

UNIVERSIDADE PAULISTA

ALEXANDRE SANTOS CAVALCANTE (N384425)

AUGUSTO CALISTO DE AQUINO (D898HI8)

HELOISA FERREIRA DA SILVA (F037CJ1)

LUIZ GUSTAVO DE OLIVEIRA DINIZ (N537BD3)

OSCAR LUIZ R DE OLIVEIRA (D85HEF7)

RAFAELA DOS SANTOS SILVA (D9219D0)

**DESENVOLVIMENTO DE UM SISTEMA DE IDENTIFICAÇÃO E AUTENTICAÇÃO
BIOMÉTRICA**

São Paulo

2021

LISTA DE FIGURAS

Figura 1 – Reconhecimento facial	09
Figura 2 – .Net Core	16
Figura 3 – Diagrama da aplicação	20
Figura 4 – Diagrama do banco de dados	21
Figura 5 – Tela de cadastro	27
Figura 6 – Tela de cadastro da face	27
Figura 7 – Tela de login	28
Figura 8 – Reconhecimento facial	28
Figura 9 – Informações que o ministro acessa	29
Figura 10 – Tela de login	29
Figura 11 – Tela de reconhecimento com erro pois não reconheceu a face do usuário.....	29

SUMÁRIO

1.	OBJETIVO GERAL	4
1.1.	Objetivo Específico	4
2.	INTRODUÇÃO	5
3.	CONCEITOS GERAIS	7
3.1	Teoria Biometria.....	7
3.2	Funcionamento da Biometria Facial	8
3.3	Estudo de Caso	14
4.	PLANO DE DESENVOLVIMENTO	16
5.	PROJETO DO PROGRAMA.....	20
6.	CONSIDERAÇÕES FINAIS.....	30
7.	REFERÊNCIAS.....	31
8.	APÊNDICE	32

1. OBJETIVO GERAL

Entender a função e aplicação de Técnicas Biométricas, com isso desenvolver um sistema de identificação e autenticação simples na linguagem C#. Contribuir com explicações sobre o processo da biometria facial.

1.1. Objetivo Específico

- Analisar o efeito desse trabalho e discutir a interdisciplinaridade e como a ferramenta pode auxiliar neste contexto;
- Compreender as principais técnicas biométricas e seu fundamento;
- Desenvolver uma ferramenta de identificação e autenticação biométrica e
- Avaliar as técnicas utilizadas e discorrer sobre seu funcionamento perante a situação problema.

2. INTRODUÇÃO

O reconhecimento facial assim como outros campos da computação como a criptografia, inteligência artificial, computação gráfica levou muito tempo para se consolidar no mercado, as pesquisas de reconhecimento de rosto têm sido conduzidas desde a década de 60.

Em 1964 e 1965, Bledsoe, junto com Wolf e Bisson, começaram a trabalhar usando computadores para reconhecer o rosto humano, o projeto inicialmente um humano tinha que localizar as coordenadas das características faciais, como o centro da pupila, o canto interno e externo dos olhos. Na década de 1970 Goldstein, Harmon e Lesk foram capazes de adicionar maior precisão a um sistema manual de reconhecimento facial. Eles usaram cerca de 21 marcadores subjetivos específicos, incluindo espessura dos lábios e cor do cabelo, a fim de identificar rostos automaticamente.

Em 1988, Sirovich e Kirby começaram a aplicar álgebra linear ao problema de reconhecimento facial sendo capaz de mostrar que menos de cem valores eram necessários para codificar com precisão uma imagem de rosto normalizada.

O reconhecimento facial é uma forma de reconhecer um rosto humano por meio da tecnologia. Um sistema de reconhecimento facial usa biometria para mapear características faciais de uma fotografia ou vídeo. Ele compara as informações com um banco de dados de rostos conhecidos para encontrar uma correspondência.

Existem diversas técnicas para abordar o tema de reconhecimento, como por exemplo, o Principal Component Analysis (PCA) que é um procedimento matemático que realiza uma redução da dimensionalidade, extraíndo o componente principal dos dados multidimensionais. O PCA é uma abordagem estatística usada para reduzir o número de variáveis no reconhecimento facial.

No PCA, cada imagem no conjunto de treinamento é representada como uma combinação linear de autovetores ponderados chamados autofaces. Esses autovetores são obtidos da matriz de covariância de um conjunto de imagens de treinamento.

Tendo em vista a necessidade do Ministério do Meio Ambiente de restringir seus artigos, foram pensadas e analisadas propostas para desenvolver uma aplicação com reconhecimento facial. No decorrer do presente estudo serão apresentados os mecanismos utilizados para prover o bom funcionamento e eficácia da solução proposta.

3. CONCEITOS GERAIS

3.1 Teoria Biometria

Do grego bios (que significa “vida”) e metron (que significa “medida”), Biometria indica as características físicas, biológicas e únicas dos seres humanos que nós estudantes de Ciências da Computação utilizamos como mecanismos de identificação e autenticação.

A biometria faz parte de um dos três pilares da transformação paperless, a identificação digital. Um sistema biométrico pode captar diversos traços humanos para realizar uma autenticação, como por exemplo a impressão digital, a face, íris, geometria da mão, retina e voz. Sendo essas são apenas algumas opções mais conhecidas.

Podemos agrupar os tipos de biometrias em três grupos: Biometria Biológica, Biometria Morfológica e Biometria Comportamental.

- Biometria Biológica: trata informações de nível genético e molecular, como por exemplo DNA, sangue, fluídos do corpo dentre outros.
- Biometria Morfológica: remete ao corpo humano, parte física como o olho, digital e mapeamento facial através de scanners.
- Biometria Comportamental: baseada em atitudes únicas de cada pessoa, por exemplo, como você fala, anda, escreve ou padrões de digitação.

Atualmente, essas métricas são utilizadas com o intuito de reforçar os sistemas de senha, telefones, assinaturas eletrônicas, salas e edifícios com acesso restrito. Um dos benefícios da biometria é que seu corpo se torna sua “chave”, logo é impossível perder ou esquecê-la e a mesma não pode ser roubada de você.

Com o avanço tecnológico, os leitores biométricos tornam-se mais sofisticados e utiliza uma machine learning treinada para seguir 4 etapas – descritas abaixo.

1. Captura: Coleta, por meio de um leitor biométrico, dos dados que alimentarão o banco de dados usado para comparação com a biometria escaneada.
2. Extração: Tradução dos dados biométricos em informação identificável. Cada sistema tem seu próprio método de tradução, com diferentes níveis de confiabilidade e de rigor analítico.

3. Padrão: Estabelecimento de um formato único para cada cadastro das características reconhecíveis pelo sistema biométrico, a fim de reduzir o tempo de análise de todo o processo. A qualidade dos dados utilizados no sistema vai evitar falhas de identificação.
4. Comparação: Ligação do dado biométrico escaneado ao registro da base de dados, identificando de maneira única uma pessoa.

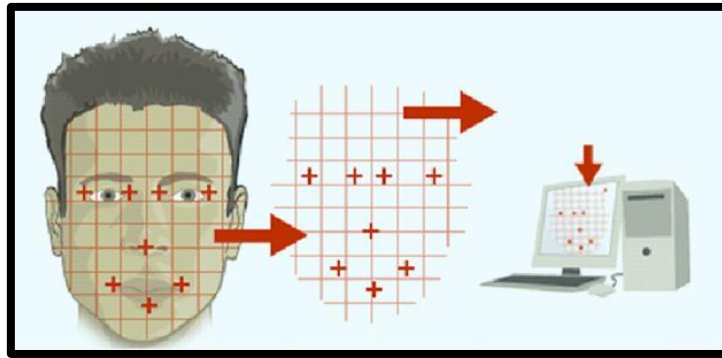
Diante desse processo, notamos que a tecnologia biométrica utiliza um sistema difícil de burlar. Em sistemas de segurança dos celulares um ótimo exemplo da utilização da biometria é a tecnologia de reconhecimento facial do iPhone X (e todas as versões após esta) da Apple, que projeta 30.000 pontos infravermelhos no rosto do usuário para autenticá-lo por correspondência de padrões. Pode ser que ainda haja questionamentos sobre a segurança dos métodos supracitados, porém de acordo com a Apple a chance de a identidade ser errada é de uma em um milhão.

O deepfake é uma ferramenta que já consegue incorporar o rosto de uma pessoa em situações ficcionais, sendo então um diferencial na incrementação tecnológica da biometria para evitar distorções. Para o futuro, ansiamos que o desenvolvimento desta tecnologia seja mais explorado a fim de proteger a individualidade e garantir a legitimidade dos dados.

3.2 Funcionamento da Biometria Facial

Para que a biometria facial seja possível, o primeiro componente obrigatório no sistema é ter algum meio em que seja possível capturar a imagem do usuário que está solicitando uma autenticação. Desta forma, a imagem pode vir da câmera do dispositivo que está acessando a aplicação. Nos dias atuais, a câmera está amplamente disponível, tornando a autenticação via métricas faciais ainda mais viável. Os sistemas que se utiliza reconhecimento facial, tem um ponto em comum, detectam o rosto do usuário, transformam e formas geométricas e realizam algoritmos para captar pontos em comum da face humana, criando-se atributos como a distância de cada característica captada, a área e formato do rosto.

Figura 1 – Reconhecimento Facial



Fonte: <https://www.biometricos.net/2014/06/algoritmo-de-reconocimiento-facial.html>

Apos a análise destas características, os dados são traduzidos para o sistema e salvo em um banco de dados, para ficar atrelado aos usuários proprietários destas características.

3.2.1. Pontos Nodais

A biometria facial para se tornar viável, precisa de diversas características para reconhecer no rosto do usuário, estas características serão o que irá diferenciar e identificar como único o indivíduo. Esse conjunto de características é chamado de pontos nodais. Existem aproximadamente 80 pontos nodais na face humana. Alguns pontos nodais que podemos exemplificar são, a distância entre os olhos, o comprimento do nariz, tamanho do queixo, linha da mandíbula e até as maçãs do rosto. Todas essas características são medidas e guardadas no banco de dados, gerando assim a assinatura facial. Os pontos nodais do rosto só podem ser capturados quando a imagem é rastreada sabendo a posição do rosto corretamente. No próximo tópico abordaremos como isto ocorre.

3.2.2. Autenticação, detecção e reconhecimento

Dentro da biometria facial nos deparamos com três tipos de situações envolvendo essa métrica biológica, é importante distinguir para ser corretamente implementado de acordo com o que cada situação do sistema pede.

A detecção facial é a primeira etapa para o fluxo da biometria facial no sistema a ser implementado, esta etapa antecede o reconhecimento facial e a autenticação. Neste momento é determinado se tem a presença de um rosto no vídeo ou em uma imagem que é recebida. Os algoritmos presentes na detecção facial utilizam de machine learning, ou seja, o algoritmo tem um aprendizado que a partir dos dados recebidos, gera um certo entendimento a respeito dos parâmetros definidos para sua

tarefa, no caso, o algoritmo de detecção facial irá “aprender” a reconhecer padrões relacionados a formatos de rosto. A precisão desta etapa depende do quanto o algoritmo de detecção está treinado. Atributos como, orientação da face, escala e localização do rosto é tratado nesta etapa. Com os pontos em comum de um rosto localizados (localização dos olhos, sobrancelhas, nariz etc.), pode-se realizar algumas etapas de pré-processamentos na imagem ou vídeo.

