DESENVOLVIMENTO DE UM SOFTWARE DE ASSIDUIDADE ESTUDANTIL, ATRAVÉS DE RECONHECIMENTO FACIAL EM REAL-TIME, NO CAMPUS UNIVERSITÁRIO.

Bruno Kobi Valadares de Amorim

Graduando de Sistemas de Informação da Unisales.
Email: brunokobi2@hotmail.com

Rômulo Ferreira Douto

Docente da Unisales.
E-mail: rdouro@salesiano.br

RESUMO

Este Artigo propende a elaboração de um software web, que visa minimizar o tempo desperdiçado com trabalhos burocráticos em sala de aula, automatizando o processo de chamada dos alunos, através do reconhecimento facial. Através de pesquisas bibliográficas sobre reconhecimento facial foi desenvolvido um software utilizando-se do framework React , em linguagem TypeScript, com a utilização da Faceapi.js, que implementa modelos de redes neurais sobre o TensoFlow.js no browser em tempo real. O software foi implementado e testado em ambiente de desenvolvimento gerando resultados satisfatórios no sistema de reconhecimento facial, devido a pandemia de covid-19, houve a impossibilidade para testes de campo.

Palavra-chave: Reconhecimento Facial, FaceApi , React, TypeScript.

ABSTRACT

This article proposes the development of a web software, which aims to minimize the time wasted with bureaucratic work in the classroom, automating the process of calling students, through facial recognition. Through bibliographic research on facial recognition, software was developed using the React framework, in TypeScript language, using Faceapi.js, which implements neural network models on TensoFlow.js in the browser in real time. The software was implemented and tested in a development environment, generating satisfactory results in the facial recognition system. Due to the covid-19, pandemic, field tests were impossible.

Keyword: Facial Recognition, FaceApi , React, TypeScript.

1. INTRODUÇÃO

O art. 47, § 3º, da Lei das Diretrizes e Bases da Educação Nacional (LDB) determina que é obrigatória a frequência de alunos, e para a aprovação, a frequência mínima de 75% do total de aulas (BRASIL,2020).

Segundo Phillips a tarefa de reconhecimento facial é realizada pelos humanos sem o menor esforço, entretanto essa tarefa é extremamente complexa no ambiente computacional (PHILLIPS, 2003).

As técnicas de reconhecimento facial têm evoluído muito, com uso de uma câmara, é possível reconhecer certos traços de uma pessoa por meio de marcadores comuns, e utilizando uma série de redes neurais convulsionais (CNNs), já é possível além de identificar a pessoa, também determinar idade, sexo e expressões físicas.

Segundo Machado em média 20% do tempo de aula, é desperdiçado com trabalhos burocráticos, entre eles está a chamada de presença de alunos. (MACHADO, 2016).

A pesquisa patrocinada pelo Banco Mundial, com observação de mais de 15 mil professores, em sete países, Brasil inclusive, apresenta conclusões preocupantes como, o fraco domínio do conteúdo acadêmico, apesar da elevada titulação; uso de práticas ineficazes em sala de aula, na qual utilizam 65%, ou menos, do tempo de aula para ensino, o que equivale à perda de um dia de aula por semana; uso limitado das tecnologias da informação (BRUNS; LUQUE, 2014).

O software a ser desenvolvido visa gerenciar a assiduidade dos acadêmicos dentro do campus universitário, atrás de reconhecimento facial, minimizando o tempo desperdiçado em aula, com o processo de chamada dos alunos, através da tecnologia de reconhecimento facial da faceapi.js, será implementada na aplicação um modulo de reconhecimento facial, e será feito uma análise do desempenho do software.

2. REFERENCIAL TEÓRICO

Nesse capitulo, iremos citar alguns trabalhos que possuem uma correlação com nosso trabalho, e serão evidenciados conceitos relacionados ao reconhecimento facial, e

também conceitos e tecnologias relacionadas ao desenvolvimento da aplicação em si com um foco maior na faceapi.js e no React.

2.1. Trabalhos Relacionados

Na China, os estudantes da universidade em Pequim hoje já contam com um sistema de reconhecimento facial com qual eles podem controlar e registrar a presença dos estudantes. A operação deste sistema consiste em que o professor deve ativar o sistema instalado em seu tablet antes de iniciar a aula e tirar fotos um por um os estudantes, em seguida, estas imagens são comparadas com as fotos armazenadas em um banco de dados e com isso é feito o registro. Este sistema é baseado na tecnologia de reconhecimento facial da plataforma de inteligência artificial da gigante da Internet chinesa, Baidu (XANGAI, 2017).

No Equador, estudantes da universidade de Guayaquil, desenvolveram um projeto para otimizar o tempo no processo do registro, e o monitoramento de comparecimento do estudante para os professores de engenharia em sistemas, através de um aplicativo móvel que implementa a tecnologia do reconhecimento facial dentro do mesmo , neste projeto o professor faz todo processo sozinho, vai levar uma ou mais fotos para agrupar seus estudantes em sua classe e o aplicativo vai identificar os rostos comparando -os com as imagens da mesma ser encontrados armazenadas em um servidor na nuvem e automaticamente gravado o a presença dos alunos (PARDILLA,2020).

No Brasil, os alunos do Instituto Federal do Espírito Santo (IFES) elaboram um aplicativo, que utiliza inteligência artificial, eles realizam as chamadas em sala de aula usando reconhecimento facial, essa tecnologia assegura o controle de presença com mais rigidez e reduz o tempo para verificar quais alunos assistiram à aula. Durante as chamadas, os professores acessam o aplicativo e pedem para os alunos olharem para a câmera de seu celular, ao tirar uma foto da turma, o aplicativo identifica a face dos alunos presentes e as compara com as imagens cadastradas no banco de dados. (FILHO,2019).

2.2. Reconhecimento Facial

O processo de reconhecimento facial, pode ser dívida nos seguintes grupos: Detecção de faces, Extração de características e Representação da Face, Reconhecimento e

Verificação. Um sistema de reconhecimento facial totalmente automatizado é capaz de receber como entrada uma imagem ou vídeo, identificar as faces presentes, caracterizá-las matematicamente, compará-las com outras previamente cadastradas em um banco de dados e caso haja alguma correspondência, informar, como saída, qual face do banco de dados condiz à imagem de entrada. (BRAGA,2013)

2.2.1. Detecção facial

Essa etapa é importante porque elimina da imagem informações desnecessárias. Se o algoritmo encontra uma ou mais faces, estas são extraídas da imagem original de modo que sejam analisadas separadamente. (BRAGA,2013).

