: < <https://professor.luzerna.ifc.edu.br/marcelo-cendron/wp-content/uploads/sites/40/2018/03/Teoria-07-serial.pdf>>. Acesso em: 15 nov. 2023.
- CUNHA, Luiz Henrique da Silva. **Avaliação de Desempenho de Visão Computacional em Aplicações Móveis**. Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Computação e Sistemas. Fevereiro, 2018. Disponível em: <https://www.monografias.ufop.br/bitstream/35400000/769/1/MONOGRAFIA_Avalia%C3%A7%C3%A3oDesempenhoVis%C3%A3o.pdf>. Acesso em: 15 nov. 2023.
- FILHO, Eduardo M. **Controle de acesso por reconhecimento facial e controle de ambientes**. 2018. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – Faculdade de Tecnologia de Americana, 2018.
- GOMEZ, Jack C. **Reconhecimento facial no controle de acesso residencial**. 2019. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Industrial Elétrica) – Centro Universitário Uniamérica, 2019. Disponível em: < <https://pleiade.uniamerica.br/index.php/bibliotecadigital/article/view/759>>. Acesso em: 15 de agosto de 2023.
- HAMAGUTI, É. K.; BREVE, F. A. **Introdução sobre Machine Learning e Deep Learning**. 2022. In: 11ª Jornada Científica e Tecnológica da Fatec de Botucatu - São Paulo, 2022.

Disponível em: <<https://www.fabriciobreve.com/artigos/2022/jornacitec2022expandido.pdf>>. Acesso em: 21 de agosto de 2023.

HOSAKI, G. Y.; RIBEIRO, D. F. **Deep Learning: Ensinando a aprender**. 2021 Revista de Gestão e Estratégia - RGE, v. 3.

INTELBRAS. **Tutorial Técnico - Mibo Cam Acesso RTSP**. 2022. Disponível em: <<https://backend.intelbras.com/sites/default/files/2022-05/mibo-cam-rtsp.pdf>>. Acesso em: 21 de agosto de 2023.

JAIN, A. K.; NANDAKUMAR, K.; ROSS, A. **50 years of biometric research: Accomplishments, challenges, and opportunities**. Pattern Recognition Letters, v. 79, p. 80-105, 2016.

JÚNIOR, W. A. M. **Estudo e Aplicação de Visão Computacional e Redes Neurais para Localização e Detecção de Falhas em Montagem de PCBS**. 2020. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) - Universidade Federal de Santa Catarina, Departamento de Engenharia Elétrica e Eletrônica, Florianópolis, 2020. Disponível em: <<https://repositorio.ufsc.br/bitstream/handle/123456789/204353/TCC.pdf?sequence=1&isAllowed=y>>. Acesso em: 21 de agosto de 2023.

KAUFFMANN, Boris. **Protocolos de Transmissão de Vídeo sobre IP com Compressão**. Setexperience, 2020. Disponível em: <<https://setexperience2020.set.org.br/wp-content/uploads/2020/11/Protocolos-de-Transmissao-de-Video-sobre-IP.pdf>>. Acesso em: 21 de agosto de 2023.

KAUFMAN, D.; SANTAELLA, L. **O papel dos algoritmos de inteligência artificial nas redes sociais**. Revista FAMECOS, Porto Alegre, v. 27, p. 1-10, jan.-dez. 2020. Disponível em: <<https://revistaseletronicas.pucrs.br/index.php/revistafamecos/article/view/34074>>. Acesso em: 21 de agosto de 2023.

Keras, Simple. Flexible. Powerful. **Keras**. 2023. Disponível em: <https://keras.io/>. Acesso em: 07 junho de 2023.

LECUN, Y.; BENGIO, Y. **Deep Learning**. 2015. Nature, v. 521, n. 7553, p. 436-444. DOI: 10.1038/nature14539. Disponível em: <https://www.researchgate.net/publication/277411157_Deep_Learning>. Acesso em: 21 de agosto de 2023.

MACIEL, F. M. B. **Python e Django: Desenvolvimento Web Moderno e Ágil**. Rio de Janeiro: Editora Alta Books, 2020.

