UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COINT - TECNOLOGIA EM SISTEMAS PARA INTERNET
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

LUCAS NICOLÁS WENZEL VALLADARES

UTFACE: UMA API RESTFUL PARA GERENCIAMENTO DE PRESENÇA COM RECONHECIMENTO FACIAL

TRABALHO DE CONCLUSÃO DE CURSO

GUARAPUAVA
2019

LUCAS NICOLÁS WENZEL VALLADARES

UTFACE: UMA API RESTFUL PARA GERENCIAMENTO DE PRESENÇA COM RECONHECIMENTO FACIAL

Trabalho de Conclusão de Curso apresentado ao Curso de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Tecnólogo.

Orientador: Prof. Dr. Roni Fabio Banaszewski
Universidade Tecnológica Federal do Paraná

Coorientador: Prof. Me. Guilherme da Costa Silva
Universidade Tecnológica Federal do Paraná

GUARAPUAVA
2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Guarapuava
Curso Superior de Tecnologia em Sistemas para Internet

ATA DE DEFESA DE MONOGRAFIA TRABALHO DE CONCLUSÃO DE CURSO

No dia **06 de Dezembro de 2019**, às 16:00 horas, em sessão pública nas dependências da Universidade Tecnológica Federal do Paraná Câmpus Guarapuava, ocorreu a banca de defesa de Trabalho de Conclusão de Curso intitulada: **"UTFACE: Uma API RESTFUL para Gerenciamento de Presença com Reconhecimento Facial"** do(a) acadêmico(a) **Lucas Nicolas Wenzel Valladares** sob orientação do(a) professor(a) **Dr. Roni Fabio Banaszewski** do curso de Tecnologia em Sistemas para Internet.

Orientação	
Membro	**Nome**
Orientador	Dr. Roni Fabio Banaszewski
Coorientador 1	Me. Guilherme da Costa Silva

Banca Avaliadora	
Membro	**Nome**
Orientador	Dr. Roni Fabio Banaszewski
Avaliador 1	Dr. Luciano Ogiboski
Avaliador 2	Me. Paulo André Filipak

Situação
Aprovado

Guarapuava, 06 de Dezembro de 2019.

Documento assinado eletronicamente por **Lucas Nicolas Wenzel Valladares, Acadêmico,** em 06/12/2019, às 18:05, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015 .

Documento assinado eletronicamente por **Roni Fabio Banaszewski, Orientador,** em 06/12/2019, às 18:36, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015 .

Documento assinado eletronicamente por **Paulo André Filipak, Avaliador,** em 06/12/2019, às 18:40, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015 .

Documento assinado eletronicamente por **Luciano Ogiboski, Avaliador,** em 06/12/2019, às 18:42, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015 .

A autenticidade deste documento pode ser conferida no site https://tcc.tsi.pro.br/autenticidade/documentos , informando o código verificador **1575665089**.

Dedico esse trabalho aos meus tutores, desde meus pais e avós, e professores de escola até aos professores do curso. Sem eles não poderia ter chegado até esta fase de minha vida.

AGRADECIMENTOS

Agradeço a todos que me apoiaram desde o começo até o final deste trabalho principalmente aos meus familiares que me deram todo o apoio e moral possível e necessários para percorrer esta jornada, que foram meus pais e avós que sempre me motivaram e me criaram para este fim. Para o fim de batalhar por uma formação e conhecimento.

Agradeço aos que me incentivaram particularmente e me motivaram a seguir os meus sonhos e correr atrás daquilo que jamais poderá ser tirado de mim, que é o conhecimento.

Agradeço a todos meus tutores e professores que desde a infância têm me ensinado o valor de aprender e o valor que tem o conhecimento.

Agradeço ao Prof. Dr. Roni Fabio Banaszewski que tem me instruído até aqui, e tem me dado oportunidades fantásticas, e agradeço a ele por me dar a oportunidade de realizar esse projeto e adquirir mais conhecimento para a minha carreira profissional.

Agradeço também à UTFPR por fornecer este projeto com bolsa com fins de realizar pesquisas e projetos de inovação tecnológica.

Agradeço por fim e não menos importante, aos professores desta instituição e curso por me passarem seu conhecimento com tanta dedicação e amor.

Sinceramente,

Lucas Nicolás Wenzel Valladares

Eu denomino meu campo de Gestão do Conhecimento, mas você não pode gerenciar conhecimento. Ninguém pode. O que pode fazer - o que a empresa pode fazer - é gerenciar o ambiente que otimize o conhecimento. (PRUSAK, Laurence, 1997).

RESUMO

VALLADARES, L. N. W.. UTFace: Uma API RESTful para gerenciamento de presença com reconhecimento facial. 2019. 27 f. Trabalho de Conclusão de Curso – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2019.

A atividade de controle de presença em sala de aula ou em eventos é sempre uma tarefa que exige um certo tempo e a confiança de quem está efetuando o controle e também de quem está respondendo a chamada, levando em conta que podem acontecer equívocos por ambos os lados. Além disso podem haver fraudes caso uma pessoa não estiver presente e outra responder ou assinar por ela. Como solução, o controle de presença pode ser realizado com tecnologias de reconhecimento facial. Pensando nisto, um sistema que pudesse capturar uma imagem das pessoas em um ambiente e identificar os presentes, solucionaria todos esses problemas já mencionados, evitando que alunos presentes pudessem receber falta enquanto os faltantes recebessem presença. Atualmente existem vários algoritmos de reconhecimento facial de código aberto, os quais podem ser utilizados e incorporados e ajustados para uma aplicação que seja útil no cotidiano universitário. Para que o sistema seja eficiente, é necessário que seja implementada uma API RESTful que eventualmente integraria o sistema com reconhecimento facial e aplicativo para gerenciamento das funcionalidades da aplicação toda. Com o desenvolvimento dessa API, pretende-se dar início a um projeto onde traria inovação tecnológica para a Universidade Tecnológica Federal do Paraná – UTFPR e realizar melhorias e atualizações em métodos, tarefas e atividades realizadas dentro da universidade.

Palavras-chave: Sistemas de computação interativos. Sistemas de computação. Software. API para controle de presença.

ABSTRACT

VALLADARES, L. N. W.. UTFace: A RESTful API for presence control with facial recognition. 2019. 27 f. Trabalho de Conclusão de Curso – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2019.

The activity of making call attendance at class or events is always a task that takes time and trust from who is making the control and also from who is answering the calling, having in mind that can happen mistakes from both sides. Also can be frauds in case someone is not present and someone else answers or signs for her. As a solution, the presence control could be made with facial recognition technologies. Thinking about that, a system that could capture an image from the people on surroundings and identify beings would solve the problems already mentioned before, avoiding beings to receive fault and missing ones receive presence. Nowadays exist plenty of open-source facial recognition algorithms that could be used and incorporated and adjusted for an application that would be useful for university day-a-day. For an efficient system, it's necessary a RESTful API implementation that eventually could integrate the system with facial recognition and application for function management on an entire application. With the development of this API, it is intended to start a project witch would bring technological innovation for Universidade Tecnológica Federal do Paraná - UTFPR and realize improvements and updates on methods, tasks, and activities that have been realized inside the university.

Key-words: Interactive computer systems. Computer systems. Software. API for presence control.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
LBPH	Local Binary Patterns Histograms (Histogramas de Padrões de Binário Local)
API	Application Programming Interface (Interface de Programação de Aplicativos)
REST	Representational State Transfer (Transferência Representacional de Estado)
HTTP	Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)
TDD	Test Driven Development (Desenvolvimento Orientado por Testes)
PCA	Primary Component Analysis (Análise de Componentes Principais)
LS	Least Squares (Mínimos Quadrados)
LDA	Linear Discriminant Analysis (Análise Linear Discriminante)
DA	Discriminant Analysis (Análise Discriminante)
LBP	Local Binary Pattern (Padrão de Binário Local)
HOG	Histogram of Oriented Gradients (Histograma de Gradientes Orientados)
HTML	Hypertext Markup Language (Linguagem de Marcação de Hipertexto)
CSS	Cascading Style Sheet (Folha de Estilo em Cascatas)
CERN	European Council for Nuclear Research (Organização Europeia para Pesquisa Nuclear)
W3C	World Wide Web Consortium (Consórcio World Wide Web)
OpenCV	Open Source Computer Vision Library (Biblioteca Multiplataforma de Livre Uso)
IoC	Inversion of Control (Inversão de Controle)
DI	Dependency Injection (Injeção de Dependência)

SUMÁRIO

1 INTRODUÇÃO

Há anos a Inteligência Artificial (IA) representava uma tecnologia totalmente futurística retratada em filmes e livros como algo que fazia parte do cotidiano das pessoas. Até pouco tempo atrás era vista por muitas pessoas como uma ficção e que o avanço tecnológico não avançaria até os níveis em que se encontra no mundo nos dias de hoje. No entanto, a IA já tem se tornado realidade em nossas vidas e é possível encontrá-la em muitas aplicações, com algoritmos capazes de aprender, reconhecer padrões e até tomar decisões de forma espontânea. Entre esses algoritmos, estão os algoritmos de reconhecimento facial. Essa tecnologia já é muito utilizada em países desenvolvidos para detecção de criminosos ou até mesmo para localizar pessoas desaparecidas.

A tecnologia de reconhecimento facial tem também grande potencial de aplicação em universidades. Pode ser utilizada para realização de controle de presença e frequência em sala de aula e eventos tais como palestras, ou até mesmo para controlar acesso em áreas restritas. Com esta tecnologia os administradores dessas tarefas, certamente teriam suas vidas facilitadas, tendo mais agilidade na realização das mesmas, sem ter que interromper o horário de aula, por exemplo, para fazer a chamada de presença, tendo assim mais tempo para a atividade principal que é a aula em si. Outra razão pela qual a implementação desta tecnologia pode ser justificada, é que seriam evitados tanto fraudes como enganos no controle de presença tradicional via lista de chamada, onde só se pode contar com a boa fé de que seja o próprio aluno assinando o seu nome. Também desta forma seriam evitados equívocos ou enganos que poderiam ser provocados por falta de atenção, acontecendo a atribuição de falta para um aluno presente ou o contrario, a atribuição de presença a um aluno não presente.