2.2.2. Extração de características

O objetivo dessa etapa é localizar regiões da imagem que contenham características significativas. Essas regiões podem ser globais ou locais e podem ser distinguidas por texturas, formas, intensidades, propriedades estatísticas e outros. De um modo geral, tenta-se extrair um conjunto compacto de características geométricas interpessoais ou características fotométricas da face. Diferentemente da detecção facial, que categoriza todas as faces numa única classe, na extração de características tenta-se encontrar fatores discriminantes que permitam que cada uma das faces possa ser diferenciada matematicamente das demais. (BRAGA,2013).

2.2.3. Reconhecimento / Identificação

Consiste em avaliar se as características faciais de entrada, representadas por um modelo matemático, condizem com a de alguma face já previamente cadastrada no banco de dados. É interessante notar que ao contrário do problema de detecção facial, que pode ser considerado como uma classificação de duas classes (faces e não-faces), o de reconhecimento facial é um problema de classificação de muitas classes (uma face contra todas as outras). A verificação é um processo de correspondência de um para um, onde a face de entrada é testada somente contra uma face do banco de dados. Já na identificação, o processo de correspondência é de um para vários, porque compara a face de entrada com todas do banco de dados.

Existem diversas abordagens para o problema de classificação de faces. Os métodos mais sofisticados se baseiam em densidades de probabilidade, e redes neurais. (BRAGA,2013).

2.3. FaceApi.js

Criada por Vincent Mühler em 2018, a faceapi.js é uma API JavaScript para detecção e reconhecimento facial implementado no browser, sobre a API principal do tensorflow.js. A Face-api.js implementa uma série de redes neurais convulsionais (CNNs), otimizada para a web e dispositivos móveis (CARLETO,2018).

Os modelos pré-treinados que atendem a um determinado objetivo, como reconhecer objetos, formas, detectar / reconhecer rostos, classificar imagens, estimativas, etc.

Eles são classificados em duas categorias: Modelos para detecção e modelos para reconhecimento.

• Modelos de detecção encontram um rosto em uma foto digital e dar as coordenadas em pixels de onde o cara.

• Modelos de reconhecimento, tomando como dados, o as coordenadas do padrão de detecção são posicionadas no rosto e eles farão campanha com uma base de rostos e dentro de um mar de fotos obtenha aquele com as características assimétricas mais semelhantes.

Esses modelos foram compactados e divididos em partes para serem armazenados na memória cache dos navegadores (MÜHLER,2018).

2.3.1. Modelos de detecção de rosto

São redes neurais que tem com objetivo localizar faces , a faceapi.js possui 3 modelos com essa finalidade.

2.3.1.1. SSD Mobilenet V1

Para detecção de rosto, este projeto implementa um SSD (Single Shot Multibox Detector) baseado em MobileNetV1. A rede neural computará as localizações de cada face em uma imagem e retornará as caixas delimitadoras junto com sua probabilidade para cada face. Este detector facial visa obter alta precisão na detecção de caixas delimitadoras de faces, em vez de baixo tempo de inferência. O tamanho do modelo é de cerca de 5,4 MB (**ssd_mobilenetv1_model**) (MÜHLER,2018).

2.3.1.2. Tiny Face Detector

O Tiny Face Detector é um detector de rosto em tempo real de muito desempenho, que é muito mais rápido, menor e consome menos recursos em comparação com o

detector de rosto SSD Mobilenet V1, por sua vez tem um desempenho um pouco menos bom na detecção de rostos pequenos. Este modelo é extremamente móvel e amigável à web, portanto, deve ser o seu detector facial GO-TO em dispositivos móveis e clientes com recursos limitados. O tamanho do modelo quantizado é de apenas 190 KB (**tiny_face_detector_model**).

O detector de rosto foi treinado em um conjunto de dados personalizado de aproximadamente 14 mil imagens marcadas com caixas delimitadoras. Além disso, o modelo foi treinado para prever caixas delimitadoras, que cobrem inteiramente os pontos de características faciais, portanto, em geral, produz melhores resultados em combinação com a detecção subsequente de pontos de referência do rosto do que o SSD Mobilenet V1.

Este modelo é basicamente uma versão ainda menor do Tiny Yolo V2, substituindo as convoluções regulares de Yolo por convoluções separáveis em profundidade. Yolo é totalmente convolucional, portanto, pode se adaptar facilmente a diferentes tamanhos de imagem de entrada para trocar a precisão pelo desempenho (tempo de inferência) (MÜHLER,2018).

2.3.1.3. MTCNN

MTCNN (Redes Neurais Convolucionais em Cascata Multi-tarefa) representa um detector facial alternativo para SSD Mobilenet v1 e Tiny Yolo v2, que oferece muito mais espaço para configuração. Ajustando os parâmetros de entrada, o MTCNN deve ser capaz de detectar uma ampla gama de tamanhos de caixa delimitadora de face. MTCNN é um CNN em cascata de 3 estágios, que retorna simultaneamente 5 pontos de referência de rosto junto com as caixas delimitadoras e pontuações para cada rosto. Além disso, o tamanho do modelo é de apenas 2 MB(MÜHLER,2018)..

MTCNN foi apresentado no artigo Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks por Zhang et al. e os pesos do modelo são fornecidos no repositório oficial da implementação do MTCNN.

2.3.2. Modelos de detecção de pontos de referência de 68 pontos

Este pacote implementa um detector de pontos de referência facial de 68 pontos muito leve e rápido, mas preciso. O modelo padrão tem um tamanho de apenas 350kb (**face_landmark_68_model**) e o modelo minúsculo tem apenas 80kb

(**face_landmark_68_tiny_model**). Ambos os modelos empregam as idéias de convoluções separáveis em profundidade, bem como blocos densamente conectados. Os modelos foram treinados em um conjunto de dados de ~ 35k imagens de rosto marcadas com 68 pontos de referência de rosto (MÜHLER,2018).

2.3.3. Modelo de reconhecimento facial

Para reconhecimento de rosto, uma arquitetura semelhante ao ResNet-34 é implementada para calcular um descritor de rosto (um vetor de recursos com 128 valores) a partir de qualquer imagem facial, que é usada para descrever as características do rosto de uma pessoa. O modelo não se limita ao conjunto de rostos usados para treinamento, ou seja, você pode usá-lo para reconhecimento de rosto de qualquer pessoa, por exemplo você mesmo. Você pode determinar a semelhança de duas faces arbitrárias comparando seus descritores de face, por exemplo, calculando a distância euclidiana ou usando qualquer outro classificador de sua escolha.

A rede neural é equivalente à FaceRecognizerNet usada em face-recognition.js e a rede usada no exemplo de reconhecimento facial dlib . Os pesos foram treinados por davisking e o modelo atinge uma precisão de predição de 99,38% no benchmark LFW (Labeled Faces in the Wild) para reconhecimento facial.