O reconhecimento facial tem como propósito determinar a quem pertence o rosto identificado. Seus procedimentos se resumem em análises comparativas com dados armazenados, previamente cadastrados no banco de dados da aplicação. A afirmação se o rosto analisado pertence ou não a um dado encontrado está atrelado a uma probabilidade que tem seu resultado a partir dos passos:

- A aplicação realiza a captura do rosto do usuário;
- Faz a análise da imagem. Neste momento as características únicas da pessoa, que foi armazenada no banco de dados, são comparadas com os pontos nodais do indivíduo;
- Feita a análise, a assinatura(características) única do usuário é adquirida e comparada com outras assinaturas armazenadas;
- Feito a detecção, extração e comparação, a aplicação determina o grau de compatibilidade entre as imagens analisadas. A partir deste grau de compatibilidade, é determinado o quanto o usuário é ou não tendo em vista as imagens e assinaturas comparadas.

A autenticação facial fica responsável em gerar uma credencial de acesso do usuário no sistema. Após o processo de verificar a identidade do usuário na etapa anterior, caso esteja aceita na compatibilidade, o usuário recebe permissão de acesso na aplicação, respeitando as restrições de acesso de acordo com seu perfil.

3.2.3. Técnicas para reconhecimento facial

- Reconhecimento de faces através de quadros de vídeo

Geralmente o reconhecimento de face pode estar em um contexto em que a câmera do dispositivo transmite as imagens em formato de vídeo, que nada mais é do que imagens sequenciais, essas imagens se denominam frames (Quadros) do vídeo. Através de cada quadro de imagem gerada pelo vídeo, é possível fazer uma análise

de movimento, baseado na subtração de quadros, resultando na localização, rastreamento e captura de imagens do rosto.

Neste sistema de análise de movimentos é que se encontram especificamente as técnicas utilizadas para trabalhar sobre a imagem, desde sua formulação matemática até possíveis processamentos de imagem para melhor definir os pontos nodais do rosto, podendo processar um contraste, brilho, saturação ou outra propriedade sobre a imagem dada.

- Principal Component Analysis (PCA)

Uma das técnicas que podem ser utilizadas na análise e implementar módulos de redução das informações é através do procedimento matemático chamado de Principal Component Analysis (PCA), ou em português, Análise de Componentes Principais. Este procedimento tem como objetivo condensar as informações vindas de várias variáveis em um conjunto de variáveis estatísticas com o mínimo de perda de informação, compactando os principais componentes dos espaços das características. Além de utilizada em reconhecimento facial o PCA pode ser utilizado também em compressão de imagem e análise de expressão gênica.

Uma imagem facial capturada de um frame vindo da câmera do dispositivo, a nível de processamento computacional, ela está representada em uma dimensão 2D. Através do PCA, ela pode ser representada em um vetor de 1D (uma dimensão) pelo concatenamento de cada linha ou coluna em um vetor único. Demonstrando matematicamente, suponha que temos M vetores de tamanho N = linhas de imagem x colunas de imagem, que representa amostras de imagens. p_j representa os valores dos pixels.

$$x_i = [p_1 \dots p_N]^T, i = 1, \dots, M \quad (1)$$

As imagens são medianamente centradas pela subtração de imagem média de cada imagem do vetor. A variável m representa a imagem média.

$$m = \frac{1}{M} \sum_{i=1}^M x_i \quad (2)$$

A variável w_i será definida como a imagem média centrada.

$$w_i = x_i - m \quad (3)$$

O objetivo é encontrar um conjunto de e_i que tenha a maior projeção possível em cada w_i . Deseja-se encontrar um conjunto de vetores M ortonormais (Ortonormalidade, da álgebra linear, são dois vetores em um Espaço vetorial de Produto interno, caso sejam vetores Ortogonais e unitários) e_i para cada quantidade.

$$\lambda_i = \frac{1}{M} \sum_{n=1}^M (e_i^T w_n)^2 \quad (4)$$

E então é maximizado com a restrição de ortonormalidade.

$$e_l^T e_k = \delta_{lk} \quad (5)$$

Além desses cálculos iniciais, é calculado também valores de autovetores (eigenvectors) e autovalores (eigenvalues) da matriz de covariância gerado pelos cálculos acima, os autovetores é um vetor que quando operado por um dado operador dá um múltiplo escalar de si mesmo. O valor gerado pelos autovetores e autovalores pode ser enorme. Por exemplo, uma imagem de tamanho 64x64, cria uma matriz de covariância de 4096x4096. Após passar por um teorema de álgebra linear e normalizado, os valores gerados na matriz de covariância produzem uma base ortonormal, na qual a maioria dos dados da imagem podem ser representada com a menor quantidade de erro.

Para concluir, uma imagem facial pode ser projetada na dimensão de M' , computando $\Omega = [v_1 v_2 \dots v_{M'}]^T$ onde $v_i = e_i^T w_i$. A variável v_i é a i -ésima coordenada da imagem facial no novo espaço, que vem a representar o componente principal.

O método mais simples para determinar qual classe de face fornece a melhor descrição de uma imagem facial de entrada é encontrar a classe de rosto k que minimiza a distância euclidiana $\epsilon_k = ||(\Omega - \Omega_k)||$ onde Ω_k é o vetor descrevendo a k -ésima classe de face. Se ϵ_k for menor que algum limite predefinido θ_ϵ , uma face é classificada como pertencente à classe k .

- Extração, Classificação e identificação de face

Tendo em mãos as imagens necessárias dos rostos dos usuários, processado por meio do PCA, é necessário a extração dessas características, localizando e identificando-as. Existem diversos algoritmos capazes de identificação dos rostos, e muitos combinam diversos algoritmos para melhorar sua precisão, é possível dividir em quatro categorias de métodos:

1) Método template matching: para tornar o reconhecimento da face possível, é utilizado diversos modelos padrões que descrevem um face. O reconhecimento se dá pela comparação entre os modelos e as imagens. Utiliza-se para localização e reconhecimento.

2) Abordagens de características invariantes: Por meio de características que não sofrem influência de variações e posições, tenta-se encontrar e utilizar essas informações estruturais. Utilizado principalmente para localização.

3) Métodos baseados no conhecimento: Utilizado principalmente para localização. É baseado no conhecimento humano para distinguir o que define um rosto.

4) Métodos baseados na aparência: Os modelos são definidos a partir de um treinamento, ou aprendizado de máquina. O algoritmo desse método aprende a identificar a face. É utilizado em sua maioria para reconhecimento.

Sendo assim, temos que ter em mente que o reconhecimento identifica as características relacionadas a face e a localização tem como objetivo definir o posicionamento da face na imagem.

Uma metodologia de classificação, para identificar corretamente cada conjunto de informações, e que se torna cada vez mais refinada, é o aprendizado de máquina. Uma das técnicas que podemos mencionar é a Máquina de Vetores Suporte (Support Vector Machine – SVMs), é embasado no aprendizado estatístico, tem boa capacidade de generalização e robustas com dados de grande dimensão. A SVM requer um treinamento para que seja possível generalizar os resultados de classificação.

- Eigenface

O método Eigenface tem como objetivo gerar uma pequena base de vetores que possibilitem a representação das imagens que estão sendo analisadas. Essas bases são geradas a partir do treinamento com uma amostra de imagens de faces, ou seja, baseia-se em linearmente projetar o espaço de imagens em um espaço de características com dimensões reduzidas obtido fazendo uso da análise de componentes principais (PCA), também conhecido como método Karhunen-Loeve.

A análise da imagem nesse algoritmo possibilita indexar as imagens, permitindo o reconhecimento das faces. Entretanto, produz direções de projeção que maximiza a dispersão dos pontos no gráfico em todas as classes, isto é, em todas as imagens faciais mantêm as variações indesejadas causadas pela iluminação e expressão facial [BELHUMEUR 1997].

- Fisherface (Fisher's Linear Discriminant)

Também conhecido como Análise de Discriminantes Linear (LDA) e desenvolvido por R. A. Fisher é um método utilizado para reconhecimento de objetos. O método tenta modelar a dispersão dos pontos visando maior confiabilidade para a classificação, desta forma busca otimizar a melhor linha em uma superfície que separa satisfatoriamente as classes [BELHUMEUR 1997]. Possui uma melhor projeção que a PCA, uma vez que ela é feita maximizando a dispersão interclasse e minimizando a intraclasse, formulado pela razão entre as determinantes de ambas as matrizes encontrando um maior número de direções de projeções discriminantes.

- KDDA (Kernel Direct Discriminat Analysis)

Proposta por Juwei Lu, a análise discriminante de Kernel tem como finalidade melhorar a atuação e superar limitações, utilizando-se de funções núcleo com o intuito de modificar o espaço dimensional, aumentando-o e obtendo uma maneira de dispor os dados linearmente separáveis. É uma versão “kernelizada” da análise discriminante linear, onde a ideia é encontrar uma projeção em que a separação das classes seja maximizada.

3.3 Estudo de Caso

Dado a situação em que temos um sistema com restrições de acessos devido a sigilos e confidencialidade de informações, temos que pensar em um modo eficaz que aumente a segurança e garanta este requisito do sistema. A maioria dos sistemas utilizam uma autenticação baseada em usuário e senha na qual o usuário sabe essas informações, pois foi ele quem as cadastrou.

Um reconhecimento facial teria muito a agregar ao sistema em complemento com usuário e senha, pois além de requisitar as credenciais a quem esteja querendo acessar, o reconhecimento facial pode reforçar que o login feito, esteja realmente

sendo realizado pela pessoa proprietária de tais credenciais, já que os atributos biométricos faciais são atrelados ao usuário e senha informados.

No estudo de caso dado, a autenticação ao sistema será feita por tais atributos informados no parágrafo anterior. Por meio da autenticação biométrica fornecida junto ao usuário e senha, o sistema irá validar os acessos e restrições do usuário autenticado. Essas restrições vão ser tratadas devido ao seu nível de acesso concedido em seu cadastro. O usuário com acesso absoluto às informações será o ministro do meio ambiente, logo terá nível 3, este acesso será concedido diretamente no sistema, no banco de dados. Os diretores de divisões terão nível 2 de acesso a informações, o acesso concedido a eles, poderá vir tanto do ministro do meio ambiente ou do sistema, no caso do sistema é responsabilidade dos administradores do sistema. Os usuários de nível 1 de acesso representam qualquer cargo e estejam trabalhando dentro do ministério. As informações do nível, está liberada também aos níveis 2 e 3.

Não importando o nível de acesso que tem o usuário, a sua autenticação biométrica será exclusivamente feita pelo usuário proprietário das credenciais. Supondo dois usuários, A e B, caso o usuário B queira acessar o sistema com usuário e senha do usuário A, mesmo que o usuário B saiba das informações, ele não será autenticado, pois o reconhecimento facial irá bloquear por ter biométricas faciais divergentes.

Tendo descrito nos parágrafos anteriores a ideia da biometria facial aplicada na autenticação do sistema, a segurança das informações estratégicas sobre propriedades rurais que utilizam agrotóxicos restritos, terá sua confidencialidade bem assegurada justamente pela métrica biológica de características faciais do usuário cadastrado.

