UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
ENGENHARIA ELÉTRICA

EDUARDO ARCE DE SALES

# DESENVOLVIMENTO DE UM SOFTWARE PARA AUTOMATIZAR O SISTEMA DE FREQUÊNCIA DE ALUNOS POR MEIO DE RECONHECIMENTO FACIAL

MANAUS
2022

**EDUARDO ARCE DE SALES**

# DESENVOLVIMENTO DE UM SOFTWARE PARA AUTOMATIZAR O SISTEMA DE FREQUÊNCIA DE ALUNOS POR MEIO DE RECONHECIMENTO FACIAL

Projeto de pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Fábio de Sousa Cardoso

Manaus

2022

***Universidade do Estado do Amazonas – UEA***
***Escola Superior de Tecnologia – EST***



*Banca Avaliadora composta por:* *Data da defesa: 21/10/2022*
***Prof. Fábio de Sousa Cardoso, Dr.*** *(Orientador)*
***Prof. Jozias Parente de Oliveira, Dr.***
***Prof. Raimundo Cláudio Souza Gomes, Dr.***

# CIP – Catalogação na Publicação

Sales, Eduardo Arce de

Desenvolvimento de um software para automatizar o sistema de frequência de alunos por meio de reconhecimento facial / Eduardo Arce de Sales; [orientado por] Prof. Dr. Fábio de Sousa Cardoso – Manaus: 2022.
59 p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica). Universidade do Estado do Amazonas, 2022.

1. Automatizar Frequência de Alunos. 2. Desenvolvimento de Software Backend. 3. Reconhecimento Facial.

I. Cardoso, Fábio de Sousa.

**EDUARDO ARCE DE SALES**

# DESENVOLVIMENTO DE UM SOFTWARE PARA AUTOMATIZAR O SISTEMA DE FREQUÊNCIA DE ALUNOS POR MEIO DE RECONHECIMENTO FACIAL

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro Eletricista.

Nota obtida: 9,2 ( nove vírgula dois )

Aprovado em 21/10/2022

Área de concentração: Software

BANCA EXAMINADORA

Orientador: Fábio de Sousa Cardoso, Dr.

Avaliador: Jozias Parente de Oliveira, Dr.

Avaliador: Raimundo Cláudio Souza Gomes, Dr.

Manaus

2022

**Dedicatória**

Dedico este trabalho aos meus pais, Francilene da Silva Arce e José Eduardo Silva de Sales, que sempre me ajudaram e apoiaram em todos os momentos da minha vida.

## AGRADECIMENTO

A Deus, que tornou possível, mesmo com muitos sacrifícios me deu determinação para finalizar este trabalho.

Agradeço aos membros da minha família, que sempre proporcionaram condições adequadas para meu crescimento na vida pessoal e profissional, especialmente minha mãe Francilene, meu pai José Eduardo, meus avós paternos Raimunda e João e meus avós maternos Francisca (in memorian) e Clinger.

Agradeço ao meu orientador, Fábio de Sousa Cardoso, por fornecer apoio e tirar dúvidas de escrita e implementação durante o desenvolvimento deste trabalho.

Aos meus amigos, Tiago Ramos e Isaque Vilson, que deste o início do curso de Engenharia Elétrica, em 2017, fizemos amizade e sempre nos apoiamos durante as disciplinas, seja para estudos ou trabalhos em grupo, sempre unidos contra os desafios do universo acadêmico.

Aos amigos que fiz durante minha jornada profissional enquanto estagiário no grupo Philco/Britânia, Fabiano Barroso, Cristiano Vezu e André Viana, bem como todos os integrantes da BRIC INJ, aos amigos do Sidia, Alexandre Magalhães e Adriano Eustáquio, assim como os outros integrantes do Field Test. E aos amigos do Itriad, empresa que trabalho atualmente, em especial o Tiago Custódio, por todo suporte e ajuda que recebi e recebo até hoje.

Agradeço aos grandes amigos que Deus me deu, Jonathan Maia, Vinícius Gabriel e Eldo Marcolino, que estão sempre presentes na minha vida.

## RESUMO

Neste projeto, foi desenvolvido um sistema para automatizar a frequência de alunos por meio do reconhecimento facial, que tem como objetivo a redução do tempo que o professor gasta em sala de aula para fazer a chamada dos alunos. Este trabalho apresenta primeiramente a implementação da modelagem do banco de dados, que serve como base para o desenvolvimento da API do sistema, assim, são iniciados a implementação das regras de negócio do projeto, tratamento de exceções e as demais funcionalidades planejadas, sendo o backend da aplicação desenvolvido na linguagem de programação Python. Em seguida, o projeto do reconhecimento facial é iniciado, e de inicio, houve a organização dos diretórios, na sequência, realizar o treino e identificação das imagens dos usuários. O treino deste projeto foi realizado com a imagem de 50 usuários e houve uma taxa de acertividade de 86%.



**Palavras chave**: Automatizar Frequência de Alunos, Desenvolvimento de Software Backend, Reconhecimento Facial.

## ABSTRACT

In this project, a system was developed to automate student attendance through facial recognition, which aims to reduce the time that the teacher spends in the classroom to call students. This work first presents the implementation of the database modeling, which serves as a basis for the development of the system's API, thus, the implementation of the project's business rules, exception handling and other planned functionalities are initiated, being the backend of the application developed in the Python programming language. Then, the facial recognition project is started, and in the beginning, there was the organization of directories, then carrying out the training and identification of users' images. The training of this project was carried out with the image of 50 users and there was an accuracy rate of 86%.



**Keywords**: Automate Student Attendance, Backend Software Development, Face Recognition.

# Lista de Figuras

# Lista de Tabelas

# Lista de Siglas e Abreveaturas

| | |
|---|---|
| API | Application Programming Interface |
| CNN | Convolutional Neural Network |
| CRUD | Create-Read-Update-Delete |
| DP | Deep Learning |
| EST | Escola Superior de Tecnologia |
| HTTP | HyperText Transfer Protocol |
| IDE | Integrated Development Environment |
| ML | Machine Learning |
| MVC | Model-View-Controller |
| ORM | Object Relational Mapper |
| RBF | Radial Basis Function |
| RGB | Red-Green-Blue |
| SVM | Support Vector Machine |
| SRC | Source |
| UEA | Universidade do Estado do Amazonas |
| URL | Uniform Resource Locator |
| XML | Extensible Markup Language |

**SUMÁRIO**

# INTRODUÇÃO

Desde o princípio das escolas e universidades, o sistema de frequência de alunos é realizado de forma manual, onde o(a) professor(a) faz a verificação dos alunos presentes em sala de aula, chamando o nome de cada aluno matriculado na disciplina de forma individual (TELTEX, 2020). Mas com o avanço da tecnologia, um sistema para automatizar a frequência de alunos é necessária por 2 motivos: (i) economizar o tempo dos professores e (ii) garantir a não existência de fraudes e erros manuais. O sistema de detecção de faces tem como algumas aplicações como o controle de acessos (DIGIFORT, 2019), registro de ponto de funcionários (GRYFO, 2021) e desbloqueio de aplicativos e smartphones (SUMUS, 2019). Esses exemplos representam algumas das diversas aplicações da detecção facial, sendo implementado usando técnicas de Inteligência Artificial e métodos de Visão Computacional (WANGENHEIM, 2017).

O presente projeto teve como objetivo principal o desenvolvimento de um modelo capaz de identificar os alunos da universidade, por meio de detecção facial e também o desenvolvimento de um software capaz de receber os dados dos alunos detectados para fazer a atualização da lista de frequência da turma. Para isso foram usados conceitos relacionados a Visão Computacional e Desenvolvimento de Software para Backend.

Este trabalho está dividido em 5 capítulos, são eles: (I) Referencial Teórico, onde está demonstrada toda a base teórica para o desenvolvimento desta pesquisa, (II) Metodologia, onde estão expostas as regras de negócio e etapas de desenvolvimento da aplicação, (III) Implementação, onde estão exibidas as fases da implementação do backend e do reconhecimento facial, (IV) Resultados, onde foram registrados os resultados dos testes realizados nos endpoints do backend e na verificação do algoritmo de reconhecimento facial, e por último (V) Conclusão, onde mostram os principais resultados do projeto e ideias de melhorias e projeções para trabalho futuros.

# 1 REFERENCIAL TEÓRICO

## 1.1 SISTEMA AUTOMÁTICO DE FREQUÊNCIA DE ALUNOS VIA RECONHECIMENTO FACIAL

O sistema para automatizar a frequência de alunos através do reconhecimento facial é um projeto que tem como objetivo facilitar o trabalho do professor, economizando o tempo em sala de aula que é usado para fazer a chamada dos alunos. Por meio desta aplicação, ao iniciar a aula, o professor vai fazer acessar o projeto através de seu login e senha e a chamada será realizada quando o aluno exibir sua face para uma câmera do dispositivo na sala de aula, assim, os alunos serão identificados e suas presenças na disciplina confirmadas.

Dessa forma, os professores podem usar o tempo economizado para trabalhar na resolução de um exercício com os alunos em sala de aula ou tirar dúvidas, e além disso, os dados da presença dos alunos estará armazenado num banco de dados, possibilitando a verificação e análise da frequência dos alunos.

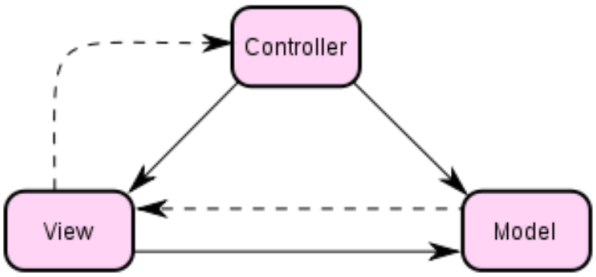
## 1.2 BACKEND

O Backend está relacionado a servidores, bancos de dados, segurança, estrutura, gerenciamento de conteúdo e atualizações. Este tipo de aplicação é usado para processar dados e também é onde os dados são inseridos, arquivados e lidos por trás do aplicativo. Em suma, esses recursos garantem a execução dos processos mais simples, como buscas em sites de compras, até mesmo as operações mais complexas em ambiente eletrônico (EWALLY, 2021).

### 1.2.1 Arquitetura MVC

O padrão MVC (Model-View-Controller) consiste em dividir o código do software em segmentos funcionais para que a aplicação esteja pronta para responder rapidamente a alterações, testes unitários e extensões sem afetar o código feito, mantendo os limites entre as três camadas. As camadas da arquitetura deste modelo consistem em três tipos de objetos, o modelo é o motivo da existência da aplicação e armazena toda a lógica de negócios e dados do sistema, a visão é a representação da saída gráfica gerada pelo modelo para o usuário e o controlador, que mediação entre camadas, interpretando dados de entrada e gerenciando modelos e visualizações (OLIVEIRA, 2013).

Figura 1 – Diagrama da Arquitetura MVC




Fonte: (LUCIANO; ALVES, 2011)

### 1.2.2 Sanic Framework

A visão original para a reutilização de software era baseada em componentes. Os frameworks têm interfaces mais complexas, mas são mais fáceis de personalizar do que os componentes. Os autores argumentam que frameworks e componentes são tecnologias diferentes, mas eles se complementam porque frameworks facilitam a construção de novos componentes e fornecem uma interface padrão para trocar dados, lidar com erros e invocar operações (JOHNSON, 1997).

Sanic é um framework web para python construído em uvloop, projetado para obter respostas HTTP rápidas com processamento de solicitação assíncrona, além disso, ainda está em desenvolvimento muito ativo e está em sua infância como um framework web (MAKAI, 2018b), tendo a possibilidade de se desenvolver e melhorar ainda mais suas funcionalidades.

### 1.2.3 Métodos HTTP

Os métodos HTTP compõem a maioria de nossas restrições de ”interface unificada”e nos fornecem ações correspondentes para recursos baseados em substantivos. Os métodos HTTP principais ou mais usados são POST, GET, PUT e DELETE. Corresponde às operações de criação, leitura, atualização e exclusão, o famoso CRUD (Create, Read, Update e Delete). Existem vários outros métodos, mas eles são usados com menos frequência (RESTAPITUTORIAL, 2017).

## 1.3 BANCO DE DADOS

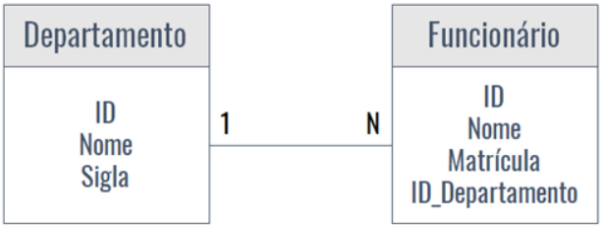
Um banco de dados é onde os dados são representados em forma de tabela. A descrição de uma determinada entidade é fornecida por uma coleção de valores de seus atributos, armazenados como linhas ou registros em uma tabela, denominadas tuplas, onde itens semelhantes de diferentes registros podem aparecer nas colunas da tabela (GREGERSEN, 2020).

### 1.3.1 Relacionamento de Tabelas

- 1:N

Também conhecido como relacionamento um-para-muitos, é o tipo mais comum. Nessa relação de tabelas, uma linha na tabela A pode ter muitas linhas correspondentes na tabela B. No entanto, uma linha na tabela B só pode corresponder a uma linha na tabela A. Se houver apenas uma, as colunas relacionadas que criarão um relacionamento um-para-muitos serão chaves primárias ou terão restrições exclusivas (MARRY, 2022).