O tamanho do modelo quantizado é de aproximadamente 6,2 MB (face_recognition_model) (MÜHLER,2018).

2.3.4. Modelo de reconhecimento de expressão facial

O modelo de reconhecimento de expressão facial é leve, rápido e oferece precisão razoável. O modelo tem um tamanho de aproximadamente 310kb e emprega convoluções separáveis em profundidade e blocos densamente conectados. Ele foi treinado em uma variedade de imagens de conjuntos de dados disponíveis publicamente, bem como imagens extraídas da web. Observe que o uso de óculos pode diminuir a precisão dos resultados da previsão (MÜHLER,2018).

2.4. TECNOLOGIAS PARA DESENVOLVIMENTO WEB

2.4.1. HTML

É uma sigla para HyperText Markup Language (Linguagem para Marcação de Hipertexto) (SILVA, 2019). O HTML pode ser definido como um código, que quando

interpretado pelo navegador, é capaz de exibir textos e imagens de forma amigável para o usuário. Atualmente é o formato principal de desenvolvimento de aplicações Web. Os navegadores não exibem as tags HTML, mas utilizam-nas para determinar a forma de exibição do documento (W3C, 2018).

2.4.2. CSS

É uma sigla para Cascading Style Sheets (Folha de Estilo em Cascatas) e é a ferramenta responsável por formatar as informações contidas no código HTML de uma aplicação Web (EIS; FERREIRA, 2012). Segundo Silva (2011), a maior funcionalidade do CSS é a apresentação de um documento, já que cabe ao HTML apenas sua estruturação, sem definição de cores e tamanhos, por exemplo. O conceito de CSS foi proposto por Håkon Wium Lie em 1994, ao ver a necessidade em formatar páginas HTML mantendo o código simples e fácil de entender e manter. Em 1995 sua proposta foi apresentada ao W3C (World Wide Web Consortium) que se interessou pelo projeto e em 1996 já foi lançada a primeira especificação CSS, a CSS Level 1 (EIS, 2018).

2.4.3. JavaScript

É uma a linguagem de programação para a web, pode atualizar e alterar HTML e CSS, pode calcular, manipular e validar dados (W3C,2018). A linguagem JavaScript é interpretada pelos navegadores de internet, sendo capaz de manipular o conteúdo de uma página web de maneira dinâmica (PRESCOTT, 2016).

2.4.4. TypeScript

O TypeScript permite o código JavaScript seja desenvolvido com Orientação a Objeto.Como Daniel Schmitz (2015) define que “Se você gosta de tipar suas variáveis e métodos, criar classes, interfaces, usar Orientação a Objetos, o TypeScript foi feito para você e, claro, pode dizer adeus ao JavaScript.”.Com ele todas as classes que escrevemos são compiladas e transformadas em JavaScript puro.

2.5. FRAMEWORKS E TECNOLÓGIAS PARA DESENVOLVIMENTO

2.5.1. React.js

Em 2013, a empresa Facebook Inc. disponibilizou o framework React, de código aberto baseado em JavaScript para a construção de interfaces com o usuário

(FEDOSEJEV, 2015).O React, é na verdade uma biblioteca do javaScript com foco no frontend das aplicações, é utilizada e sites de grandes empresas da Web.

2.5.2. TensorFlow

Criado pela equipe do Google Brain, TensorFlow é um framework *open-soure* desenvolvido paraPython e JavaScript, que auxilia no desenvolvimento de soluções com *machine learning*. Pode ser executadosobre diversas plataformas e arquiteturas, incluido CPUs, GPUs e as recentes TPUs (*Tensor ProcessingUnit*). Atualmente, é um dos principais *frameworks* do mercado para criação de redes neurais *deep learning*. Com ele, é possível agilizar e facilitar o processo de obtenção de dados, treinar modelos, realizar *prediction*se refinar resultados futuros (TENSORFLOW, 2019).

2.5.3. Node.js

Em 2009, apareceu o Node.js, o Node.js pegou o V8, o poderoso motor JavaScript do Google Chrome , e permitiu ser executado em servidores . Em adição para o Ruby, Python , C # , Java e outras linguagens , os desenvolvedores agora podem escolher JavaScript no desenvolvimento de aplicações sobre o servidor . Node.js tem benefícios reais. Para uma coisa, o V8 JavaScript motor é rápido, e Node.js incentiva um estilo de codificação assíncrona , de modo que marcas que o código é mais rápido e evitar os pesadelos de vários segmentos. JavaScript também tinha um grande número de bibliotecas úteis devido à sua popularidade. Mas o maior benefício do Node.js é a capacidade de compartilhar código entre o navegador e o servidor (HAHN , 2016).

2.5.4. JSON

JSON, em seu significado teórico é "Javascript Object Notation", do qual nada mais é que o formato mais leve conhecido de transferência/intercâmbio de dados, ele é similar ao XML, e tem a mesma utilidade, mesmo intuito, porém é mais leve, o detalhe é que não necessariamente, apesar do nome, você tem que usá-lo com Javascript. Muitas linguagens hoje em dia dão suporte ao JSON Ele é muito usado para retornar dados vindos de um servidor utilizando requisições AJAX para atualizar dados em tempo real (LENGSTORF,2016).

2.5.5. Express.js

É um framework para NodeJS. Ele é minimalista, flexível e contém um robusto conjunto de recursos para desenvolver aplicações web, como um sistema de Vi- ews

intuitivo (MVC), um robusto sistema de roteamento, um executável para geração de aplicações e muito mais(ADMIN,2020).

2.5.6. Multer

É um middleware para NodeJS para manipulação de multipart/form-data, que é usado principalmente para o upload de arquivos. É escrito sobre o busboy para maior eficiência (Multer,2020).

2.5.7. TypeORM

O TypeORM é um módulo avançado de gerenciamento de relações de objeto que é executado no Node.js. Como o nome indica, o TypeORM deve ser usado com o TypeScript para persistência de dados no node.js. (HERON,2018).

3. METODOLOGIA

O presente trabalho, do ponto de vista da natureza pode ser classificado como uma pesquisa aplicada, que visa desenvolver um software que gerencie a assiduidade dos acadêmicos dentro do campus universitário, através do reconhecimento facial, visando automatizar o processo de chamada dos alunos, com objetivo de otimizar o tempo de aula dentro do campus.

Para o desenvolvimento da aplicação foi necessária uma pesquisa bibliográfica na área do reconhecimento facial, algoritmos e soluções de reconhecimento facial que atendiam o objetivo do trabalho.

A metodologia de desenvolvimento utilizada na aplicação foi a metodologia ágil Scrum, tendo como principais vantagens a velocidade de desenvolvimento e a entrega de partes software antes da finalização do software por completo.