Algumas tecnologias de IA ainda são muito custosas por demandar equipamentos de alta qualidade e tecnologias proprietárias, o que dificulta a popularização da aplicação das mesmas. Porém, ao buscar a utilização de tecnologias de acesso livre, com a adaptação das mesmas e com estudo sobre o assunto, é possível conseguir uma solução mais barata para aplicação efetiva da IA no cotidiano das pessoas. Com isso, por exemplo, a aplicação de tais tecnologias pode impactar positivamente a produtividade no âmbito universitário.

1.1 OBJETIVOS

Nesta seção serão apresentados o objetivo geral e objetivos específicos do presente trabalho.

1.1.1 Objetivo Geral

O objetivo deste trabalho é desenvolver uma API RESTful para ser aplicada em uma sistema com reconhecimento facial para controle de presença em sala de aula.

1.1.2 Objetivos Específicos

- Estudar algoritmos de reconhecimento facial de código aberto: EigenFaces, FisherFaces e LBPH (JavaCV);

- Desenvolver uma API RESTful para gerenciar controle de presença em sala de aula;

- Testar rotas e funções REST via Postman e banco de dados local;

- Incluir documentação Swagger para a API;

- Fazer deploy da API no Heroku para disponibilização online;

- Popular banco de dados Heroku para testes da aplicação online e apresentação do projeto.

1.2 ORGANIZAÇÃO DO TRABALHO

Este documento está organizado da seguinte forma: na Seção 2.1 são apresentados os objetivos gerais e específicos do projeto. A Seção 2.2 descreve o diferencial tecnológico. A Seção 3 está descrita a resenha literária. A Seção 3.1 traz o estado da arte. A Seção 3.2 apresenta a fundamentação teórica para o projeto. A Seção 4 apresenta a metodologia para o mesmo. Na Seção 5 o desenvolvimento. Na Seção 6 as considerações finais e por fim na Seção 7 estão listadas as referências.

2 RESENHA LITERÁRIA

Neste capítulo serão apresentadas aplicações que utilizam algoritmos e API para reconhecimento facial. Essas APIs pertencem às empresas as quais utilizam o reconhecimento para suas aplicações, portanto são APIs de acesso restrito. Também serão apresentados os algoritmos estudados. O estudo dos algoritmos de reconhecimento facial para este projeto, dá-se pela importância que eles têm para o funcionamento e desenvolvimento da API. Por fim, serão apresentadas as tecnologias para o desenvolvimento da API proposta neste trabalho.

2.1 Estado da Arte

2.1.1 FindFace

É um aplicativo desenvolvido na Rússia que usa a tecnologia de reconhecimento de faces de pessoas. O aplicativo funciona de forma que ao fotografar uma pessoa, a imagem é comparada com os rostos das redes sociais com uma precisão de 70 por cento. Segundo as estatísticas de 2016, o aplicativo já tinha conseguido identificar 500 mil usuários e processado cerca de 3 milhões de buscas. Desde o dia 1 de Setembro de 2018, a empresa finalizou seus serviços com esta aplicação e passou a prestar serviços de reconhecimento facial para o governo (LEANDRO, 2019). A empresa oferece para aqueles que querem utilizar o sistema, um sistema similar chamado FindFace.PRO.

Figura 1 – Logo FindFace

Fonte: (FINDFACE, 2018)

2.1.2 Face++

É uma plataforma chinesa de extrema qualidade que superou a precisão dos algoritmos do Facebook e da Google. Ela já está sendo utilizada nas universidades para realização das

chamadas presenciais (ELOLA, 2018). A plataforma permite a comparação de duas faces, buscar uma face no banco de dados da plataforma com base em uma imagem que tenha sido colocada para fazer a busca e cada face detectada é armazenada para futuras análises. Todas as funções do sistema retornam uma pontuação e limites de confiabilidade das comparações.

Figura 2 – Logo Face++

Fonte: (FACE++, 2018)

2.1.3 CredDefense

É uma plataforma brasileira de consignação de crédito. Os dados biométricos são fornecidos pelo próprio cliente e mantidos pela empresa que mantém a plataforma. A empresa se compromete a gerar códigos criptografados das características faciais de seus clientes e armazená-los em base de dados, a fim de evitar fraudes e realizar autenticação em transações (SULIVAN, 2018).

Figura 3 – Logo Creddefense

Fonte: (CREDDEFENSE, 2018)

2.1.4 TÓV-TEC Expert

Esta é uma plataforma brasileira para controle de acesso à áreas restritas. O Expert Fixo funciona com uma câmera instalada na entrada da localidade, como catracas ou barreiras de acesso, que faz o reconhecimento da face e compara com seu banco de dados de faces cadastradas para aquela área e caso seja confirmado, é liberada a passagem para essa pessoa (TÓV-TEC, 2017).

Figura 4 – Logo Tóv-Tec

Fonte: (TóV-TEC, 2018)

2.2 Fundamentação Teórica

Nesta seção serão apresentadas as tecnologias que serão utilizadas para implementação da API que será responsável por realizar o reconhecimento facial. Com o intuito de obter maior conhecimento teórico, um estudo foi realizado sobre as linguagens de programação, banco de dados e linguagens de marcação. Este estudo também contribui para uma visão mais concreta para o projeto e para conhecer como ele será colocado em prática. Essa mesma metodologia de estudo se aplica aos algoritmos de reconhecimento facial de livre uso, levando em conta que é necessário saber como eles funcionam para que a API seja desenvolvida.

2.2.1 Algoritmos de Reconhecimento Facial

Nesta sessão serão apresentados algoritmos correlatos de reconhecimento facial de livre uso que são utilizados atualmente no mundo todo, entre eles o Eigenfaces que é o algoritmo mais antigo de reconhecimento facial (BISSI, 2018), a fim de adquirirmos um comparativo entre os algoritmos e embasamento para utilização dos mesmos. Também são abordadas, em estado da arte, algumas aplicações que utilizam reconhecimento facial.

2.2.1.1 EigenFaces

A prática de realizar o reconhecimento facial consiste em isolar entradas ou dados de uma imagem em várias classes, as quais podem se referir a diferentes pessoas. Essas entradas

podem apresentar ruídos, os quais podem ser causados por diferentes condições de iluminação, posição da pessoa, sombras, entre outros. No entanto, apesar das diferenças entre uma imagem e outra, existem padrões quando se leva em conta imagens de uma mesma pessoa. Esses padrões podem ser observados em todos os sinais de entrada, como a presença de olhos, nariz, boca, bem como certas características de uma face, e também, as distâncias relativas destes membros que são caracterizados como objetos na imagem. Essas características são chamadas de "eigenfaces"no domínio de reconhecimento facial. Essas características são extraídas da imagem por uma ferramenta chamada Análise de Componentes Principais (PCA) (Z.; LI, 2011).

Por meio da PCA é possível extrair essas características de todo o conjunto de imagens de treinamento e formar, por meio destas, uma base de reconhecimento, como uma imagem média de todas essas características, que juntas, somadas compõem uma imagem original da pessoa em questão. Cada face é um valor, e cada característica tem um valor próprio, e para que a face esteja completa, todas as características somadas devem chegar ao valor da face ou ao menos bem próximo para ser reconhecida como aquela face. Deve-se levar em conta que ao juntar todos os valores ou pesos dessas características ou eigenfaces, alguns dados podem ser perdidos na composição da imagem. Neste caso, a formação da imagem não seria perfeitamente igual à imagem original.

Duas coisas devem ser levadas em consideração. A primeira delas é que quando uma imagem é desmanchada por assim dizer, e os pesos das características principais de um rosto forem muito diferentes dos padrões, aquele objeto não será reconhecido como uma face. A segunda consideração é que quando temos faces muito similares, os pesos de suas características serão muito similares. Portanto, estas poderão ser consideradas como a face de uma mesma pessoa. Isto pode gerar uma margem de erro se duas pessoas forem muito parecidas fisicamente.

O algoritmo é chamado de eigenfaces pois ele utiliza de eigenfaces (as características principais de uma imagem) para realizar a detecção facial (BELHUMEUR; HESPANHA; KRIEGMAN, 1997). Ele se comporta da seguinte forma: primeiro, as imagens coletadas do conjunto de treinamento são transformadas em um conjunto de eigenfaces. Então, cada imagem será dividida em pesos para suas eigenfaces e armazenadas em um arquivo. Ao detectar uma face em uma imagem desconhecida, os pesos são calculados para suas eigenfaces e então comparadas com os pesos que foram anteriormente gravados no arquivo.

A maneira que são comparados esses pesos é por distância de valor entre os pesos, pois cada imagem, será dividida em pesos distintos. Portanto, seria quase impossível ter duas imagens distintas de uma mesma face e obter pesos totalmente similares. Assim, a distância que for menor em relação à base de pesos gravados, corresponderá à face que está sendo detectada. Quando os valores detectados forem muito distantes de qualquer valor em base de dados, a mesma não será considerada como uma face.

2.2.1.2 FisherFaces

Um problema corrente, quando se trata de visão computacional, reconhecimento de padrões e até mesmo aprendizado de máquina, é a definição da representação dos dados apropriados para realização de determinadas tarefas. Assim como o algoritmo EigenFaces, o algoritmo FisherFaces utiliza o PCA para dividir uma imagem total em diferentes partes que representam a maior parte de variação de dados em uma imagem, produzindo assim, um conjunto de eigenfaces (BELHUMEUR; HESPANHA; KRIEGMAN, 1997). Esses autovetores detectados são comparados com os autovetores da matriz obtida pelo treinamento das imagens, assim, esses autovetores são correspondentes à solução de mínimos quadrados (LS). O LS é uma excelente maneira de representação de dados, pois garante que a variação de dados entre uma imagem e outra seja mantida, eliminando assim as características originais de uma face e focando nas partes de maior variação.

Quando o objetivo se trata de classificação e não de representação, a solução LS pode não produzir os resultados esperados. Nesses casos, deseja-se encontrar um subespaço que mapeie os vetores de amostra da mesma classe em um único ponto da representação do recurso e nos de classes diferentes que estão o mais distantes um do outro quanto possível. As técnicas derivadas para atingir esse objetivo são conhecidas como análise discriminante (DA).