Figura 2 – Relação de Tabelas 1:N



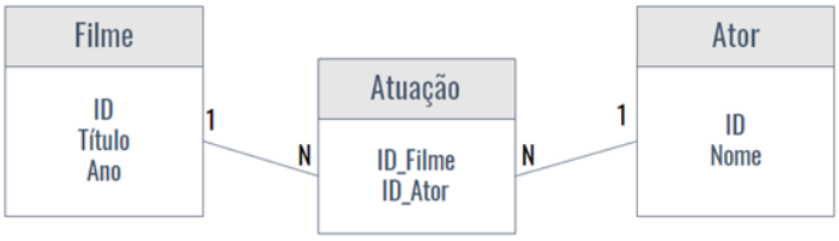

Fonte: (GOMES, 2019)

Na Figura 2, verifica-se o exemplo de relação 1:N, por meio das tabelas de Departamento e Funcionário, onde um departamento pode conter vários funcionários, mas um funcionário pode fazer parte apenas de um departamento.

- N:N

Em um relacionamento muitos-para-muitos, uma linha na tabela pode ter muitas linhas correspondentes na tabela B e vice-versa. Você cria esse relacionamento definindo uma terceira tabela chamada de tabela de junção. A chave primária da tabela de junção consiste nas chaves estrangeiras da tabela A e da tabela B (MARRY, 2022).

Figura 3 – Relação de Tabelas N:N




Fonte: (GOMES, 2019)

Na Figura 3, é exibido o exemplo da relação de tabelas N:N, com a análise das tabelas Filme e Ator, onde um filme existem vários atores e um ator também pode participar de diversos filmes, então existe a necessidade de criar uma nova tabela, para que seja possível vincular os filmes com os atores.

### 1.3.2 Peewee

Peewee é um ORM (Object Relational Mapper) para conectar dados armazenados em tabelas de banco de dados relacionais com objetos Python (MAKAI, 2018a).

A função deste ORM é converter classes Python em tabelas no banco de dados, além de permitir que você construa consultas diretamente usando objetos Python ao invés de SQL. O Peewee é voltado para projetos de pequeno e médio porte e se destaca pela simplicidade em comparação com outros ORMs mais populares. Em relação aos recursos que oferece, podemos citar que possui suporte nativo para SQLite, PostgreSQL e MySQL (STUTTGART, 2017).

### 1.3.3 PostegreSQL

PostgreSQL é um sistema de gerenciamento de banco de dados relacional que foi desenvolvido ativamente como um projeto de código aberto. O PostgreSQL tem uma reputação de confiabilidade, robustez, recursos e desempenho, graças em grande parte à dedicação da comunidade de código aberto por trás do software. O PostgreSQL suporta cargas de trabalho consideráveis e pode processar grandes quantidades de informações. Além disso, aproveita e estende a linguagem SQL (Structured Query Language), as instruções da linguagem para comunicação com bancos de dados relacionais, capazes de serem executadas em todos os principais sistemas operacionais (CAVALCANTE, 2021).

## 1.4 SUPPORT VECTOR MACHINE

O Support Vector Machine (SVM, em português: Máquina de Vetores de Suporte) é um conjunto de métodos relacionados ao aprendizado supervisionado para classificação e regressão. Eles pertencem à família de classificação linear generalizada. Uma de suas propriedades é que simultaneamente minimiza a classificação empírica incorreta e maximiza a a geometria da margem (HSU; CHANG; LIN, 2016).

O objetivo de um SVM é gerar um modelo (com base nos dados de treinamento) que preveja o valor alvo dos dados de teste apenas dadas as propriedades dos dados de teste (SRIVASTAVA; BHAMBHU, 2010).

Seguem as equações para otimização do SVM:

$$(x_i, y_i), i = 1, ..., l \tag{1}$$

Enquanto $x_i \in R^n$ e $y_i \in 1, -1^l$.

$$min \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \tag{2}$$

Sujeito à $y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i$ e $\xi_i \geq 0$.

Seguem as seguintes funções de Kernel: Linear, Polinomial e Radial Basis Function (RBF, em português: Função de Base Radial), Equações 3, 4 e 5, respectivamente, para mapear os dados num espaço dimensional superior.

$$k(x_i, x_j) = X_i^T X_j. \tag{3}$$

$$k(x_i, x_j) = (\gamma X_i^T X_j + r)^d, \gamma < 0 \tag{4}$$

$$k(x_i, x_j) = e(-\gamma||X_i - X_j||^2), \gamma < 0 \tag{5}$$

## 1.5 DEEP LEARNING

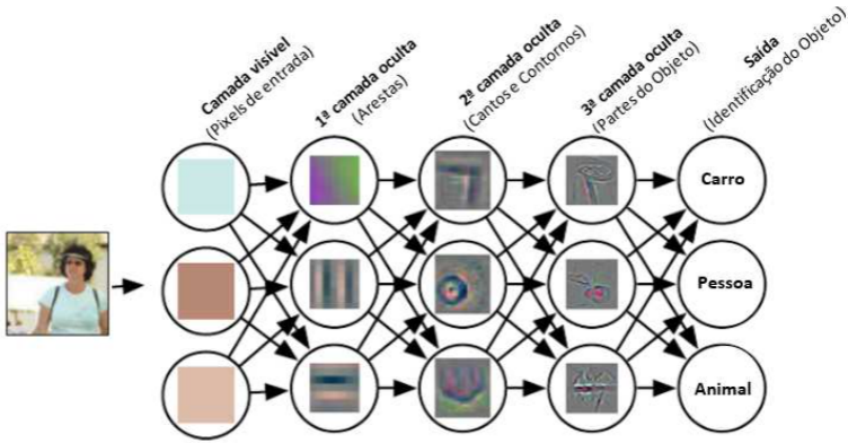
O Deep Learning (DL, em português: Aprendizado Profundo) é um conceito de aprendizado de máquina baseado em redes neurais artificiais. Para muitas aplicações, os modelos de aprendizado profundo superam os modelos de Machine Learning (ML, em português: Aprendizado de Máquina) e as abordagens tradicionais de análise de dados (HEINRICH, 2021). Esses modelos podem ser desenvolvidos para as aplicações mais diversas, como o reconhecimento de voz, objetos, sinais, visão computacional, dentre outros.

### 1.5.1 Reconhecimento de Imagens com Deep Learning

Existe uma técnica específica de DL utilizada para reconhecimento de imagem, denominada Redes Neurais Convolucionais. Uma rede neural convolucional é uma estrutura especializada no processamento de sinais brutos, tais como imagens e áudios, e os recursos são extraídos aplicando uma várias convoluções de filtros em sequência, com pesos convolucionais previamente aprendidos durante o treino (VENUGOPAL, 2016).

Na Figura 4 é exposto um simples exemplo de DP, em que são verificadas características de cantos e contornos processados em uma estrutura de camadas hierarquicamente distribuída, onde os elementos de cada camada são conectados, afim de classificar a imagem verificação, neste caso, uma pessoa.

Figura 4 – Reconhecimento de Imagem Usando Deep Learning

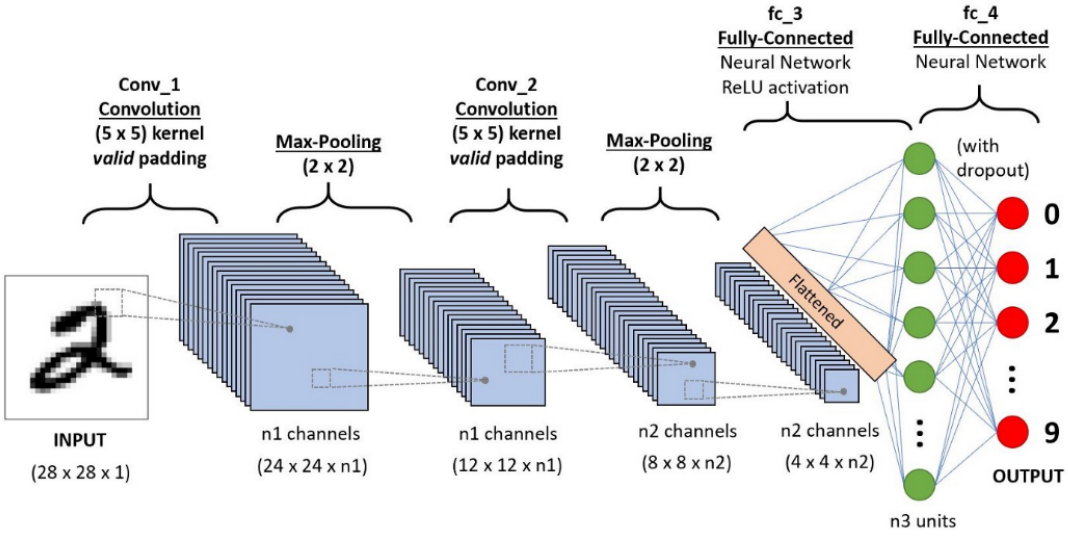



Fonte: (RAZORTHINK, 2020)

## 1.6 REDES NEURAIS CONVOLUCIONAIS

As Redes Neurais Convolucionais (CNN, do inglês: Convolutional Neural Network) usam um processo de convolução sequencial no conjunto de dados, usando filtros e métodos previamente aprendidos para obter as características desejadas. Na Figura 5 são identificadas as etapas de uma CNN, que passam por Convolução, Max-Pooling e Camada Conectada, até identificar a imagem de entrada através do SoftMax.

Figura 5 – Fluxo de uma Rede Neural Convolucional




Fonte: (EBERMAM; KROHLING, 2018)

### 1.6.1 Convolução

Matematicamente, a convolução é definida como a integral de uma função em todo o espaço x vezes outra função em $\tau$x. A integração é feita sobre a variável x (que pode ser uma variável 1D, 2D ou 3D), geralmente do infinito negativo ao infinito em todas as

dimensões (READ, 2009). Portanto, a convolução é uma função de uma nova variável $\tau$ conforme mostrado na Equação 6.

$$h(t) = (f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau \tag{6}$$

O cálculo da convolução pela definição, como mostrado na Equação III é bastante complexo e demanda um certo tempo para ser solucionado, mas este cálculo pode ser simplificado através do uso da Transformada de Laplace, conforme a Equação 7.

$$\mathcal{L}\{(f * g)(t)\} = H(S) = F(S)G(S) \tag{7}$$

Feito isso, para obter a resposta do sinal $h(t)$, é necessário aplicar a Transformada de Laplace inversa, conforme a Equação 8, e este resultado será equivalente a convolução, como é identificado na Equação 9.

$$\mathcal{L}^{-1}\{F(S)G(S)\} = (f * g)(t) \tag{8}$$

$$h(t) = \mathcal{L}^{-1}\{F(S)G(S)\} \tag{9}$$

Dado que a convolução em imagens é uma discretizada no tempo, as Equações 10, 11, 12 e 13 serão respectivamente.

$$y(n) = x[n] * h[n] = \sum_{\tau=0}^{n-1} x[\tau]h[n-\tau] \tag{10}$$

$$\mathcal{Z}\{x[n] * h[n]\} = Y[Z] = X(Z)H(Z) \tag{11}$$

$$\mathcal{Z}^{-1}\{X(Z)H(Z)\} = x[n] * h[n] \tag{12}$$

$$y(n) = \mathcal{Z}^{-1}\{X(Z)H(Z)\} \tag{13}$$

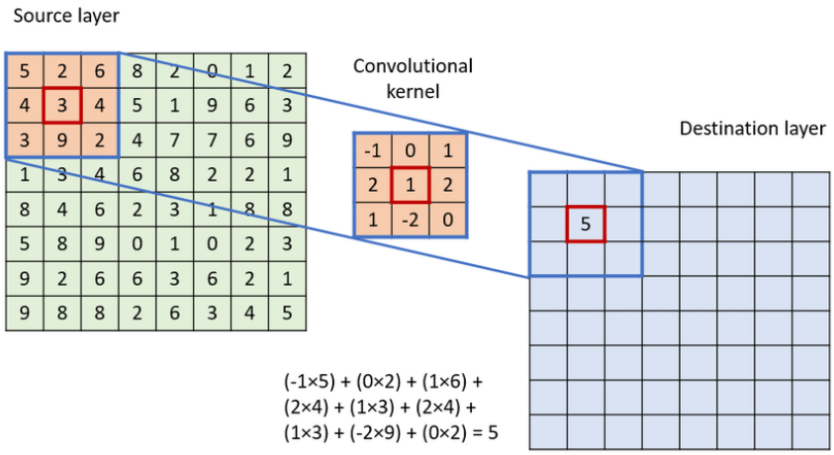
As definições de convolução serão usadas de forma prática nos próximos tópicos desta subseção.

### 1.6.2 Camada de Convolução

A convolução em imagens é a operação que realiza a transformação linear de uma matriz principal, se for uma representação de imagem RGB (Red-Green-Blue, em português: Vermelho-Verde-Azul) pode ser um conjunto de 3 matrizes, caso contrário, para

imagens em tons de cinzas ou preto e branco, um kernel ou matriz de convolução é formado pela matriz principal. Em seguida, a própria convolução é realizada, multiplicando a matriz principal pela matriz de convolução para obter uma terceira matriz que carrega as informações de característica da imagem ou matriz original (OLIVEIRA, 2022). A Figura 6 mostra como é feito o cálculo da convolução entre as matrizes descritas.

Figura 6 – Cálculo da Convolução em Imagens



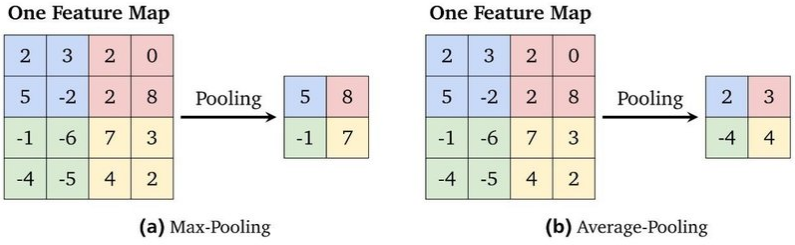

Fonte: (PODAREANU, 2019)

### 1.6.3 Camada de Pooling

As camadas convolucionais representam a presença de feições na imagem de entrada. Mas eles têm um problema, eles são sensíveis à localização do recurso no ponto de entrada. Isso nos fornece dados específicos em vez de dados generalizados, agrava o problema de overfitting e não fornece bons resultados em dados fora do conjunto de treinamento (SHARMA, 2022). Portanto, precisamos generalizar a existência de recursos. Isso é feito através da Camada de Pooling.