4. RESULTADOS

4.1. ENGENHARIA DE SOFTWARE

Aplicado os conhecimentos da engenharia de software foi desenvolvido diagramas para o auxiliar no desenvolvimento do software.

4.1.1. Análise de Requisitos

Em entrevista realizada com Professor Marcelo Schuster, coordenador de curso de sistema de informação na Unisales, identificou-se a necessidade de um sistema, que

pudesse automatizar o processo de chamada atual dos professores, visando reduzir o desperdício de tempo de aula, que segundo Machado e Bruns , podem chegar a 20%, foram identificados os seguintes requisitos inicialmente:

Tabela 1: Requisitos Funcionais

Número	**Requisito**	**Descrição**
RF1	O sistema permitirá o cadastramento de usuários para acessar o sistema	Registro de dados básicos, iniciar a sessão.
RF2	O sistema permitirá cadastramento dos Alunos	Crie, edite e exclua aluno.
RF3	O sistema irá permitir upload arquivos de imagem dos alunos.	Upload das imagens dos alunos
RF4	O sistema criará um registro do reconhecimento facial obtido	Registro da matricula, hora e local do reconhecimento

Fonte: elaboração própria

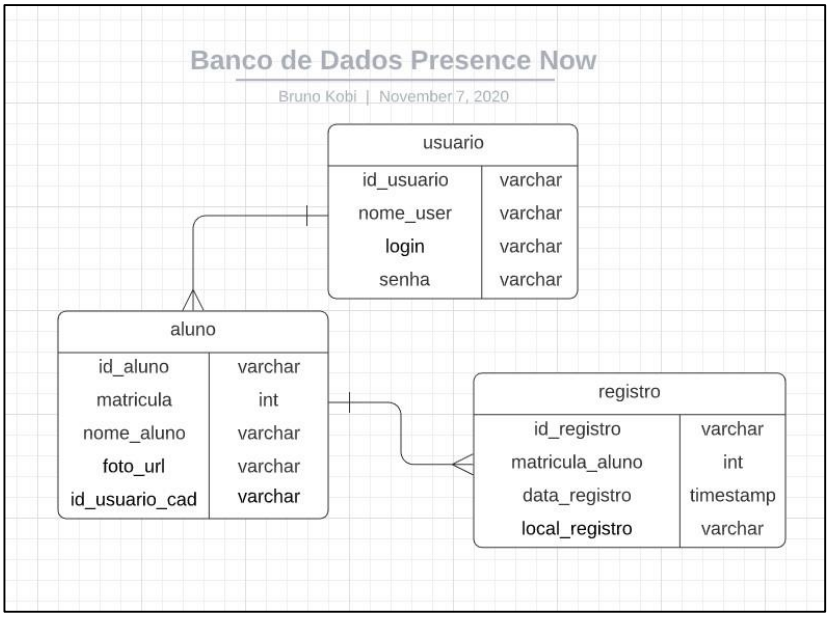
Tabela 2: Requisitos Não Funcionais

Número	Requisito	Descrição
RNF1	Facilidade de Uso.	Registro de dados básicos para iniciar.
RNF2	Segurança	O login será restrito por senhas criptografados.

Fonte: elaboração própria

4.1.2. Diagrama de Entidade Relacionamento

Figura 1 – Digrama de Entidade Relacionamento



Fonte: *elaboração própria*

4.2. DESENVOLVIMENTO

O resultado desse trabalho foi um software chamado **Presence Now**, que permite cadastrar usuários e alunos. O presence now pode ser aberto em qualquer PC que possua uma Webcam, ele deve ser instalado em locais estratégicos, preferencialmente com uma boa iluminação, quando a identificação do aluno é positiva, o sistema registra o local, a hora, o dia, e a matricula do aluno.

A figura 2, tratasse do logo da aplicação, ele foi desenvolvido e editado pelo GIMP.A aplicação foi desenvolvida em três partes: FrontEnd, BackEnd e Módulo de Reconhecimento.

Figura 2 – Logo do Software

Fonte: *elaboração própria*

4.2.1. FrontEnd

É a parte visual da aplicação, foi desenvolvida em React.js com Typescprit, através do Express.js, fazendo a comunicação através de JSON com API BackEnd.

O FrontEnd inicialmente foi desenvolvido com 6 telas:

A figura 3 é a tela de login, onde o usuário que estiver cadastrado na aplicação como usuário vai pode logar no sistema.

A tela possui um sistema de validação nos campos de preenchimento, e também possui um sistema de mensagens, relativo a autenticação do usuários no sistema.

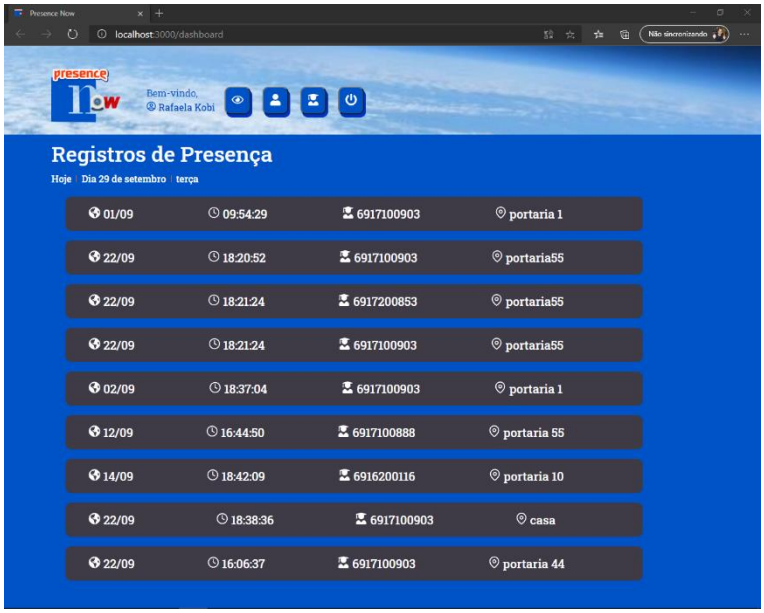
Figura 3 – Tela de Login

Fonte : elaboração própria

A figura 4 é a tela lista de registros, onde o usuário é direcionado logo após fazer o login com sucesso.

A tela possui uma lista de todos os registros armazenados no banco de dados, através de um GET na rota registros do BackEnd, essa tela também possui um menu principal na parte superior, onde o usuário tem acesso ao cadastro de aluno, ao cadastro de usuário e ao modulo de reconhecimento. Na parte superior também possui a logo da aplicação com uma saudação ao usuário que efetuou o login no sistema.