A DA mais conhecida é a Análise Linear Discriminante (LDA), que pode ser derivada de uma ideia sugerida pela R.A. Fisher em 1936 (BELHUMEUR; HESPANHA; KRIEGMAN, 1997). Quando o LDA é usado para encontrar a representação do subespaço de um conjunto de imagens de faces, o resultado desses vetores de base que definem esse espaço são conhecidos como Fisherfaces (Z.; LI, 2011).

Para computar os Fisherfaces, assume-se que os dados em cada classe são normalmente distribuídos. Faz-se então uma distribuição normal multivariada, definindo uma matriz média e a covariância, e define-se uma função de densidade de probabilidade.

Em síntese, o FisherFaces é definido pela maior quantidade de variação de uma face, trabalhando assim, não com as características padrões de uma face e sim naquilo que varia entre uma face e outra, utilizando de comparativos de distância entre matrizes.

2.2.1.3 LBPH

O LBPH ou (Local Binary Pattern Histogram) tem sua origem do LBP (Local Binary Pattern) que é um padrão de binário local, um tipo de característica utilizado para classificações. O LBP consiste em uma textura em modelo de espectro que foi proposto em 1990 (WANG; HE, 1990). Finalmente, em 1994 (OJALA; PIETIKäINEN; HARWOOD, 1996), esse padrão foi publicado e a partir desse momento foi descoberta uma das melhores formas de classificação para texturas e também foi concluído que a combinação com HOG (Histogram of Oriented Gradients) levava a uma notável melhora de desempenho em detectar os dados (WANG; YAN; HAN, 2009).

Basicamente, o algoritmo funciona gerando uma matriz, onde cada posição guarda o valor dos pixels dessa parte da imagem. Então, é feita uma comparação entre eles utilizando de vizinhança circular e utilizando uma equação para formar uma matriz binária. A partir dessa matriz, é gerado um histograma que é utilizado para realizar o reconhecimento facial, utilizando os valores de vizinhança.

Para formar um histograma é possível utilizar diferentes critérios de LBP como invariante à rotação ou uniforme, que desta forma pode gerar uma maior discriminação em comparações aos histogramas de padrões individuais. Desta forma, a frequência em que os padrões do LBPH ocorrem é muito rara, então as probabilidades não poderiam ser confiadas (PARRA, 2014).

2.2.2 Tecnologias de Desenvolvimento

2.2.2.1 Linguagem de Marcação HTML, Linguagem de Estilização CSS e Linguagem de Programação JavaScript

A sigla HTML (Hypertext Markup Language) significa Linguagem de Marcação de Hipertexto. Isto quer dizer que é uma linguagem feita para interpretação de textos com marcações, ou seja, para criação de páginas web que serão interpretadas pelos navegadores. Hoje em dia, os navegadores mais utilizados são Google Chrome, Mozila Firefox, Microsoft Edge (substituiu o Internet Explorer), Safari e Opera (ALVES, 2016).

Quando se desenvolve para internet, é comum a utilização de várias linguagens para criar uma aplicação ou website. Para que todas essas linguagens funcionem juntas em uma única aplicação, é necessário separar os códigos em camadas. Como exemplo, o HTML é usado para estruturação, o CSS para estilo e aparência e o JavaScript para tratar eventos e manipular dinamicamente a estrutura e aparência de uma página. Também é possível utilizar linguagens para cuidar de toda a parte de servidor, banco de dados e também de toda a parte lógica do sistema, como por exemplo cálculos. Exemplo destas linguagesm são: PHP, Python, Ruby, Java, C, C++, etc.

O HTML foi criado por Tim Berners-Lee em 1991 no CERN (European Council for Nuclear Research) na Suíça (CASTRO; HYSLOP, 2012). O HTML foi criado para integrar instituições de pesquisas próximas e compartilhar documentos com facilidade. Em conjunto com a biblioteca de desenvolvimento WWW (World Wide Web), que foi liberada em 1992, o HTML alcançou proporção mundial.

A linguagem CSS (Cascading Style Sheet) é definida pela World Wide Consortium (W3C) como folha de estilo em cascata. Essa definição diz que CSS é um mecanismo para adicionar estilo às páginas algum estilo, como por exemplo fontes, espaçamentos, cores, etc (CASTRO; HYSLOP, 2012).

O JavaScript é uma linguagem de programação (HAVERBEK, 2018) que permite com que você implemente partes mais complexas de uma aplicação web no lado cliente e

também no servidor. Normalmente, é utilizado para tratar eventos e ações de usuários no lado cliente. Basicamente, o JavaScript é responsável pela maioria dos elementos dinâmicos de uma aplicação, ou seja, informações que se atualizam continuamente, como controle de multimídia, imagens animadas, gráficos, etc. O diferencial do JavaScript é que pode ser utilizado diretamente no código HTML apenas colocando o conteúdo dentro de uma tag "script".

2.2.2.2 OpenCV e JavaCV

O OpenCV (Open Source Computer Vision Library) é uma biblioteca multiplataforma de código aberto desenvolvida pela Intel para desenvolvimento na área de visão computacional (CULJAK et al., 2012). Ela provê uma infraestrutura para as aplicações nessa área e também acelera a percepção de máquinas para produtos, utilizando de módulos de processamento de imagem e vídeo. Ela possui mais de 2500 algoritmos otimizados de visão computacional e aprendizado de máquina para uso no reconhecimento de objetos ou pessoas, análises estruturais, calibração de câmera, filtros de imagem, e inclusive para extração de modelos 3D de objetos.

Os algoritmos da OpenCV são utilizados por várias grandes empresas e startups. Estes podem ser utilizados em diferentes aplicações. Por exemplo, uso na vigilância pelo governo de Israel; monitoramento de equipamentos de mineração na China; detecção de acidentes de afogamentos em piscinas na Europa e até mesmo reconhecimento e marcação de produtos no Japão (ELOLA, 2018).

Para utilização de tais algoritmos com Java, existe uma extensão da biblioteca OpenCV que se chama JavaCV. Esta contêm classes facilitadoras para programação em diversas plataformas, inclusive para Android. O JavaCV também contêm aceleradores de hardware para exibição de imagem em tela cheia utilizando canvas, e entre outros aprimoramentos para se trabalhar com imagens. (DAVISON, 2013).

2.2.2.3 API Restful

Uma API (Apllication Programming Interface) é conjunto de padrões e rotinas que são estabelecidos e documentados para que seja utilizada por uma aplicação (PIRES, 2017). Não é necessário que aplicação conheça de fato os detalhes da implementação do software, utilizando apenas seus métodos a fim de integrar dados entre diferentes fontes e softwares ou aplicações e até mesmo usuários.

Uma API é considerada RESTful quando atende os princípios de REST (Representational State Transfer). Basicamente, REST é a abstração de uma arquitetura e comunicação utilizada no desenvolvimento web, que permite a criação de uma interface bem definida permitindo que as aplicações consigam se comunicar de forma simples. (RICHARDSON, 2013).

2.2.2.4 Spring

Spring é um framework Java de código aberto (open source), criado por Rod Johnson, em 2003 (JOHNSON, 2003), que foi criado no intuito de auxiliar os programadores Java a desenvolverem seus sistemas, sem terem que se preocupar demasiadamente com configurações avançadas. O framework utiliza de Inversão de Controle (IoC) e Injeção de Dependências (DI), a fim de facilitar o desenvolvimento de aplicações. O Spring fornece módulos de persistência de dados, integração, segurança e testes, entre outros, que são muito mais fáceis de compreender e implementar em uma aplicação.

A IoC permite que seja transferido a outro elemento o controle de quando e como um objeto deve ser criado ou instanciado ou quando um método deve ser chamado e executado. Ao utilizar o Spring, o programador não deve mais se preocupar em gerenciar isto, transferindo essa responsabilidade ao elemento que é chamado de "container". Na prática, isso pode ser aplicado, por exemplo, na interface do usuário onde não será mais necessário ficar escutando um componente para ser disparada uma ação do usuário, basta implementar diretamente a ação para os botões. A DI é uma das formas de implementar a IoC, deixando com que a DI cuide das dependências (EFRAIM, 2012).

2.2.3 Documentação Swagger

Swagger é uma poderosa ferramenta desenvolvido pela SmartBear Software, de pouca dificuldade de uso para desenvolvedores de API, que permite ao desenvolvedor construir sua API desde o design e documentação até os testes e implantação da mesma. O Swagger é uma mistura de ferramentas de código aberto, gratuitas e comercialmente disponíveis para que qualquer pessoa possa desenvolver sua API sem dificuldades (SMARTBEAR, 2019).

Com o Swagger é possível deixar a API mais interativa, de forma que o usuário possa visualizar todos os métodos disponibilizados pela API. Com um arquivo de configuração podemos incluir em nossa API a documentação Swagger que irá demonstrar as rotas, métodos, parâmetros, formatos json que devem ser submetidos para a API.

3 METODOLOGIA

Este capítulo apresenta a metologia a ser utilizada para o desenvolvimento do projeto. A metologia será dividida em etapas que serão abordadas e descritas a seguir:

3.1 Levantamento dos requisitos do sistema

Primeiramente, para embasar a construção dos artefatos referente a modelagem do projeto, os requisitos foram coletados utilizando a técnica de brainstorming com a presença do Professor orientador, elaboração das histórias dos usuários e estudo dos algoritmos de reconhecimento facial para determinar os métodos e funcionalidades que uma aplicação deve ter para utilizar as funções do reconhecimento facial. Também, para facilitar a detecção de tais requisitos, foi utilizado a técnica de prototipação de telas de um aplicativo a ser construído em trabalhos futuros para melhor visualização e entendimento do escopo do problema.

3.2 Modelagem do sistema

Após o levantamento de requisitos, deu-se a construção da arquitetura do sistema, basicamente no lado do servidor. A arquitetura permite antever como será feita a integração de dados, aonde serão armazenados os dados e como serão processados. Neste momento, projeta-se também o modelo da base de dados. Com isso, consegue-se também projetar a própria API RESTful.