De maneira geral, a camada de Polling pode ser representada por meio de dois processos, o Max-Pooling e o Avg-Pooling, que representam respectivamente o máximo e a média do valor na região analizada da imagem. Essas definições são ilustradas na Figura 7.

Figura 7 – Representação do Max-Pooling e Avg-Pooling




Fonte: (GUISSOUS, 2019)

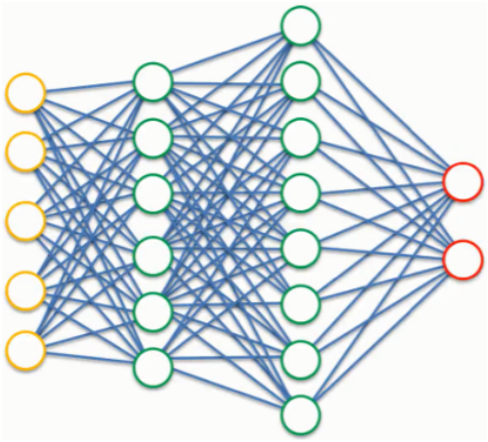
### 1.6.4 Camada Totalmente Conectada

A Camada Totalmente Conectada, como o próprio nome já diz, é a camada onde todos os neurônios estão conectados com os neurônios da camada anterior. Assim, todas as características adquiridas nas camadas anteriores são usadas para encontrar padrão para a imagem de entrada. Assim, existe uma fórmula que ajusta os pesos e tomada de decisão, esposta na Equação 14.

$$S = X * W + b \tag{14}$$

Onde, $S$ é a decisão, $X$ a informação de entrada, $W$ a matriz com os pesos das conexões e $b$ a tendência.

Figura 8 – Representação da Camada Totalmente Conectada



Fonte: (SUPERDATASCIENCE, 2018)

### 1.6.5 Camada de Softmax

A Camada de Softmax é a responsável por obter a probabilidade da imagem de entrada ser de determinada classe. Este cálculo é realizado conforme a Equação 15.

$$\phi_i = \frac{e^{Z_i}}{\sum_{j \in group} e^{Z_j}} \tag{15}$$

Onde $i$ representa o índice do neurônio de saída $\phi$ sendo calculado e $j$ representa os índices de todos os neurônios de um nível. A variável $Z$ designa o vetor de neurônios de saída.

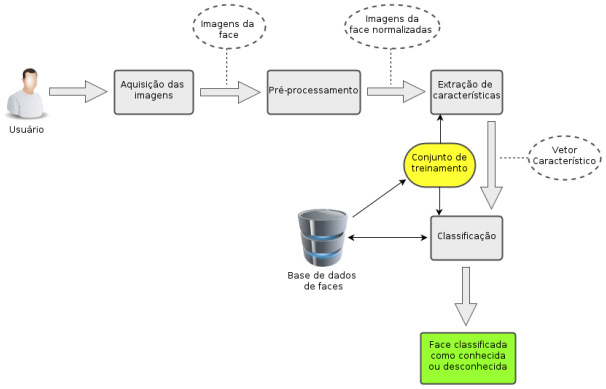
## 1.7 SISTEMA DE DETECÇÃO DE FACES

No fluxograma da Figura 9, são apresentadas as etapas para a realização das tarefas para a detecção das faces, onde primeiramente é feita a aquisição das imagens dos

usuários deste sistema, e essas imagens de face são pré-processadas, para normalizar e corrigir imagens, modificando-as para a escala de cinzas e alterando resolução, dimensão e iluminação, com o objetivo de extrair suas características. A partir disso, com a verificação da base de imagens, são realizadas as extrações de características, por meio das Redes Neurais Convolucionais, e por último a classificação, para identificar os indivíduos conhecidos.

Com esse treinamento, quando a câmera captura uma pessoa, esta é classificada como conhecida ou desconhecida, e para o sistema desenvolvido neste trabalho, caso o aluno seja conhecido, sua frequência na disciplina será atualizada.

Figura 9 – Etapas para Detecção de Face



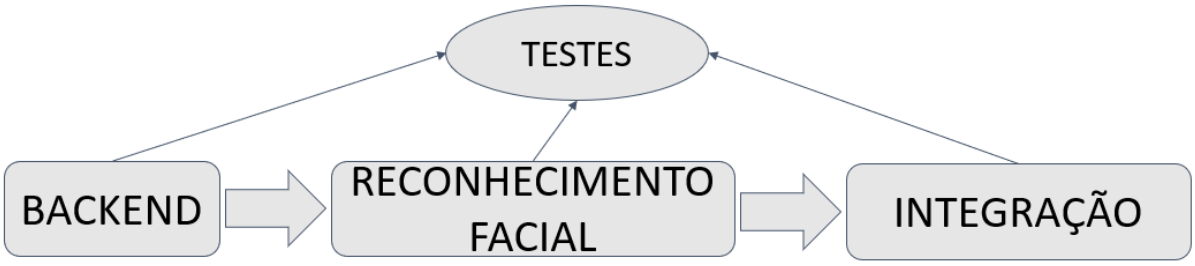

Fonte: (DINIZ, 2012)

# 2 METODOLOGIA

O planejamento de um projeto é uma etapa que deve anteceder as atividades do desenvolvimento, pois é nesta fase que são definidas as regras de negócio para as funcionalidades da aplicação.

Na Figura 10, estão identificadas as etapas do desenvolvimento da aplicação, primeiramente o desenvolvimento e teste do backend, depois o desenvolvimento e teste do algoritmo de reconhecimento facial, por último, a integração entre as duas etapas desenvolvidas anteriormente. Com isso, o sistema de frequência de alunos por meio do reconhecimento facial estará finalizado.

Figura 10 – Fluxograma do Desenvolvimento das Etapas




Fonte: Própria.

## 2.1 REQUISITOS DO PROJETO

Para implementação do projeto, é necessário primeiramente entender os requisitos. Para isso, verifica-se que a universidade é formada por alunos, professores, coordenadores de curso, diretores de unidades acadêmicas e o reitor da universidade, e para integrar a aplicação com todos estes membros, é necessário desenvolver o projeto que tenha o acesso de todos.

Com isso, verifica-se as Tabelas, de 1 a 5, onde está detalhado as funcionalidades para cada usuário deste projeto, e além de todos participantes citados no parágrafo anterior, faz-se necessário a presença de um funcionário responsável pela manutenção deste projeto, para a aplicação, este usuário pe chamado de administrador.

Tabela 1 – Acesso dos Professores

| | |
|---|---|
| Usuário | Professores |
| Posição | 1 |
| Descrição | Este usuário tem acesso as funcionalidades apenas quando é cadastrado previamente por um administrador na posição de professor. Com isso, tem acesso apenas para cadastro de horários de disciplina e consultas na base de dados das disciplinas que este usuário é professor. |
| Condição | Professores da UEA. |

Fonte: Própria.

Tabela 2 – Acesso dos Coordenadores de Curso

| | |
|---|---|
| Usuário | Coordenadores |
| Posição | 2 |
| Descrição | Este usuário tem acesso as funcionalidades apenas quando é cadastrado previamente por um administrador na posição de coordenador de curso. Com isso, tem acesso a todos os dados de alunos, professores e disciplinas referente ao curso que é coordenador. |
| Condição | Coordenadores de curso na UEA. |

Fonte: Própria.

Tabela 3 – Acesso dos Diretores de Unidade

| | |
|---|---|
| Usuário | Diretores |
| Posição | 3 |
| Descrição | Este usuário tem acesso as funcionalidades apenas quando é cadastrado previamente por um administrador na posição de diretor de unidade. Com isso, tem acesso a todos os dados de alunos, professores, disciplinas e cursos referente a unidade em que é diretor. |
| Condição | Diretores de unidade institucional na UEA. |

Fonte: Própria.

Tabela 4 – Acesso do Reitor da Universidade

| | |
|---|---|
| Usuário | Reitor |
| Posição | 4 |
| Descrição | Este usuário tem acesso as funcionalidades apenas quando é cadastrado previamente por um administrador na posição de reitor da universidade. Com isso, tem acesso a todos os dados de alunos, professores, disciplinas, cursos e as unidades institucionais da universidade. |
| Condição | Reitor da UEA. |

Fonte: Própria.

Tabela 5 – Acesso do Administrador

| | |
|---|---|
| Usuário | Administrador |
| Posição | 5 |
| Descrição | Ser responsável pela manutenção do sistema e resolver os problemas relatados dos usuários da aplicação.<br>Com isso, tem acesso a todos os dados do projeto, assim como acesso ao banco de dados do projeto para resolver problemas quando solicitado. |
| Condição | Funcionário responsável pela manutenção deste sistema. |

Fonte: Própria.

Tabela 6 – Verificação do Reconhecimento Facial

| | |
|---|---|
| Usuário | Alunos |
| Posição | Não Existe |
| Descrição | Ser aluno regular matriculado em um dos cursos da Universidade do Estado do Amazonas. tados dos usuários da aplicação.<br>Não tem acesso à aplicação, mas as imagens de suas faces serão usadas para treinar o algoritmo de reconhecimento facial, para fazer a verificação de presença de forma automática a partir da visualização de suas faces por um dispositivo eletrônico em sala de aula. |
| Condição | Aluno matriculado na UEA. |

Fonte: Própria.

O sistema é desenvolvido para disponibilizar acessos específicos de acordo com a posição que ocupa cada usuário, sendo assim, os usuáios que estão cadastrados com a posição 4 e 5, possuem os acessos mais completos, devido a posição de TI e Reitoria, respectivamente, enquanto os demais usuários tem acesso apenas aos dados em que eles são responáveis, como por exemplo Direção e Coordenação, 3 e 2 respectivamente, responsáveis pelos dados de seus setores.

## 2.2 MATERIAIS UTILIZADOS

Para realizar o desenvolvimento do projeto, foi utilizado o notebook Acer Aspire Nitro 5 AN515-54-58CL, que possui as seguintes configurações de hardware e software:

- Sistema Operacional Linux: Ubuntu 22.04 LTS;
- Processador Intel Core i5 de 9ª Geração;
- Placa de Vídeo Nvidia GTX 1650;
- Webcam com Resolução de 720p (1280 x 720 pixels);
- Memória RAM de 8GB;
- SSD de 128GB;
- HDD de 1TB;

# 3 IMPLEMENTAÇÃO

Este capítulo aborda o desenvolvimento de software do backend e da identificação facial da aplicação, de acordo com a metodologia citada anteriormente.

## 3.1 BANCO DE DADOS

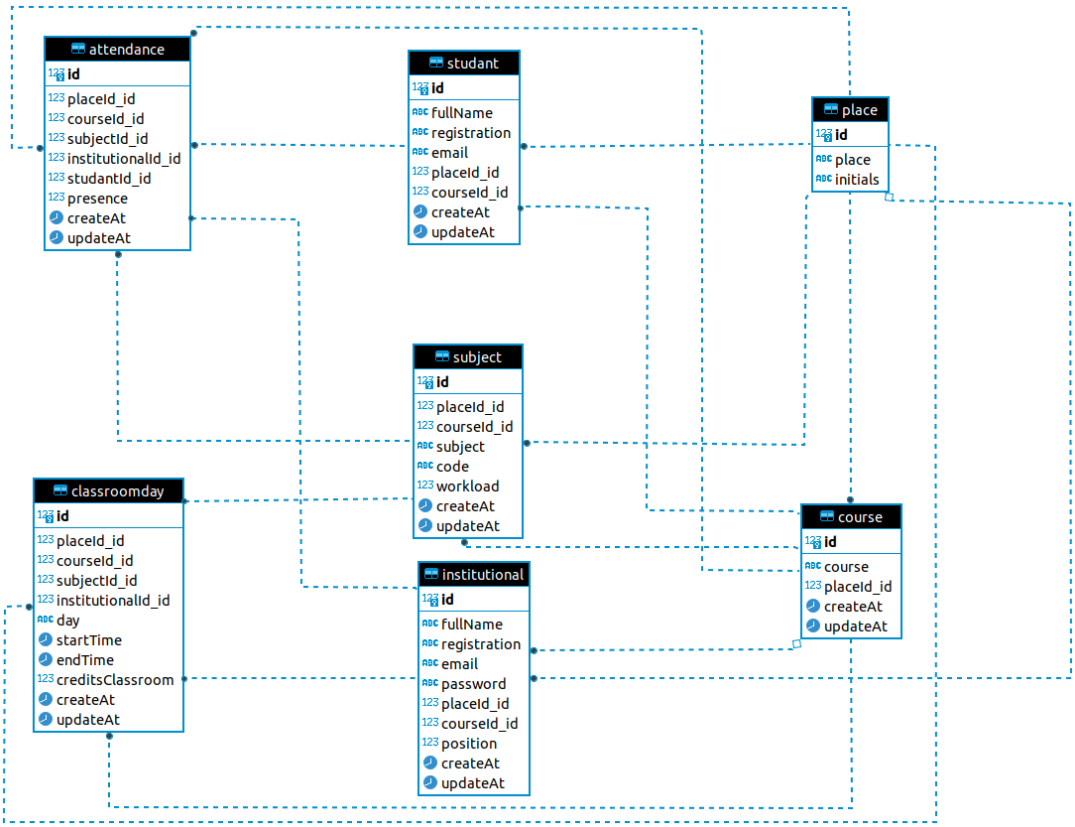
A primeira etapa da implementação foi o desenvolvimento da estrutura da aplicação, que foi a idealização das tabelas do banco de dados.