Figura 4 – Tela de Lista de Registros

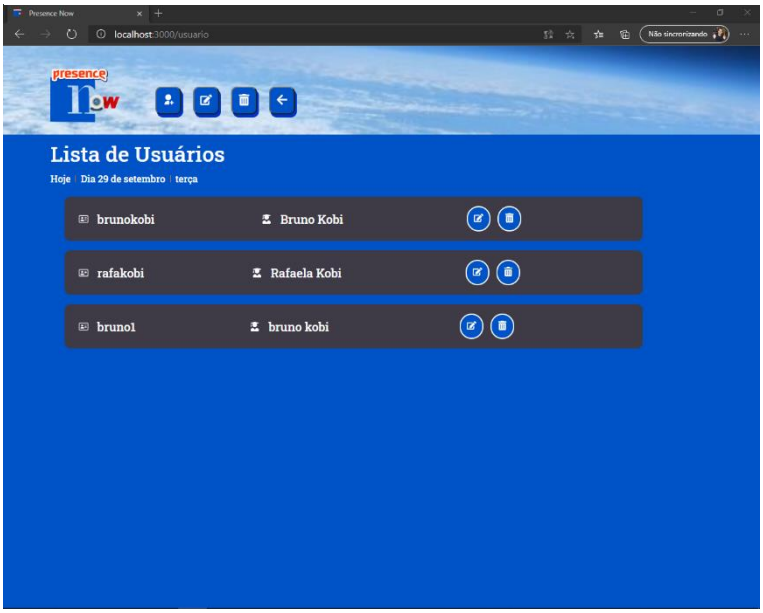


Fonte: elaboração própria

A figura 5 é a tela lista de usuários, onde o usuário é direcionado logo após clicar no botão cadastro de usuários.

A tela possui uma lista de todos os registros armazenados no banco de dados, através de um GET na rota usuários do BackEnd, essa tela também possui um menu na parte superior, onde o usuário tem a opção de cadastrar usuário, editar e voltar.

Figura 5 – Tela de Lista de Usuários

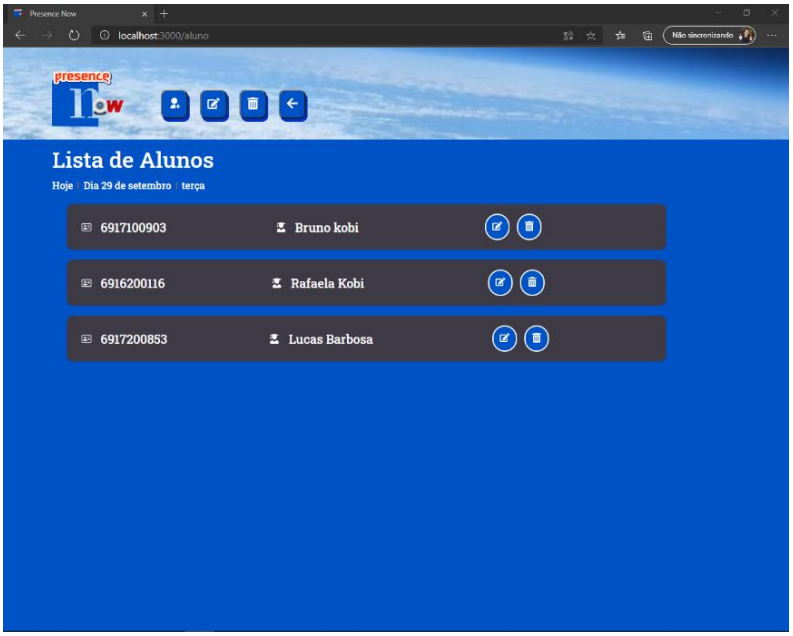


Fonte: elaboração própria

A figura 6 é a tela lista de alunos, onde o usuário é direcionado logo após clicar no botão cadastro de alunos.

A tela possui uma lista de todos os registros armazenados no banco de dados, através de um GET na rota alunos do BackEnd, essa tela também possui um menu na parte superior , onde o usuário tem a opção de cadastrar aluno, editar e voltar.

Figura 6 – Tela de Lista de Usuários



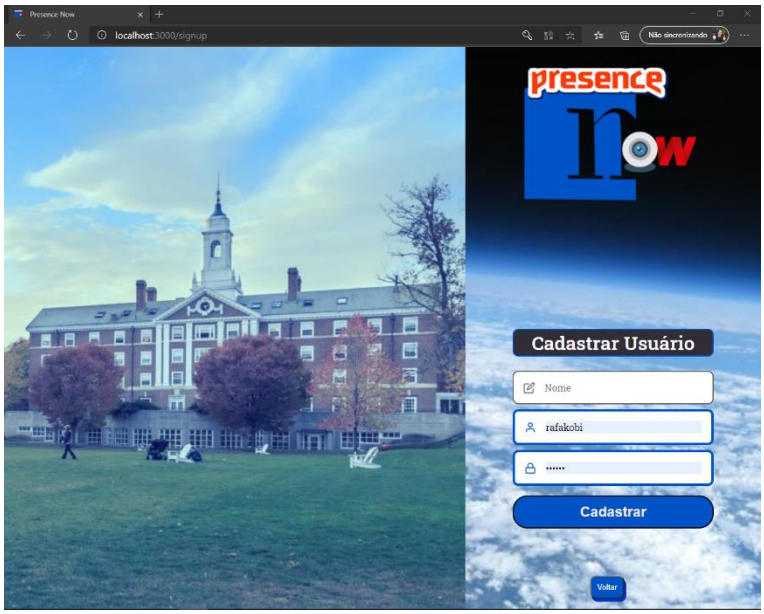
Fonte: elaboração própria

A figura 7 é a tela de cadastrar usuário, onde um usuário pode ser cadastrado na aplicação.

A tela possui um sistema de validação nos campos de preenchimento, e também possui um sistema de mensagens, relativo ao cadastro do usuário no sistema, no caso de já haver usuário cadastrado com mesmo nome.

Através de um POST na rota usuários do BackEnd, é realizado adição de um novo usuário no sistema.