3.3 Estudo e definição das tecnologias a serem utilizadas

Para o desenvolvimento do projeto, também é necessário adquirir conhecimento sobre as tecnologias que serão utilizadas. Desta forma, se faz necessário o estudo e aprofundamento em conceitos de frameworks e dos algoritmos de reconhecimento facial.

3.4 Desenvolvimento do sistema e testes do sistema

Nesta fase dá-se início ao desenvolvimento da API, acompanhado de testes manuais para verificar o funcionamento das rotas e métodos desenvolvidos. Os testes são feitos subindo a aplicação em servidor local, com base de dados local e testada as rotas via Postman submetendo ou retornando o JSON dos objetos a serem testados.

4 DESENVOLVIMENTO

Neste capítulo serão apresentadas as funcionalidades e resultados alcançados com o desenvolvimento da API UTFace. A API foi desenvolvida para que futuramente possa ser inserida em uma aplicação que permita com que os professores façam o controle de presença dos alunos em sala de aula a partir do reconhecimento das faces dos presentes. Para isso a API deve ser inteiramente programada para atender as funções da aplicação sendo necessário que a API esteja preparada para submeter uma imagem para o servidor para que possam ser reconhecidos os alunos presentes, e já atribuir presença, e também atribuir falta aos alunos não presentes.

Para que a chamada de presença possa ser efetuada com sucesso, primeiro deve ser criada uma disciplina, e nesta, adicionado um professor que irá efetuar o controle de presença. Os alunos devem estar todos cadastrados e adicionados na turma. Para adicionar os alunos nas disciplinas, será necessário cadastrar as aulas e vincular as disciplinas aos alunos, desta forma já poderá ser atribuída a presença do aluno.

Para a realização do reconhecimento facial, são necessárias três etapas, que são elas: a captura, que seria a coletânea de imagens de uma pessoa, o treinamento dessas imagens e por fim o reconhecimento. Os alunos cadastrados também terão suas fotos, que deverão ser submetidas, por meio da API, ao servidor para que o algoritmo execute o treinamento. Apenas desta forma, cada aluno poderá ser reconhecido quando for submetida uma imagem dos alunos em sala de aula. As imagens dos alunos deverão ser hospedadas online.

A API será responsável por submeter os dados de status do aluno de presente ou ausente, ao submetermos via API uma imagem, que encaminhará ao servidor para executar o reconhecimento facial. Levando em conta que o treinamento gera apenas um arquivo para todas as imagens, inclusive de pessoas diferentes, então será um processo que custará processamento, e quanto mais imagens, mais processamento, por tanto, o treinamento inicial será feito uma única vez. Existe a possibilidade de um aluno submeter novas fotos pelo seu próprio aplicativo, caso isso aconteça, essas imagens serão armazenadas, e um "job" executará o processo de treinamento semanalmente em horário que o sistema não estiver em funcionamento, assim evitando um maior custo de processamento, como seria o caso de executar o treinamento das imagens cada vez que uma pessoa submeter uma nova imagem.

4.1 Análise do Sistema

Nesta seção será apresentada a documentação referente a coleta de requisitos que dá embasamento para as funcionalidades do que foram propostas no sistema.

Para coletar informações e projetar a ideia, se fez necessário realizar a prototipação das telas de como ficaria uma aplicação final utilizando a API que foi proposta a ser desenvolvida.

A prototipação das telas será utilizada apenas para levantamento de requisitos. Também foi realizado um brainstorming para determinar as funcionalidades do sistema e a partir deste brainstorming foram criadas histórias com o levantamento de requisitos, conforme a metodologia Scrum.

ID	Histórias
1	Como aluno, gostaria de receber meu status de presença após ser efetuado o controle de presença.
2	Como professor, gostaria de submeter uma imagem para efetuar o controle de presença em uma aula de uma disciplina.
3	Como professor, gostaria de verificar as disciplinas as quais leciono.
4	Como professor, gostaria de verificar os alunos das turmas das disciplinas as quais leciono.
5	Como secretária, gostaria de cadastrar e efetuar o controle de usuários.
6	Como secretária, gostaria de cadastrar e efetuar o controle de alunos.
7	Como secretária, gostaria de atualizar os dados de um aluno e adicionar novas fotos.
8	Como secretária, gostaria de cadastrar e efetuar o controle de professores.
9	Como secretária, gostaria de cadastrar e efetuar o controle de disciplinas.
10	Como secretária, gostaria de cadastrar e efetuar o controle de aulas.

Tabela 1 – Tabela de Histórias

4.1.0.1 Aplicação na visão do professor

Na figura 5 é apresentada o protótipo de uma tela onde mostraria as disciplinas cadastradas para um determinado professor, e apenas essas, inicialmente, seriam as disciplinas que este mesmo professor poderia realizar a chamada presencial.

Figura 5 – Protótipo tela professor

Na segunda tela ainda da figura 5 está representada como o professor irá realizar a chamada presencial, podendo tirar uma foto ou até mesmo fazendo manualmente e então submetendo essa lista com o status do aluno em presente ou ausente.

Inicialmente existe uma lista de todos os alunos que foram matriculados na disciplina, e todos estão com o status de "ausente". A partir do momento que o aluno é reconhecido em uma foto submetida pelo professor, o status desse aluno passa a ser de "presente", podendo o professor ainda assim desmarcar ou marcar algum aluno que desejar, em caso de falha.

Na terceira tela da figura 5 está sendo representado o menu lateral do aplicativo onde constam as informações do professor e das disciplinas que ele pode realizar uma chamada presencial.

4.1.0.2 Aplicação na visão do administrador

Na figura 6 é ilustrado como seria o aplicativo na visão de um administrador, que poderá ter a função de cadastrar uma disciplina e incluir os alunos à mesma. O administrador poderá também submeter fotos de um aluno ao cadastrá-lo no sistema, ou até mesmo atualizar seu cadastro submetendo novas fotos.

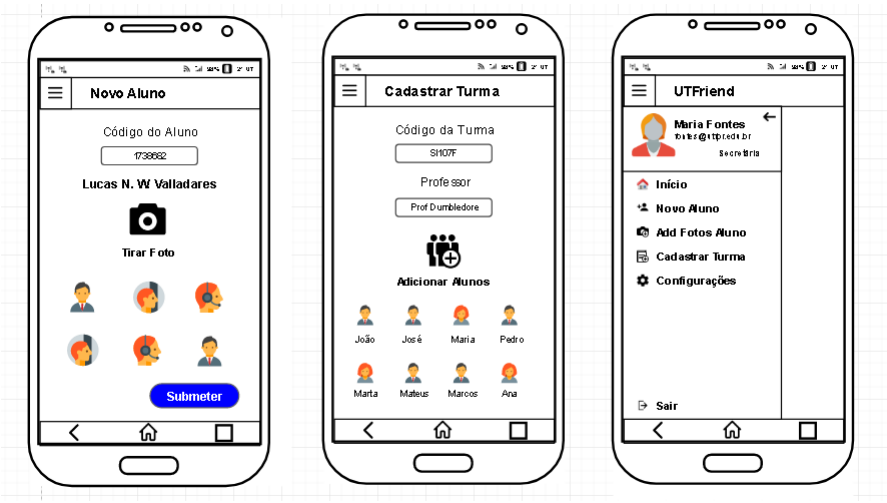

Figura 6 – Protótipo tela administrador

Na primeira tela da figura 6, é representado o cadastro de um aluno no sistema. Enquanto na segunda tela está representado o cadastro de uma disciplina e já a inclusão do professor responsável pela turma e os alunos que deverão participar da disciplina.

A terceira tela da figura 6, representa o menu lateral contendo as informações do administrador e as funções que podem ser realizadas pelo mesmo.

4.2 Projeto do Sistema

Nesta seção serão apresentados os artefatos resultantes do projeto do sistema. Os artefatos serão divididos em Projeto da API, Projeto da Base de Dados da API e Projeto de

Arquitetura da Estrutura da Rede.

4.2.1 Projeto da API

Na figura 7, é demonstrado como será a documentação da API e suas funções REST. Apenas como exemplo, foi utilizado apenas um dos modelos das entidades que irão existir na API.

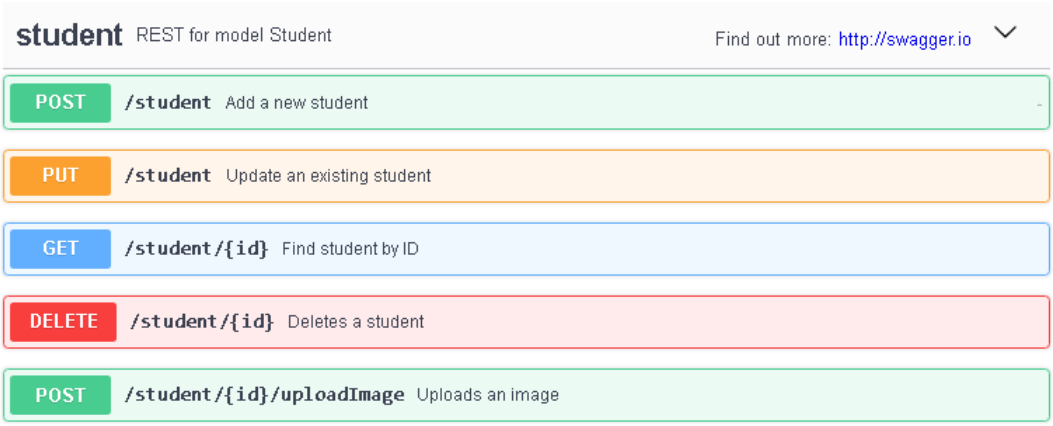

Figura 7 – Protótipo requisições REST

A API será documentada com o Swagger (SWAGGER, 2019), desta forma os usuários poderão facilmente utilizar a documentação para acessar as funções da API e até mesmo ver as entidades e os tipos de atributos que existem nas entidades, como pode-se observar na figura 8.

4.2.2 Projeto da Base de Dados da API

Na Figura 9 está representada a Base de Dados, demonstrando o "schema"da base de dados e as ligações entre as tabelas, que é equivalente aos modelos de entidades demonstrados na figura 8.