4. PLANO DE DESENVOLVIMENTO

As tecnologias e ferramentas de desenvolvimento utilizadas neste trabalho foram:

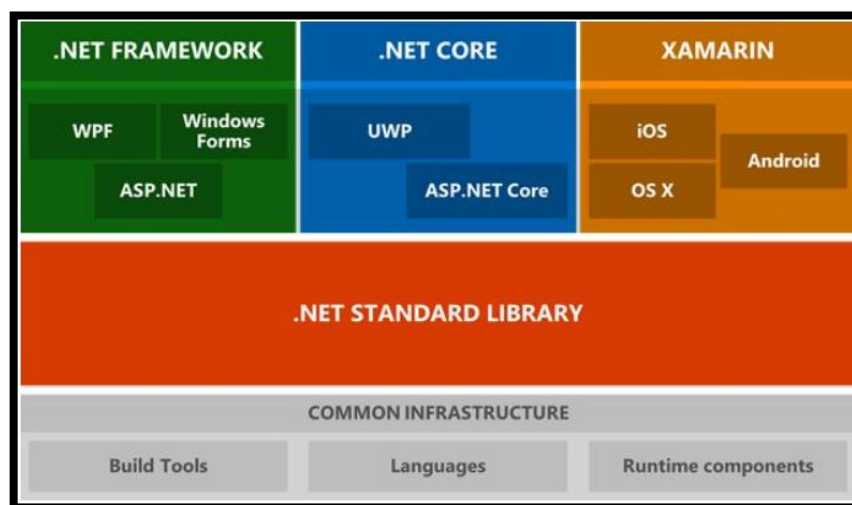
- .NET Core 5
- Microsoft SQL Server
- Visual Studio 2019
- Vue JS
- Bootstrap 4
- Python

4.1. .NET Core 5

O .NET Core é uma plataforma de desenvolvimento da Microsoft lançado em 2016 sendo feito totalmente do zero seguindo os padrões de desenvolvimento, que tem como intuito a criação de aplicações desktop e web.

As vezes as pessoas se confundem entre .NET framework e .NET Core, mas, ambas suportam as linguagens de programação C#, F# e Visual Basic, que podem ser utilizadas para fazer aplicações desktop e web, mas, a diferença é que o .NET framework é suportado apenas no Windows, enquanto, o .NET Core por ser mais novo, ele é multiplataforma, ou seja, roda em diversos sistemas operacionais como Windows, Linux, Mac OS).

Figura 2 - .Net Core



Fonte: <http://arun-ts.blogspot.com/2019/06/net-ecosystem-net-core-net-framework.html>

Framework é um conjunto de códigos genéricos que tem como objetivo entregar diversas funcionalidades, ou seja, ajudar o desenvolvedor na construção de softwares

4.2. SQL Server

O SQL Server é um sistema gerenciador de banco de dados relacional (SGBDR), onde no modelo relacional, existem tabelas que possuem atributos/campos e obrigatoriamente deve ter uma Primary Key (PK) que identifica de forma única cada registro da tabela e para que haja um relacionamento entre essas tabelas, o campo identificador único de uma tabela deverá estar contido na outra tabela, assim sendo conhecida como Foreign Key (FK).

Ele se baseia na linguagem Transact-SQL (T-SQL) que é uma extensão do SQL que acrescenta algumas características como:

- Controle do fluxo da língua;
- Variáveis Locais;

```
DECLARE @estado CHAR(2) SET @estado = 'SP'
```

```
SELECT @estado
```

- Várias funções de suporte ao processamento de string, data, matemática;
- Melhoramento para a declaração DELETE e UPDATE;

4.3. Visual Studio 2019

O Visual Studio é uma IDE (Integrated Development Environment ou Ambiente de Desenvolvimento Integrado) da Microsoft para o desenvolvimento de softwares com .NET Framework, Visual Basic, C, C++, C# e F#. Assim como toda IDE, ele cria a estrutura do projeto conforme a necessidade do desenvolvedor, integração com o Git, preenchimento automático dos comandos para agilizar a construção do seu código, conta com assistentes que permite integrar seus códigos a bancos de dados sem a necessidade de escrever códigos, além de ter diversas bibliotecas para o desenvolvimento de aplicações desktop, web, mobile e games.

4.4. Vue JS

Vue JS é um framework javascript para construção de interfaces de usuário, com o foco de ser uma Single Page Application (SPA) ou aplicação de página única.

Em uma single page application, todo o projeto é focado em componentes, que são instâncias reutilizáveis do núcleo do Vue. Cada componente tem a seguinte estrutura:

- <template>
- <script>
- <style>

O template consiste em ter todo o código HTML do componente, assim no arquivo index.html será substituído por todo o conteúdo do <template>. Caso ocorra de ter duas ou mais tags devemos sempre colocá-las dentro de uma tag container, caso não coloque não irá funcionar.

O script irá conter todo o código Javascript, além das opções como data, computed, watch, methods e os lifecycle hooks do Vue.

Na função data, iremos criar as propriedades para que possamos dar dinamicidade, usamos o data para rastrear alterações em uma propriedade particular que gostaríamos que fosse reativa.

O computed nos permite definir uma propriedade que será usada da mesma maneira que no data, porém, irá ter uma lógica personalizada que é armazenada em cache com base em suas dependências.

O watch é uma função que o Vue executa automaticamente quando a propriedade observada é alterada.

No methods iremos criar as funções do componente. Eles são incrivelmente úteis para conectar funcionalidade a diretivas de eventos, ou mesmo apenas criar um pequeno pedaço de lógica para ser reutilizado como qualquer outra função.

No style, irá toda a estilização do componente feita através do CSS.

4.5. Bootstrap 4

Bootstrap é um framework front-end para construção de páginas e aplicações responsivas utilizando arquivos HTML, CSS e JavaScript,, o principal motivo dos desenvolvedores gostarem dele é o fato de você pode criar protótipos rapidamente, facilitando de ter que escrever novamente os mesmos templates do dia a dia.

Existe três formas de utilizar o bootstrap:

- 1) Baixando os arquivos e injetando no seu projeto;
- 2) Instalando através do npm que é o gerenciador de pacotes para o Node.JS;
- 3) Adicionando os arquivos Css e Js na sua página HTML entre a tag `<head></head>`;

4.6. Python

Face Recognition é uma biblioteca em python construída em dlib para reconhecimento de imagens. A biblioteca oferece recursos de reconhecimento facial, comparação de faces e permite localizar faces em tempo real. É possível utilizar suas funcionalidades tanto em scripts python quanto em linha de comando. O dlib é uma biblioteca construída em C++ que possui diversos algoritmos e ferramentas de deep learning.

Para utilizar a biblioteca é necessário instalar os seguintes pacotes:

- Python
- Compilador de C++
- CMake
- Dlib
- Face Recognition

Para fazer a utilização da API basta importar o módulo do Face Recognition e utilizar os métodos necessários para capturar as imagens do disco e fazer as comparações. Exemplo da utilização da biblioteca na nossa aplicação:

Arquivo: `execute_face_recognition.py`

```
# Importação dos módulos
import face_recognition
import os

# Load da imagem vinda da API de autenticação que a aplicação C# salva no
diretório de execução
picture_of_me =
face_recognition.load_image_file(os.path.abspath("C:/Users/Luiz/source/repos/F
acialBiometrics/FacialBiometricsBack/bin/Debug/net5.0/TempUploads/Received/rec
eived_image.png"))
my_face_encoding = face_recognition.face_encodings(picture_of_me)[0]
```

```
# Load da imagem salva em banco de dados no momento do cadastro do usuário que
a aplicação C# salva no diretório de execução
unknown_picture =
face_recognition.load_image_file(os.path.abspath("C:/Users/Luiz/source/repos/F
acialBiometrics/FacialBiometricsBack/bin/Debug/net5.0/TempUploads/Database/dat
abase_image.png"))
unknown_face_encoding = face_recognition.face_encodings(unknown_picture)[0]

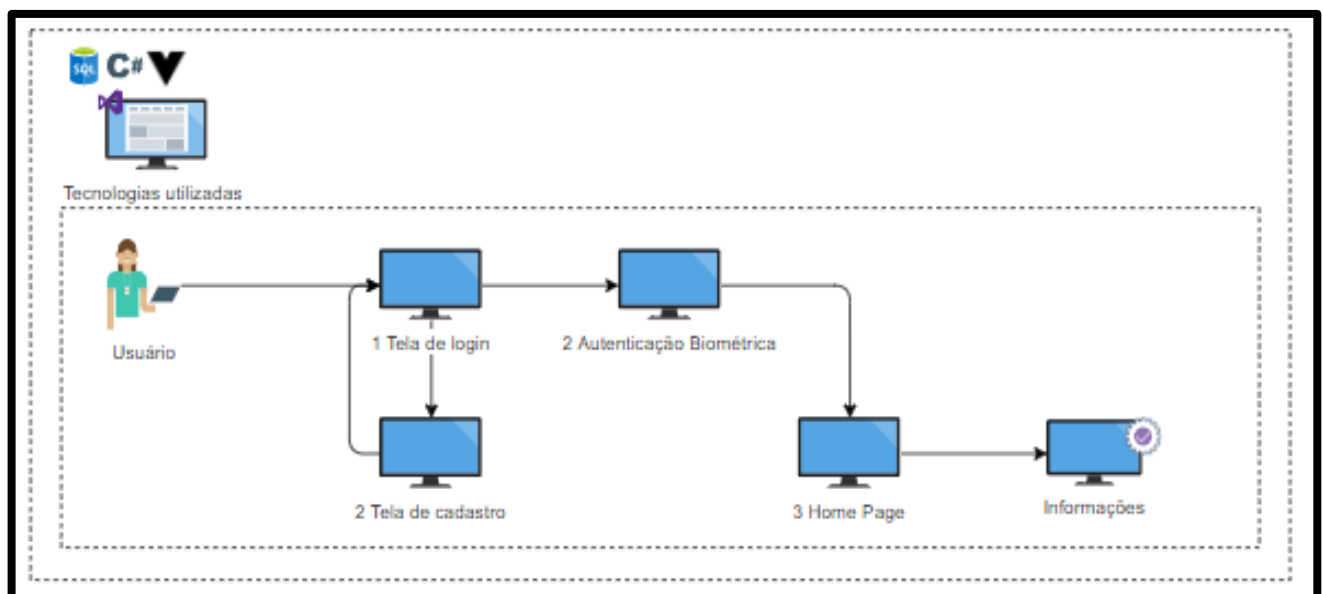
# Comparação das duas imagens
results = face_recognition.compare_faces([my_face_encoding],
unknown_face_encoding)

# Resultado da comparação
if results[0] == True:
    print("true")
else:
    print("false")
```

5. PROJETO DO PROGRAMA

O fluxograma a seguir descreve como a aplicação funciona desde o momento em que o usuário entra na tela de login até a autenticação dele e visualização das informações.

Figura 3 – Diagrama da aplicação.