Figura 7 – Tela de Cadastrar Usuário



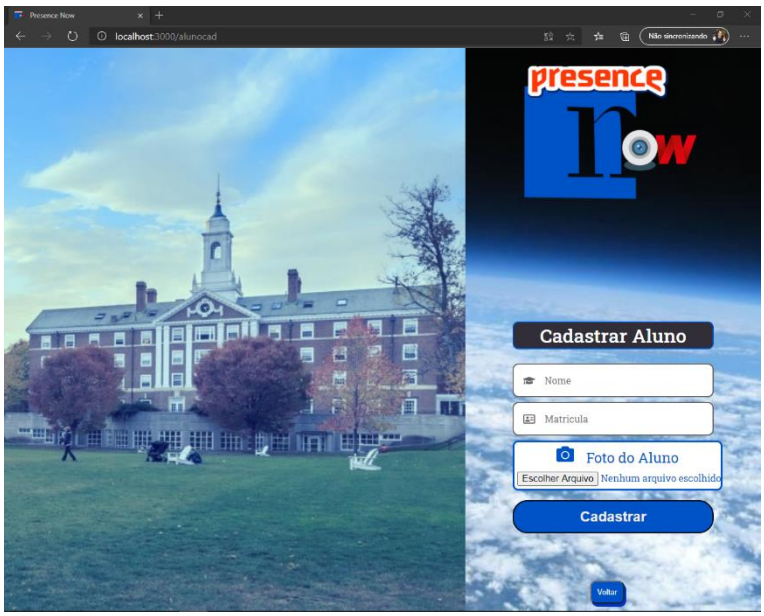
Fonte: elaboração própria

A figura 8 é a tela de cadastrar aluno, onde pode ser cadastrado um aluno na aplicação.

A tela possui um sistema de validação nos campos de preenchimento, e também possui um sistema de mensagens, relativo ao cadastro do aluno no sistema, no caso de já haver aluno cadastrado. Nessa tela também é feito o carregamento da foto do aluno através de input do tipo file, que será utilizada para fazer reconhecimento facial do aluno.

Através de um POST na rota alunos do BackEnd, é realizado adição de um novo aluno no sistema.

Figura 8 – Tela de Cadastrar Aluno



Fonte: elaboração própria

4.2.2. BackEnd

BackEnd é a parte que faz a conexão com o banco de dados Postgre, foi desenvolvida em Node.js com Typescript através do Express.js utilizado a arquitetura REST (*Representational State Transfer*) na criação da API (*Application Programming Interface)*, ele é composto de 4 rotas:

Tabela 3: Tabela de Rotas da API

Requisição	Rota	Objetivo
POST	api/autenticacao	Fazer login na aplicação
POST	api/alunos	Cadastra um aluno na aplicação
GET	api/alunos	Retorna todos alunos cadastrados
POST	api/registros	Cadastra um novo registro de presença
GET	api/registros	Retorna todos os Registros na aplicação
POST	api/usuarios	Cadastra um usuário na aplicação
GET	api/usuarios	Retorna todos usuários cadastrados

Fonte: elaboração própria

4.2.2.1. Rota Alunos

Alunos é a rota responsável por cadastrar os alunos no banco de dados, ela utiliza o Multer para fazer o upload das fotos dos aluno, salvando elas na pasta temp, renomeando as fotos, para o número de matricula dos alunos, através do serviço CriarAlunoService é feito a verificação se o aluno só possui cadastrado, caso o mesmo não esteja cadastrado, é feito a adição do aluno pelo repositório do TypeORM, que funciona como um camada de persistência dos dados no BackEnd.

4.2.2.2. Rota Registros

Registros é a rota responsável por cadastrar o registro no banco de dados, essa rota é utilizada pelo módulo de reconhecimento facial, ou seja, POST (método HTTP de envio de dados) é feito de maneira automatizada, e é feita a adição do registro pelo repositório do TypeORM no BackEnd.

4.2.2.3. Rota Usuários

Usuários é a rota responsável por cadastrar usuários no banco de dados, através do serviço CriarUsuarioService é feito a verificação se o usuário já possui cadastrado, caso o mesmo não esteja cadastrado, é feito a adição do usuário pelo repositório do TypeORM no BackEnd.

4.2.2.4. Rota de Autenticação

Autenticação é a rota responsável por verificar se o usuário e a senha estão corretos, através do serviço AutenticacaoService é feito a comparação da senha digitada, com a senha do usuário encontrada no repositório, após realizar um find no banco de dados, sendo que a senha armazenada no banco possui criptografia, fazendo necessario o uso do método compare do bcryptjs para a confirmação da autenticação do usuário.

4.2.3. Módulo de Reconhecimento

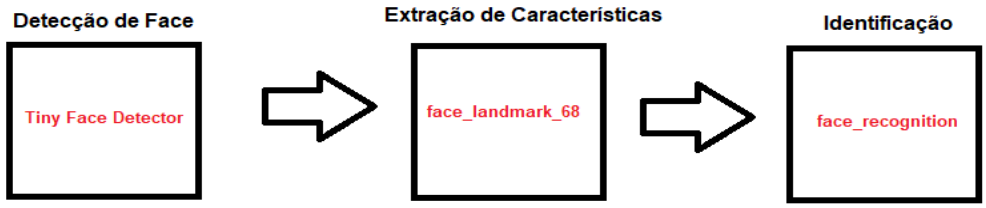
O módulo de reconhecimento foi desenvolvido em HTML, CSS e JavaScript, com objetivo de otimizar o máximo o desempenho, ao abrir o modulo é necessario especificar o local aonde o modulo foi aberto, ele faz conexão com a faceapi.js, onde são carregadas redes neurais convulsionais (CNNs) sobre o Tensorflow.js, através de dados obtidos do BackEnd é possível fazer a identificação dos alunos cadastrados na

aplicação em tempo real , e assim que essa identificação é positiva , é feito o registro do mesmo no banco de dados através da API.

Seguindo o conceito do reconhecimento facial podemos dividir em modulo em três etapas: Detecção de faces, Extração de características e Identificação /Reconhecimento.

A figura 9, exibe o esquema de reconhecimento facial, fazendo alusão as suas respectivas redes neurais da FaceApi.js

Figura 9 – Esquema de Módulo de Reconhecimento

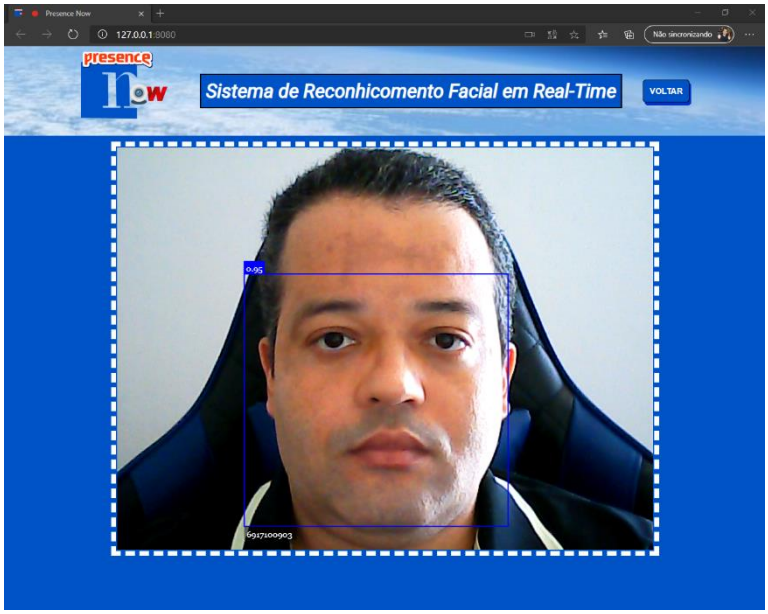


Fonte: elaboração própria

4.2.3.1. Detecção de Face

A figura 10 mostra a tela do reconhecimento, o desenho do quadrado azul na área de detecção do rosto, é criado utilizando-se da rede neural Tiny Face Detector, que detecta rosto em tempo real, com um excelente desempenho. Ela é muito mais rápida, menor e consome menos recursos em comparação com o detector de rosto SSD Mobilenet V1, por sua vez tem um desempenho um pouco inferior na detecção de rostos pequenos.