A tabela de curso, representa os diferentes cursos que uma instituição pode ter, cada curso pode ter vários professores e várias disciplinas, enquanto os professores e disciplinas terão apenas um curso.

Um professor pode ter várias disciplinas, no entanto, cada disciplina pode ter apenas um professor, sendo assim uma relação de (1, n).

As disciplinas terão uma relação com aulas, pois assim, deixamos em aberto para que um evento possa ser considerado uma aula, e possa ser relacionada aos alunos, contendo um atributo para armazenar o status de presença de cada aluno.

A tabela de alunos também tem uma relação de (1, n) com a tabela de fotos. Um único aluno poderá ter várias fotos, no entanto, uma foto sempre irá pertencer a um único aluno.

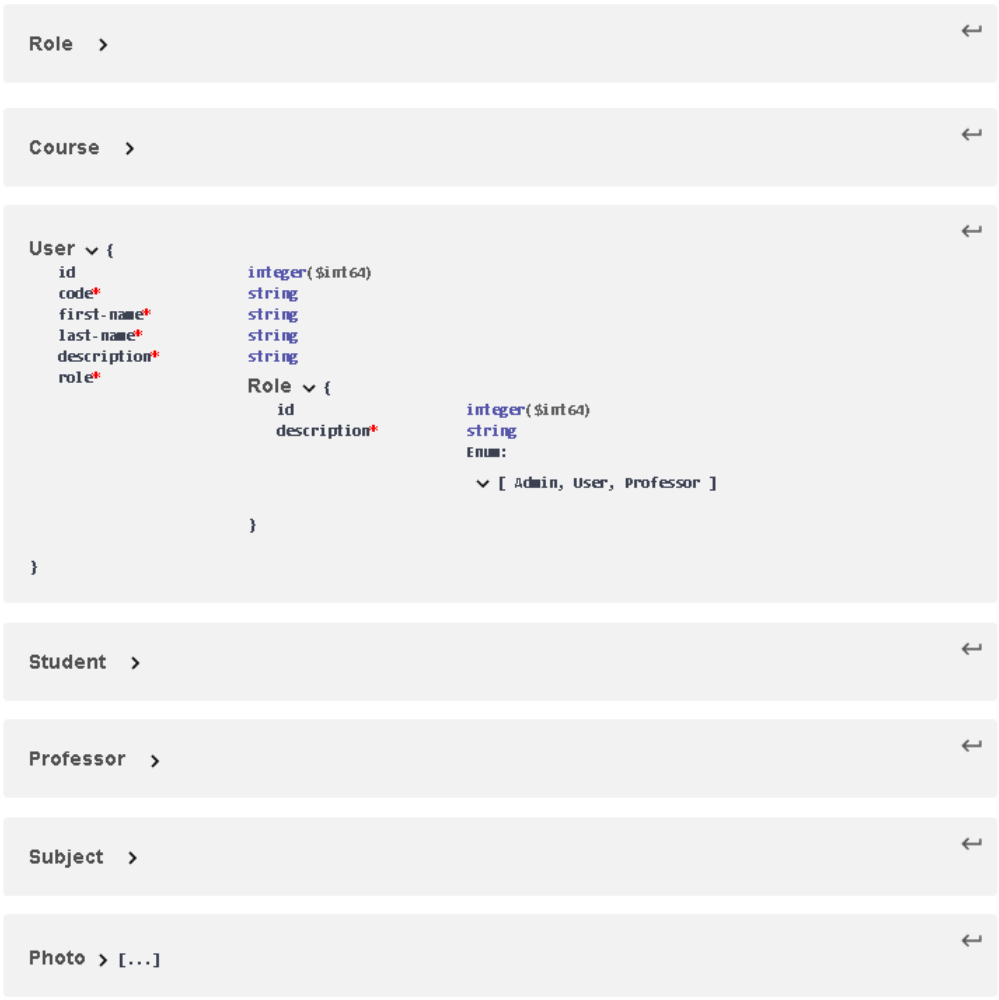

Figura 8 – Protótipo entidades na API

Um usuário poderá ser um administrador ou uma secretária. A tabela de usuário receberá um papel, que pode ser um papel de administrador, como mencionado. E a tabela de papel, também será relacionada aos professores e aos alunos.

4.2.3 Projeto de Arquitetura da Estrutura da Rede

Por meio de requisições HTTP serão submetidas imagens para treinamento ao servidor, e a API irá realizar o tráfego de dados e irá fornecer toda a documentação para que os métodos possam ser utilizados sem dificuldades. Os modelos das requisições HTTP e o corpo do JSON que devem ser submetidos poderão ser checados no Swagger (SWAGGER, 2019), plataforma que será utilizada para documentar a API.

Como representado na Figura 10, as imagens serão enviadas a partir de um aplicativo mobile para via API para o servidor que estará hospedado em uma máquina virtual em uma plataforma como a DigitalOcean por exemplo. Como serão utilizadas inúmeras imagens e imagens são arquivos pesados, será necessário salvar em um atributo o código Sha256 de um código Base64 da imagem.

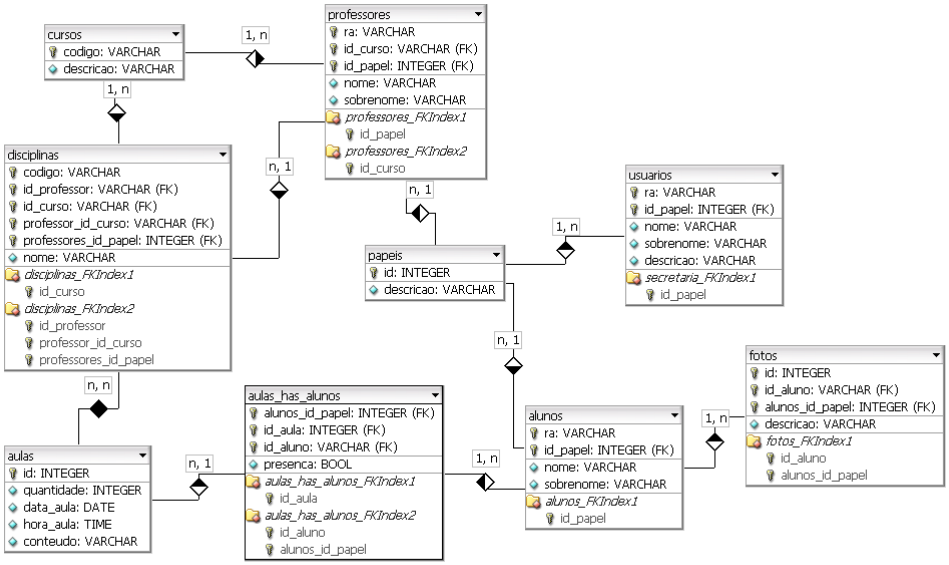

Figura 9 – Base de Dados

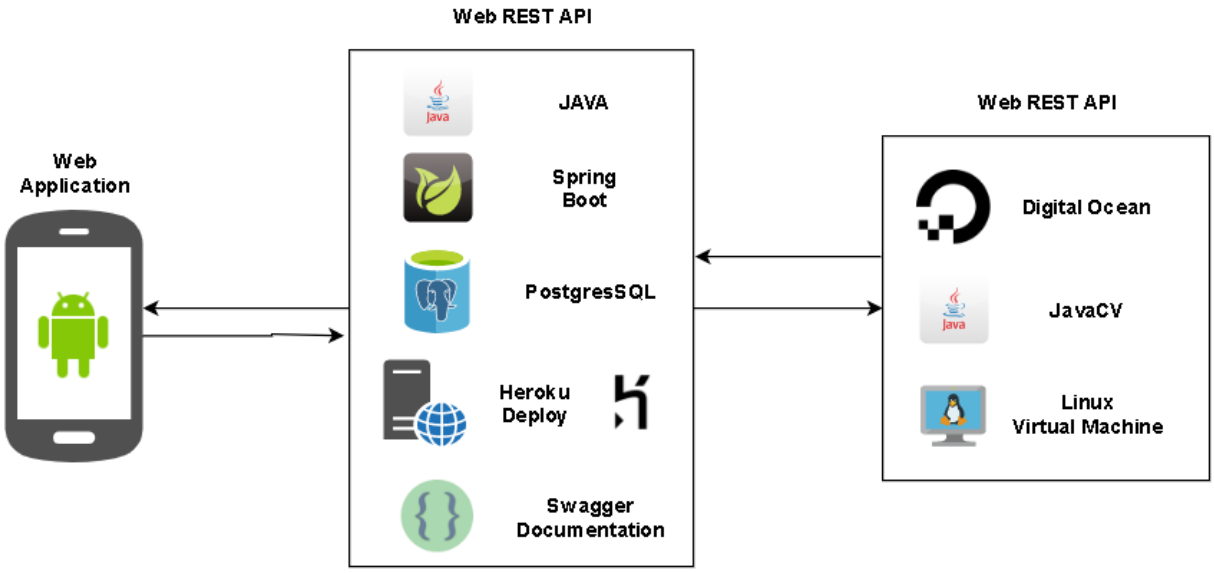

Figura 10 – Arquitetura da Estrutura da Rede

4.3 DESENVOLVIMENTO DO SISTEMA

Realizada a etapa de análise e projeto do sistema, deu-se início a fase de implementação da API. As tarefas serão dividas em papéis de Secretária, Professor e Aluno.

4.3.1 Reconhecimento Facial

Realizados os estudos em cima dos algoritmos de reconhecimento facial, podemos chegar a conclusão de que para que uma aplicação de reconhecimento facial funcione, são

necessárias no mínimo três classes implementadas. A primeira classe implementada é a de detenção facial. Essa classe é responsável por encontrar uma face em uma imagem, e guardar em variáveis aquela face. Essa face guardada deve ser enviada para o treinamento para ser rotulada, ou seja, para dizer que essa face é de uma determinada pessoa com uma determinada identificação. O papel de treinamento das imagens é realizado pela segunda classe, enquanto uma terceira classe é responsável por realizar o reconhecimento. Após feito o treinamento, as imagens estão rotuladas, então o algoritmo é capaz de identificar se uma face corresponde a uma pessoa ou não.