A partir disso, as tabelas foram desenhadas para conter as informações das unidades acadêmicas, dos cursos, dos estudantes, dos professores e departamentos institucionais, datas, horários e créditos das disciplinas e por último, da tabela que faz o controle da frequência dos alunos.

Esses dados são primordiais para o inicio do projeto, haja vista que todo o fluxo de teste e validação da aplicação necessita das informações.

Com essas condições iniciais, o banco de dados foi desenvolvido com as tabelas exibidas na Figura 11, onde as informações necessárias para cadastro de cada tabela estão mostradas dentro de cada card.

Figura 11 – Banco de Dados da Aplicação.




Fonte: Própria.

Tabelas do Banco de Dados:

- Place: Tabela para cadastro das unidades acadêmicas da universidade;
- Course: Tabela para cadastro dos cursos lecionados na universidade;
- Studant: Tabela para cadastro dos alunos da universidade;
- Subject: Tabela para cadastro das disciplinas lecionadas na universidade;
- Institutional: Tabela para cadastro dos perfis de professores, coordenadores de cursos, diretores de unidades acadêmicas e reitor da universidade;
- ClassrommDay: Tabela para cadastro e consulta das datas, horários, créditos/aula e professor das disciplinas da universidade;
- Attendance: Tabela para cadastro e atualização da frequência dos alubnos nas disciplinas da universidade;

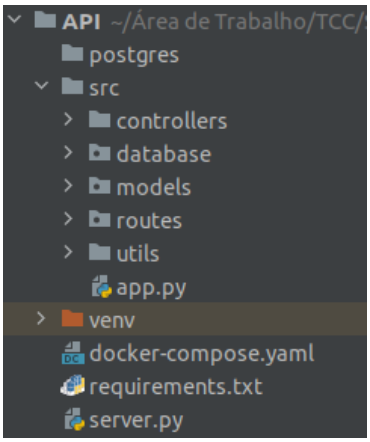
## 3.2 ARQUITETURA DO BACKEND

### 3.2.1 Diretórios da API

O backend deste projeto foi implementado usando o padrão MVC, dividindo a aplicação em blocos. Na pasta da API, verifica-se a presença das respectivas pastas: postgres (armazenar o banco de dados, bem como os dados salvos), src (armazenados os scripts responsáveis por fazer as regras de negócio do projeto), venv (ambiente virtual onde ficam armazenados os pacotes instalados, necessários para a funcionalidade das importações feitas no projeto), docker-compose.yaml (container para iniciar ou interromper o banco de dados da aplicação com um único comando), requirements.txt (armazena as versões e seus respectivos pacotes importados no projeto) e server.py (python file responsável por iniciar aplicação).

Na Figura 12, verifica-se a organização das pastas na IDE do PyCharm.

Figura 12 – Estrutura do Backend da Aplicação.

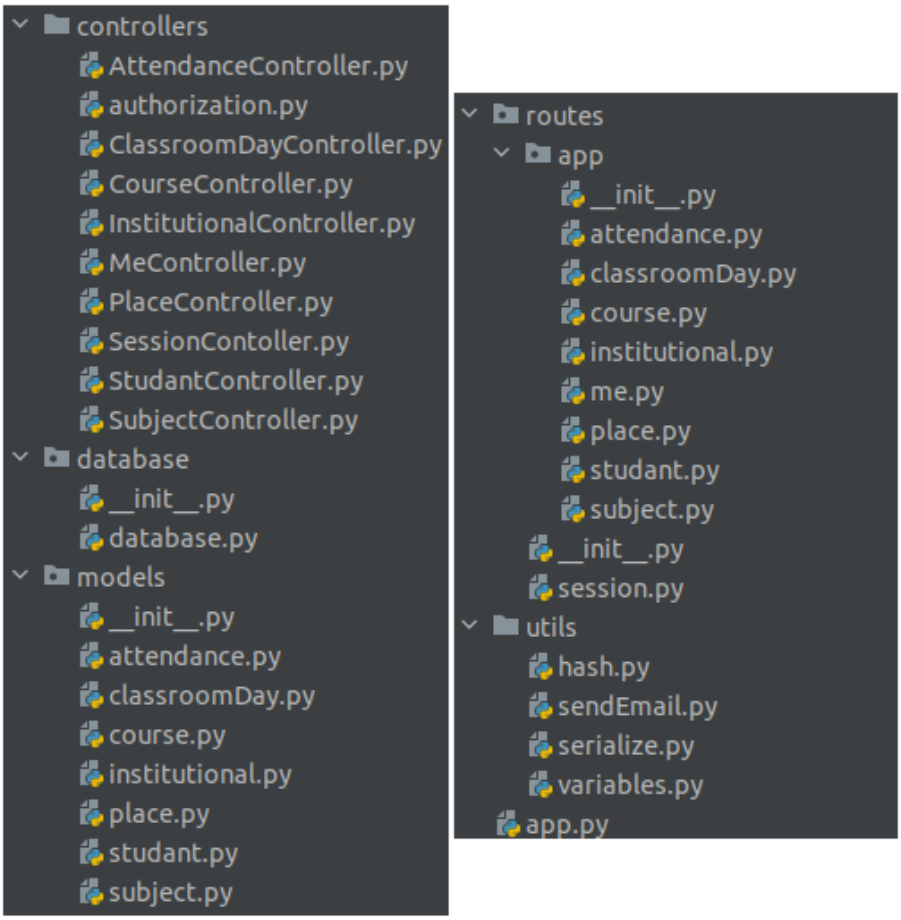



Fonte: Própria.

### 3.2.2 Diretório SRC

A pasta SRC (origem) do projeto é composta por controllers (faz o controle dos dados entre backend e frontend), database (faz a conexão com o banco de dados do projeto), models (cria as tabelas do banco usando o ORM Peewee), routes (fica armazenado as rotas dos endpoints da aplicação), utils (funções para armazenar variáveis, envio de emails, serialização e gerar e verificar hash para tokens) e app.py (conectar o framework Sanic com o servidor).

Figura 13 – Estrutura dos Diretórios da SRC.




Fonte: Própria.

## 3.3 ENDPOINTS

Os endpoints são as URLs para conexão com o backend, para fazer a solicitação a requisição de informações. A partir desse acesso, podem ser necessárias informações para envio para que as informações retornadas sejam de acordo com o que é desejado.

Os endpoints do backend serão usados numa atividade futura, para realizar a integração com o frontend (interface gráfica para uso do software), para finalizar o desenvolvimento deste projeto de forma completa.

Os endpoints foram desenvolvidos conforme as tabelas do banco, mostradas anteriormente, e têm as opções de retornar todos os dados do banco, retornar um dado específico conforme o id enviado, salvar e atualizar dados, para este projeto não foi feito endpoint para deletar dados, haja vista que este sistema é para verificação de frequência de alunos, e é necessário manter o histórico dos dados, sem existir a possibilidade de deletar algo.

Todos os endpoints desenvolvidos foram testados por meio do software Postman, para fazer as requisições para a API e retornar o resultado do backend, e assim, comparar com o resultado esperado. A seguir, na seção 3.4 estão disponíveis as funcionalidades do backend, bem como a exibição de alguns scripts usados para a requisição de informações dos endpoints.

## 3.4 FUNCIONALIDADES DO BACKEND

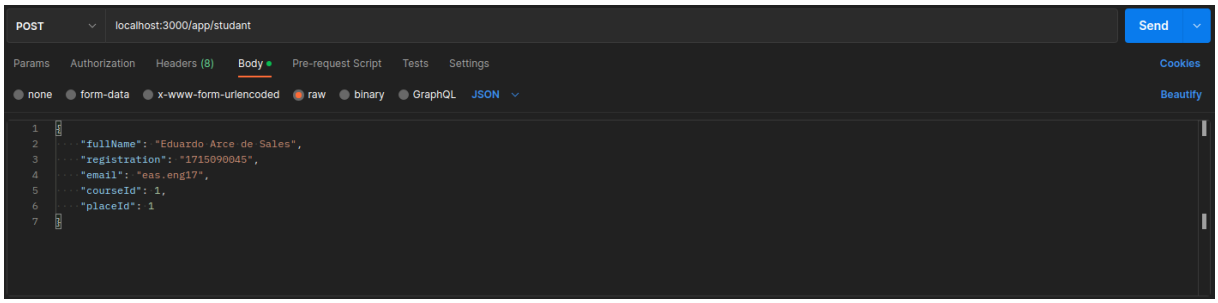
As funcionalidades deste projeto estão relacionadas ao fluxo de informações para marcação de presença do aluno e confirmação da presença para o aluno. Mas outras aplicações são para cadastro de novos usuários por um perfil de administrador, login, a existência de criptografia de senhas de usuário e o envio e verificação do hash dos tokens, mudança de senha e o envio de email para confirmação da presença do aluno.

### 3.4.1 Cadastro de Usuários

Neste projeto, o login de um usuário foi mockado (dado está fixo no backend), fazendo com que após a criação do banco, este usuário ja esteja cadastrado no sistema, este é o usuário do Admin (administrador).

O usuário Admin tem acesso a funcionalidade de cadastrar novos usuários, e o objetivo é que este usuário faça o cadastro dos professores e alunos, fornecendo aos professores cadastrados os seus emails de acesso (email institucional) e senha.

Figura 14 – Endpoint para Cadastro de Novos Usuários.

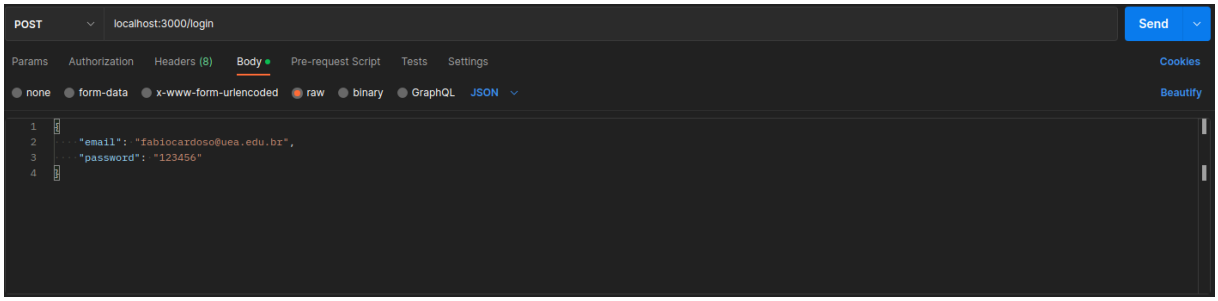



Fonte: Própria.

### 3.4.2 Login do Usuário e Esquecimento de Senha

O login de usuário é armazenado na pasta de SessionController, conforme a Figura 13, descrito no Apêncide A. Com isso, o usuário acessa a aplicação com seu login e senha, e será retornado os seus dados cadastrados, bem como um token de acesso com um tempo de expiração, limitando o acesso desse token para um intervalo de tempo predeterminado.

Figura 15 – Endpoint para Acessar a Aplicação




Fonte: Própria.

Caso o usuário esqueça sua senha de acesso, o endpoint ”recover-password”pode ser acessado e inserir seu email institucional (função de modificação de senha exibida no Apêncice D), dessa forma, caso este usuário esteja cadastrado na base de dados do projeto, um email será enviado com um link de acesso para modificação de senha, com um token que possui um período de expiração muito curto, com o objetivo de reduzir ao máximo a possibilidade de fraudes, havendo tempo suficiente apenas para o prório usuário fazer a modificação de sua senha.

Na seção 3.4.3, estão disponíveis informações mais detalhadas sobre os tokens descritos neste tópico.

### 3.4.3 Criptografia de Senhas e Hash de Tokens

Para segurança dos dados dos usuários desta aplicação, todas as senhas são criptografadas. Para criptografar e verificação de criptografia de senhas é usada a função descrita no Apêndice B, com isso, quando um usuário é criado, a função ”createCriptography”é usada para salvar a senha do usuário criptografada no banco de dados, e quando o usuário vai acessar a aplicação, a senha recebida no backend é comparada com a senha armazenada no banco de dados, através da função ”verifyCriptography”.

Para a criação de hash de tokens, são analisadas duas situações, pois ambos a primeira está demonstrada no Apêndice C, onde um hash é criado para o usuário acessar sua conta na aplicação, então este token é disponibilizado com um intevalo de expiração longo (7 dias), para melhorar a usabilidade e evitar que a senha seja solicitada diversas vezes,

enquanto a segunda está no Apêndice D, o hash é criado por conta do esquecimento da senha do usuário, então o período de validade do token é bastante curto (15 minutos), tempo suficiente apenas para o usuário acessar seu email e fazer a mudança de sua senha.

### 3.4.4 Filtros no Banco de Dados

O filtro no banco de dados é feito na tabela Attendance, pois esta é a tabela principal do armazenamento dos dados de frequência de alunos, nesta tabela são realizados filtros conforma a unidade acadêmica, curso, disciplina, professor e aluno.

Neste filtro, existem algumas restrições, pois o acesso do usuário institucional é dividido em categorias, onde pode ser: reitoria da universidade, diretoria de unidade acadêmica, coordenação de curso ou professor acadêmico, onde cada posição tem um acesso diferente, ou seja, o login da reitoria da universidade pode fazer filtros de frequência de todas as unidades acadêmicas, coordenações de curso, professores, disciplinas ou alunos, assim como a direção de unidade pode filtrar por apenas por coordenações, professores, disciplinas ou alunos relacionados a sua unidade, coordenações de cursos filtram por professores, disciplianas e alunos vinculados ao curso e professores tem acesso apenas as disciplinas que ele leciona.