Figura 10 – Tela de Reconhecimento com Tiny Face Detector

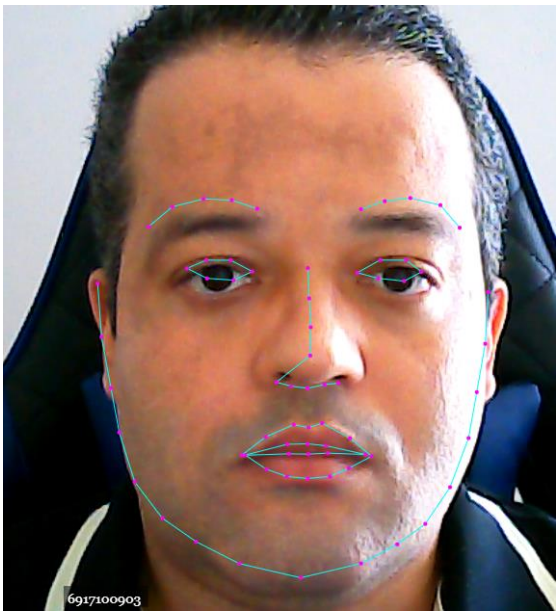


Fonte:elaboração própria

4.2.3.2. Extração de características

A figura 11, mostra o desenho obtido após a definição dos 68 pontos de referência no rosto, utilizando-se da rede neural face_landmark_68_model, na aplicação esse desenho não é exibido, mas os 68 pontos são trançados ocultamente, sendo base para última etapa do processo de reconhecimento.

Figura 11 – Tela de Reconhecimento com Face Landmark 68



Fonte:elaboração própria

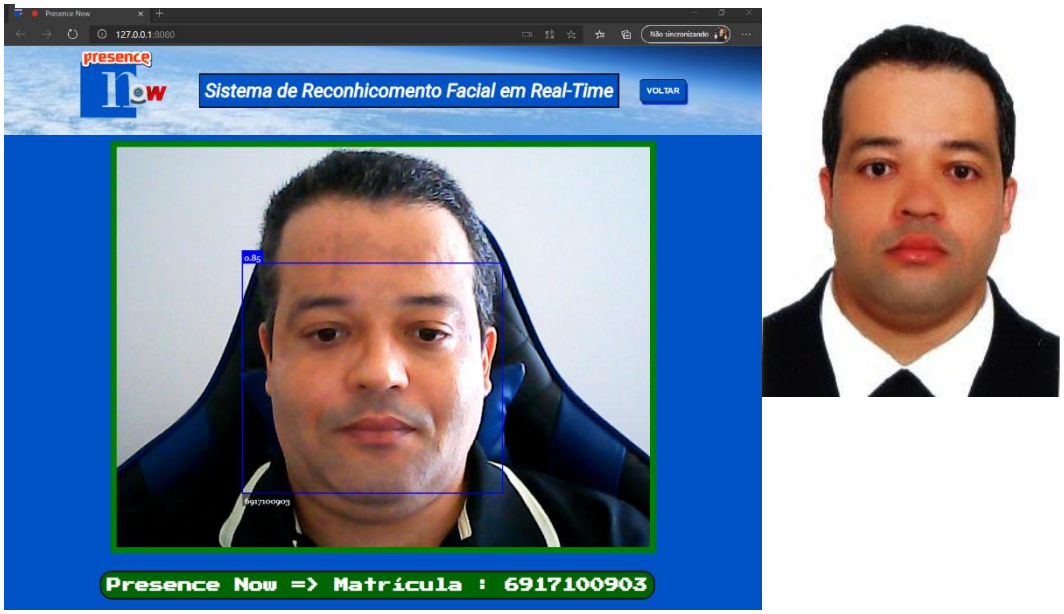
4.2.3.3. Identificação /Reconhecimento

A figura 12, mostra a tela de reconhecimento quando a identificação é positiva e a foto do banco de dados do cadastro do aluno, a rede neural face_recognition_model realiza a comparação dos 68 pontos da referência realizada pela etapa anterior pela face_landmark_68_model, e faz uma análise com as imagens dos alunos pré-carregadas do BackEnd , após essa análise é identificado o aluno que possui maior accuracy (precisão) ,e é dado um percentual de accuracy, após alguns testes observou-se que uma accuracy acima de 50%, reduziria eventuais problemas com identificações erradas por baixa accuracy, para a identificação ser considerada positiva foi determinado que a accuracy tem que ser acima de 50%.

É possível cadastrar várias fotos para o mesmo aluno, treinando a rede neural com mais faces, minimizando as falhas de reconhecimento.

Ao identificar o aluno o sistema exibe a matrícula do aluno conforme a figura 12, foi definido também o tempo mínimo para reconhecer o mesmo rosto de 10 minutos, afim de minimizar requisições ao Backend.

Figura 12 – Tela de Reconhecimento com identificação positiva / imagem do banco de dados

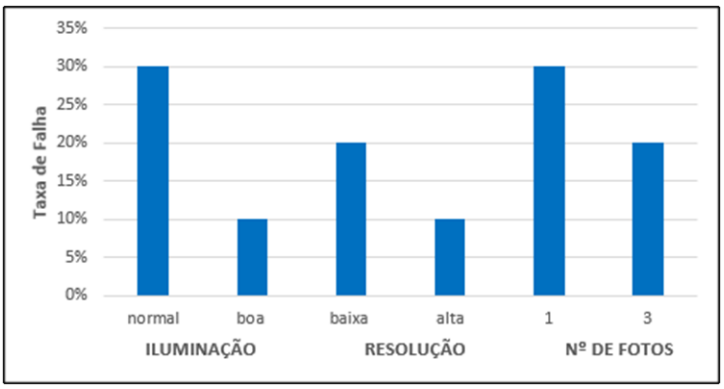


Fonte: elaboração própria

4.3. ANÁLISE PRÁTICA

A aplicação foi colocada na web através do Netlify e do Heroku, a mesma apresentou resultados satisfatórios, observou-se através de testes internos que a aplicação tem resultados melhores, com uma menor taxa de falha, com uma boa iluminação, aproximadamente 20% a menos com relação a uma iluminação normal , já com a utilização de uma webcam com resolução maior teve uma taxa de 10% a menos de falhas com relação a uma webcam comum de 12mb, já com a utilização de 3 fotos para cada aluno também obteve uma taxa de 10% a menos de falhas , com relação apenas um foto, conforme a figura 13.O teste foi baseado na observação de falhas no intervalo de 10 minutos seguidos em cada cenário, foi considerado falha, o registro de identificação incorreto , com uma base de dados com 30 alunos cadastrados.