Fonte: Própria, 2021

No momento em que o usuário entra na aplicação, a primeira página que ele visualiza é a tela de login. Caso o usuário não tenha um login, ele pode entrar na tela de cadastro, preencher o formulário de cadastro com suas informações – é nesse momento em que o usuário registra a sua biometria facial – e após a finalização ele é redirecionado para a tela de login.

No momento em que o usuário realiza a autenticação na tela de login via usuário e senha, será necessário passar por mais uma verificação, a autenticação biométrica. Caso o a autenticação ocorra com sucesso, o usuário será redirecionado para a página principal da aplicação, contendo as informações que o usuário pode visualizar, de acordo com o seu cargo no ministério.

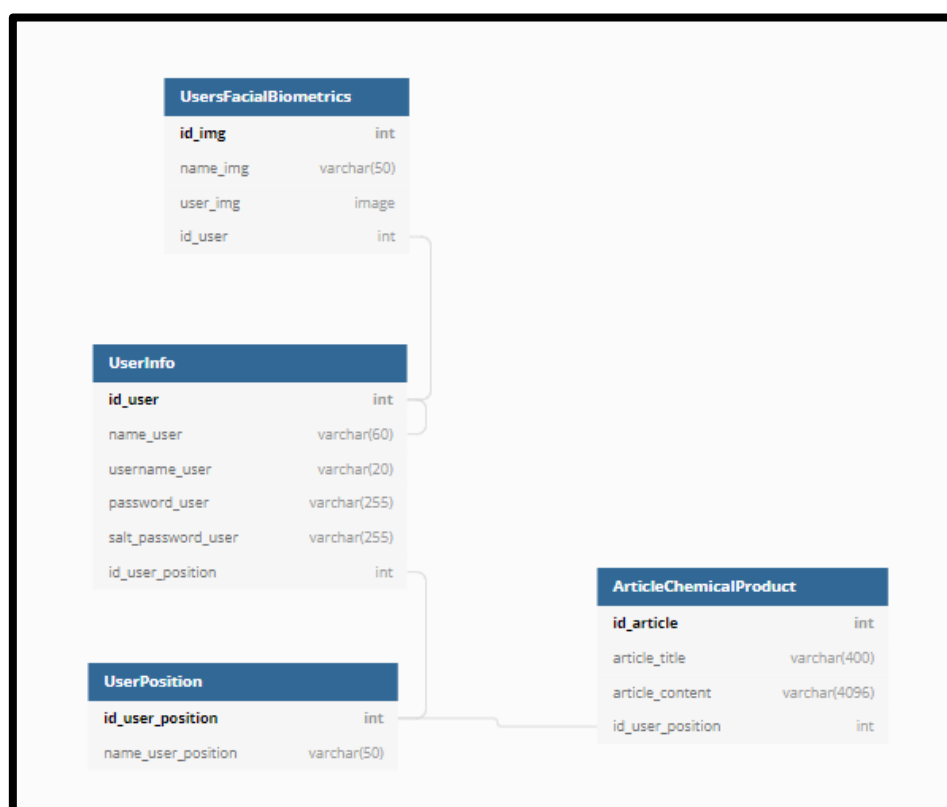
As tecnologias usadas na aplicação são:

- Backend: C#, .Net 5 e SQL Server
- Frontend: Vue

5.1. Banco de dados

O diagrama a seguir descreve a estrutura do banco de dados utilizados na aplicação.

Figura 4 – Diagrama de banco de dados.



5.1.1. Descrição das tabelas e colunas

UsersInfo – Tabela que contém as informações pertinentes ao cadastro do usuário

- id_user – Chave primária da tabela
- name_user – Nome do usuário
- username_user – Username do usuário
- password_user – Senha do usuário
- salt_password_user – Hash para manter a senha segura no banco de dados
- id_user_position – Chave estrangeira da tabela UserPosition, dessa forma o usuário tem um cargo definido

UsersFacialBiometrics – Tabela responsável por salvar as imagens do rosto do usuário

- id_img – Chave primária da tabela
- user_img – Imagem do usuário
- id_user – Chave estrangeira da tabela UsersInfo, dessa forma as imagens ficam relacionadas a um usuário específico

UserPosition – Tabela com as informações sobre os cargos do ministério

- id_user_position – Chave primária da tabela
- name_user_position – Nome do cargo

ArticleChemicalProduct – Tabela com os artigos que os usuários podem visualizar na tela principal da aplicação

- id_article – Chave primária de tabela
- article_title – Título do artigo
- article_content – Conteúdo do artigo
- id_user_position – Chave estrangeira da tabela UserPosition, dessa forma é possível definir quais cargos terão acesso a essa informação

5.2 Módulos e rotas desenvolvidos

A aplicação foi elaborada para rodar em um ambiente web, com isso separamos a interface em que o usuário acessa, denominada front-end, e temos a parte onde o processamento de imagem ocorre, que denominamos back-end.

Referente a interface, é onde obtemos a imagem, por meio da abertura de WebCan do usuário, onde será exibida a imagem da câmera no browser para o

usuário cadastrar ou autenticar o rosto. É também desenvolvido na interface, os componentes gráficos que têm interações com usuário.

No processamento das imagens, a interface envia os dados coletados por meio de APIs (Application Programming Interface), essas APIs são acessíveis por meio de URLs (Uniform Resource Locator). Por padrão as aplicações do front-end e back end trocam os dados necessários através do formato JSON (JavaScript Object Notation), que é um padrão de troca de dados. Para a passagem das imagens especificamente, é feita a serialização da imagem por meio de um método de codificação chamado Base64 que irá transformar os dados binário da imagem em texto, este texto pode ser convertido novamente para o estado binário e assim construindo a imagem novamente.

Neste detalhamento a aplicação back-end estará exemplificado com a URL principal <https://localhost:44343/>, na qual localhost com porta 44343 é a máquina local onde está sendo executado a aplicação em c# .net 5.

5.2.1 Rota para registro

Para que o front-end consiga cadastrar o usuário, é necessário que a aplicação da interface chame a API de cadastro no back-end por meio da rota <https://localhost:44343/User/register> fazendo uma requisição POST, esta API requer que um JSON que seja enviado contendo os atributos “name” para nome do usuário, “username” para o identificador do usuário, “password” que é a senha digitada pelo usuário, e “face_images” onde é atribuída uma lista de imagens tiradas pela webcam do usuário, e estas imagens estão em Base64 no formato textual e por último “id_user_position” que representa o nível de acesso que o usuário terá nas informações ao entrar na homepage da aplicação.

No código .net, esta API está presente no Controller UserController.cs programado para receber essas requisições, mais especificamente no método UserRegistration(UserFrontModel userData) recebendo um modelo para receptor os dados do JSON recebido do front-end. Neste método será convertido o base64 da imagem de texto para binário e juntamente com os dados de nome, senha, username e nível de acesso, será gravado no banco de dados, feito o cadastro corretamente a API retorna 200 indicando sucesso ou 400 caso haja algum problema e não realizando o cadastro.

5.2.2 Rota para login

Para o tratamento do login, o front-end deve chamar a API de login, que no back-end é acessada através da url `https://localhost:44343/User/login`, sendo esta, uma requisição POST. O JSON que está API recebe será com os atributos “username” que representa o identificador do usuário e “password” que será a senha cadastrada em banco.

No código .net, esta API se encontrará no controller `UserController.cs`, que receberá a requisição no método `login(LoginModel userCredentials)`, o `LoginModel` receberá o JSON fazendo sua conversão para objeto devido ao .net. Neste método será feita a chamada do procedimento `Login` da classe `FacialBiometricsServices`, responsável em ser um intermediário para acesso e manipulações no banco de dados, este método `Login`, irá chamar a classe do banco de dados `DataAccessFacialBiometrics` pelo método `GetUserByUsername`, buscando no banco o username fornecido, caso encontre, será montado o objeto para `UserInfo`. Caso encontre o usuário, o método `Login` do `FacialBiometricsServices` irá comparar a senha fornecida com a senha do banco, se tudo certo o objeto com o `UserInfo` é retornado para a `UserController` que por fim retorna sucesso caso senha certa e usuário encontrado, e retorna falha quando não encontrado ou senha incorreta.

5.2.3 Rota para validação do rosto

A rota para a validação do rosto é chamada logo após a chamada da validação de login ter sido validada. Para o tratamento da validação do rosto, o front-end irá abrir a webcam do usuário e solicitar que envie a imagem de seu rosto. A interface do usuário irá fazer uma requisição POST para `https://localhost:44343/User/validate`, o JSON enviado será com os atributos “idUser” que representa um número identificador do cadastro na base dados e “image” na qual é enviada a imagem em formato textual base64 representando seus binários.

Do lado back-end, o controller `UserController` receberá a requisição através do método `validateLogin(UserImageInfo user)`, a primeira etapa deste método é a conversão da imagem em base64 para binários, com a imagem em binário o método `CompareImages` da classe `FacialBiometricsServices` é chamado, passo o id do usuário e a imagem em binário. No método `CompareImages(int idUser, List<byte[]> receivedImages)` uma lista de imagens é carregada do bando de dados a partir do

idUser fornecido, identificando o usuário, estas imagens foram cadastradas previamente quando foi acessada a rota de cadastro.

O próximo passo do método `CompareImages()` da classe `FacialBiometricsServices` é instanciar a classe `ImageComparerService()`. Esta classe é responsável em fazer o tratamento de análise da imagem, utilizando uma API de reconhecimento facial chamada `face_recognition` desenvolvida em Python. Esta biblioteca que fornece a API para o reconhecimento facial é construída com base na “Dlib” uma ferramenta desenvolvida com c++ que fornece algoritmos de machine learning.

A classe `FacialBiometrics Services` irá chamar o método `CompareImages()` da instância de `ImageComparerService`, passando nos parâmetros as imagens resgatadas do banco e a imagem recebida para fazer autenticação.

No método `CompareImages()` será criado o caminho para uma pasta temporária que irá ficar armazenado temporariamente a imagem do banco de dados nomeada de “`database_image.png`” e a imagem recebida para a validação nomeada de “`received_image.png`”. Para que as imagens sejam escritas na pasta temporária, um método chamado `CreateTemImage()` será chamado, recebendo como parâmetros o caminho e nome da imagem a ser gravada, os bytes da imagem e o tamanho da imagem.

Antes que seja feita a próxima etapa no código .net é importante notar que para funcionar o script python, o ambiente onde a aplicação estiver hospedada, deve estar previamente configurada com Python e ter instalado a biblioteca `face_recognition` para ter acessos às suas APIs.

Feita a construção das imagens, o método `RunPythonScript()` é chamado, este é o principal método para o processamento de autenticação facial. Este método irá instanciar uma classe `Process` que permite rodar processo de sistemas na máquina local onde está a aplicação back-end. O process irá chamar o “`cmd.exe`” que é o terminal de linha de comandos do Windows, então é passado o comando para acessar o caminho onde se encontra o script python “`execute_face_recognition.py`”, por exemplo “`cd C:\\Users\\Alexandre\\Documents\\UNIP\\APS\\FacialBiometrics\\myenvpy\\Scripts`”, sequencialmente é dado o comando “`activate`”, que ativa o ambiente Python

preparado para rodar a biblioteca `face_recognition`, logo em seguida o comando `"python execute_face_recognition.py"` irá rodar o script python, onde irá pegar a imagem recebida para autenticação e irá comparar com a imagem gravada do banco que representa o usuário proprietário do login, o python irá aplicar alguns algoritmos de machine learning e outros processos, quando finalizado a comparação irá retornar `true` ou `false` na saída da linha de comando aberta.