### 3.4.5 Envio de Email

O envio de email da aplicação é utilizado para confirmar a presença dos alunos após o final da aula, conforme a função exposta no Apêndice E. Esta função recebe parâmetros como o email e nome do aluno, a disciplina, a quantidade de aulas presentes e o total de aulas lecionadas, e a partir disso, é calculada a frequência do aluno até o momento, e enviado um email com essas informações, para que o estudante tenha o controle de sua participação nas aulas.

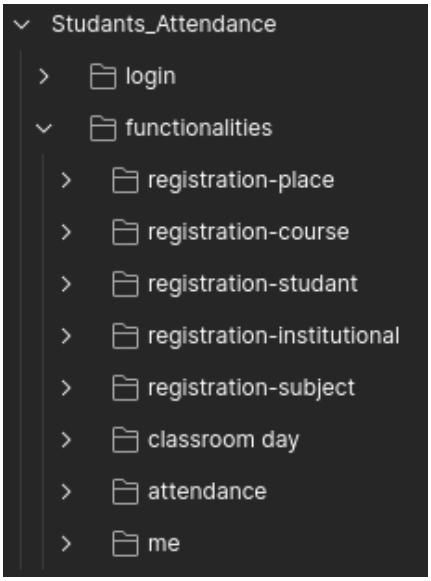
### 3.4.6 Configurações de Instalação

Para fazer a instalação do projeto, basta acessar o link do do projeto no GitLab e fazer um clone para um repositório local, com isso, basta fazer a instalação dos pacotes com as respectivas versões informadas no Apêndice F. Para fazer a instalação desses pacotes, basta apenas escolher ambiente virtual (venv) e executar o comando ”pip install -r requirements.txt”no terminal do repositório principal do projeto.

## 3.5 TESTES DO BACKEND

Os testes da API foram realizados usando a plataforma Postman, e os endpoints foram divididos em pastas, para facilitar os diferentes testes a ser realizados. Na Figura 16 são exibidas as pastas de teste no Postman.

Figura 16 – Estrutura dos Endpoints no Postman




Fonte: Própria.

Os testes foram realizados na ordem das pastas da Figura 16 (de cima para baixo), respeitando a ordem de acesso do usuário.

- Login: Teste de login e recuperação de senha;
- Registration-Place: Cadastro das unidades acadêmicas no banco de dados;
- Registration-Course: Cadastro dos cursos acadêmicos no banco de dados;
- Registration-Studant: Cadastro dos estudantes no banco de dados;
- Registration-Institucional: Cadastro dos professores e cursos no banco de dados;
- Registration-Subject: Cadastro das disciplinas no banco de dados;
- Classroom Day: Cadastro, atualização e buscas por horários das disciplinas e professor responsável;
- Attendance: Cadastro, atualização, buscas por horários das disciplinas, professor responsável, alunos e suas frequências de presença na aula;
- Me: Atualização de dados pessoais do usuáio e buscas específicas por unidade, curso, professor ou aluno;

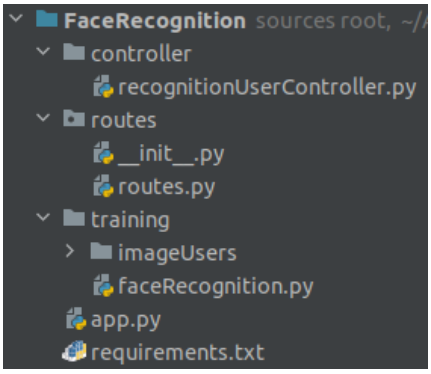
## 3.6 RECONHECIMENTO FACIAL

Nesta seção, será exposta a implementação do reconhecimento facial, para identificar os usuários da aplicação a partir de imagens dos seus rostos.

### 3.6.1 Estrutura dos Diretórios

A etapa do reconhecimento facial foi construída no mesmo formato do Backend, com diretórios para o controlador e rota do endpoint, há também a pasta de treino, onde está armazenado o diretório de imagem dos usuários e o arquivo de treino e identificação de usuário. Por último, há o arquivo principal que inicia o projeto, app.py e o arquivo de versão dos pacotes usados no desenvolvimento desta etapa do projeto. Na Figura 17, está mostrada a imagem da estrutura do projeto do reconhecimento facial.

Figura 17 – Estrutura dos Diretórios do Reconhecimento Facial




Fonte: Própria.

### 3.6.2 Treino

O treino das imagens foi realizado a partir do acesso da pasta ”imageUsers”, verificando todas as imagens do diretório e em uma lista armazena o codificação das imagens dos usuários e em outra lista, fica armazenada o nome de cada imagem codificada, de forma que as listas estão numa relação de 1 para 1, porque a posição da lista de codificação das faces é equivalente a posição na lista da com o nome referente a identificação das faces. Esta definição está a mostra no Apêndice G.

### 3.6.3 Identificação

Para realizar a identificação do usuário, primeiramente é exibida sua face para a uma câmera vinculada ao projeto, a partir disso, será comparado características das codificações das imagens do usuárias do treino, guardadas na lista, com a imagem recebida do usuário pela câmera, que também será codificada, ou seja, será comparado as codificações das imagens, conforme exposto no Apêndice H. A partir disso, o usuário será identificado com o nome cadastrado nas imagens de treino.

# 4 RESULTADOS

Nesta seção serão expostos separadamente os resultados das duas etapas desenvolvidas neste projeto, o Backend e o Reconhecimento Facial.

## 4.1 BACKEND

Neste tópico, estão identificados todos os endpoints da backend da aplicação, com os respectivos resultados dos testes.

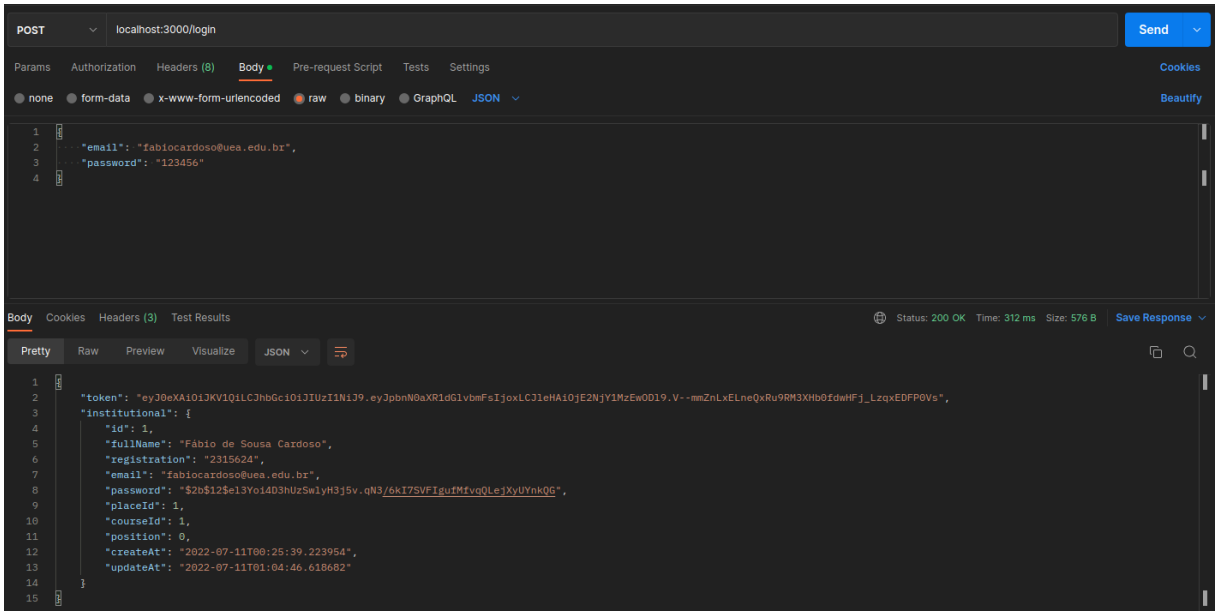
### 4.1.1 Teste dos Endpoints

Para execução do teste dos endpoints, será usado o software Postman, ferramenta necessária para realizar testes de API, onde são criados os endpoints para envio de requisições para o backend, de acordo com o formato do objeto pré estabelecido no desenvolvimento da aplicação.

- Login:

Para teste e validação do endpoint de login, é necessário inserir o email e a senha que foram cadastradas anteriormente, e caso esses dados recebidos estejam de acordo com o que está salvo para este usuário no banco de dados, são retornados os dados deste usuário, juntamente com um token de acesso, conforme a Figura 18, para garantir que o usuário permaneça com acesso liberado em sua conta, na aplicação, com tempo máximo de 7 dias após o primeiro acesso.

Figura 18 – Login na Aplicação com Email de Usuário e Senha

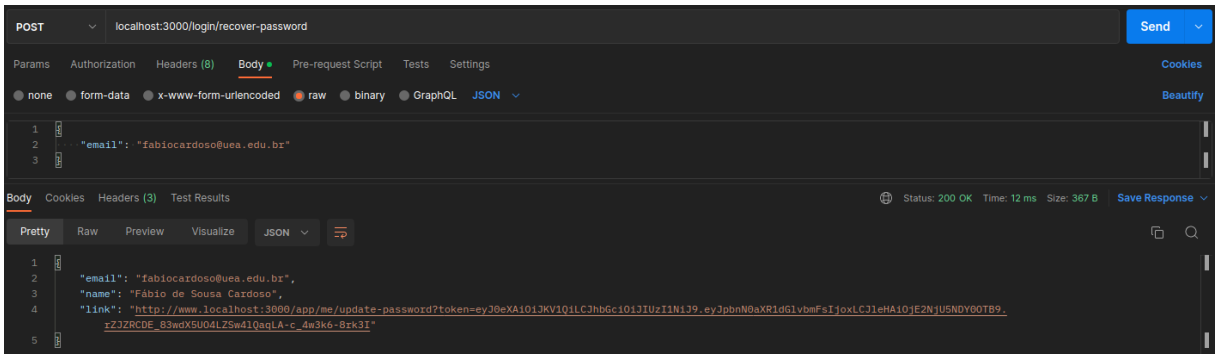



Fonte: Própria.

- Recuperar Senha:

No endpoint de recuperação de senha, o usuário deve informar seu email cadastrado na aplicação, com isso, será buscado seus dados pessoais e um link para recuperação de senha é criado, de acordo com a Figura 19, essas informações são necessárias para fazer o envio para o email do usuário, para que faça a modificação de senha, com tempo de expiração de 15 minutos, após a criação do token.

Figura 19 – Recuperar Senha de Usuário



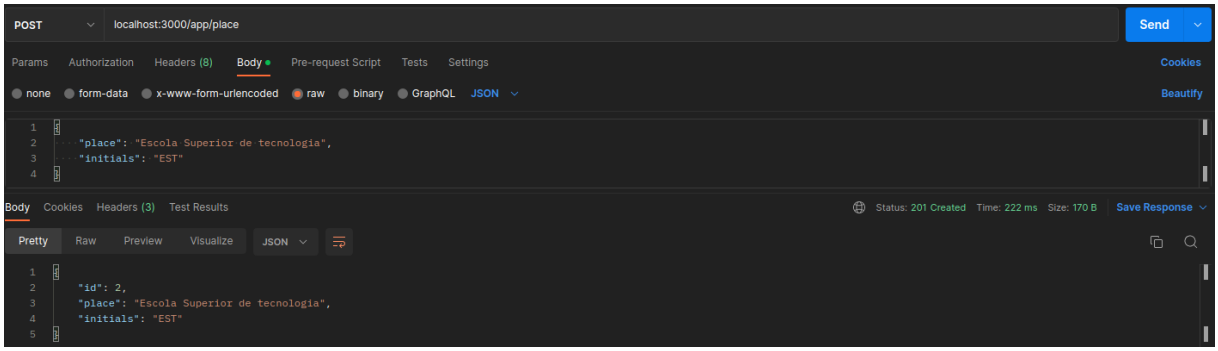

Fonte: Própria.

- Cadastro de Novas Unidades Acadêmicas:

Para uso da aplicação, é necessário que sejam cadastradas as unidades acadêmicas da universidade, e este endpoint faz o cadastro da sigla e do nome compledo da unidade, conforme a Figura 20, para que seja usado como parâmetro de definição de local para os demais endpoints desenvolvidos.

Outros endpoints desenvolvidos neste mesmo controller das unidades acadêmicas fazem a atualização e a exclusão dos dados, para que sejam usados quando necessário.

Figura 20 – Cadastro de Novas Unidades Acadêmicas

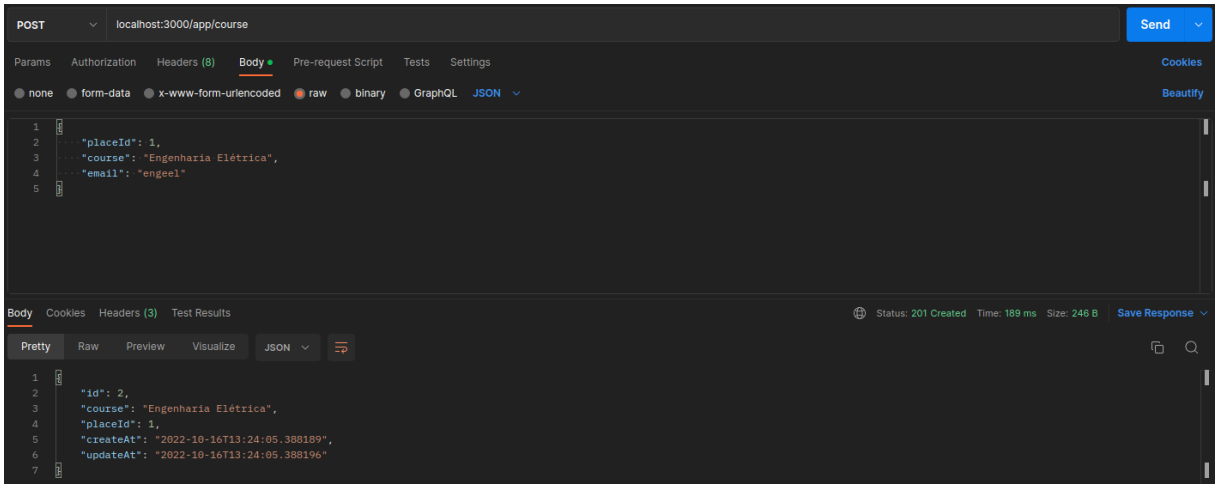



Fonte: Própria.