MANZANO, J. A. N. G. **Introdução à linguagem Python**. São Paulo: editora Novatec, 2018.

NUSSEY, J. **Arduino For Dummies**. Nova Jersey: editora Wiley, 2018.

Oliveira, H. S. **Processamento de Imagens com OpenCV e Python: Uma Abordagem Prática** São Paulo: editora Novatec, 2017.

PANOSSO, Alan Rodrigo. **Linguagem de Programação C++**. 2019. Universidade Estadual Paulista Júlio de Mesquita Filho, Faculdade de Ciências Agrárias e Veterinárias,

Departamento de Ciências Exatas. Disponível em: <https://www.fcav.unesp.br/Home/departamentos/cienciasexatas/alanrodrigopanosso/apostila_cpp_2019.pdf>. Acesso em: 13 de novembro de 2023.

PONTI, M. A.; COSTA, G. B. P.. Como funciona o Deep Learning. In: Tópicos em Gerenciamento de Dados e Informações. SBC, 1ª ed., 2017.

Python Package Index.**face-recognition 1.3.0**. Pypi documentation. Disponível em: <<https://pypi.org/project/face-recognition/#description>>. Acesso em: 20 de novembro.

Python Software Foundation. **Pickle — Python object serialization**. Python 3.10.2 documentation. Disponível em: <<https://docs.python.org/3/library/pickle.html>>. Acesso em: 01 de março de 2023.

RIBEIRO, L. O. **OpenCV com Python: um guia prático para construção de aplicativos de visão computacional**. São Paulo: editora Novatec, 2020.

Setor de segurança eletrônica cresce 14% no Brasil e fatura R\$ 9,24 bilhões. **TIINSIDE**, 2022. Disponível em: <<https://tiinside.com.br/09/03/2022/setor-de-seguranca-eletronica-cresce-14-no-brasil-e-fatura-r-924-bilhoes/>>. Acesso em: 01 de junho de 2023.

SICHMAN, J. S. **Inteligência Artificial e sociedade: avanços e riscos**. 2021. Universidade de São Paulo (USP), Escola Politécnica, Departamento de Engenharia de Computação e Sistemas Digitais, São Paulo, Brasil. Disponível em: <<https://www.scielo.br/j/ea/a/c4sqqrthGMS3ngdBhGWtKhh/?format=pdf>>. Acesso em: 21 de agosto de 2023.

SILVA, A. M. A. **Segurança para câmeras IP**. 2018. 61f. Trabalho de conclusão de curso (Bacharelado em Sistemas de Informação) - Universidade Federal de Uberlândia, Uberlândia, 2018. Disponível em: <https://repositorio.ufu.br/bitstream/123456789/26148/3/Seguran%c3%a7aC%c3%a2merasIP.pdf>. Acesso em: 21 de agosto de 2023.

SILVA, I.N; OLIVEIRA, M.V. **Redes Neurais Artificiais para Engenharia e Ciências Aplicadas**. Saarbrücken: Editora Novas Edições Acadêmicas, 2019

STROUSTRUP, Bjarne. Thriving in a Crowded and Changing World: C++ 2006–2020. In: Proc. ACM Program. Lang., Vol. 4, No. HOPL, Article 70. Publication date: June 2020. Shepherd: Yannis Smaragdakis, University of Athens, Greece. Acesso em: 13 de novembro de 2023.

VIEIRA V.; RAZENTE H. L.; CAMILA M. N. **Tópicos Em Gerenciamento De Dados E Informações 2017**. In: 32º SIMPÓSIO BRASILEIRO DE BANCOS DE DADOS, 2017, Uberlândia. Anais [...]. Uberlândia: Universidade Federal de Uberlândia – UFU, 2017.

VIEIRA, Larissa. Pesquisa mostra aumento de 45% no número de ataques em escolas no Brasil. **Multiverso Notícias**, 2023. Disponível em: <<https://multiversonoticias.com.br/aumento-ataques-a-escolas-no-brasil/>>. Acesso em: 7 de maio de 2023.