Com base na saída `true` ou `false` na linha de comando, o código .net na classe `CompareImages` irá verificar se contém `"true"` então a imagem facial recebida é do autor do usuário logado, caso `"false"` a imagem do rosto recebida não é do proprietário do usuário logado. Por fim, quando retornado sucesso ao front-end, a tela com as informações será carregada de acordo com o nível do usuário logado. Caso haja falha o front-end irá acusar erro e não prosseguirá com a tela das informações.

5.2.4 Rota para retorno de informações dos artigos

Esta rota será responsável em carregar os artigos permitidos de acordo com o nível de acesso do usuário. Para isto, o front-end deverá fazer a requisição para a url `https://localhost:44343/User/articles`, como é uma requisição do tipo GET, não será necessário fornecer um JSON, apenas um parâmetro na URL, fornecendo o `"idUser"`. Esta rota só será acessível caso o usuário já tenha logado.

No código .net, o método que irá receber a requisição será o `getArticles(int idUser)` do `UserController.cs`, este método recebe o `idUser`, com base neste dado ele chama o método `GetArticles` do `FacialBiometricsServices` que busca na base de dados através de uma consulta SQL, a relação do nível de acesso do usuário com o nível de acesso do artigo, sendo assim, o `GetArticles` retorna uma lista com os artigos de nível permitido com base no `IdUser`, esta lista é serializada no formato JSON e retornado para o front-end trabalhar na exibição daquelas informações.

5.3. Imagens da aplicação

- 1- Cadastro do usuário, captura das informações via formulário e cadastro da biometria facial:

Figura 5 – Tela de cadastro.

SIGN UP YOUR USER ACCOUNT
Fill all form field to go to next step

[Account](#) [Camera](#)

Account Information: Step 1 - 3

Name: *
Luiz Diniz

Username: *
luizdiniz

Password: *

Position: *
Official

[Login](#) [Next →](#)

Fonte: Própria, 2021

Figura 6 – Tela de cadastro da face.

SIGN UP YOUR USER ACCOUNT
Fill all form field to go to next step

[Account](#) [Camera](#)

Images Upload: Step 2 - 3

☒ Photo 1 ☒ Photo 2 ☒ Photo 3

[Capture](#)


[Login](#) [Previous](#) [Submit](#)

Fonte: Própria, 2021

- 2- Login na aplicação. Primeiro é necessário informar o login e senha e após a validação das credenciais, a aplicação irá requisitar a validação biométrica, caso ela esteja correta, o usuário é redirecionado para a página principal, do contrário o login será invalidado.

Exemplo de caso de sucesso na validação

Figura 7 – Tela de login.

An illustration on the left side of the login screen shows a green globe with a ladder leaning against it. A person is watering the globe with a watering can, another person is planting a small tree, and a third person is holding a thermometer. A heart shape is also visible in the background.

Sign In

Username:

Password:

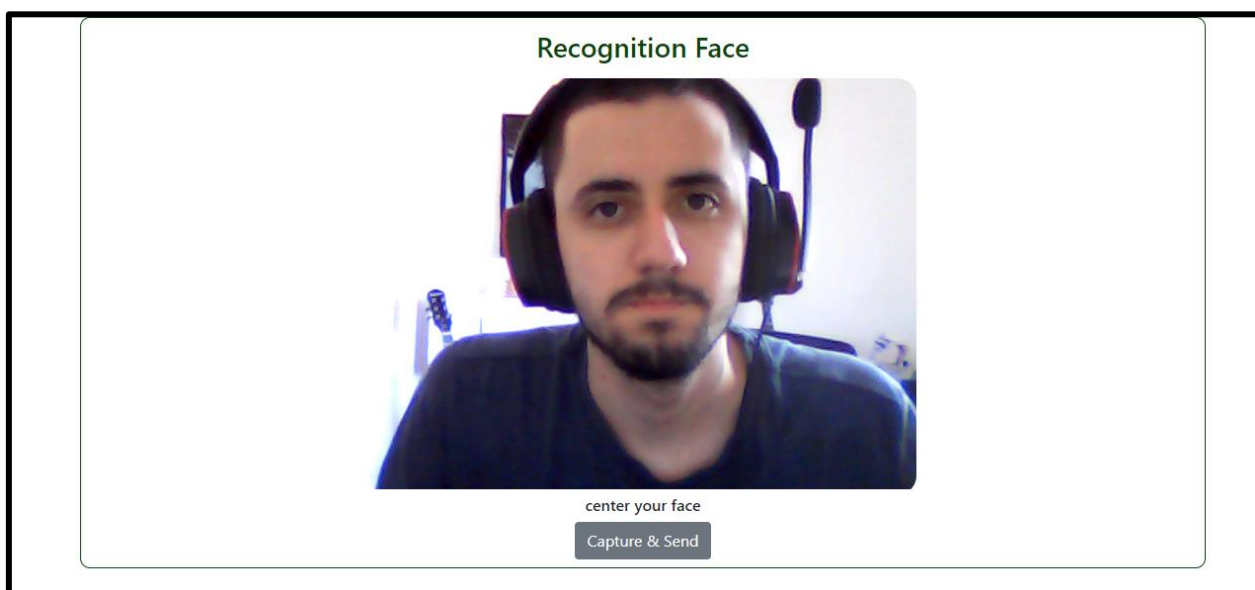
Remember me ☒ [Forgot Password](#)

[Log In](#)

Don't have an account? [Register](#)

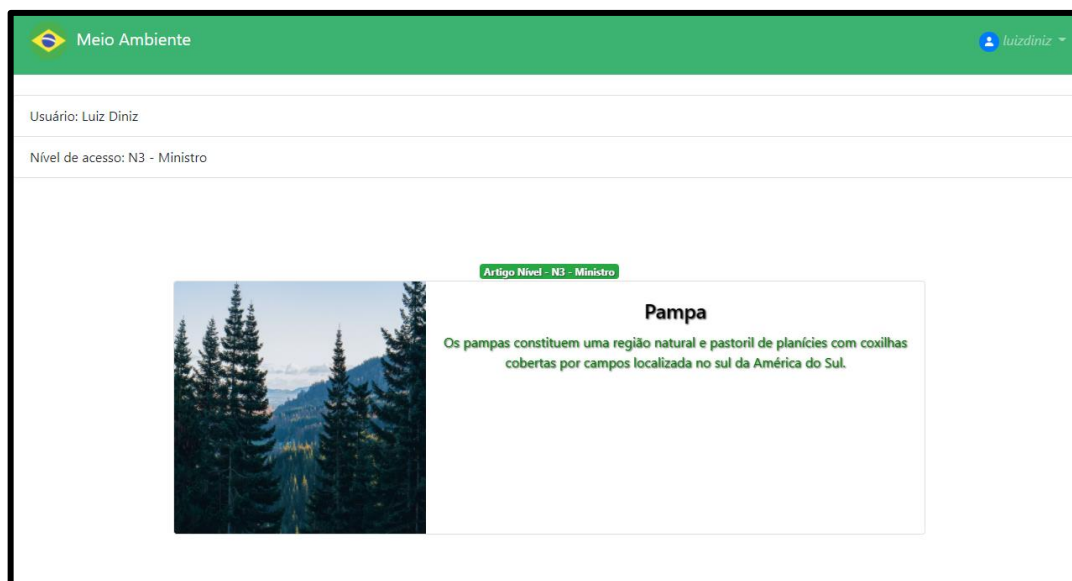
Fonte: Própria, 2021

Figura 8 – Reconhecimento facil.



Fonte: Própria, 2021

Figura 9 – Informações que o ministro acessa.



Fonte: Própria, 2021

Exemplo de caso de erro na validação

Figura 10 – Tela de login.

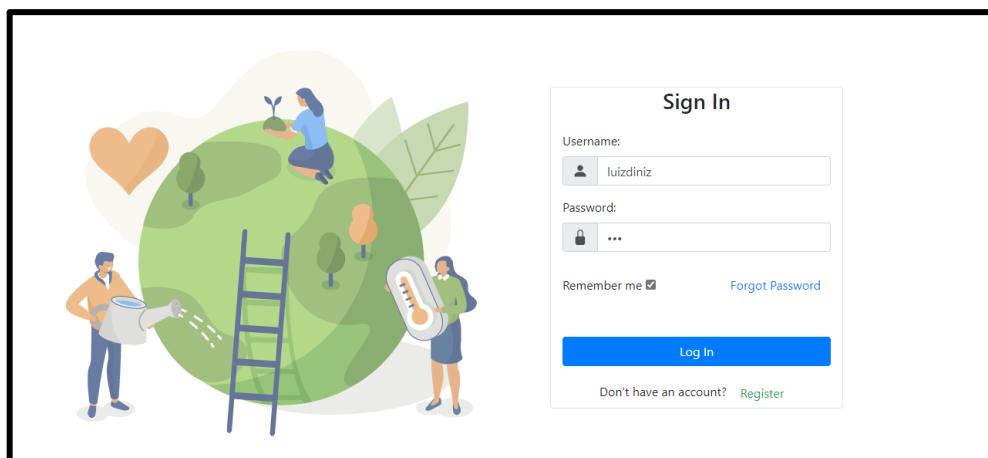
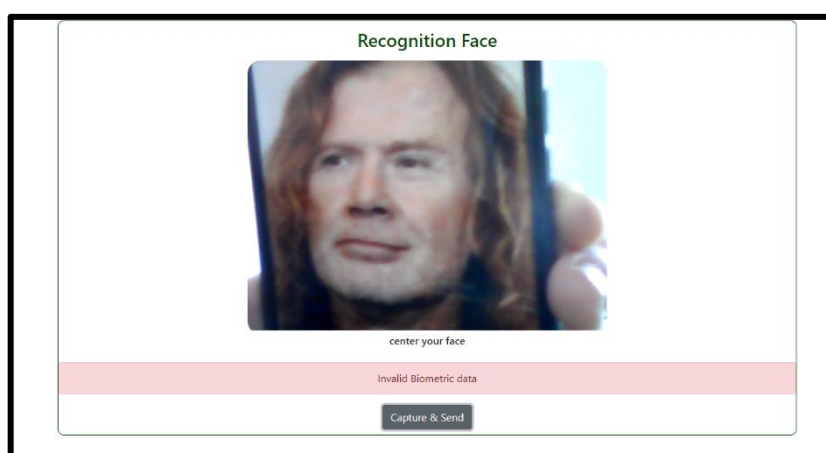


Figura 11 – Tela de reconhecimento com erro pois não reconheceu a face do usuário.



Fonte: Própria, 2021

6. CONSIDERAÇÕES FINAIS

Conhecida como um dos procedimentos mais estáveis de reconhecimento e está cada dia mais destacada na sociedade, a biometria é a pesquisa dos traços físicos e comportamentais de cada indivíduo. A origem fundamental deste mecanismo para identificação é o corpo humano, porque ele é a chave para acessar determinadas informações.