Também torna-se bem útil implementar uma classe de testes. Como neste momento as classes de captura da face, treinamento e reconhecimento estão implementadas, todos os três algoritmos propostos (EigenFaces, FisherFaces e LBPH) estão implementados nas três classes, é possível realizar testes com os três algoritmos, e ser montada uma tabela com os resultados obtidos, como é possível observar na figura 11, e assim como pode ser observado os parâmetros utilizados para cada algoritmo.

Eigenfaces		
Acertos	**Confiança**	**Parâmetros**
90	2467	Padrão
86,6	1536	30

Fisherfaces		
Acertos	**Confiança**	**Parâmetros**
86,6	915	Padrão
86,6	915	30

LBPH		
Acertos	**Confiança**	**Parâmetros**
83,3	15	Padrão
100	223	12,10,15,15

Figura 11 – Resultado dos Testes dos Algoritmos

4.3.2 API RESTful

A implementação da API foi realizada com Spring Boot, que auxilia na criação do projeto, auxiliando nas importações com anotações que facilitam a vida do programador, tendo uma documentação completa bem explicativa. Por ser a parte principal do projeto, deve ser pensada em todos os aspectos. Ao ser feita a prototipação de um aplicativo para utilização da API, é possível conhecer todas as ações que devem ser tomadas na aplicação, desta forma, podem ser conhecidas todas as funções REST que deveriam ser implementadas. E para fazer a prototipação foi necessário definir as histórias que foram descritas na tabela 1.

Esta API é responsável por fornecer todas as operações e dados ao cliente. Para certificar que as rotas estavam todas funcionando, foi necessário utilizar o Postman. Para realizar o teste local utilizando o Postman, é necessário criar o banco de dados na máquina

e rodar a aplicação em um servidor local. Neste trabalho foi utilizado o Tomcat para subir a aplicação em um servidor local e realizar os testes. Com o banco de dados criado, aplicação rodando em servidor local e as rotas criadas devidamente no aplicativo Postman, podem então ser feitas as requisições REST. Ao ser feita uma requisição para uma rota e obter o retorno desejado com status 200, têm-se então a confirmação de que a rota está funcionando conforme os planos. Pode ser verificado na figura 12.

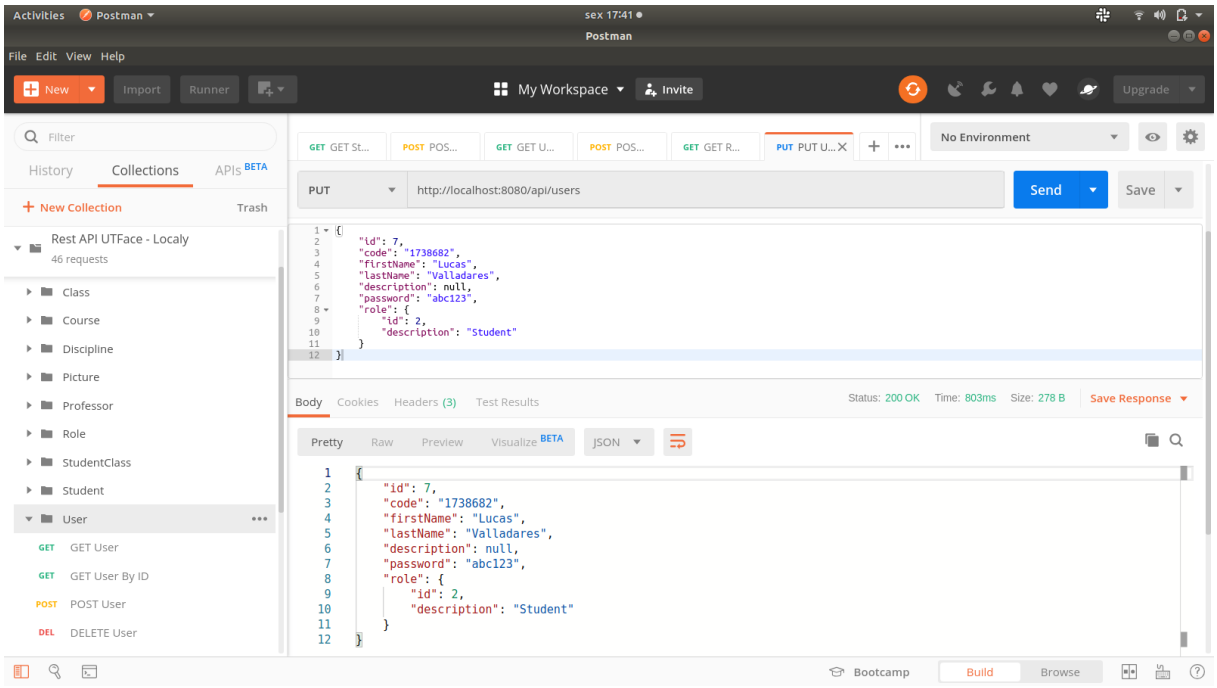

Figura 12 – Requisição PUT De Um Usuário Via Postman

Os dados que serão retornados da API para a o aplicativo, dependerá do papel que o usuário tiver. Caso o papel do usuário seja de aluno, ele poderá receber assim que concluído o controle de presença se recebeu presença ou falta, como descrito na história número 1 da tabela 1. A API consta com uma rota para presenças que contêm o ID da aula, o código do aluno e o status de presença. Existe também uma rota que irá retornar todas as presenças inserindo o código do aluno, como pode ser visto na figura 13. Como professor, o usuário poderá realizar o controle de presença, visualizar disciplinas que administra, previstas pelas histórias número 2 e 3 da tabela 1. Como papel de secretária, o usuário poderá cadastrar e gerenciar disciplinas, alunos e professores, descritas nas histórias do usuário de números 5 até 10 da tabela 1.

Inicialmente, um super usuário da aplicação cria um usuário com o papel de secretária para uma pessoa que irá administrar vários setores da aplicação. Este novo usuário poderá criar novos usuários com papel de professor e aluno (histórias 6 e 8 da tabela 1). No momento da criação do usuário, é definido também uma senha de acesso para ele, que poderá ser utilizada para efetuar login na aplicação. Essas verificações deverão ser realizados no lado do cliente. Para cadastrar um novo usuário o usuário deverá fazer uma requisição POST para a rota api/user. A

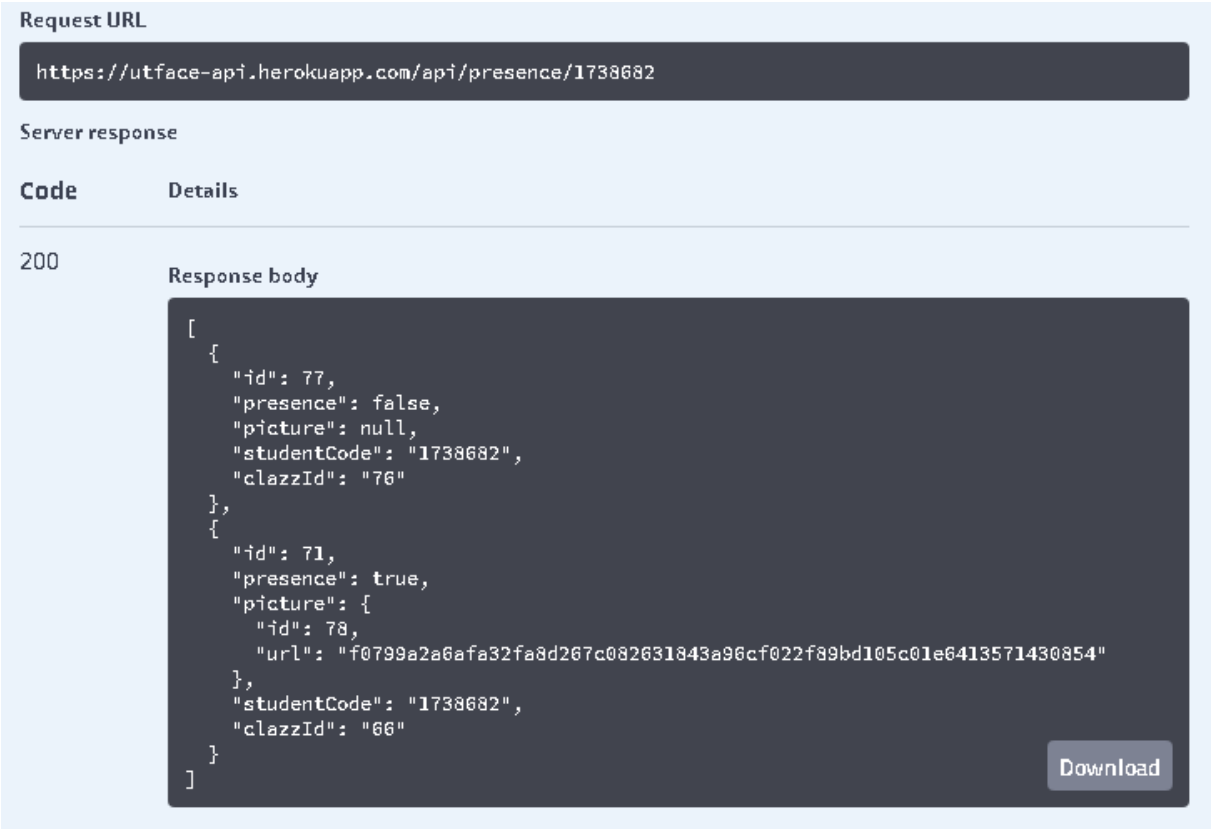

Figura 13 – Requisição GET De Presenças De Um Aluno

forma como o JSON deve ser submetido para a API fica bem claro na documentação Swagger da API, como pode ser verificado na figura 14.