- Cadastro de Cursos:

Com o mesmo objetivo do tópico anterior, este endpoint é usado para fazer o cadastro dos cursos fornecidos pela universidade, de acordo com a Figura 21, onde são necessários as informações do nome do curso e do email, para realização do cadastro.

Figura 21 – Cadastro de Novos Cursos

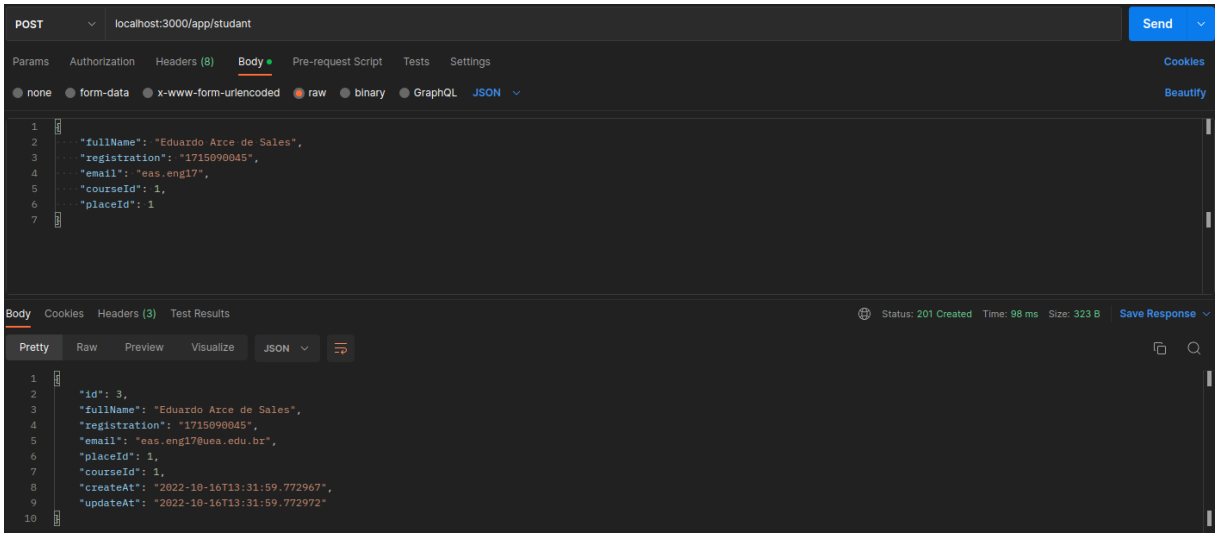



Fonte: Própria.

- Cadastro dos Alunos:

Neste endpoint, os dados pessoais do alunos são cadastrados para que tenha acesso as ferramentas da aplicação, de acordo com a Figura 22. Com isso, é salvo no banco de dados o nome completo, matrícula, email, unidade e curso.

Figura 22 – Cadastro de Novos Alunos

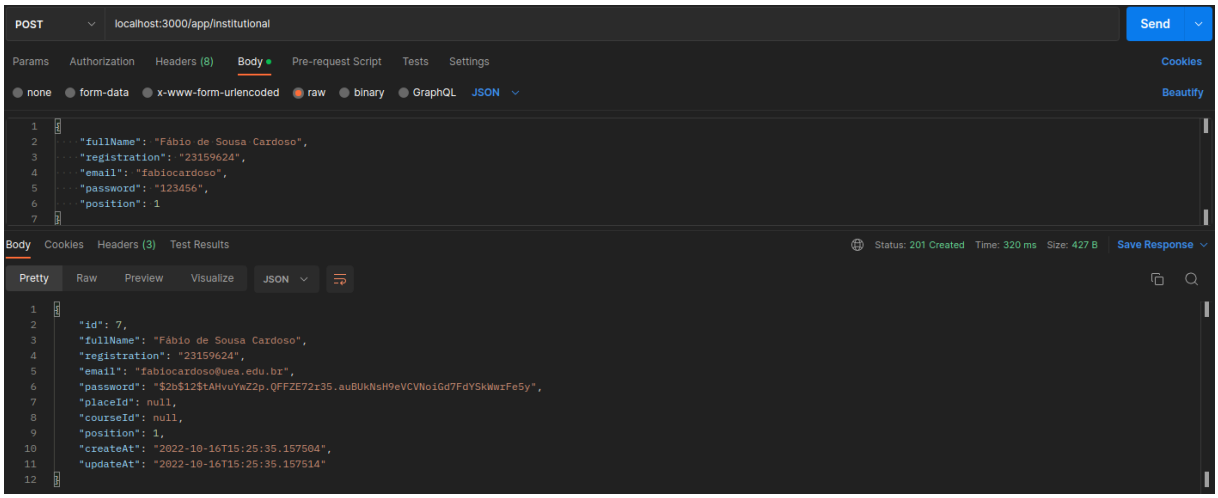



Fonte: Própria.

- Cadastro dos Professores:

Neste endpoint, os dados dos professores da instituição são cadastrados na base de dados da aplicação, de acordo com a Figura 23.

Figura 23 – Cadastro de Novos Professores

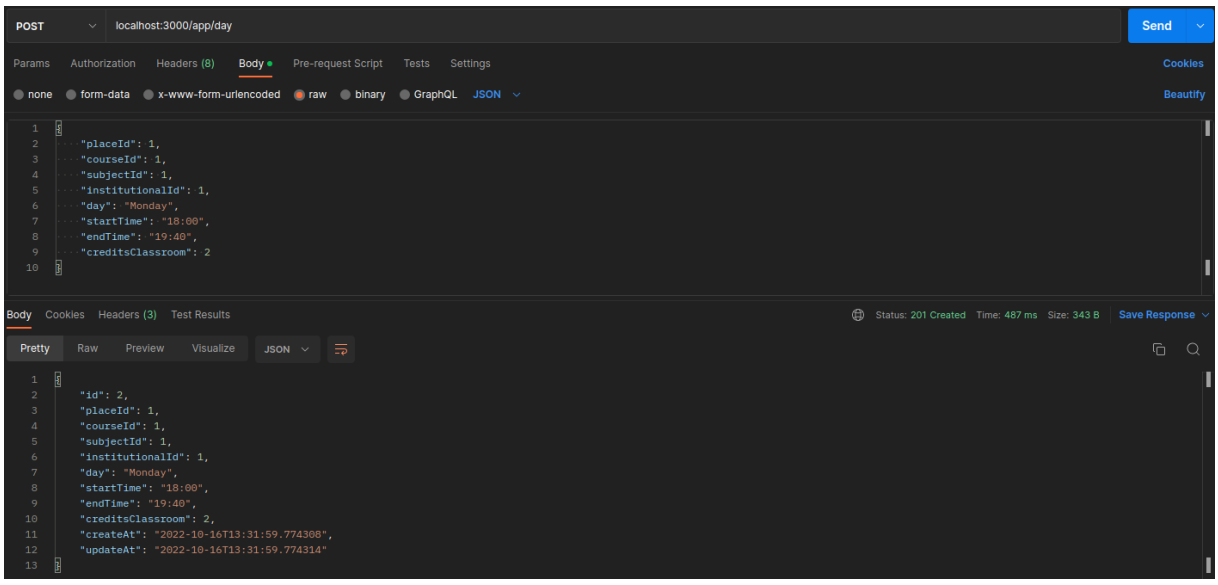



Fonte: Própria.

- Cadastro dos Turmas:

Neste endpoint, são cadastrados os horários das disciplinas, de acordo com a Figura 24. Para isso é necessário inserir os dados referente a unidade, curso, discipina e professor responsável, a quantidade de créditos por aula e o horário de início e fim da aula.

Figura 24 – Cadastro de Novos Cargos Institucionais

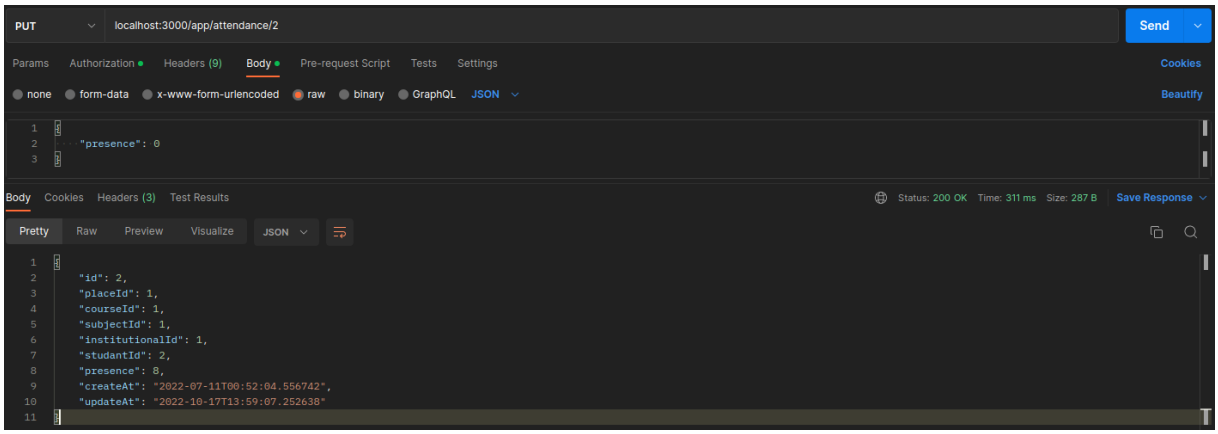



Fonte: Própria.

- Atualização de Freqência:

Neste endpoint, é realizado a atualização da frequência do aluno, por meio do Id cadastrado, de acordo com a Figura 25. Esse endpoint é acionado quando o aluno é identificado através do reconhecimento facial, e a partir da crédito por aula cadastrado no endpoint do tópico anterior, a frequência do aluno é acrescentado com este crédito da disciplina.

Figura 25 – Atualização da Frequência do Aluno



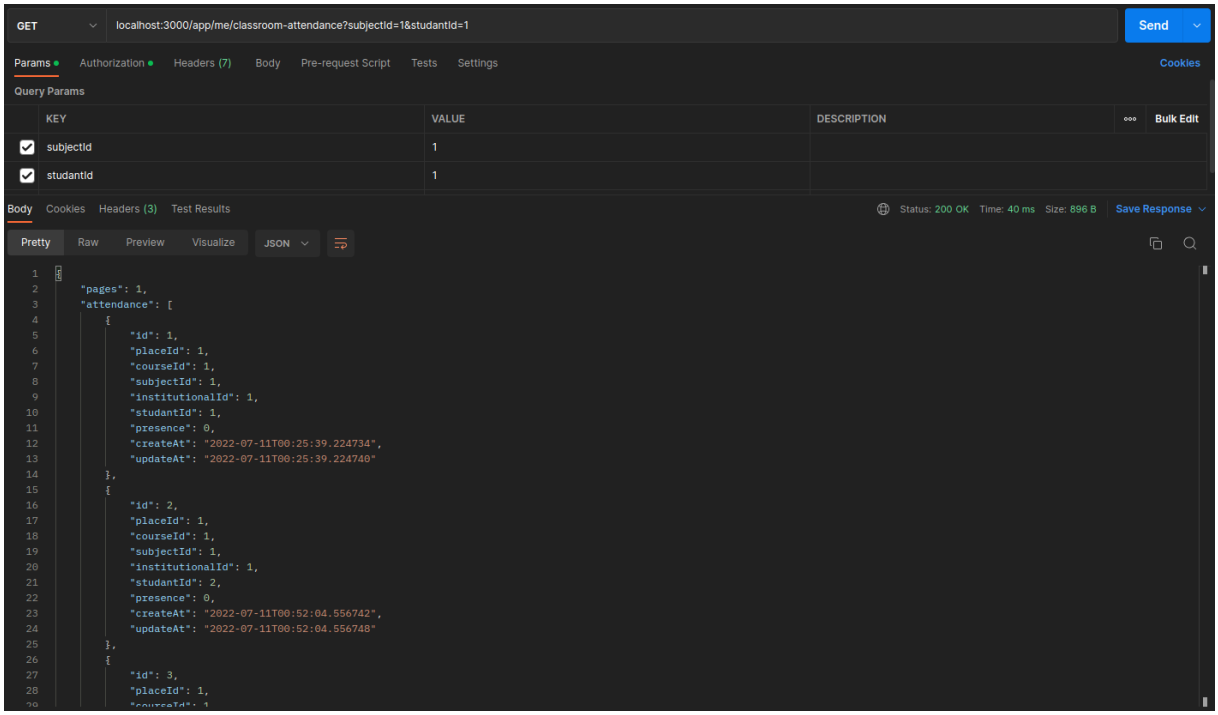

Fonte: Própria.

- Filtro dos Dados de Alunos, Professores, Cursos, e Unidades:

Neste endpoint, são feitos diversos filtros, de acordo com a Figura 26. Podem ser realizados filtros apenas por unidades acadêmicas, por cursos, professores ou alunos, para verificação e acompanhamento da frequência dos alunos de todas as setores citados.

Não necessáriamente é necessário que sejam selecionados todos os casos, pode ser escolhido apenas um destes, para buscar as informações da presença dos alunos em determinada disciplina ou em determinado curso.