Figura 13 – Gráfico de taxa de Falhas



Fonte: elaboração própria

5. CONCLUSÃO

Esse trabalho apresentou um software de reconhecimento facial em tempo real para automatizar o processo de chamada atual, visando minimizar a perda de tempo de aula. O trabalho se mostrou pioneiro na utilização da faceapi.js implementada direto no browser, visto que os trabalhos relacionados utilizam a implementações no backend.

O trabalho de teve como foco principalmente a entrega de uma aplicação funcional, em decremento da regra de negócio, visando realização de testes práticos de campo, para obtenção de resultados conclusivos da solução desenvolvida no trabalho, infelizmente devido a pandemia do covid-19 no mundo, houve a impossibilidade de realizar esses testes de campo, criando assim um resultado inconclusivo sobre a real eficácia do software.

Para trabalhos futuros, observou-se a necessidade da ampliação da regra de negócio do software, tendo em vista a criação de um cadastro de disciplinas e de professores, gerando registros mais detalhados, podendo haver uma ligação direta com portal da faculdade.

Com relação à segurança, a utilização de tokens de autenticação seria uma melhoria a ser implementada, com a utilização da JWT.

6. REFERÊNCIAS

ADMIN. **Primeiros passos com Express em Node.js**.2020. Disponível em: <http://nodebr.com/primeiros-passos-com-express-em-node-js/>. Acesso em outubro de 2020.

BRAGA, Luis. **Sistema de reconhecimento facial**. TCC. Escola de Engenharia de São Carlos, Universidade de São Paulo, São Paulo.2013.

BRUNS, Barbara; LUQUE, Javier. **Professores excelentes: como melhorar a aprendizagem dos estudantes na América Latina e no Caribe. Washington D.C: Banco Internacional para Reconstrução e Desenvolvimento/Banco Mundial**, 2014. Disponível em: <https://openknowledge.worldbank.org/handle/10986/20488 >. Acesso em: 16 mar. 2020.

CARLETO, D. **Face-api.js: JavaScript Face Recognition Leveraging TensorFlow.js**.2018. Disponível em < https://www.infoq.com/news/2018/11/faces-api-js/ > Acesso em setembro de 2020.

FEDOSEJEV, Artemij. **React. js Essentials**. Packt Publishing Ltd, 2015.

FILHO, F. **Professores brasileiros realizam chamada por reconhecimento facial**. 2019. Disponível em < https://olhardigital.com.br/noticia/professores-brasileiros-realizam-chamada-por-reconhecimento-facial/92046 > Acesso em outubro de 2020.

HAHN, E. M. (2016). ***Express in action.*** **Shelter Island, NY: Manning Publications Co**. EIS, Diego; FERREIRA, Elcio. *HTML5 e CSS3 com farinha e pimenta*. Lulu. com,2012.

HERON, David**. Guia completo para usar TypeORM e TypeScript para persistência de dados no módulo Node**. 2018. Disponível em: < https://www.agatetepe.com.br/guia-completo-para-usar-typeorm-e-typescript-para-persistencia-de-dados-no-modulo-node-js/>. Acesso em outubro de 2020.

LENGSTORF, J. JSON: **What It Is, How It Works, e How to Use It**. 2016. Disponível em: <https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/>. Acesso em outubro de 2020.

MACHADO, J. L. A. ***Perde-se 1/5 de uma hora aula, em média, no Brasil...*** **O que fazer?**,São José dos Campos, 06 jan. 2016. Disponível em < https://acervo.plannetaeducacao.com.br/portal/artigo.asp?artigo=2686 > Acesso em outubro de 2020.

MÜHLER, V. **Faceapi.js**. 2018. Disponível em < https://github.com/justadudewhohacks/face-api.js > Acesso em setembro de 2020.

MULTER. **Express.js**. 2020. Disponível em: <https://github.com/expressjs/multer>. Acesso em outubro de 2020.

PARDILLA Cando, A. L., & Sánchez Pilay, J. Y. 2020. Teses. **Desarrollo de una aplicación móvil prototipo para el registro y control de asistencia estudiantil en la Carrera de Ingenieria en Sistemas Computacionales basada en tecnología de reconocimiento facial**. Disponível em: <http://repositorio.ug.edu.ec/handle/redug/48932> Acesso em outubro de 2020.

PHILLIPS, P. J.; ROSENFELD, A.; ZHAO, W. e CHELLAPPA, R. **Face recognition: A literature survey. ACM Computing Surveys**, Vol. 35, No. 4, , p. 399–458, December 2003.

PRESCOTT, Preston. ***Programação em JavaScript***. Babelcube Inc., 2016.

SCHMITZ, Daniel. **Diga olá ao TypeScript e adeus ao JavaScript**. 2015. Disponível em:<https://tableless.com.br/diga-ola-ao-typescript-e-adeus-ao-javascript/>. Acesso em outubro de 2020.

SILVA, Maurício Samy. ***CSS3: desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3***. Novatec Editora, 2011.

SILVA, Maurício Samy. ***HTML 5: A linguagem de marcação que revolucionou a web***. Novatec Editora, 2019.

TensorFlow. "**Tensors**". 2019. Disponível em: <https://www.tensorflow.org/>. Acesso em outubro de 2020.

W3C. **HTML Introduction**. Disponível em: < https://www.w3schools.com/html/html_intro.asp >. Acesso em outubro de 2020

Xangai. **Una universidad china utilizará el reconocimiento facial para controlar la asistencia**. 2017. Disponivel em < https://www.efe.com/efe/america/tecnologia/una-universidad-chinautilizara-el-reconocimiento-facial-para-controlar-la-asistencia/20000036-3418428> Acesso em outubro de 2020.

APLICAÇÃO FUNCIONAL TESTE - PRESENCE NOW

Site : https://presencenow.netlify.app

usuário: admin

senha:123456

API: https://radiant-gorge-04331.herokuapp.com