O desenvolvimento do presente estudo possibilitou o entendimento dos diversos tipos de biometria, com o foco no reconhecimento facial. Apesar de algumas dificuldades encontradas ao decorrer da pesquisa, explicamos o funcionamento das principais técnicas de processamento de imagens tendo seus prós e contras explorados, além de suas aplicações no mundo real.

Por tudo que foi exposto ao longo deste trabalho, foi desenvolvido um sistema onde o Ministério do Meio Ambiente poderá restringir os seus dados com base no cargo atribuído ao usuário. As ferramentas principais foram Visual Studio Community 2019 (desenvolvimento do aplicativo), o SQL Server (desenvolvimento do banco de dados) e o VUE (desenvolvimento do front) e Python (Cognition).

Após realizações de teste, pode-se concluir que os objetivos foram alcançados e a aplicação atendeu as expectativas gerando resultados satisfatórios, podendo ser aperfeiçoada.

O futuro do reconhecimento facial é promissor, por tanto, torna-se necessário o desenvolvimento de formas de aperfeiçoar os algoritmos de reconhecimento para que os próximos software possam ter uma qualidade cada vez maior, com ferramentas que facilite o desenvolvimento das próximas ferramentas de reconhecimento facial. Neste sentido, como foi demonstrado nesse trabalho, as técnicas de reconhecimento biométrico só têm a evoluir cada vez mais, mostrando os passos percorrido que esse projeto traçou para a criação desse aplicativo, que por sua vez permite o reconhecimento facial, e que um dia se espera que esse projeto ajude outras pessoas a criar seus próprios aplicativos. Em uma futura versão queremos implementar o método de autenticação JWT além de ser possível acessar por aparelhos móveis utilizando biometria digital.

7. REFERÊNCIAS

- Cavalcante, J. (25 de Maio de 2018). *Descomplicando SPA's*. Fonte: Medium: <https://medium.com/trainingcenter/descomplicando-spas-caa8f57bdbf3>
- Colabador Kaspersky. (2021). *O que é reconhecimento facial – definição e explicação*. Fonte: Kaspersky: <https://www.kaspersky.com.br/resource-center/definitions/what-is-facial-recognition>
- Control iD. (11 de Outubro de 2020). *O que é biometria e qual o futuro da identificação*. Acesso em 28 de Setembro de 2021, disponível em Control iD: <https://www.controlid.com.br/blog/biometria/o-que-e-biometria-futuro/>
- Cozer, C. (19 de Fevereiro de 2021). *O futuro dos sistemas de biometria pode estar nas nossas veias das mãos*. Acesso em 28 de Setembro de 2021, disponível em Whow: <https://www.whow.com.br/tecnologia/o-futuro-dos-sistemas-de-biometria-pode-estar-nas-nossas-veias-das-maos/#:~:text=No%20futuro%2C%20os%20cientistas%20planejam,informou%20em%20outubro%20de%202020>
- FIGUEIRAS, C., & GARROT, J. (2008). *Introdução ao Processamento Digital de Imagem*. FCA.
- Glaucio. (2006). *Introducao a serializacao de objetos*. Acesso em 10 de May de 2021, disponível em DevMedia: <https://www.devmedia.com.br/introducao-a-serializacao-de-objetos/3050>
- GOMES, J. M., & VELHO, L. (1997). *Image Processing for Computer Graphics*. Springer.
- Kinuta, C., Molina, D., Dorneles, E., Dias, G., Santana, J., & Junior, O. (s.d.). ESTUDO COMPARATIVO DE ALGORITMOS PARA RECONHECIMENTO FACIAL. São Caetano do Sul, São Paulo, Brasil. Fonte: ESTUDO COMPARATIVO DE ALGORITMOS PARA: https://www.aedb.br/seget/arquivos/artigos06/916_Copia%20de%20Artigo%20Comparativo%20Facial.pdf
- Kleina, N. (24 de Março de 2021). *Como funciona o reconhecimento facial*. Fonte: TecMundo: <https://www.tecmundo.com.br/camera-digital/10347-como-funcionam-os-sistemas-de-reconhecimento-facial.htm>
- Lima, G. (21 de Julho de 2021). *Bootstrap - O que é, como e quando usar?* Fonte: Alura: <https://www.alura.com.br/artigos/bootstrap>
- Mundo + Tech. (25 de Fevereiro de 2021). *As quatro eras do reconhecimento facial e seu efeito na privacidade*. Fonte: Mundo Mais Tech: <https://mundomaistech.com.br/inteligencia-artificial/as-quatro-eras-do-reconhecimento-facial-e-seu-efeito-na-privacidade/>
- Nascimento, W. M. (2020). *O que é o ASP.NET Core?* Fonte: Treina Web: <https://www.treinaweb.com.br/blog/o-que-e-o-asp-net-core>
- PEDRINI, H., & SCHWARTZ, W. R. (2008). *Análise de Imagens Digitais*. São Paulo: Thomson.
- VIEIRA NETO, H., & MARQUES FILHO, O. (1999). *Processamento Digital de Imagens*. Acadêmica - Brasport.

8. APÊNDICE

APÊNDICE A – LINHAS DE CÓDIGO

```
using FacialBiometrics.Models;
using FacialBiometricsBack.Models;
using FacialBiometricsBack.Services;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;

namespace FacialBiometricsBack.Controllers
{
    [Route("User")]
    [ApiController]
    public class UserController : Controller
    {
        private IFacialBiometricsServices _facialBiometricsService;

        public UserController(IFacialBiometricsServices
facialBiometricsServices)
        {
            _facialBiometricsService = facialBiometricsServices;
        }

        [HttpPost("register")]
        public JsonResult UserRegistration(UserFrontModel userData)
        {
            if (ModelState.IsValid)
            {
                if (userData.face_images.Count() == 0)
                {
                    return Json(new { message = "No user image received.",
statusCode = HttpStatusCode.BadRequest });
                }

                List<UserFaceImg> imageData = new List<UserFaceImg>();

                foreach (var img in userData.face_images)
                {
                    string[] imgData = img.Split(',');

                    imageData.Add(new UserFaceImg
                    {
                        metaData = imgData[0],
                        extension = imgData[0].Split(';')[0].Split('/')[1],
                        imageBytes = Convert.FromBase64String(imgData[1])
                    });
                }
            }
        }
    }
}
```



```

        });
    }

    int idUser = _facialBiometricsService.CreateUser(new UserInfo
    {
        NameUser = userData.name,
        Username = userData.username,
        Password = userData.password,
        UserPositionInfo = new UserPosition { IdUserPosition =
userData.id_user_position }
    });

    foreach (var faceImg in imageData)
    {
        _facialBiometricsService.CreateFacialBiometrics(new
UsersFacialBiometrics
        {
            ImageName = Guid.NewGuid().ToString(),
            ImageBytes = faceImg.imageBytes,
            IdUser = idUser
        });
    }

    return Json(new { message = "User registered successfully.",
statusCode = HttpStatusCode.OK });
}
else
{
    return Json(new { message = "Invalid registration.",
statusCode = HttpStatusCode.BadRequest });
}
}

[HttpPost("login")]
public JsonResult login(LoginModel userCredentials)
{
    var user =
_facialBiometricsService.Login(userCredentials.username,
userCredentials.password);

    if (user == null)
        return Json(new { isValid = false, message = "User Empty -
Invalid username or password.", statusCode = HttpStatusCode.Unauthorized });

    return Json(new { isValid = true, message = "Login first step
completed", user, statusCode = HttpStatusCode.OK });
}

[HttpPost("validate")]

```

```

public JsonResult validateLogin(UserImageInfo user)
{
    List<byte[]> imageData = new List<byte[]>();

    string[] imgDados = user.image[0].Split(',');
    imageData.Add(Convert.FromBase64String(imgDados[1]));

    bool resultFaceImgs =
        _facialBiometricsService.CompareImages(user.idUser, imageData);

    if (resultFaceImgs)
    {
        return Json(new { isValid = true, statusCode =
            HttpStatusCode.OK });
    }
    else
    {
        return Json(new { isValid = false, message = "Invalid
            Biometric data.", statusCode = HttpStatusCode.Unauthorized });
    }
}

[HttpGet("articles")]
public JsonResult getArticles(int idUser)
{
    Console.WriteLine(">getArticles: idUser(" + idUser + ")");

    List<ArticleModel> list_articles =
        _facialBiometricsService.GetArticles(idUser);

    return Json(new { listArticles = list_articles });
}

[HttpGet("levels")]
public JsonResult getUserByLevel(int idPosition)
{
    if (idPosition == 0)
        return Json(new { message = "User level not specified." });

    List<string> list_users =
        _facialBiometricsService.GetUsersByLevel(idPosition);

    return Json(new { message = "Job Leveling Successfully!",
        list_users = list_users });
}
}

```

```

using FacialBiometrics.Models;
using FacialBiometricsBack.Models;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;

namespace FacialBiometricsBack.DataAccessFacialBiometrics
{
    public class DataAccessFacialBiometrics : IDataAccessFacialBiometrics
    {
        private string connectionString = @"Data
Source=localhost\SQLEXPRESS;Initial
Catalog=RuralPropertiesInformations;Integrated Security=SSPI";

        public int CreateUser(UserInfo userInfo)
        {
            string query = "INSERT INTO UserInfo VALUES (@P0, @P1, @P2, @P3,
@P4); SELECT SCOPE_IDENTITY()";

            using (SqlConnection conn = new SqlConnection(connectionString))
            {
                conn.Open();

                using (SqlCommand cmd = new SqlCommand(query, conn))
                {
                    cmd.CommandType = CommandType.Text;
                    cmd.Parameters.Add(new SqlParameter("P0",
userInfo.NameUser));
                    cmd.Parameters.Add(new SqlParameter("P1",
userInfo.Username));
                    cmd.Parameters.Add(new SqlParameter("P2",
userInfo.Password));
                    cmd.Parameters.Add(new SqlParameter("P3",
userInfo.Password));
                    cmd.Parameters.Add(new SqlParameter("P4",
userInfo.UserPositionInfo.IdUserPosition));

                    return Convert.ToInt32(cmd.ExecuteScalar());
                }
            }
        }

        public void CreateFacialBiometrics(UsersFacialBiometrics userImages)
        {
            string query = "INSERT INTO UsersFacialBiometrics VALUES (@P0,
@P1, @P2)";

            using (SqlConnection conn = new SqlConnection(connectionString))

```

```

        {
            conn.Open();

            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                cmd.CommandType = CommandType.Text;
                cmd.Parameters.Add(new SqlParameter("P0",
userImages.ImageName));
                cmd.Parameters.Add(new SqlParameter("P1",
userImages.ImageBytes));
                cmd.Parameters.Add(new SqlParameter("P2",
userImages.IdUser));
                cmd.ExecuteNonQuery();
            }
        }
    }

    public int GetUserPosition(UserInfo userInfo)
    {
        string query = "SELECT id_user_position FROM UserInfo WHERE
id_user = @P0";

        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            conn.Open();

            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                cmd.CommandType = CommandType.Text;
                cmd.Parameters.Add(new SqlParameter("P0",
userInfo.IdUser));

                SqlDataReader reader = cmd.ExecuteReader();

                var idUser = 0;

                if (reader.Read())
                {
                    idUser = Convert.ToInt32(reader["id_user"]);
                }

                return idUser;
            }
        }
    }

    public List<ArticleModel> GetArticles(int idUser)
    {
        string query = @"SELECT * FROM ArticleChemicalProduct A