Figura 26 – Filtro de Alunos, Professores, Cursos e Unidades

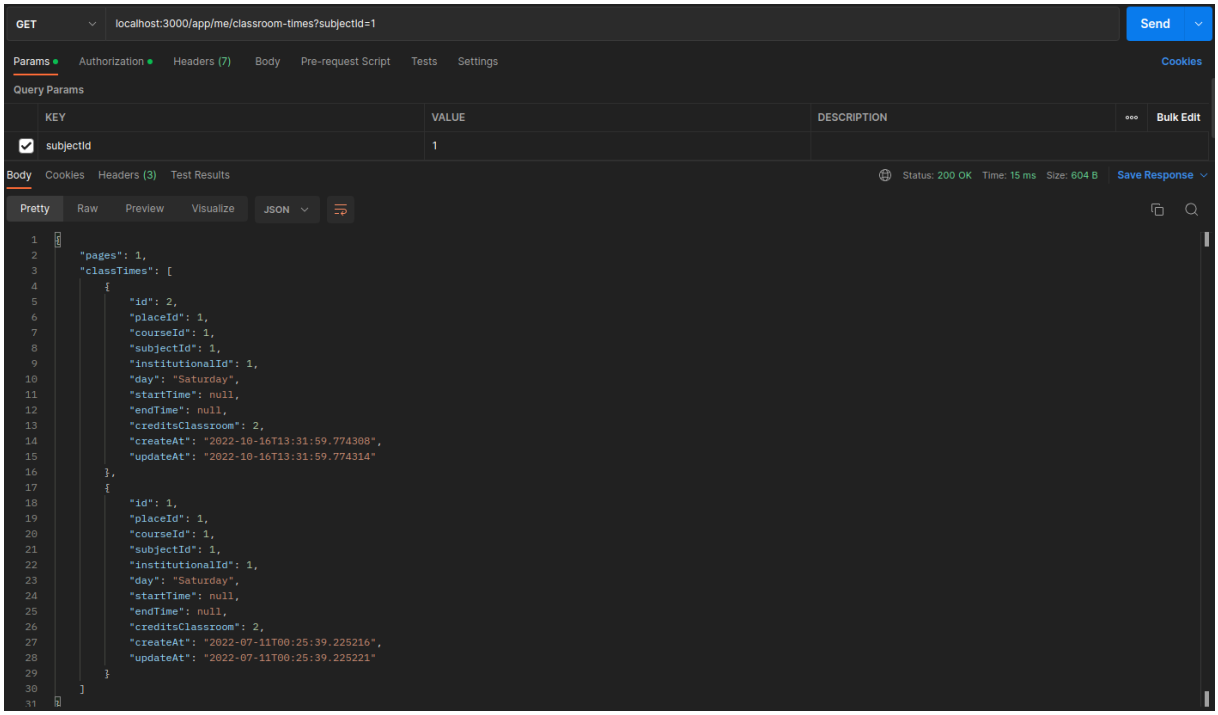



Fonte: Própria.

- Filtro dos Dados dos Horários das Turmas:

Neste endpoint, o professor ao acessar a aplicação, pode verificar as disciplinas e os horários que ele deve ministrar as aulas, de acordo com a Figura 27. Além disso, podem ser aplicados filtros por disciplina e dia da semana, para identificar as aulas que devem ser realizadas no dia desejado.

Figura 27 – Filtro dos Horários das Turmas

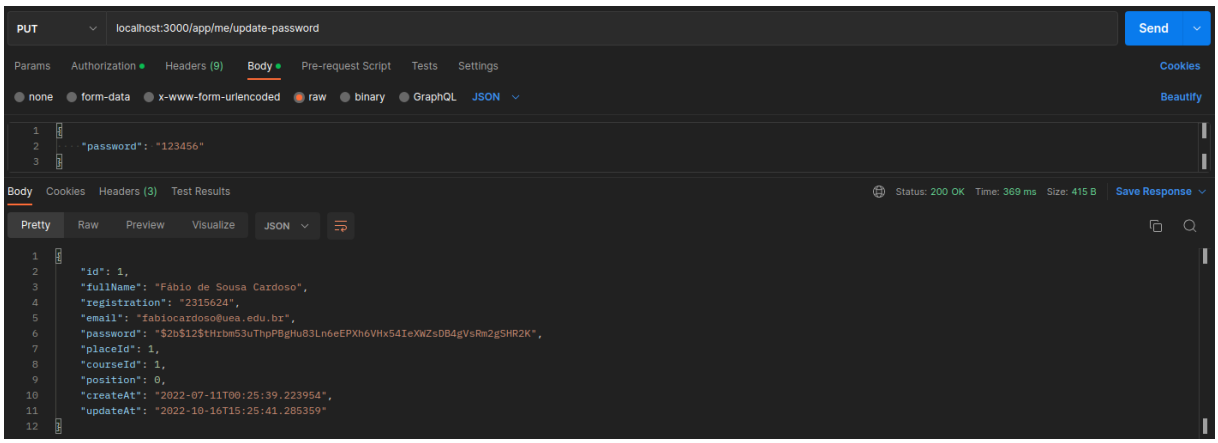



Fonte: Própria.

- Atualização de Senha:

Neste endpoint, é possível fazer a modificação de senha dentro da aplicação, de acordo com a Figura 30. Basta acessar este endpoint e inserir a nova senha desejada, com isso, a nova senha é criptografada e atualizada no banco de dados.

Figura 28 – Atualização de Senha




Fonte: Própria.

### 4.1.2 Métodos HTTP dos Endpoints

Nesta seção, estão identificados na Tabela 7 todos os endpoints desenvolvidos, os métodos utilizados e os resultados dos testes.

Tabela 7 – Métodos HTTP dos Endpoints

| Endpoints | Método | Resultado |
|---|---|---|
| *http://localhost:3000/login* | POST | PASS |
| *http://localhost:3000/login/recover-password* | POST | PASS |
| *http://localhost:3000/app/place* | POST | PASS |
| *http://localhost:3000/app/course* | POST | PASS |
| *http://localhost:3000/app/studant* | POST | PASS |
| *http://localhost:3000/app/institutional* | POST | PASS |
| *http://localhost:3000/app/subject* | POST | PASS |
| *http://localhost:3000/app/day* | POST | PASS |
| *http://localhost:3000/app/attendance* | POST | PASS |
| *http://localhost:3000/app/attendance* | PUT | PASS |
| *http://localhost:3000/app/me* | GET | PASS |
| *http://localhost:3000/app/me* | PUT | PASS |
| *http://localhost:3000/app/me/general* | GET | PASS |
| *http://localhost:3000/app/me/classroom-attendance* | GET | PASS |
| *http://localhost:3000/app/me/classroom-times* | POST | PASS |
| *http://localhost:3000/app/me/classroom-times* | PUT | PASS |
| *http://localhost:3000/app/me/update-password* | PUT | PASS |

Fonte: Própria.

## 4.2 RECONHECIMENTO FACIAL

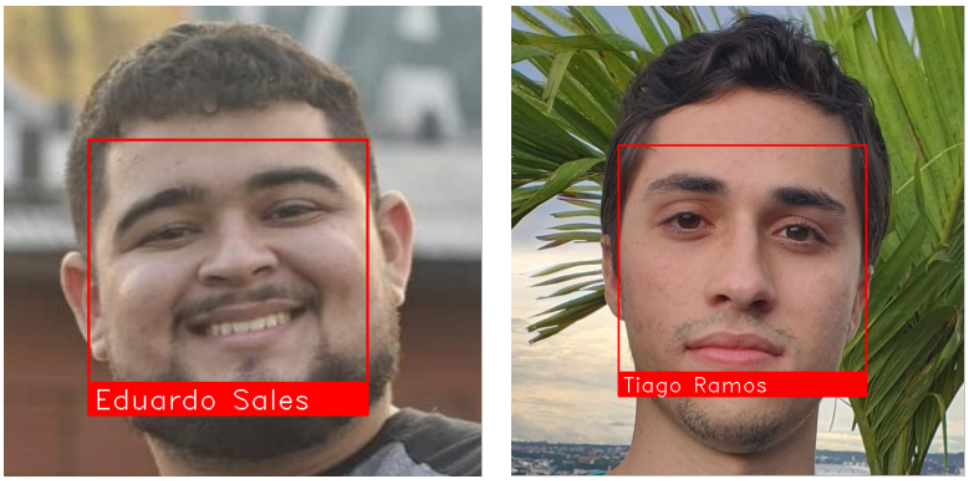
Nesta seção, serão apresentados os resultados do que foi desenvolvido na etapa do reconhecimento facial.

### 4.2.1 Detecção de Imagens dos Usuários

No teste do algoritmo de detecção das faces, foram utilizadas imagens dos usuários, para simular cenários de imagens que o endpoint de reconhecimento facial recebe do backend, dessa maneira, o algoritmo faz a identificação das faces e a detecção dos usuários previamente treinados.

Na Figura 29 está exibida a identificação de dois usuários treinados na aplicação, de forma que, quando o algoritmo recebe uma imagem desses usuários, é possível identificar cada um.

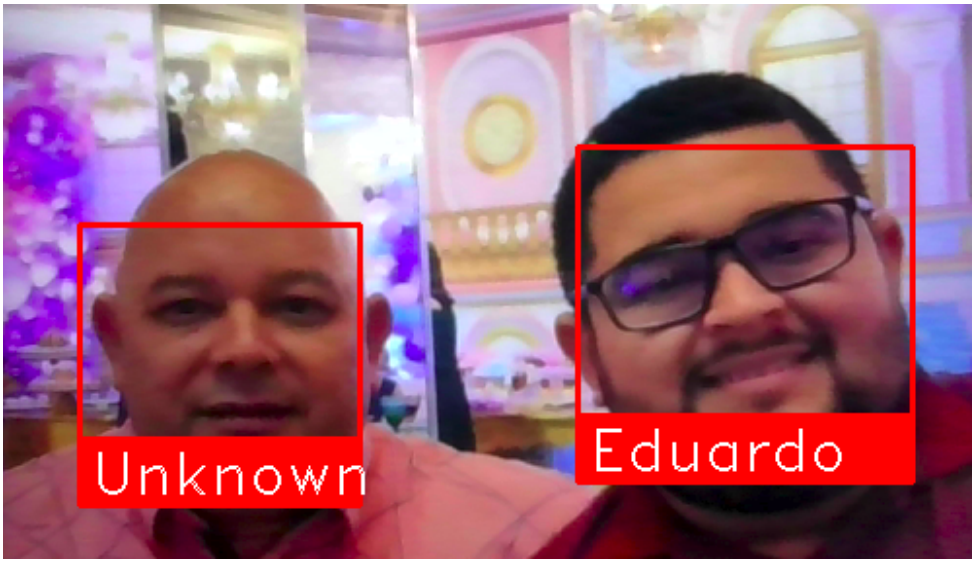
Figura 29 – Detecção de Usuários Cadastrados




Fonte: Própria.

Caso o algorimo receba uma imagem que tenha pessoas que foram treinadas e que não foram treinadas, será feito primeiramente a identificação das faces da imagem, e depois cada imagem será consultada no treino, de forma que aquele que seja identificado, será retornado seu nome enquanto o desconhecido, é retornado uma mensagem padrão de "Unknow", ou seja, desconhecido pelo treino do algoritmo.

Figura 30 – Detecção de Usuários Não Cadastrado e Cadastrado




Fonte: Própria.

Ou seja, para todos os casos, o algoritmo de reconhecimento facial retornará uma resposta, seja um usuário conhecido, retornando seu nome, ou de um usuário desconhecido, retornando "Unknow".

### 4.2.2 Acertividade do Reconhecimento Facial

Nos testes desta etapa da aplicação, foram usadas imagens de 50 usuários para o treino do algoritmo. A partir disso, pode-se verificar o resultado do grau de acertividade ao reconhecimento facial desenvolvido através da Tabela 8.

Tabela 8 – Acertividade do Algoritmo de Reconhecimento Facial

| Usuários Treinados | Acertos | Falhas | Acertividade |
|---|---|---|---|
| **50** | **43** | **7** | **86%** |

Fonte: Própria.

## 4.3 COMENTÁRIOS DO AUTOR

Após a implementação das atividades descritas neste capítulo, era previso a integração entre o endpoint de atualização de frequência do backend e o algoritmo de detecção facial, mas por conta do atraso no desevolvimento do backend, não foi possível concluir a integração do projeto, e por conta disso, ficou concluído apenas o backend e o reconhecimento facial de forma independente.

# 5 CONCLUSÃO

Para os objetivos propostos por este trabalho, conseguiu-se fazer o desenvolvimento de vários endpoints do backend, para realizar todo o fluxo de dados, desde o cadastro dos alunos, passando pela atualização e busca dos dados, além disso, são feitas validações dos tokens para liberar login ao usuário e também a criptografia de senhas, gerando segurança para o acesso na aplicação. E para o algoritmo de detecção facial, foi alcançado uma acertividade de 86% para os usuários treinados, uma taxa bastante satisfatória levando em consideração as espectativas iniciais do desenvolvedor.

Ao iniciar o desenvolvimento do projeto, a meta inicial era realizá-lo em 3 etapas: Backend, Reconhecimento Facial e a Integração dos itens anteriores. Para o backend, os endpoints foram desenvolvidos e testados em um tempo maior do que foi estipulado, com isso, gerou um atraso nas atividades, após isso, iniciou-se a etapa do reconhecimento facial, e o período de desenvolvimento desta etapa ficou dentro do que foi estipulado, mas por conta do atraso da primeira atividade, não foi possível realizar a atividade da integração e teste entre o backend e o reconhecimento facial.

Este é um projeto para armazena os dados da presença dos alunos nas disciplinas em um banco de dados, e a partir dessas informações, novas funcionalidades podem ser criadas, permitindo a partir dos dados, fazer análises, projeções e acompanhamento de alunos, assim, facilita o monitoramento do desempenho dos alunos nos cursos, criando a oportunidade de adicionar a esta verificação, um acompanhamento psicológico quando o desempenho de um aluno cair abruptamente, permitindo que a coordenação tenha essa informação e entre em contato com o aluno para identificação do problema.