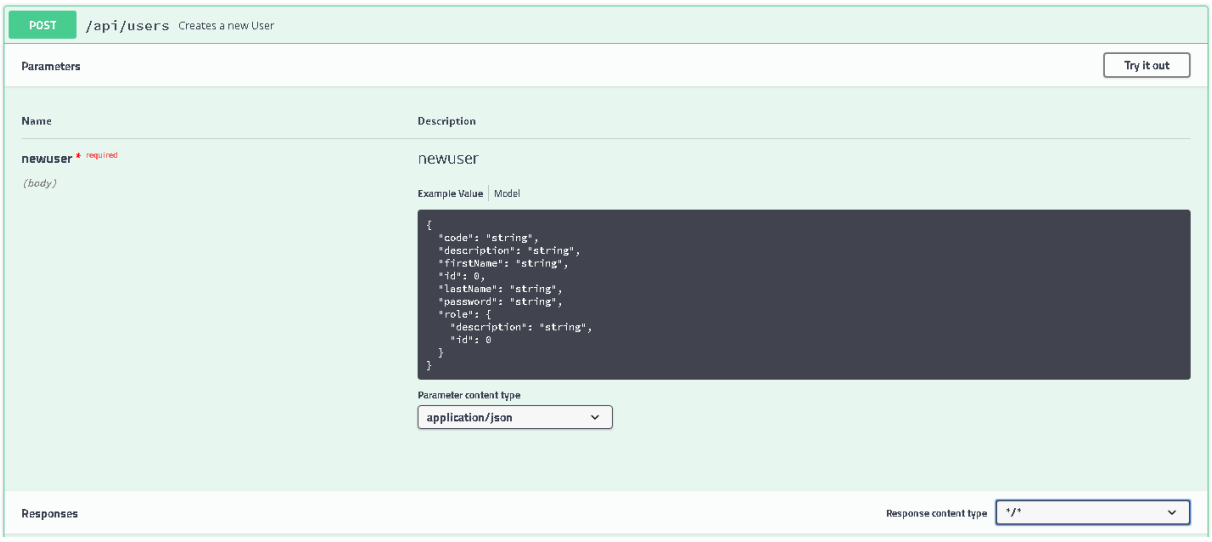

Figura 14 – Requisição POST De Um Novo Usuário API

Criada o usuário, é necessário adicionar o mesmo como estudante ou professor para

que o usuário possa receber seu papel e então possa ser relacionado às disciplinas. Ao adicionar um novo aluno, também deverão ser incluídas as suas imagens. A entidade Student conta com uma coleção de imagens. Essas imagens não serão submetidas como fotos realmente, desta forma, deverá ser extraído o código base64 da imagem, e então extraído o código Sha256 do código em base64. Esse código Sha256 é o que deve ser submetida para o atributo url da imagem para a API. O modelo de Picture pode ser visto na figura 15

Figura 15 – Requisições REST de Disciplina

Para que o controle de presença possa ser realizado é necessário fazer uma série de cadastros e popular a API. O processo se inicia quando o usuário com papel de secretária cadastra uma nova disciplina inserindo o código da disciplina, o ministrante da disciplina com papel de professor e os usuários com papel de aluno que compõem a turma. Para que esse processo possa ser concluído, foi desenvolvido o padrão REST para a classe disciplina, como pode ser visto na figura 16. Para ser criada uma nova disciplina é necessário fazer uma requisição POST para API com a rota de disciplinas. Após esse passo, o usuário cria as aulas com as datas específicas, a partir de uma requisição POST para a rota de aulas.

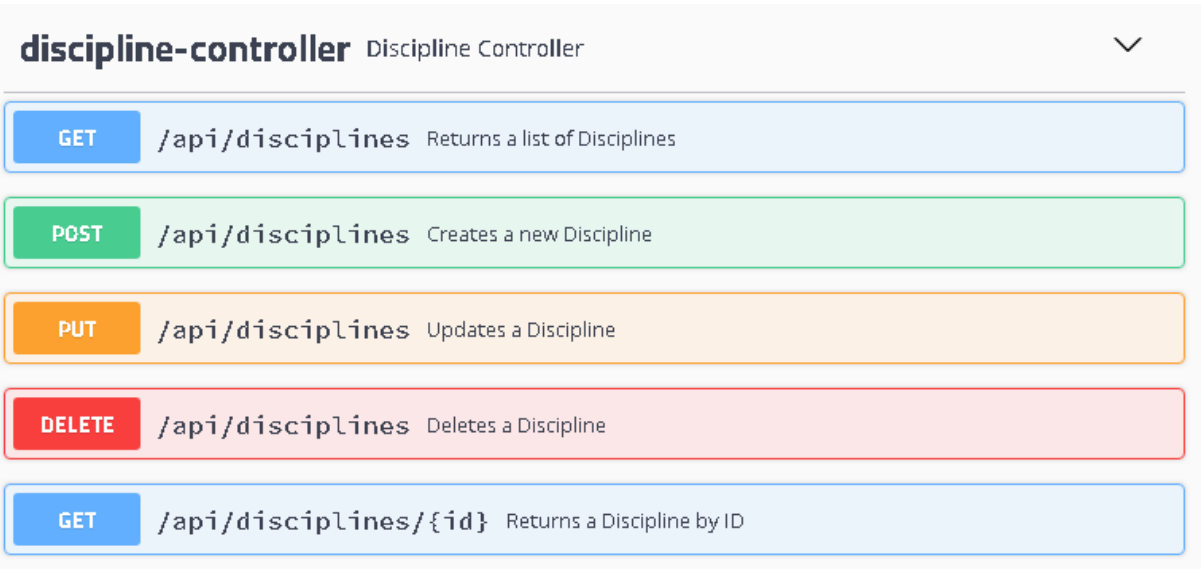

Figura 16 – Requisições REST de Disciplina

Com a disciplina cadastrada, quando chega a data e horário da aula, é liberada a aula para ser realizado o controle de presença dos alunos em sala de aula. O professor, a partir do seu aplicativo submete uma imagem dos alunos presentes, com uma requisição POST para imagens, modificando o atributo de presença de cada aluno na aula. Então realizada a chamada, todos os alunos daquela disciplina recebem uma notificação, a qual retorna o status de presença dos alunos (requisição PUT para aulas que pode ser verificado na figura 13).

A entidade Classe conta com um atributo de imagem. Quando essa entidade é criada, ou seja, quando é feita uma nova instância da mesma, o atributo de imagem deve ser nulo, e quando a imagem da sala de aula com os alunos for submetida pelo professor, esse atributo que antes era nulo, agora é composto pelo código Sha256 do código Base64 da imagem submetida. Desta forma, do lado do servidor será feito o reconhecimento facial, e então retornar o status de presença dos alunos cadastrados naquela aula.

Como uma aula é composta por vários alunos e um aluno também tem várias aulas, então é criado um relacionamento ManyToMany (MuitosParaMuitos) e para que o relacionamento possa ser feito, foi desenvolvida uma classe intermediária chamada StudentClass, que tem, além do relacionamento com Class e Student, uma atributo de presença para o aluno daquela classe. A modelagem da classe pode ser vista na figura 17.

Figura 17 – Modelo StudentClass

4.3.3 Deploy Heroku

Heroku Platform é uma plataforma baseada em serviços em um sistema de controle de containers, com serviços de dados integrados e um poderoso ecossistema, para desenvolver e rodar aplicações modernas (PLATFORM, 2019).

Após o desenvolvimento da API, a mesma foi alocada no Heroku para que possa ter acesso online. A partir deste momento a API não estará rodando em um servidor online, a aplicação será exibida como na figura 18, portando será utilizado um banco de dados integrado com o Heroku Platform. Ao ser feito o deploy, a plataforma cria o banco de dados automaticamente, no entanto, o banco de dados, naturalmente, está vazio, então é necessário incluir os dados para que as requisições de retorno da API possam retornar algum dado. Naturalmente as requisições de postagem, ou de criação de objetos a partir da API e de suas

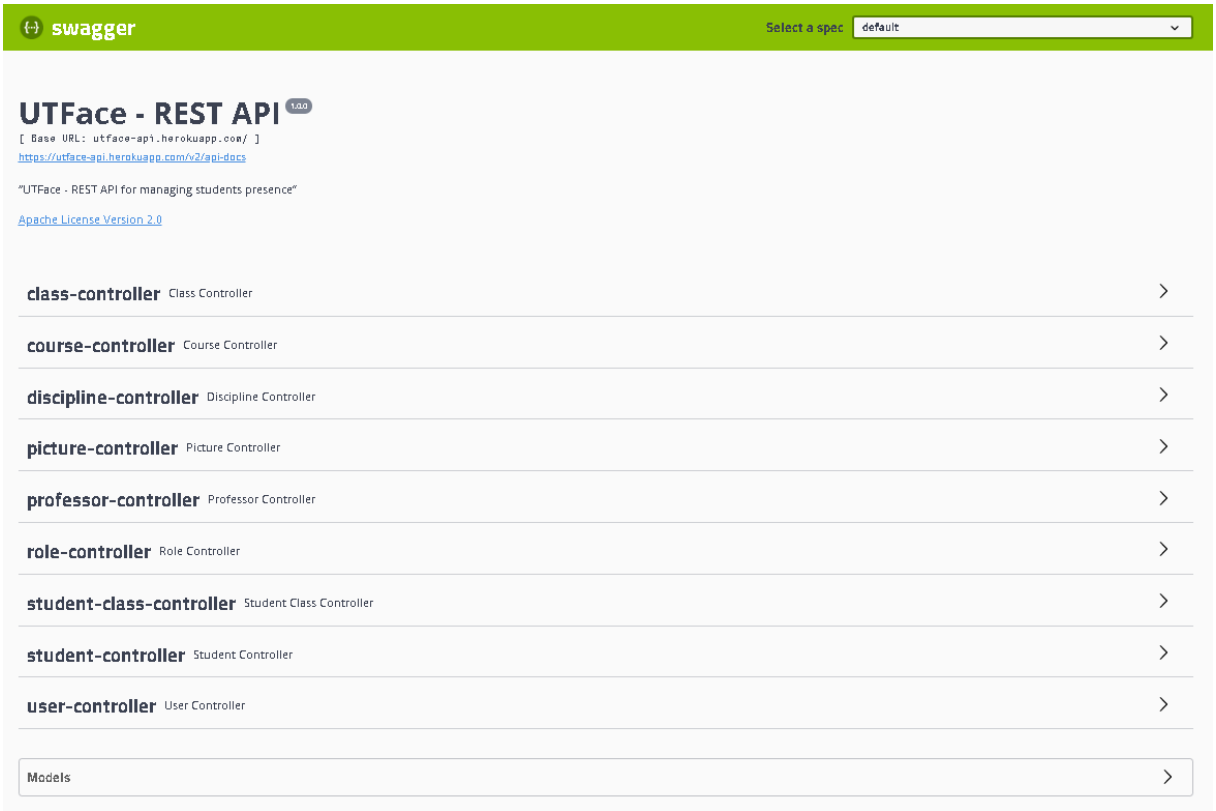

Figura 18 – Modelo StudentClass

rotas estarão funcionando. Desta forma, podemos inserir os dados no banco de dados da plataforma Heroku com comandos sql ou até mesmo diretamente pela API.