```

```

        WHERE A.id_user_position <= (select U.id_user_position
FROM UserInfo U WHERE U.id_user= @P0)";

        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            conn.Open();

            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                cmd.CommandType = CommandType.Text;
                cmd.Parameters.Add(new SqlParameter("P0", idUser));

                SqlDataReader reader = cmd.ExecuteReader();

                List<ArticleModel> listArticles = new
List<ArticleModel>();

                while (reader.Read())
                {
                    listArticles.Add(new ArticleModel
                    {
                        idArticle = Convert.ToInt32(reader["id_article"]),
                        title = Convert.ToString(reader["article_title"]),
                        content =
Convert.ToString(reader["article_content"])
                    });
                }

                return listArticles;
            }
        }
    }

    public UserInfo GetUserByUsername(string userName)
    {
        string query = "SELECT * FROM UserInfo WHERE username_user = @P0";

        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            conn.Open();

            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                cmd.CommandType = CommandType.Text;
                cmd.Parameters.Add(new SqlParameter("P0", userName));

                SqlDataReader reader = cmd.ExecuteReader();

                var user = new UserInfo();

```

```

        user.UserPositionInfo = new UserPosition();

        if (reader.Read())
        {
            user.IdUser = Convert.ToInt32(reader["id_user"]);
            user.NameUser = Convert.ToString(reader["name_user"]);
            user.Username =
Convert.ToString(reader["username_user"]);
            user.Password =
Convert.ToString(reader["password_user"]);
            user.SaltPassword =
Convert.ToString(reader["salt_password_user"]);
            user.UserPositionInfo.IdUserPosition =
Convert.ToInt32(reader["id_user_position"]);
        }

        return user;
    }
}

public List<string> GetUserByLevel(int idPosition)
{
    string query = @"SELECT name_user FROM UserInfo WHERE
id_user_position = @P0";

    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        conn.Open();

        using (SqlCommand cmd = new SqlCommand(query, conn))
        {
            cmd.CommandType = CommandType.Text;
            cmd.Parameters.Add(new SqlParameter("P0", idPosition));

            SqlDataReader reader = cmd.ExecuteReader();

            List<string> listUsers = new List<string>();

            while (reader.Read())
            {
                listUsers.Add(Convert.ToString(reader["name_user"]));
            }

            return listUsers;
        }
    }
}

```

```

        public List<UsersFacialBiometrics> GetFacialBiometric(int idUser)
        {
            string query = @"SELECT * FROM UsersFacialBiometrics fb WHERE
fb.id_user = @P0";

            using (SqlConnection conn = new SqlConnection(connectionString))
            {
                conn.Open();

                using (SqlCommand cmd = new SqlCommand(query, conn))
                {
                    cmd.CommandType = CommandType.Text;
                    cmd.Parameters.Add(new SqlParameter("P0", idUser));

                    SqlDataReader reader = cmd.ExecuteReader();

                    List<UsersFacialBiometrics> imgs = new
List<UsersFacialBiometrics>();

                    while (reader.Read())
                    {
                        imgs.Add(new UsersFacialBiometrics
                        {
                            IdImg = Convert.ToInt32(reader["id_img"]),
                            ImageName = Convert.ToString(reader["name_img"]),
                            ImageBytes = (Byte[])reader["user_img"],
                            IdUser = Convert.ToInt32(reader["id_user"])
                        });
                    }

                    return imgs;
                }
            }
        }
    }
}

```

```

using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace FacialBiometricsBack.Models
{
    public class UserFrontModel
    {
        [Required]
        public string name { get; set; }
    }
}

```

```

        [Required]
        public string username { get; set; }

        [Required]
        public string password { get; set; }

        [Required]
        public List<string> face_images { get; set; }

        [Required]
        public int id_user_position { get; set; }
    }
}

```

LoginModel.cs

```

using System.Collections.Generic;

namespace FacialBiometrics.Models
{
    public class LoginModel
    {
        public string username { get; set; }
        public string password { get; set; }
        public List<string> face_images { get; set; }
    }
}

```

UserImageInfo.cs

```

namespace FacialBiometricsBack.Models
{
    public class UserImageInfo
    {
        public int idUser { get; set; }
        public string[] image { get; set; }
    }
}

```

FacialBiometricsServices.cs

```

using FacialBiometrics.Models;

```



```

using FacialBiometricsBack.DataAccessFacialBiometrics;
using FacialBiometricsBack.Models;
using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;

namespace FacialBiometricsBack.Services
{
    public class FacialBiometricsServices : IFacialBiometricsServices
    {
        private IDataAccessFacialBiometrics _dataAccess;

        public FacialBiometricsServices(IDataAccessFacialBiometrics
dataAccess)
        {
            _dataAccess = dataAccess;
        }

        public int CreateUser(UserInfo userInfo)
        {
            try
            {
                if (userInfo == null) throw new
ArgumentNullException(nameof(userInfo));

                return _dataAccess.CreateUser(userInfo);
            }
            catch
            {
                throw;
            }
        }

        public void CreateFacialBiometrics(UsersFacialBiometrics userImages)
        {
            try
            {
                if (userImages == null) throw new
ArgumentNullException(nameof(userImages));

                _dataAccess.CreateFacialBiometrics(userImages);
            }
            catch
            {
                throw;
            }
        }
    }
}

```

```

public int GetUserPosition(UserInfo userInfo)
{
    try
    {
        if (userInfo == null) throw new
ArgumentNullException(nameof(userInfo));

        return _dataAccess.GetUserPosition(userInfo);
    }
    catch
    {
        throw;
    }
}

public List<ArticleModel> GetArticles(int idUser)
{
    try
    {
        return _dataAccess.GetArticles(idUser);
    }
    catch
    {
        throw;
    }
}

public UserInfo Login(string userName, string password)
{
    try
    {
        if (String.IsNullOrEmpty(userName) ||
String.IsNullOrEmpty(password))
            throw new ArgumentException("Username or password
null.");

        var result = _dataAccess.GetUserByUsername(userName);

        if (result == null)
            return null;

        if (result.Password != password)
            return null;

        return result;
    }
    catch
    {
        throw;
    }
}

```

```

    }
}

private string CreateSalt(int size)
{
    var rng = new RNGCryptoServiceProvider();

    var buff = new byte[size];

    rng.GetBytes(buff);

    return Convert.ToBase64String(buff);
}

private string GenerateHash(string password, string salt)
{
    byte[] bytes = Encoding.UTF8.GetBytes(password + salt);

    SHA256Managed sha256 = new SHA256Managed();

    byte[] hash = sha256.ComputeHash(bytes);

    return Convert.ToBase64String(hash);
}

public List<string> GetUsersByLevel(int idPosition)
{
    return _dataAccess.GetUserByLevel(idPosition);
}

public bool CompareImages(int idUser, List<byte[]> receivedImages)
{
    List<UsersFacialBiometrics> imgsDb =
_dataAccess.GetFacialBiometric(idUser);

    var imageComparerService = new ImageComparerService();

    bool result = imageComparerService.CompareImages(imgsDb,
receivedImages);

    return result;
}
}
}

```

```

using FacialBiometrics.Models;
using System;
using System.Collections.Generic;

```

```

using System.Diagnostics;
using System.IO;

namespace FacialBiometricsBack.Services
{
    public class ImageComparerService
    {
        private string TempPath =
Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "TempUploads\\");

        public bool CompareImages(List<UsersFacialBiometrics> imgsDb,
List<byte[]> imgsRecebidas)
        {
            try
            {
                if (imgsDb == null || imgsRecebidas == null) throw new
ArgumentNullException();

                var pathUploadDatabase = TempPath + "Database\\";
                var pathUploadReceived = TempPath + "Received\\";

                if (!Directory.Exists(pathUploadDatabase))
                    Directory.CreateDirectory(pathUploadDatabase);

                if (!Directory.Exists(pathUploadReceived))
                    Directory.CreateDirectory(pathUploadReceived);

                pathUploadReceived = pathUploadReceived +
"received_image.png";
                pathUploadDatabase = pathUploadDatabase +
"database_image.png";

                CreateTempImage(pathUploadReceived, imgsRecebidas[0],
imgsRecebidas[0].Length);
                CreateTempImage(pathUploadDatabase, imgsDb[0].ImageBytes,
imgsDb[0].ImageBytes.Length);

                var result = RunPythonScript();

                DeleteTempPath(TempPath);

                if (result.Contains("true"))
                    return true;
                else if (result.Contains("false"))
                    return false;
                else
                    throw new Exception("Error executing the python face
recognition engine.");
            }
        }
    }
}

```

```

        catch (Exception e)
        {
            throw new Exception("Error CompareImages: " + e.Message);
        }
    }

    private string RunPythonScript()
    {
        Process process = new Process();

        process.StartInfo.FileName = "cmd.exe";
        process.StartInfo.CreateNoWindow = true;
        process.StartInfo.RedirectStandardInput = true;
        process.StartInfo.RedirectStandardOutput = true;
        process.StartInfo.UseShellExecute = false;

        process.Start();

        process.StandardInput.WriteLine("cd
C:\\Users\\Luiz\\source\\repos\\FacialBiometrics\\myenvpy\\Scripts\\");
        process.StandardInput.WriteLine("activate");
        process.StandardInput.WriteLine("python
execute_face_recognition.py");
        process.StandardInput.Flush();
        process.StandardInput.Close();

        return process.StandardOutput.ReadToEnd();
    }

    private void CreateTempImage(string path, byte[] image, int
imageLength)
    {
        if (!File.Exists(path))
        {
            using (var fs = new FileStream(path, FileMode.Create,
FileAccess.Write))
            {
                fs.Write(image, 0, imageLength);
            }
        }
    }

    private void DeleteTempPath(string path)
    {
        var directories = Directory.GetDirectories(path);

        foreach (var directory in directories)
        {
            DeleteTempPath(directory);
        }
    }

```

```
    }

    var files = Directory.GetFiles(path);

    foreach (var file in files)
    {
        File.Delete(file);
    }

    Directory.Delete(path);
}
}
```

APÊNDICE B – QR CODE

Para acessar o código da aplicação inteiro, acesse pelo link ou QR Code abaixo.