Este projeto foi desenvolvido tomando como base a UEA, mas pode ser reaproveitado em outras universidades ou escolas, através de adaptações.

## 5.1 MELHORIAS E TRABALHOS FUTUROS

O desenvolvimento da etapa do reconhecimento facial pode ser melhorado com aumentando do número de imagens por usuário, para que o algoritmo tenha mais imagens para treinar a aprendizagem das faces de cada usuário, e outra melhoria deve ser feita no armazenamento dos dados do treino, que atualmente são guardados em duas listas, onde em uma lista ficam as características dos usuários e em outra os nomes dos usuários, o ideal é que após o treino das faces, o algoritmo tenha os dados das características dos usuários salvos em um arquivo XML, para que quando necessário fazer a verificação de um usuário, apenas será comparada as características da imagem recebidas com o que está slavo no arquivo XML.

Para trabalhos futuros, recomenda-se fazer a integração entre os endpoints do backend e do reconhecimento facial, para ter este projeto funcionando em sua totalidade, e além

disso, criar um design para desenvolver algumas telas para o frontend da aplicação, para integrar com os endpoints do backend, para que o projeto esteja plenamente concluído e pronto para subir para produção.

# REFERÊNCIAS


CAVALCANTE, P. H. A. ***Tutorial PostgreSQL: introdução prática ao serviço***. 2021. Disponível em: ⟨https://blog.geekhunter.com.br/tutorial-postgresql-introducao-pratica-ao-servico/⟩. Acesso em: 10 abr. 2022.

DIGIFORT. ***Marcação de Ponto por Registro Eletrônico com Reconhecimento Facial***. 2019. Disponível em: ⟨https://www.digifort.com.br/reconhecimento-facial.php⟩. Acesso em: 28 abr. 2022.

DINIZ, F. A. ***RedFace: Um Sistema de Reconhecimento Facial para Identificação de Estudantes em um Ambiente Virtual de Aprendizagem***. Porto Alegre: CINTED-UFRGS, 2012.

EBERMAM, E.; KROHLING, R. A. ***Uma Introdução Compreensiva às Redes Neurais Convolucionais: Um Estudo de Caso para Reconhecimento de Caracteres Alfabéticos***. Vitória: Trilha Principal, 2018.

EWALLY. ***Back-end: O Que É, Para Que Serve e Quais Suas Linguagens?*** 2021. Disponível em: ⟨https://www.ewally.com.br/blog/ajudando-sua-empresa/backend/\#:~:text=O\\%20backend\\%20\\%C3\\%A9\\%20a\\%20estrutura,ambientes\\%20eletr\\%C3\\%B4nicos\\%20operem\\%20em\\%20sincronia.⟩ Acesso em: 10 abr. 2022.

GOMES, R. G. ***Modelagem de dados: 1:N ou N:N?*** 2019. Disponível em: ⟨https://www.devmedia.com.br/modelagem-1-n-ou-n-n/38894⟩. Acesso em: 14 mai. 2022.

GREGERSEN, E. ***Relational Database***. 2020. Disponível em: ⟨https://www.britannica.com/technology/relational-database⟩. Acesso em: 15 mai. 2012.

GRYFO. ***Marcação de Ponto por Registro Eletrônico com Reconhecimento Facial***. 2021. Disponível em: ⟨https://gryfo.com.br/blog/2021/12/01/marcacao-de-ponto-com-reconhecimento-facial/⟩. Acesso em: 28 abr. 2022.

GUISSOUS, A. E. ***Skin Lesion Classification Using Deep Neural Network***. Argélia: Research Gate, 2019.

HEINRICH, C. J. . P. Z. . K. ***Machine learning and deep learning***. Dortmund: Electronic Markets, 2021.

HSU, C.-W.; CHANG, C.-C.; LIN, C.-J. ***A Practical Guide to Support Vector Classification***. Índia: [s.n.], 2016.

JOHNSON, R. E. ***Frameworks = (Components + Patterns)***. Urbana: Communications Of The ACM, 1997.

LUCIANO, J.; ALVES, W. J. B. ***Padrão de Arquitetura MVC: Model-View-Controller***. Bebedouro: Universidade de São Paulo, 2011.

MAKAI, M. ***Peewee***. 2018. Disponível em: ⟨https://www.fullstackpython.com/peewee.html⟩. Acesso em: 15 mai. 2022.

MAKAI, M. ***Sanic***. 2018. Disponível em: ⟨https://www.fullstackpython.com/sanic.html⟩. Acesso em: 07 mai. 2022.

MARRY. ***Como definir relações entre tabelas em um banco de dados do Access***. 2022.

OLIVEIRA, J. P. de. ***Sistema de detecção e identificação de placas de trânsito brasileiras utilizando técnicas de processamento digital de imagens e RCNN***. [S.l.]: Conjecturas, 2022.

OLIVEIRA, L. F. ***Entendendo a Tríade Model-View-Controller (MVC) Utilizando Padrões de Projeto Orientado a Objetos***. Campinas: Conic-Semesp, 2013.

PODAREANU, D. ***Best Practice Guide - Deep Learning***. Alemanha: Research Gate, 2019.

RAZORTHINK, A. ***Tutorial PostgreSQL: introdução prática ao serviço***. 2020. Disponível em: ⟨https://guide.razorthink.com/use-case-examples/facial-feature-recognition.html\#introduction⟩. Acesso em: 10 abr. 2022.

READ, R. J. ***The convolution theorem and its applications***. 2009. Disponível em: ⟨https://www-structmed.cimr.cam.ac.uk/Course/Convolution/convolution.html⟩. Acesso em: 15 mai. 2022.

RESTAPITUTORIAL. ***Using HTTP Methods for RESTful Services***. 2017. Disponível em: ⟨https://www.restapitutorial.com/lessons/httpmethods.html⟩. Acesso em: 23 mai. 2022.

SHARMA, P. ***MaxPool vs AvgPool***. 2022. Disponível em: ⟨https://iq.opengenus.org/maxpool-vs-avgpool/\#:~:text=In\\%20short\\%2C\\%20in\\%20AvgPool\\%2C\\%20the,of\\%20a\\%20Convolutional\\%20Neural\\%20Network.⟩ Acesso em: 25 abr. 2022.

SRIVASTAVA, D. K.; BHAMBHU, L. ***Data Classification using Support Vector Machine***. National Taiwan University, Taipei 106, Taiwan: Journal of Theoretical and Applied Information Technology, 2010.

STUTTGART, M. ***Peewee - Um ORM Python minimalista***. 2017. Disponível em: ⟨http://pythonclub.com.br/peewee-um-orm-python-minimalista.html⟩. Acesso em: 24 mai. 2022.

SUMUS. ***Desbloquear smartphones por reconhecimento facial é seguro?*** 2019.

SUPERDATASCIENCE, T. ***Convolutional Neural Networks (CNN): Step 4 - Full Connection***. 2018. Disponível em: ⟨https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection⟩. Acesso em: 22 abr. 2022.

TELTEX. ***Reconhecimento Facial para Segurança e Gestão de Escolas e Universidades***. 2020. Disponível em: ⟨https://teltex.com.br/2017/

reconhecimento-facial-para-seguranca-e-gestao-de-escolas-e-universidades/⟩. Acesso em: 28 abr. 2022.

VENUGOPAL, V. ***Efficient Classification of Breast Lesion based on Deep Learning Technique***. Índia: Bonfring International Journal of Advances in Image Processing, 2016.

WANGENHEIM, A. V. ***Inteligência Artificial aplicada à Visão Computacional***. 2017. Disponível em: ⟨http://www.inf.ufsc.br/~aldo.vw/visao/ia.html⟩. Acesso em: 28 abr. 2022.

## APÊNDICE A - VERIFICAR TOKEN DE ACESSO

```
def app_authorization():
    def decorator(func):
        @wraps(func)
        async def authorization(request: Request, *args, **kwargs):
            token = request.token
            if token is None:
                return response.json({'token': 'token not provider'},
                    status=403)
            secret = SECRET
            algorithm = ALGORITHM
            try:
                token_decoded = jwt.decode(
                    token,
                    secret,
                    [algorithm]
                )
            except jwt.ExpiredSignatureError:
                return response.json({'token': 'expired token'},
                    status=403)
            except jwt.InvalidTokenError:
                return response.json({'token': 'invalid token'},
                    status=403)
            uid = token_decoded['institutional']
            institutional = Institutional.get_or_none(id=uid)
            if institutional is None:
                return response.json({'token': 'invalid token'},
                    status=403)
            request.headers['institutional'] = institutional.id
            return await func(request, *args, **kwargs)
        return authorization
    return decorator
```

## APÊNDICE B - GERAR E VERIFICAR CRIPTOGRAFIA DE SENHAS

```
pwd_context = CryptContext(schemes=[CRYPTOGRAPHY])

def createCriptography(password):
    return pwd_context.hash(password)

def verifyCriptography(password, passwordCriptography):
    return pwd_context.verify(password, passwordCriptography)
```

## APÊNDICE C - LOGIN DE USUÁRIO

```
async def acess(request: Request):
    email = request.json['email']
    password = request.json['password']

    institutional = Institutional.get_or_none(email=email)

    verify = verify_hash(password, institutional.password)

    if institutional is None or not verify:
        return response.json({'institutional': 'incorrect
            username or password'}, status=404)

    secret = SECRET
    algorithm = ALGORITHM
    expired = datetime.utcnow() + timedelta(days=7)

    payload = {
        'institutional': institutional.id,
        'exp': expired
    }

    token = jwt.encode(payload, secret, algorithm=algorithm)

    data = dict()
    data['token'] = token
    data['institutional'] = institutional.json

    return response.json(data, dumps=json.dumps, cls=Serialize)
```

# APÊNDICE D - MODIFICAÇÃO DE SENHA

```
async def recoverPassword(request: Request):
    email = request.json['email']

    institutional = Institutional.get_or_none(email=email)

    if institutional is None:
        return response.json({'institutional': 'email not
            found'}, status=404)

    secret = SECRET
    algorithm = ALGORITHM
    expired = datetime.utcnow() + timedelta(minutes=15)

    payload = {
        'institutional': institutional.id,
        'exp': expired
    }

    token = jwt.encode(payload, secret, algorithm=algorithm)

    data = dict()
    data['authorization'] = sendEmailUpdatePassword(email,
        institutional.fullName, URL_RECOVERY_PASSWORD + token)

    return response.json(data, dumps=json.dumps, cls=Serialize)
```

# APÊNDICE E - CONFIRMAÇÃO DE PRESENÇA

```
def sendEmail(emailStudant, nameStudant, presence, participation,
    totalClasses):

    emailAddress = EMAIL
    emailPassword = PASSWORD

    message = EmailMessage()

    frequency = (participation / totalClasses) * 100

    message['Subject'] = 'Attendance Confirmation'
    message['From'] = emailStudant
    message['To'] = f'Hello, {nameStudant}!!!' \
                    f'Your presence in class {presence}
                        has been confirmed.' \
                    f'You participated in {participation} classes,
                         of a total of {totalClasses}.' \
                    f'His current presence is {"%.2f" %
                        round(frequency, 2)}%'

    message.set_content()

    with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
        smtp.login(emailAddress, emailPassword)
        smtp.send_message(message)

    return 'Send Email'
```

## APÊNDICE F - PACOTES INSTALADOS NO BACKEND

```
aiofiles==0.8.0
cffi==1.15.0
cryptography==37.0.2
httptools==0.4.0
jose==1.0.0
jwt==1.3.1
multidict==6.0.2
passlib==1.7.4
peewee==3.1.0
peewee-validates==1.0.7
psycopg2-binary==2.9.3
pycparser==2.21
PyJWT==2.3.0
python-dateutil==2.8.2
sanic==22.3.1
sanic-routing==22.3.0
six==1.16.0
ujson==5.2.0
uvloop==0.16.0
websockets==10.3
```

## APÊNDICE G - CODIFICAR IMAGEM DE USUÁRIOS EM LISTAS

```
known_face_encodings = []
known_face_ids = []

path = 'imageUsers/'
i = 0
for _, _, image in os.walk(path):
    qty = len(image)
    while (i < qty):
        user = image[i]
        userId = user.split('.')[0]
        userImage = face_recognition.load_image_file(path + user)
        userEncoding = face_recognition.face_encodings(userImage)[0]
        known_face_encodings.append(userEncoding)
        known_face_ids.append(userId)
        i += 1
```

# APÊNDICE H - IDENTIFICAR USUÁRIOS

```
frame = cv2.imread(imageUser)

small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
rgb_small_frame = small_frame[:, :, ::-1]

face_locations = face_recognition.face_locations(rgb_small_frame)
face_encodings = face_recognition.face_encodings(rgb_small_frame,
    face_locations)

face_names = []
for face_encoding in face_encodings:
    matches = face_recognition.compare_faces(known_face_encodings,
        face_encoding)
    name = 'Unknow'
    face_distances = face_recognition.face_distance(
        known_face_encodings, face_encoding)
    best_match_index = np.argmin(face_distances)
    if matches[best_match_index]:
        name = known_face_ids[best_match_index]

    face_names.append(name)

for (top, right, bottom, left), name in zip(face_locations,
    face_names):
    top *= 4
    right *= 4
    bottom *= 4
    left *= 4

    cv2.rectangle(frame, (left, top), (right, bottom),
        (0, 0, 255), 2)

    cv2.rectangle(frame, (left, bottom - 35), (right, bottom),
        (0, 0, 255), cv2.FILLED)
    font = cv2.FONT_HERSHEY_DUPLEX
    cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0,
        (255, 255, 255), 1)
```