5 CONSIDERAÇÕES FINAIS

Com a realização de reuniões e discussões a fim de gerar um brainstorm foram coletados os requisitos para o projeto, com os quais foi possível definir uma visão do produto, que foi transformada em prototipação de telas, histórias de usuários, base de dados e prototipação de API (rotas e métodos) dando aos participantes do projeto a mesma ideia sobre o que está sendo proposto e projetado ao desenvolvimento.

O objetivo deste trabalho foi desenvolver uma API que possa ser utilizada por um aplicativo para que os professores possam realizar a chamada de presença em sala de aula de forma automática, apenas submetendo uma ou várias fotos da turma para que cada aluno presente seja reconhecido e contado como presente na aula. Também poderá ser utilizada em eventos onde são passadas listas de presença entre muitos alunos e onde não se têm o controle e garantia da assinatura que consta na lista. Para que a API tivesse um bom desempenho no que foi proposto, foi necessário o estudo de algoritmos de reconhecimento facial e também a implementação dos mesmos, desta forma, evitando o desenvolvimento de uma ferramenta que não teria utilidade no futuro.

Os professores experienciam no seu cotidiano a custosa tarefa ao realizarem a chamada presencial em sala de aula e em eventos, e esta exige com que seja utilizado um sistema complexo que tomam tempo de aula ou deve ser passada uma lista de presença onde o aluno deve assinar para comprovar sua presença, no entanto, em ambas as formas podem ocorrer equívocos ou até mesmo fraudes. Podemos então imaginar o quanto a automatização da tarefa seria sonhada já que vivemos em uma época com predominância da tecnologia onde a realização de chamada presencial com um papel e caneta pode ser considerada até mesmo um método arcaico para as próximas gerações. Com isto trazemos a inovação tecnológica para tarefas simples do nosso cotidiano, para uma universidade que leva em seu nome a tecnologia.

É notável o alto nível de aceitação com professores ao discutirmos a respeito da implementação de uma ferramenta com essas características e funcionalidades, trazendo maior motivação para realização do projeto proposto, buscando realizar um projeto onde trará comodidade não apenas à uma pessoa, mas sim para uma comunidade inteira, que é a comunidade acadêmica da UTFPR.

No entanto, esse futuro tão esperado ainda não chegou, e com o desenvolvimento dessa API, inicia-se o sonho de automatizar o controle de presença em sala de aula nas universidades. E com isto, deixa-se o convite para futuros trabalhos, para o desenvolvimento do aplicativo, utilizando a API e o reconhecimento facial, assim concluindo a sonhada aplicação.

5.1 Trabalhos Futuros

A API está desenvolvida e foi testada manualmente na aplicação já online e também via Postman, onde existem todas as rotas já formadas e com o corpo JSON dos objetos prontos para serem editados e testar a aplicação. Para que o próximo trabalho, será fornecida coleção de testes Postman para que o próximo desenvolvedor possa realizar os próprios testes. O desenvolvedor também poderá solicitar o projeto de reconhecimento facial com as classes necessárias e devidos testes já implementados para fim de estudo deste mesmo projeto.

Utilizando as ferramentas que foram desenvolvidas neste projeto, um grande passo é tomado para o desenvolvimento de uma sistema completo, onde integrando todas essas ferramentas com um aplicativo, se torna o tão sonhado sistema de controle de presenças com reconhecimento facial.

A API encontra-se sem segurança, então para que a API funcione de acordo e sem fraudes, como é a proposta do projeto, deverá ser incluído na API a autenticação e validação da mesma, tornando assim uma aplicação segura e os dados seguros contra ataques e/ou fraudes. Para a autenticação, poderá ser utilizada o JWT (JSON Web Token) fornecido pelo Spring Boot que é o framework utilizado para desenvolver a API. JWT utiliza um token que pode ser enviado via escopo de requisição para que possa ser efetuada a autenticação do usuário.

5.2 Conclusão

Conclui-se nesta sessão que o projeto da API proposto foi desenvolvido e todas as ferramentas necessárias para continuação do projeto estão totalmente disponíveis e acessíveis para que possam ser utilizadas. Aceita-se também que ao fazer utilização da API na integração de uma aplicação, possíveis e cabíveis modificações poderão ser realizadas a fim de alcançar melhor aproveitamento das ferramentas e aplicações disponibilizadas.

Referências

ALVES, P. **Os navegadores de Internet mais usados no Brasil e no mundo**. TechTudo, 2016. Disponível em: <https://www.techtudo.com.br/listas/noticia/2016/01/os-navegadores-de-internet-mais-usados-no-brasil-e-no-mundo.html>. Citado na página 8.

BELHUMEUR, P. N.; HESPANHA, J. P.; KRIEGMAN, D. J. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 19, 1997. Citado 2 vezes nas páginas 6 e 7.

BISSI, T. D. Reconhecimento facial com os algoritmos eigenfaces e fisherfaces: proposta metodológica. p. 12, 2018. Citado na página 5.

CASTRO, E.; HYSLOP, B. **HTML5 and CSS3, Seventh Edition: Visual QuickStart Guide**. 7. ed. [S.l.]: Peachpit Press, 2012. Citado na página 8.

CREDDEFENSE. 2018. Disponível em: <https://creddefense.com.br/>. Citado na página 4.

CULJAK, I. et al. A brief introduction to opencv. IEEE, 2012. Citado na página 9.

DAVISON, A. **Vision-based User Interface Programming in Java**. [S.l.]: Amazon Digital Services, 2013. Citado na página 9.

EFRAIM. Introdução ao spring framework. DevMedia, 2012. Disponível em: <https://www.devmedia.com.br/introducao-ao-spring-framework/26212>. Acesso em: 28 de Junho de 2019. Citado na página 10.

ELOLA, J. **O reconhecimento facial abre caminho para o pesadelo de George Orwell**. 2018. Disponível em: <https://brasil.elpais.com/brasil/2018/01/05/tecnologia/1515156123_044505.html>. Acesso em: 11 de Setembro de 2018. Citado 2 vezes nas páginas 4 e 9.

FACE++. 2018. Disponível em: <https://www.faceplusplus.com/face-detection/>. Citado na página 4.

FINDFACE. 2018. Disponível em: <https://findface.pro/en/>. Citado na página 3.

HAVERBEK, M. **Eloquent JavaScript**: A mordern introduction to programming. 3. ed. [S.l.: s.n.], 2018. Citado na página 8.

JOHNSON, R. **Expert One-on-one J2ee Design and Development**. [S.l.]: Wiley Publishing, 2003. Citado na página 10.

LEANDRO, E. **Aplicativo FindFace permite que estranhos identifiquem você**. 2019. Disponível em: <https://www.areah.com.br/vibe/aplicativo/materia/171535/1/pagina_1/aplicativo-findface-permite-que-estranhos-identifiquem-voce.aspx>. Acesso em: 28 de Junho de 2019. Citado na página 3.

OJALA, T.; PIETIKäINEN, M.; HARWOOD, D. **A comparative study of texture measures with classification based on featured distributions**. [S.l.]: Elsevier, 1996. v. 29. Citado na página 7.

PARRA, D. E. Parra. comparação entre algoritmos de reconhecimento de face no contexto de acessibilidade. p. 56–58, 2014. Citado na página 8.

PIRES, J. **O que é API? REST e RESTful? Conheça as definições e diferenças!** 2017. Disponível em: <https://becode.com.br/o-que-e-api-rest-e-restful/>. Acesso em: 27 de Junho de 2019. Citado na página 9.

PLATFORM, H. **The Heroku Platform**. 2019. Disponível em: <https://www.heroku.com/platform>. Acesso em: 09 de Novembro de 2019. Citado na página 22.

RICHARDSON, L. **RESTful Web APIs**. [S.l.]: Reilly Media, 2013. Citado na página 9.

SMARTBEAR. **About Swagger**. 2019. Disponível em: <https://www.heroku.com/platform>. Acesso em: 01 de Setembro de 2019. Citado na página 10.

SULIVAN, F. . **Sistema de Reconhecimento Facial para Prevenção de Fraudes de Crédito**. 2018. Disponível em: <https://br.nec.com/pt_BR/safety/pdf/finance_infographics.pdf>. Acesso em: 11 de Setembro de 2018. Citado na página 4.

SWAGGER. 2019. Disponível em: <https://swagger.io/>. Acesso em: 28 de Junho de 2019. Citado 2 vezes nas páginas 15 e 16.

TÓV-TEC. **A TÓV-TEC É UMA EMPRESA DE TECNOLOGIA QUE REALIZA O DESENVOLVIMENTO E PERSONALIZAÇÃO DE SOFTWARES**. 2017. Disponível em: <http://feirasenegocios.com.br/noticias/2017/06/tov-tec/>. Acesso em: 11 de Setembro de 2018. Citado na página 5.

TóV-TEC. 2018. Disponível em: <http://tovtec.com.br/>. Citado na página 5.

WANG, L.; HE, D.-C. **Texture classification using texture spectrum**. [S.l.]: Elsevier, 1990. v. 23. Citado na página 7.

WANG, X.; YAN, S.; HAN, T. X. An hog-lbp human detector with partial occlusion handling. **Computer Vision - IEEE 12th International Conference**, p. 32–39, 2009. Citado na página 7.

Z., S.; LI, A. K. **Handbook of Face Recognition**. [S.l.]: London: Springer, 2011. Citado 2 vezes nas páginas 6 e 7.