Fonte: <https://github.com/rafass04/FacialBiometrics>



UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: ALEXANDRE SANTOS CAVALCANTE

TURMA: CC6P68

RA: N384425

CURSO: Ciência da Computação

CAMPUS: Paraíso

SEMESTRE: 6

TURNIO: Noturno

CÓDIGO DA ATIVIDADE: 77B3

SEMESTRE: 6

ANO GRADE: 2/2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
21/09/2021	Reunião com Grupo	5,00	Alexandre Santos Cavalcante		
22/09/2021	Criação de repositório no GitHub e início de pesquisas	4,00	Alexandre Santos Cavalcante		
24/09/2021	Parte teórica: Biometria Facial no Situação Problema	5,00	Alexandre Santos Cavalcante		
26/09/2021	Pesquisa sobre as tecnologias a serem utilizadas	4,00	Alexandre Santos Cavalcante		
27/09/2021	Desenvolvimento do Banco e Diagramas	6,00	Alexandre Santos Cavalcante		
29/09/2021	Desenvolvimento aplicação	5,00	Alexandre Santos Cavalcante		
04/10/2021	Parte teórica: Funcionamento da Biometria Facial	4,00	Alexandre Santos Cavalcante		
05/10/2021	Pesquisas sobre EMGU e testes	4,00	Alexandre Santos Cavalcante		
10/10/2021	Reunião com Grupo e fluxograma da Aplicação	6,00	Alexandre Santos Cavalcante		
20/10/2021	Desenvolvimento API para controle de usuário	4,00	Alexandre Santos Cavalcante		
22/10/2021	Inclusão dos novos métodos de banco e serviço	6,00	Alexandre Santos Cavalcante		
23/10/2021	Inserção de dados no Banco e reunião de grupo	4,00	Alexandre Santos Cavalcante		
24/10/2021	Atualização da documentação, criação de método p/ validar login	6,00	Alexandre Santos Cavalcante		
26/10/2021	Inserts no banco, organização documento, desenvolvimento API	3,00	Alexandre Santos Cavalcante		
10/11/2021	Testes na aplicação, reunião com o grupo	8,00	Alexandre Santos Cavalcante		
14/11/2021	Revisão geral do programa e parte teórico em grupo e envio	6,00	Alexandre Santos Cavalcante		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 80

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Heloisa Ferreira da Silva

TURMA: CC6P68

RA: F037CJ1

CURSO: Ciência da Computação

CAMPUS: Paraíso

SEMESTRE: 5/6

TURNO: Noturno

CÓDIGO DA ATIVIDADE: 77B3

SEMESTRE: 5/6

ANO GRADE: 2/2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
21/09/2021	Reunião com Grupo	5,00	Heloisa Ferreira		
22/09/2021	Criação de repositório no GitHub e início de pesquisas	4,00	Heloisa Ferreira		
24/09/2021	Parte teórica: Biometria Facial no Situação Problema	5,00	Heloisa Ferreira		
26/09/2021	Pesquisa sobre as tecnologias a serem utilizadas	4,00	Heloisa Ferreira		
27/09/2021	Desenvolvimento do Banco e Diagramas	6,00	Heloisa Ferreira		
29/09/2021	Desenvolvimento aplicação	5,00	Heloisa Ferreira		
04/10/2021	Parte teórica: Funcionamento da Biometria Facial	4,00	Heloisa Ferreira		
05/10/2021	Pesquisas sobre EMGU e testes	4,00	Heloisa Ferreira		
10/10/2021	Reunião com Grupo e fluxograma da Aplicação	6,00	Heloisa Ferreira		
20/10/2021	Desenvolvimento API para controle de usuário	4,00	Heloisa Ferreira		
22/10/2021	Inclusão dos novos métodos de banco e serviço	6,00	Heloisa Ferreira		
23/10/2021	Inserção de dados no Banco e reunião de grupo	4,00	Heloisa Ferreira		
24/10/2021	Atualização da documentação, criação de método p/ validar login	6,00	Heloisa Ferreira		
26/10/2021	Inserts no banco, organização documento, desenvolvimento API	3,00	Heloisa Ferreira		
10/11/2021	Testes na aplicação, reunião com o grupo	8,00	Heloisa Ferreira		
14/11/2021	Revisão geral do programa e parte teórico em grupo e envio	6,00	Heloisa Ferreira		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 80

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: OSCAR LUIZ R DE OLIVEIRA

TURMA: CC6P68

RA: D85HEF7

CURSO: Ciência da Computação

CAMPUS: Paraíso

SEMESTRE: 5/6

TURNIO: Noturno

CÓDIGO DA ATIVIDADE: 77B3

SEMESTRE: 5/6

ANO GRADE: 2/2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
21/09/2021	Reunião com Grupo	5,00	Oscar Oliveira		
22/09/2021	Criação de repositório no GitHub e início de pesquisas	4,00	Oscar Oliveira		
24/09/2021	Parte teórica: Biometria Facial no Situação Problema	5,00	Oscar Oliveira		
26/09/2021	Pesquisa sobre as tecnologias a serem utilizadas	4,00	Oscar Oliveira		
27/09/2021	Desenvolvimento do Banco e Diagramas	6,00	Oscar Oliveira		
29/09/2021	Desenvolvimento aplicação	5,00	Oscar Oliveira		
04/10/2021	Parte teórica: Funcionamento da Biometria Facial	4,00	Oscar Oliveira		
05/10/2021	Pesquisas sobre EMGU e testes	4,00	Oscar Oliveira		
10/10/2021	Reunião com Grupo e fluxograma da Aplicação	6,00	Oscar Oliveira		
20/10/2021	Desenvolvimento API para controle de usuário	4,00	Oscar Oliveira		
22/10/2021	Inclusão dos novos métodos de banco e serviço	6,00	Oscar Oliveira		
23/10/2021	Inserção de dados no Banco e reunião de grupo	4,00	Oscar Oliveira		
24/10/2021	Atualização da documentação, criação de método p/ validar login	6,00	Oscar Oliveira		
26/10/2021	Inserts no banco, organização documento, desenvolvimento API	3,00	Oscar Oliveira		
10/11/2021	Testes na aplicação, reunião com o grupo	8,00	Oscar Oliveira		
14/11/2021	Revisão geral do programa e parte teórico em grupo e envio	6,00	Oscar Oliveira		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 80

AValiação: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS
NOME: Luiz Gustavo De Oliveira Diniz

TURMA: CCSP68

RA: N537BD3

CURSO: Ciência da Computação

CAMPUS: Paraíso

SEMESTRE: 5/6

TURNO: Noturno

CÓDIGO DA ATIVIDADE: 77B3

SEMESTRE: 5/6

ANO GRADE: 2/2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
21/09/2021	Reunião com Grupo	5,00	Luiz Diniz		
22/09/2021	Criação de repositório no GitHub e início de pesquisas	4,00	Luiz Diniz		
24/09/2021	Parte teórica: Biometria Facial no Situação Problema	5,00	Luiz Diniz		
26/09/2021	Pesquisa sobre as tecnologias a serem utilizadas	4,00	Luiz Diniz		
27/09/2021	Desenvolvimento do Banco e Diagramas	6,00	Luiz Diniz		
29/09/2021	Desenvolvimento aplicação	5,00	Luiz Diniz		
04/10/2021	Parte teórica: Funcionamento da Biometria Facial	4,00	Luiz Diniz		
05/10/2021	Pesquisas sobre EMGU e testes	4,00	Luiz Diniz		
10/10/2021	Reunião com Grupo e fluxograma da Aplicação	6,00	Luiz Diniz		
20/10/2021	Desenvolvimento API para controle de usuário	4,00	Luiz Diniz		
22/10/2021	Inclusão dos novos métodos de banco e serviço	6,00	Luiz Diniz		
23/10/2021	Inserção de dados no Banco e reunião de grupo	4,00	Luiz Diniz		
24/10/2021	Atualização da documentação, criação de método p/ validar login	6,00	Luiz Diniz		
26/10/2021	Inserts no banco, organização documento, desenvolvimento API	3,00	Luiz Diniz		
10/11/2021	Testes na aplicação, reunião com o grupo	8,00	Luiz Diniz		
14/11/2021	Revisão geral do programa e parte teórico em grupo e envio	6,00	Luiz Diniz		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 80

AValiação: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Augusto Calisto de Aquino

TURMA: CC6P68

RA: D898HI-8

CURSO: Ciência da Computação

CAMPUS: Paraíso

SEMESTRE: 6º

TURNO: Noturno

CÓDIGO DA ATIVIDADE: 77B3

SEMESTRE: 6º

ANO GRADE: 2/2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
21/09/2021	Reunião com Grupo	5,00	Augusto Calisto		
22/09/2021	Criação de repositório no GitHub e início de pesquisas	4,00	Augusto Calisto		
24/09/2021	Parte teórica: Biometria Facial no Situação Problema	5,00	Augusto Calisto		
26/09/2021	Pesquisa sobre as tecnologias a serem utilizadas	4,00	Augusto Calisto		
27/09/2021	Desenvolvimento do Banco e Diagramas	6,00	Augusto Calisto		
29/09/2021	Desenvolvimento aplicação	5,00	Augusto Calisto		
04/10/2021	Parte teórica: Funcionamento da Biometria Facial	4,00	Augusto Calisto		
05/10/2021	Pesquisas sobre EMGU e testes	4,00	Augusto Calisto		
10/10/2021	Reunião com Grupo e fluxograma da Aplicação	6,00	Augusto Calisto		
20/10/2021	Desenvolvimento API para controle de usuário	4,00	Augusto Calisto		
22/10/2021	Inclusão dos novos métodos de banco e serviço	6,00	Augusto Calisto		
23/10/2021	Inserção de dados no Banco e reunião de grupo	4,00	Augusto Calisto		
24/10/2021	Atualização da documentação, criação de método p/ validar login	6,00	Augusto Calisto		
26/10/2021	Inserts no banco, organização documento, desenvolvimento API	3,00	Augusto Calisto		
10/11/2021	Testes na aplicação, reunião com o grupo	8,00	Augusto Calisto		
14/11/2021	Revisão geral do programa e parte teórico em grupo e envio	6,00	Augusto Calisto		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 80 horas

AValiação: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Rafaela Dos Santos Silva

TURMA: CC6P68

RA: D9219D0

CURSO: Ciência da Computação

CAMPUS: Paraíso

SEMESTRE: 5/6

TURNO: Noturno

CÓDIGO DA ATIVIDADE: 77B3

SEMESTRE: 5/6

ANO GRADE: 2/2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
21/09/2021	Reunião com Grupo	5,00	Rafaela Silva		
22/09/2021	Criação de repositório no GitHub e início de pesquisas	4,00	Rafaela Silva		
24/09/2021	Parte teórica: Biometria Facial no Situação Problema	5,00	Rafaela Silva		
26/09/2021	Pesquisa sobre as tecnologias a serem utilizadas	4,00	Rafaela Silva		
27/09/2021	Desenvolvimento do Banco e Diagramas	6,00	Rafaela Silva		
29/09/2021	Desenvolvimento aplicação	5,00	Rafaela Silva		
04/10/2021	Parte teórica: Funcionamento da Biometria Facial	4,00	Rafaela Silva		
05/10/2021	Pesquisas sobre EMGU e testes	4,00	Rafaela Silva		
10/10/2021	Reunião com Grupo e fluxograma da Aplicação	6,00	Rafaela Silva		
20/10/2021	Desenvolvimento API para controle de usuário	4,00	Rafaela Silva		
22/10/2021	Inclusão dos novos métodos de banco e serviço	6,00	Rafaela Silva		
23/10/2021	Inserção de dados no Banco e reunião de grupo	4,00	Rafaela Silva		
24/10/2021	Atualização da documentação, criação de método p/ validar login	6,00	Rafaela Silva		
26/10/2021	Inserts no banco, organização documento, desenvolvimento API	3,00	Rafaela Silva		
10/11/2021	Testes na aplicação, reunião com o grupo	8,00	Rafaela Silva		
14/11/2021	Revisão geral do programa e parte teórico em grupo e envio	6,00	Rafaela Silva		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 80

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO